Maximizing Submodular Functions under Submodular Constraints

Madhavan R. Padmanabhan[1] **Yanhui Zhu**[1] **Samik Basu**[1] **A. Pavan**[1]

[1]Department of Computer Science,
Iowa State University,
Ames, Iowa, USA

Abstract

We consider the problem of maximizing submodular functions under submodular constraints by formulating the problem in two ways: SCSK-C and DIFF-C. Given two submodular functions f and g where f is monotone, the objective of SCSK-C problem is to find a set S of size at most k that maximizes $f(S)$ under the constraint that $g(S) \leq \theta$, for a given value of θ. The problem of DIFF-C focuses on finding a set S of size at most k such that $h(S) = f(S) - g(S)$ is maximized. It is known that these problems are highly inapproximable and do not admit any constant factor multiplicative approximation algorithms unless NP is easy. Known approximation algorithms involve data-dependent approximation factors that are not efficiently computable.

We initiate a study of the design of approximation algorithms where the approximation factors are efficiently computable. For the problem of SCSK-C, we prove that the greedy algorithm produces a solution whose value is at least $(1-1/e)f(\mathrm{OPT}) - A$, where A is the data-dependent additive error. For the DIFF-C problem, we design an algorithm that uses the SCSK-C greedy algorithm as a subroutine. This algorithm produces a solution whose value is at least $(1-1/e)h(\mathrm{OPT}) - B$, where B is also a data-dependent additive error. A salient feature of our approach is that the additive error terms can be computed efficiently, thus enabling us to ascertain the quality of the solutions produced.

1 INTRODUCTION

For a ground set V of size n, a function $f : 2^V \to \mathbb{R}$ is *submodular* if for every $S \subseteq T \subseteq V$, and for every $x \in V - T$, $f(S \cup \{x\}) - f(S) \geq f(T \cup \{x\}) - f(T)$. I.e., the gain in the value of the function when x is added to S is at least the gain when x is added to a superset of S. Optimizing submodular functions under various constraints has been studied extensively. These problems are of the following form: For a submodular function f, *find a set $S \subseteq V$ that maximizes $f(S)$ subject to the constraint that $S \in \mathcal{F}$, where $\mathcal{F}$ is a family of sets*. A few of the well-studied constraints are *cardinality constraint*, *knapsack/modular constraint*, and *matroid constraints*. Even for the least restrictive constraint, cardinality constraint, the problem is known to be NP-hard. The classical work of Nemhauser *et al.* showed that a greedy algorithm achieves a $(1-1/e)$ approximation ratio if the submodular function f is monotone Nemhauser et al. [1978a].

Submodular Constraints. Often, in submodular maximization problems, there is a conflicting minimization constraint. The generic nature of these problems is of the following form: Given a submodular function f, another function g, find a set S of size at most k that *maximizes $f(S)$, while minimizing $g(S)$*. In this work, we study the case where the function *g is also a submodular function*. The problem of maximizing a submodular function under a submodular constraint appears in a few application domains. The works of Iyer and Bilmes [2012b, 2013] discuss several scenarios where these problems arise naturally. These application domains include sensor placement, speech data set selection, probabilistic inference, and information diffusion Kempe et al. [2003], Lin and Bilmes [2009, 2011], Krause et al. [2008], Jegelka and Bilmes [2011].

SCSK-C and DIFF-C. Two of the standard ways to formalize the above-mentioned maximization-minimization problem is via introducing a submodular constraint Iyer and Bilmes [2013], Crawford et al. [2019], Wan et al. [2010] and as maximizing the difference between submodular functions Iyer and Bilmes [2012b], Narasimhan and Bilmes [2005], Jin et al. [2021a], Kawahara and Washio [2011]. The *Submodular Cost Submodular Knapsack (SCSK)* is as follows: given two non-negative, submodular functions f and g over a ground set such that f is monotone and a value θ, the goal is to find a set S that maximizes $f(S)$ subject

Accepted for the 39th Conference on Uncertainty in Artificial Intelligence (UAI 2023).

to the constraint that $g(S) \leq \theta$. The Diff problem is the following: Given non-negative, submodular functions f and g where f is monotone, find a set S such that $f(S) - g(S)$ is maximized. In this paper, we will consider generalizations of these problems called SCSK-C and DIFF-C. The SCSK-C problem is to find set S that maximizes f subject to the constraint $g(S) \leq \theta$ and $|S| \leq k$. Similarly, DIFF-C problem is to find a set S that maximizes the function $f - g$ such that $|S| \leq k$. In SCSK-C setting, we will refer to $g(X) \leq \theta$ as submodularity budget constraint and θ as the submodular budget. In this work, we address the problem of obtaining approximation algorithms for these problems whose *approximation factors can be efficiently computed*.

Data Dependent Approximations. Unfortunately, for both SCSK and DIFF (and hence for both SCSK-C and DIFF-C) it is known that efficient approximation algorithms are not feasible. From the work of Iyer and Bilmes Iyer and Bilmes [2013] it follows that if P does not equal NP, then SCSK does not admit even admit $1/n^{1/2-\epsilon}$-multiplicative approximation algorithms, for any $\epsilon > 0$. Narasimhan and Bilmes Narasimhan and Bilmes [2005] showed that every set function can be represented as the difference between two submodular functions, and thus DIFF is inapproximable Narasimhan and Bilmes [2005], Iyer and Bilmes [2012b]. Given the high inapproximability of these problems, it seems that there is no hope of theoretically analyzing the quality of the solutions produced by efficient algorithms for these problems. However, it turns out that *data-dependent approximation* guarantees can be obtained for the SCSK-C problem.

A d-multiplicative approximation for submodular maximization produces a solution S such that $f(S) \geq d \times f(OPT)$, where OPT is the optimal solution. Typically the value d is independent of the actual function f that is being maximized. This is either a universal constant (such as $1/2$ or $(1 - 1/e)$) or depends on the size of the ground set V. On the contrary, algorithms with data-dependent approximation guarantees have the following flavor: For a given function f, the value of the solution produced by the algorithm is at least $d_f \times OPT$, where d_f depends on the function f that we seek to maximize and the constraints.

For the problem of SCSK-C, Iyer and Bilmes [2013], Iyer et al. [2013] presented data-dependent approximation algorithms, However, we observe that this data-dependent approximation factor is NP-hard to compute. Given this, it is hard to judge the quality of the solution produced by the approximation algorithm (i.e., how close it is to the optimal solution). We discuss this issue in detail in Section 2. For the problem of DIFF-C, to the best of our knowledge, no data-dependent approximation bounds have been established. The works of Iyer and Bilmes [2012b], Narasimhan and Bilmes [2005] provide a heuristic approach to solve the problems of maximizing the difference between two submodular functions f and g by replacing g with a surrogate modular function g' that bounds g and instead maximize $f - g'$, which is submodular.

Our thesis is that *data-dependent approximation factors are more meaningful only when they can be computed efficiently*. Motivated by this, we explore the possibility of designing algorithms with *efficiently computable* data-dependent approximation factors.

Our Contributions. To achieve efficiently computable approximation factors, we study the notion of *multiplicative-additive error approximation* algorithms. We say that $\mathcal{A}$ is a (d, A)-multiplicative-additive approximation algorithm for the problem maximization problem if the $output\ of\ \mathcal{A} \geq d \times f(OPT) - A$. We refer to d as *multiplicative factor* and A as *additive error*.

Algorithms with Efficiently Computable Approximation Factors for SCSK-C. We first consider the standard greedy algorithm for SCSK-C. We establish a new guarantee on the quality of the solution produced by the solution. Namely, we prove that if S is the solution produced, then $f(S) \geq (1 - 1/e)f(OPT) - A_{fg}$. Here the additive error A_{fg} is the data-dependent factor that depends on f and g. A hallmark of our proof and analysis is that A_{fg} can be computed while running the greedy algorithm with very little overhead, thus making the computation of A_{fg} efficient. Combining this proof with ideas from Conforti and Cornuéjols [1984], we refine the multiplicative error to $\frac{1}{c_f}(1 - (1 - \frac{c_f}{k})^k)$, while keeping the additive error same. We remark that while our proofs start with the standard arguments known in the literature, there are critical departure points. The main contribution in the proofs is conceptual rather than technical, which enables us to obtain the desired bounds.

DIFF-C *via* SCSK-C. We first observe that the DIFF-C problem reduces to SCSK-C problem when the range of g is non-negative integers. Building on this, we design an approximation algorithm for DIFF-C that uses the natural greedy algorithm for SCSK-C as a subroutine. Building upon our theoretical analysis of the greedy algorithm for SCSK-C, we analyze the quality of the solution S produced for the DIFF-C problem and show that $f(S) - g(S) \geq (1 - 1/e)[f(OPT) - g(OPT)] - A_{fg}$, where A_{fg} is the efficiently computable additive error.

Experimental Validation: As proof of concept, we have conducted experiments in the domains of information diffusion. For both problems, these experiments reveal that in practice the additive error is small, thus indicating that our proposed algorithms produce a solution whose value is close to $(1 - 1/e)$ of the optimal solution.

Prior and Related Work. For submodular maximization with knapsack/modular constraint, Sviridenko [2004] proposed a greedy algorithm with $1 - 1/e$ approximation ratio, albeit with time complexity of $O(n^5)$ oracle calls.

Later works improved the run-time Feldman et al. [2020], Yaroslavtsev et al. [2020], Li et al. [2022] with a small sacrifice in the approximation quality. One of the well-studied constraints is the matroid constraint for which Nemhauser et al. [1978b] provided a $1/2$-approximation algorithm. The breakthrough work of Calinescu et al. [2011] presented a randomized algorithm with the optimal approximation ratio to $1 - 1/e$. The work Buchbinder et al. [2019] proposed the first deterministic algorithm with an approximation ratio of 0.5008.

The work in Harshaw et al. [2019] studied maximizing $f - g$ under a cardinality constraint when f is submodular and g is modular, whereas Jin et al. [2021b] studied the problem without the cardinality constraint and provided a *multiplicative-additive* error approximation algorithm.

SCSC is a dual problem of SCSK studied in Iyer and Bilmes [2013], Crawford et al. [2019]. The problem involves minimizing a submodular function g while ensuring that another submodular function f is no less than a given threshold τ. There has been a vast amount of prior and related work on submodular optimization. We refer the reader to the survey articles Krause and Golovin [2014], Buchbinder and Feldman [2018].

2 HARDNESS OF APPROXIMATION FACTORS

In Iyer and Bilmes [2013], Iyer et al. [2013], building on the work of Conforti and Cornuéjols [1984], the authors show that for Submodular Maximization under a down-monotone constraint, the greedy algorithms can be analyzed using data-dependent approximation factors. When applied to SCSK-C, it follows that the natural greedy algorithm is a $\frac{1}{c_f}(1 - (\frac{(K_g - c_f)}{K_g})^{k_g})$-approximation algorithm where

- c_f is the curvature of the function f. $c_f = \min_{x \in V} \frac{1 - f(x|V - \{x\})}{f(x)}$.
- K_g is the size of the largest feasible set that satisfies both the constraints, $K_g = \max\{|X| : g(X) \leq \theta \ \texttt{and} \ |X| \leq k\}$.
- k_g is the size of the smallest feasible S that satisfies the constraints, but adding some element to S violates the constraint. $k_g = \min\{|X| : g(X) \leq \theta \ \texttt{and} \ \exists j \notin X, g(X \cup j) > \theta\}$.

To gauge the quality of the solution produced by the algorithm, one should be able to effectively compute the value of the expression $\frac{1}{c_f}(1 - (\frac{(K_g - c_f)}{K_g})^{k_g})$. We observe that it is NP-Hard to compute K_g.

OBSERVATION 1. *Given a submodular function g and θ, it is NP-Hard to calculate K_g where $K_g = max\{|X| : g(X) \leq \theta$ and $|X| \leq k\}$.*

Algorithm 1 Basic Greedy Algorithm

$X = \emptyset$
for $i = 1$ to k **do**
  $X = X \cup \{\arg\max_v f(X \cup \{v\}) | g(X \cup \{v\}) \leq \theta\}$
end for
return X

The proof is provided in the Appendix.

This limitation implies that while we can run the algorithm knowing that it has an approximation factor of $\frac{1}{c_f}(1 - (\frac{(K_g - c_f)}{K_g})^{k_g})$, we cannot hope to effectively compute what this term evaluates to and thus we will not be able to ascertain the quality of the solution produced. If we attempt to bound the c_f, K_g and k_g, then in the worst case $c_f = 1, K_g = k, k_g = 1$, leading to trivial $\frac{1}{k}$-approximation when applied to SCSK-C.

The above observation and discussion motivate the need for establishing guarantees with efficiently computable approximation factors.

3 GREEDY ALGORITHM FOR SCSK-C

In this section, we provide approximation guarantees, with efficiently computable approximation factors, for the natural greedy algorithm for SCSK-C. The GREEDY algorithm for SCSK-C problem is described in Algorithm 1.

The Algorithm 1 computes X by iteratively adding the "best" element to the partial solution. Given an element $v \in V$ and $X \subseteq V$, the marginal gain of v with respect to X, denoted $f(v|X)$, is $f(X \cup \{v\}) - f(X)$. Given a set $S \subseteq V$ and an integer γ, we define *Maximum Constrained Gain Element* (denoted MCGE) as the element v that achieves the maximal marginal gain, $f(v|X)$ subject to the constraint $g(X \cup \{v\}) \leq \gamma$. More formally

$$\text{MCGE}(S, \gamma) = \arg\max_{v \in V} \{f(v|S) \mid g(S \cup v) \leq \gamma\}$$

where $\arg\max\{\emptyset\}$ is considered as undefined. Given X and γ, we define *Maximum Constrained Gain* (denoted MCG) as the marginal gain of f due to $\text{MCGE}(S, \gamma)$ with respect to S. i.e., $\text{MCG}(S, \gamma) = f(\text{MCGE}(S, \gamma)|S)$.

The following theorem characterizes the solution obtained using Algorithm 1 using additive and multiplicative errors.

THEOREM 1. *Let $OPT_{k,\theta}$ be the optimal value of f under the constraints, and X be the solution returned by the Algorithm 1, then the following holds*

$$f(X) \geq (1 - 1/e)[OPT_{k,\theta}] - \sum_{i=1}^{k-1} [\text{MCG}(X_i, 2\theta) - \text{MCG}(X_i, \theta)]$$

Proof. Let $X^*_{k,\theta}$ be an optimal solution such that $f(X^*_{k,\theta}) = OPT_{k,\theta}$ and let O denote the size of $X^*_{k,\theta}$. Note that $O \leq k$.

Let X_{i-1} denote the partial solution at the start of the ith iteration of the greedy algorithm; and initially $X_0 = \emptyset$.

With each iteration i, we associate an additional set X'_i as follows. During iteration i, let u_i be an element that can maximize $f(X_{i-1} \cup \{u\})$ such that $g(X_{i-1} \cup \{u\}) \leq 2\theta$. More precisely, let $u_i = \text{MCGE}(X_{i-1}, 2\theta)$, and we say that $X'_i = X_{i-1} \cup \{u_i\}$. Note that the set X'_i is not constructed by the greedy algorithm (X'_i may violate the constraint $g(X'_i) \leq 2\theta \implies g(X'_i) \leq \theta$). The set X'_i is used for the analysis of the algorithm.

For every $1 \leq i \leq k-1$, we have the following inequalities.

$$\begin{aligned} OPT_{k,\theta} &\leq f(X^*_{k,\theta} \cup X_i) \leq f(X_i) + \sum_{e \in X^*_{k,\theta}} f(e|X_i) \\ &\leq f(X_i) + \sum_{e \in X^*_{k,\theta}} [f(X'_{i+1}) - f(X_i)] \\ &\leq f(X_i) + O \times f(X'_{i+1}) - O \times f(X_i) \end{aligned}$$

The first two inequalities follow since f is monotone and submodular. We now explain the third inequality: a subtle point here is that we cannot claim that $f(e|X_i) \geq f(X_{i+1}) - f(X_i)$ as it might be possible that $g(X_i \cup \{e\}) > \theta$ and this element e is not considered during iteration i. However, as $X^*_{k,\theta}$ is an optimal solution, we have $g(X^*_{k,\theta}) \leq \theta$, which, in turn, implies that $g(e) \leq \theta$ for every $e \in X^*_{k,\theta}$. Therefore, $g(X_i \cup \{e\}) \leq 2\theta$ due to submodularity of g. Recall that X'_{i+1} is obtained by adding $u_{i+1} = \text{MCGE}(X_i, 2\theta)$ to the set X_i. Since $g(X_i \cup \{e\}) \leq 2\theta$, it must be the case that $f(u_{i+1}|X_i) \geq f(e|X_i)$. Thus, $f(e|X_i) \leq f(X'_{i+1}) - f(X_i)$. The last inequality follows because the size of the optimal solution is O.

By adding $(O-1)OPT_{k,\theta}$ on both sides of the last inequality and rearranging terms, we obtain

$$OPT_{k,\theta} - f(X'_{i+1}) \leq \frac{O-1}{O}\left(OPT_{k,\theta} - f(X_i)\right) \quad (1)$$

This inequality relates X'_{i+1} with X_i. However, if we could relate X_{i+1} with X_i instead, then we could obtain a recurrence relation. To achieve this, we now consider the relationship between the sets X_{i+1} and X'_{i+1}.

By our definitions of X_{i+1} and X'_{i+1}, we have
$f(X_{i+1}) = f(X_i) + \text{MCG}(X_i, \theta)$.
$f(X'_{i+1}) = f(X_i) + \text{MCG}(X_i, 2\theta)$.

Thus,
$-f(X'_{i+1}) = -f(X_{i+1}) - [\text{MCG}(X_i, 2\theta) - \text{MCG}(X_i, \theta)]$.

Substituting this in Equation 1, we obtain the following recurrence relation.

$$\begin{aligned} OPT_{k,\theta} - f(X_{i+1}) \leq \frac{O-1}{O}\left(OPT_{k,\theta} - f(X_i)\right) \\ + [\text{MCG}(X_i, 2\theta) - \text{MCG}(X_i, \theta)] \end{aligned} \quad (2)$$

For notational brevity, we use $MCGD_i$ to denote $\text{MCG}(X_i, 2\theta) - \text{MCG}(X_i, \theta)$.

Algorithm 2 Basic Greedy with Additive Error Computation

```
X = ∅; A = 0
for i = 1 to k do
    w = arg max_v {f(X ∪ {v}) | g(X ∪ {v}) ≤ θ}.
    if (i ≠ 1) then
        u = arg max_v {f(X ∪ {v}) | g(X ∪ {v}) ≤ 2θ}.
        A = A + f(u|X) − f(w|X).
    end if
    X = X ∪ {w}.
end for
return A and X.
```

CLAIM 1.

$$\begin{aligned} OPT_{k,\theta} - f(X_k) \leq \left(\frac{O-1}{O}\right)^{k-1}(OPT_{k,\theta} - f(X_1)) \\ + \sum_{i=1}^{k-1} MCDG_i \end{aligned}$$

The proof of the claim is provided in the Appendix.

Since $f(X_1) \geq \frac{OPT_{k,\theta}}{O}$, it follows that $OPT_{k,\theta} - f(X_1) \leq \frac{O-1}{O} \cdot OPT_{k,\theta}$. Plugging this in the inequality from Claim 1 we obtain that

$$f(X_k) \geq (1 - 1/e)OPT_{k,\theta} - \sum_{i=1}^{k-1} MCDG_i$$

This concludes the proof. □

3.1 ADDITIVE ERROR: COMPUTATION INTERPRETATION AND TIGHTNESS

Computation. We show that additive error term $\sum_{i=i}^{k-1} \text{MCG}(X_i, 2\theta) - \sum_{i=1}^{k-1} \text{MCG}(X_i, \theta)$ can be computed very efficiently. Consider Algorithm 2. Consider an iteration ℓ of this algorithm, note that $u = \text{MCG}(X_{\ell-1}, \theta)$ and $v = \text{MCG}(X_{\ell-1}, 2\theta)$. Thus at the end of the algorithm A equals $\sum_{i=i}^{\ell-1} \text{MCG}(X_i, 2\theta) - \sum_{i=1}^{\ell-1} \text{MCG}(X_i, \theta)$. Clearly, the set X is the greedy solution. Note that the total number of calls made by Algorithm 2 to f and g is $O(nk)$, which is asymptotically the same as the number of calls made by the Algorithm 1. Here n is the size of the ground set. As stated in the introduction, this paves way for a quick understanding of the quality of the result generated by the greedy algorithm.

Interpretation. We now discuss the interpretation of the additive error. Informally, additive error captures the difference between the solutions produced by the greedy algorithms that are run with submodular budgets of 2θ and θ. More precisely, it is the following. Let X_i be the set at the end of the ith iteration of the greedy algorithm (with submodular budget θ). Let w_i be the maximum marginal gain possible with respect to X_i with submodular budget of θ and u_i be

the maximum marginal gain possible with respect to X_i with submodular budget of 2θ. The additive loss is the sum of the differences $u_i - w_i$.

Tightness. Next, we consider whether the approximation factors in the above analysis can be improved. In the above, the additive error is data-dependent, and it is natural to ask whether this is necessary. Our next result establishes that the additive error can not be made data-independent even if we settle for a multiplicative factor that is lower than $(1 - 1/e)$. We establish the following result whose proof appears in the appendix.

THEOREM 2. *There does not exist a polynomial time algorithm $\mathcal{A}$ for SCSK and SCSK-C such that it outputs a set X with guarantee $f(X) \geq d \cdot OPT - A$ where $d < 1, A > 0$ are universal constants.*

3.2 EXTENSIONS

We extend the above proof and analysis in two different directions. First, we can refine the above result and capture the multiplicative error using the curvature of the function f, denoted by c_f and defined as $1 - \min_x \frac{f_{V-\{x\}}(x)}{f(x)}$. The proof of the following theorem is provided in the appendix.

THEOREM 3. *Let X be the solution produced by Algorithm 1, then*

$$f(X) \geq \frac{1}{c_f}\left(1 - (1 - \frac{c_f}{k})^k\right) OPT_{k,\theta} - A,$$

where A is the additive error same as in Theorem 1.

We next consider a slight modification of Algorithm 2. Note that the for loop is executed exactly k times. Suppose that during an iteration i, there is no element v such that $g(X \cup \{v\}) \leq \theta$. Once this happens the algorithm does not append any new elements to X in future iterations, however, the value A could keep changing (as there could be elements u for which $g(X \cup \{u\}) \leq 2\theta$. Consider a modification where the algorithm stops when it fails to find an element v such that $g(X \cup \{v\}) \leq \theta$. In this case, the algorithm will produce a set X of size $\ell \leq k$. We can bound the quality of the solution produced as stated in the following theorem.

THEOREM 4. *Let $OPT_{k,\theta}$ be the optimal value of f under the constraints, and X be the solution with $|X| = \ell$ obtained from above describe modified version of Algorithm 2, then the following holds*

$$f(X) \geq (1 - (1 - 1/k)^\ell)[OPT_{k,\theta}] - \sum_{i=1}^{\ell-1} [MCG(X_i, 2\theta) - MCG(X_i, \theta)]$$

The proof of the above theorem is exactly the same as the proof of Theorem 1. Thus we omit the proof. Note that, both the additive error and multiplicative error (which is $(1 - (1 - 1/k)^\ell)$ can be computed efficiently in this case as well. The main difference between Theorem 1 and 4 is that Theorem 1 has a higher (and thus better) multiplicative factor but also a higher additive error (and thus worse) compared to Theorem 4.

Algorithm 3 Algorithm for DIFF-C: LINEAR-APPROX

```
S = φ
for i = 0 to λ do
    X = A(f, g, k, i)
    if f(X) − g(X) > f(S) − g(S) then
        S = X
    end if
end for
return X
```

4 FROM SCSK-C TO DIFF-C

In this section, we design algorithms for DIFF-C, that use algorithm for SCSK-C as a subroutine. Algorithm 3 (LINEAR-APPROX algorithm) presents the algorithm for DIFF-C problem.

The bound λ on the iteration is based on the maximum valuation of g; $\mathcal{A}$ denotes the algorithm for addressing the SCSK-C problem. In each iteration i (i.e., for each valuation of g), $\mathcal{A}$ is used to compute the set X for which f is maximal under the constraint that g's valuation is $\leq i$ and $|X| \leq k$. The difference between f and g at X is then compared against the prior computed difference and the larger of the two is considered as the current maximal difference.

THEOREM 5. *Let f and g be two submodular functions where f is monotone, and let $h = f - g$. In Algorithm 3, if the subroutine $\mathcal{A}$ can solve* SCSK-C *exactly, then the algorithm produces a set S such that $h(S) \geq h(OPT) - 1$. Algorithm 3 makes $O(\lambda)$ calls to $\mathcal{A}$, where $\lambda = k \times max_{e \in V} g(e)$.*

The proof is provided in the Appendix.

THEOREM 6. *In Algorithm 3, suppose that Algorithm $\mathcal{A}$ is the Basic Greedy Algorithm (Algorithm 1) for* SCSK-C*, let $h = f - g$. If Algorithm 3 outputs a set G then*

$$h(G) \geq (1 - 1/e)\, h(OPT) - A,$$

where the additive error A can be computed efficiently.

Proof. We will start with some notation. Let S_i^* is the optimal solution to the SCSK-C instance with $\theta = i$. Let G_i be the set returned by the Basic Greedy Algorithm for SCSK-C instance with $\theta = i$. Let $A(i)$ be the corresponding additive error. We first consider the case when the range of g is integers. By Theorem 1, we have for $1 \leq i \leq \lambda$,

$$f(G_i) \geq (1 - 1/e) f(S_i^*) - A(i) \tag{3}$$

Let OPT be the optimal solution for $h = f - g$, and let $\theta^* = g(OPT)$. Note that $h(OPT) = f(OPT) - \theta^*$. Let the solution returned by the Algorithm 3 occur at $i = \beta$. Thus the set G returned by the algorithm is G_β and $h(G) = h(G_\beta) = f(G_\beta) - g(G_\beta)$. Note that $g(G_\beta)$ must equal β, otherwise the algorithm would not have returned the set G_β.

Since the algorithm returned the set G_β, we have $f(G_\beta) - \beta \geq f(G_{\theta^*}) - \theta^*$. And we also know that by Inequality 3 $f(G_{\theta^*}) \geq (1 - 1/e)\, f(OPT) - A(\theta^*)$. Thus

$$\begin{array}{rl} & f(G_\beta) - \beta \\ \geq & f(G_{\theta^*}) - \theta^* \\ \geq & (1 - 1/e)\, f(OPT) - A(\theta^*) - \theta^* \\ = & (1 - 1/e)\, (f(OPT) - \theta^*) - (\theta^*/e + A(\theta^*)) \\ = & (1 - 1/e)\, h(OPT) - (\theta^*/e + A(\theta^*)) \end{array}$$

In the above we can view $\frac{\theta^*}{e} + A(\theta^*)$ as additive error. However, since we do not know the value of θ^*, we do not know how to compute this value efficiently, instead will exhibit and upper bound on this quantity that can be comouted efficiently. One way to achieve this is to compute $i/e + A(i)$, $1, \leq i, \leq \lambda$ and take the maximum of these values. This will be an upper bound on the additive error and clearly, this quantity can be computed efficiently. Below we employ another approach to bound the above quantity. We will first derive a bound on θ^*. Building on this, we derive an efficiently computable upper bound on $\frac{\theta^*}{e} + A(\theta^*)$.

We know that $f(S^*_\beta) - \beta$ is at most $f(OPT) - \theta^*$ and $f(G_\beta) - \beta$ is at least $f(G_{\theta^*}) - \theta^*$. A worst possible scenario at which this happens is $f(G_\beta)$ is as large as possible and $f(G_{\theta^*})$ is as small as possible. This happens when $f(G_\beta) = f(S^*_\beta)$ and $f(G_{\theta^*})$ equals $(1-1/e)f(OPT) - A(\theta^*)$. Thus in this scenario

$$f(G_\beta) - \beta = f(S^*_\beta) - \beta \leq f(OPT) - \theta^*$$

Since $f(G_{\theta^*}) = (1 - 1/e)f(OPT) - A(\theta^*)$, we obtain that

$$f(G_\beta) - \beta \leq \frac{f(G_{\theta^*}) + A(\theta^*)}{1 - 1/e} - \theta^*$$

Thus

$$\theta^* \leq \frac{f(G_{\theta^*}) + A(\theta^*)}{1 - 1/e} - f(G_\beta) + \beta$$

From this it follows that

$$\theta^* \leq B = \max_i \frac{f(G_i) + A(i)}{1 - 1/e} - f(G_\beta) + \beta$$

Thus B is the desired upperbound on θ^*. Note that for every i, we can compute $f(G_i) + A(i)$ while running Algorithm 3. Thus the bound B can be efficiently computed. Let $A = \max_{i \leq B}(A(i) + i/e)$. Note that $\frac{\theta^*}{e} + A(\theta^*) \leq A$. Thus we have

$$h(S) = h(G_\beta) \geq \left(1 - \frac{1}{e}\right) h(OPT) - A$$

When the range of g is not necessarily positive integers, then, as in the proof of Theorem 5 the additive error will have an additional factor of 1.

Computing the Additive Error. We note that the additive error A can be computed efficiently as follows: When call the Greedy algorithm for SCSK-C in Step 3, we can compute $A(i)$. Thus we keep track of $A(i) + i/e$ for every $1 \leq i \leq \lambda$. As discussed above we can compute the value B while running the algorithm. This implies that $A = \max_{i \leq B}(A(i) + i)$ can be computed efficiently. □

LOG-APPROX **Algorithm: a faster approximation for** DIFF-C**.** We now make a few remarks about improving the runtime of Algorithm 3. The run time of the is proportional to λ, which in turn depends on the range of g — the algorithm is invoking $\mathcal{A}(f, g, k, i)$ for every i, $1 \leq i \leq \lambda$. This could be expensive in practice. Thus we propose a modification to the Algorithm; we refer to the modified version as LOG-APPROX algorithm. This algorithm calls $\mathcal{A}(f, g, k, 2^i)$ for every i, $1 \leq \log \lambda$. This will ensure that we make only $\log \lambda$ invocations of the subroutine $\mathcal{A}$ and thus drastically reduce the run time. By doing the same analysis as above we can prove that $h(S) \geq \frac{1}{2}(1 - 1/e)h(OPT) - A$.

5 EXPERIMENTS

In this section, we empirically examine the performance of SCSK-C and DIFF-C on the application of Information Diffusion in social networks. All the algorithms are implemented in C++ and run on a Linux server with AMD Opteron 6320 CPU (8 cores and 2.8 GHz) and 64GB RAM.

Information Diffusion. The diffusion of information in a social network under various probabilistic diffusion models is captured as a submodular function Kempe et al. [2003]. For a (seed) set $X \subseteq V$, the submodular function $f(X)$ is the expected number of users influenced by X. On the other hand, there is often some cost function g associated with each seed set; a candidate g, in the context of social influence, quantifies the value of a set of entities in the network based on the number of followers of the set. We use such a submodular cost function in our experiments. The goal is to find a seed set of size $\leq k$ that maximizes f (influence) while minimizing g (cost).

Datasets. For the application of information diffusion, we collect six directed networks to conduct experiments: NetHept Net [2009], p2p-Gnutella31 Ripeanu et al. [2002], Facebook Leskovec and Mcauley [2012], Bitcoin Kumar et al. [2016], Wikipedia Leskovec et al. [2010] and DBLP Yang and Leskovec [2012]. The number of nodes of them ranges from 3,783 to 317,080. Due to space limitations, we present the plots only for three of these graphs.

Algorithm 4 Budget-Conscious Greedy Algorithm

Input: $\theta_1, \cdots, \theta_k$.
$X = \emptyset$
for $i = 1$ to k **do**
  If there is no v such that $g(X \cup \{v\}) \leq \theta_i$, then
    X remains unchanged
  Else $X = X \cup \{\arg\max_v f(X \cup \{v\}) | g(X \cup \{v\}) \leq \theta_i\}$
end for
return X

5.1 EXPERIMENTS FOR SCSK-C

The main objective we seek in these experiments is to demonstrate that the approximation factors can be computed efficiently, which helps to gain an understanding of the quality of the solution. For the Natural greedy algorithm (Algorithm 2), we compute the additive error produced and also study how the additive error changes as the submodular budget θ increases.

Comparison Algorithms. We compare the solutions produced by the Natural Greedy algorithm (Algorithm 2) with two variants. Note that during each iteration of the Algorithm 2, the entire submodular budget θ is made available. We obtain a *budget-conscious* variant of this algorithm that allows iteration i to spend at most $\theta_i < \theta$ budget. Algorithm 4 describes this strategy. By following an analysis that is very similar to that of Theorem 1, we can show that this algorithm produces a set X for such that $f(X)$ is at least $(1 - 1/e)f(OPT) - A$, and A can be computed efficiently. We use the following budget-conscious algorithms (Algorithm 4). **Equal Partition:** Use $\theta/k, 2\theta/k, \cdots \theta$ as input to the budget-conscious Greedy algorithm. **Random Partition:** Select a random sequence of thresholds to use in the budget-conscious Greedy Algorithms.

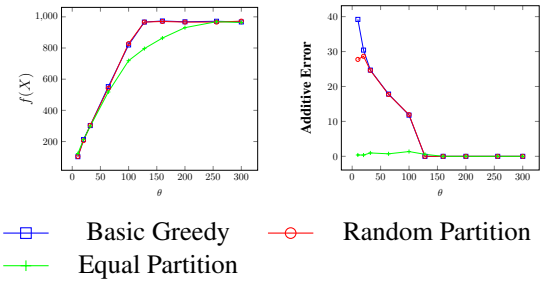

Figure 1: SCSK-C, NetHept; $k = 50$, a) θ vs. $f(X)$; b) θ vs. Additive Error with submodular cost on Basic Greedy, Random Partition and Equal Partition

Results Analyses. We chose $k = 50$ and varied the submodularity budget θ from 10 to 300. The results are shown in Fig. 1 to 4. As can be seen, the Basic Greedy, Equal Partition and Random Partition algorithms produce very similar results, except for Facebook. It can be seen from Fig. 1b, 2b, 4b, that as the submodular budget increases, the additive error decreases. Recall that the additive factor is approximately the difference between the quality of the

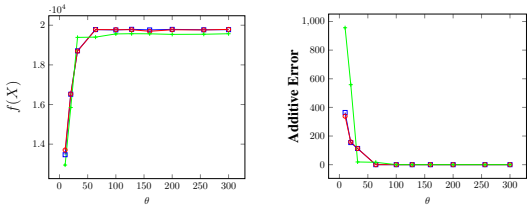

Figure 2: SCSK-C, p2p-Gnutella31; $k = 50$, a) θ vs. $f(X)$; b) θ vs. Additive Error with submodular cost on Basic Greedy, Random Partition and Equal Partition

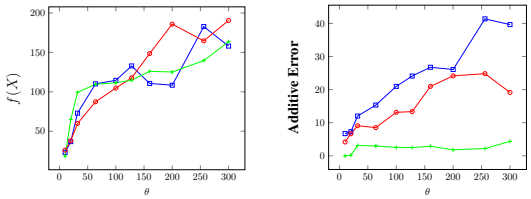

Figure 3: SCSK-C, Wikipedia; $k = 50$, a) θ vs. $f(X)$; b) θ vs. Additive Error with submodular cost on Basic Greedy, Random Partition and Equal Partition

seed sets produced with submodular constraints θ and 2θ. Thus all θ grows larges there may not be much difference between the constraints $g(X) \leq \theta$ and $g(X) \leq 2\theta$. It is likely that a set that satisfies the latter constraint will also satisfy the former constraint.

We analyze the quality of the produced solutions. For NetHept, p2p-Gnutella31 and DBLP, the additive error is less than 10% of $f(X)$ most of the time and much smaller many times. When this happens, we can conclude that for all these sets $f(X) \geq 0.53f(OPT)$. For example, for NetHept, when $k = 50, \theta = 200$, the greedy algorithm produced a solution X of size 50, and the additive error is 0 and $f(X) = 968.21$. This implies that $f(X) \geq 0.63f(OPT)$. Another example is DBLP, at $\theta = 20$, Basic Greedy produced a solution with value 13810 and the additive error is 1044. This implies that additive error is less than 7.5% of the optimal value. Thus we can be guaranteed that the value produced by the algorithm is at least $0.55f(OPT)$. For graphs such as Facebook, Bitcoin, and Wikipedia, additive errors are higher. For example, for Bitcoin with $\theta = 160$, the Basic Greedy produced a solution with value 110, whereas the additive error is 26. This implies that the value of the solution is at least $0.4f(OPT)$. The density of the graphs could explain this phenomenon. The Average degrees of Facebook, Bitcoin, and Wikipedia graphs are 43, 12, and 29, whereas, for the other graphs, the average degree is less than 8. For higher average degree graphs, there is a larger difference between the constraints $g(X) \leq \theta$ and $g(X) \leq 2\theta$.

In terms of running time, all the three algorithms can finish in 12 seconds on the NetHept network with over 15,000 nodes, demonstrating the time-efficiency of our algorithm (the details are presented in supplementary materials). Compared to Random Partition and Basic Greedy algorithms, Equal Partition is faster because it started from a small cost,

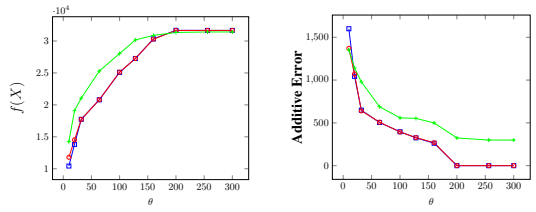

Figure 4: SCSK-C, DBLP; $k = 50$, a) θ vs. $f(X)$; b) θ vs. Additive Error with submodular cost on Basic Greedy, Random Partition and Equal Partition

which allows for faster identification of the element incurring maximal marginal gain within the cost budget (at a specific iteration). In contrast, Random Partition can generate various cost sequences while the submodular cost of each iteration for Basic Greedy is fixed.

5.2 EXPERIMENTS FOR DIFF-C

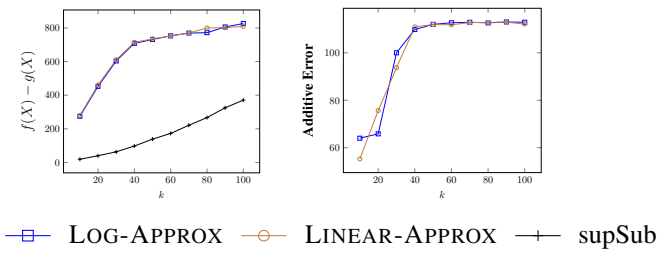

Figure 5: DIFF-C, NetHept; a) Budget vs Difference on LOG-APPROX, LINEAR-APPROX and supSub; b) Budget vs Additive error on LOG-APPROX and LINEAR-APPROX

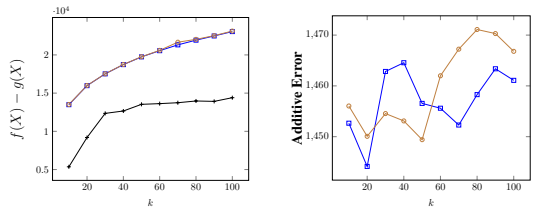

Figure 6: DIFF-C, p2p-Gnutella31; a) Budget vs Difference on LOG-APPROX, LINEAR-APPROX and supSub; b) Budget vs Additive error on LOG-APPROX, LINEAR-APPROX

We use Basic Greedy of SCSK-C (Algorithm 1) as a subroutine of LOG-APPROX and LINEAR-APPROX.

Baseline Algorithm. We compare our methods against the supSub method proposed by Iyer and Bilmes [2012a]. This replaces the submodular function g with a surrogate modular function g' and attempts to maximize $f - g'$. In addition, this method iteratively updates the surrogate modular function g' the seed set until convergence. The work of Jin et al. [2021b] presents the best known algorithm (called ROI-Greedy) to maximize $f - g'$, when f is submodular and g' is modular. In our implementation of supsub, we use this algorithm. We vary the cardinality constraint k from 10 to 100 to compare our LOG-APPROX and LINEAR-APPROX with supSub.

Results Analyses. As we see in Fig. 6 to 8, Algorithm LOG-APPROX and LINEAR-APPROX perform better than the supSub method. Interestingly, we observe that LOG-APPROX and LINEAR-APPROX produced similar results on NetHept, p2p-Gnutella31, Bitcoin and DBLP. The plots of the Bitcoin network are presented in supplementary materials. Based on this observation, it is sufficient to use Algorithm LOG-APPROX when the cost function is submodular, as it is fast and only sacrifices a small amount of objective value. While supSub performed well on Wikipedia, it required more time to converge on the Bitcoin network. Overall, there is still a substantial performance gap between our LINEAR-APPROX/LOG-APPROX and supSub. Details of the timing results are presented in supplementary materials. When we examine the additive errors, we find the same pattern as for SCSK-C. For low average degree graphs, the average (over all choices of k) additive errors are small ($8\%, 4\%, 6\%$ for NetHept, P2P and DBLP) and larger for graphs denser graphs ($29\%, 43\%, 13\%$ for Wiki, Facebook, and Bitcoin). This implies that for the Nethept graph, the (average) quality of the solution produced is at least $0.55OPT$ whereas for the Wiki graph, the (average) quality of the solution is at least $0.34OPT$.

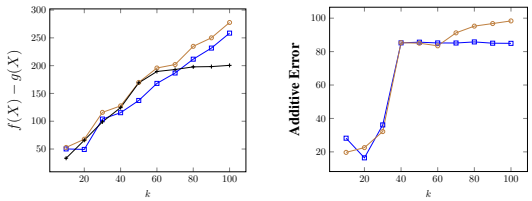

Figure 7: DIFF-C, Wikipedia; a) Budget vs Difference on LOG-APPROX, LINEAR-APPROX and supSub; b) Budget vs Additive error on LOG-APPROX and LINEAR-APPROX

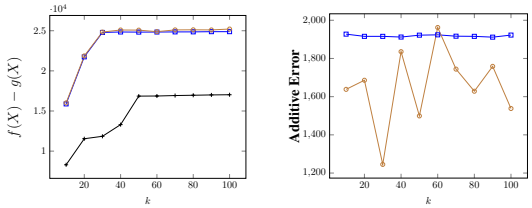

Figure 8: DIFF-C, DBLP; a) Budget vs Difference on LOG-APPROX, LINEAR-APPROX and supSub; b) Budget vs Additive error on LOG-APPROX and LINEAR-APPROX

6 CONCLUSIONS

In this work, for SCSK-C and DIFF-C, we designed algorithms, and established multiplicative-additive approximation guarantees on the quality of the solutions produced while ensuring that the multiplicative factor and the additive error can be computed efficiently. An interesting research direction is to extend this methodology to other submodular optimization problems.

Acknowledgements

The work was supported in part by the NSF grants 1934884 and 2130536.

References

Nethept. https://microsoft.com/en-us/research/people/weic/, 2009.

Niv Buchbinder and Moran Feldman. Submodular functions maximization problems. In Teofilo F. Gonzalez, editor, *Handbook of Approximation Algorithms and Metaheuristics, Second Edition, Volume 1: Methologies and Traditional Applications*, pages 753–788. Chapman and Hall/CRC, 2018.

Niv Buchbinder, Moran Feldman, and Mohit Garg. Deterministic $(1/2+\varepsilon)$-approximation for submodular maximization over a matroid. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 241–254. SIAM, 2019.

Gruia Calinescu, Chandra Chekuri, Martin Pal, and Jan Vondrák. Maximizing a monotone submodular function subject to a matroid constraint. *SIAM Journal on Computing*, 40(6):1740–1766, 2011.

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

Victoria G. Crawford, Alan Kuhnle, and My T. Thai. Submodular cost submodular cover with an approximate oracle. In *International Conference on Machine Learning*, 2019.

Moran Feldman, Zeev Nutov, and Elad Shoham. Practical budgeted submodular maximization. *arXiv preprint arXiv:2007.04937*, 2020.

Chris Harshaw, Moran Feldman, Justin Ward, and Amin Karbasi. Submodular maximization beyond non-negativity: Guarantees, fast algorithms, and applications. In *International Conference on Machine Learning*, pages 2634–2643. PMLR, 2019.

Rishabh Iyer and Jeff Bilmes. Submodular optimization with submodular cover and submodular knapsack constraints. In *International Conference on Neural Information Processing Systems*, pages 2436–2444, USA, 2013. Curran Associates Inc. URL http://dl.acm.org/citation.cfm?id=2999792.2999884.

Rishabh K. Iyer and Jeff A. Bilmes. Algorithms for approximate minimization of the difference between submodular functions, with applications. *CoRR*, abs/1207.0560, 2012a. URL http://arxiv.org/abs/1207.0560.

Rishabh K. Iyer and Jeff A. Bilmes. Algorithms for approximate minimization of the difference between submodular functions, with applications. In *Conference on Uncertainty in Artificial Intelligence*, 2012b.

Rishabh K. Iyer, Stefanie Jegelka, and Jeff A. Bilmes. Fast semidifferential-based submodular function optimization. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, volume 28 of *JMLR Workshop and Conference Proceedings*, pages 855–863. JMLR.org, 2013.

Stefanie Jegelka and Jeff A. Bilmes. Submodularity beyond submodular energies: Coupling edges in graph cuts. In *The 24th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2011, Colorado Springs, CO, USA, 20-25 June 2011*, pages 1897–1904. IEEE Computer Society, 2011.

Tianyuan Jin, Yu Yang, Renchi Yang, Jieming Shi, Keke Huang, and Xiaokui Xiao. Unconstrained submodular maximization with modular costs: Tight approximation and application to profit maximization. *Proc. VLDB Endow.*, 14(10):1756–1768, 2021a.

Tianyuan Jin, Yu Yang, Renchi Yang, Jieming Shi, Keke Huang, and Xiaokui Xiao. Unconstrained submodular maximization with modular costs: Tight approximation and application to profit maximization. *Proceedings of the VLDB Endowment*, 14(10):1756–1768, 2021b.

Yoshinobu Kawahara and Takashi Washio. Prismatic algorithm for discrete D.C. programming problem. In John Shawe-Taylor, Richard S. Zemel, Peter L. Bartlett, Fernando C. N. Pereira, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain*, pages 2106–2114, 2011.

D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 137–146, 2003.

Andreas Krause and Daniel Golovin. Submodular function maximization. In Lucas Bordeaux, Youssef Hamadi, and Pushmeet Kohli, editors, *Tractability: Practical Approaches to Hard Problems*, pages 71–104. Cambridge University Press, 2014.

Andreas Krause, Ajit Paul Singh, and Carlos Guestrin. Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. *J. Mach. Learn. Res.*, 9:235–284, 2008.

Srijan Kumar, Francesca Spezzano, VS Subrahmanian, and Christos Faloutsos. Edge weight prediction in weighted signed networks. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on*, pages 221–230. IEEE, 2016.

Jure Leskovec and Julian Mcauley. Learning to discover social circles in ego networks. *Advances in neural information processing systems*, 25, 2012.

Jure Leskovec, Daniel Huttenlocher, and Jon Kleinberg. Predicting positive and negative links in online social networks. In *Proceedings of the 19th international conference on World wide web*, pages 641–650, 2010.

Wenxin Li, Moran Feldman, Ehsan Kazemi, and Amin Karbasi. Submodular maximization in clean linear time. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=JXY11Tc9mwY.

Hui Lin and Jeff Bilmes. How to select a good training-data subset for transcription: Submodular active selection for sequences. In *Annual Conference of the International Speech Communication Association*, pages 2859–2862, 01 2009.

Hui Lin and Jeff Bilmes. Optimal selection of limited vocabulary speech corpora. In *Annual Conference of the International Speech Communication Association*, pages 1489–1492, 01 2011.

M. Narasimhan and J. Bilmes. A submodular-supermodular procedure with applications to discriminative structure learning. In *(UAI)*, pages 404–412, 2005.

George Nemhauser, Laurence Wolsey, and M L. Fisher. An analysis of approximations for maximizing submodular set functions. *Mathematical Programming*, 14:265–294, 12 1978a.

George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functions—i. *Mathematical programming*, 14(1):265–294, 1978b.

Matei Ripeanu, Ian Foster, and Adriana Iamnitchi. Mapping the gnutella network: Properties of large-scale peer-to-peer systems and implications for system design. *arXiv preprint cs/0209028*, 2002.

Maxim Sviridenko. A note on maximizing a submodular set function subject to a knapsack constraint. *Operations Research Letters*, 32(1):41–43, 2004.

Peng-Jun Wan, Ding-Zhu Du, Panos Pardalos, and Weili Wu. Greedy approximations for minimum submodular cover with submodular cost. *Computational Optimization and Applications*, 45(2):463–474, Mar 2010. ISSN 1573-2894. doi: 10.1007/s10589-009-9269-y. URL https://doi.org/10.1007/s10589-009-9269-y.

Jaewon Yang and Jure Leskovec. Defining and evaluating network communities based on ground-truth. In *Proceedings of the ACM SIGKDD Workshop on Mining Data Semantics*, pages 1–8, 2012.

Grigory Yaroslavtsev, Samson Zhou, and Dmitrii Avdiukhin. "bring your own greedy"+ max: near-optimal 1/2-approximations for submodular knapsack. In *International Conference on Artificial Intelligence and Statistics*, pages 3263–3274. PMLR, 2020.