Efficient Algorithms and Lower Bounds for Submodular Maximization

Wenxin Li
The Ohio State University
wenxinliwx.1@gmail.com
li.7328@osu.edu

Ness B. Shroff
The Ohio State University
shroff.11@osu.edu

April 16, 2020

Abstract

In this work, we study constrained submodular maximization problems and design algorithms that improve the state-of-the-art in terms of query complexity and/or approximation guarantee. We first present the *adaptive decreasing threshold* algorithm, which achieves an approximation ratio of $(1 - 1/e - \varepsilon)$ by performing $O(\max\{\varepsilon^{-1}, \log\log k\})$ queries per element. To the best of our knowledge, this is currently the fastest known **deterministic** algorithm, and nearly achieves the optimal approximation ratio.

We also study several other well-known monotone submodular maximization problems. First, for the intersection of p-system and d knapsack constraints, we propose the *backtracking threshold* algorithm that obtains an $(1/(p + \frac{7}{4}d + 1) - \Theta(1))$-approximate solution by performing $O(\log n \cdot \log\log n)$ queries per element, which improves the state-of-the-art in both time complexity and approximation ratio. We next present an $(7/16 - \varepsilon)$-approximate algorithm for a single knapsack constraint, which requires $O(\max\{\varepsilon^{-1}, \log\log n\})$ queries per element and two passes in the streaming setting.

Query complexity lower bounds of submodular maximization problems are also studied in this paper. We first show a $o(n/\log n)$ lower bound for cardinality constraint and monotone objective. Using a similar approach, we are able to present a complete characterization of the query complexity of unconstrained submodular maximization. To establish the query complexity lower bounds, we introduce a general relationship between randomized and deterministic complexity of approximation algorithms. We envision that this characterization may be used in other problems and may be interesting in its own right.

1 Introduction

A set function $f : 2^E \to \mathbb{R}^+$ defined on ground E of size n is *submodular*, if for any two subsets $S, T \subseteq E$, the inequality $f(S) + f(T) \geq f(S \cup T) + f(S \cap T)$ holds. It is *monotone non-decreasing* if $f(S) \leq f(T)$ holds for any two sets $S \subseteq T \subseteq E$. Submodular functions form a natural class of set functions with the property of diminishing returns, which has numerous applications in computer science, economics, and operation research. For example, viral marketing [35, 36], sensor and caching networks [72, 39, 34], context-aware mobile computing [42, 43], and optimal control [70]. Many machine learning problems which are inherently discrete, such as feature selection [39], document

summarization [46, 47, 30, 53] and news recommendation [68], can also be cast as submodular maximization problems. Because of its widespread applicability, there has been a vast amount of literature on submodular maximization subject to diverse types of constraints [58, 37, 63, 12, 17, 26, 67, 28, 27, 16, 62]. However, many of these algorithmic results do not scale well for practical applications of large-size [4, 64]. Badanidiyuru and Vondrák [4] formally investigated the problem of designing fast algorithms with provable guarantees for submodular maximization problems. Obtaining fast running time is of fundamental importance in both theory and practice [20] and there has been a considerable amount of interest in this direction [2, 27, 41, 10, 16, 54, 19, 20], together with related large-scale scenarios including distributed settings [5, 52, 55], online and streaming models [1, 3, 38, 15, 11, 13, 14, 32, 61, 25], etc.

In this paper, we focus on improving both the computational speed and the approximation guarantee of submodular maximization algorithms. The first problem we consider is maximizing a monotone submodular function subject to the cardinality constraint. The greedy algorithm [58] is known to achieve the optimal approximation ratio of $(1-1/e)$ [58, 57, 21], and requires $O(k)$ queries per element. To accelerate the standard greedy paradigm, Badanidiyuru and Vondrák [4] proposed the *threshold greedy algorithm*, which avoids directly seeking the element with the maximum marginal increment, by selecting elements with marginal values no less than multiplicatively decreasing thresholds. The threshold greedy algorithm is shown to achieve an approximation ratio of $(1-1/e-\varepsilon)$ via $O((1/\varepsilon)\log(n/\varepsilon))$ function evaluations per element. It is natural to wonder: can the complexity barrier of $O((1/\varepsilon)\log(n/\varepsilon))$ queries per element for achieving the almost optimal approximation ratio *deterministically* be further improved? We answer this question by providing a faster algorithm[1] in Theorem 1.

As a natural generalization of the cardinality case, the classic knapsack constraint has also been well studied. Sviridenko [63] proposed the density greedy algorithm with partial enumeration, which achieves the optimal ratio of $(1-1/e)$ in $O(n^5)$ time. For the line of algorithm acceleration, the current best result is due to Ene and Nguyen [19], in which a nearly linear randomized $O((1/\varepsilon)^{O(1/\varepsilon^4)} n\log^2 n)$ time algorithm with approximation factor $(1-1/e-\varepsilon)$ was proposed. However, the problem under the streaming model is less well understood, in which there is an inherent trade-off among performance metrics including approximation guarantee, time complexity and number of passes over the stream. Huang et al. [32, 33] proposed an $(0.363-\varepsilon)$-approximate single pass streaming algorithm that requires $O((1/\varepsilon^4)\log^4 n)$ space and queries per element, together with a $(0.4-\varepsilon)$-approximate three pass algorithm with the same space and running time requirements as the single pass algorithm. However, closing the gap between the best algorithm and hardness result in the streaming model for a single knapsack constraint has remained open. We present our algorithm in Theorem 5.

For the more general type of multiple knapsack constraints, Azar and Gamzu [2] proposed a multiplicative weight update (MWU)-based greedy algorithm that achieves a width-dependent approximation ratio of $\Omega(1/d^{1/W})$, where d represents the number of constraints and W refers to the width of the packing system. The approximation guarantee can be further improved to $\Omega(1/d^{1/W+1})$ for a binary cost matrix. There also exists a randomized $(1-1/e-\varepsilon)$-approximate algorithm [40], when the number of knapsack constraints is constant. In addition to the above-mentioned results, one question in need of further investigation is: for multiple (even constant number of binary) packing constraints, does there exist a time efficient *deterministic* algorithm with *width independent*

[1]There is a $O(n\varepsilon^{-1}\log(\varepsilon^{-1}\log n))$ time algorithm that was obtained independently in [31]. Comparisons between our algorithm [45] with [31] is presented in Section 2.2.

constant approximation ratio? We answer this in affirmative in Lemma 6.

The most general type of constraint considered in our paper is the intersection of a p-system and d knapsack constraints [4]. p-system generalizes many classic combinatorial constraints such as the intersection of p matroids, p-set packing [4], etc. The current best result for maximizing a monotone submodular function subject to the intersection of p-system and d knapsack constraints is due to Badanidiyuru and Vondrák [4], in which an $(1/(p+2d+1)-\varepsilon)$-approximate algorithm was proposed and $O((n/\varepsilon^2)\log^2(n/\varepsilon))$ oracle queries per element are required. On the inapproximability side, there is a lower bound of $(1-e^{-p})+\varepsilon \leq 1/(p+1/2)+\varepsilon$ even for p-extendible system [24]. To the best of our knowledge, there is no algorithm that is able to move closer towards the lower bound using a comparable number of queries. We address this challenge by providing an algorithm with better performance guarantee in Theorem 4.

Finally, in spite of all the attention that submodular maximization has received and the large literature on the complexity upper bound side, its fine-grained query complexity lower bound remains open. Recall that the decreasing marginal increment property of the submodular function is widely known as the discrete analog of decreasing derivative of a concave function. However, while tight complexity bounds for continuous optimization, which first appeared in Nemirovski and Yudin [59] in 1983, have already been established for various structural assumptions (*e.g.*, smooth and non-smooth, convex and strongly convex [59, 60, 6]), *the query complexity lower bound of maximizing a submodular function is still not fully understood.* Given the widespread applications of submodular maximization[2], understanding its inherent fundamental computational complexity is an urgent issue to address. To that end, we present our lower bound results in Theorems 2 and 3.

1.1 Results Overview

Cardinality Constraint. In Section 2, we present a faster deterministic algorithm for the cardinality constraint, with performance guarantee claimed by the following theorem. We **improve** the query complexity of the threshold greedy algorithm [4], which requires $O((1/\varepsilon)\log(n/\varepsilon))$ function evaluations per element, to $O(\max\{\varepsilon^{-1}, \log\log n\})$.

Theorem 1. *There is an $(1-1/e-\varepsilon)$-approximate deterministic algorithm for the cardinality constrained monotone submodular maximization problem $\max_{S\subseteq E, |S|\leq k} f(S)$, which performs $O(\max\{\varepsilon^{-1}, \log\log n\})$ value oracle queries per element.*

Related work [31]. An $O(n\varepsilon^{-1}\log(\varepsilon^{-1}\log n))$ time algorithm was obtained independently in [31]. We compare our algorithm [45] with that in [31] in Section 2.2.

Query complexity lower bounds of submodular maximization. In Section 3 we also establish an $O(n/\log n)$ complexity lower bound[3] when cardinality bound $k \in [n/2]$, for which we reveal the following general relationship on query complexity between deterministic and randomized algorithms with desirable approximation guarantees. The conclusion enables us to extend our arguments to the randomized setting and it relies heavily on the *scale free* property that is specified in Definition 10. Due to space limitation, the proof of Theorem 2 is presented in Appendix B.4.

Theorem 2. *Suppose that there exists a scale-free input instance set $\mathcal{I}_s$ for problem $\mathcal{P}$, on which the worst case time complexity of any α-approximate deterministic algorithm is in the order of*

[2]There exists lower bounds for submodular minimization [29].

[3]There exists a linear complexity bound when k is constant.

$\Omega(\mathcal{T}(\mathcal{P}))$. *Then the time complexity of any* $(\alpha+\delta)$*-approximate randomized algorithm* $\mathcal{A}$ *of problem* $\mathcal{P}$ *must be in the order of* $\Omega(\mathcal{T}(\mathcal{P}))$*, where* δ *is any constant such that* $\alpha+\delta \leq 1$.

Based on Theorem 2 and similar arguments as that for cardinality constraint, we prove that the double greedy algorithm is optimal in query complexity[4]. Consequently, we have the following complete characterization of query complexity of USM.

Theorem 3. *For any constant* $\delta > 0$*, no (randomized) algorithm can achieve an approximation ratio of* $(\frac{1}{4}+\delta)$ *for USM, while making* $o(n/\log n)$ *queries to* $f(\cdot)$*. In addition, there exists an algorithm*[5] *for the instance of finding the maximum cut in a complete bipartite directed graph, using* $O(n/\log n)$ *queries to the cut function* $f_C(\cdot)$ *in (16). As a consequence, let* $f^\sharp(\rho)$ *denote the minimum number of queries required to obtain a* ρ*-approximation for USM, as shown in Figure 1, we have* $f^\sharp(\rho) = 0$ *for* $\rho \in [0, \frac{1}{4}]$ *[22],* $f^\sharp(\rho) = \widetilde{\Theta}(n)$ *for*[6] $\rho \in (\frac{1}{4}, \frac{1}{2}]$ *[9] and* $f^\sharp(\rho) = \Omega(\exp(n))$ *for* $\rho \in (\frac{1}{2}, 1]$ *[22].*

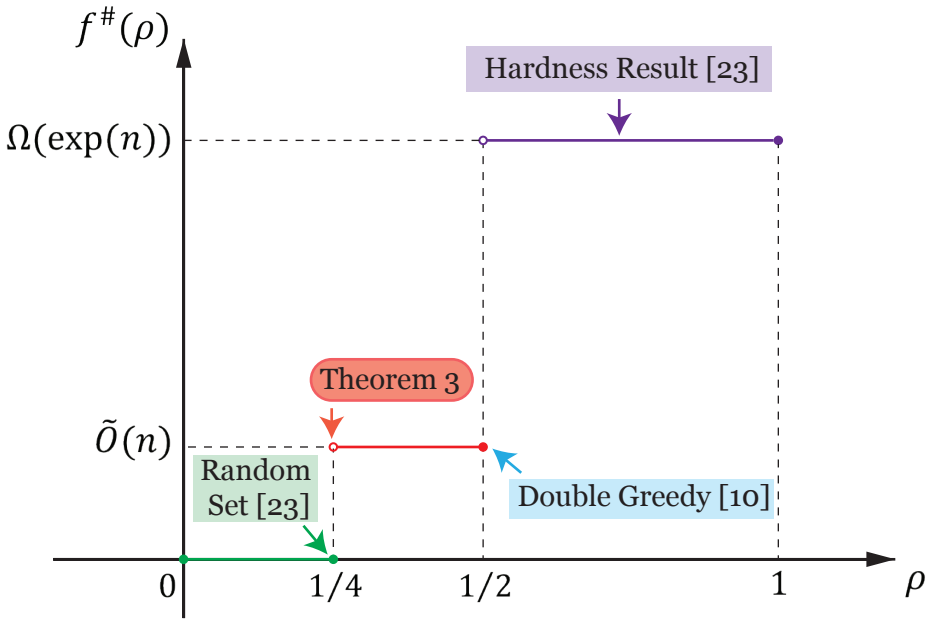

Figure 1: Query complexity of USM: a graphic illustration of $f^\sharp(\cdot)$. Here $f^\sharp(\rho)$ refers to the minimum query complexity of ρ-approximate algorithm for USM.

Intersection of p-system and d-knapsack constraints. In Section 4, we study the problem when the constraint set $\mathcal{I}$ represents the intersection of p-system and d knapsack constraints.

Theorem 4. *There is an* $(1/(p+\frac{7}{4}d+1) - \Theta(1))$*-approximation algorithm for maximizing a non-negative monotone submodular function subject to a* p*-system and* d *knapsack constraints, which performs* $O(\log n \cdot \log\log n)$ *value and membership oracle queries per element.*

[4]We thank Moran Feldman as this application is noted and realized from [23].
[5]We thank XYZ for pointing out this fact.
[6]The $\widetilde{\Theta}(\cdot)$ notation hides terms poly-logarithmic in n.

In Theorem 4 we **improve** the $(1/(p+2d+1)-\varepsilon)$-approximate algorithm in [4], which requires $O((n/\varepsilon^2)\log^2(n/\varepsilon))$ oracle queries per element. It is important to remark that the result established in Theorem 4 surpasses the state-of-the-art [4] **both** in approximation ratio and time complexity, and the improvement in approximation ratio is constant for constant value of p and d. The proof of Theorem 4 is given in Appendix C.9. It is worth pointing out that we can obtain a similar result for non-monotone objective function, though we omit the details here.

Packing Constraints. In Section 5 we study the case when $\mathcal{I}$ represents a packing constraint, *i.e.*, $\mathcal{I} = \{S \subseteq E | \mathbf{A}\mathbf{x}_S \leq \mathbf{b}\}$. We first note that within $O(n \cdot \max\{\varepsilon^{-1}. \log\log n\})$ time, it is possible to obtain an approximation of $1/(\frac{7}{4}d + \frac{9}{10} + \frac{9}{120d+16}) - O(\varepsilon)$, which is better than the ratio of $(1/(1+\frac{7}{4}d)-\varepsilon)$ implied by Theorem 4. The proof is presented in Appendix D.1. Furthermore, we investigate two specific forms of packing constrained optimization problems. The first one is the well-known knapsack constrained submodular maximization problem, for which the result is summarized in the following Theorem 5.

Theorem 5. *There is an $(7/16-\varepsilon)$-approximate algorithm for maximizing a monotone submodular function subject to a single knapsack constraint, which requires $O(\max\{\varepsilon^{-1}, \log\log n\})$ value queries per element. Our algorithm can be adapted to the* streaming *setting, in which the approximation ratio of $(7/16-\varepsilon)$ can be achieved within two passes over the data stream, while using $O(n\log n/\varepsilon)$ space and performing $O(\log n/\varepsilon)$ queries per element.*

In Theorem 5 we **improve** the $(0.4-\varepsilon)$-approximate algorithm in [32, 33], which requires $O(\log^4 n/\varepsilon^4)$ queries per element, $O(n\log^4 n/\varepsilon^4)$ space and three passes on the stream. We remark that our algorithm achieves better performance guarantees in **all** three dimensions of time complexity, space complexity and number of passes. The proof of Theorem 5 is presented in Appendix D.4

Our next result applies to binary packing constraint with constant dimension. We would like to emphasize that our approximation ratio is **width independent** and holds deterministically. It is worth mentioning that [56] proposed the first deterministic non-trivial algorithm for a constant number of packing constraints. However, it requires $O(n^{O(\text{poly}(1/\varepsilon))})$ time to achieve an approximation ratio of $(1/e-\varepsilon)$, while the running time of our algorithm for constant number of binary packing constraint is **nearly linear** in the input size.

Lemma 6. *There is a $(1/2-\varepsilon)$-approximate deterministic algorithm that performs $O_\varepsilon(\log\log n)$ value oracle queries per element*[7]*, for the binary packing constraint with constant dimension,* i.e.*, $\mathbf{A} \in \{0,1\}^{d\times n}$ where $d = O(1)$. The result also holds if $\mathbf{A} \in \mathbb{F}^{d\times n}$, where $\mathbb{F}$ consists of constants and $|\mathbb{F}| = O(1)$.*

In some of our results, we try to understand the boundary of the approximation guarantee that one algorithm can achieve via nearly linear number of queries in total, *e.g.*, $O_\varepsilon(\log\log n)$ queries per element, while $\Omega_\varepsilon(\log n)$ queries per element are required in previous works, which is exponentially higher than the complexity result(s) in this paper.

[7] As in the literature (*e.g.*, [1]), the $O_\varepsilon(\cdot)$ notation hides the dependence on ε, as the hidden term $h(\varepsilon)$ is constant for constant ε.

1.2 Techniques

Adaptive decreasing threshold algorithm. We first show that an $(1 - 1/e - \varepsilon)$-approximate solution can be obtained via $O(n/\varepsilon)$ queries, if we are given access to a constant approximation of $f(\mathrm{OPT})$. To this end, we maintain an upper and lower estimate on the value of $f(\mathrm{OPT})$, and adaptively update the estimates based on the outcome of each iteration. More specifically, we utilize a set of carefully designed thresholds that are related to the estimates in the previous iteration, to select elements with marginal gains above the threshold. Then the estimates will be updated according to the objective value of sets selected in current iteration. The two estimates are shown to have a constant gap at the end of the algorithm. We compare ADT with existing techniques in Section 2.2.

Backtracking threshold algorithm. The *backtracking threshold* (BT) algorithm is proposed for improving the approximation ratio when knapsack constraints are involved. The BT algorithm selects elements with marginal increments above a threshold into the candidate solution set, until the total cost exceeds the given budget. Then starting from current element, it constructs a new candidate solution set by recursively seeking elements in the first candidate solution set under the budget constraint. Those recursively constructed sets perform as important counterparts of the first solution. In addition, the *down-closed* property of the constraint set ensures the feasibility of obtained solution.

Approximately guessing in the value space deterministically. For binary packing constraint, the desired performance guarantee is obtained by considering the appropriate residual problem, for which we need to select a target set. Directly searching among all the possible candidate sets may incur high query complexity, we reduce the complexity by selecting the element with almost identical marginal gain (with respect to the current solution) at each step. The solution to the corresponding residual problem is proved to achieve a deterministic approximation ratio of $(1/2 - \varepsilon)$. One salient feature of this deterministic selection procedure is that the query complexity has a near linear dependence on the input instance size.

2 Faster Algorithm for the Cardinality Constraint

Our *adaptive decreasing threshold* (ADT) algorithm is specified in Algorithm 1, which mainly consists of two phases—a *preprocessing procedure* and a *refined threshold decreasing procedure*. In the i-th iteration of the preprocessing procedure, we maintain $\bar{w}_i$ and $\underline{w}_i$ as an upper and lower estimate on the optimal objective value. At the end iteration i, the lower estimate of $f(\mathrm{OPT})$ is

updated as the maximum function value of the sets obtained in i-th iteration.

Algorithm 1: Adaptive Decreasing Threshold (ADT) Algorithm

Initialization: $\underline{\omega}_1 \leftarrow \max_{e\in E} f(e)$, $\bar{\omega}_1 \leftarrow k \cdot \underline{\omega}_1$, $U \leftarrow \varnothing$, $\ell \leftarrow \lceil \log\log k \rceil$.
for $i = 1 : \ell$ **do**
  $\alpha_i = \exp(\log k \cdot e^{-i}) - 1$, $\theta = \underline{\omega}_i$
  while $\theta \le \bar{\omega}_i$ **do**
    $S_\theta^{(i)} \leftarrow \varnothing$
    for $e \in E$ **do**
      if $f(S_\theta^{(i)} + e) - f(S_\theta^{(i)}) \ge \theta/2k$ *and* $|S_\theta^{(i)}| \le k$ **then**
        $S_\theta^{(i)} \leftarrow S_\theta^{(i)} + e$
    $\theta \leftarrow \theta(1 + \alpha_i)$
  $\underline{\omega}_{i+1} \leftarrow \max_\theta f(S_\theta^{(i)})$, $\bar{\omega}_{i+1} \leftarrow 2(1 + \alpha_i) \cdot \underline{\omega}_{i+1}$
$\tau \leftarrow \frac{2e\underline{w}_\ell}{k}$
while $\tau \ge \frac{w_\ell}{ek}$ **do**
  for $e \in E$ **do**
    if $f(U + e) - f(U) \ge \tau$ *and* $|U| \le k$ **then**
      $U \leftarrow U + e$
  $\tau \leftarrow \tau - \frac{\varepsilon\underline{\omega}_\ell}{k}$
return U

2.1 Performance analysis of ADT

Analysis of the preprocessing procedure. We first prove that in the first phase of our algorithm, $\underline{w}_i$ and $\bar{w}_i$ are always valid lower and upper bounds on $f(\mathrm{OPT})$.

Lemma 7. *For any $i \in [\ell]$, the optimal objective value always lies between $\underline{w}_i$ and $\bar{w}_i$,* i.e.*, $\underline{w}_i \le f(\mathrm{OPT}) \le \bar{w}_i$.*

Proof: We prove this lemma by induction. For the base case when $i = 1$, as $\underline{w}_1$ is initialized to be the maximum objective value of a singleton, Lemma 7 is equivalent to $\max_e f(e) \le f(\mathrm{OPT}) \le k \cdot \max_e f(e)$, which follows from the submodularity of $f(\cdot)$. Notice that for $i \ge 2$, we have $\underline{w}_i = \max_\theta f(S_\theta^{(i-1)})$, where $S_\theta^{(i-1)}$ is a feasible solution as $|S_\theta^{(i-1)}| \le k$. Hence $\underline{w}_i$ is always a valid lower bound of $f(\mathrm{OPT})$ and what remains to prove is $\bar{w}_i \ge f(\mathrm{OPT})$ for $\forall i \in [\ell]$.

Induction Step. Assume that $\bar{w}_i \ge f(\mathrm{OPT})$ holds for $i = q$. In the following, we complete the proof for $i = q + 1$ by lower bounding the objective value of $f(S_{\theta_q^*}^{(q)})$. Observe that in the q-th iteration, θ takes values in set $\Theta_q = \{\underline{w}_q, \underline{w}_q(1 + \alpha_q), \ldots, \underline{w}_q(1 + \alpha_q)^{\lfloor \log(\bar{w}_q/\underline{w}_q)/\log(1+\alpha_q) \rfloor}$. Combined with the induction assumption $\bar{w}_q \ge f(\mathrm{OPT})$, there must exist some $\theta_q^* \in \Theta_q$ such that $\theta_q^* \le f(\mathrm{OPT}) \le (1 + \alpha_q)\theta_q^*$. Consider the iteration in which $\theta = \theta_q^*$, it can be seen that $f(S_{\theta_q^*}^{(q)})$ must be no less than its size multiplied by the corresponding threshold, if there are exactly k elements in $S_{\theta_q^*}^{(q)}$, *i.e.*, $f(S_{\theta_q^*}^{(q)}) \ge \frac{\theta_q^*}{2k} \cdot |S_{\theta^*}^{(q)}| \ge \frac{\theta_q^*}{2} \cdot \mathbb{1}_{|S_{\theta_q^*}^{(q)}|=k} \ge \frac{f(\mathrm{OPT})}{2(1+\alpha_q)} \cdot \mathbb{1}_{|S_{\theta_q^*}^{(q)}|=k}$. If there are less than k elements

in set $S^{(q)}_{\theta^*_q}$, elements in $\mathrm{OPT} \setminus S^{(q)}_{\theta^*_q}$ will have a small marginal gain with respect to $S^{(q)}_{\theta^*_q}$ and

$$\begin{aligned}[f(\mathrm{OPT}) - f(S^{(q)}_{\theta^*_q})] \cdot \mathbb{1}_{|S^{(q)}_{\theta^*_q}|<k} \leq & f(\mathrm{OPT} \cup S^{(q)}_{\theta^*_q}) - f(S^{(q)}_{\theta^*_q}) && \text{(monotonicity)} \\ \leq & \sum_{e \in \mathrm{OPT}} \left[f(S^{(q)}_{\theta^*_q} + e) - f(S^{(q)}_{\theta^*_q}) \right] \leq \sum_{e \in \mathrm{OPT}} \frac{\theta^*_q}{2k} = \frac{\theta^*_q}{2} \leq \frac{f(\mathrm{OPT})}{2}.\end{aligned}$$

This implies that $f(S^{(q)}_{\theta^*}) \geq \frac{f(\mathrm{OPT})}{2} \cdot \mathbb{1}_{|S^{(q)}_{\theta^*_q}|<k}$. To summarize, we have

$$f(S^{(q)}_{\theta^*}) \geq \frac{f(\mathrm{OPT})}{2(1+\alpha_q)} \cdot \mathbb{1}_{|S^{(q)}_{\theta^*_q}|=k} + \frac{f(\mathrm{OPT})}{2} \cdot \mathbb{1}_{|S^{(q)}_{\theta^*_q}|<k} \geq \frac{f(\mathrm{OPT})}{2(1+\alpha_q)}. \tag{1}$$

Therefore $\bar{w}_{q+1} = 2(1+\alpha_q)\underline{w}_{q+1} = 2(1+\alpha_q)\max_\theta f(S^{(q)}_\theta) \geq 2(1+\alpha_q) f(S^{(q)}_{\theta^*_q}) \geq f(\mathrm{OPT})$, which are mainly based on (1) and the definition of $\underline{w}_i$, $\bar{w}_i$. The proof is complete. □

As a consequence, we have the following corollary characterizing the relationship between $\underline{w}_\ell$ and $f(\mathrm{OPT})$. The correctness of Corollary 8 follows from Lemma 7 and the fact that $\bar{w}_\ell = (1+\alpha_\ell)\underline{w}_\ell = \exp(\log k \cdot e^{-\lceil \log\log k \rceil})\underline{w}_\ell \leq e\underline{w}_\ell$.

Corollary 8. *For the lower estimate obtained at the end of the ℓ-iteration, we have $\frac{f(\mathrm{OPT})}{2e} \leq \underline{w}_\ell \leq f(\mathrm{OPT})$.*

Analysis and motivation of the refined threshold decreasing procedure. For set U returned by Algorithm 1, we have $f(U) \geq (1-1/-\varepsilon)\cdot f(\mathrm{OPT})$, the analysis and proof of Theorem 1 is presented in Appendix B.1. Here we explain the motivation of the second phase in detail. Recall that in the standard greedy algorithm, each selected element has a marginal gain that is no less than the average residual distance between OPT and the function value of current solution, which ensures the $(1-1/e)$ approximation ratio. We make the observation that, we can always assume that the average residual distance lies in an interval with constant multiplicative gap, *i.e.*, $\frac{f(\mathrm{OPT})-f(U)}{k} \in \left[\frac{f(\mathrm{OPT})}{ek}, \frac{f(\mathrm{OPT})}{k}\right]$, since U is already a desirable solution set if $f(U) \geq (1-1/e)\cdot f(\mathrm{OPT})$. This motivates us to guess the average residual distance in each step of the greedy algorithm, via the *arithmetic* sequence $\widetilde{\mathcal{G}} = \left\{\frac{f(\mathrm{OPT})}{k}, (1-\varepsilon)\cdot\frac{f(\mathrm{OPT})}{k}, (1-2\varepsilon)\cdot\frac{f(\mathrm{OPT})}{k}, \ldots, \frac{f(\mathrm{OPT})}{ek}\right\}$.

Proposition 9 (Complexity of Algorithm 1)**.** *Algorithm 1 performs $O(\max\{\varepsilon^{-1}, \log\log n\})$ queries per element.*

Proof: Note that in the i-th iteration of the preprocessing procedure, we perform $O(\frac{\log(\underline{\omega}_i/\bar{\omega}_i)}{\log(1+\alpha_i)}) = O(\frac{\log(1+\alpha_{i-1})}{\log(1+\alpha_i)})$ queries on each element, which implies that the total number of queries performed per element is in the order of $O(\sum_{i=1}^{\ell} \frac{\log(1+\alpha_{i-1})}{\log(1+\alpha_i)}) = O(\sum_{i=1}^{\ell} \frac{\log(\exp(\log k\cdot e^{-i+1}))}{\log(\exp(\log k\cdot e^{-i}))}) = e\ell = O(\log\log n)$. The refined threshold decreasing procedure has $e^2/\varepsilon = O(1/\varepsilon)$ iterations in total, while each iteration requires constant number of queries per element. The proof is complete. □

2.2 Review and comparisons with existing methods

In the following we briefly review the threshold greedy algorithm and the algorithm in [31]. Due to space limitation, We defer the overview of the lazy greedy heuristic and stochastic greedy algorithm to Appendix B.2.

Threshold Greedy Algorithm [4]. The decreasing threshold algorithm utilizes a series of geometrically decreasing thresholds to select elements when its marginal increment is above the threshold. As pointed out in [4, page 3], the motivation of this algorithm can be thought of inspiring by lazy greedy, as lazy greedy can be regarded as maintaining continuously decreasing thresholds to avoid searching the maximum marginal value.

Comparison with algorithm in [31]. The main subroutine in [31], `Simple`, is in a similar spirit to the classic greedy algorithm, which continuously selects elements according to the objective value of current candidate set and parameter v, an $(1-\varepsilon)$-approximation of $f(\mathrm{OPT})$ (*i.e.*, $v \in [(1-\varepsilon)f(\mathrm{OPT}), f(\mathrm{OPT})]$). To obtain a feasible value of v, a binary search procedure is applied on $O(\varepsilon^{-1}\log n)$ predefined well-spaced guesses. In each iteration of the binary search [31], the `Simple` procedure is applied, which requires $O(\varepsilon^{-1})$ passes on the elements. Our ADT algorithm requires a constant approximation of $f(\mathrm{OPT})$. To this end, the preprocessing phase uses thresholds decreasing at a speed that is related to the numerical output in last iteration, and update the estimations of $f(\mathrm{OPT})$ according to objective values of $\lfloor \log(\bar{w}_q/\underline{w}_q)/\log(1+\alpha_q)\rfloor$ sets. For each fixed threshold, ADT only makes a single pass on the elements to get the result.

While in ADT we can obtain an interval $[\underline{w}_i, \bar{w}_{i+1}]$ that contains $f(\mathrm{OPT})$ and shrinks at each iteration, the algorithm in [31] is not able to obtain estimates of $f(\mathrm{OPT})$ until the whole binary search procedure terminates, as it requires the value of all the historic midpoints during the binary search.

3 Complexity lower bound for cardinality constraint and beyond

Besides the aforementioned $O(n\cdot\max\{\varepsilon^{-1}, \log\log n\})$ upper bound, we show a $o(n/\log n)$ lower bound for $k \in [n/2]$ in Appendix B.3. A widely used approach to show time complexity lower bounds for deterministic algorithms is to figure out a set of hard instances on which, for any deterministic algorithm, there always exists a specific instance to prevent the algorithm from finding the desired answer efficiently. Theorem 2 indicates that if the aforementioned instance set satisfies the following property, termed *scale-free*, then the deterministic complexity lower bounds can be extended to randomized setting, with a slight increment (arbitrarily small constant) in approximation ratio.

Definition 10 (scale-free instance set)**.** *A finite instance set $\mathcal{I}$ is called* scale-free*, if there exists a measure μ defined on $\mathcal{I}$ such that*

$$\frac{\max_{I\in\mathcal{I}'} T(\mathrm{A}, I)}{\max_{I\in\mathcal{I}} T(\mathrm{A}, I)} = \Theta(1)$$

holds for any deterministic algorithm A and any set $\mathcal{I}' \subseteq \mathcal{I}$ such that $\frac{\mu(\mathcal{I}')}{\mu(\mathcal{I})} = \mu(\mathcal{I}') = \Theta(1)$, where $T(A, I)$ denotes the time that running time of algorithm A on instance I.

We present the proof of Theorem 2 in Appendix B.4, based on which we further develop the

following lower bound for USM. We remark that the concept of scale free is potentially useful due to the symmetry of hard instance sets.

Query complexity of USM. Many well-known problems including *maximum facility location*, *max-cut* are generalized by USM. Buchbinder et al. [9] presented the optimal linear time *double greedy* algorithm that achieves 1/2-approximation for USM, we further show that the time complexity is optimal. Recall that for a complete bipartite directed graph on (C, D), the size of the cut induced by $S \subseteq E = C \cup D$ is equal to $f_C(S) = |S \cap C| \cdot |\bar{S} \cap D|$, which is a submodular function with respect to S. Let μ be the uniform distribution over $\mathcal{C} = \{C \subseteq E || C| = n/2\}$. Via similar adversary arguments as that for the cardinality constraint and monotone objective function, we show that $o(n/\log n)$ queries are insufficient for deterministic algorithms to determine whether functions in $\{f_C(\cdot)\}_{C \in \mathcal{C}'}$ that has an objective value no less than 1/4 times optimal or not, as long as $\mu(\mathcal{C}') = \Theta(1)$. Combining with the fact that a uniform random set of E achieves an approximation ratio of 1/4, which generalizes the performance of a random cut in Max Di-Cut, we obtain the complete characterization of the query complexity of USM. It is worth noting that there is a matching $o(n/\log n)$ upper bound for the instance above, based on the fact about retrieving a binary vector under certain conditions established in [44]. The complete proof of Theorem 3 is deferred to Appendix B.5.

4 Intersection of p-System and d Knapsack Constraints

In this section we consider the problem of maximizing a monotone submodular function under constraint $\mathcal{I} = (\cap_{i=1}^{d} \mathcal{K}_i) \cap \mathcal{I}_p$, the intersection of a p system constraint and d knapsack constraints. Here the p system constraint is denoted by $\mathcal{I}_p$ and we use $\mathcal{K}_i = \{S \subseteq E \mid c_i(S) \leq 1\}$ $(\forall i \in [d])$ to represent the i-th knapsack constraint. Element weights in the i-th dimension are specified by weight function $c_i(\cdot) : 2^E \to \mathbb{R}^+$. Without loss of generality, we assume that all the knapsack constraints have a unit budget constraint.

Backtracking threshold and main algorithm. As shown in Appendix C.2, in backtracking threshold algorithm (Algorithm 4) we maintain a threshold Δ for element selection, which will be decreased by a multiplicative factor at the end of each iteration. To start with, we eliminate elements with high cost that are collected by set B. Among the remaining elements, those with marginal gain no less than Δ and profit density no less than the predetermined threshold θ, will be added into the candidate set, as long as the newly constructed set is feasible. When the cost of the chosen element e is larger than the residual budget, a new feasible solution $\widetilde{S}$ is constructed. The element with largest total cost in set S and element e are added into $\widetilde{S}$, we next add remaining elements in S into $\widetilde{S}$ until exceeding the budget. At the end of Algorithm 4, we return the best candidate solution set constructed.

Our main algorithm (Algorithm 3) takes the output β of Algorithm 5 as input and obtains the final solution by feeding a series of well-spaced parameters, which are related to β, to the BT algorithm. In Algorithm 5 we combine ADT and BT to obtain a constant approximation of $f(\mathrm{OPT})$. The main algorithm and Algorithm 5 are presented in Appendix C.1 and Appendix C.3 respectively.

Performance Analysis. For each element $e \in E$, it is called a *large element* if $2c_i(e) > 1$ holds for at least one index $i \in [d]$, otherwise we call it a small element. Let $B \subseteq E$ denote the set of large elements in the ground set. A simple but crucial consequence is the following upper bound on the number of large elements in OPT. We defer the proof to Appendix C.4.

Corollary 11. *The optimal solution* OPT *contains at most* $|\mathrm{OPT} \cap B| \leq d$ *large elements.*

Let $e^\flat$ be the element that is not added to set S due to violation of some knapsack constraints, *i.e.*, $e^\flat$ activates the *else* loop in line 11 of Algorithm 4. We remark that element $e^\flat$ in Algorithm 4 may not exist, hence we divide the analysis of backtracking threshold algorithm into two cases in the following propositions, based on the existence of $e^\flat$. The proofs of Proposition 12 and 13 are presented in Appendix C.5 and Appendix C.6 respectively. We defer the proof of Theorem 4 to Appendix C.9.

Proposition 12. $3f(S^*) \geq 2\theta$ *holds when element* $e^\flat$ *exists.*

Proposition 13. *If element* $e^\flat$ *does not exist,* $f(S^*) \geq \frac{f(\mathrm{OPT})-\theta(d-|\mathrm{OPT}\cap B|/2)}{p+|\mathrm{OPT}\cap B|+1} - (p\varepsilon+\varepsilon)\cdot f(\mathrm{OPT})$.

5 Efficient Algorithms for Packing Constraint

In this section, we study the packing constrained submodular maximization problem [41, 4], in which we have d-dimensional capacity vectors $\mathbf{b} \in \mathbb{R}_+^d$ and items $e \in E$ with cost vector $\mathbf{c}(e) \in \mathbb{R}_+^d$. The objective is to select a set $S \subseteq E$ to maximize submodular function $f(\cdot)$ while satisfying the packing constraints $\mathbf{A}\mathbf{x}_S \leq \mathbf{b}$, where $\mathbf{x}_S$ denotes for the characteristic vector of the set S.

We first present an efficient algorithm with approximation $1/(\frac{7}{4}d+\frac{9}{10}+\frac{9}{120d+16})-O(\varepsilon)$ for general packing constraint in Appendix D.1, which has a time complexity of $O(n\cdot\max\{\varepsilon^{-1},\log\log n\})$ and a constant approximation ratio gap to the lower bound of $\frac{1}{d^{1-\epsilon}}$ [18] for constant d. Specific cases of single knapsack constraint and binary packing constraint are investigated in Section 5.1 and Section 5.2.

5.1 An Improved Streaming Algorithm for A Single Knapsack Constraint

Our improved solution for a single knapsack constraint consists of Algorithm 2, a backtracking algorithm utilizing multiple thresholds, and Algorithm 7, which takes Algorithm 2 as an input subroutine to output the final solution. In the following we give an overview of these two procedures. Algorithm 7 is deferred to Appendix D.2.

Overview of the two subroutines. We run two threads in parallel in Algorithm 2, the *double threshold backtracking algorithm*, where each thread contains two sequential stages. In stage j of thread i ($i, j \in [2]$), we select elements whose profit density is no less than the threshold $\tau_j^{(i)}$ and cost no more than $1/i$ (this constraint is trivial in the first thread). Similar to our previous approaches, we use the backtracking algorithm to recursively construct a new candidate solution if there exists constraint violation in the proceeding procedure. In the end, we output the best solution among the collection of sets obtained in the aforementioned two threads.

In our main algorithm, Algorithm 7, we first select a singleton with both function value and cost close to that of the element with largest cost in E, then Algorithm 2 is utilized to solve the

corresponding residual problem and compute the final solution set.

Algorithm 2: Double Threshold Backtracking Algorithm

Initialization: $q \leftarrow 2$, $T_0^{(i)}, T_1^{(i)}, T_2^{(i)}, \widetilde{T}^{(i)} \leftarrow \varnothing$ $(\forall i \in [q])$
for $i \in [q]$ **do**
  for $j \in [q]$ **do**
    $T_j^{(i)} \leftarrow T_{j-1}^{(i)}$, $c_1^{(i)} \leftarrow \frac{3i}{3+i}$, $c_2^{(i)} \leftarrow \frac{9}{(3+i)^2}$, $\tau_j^{(i)} \leftarrow \lambda \cdot c_j^{(i)} (j \in [q])$
    for *each* $e \in E \setminus T_j^{(i)}$ **do**
      if $f(T_j^{(i)} + e) - f(T_j^{(i)}) \geq c(e) \cdot \tau_j^{(i)}$ *and* $c(e) \leq 1/i$ **then**
        if $c(T_j^{(i)} + e) \leq 1$ **then**
          $T_j^{(i)} \leftarrow T_j^{(i)} + e$
        else
          $\hat{e}_j^{(i)} \leftarrow e$, $\widetilde{T}_j^{(i)} \leftarrow \widetilde{T}_j^{(i)} + e$
          for $e \in T_j^{(i)}$ **do**
            if $c(\widetilde{T}_j^{(i)} + e) \leq 1$ **then**
              $\widetilde{T}_j^{(i)} \leftarrow \widetilde{T}_j^{(i)} + e$
            else
              break
  for $i \in [q]$ **do**
    $\tilde{e}_1^{(i)} \leftarrow \operatorname{argmax}_{e \in T^{(i)}} c(e)$, $\tilde{e}_2^{(i)} \leftarrow \operatorname{argmax}_{e \in T_2^{(i)} \setminus T_1^{(i)}} c(e)$
    $T^{(i)} \leftarrow \cup_{j \in [q]} T_j^{(i)}$, $U_1^{(i)} \leftarrow \{\hat{e}^{(i)}, \tilde{e}_1^{(i)}\}$, $U_2^{(i)} \leftarrow \{\hat{e}^{(i)}, \tilde{e}_2^{(i)}\}$, $U_3^{(i)} \leftarrow \{\hat{e}^{(i)}, \tilde{e}_2^{(i)}\} \cup T_1^{(i)}$
Return $S^* \leftarrow \operatorname{argmax} \{f(S) | S \in \{U_\ell^{(i)}\}_{1 \leq \ell \leq 3} \cup \{T^{(i)}, \widetilde{T}^{(i)}\}_{i=1,2}, c(S) \leq 1\}$

Performance analysis. We give the following lower bound on $f(S^*)$. The proof of Theorem 5 is given in Appendix D.4.

Lemma 14. *For set S^* returned by Algorithm 2, its objective value satisfies*

$$f(S^*) \geq \frac{7}{16} f(\mathrm{OPT}) \cdot \mathbb{1}_{\mathrm{OPT} \cap B = \varnothing} + \frac{16}{25} [f(\mathrm{OPT}) - f(O \cap B)] \cdot \mathbb{1}_{\mathrm{OPT} \cap B \neq \varnothing} - O(\varepsilon \cdot f(\mathrm{OPT}))$$

Proof sketch. We divide our analysis into three cases, according to the existence of budget violation in each stage and iteration. If Algorithm 2 stops at $\tau_1^{(i)}$, we prove that $f(S^*) \geq \max\{f(T_1), f(\hat{e}_1^{(i)}, \tilde{e}_1^{(i)}), f(\widetilde{T}_1)\} \geq \frac{2}{3}\tau_1^{(i)}$. If Algorithm 2 stops at $\tau_2^{(i)}$, objective values of sets $T_2, \widetilde{T}_2, T_1 \cup \{\hat{e}_2^{(i)}, \tilde{e}_2^{(i)}\}$ provides a lower bound of $(f(\mathrm{OPT} \setminus B) - c(\mathrm{OPT} \setminus B) \cdot \tau_1^{(i)} + \frac{4}{9}\tau_2^{(i)}) \cdot \mathbb{1}_{c(T_1) \leq \frac{1}{3}} + (\frac{\tau_1}{3} + \frac{4}{9}\tau_2) \cdot \mathbb{1}_{c(T_1) \geq \frac{1}{3}}$ on $f(S^*)$. For the third case when Algorithm 2 stops without exceeding the budget, we have $f(S^*) \geq f(\mathrm{OPT} \setminus B) - \tau_1^{(i)} \cdot c(\mathrm{OPT} \setminus B)$. Combining the three cases above, we can obtain Lemma 14. See Appendix D.3 for a complete proof.

5.2 A Near Linear Time $(1/2 - \varepsilon)$-Approximation Deterministic Algorithm for Binary Packing Constraints

We start with the formal definition about the residual problem with respect to a given set T.

Definition 15 (T-Residual Problem)**.** *Let f be a submodular function, its contracted function $f_T : 2^{E\setminus T} \to \mathbb{R}_+$ is given as $f_T(S) = f(S\cup T)-f(T)$. For the optimization problem $\max_{S\subseteq E} f(S)$, we define its T-residual problem as $\max_{S\in\mathcal{I}_T} f_T(S)$, where the constraint set $\mathcal{I}_T = \{S|S \subseteq E\setminus T, S\cup T \in \mathcal{I}\}$.*

In several constrained submodular maximization problems [37, 63, 40, 19, 4], we are able to obtain a desirable approximation guarantee for the residual problem, by carefully choosing set T. For example, in the single knapsack constraint [63], T represents the collection of three elements that have the largest marginal increments, while T consists of elements with high costs for constant number of knapsack constraints [40]. However, directly searching set T takes $O(n^3)$ and $\Theta(n^d)$ time respectively in the aforementioned two examples, which are computationally expensive. Apart from the aforementioned approaches, we investigate the residual problem with respect to $T^\sharp$, the shadow set of the target set T. The formal definition of shadow set is specified as follows.

Definition 16 ((α,β)-shadow set)**.** *$S^\sharp$ is called (α,β)-shadow set of $S \subseteq E$ iff $\mathrm{OPT}\setminus(S\cup S^\sharp)$ is a feasible solution to $S^\sharp$-residual problem, while $f((\mathrm{OPT}\cup S^\sharp)\setminus S) + \beta\cdot f(S^\sharp) \geq f(\mathrm{OPT})$ and $f(S^\sharp) \geq \alpha f(S)$.*

The first inequality in Definition 16 states that replacing set $O\cap S$ with $O\cap S^\sharp$ will incur an additive loss that is no more than $\beta\cdot f(S^\sharp)$. We have the following relationship between the approximation guarantee of residual problem and that of the original problem. The proof is presented in Appendix D.5.

Claim 17. *A γ-approximate polynomial time algorithm for the $U^\sharp$-residual problem implies a polynomial time $\min\{\gamma, \frac{1}{1+\beta}\}$-approximate algorithm.*

5.2.1 Deterministic algorithm for binary packing constraint

Overview of Algorithm 8. Without loss of generality, we assume that elements in OPT are in greedy ordering, *i.e.*, $o_{i+1} = \mathrm{argmax}_{e\in\mathrm{OPT}}\{f(\mathrm{OPT}_i + e) - f(O_i)\}$. Let $\mathrm{OPT}_i = \{o_1, o_2, \ldots, o_i\}$, $\Delta_i = f(\mathrm{OPT}_i) - f(\mathrm{OPT}_{i-1})$ and $i_\varepsilon = \max\{i|\Delta_i \geq \frac{\varepsilon}{d}\cdot f(\mathrm{OPT})\}$. As shown in Appendix D.6, in Algorithm 8 we first construct $S^\sharp_\varepsilon$, a $(\frac{1}{2}-\varepsilon, 1+\varepsilon)$-shadow set of $\mathrm{OPT}_{i_\varepsilon}$, *i.e.*, for each element in $\mathrm{OPT}_{i_\varepsilon}$, an element with comparable cost and similar marginal increment is selected. We next consider the residual problem $\max_{S\subseteq E\setminus S^\sharp}\{f(S\cup S^\sharp)|S\subseteq E\setminus S^\sharp, \mathbf{A}\mathbf{x}_{S\cup S^\sharp} \leq \mathbf{b}\}$, for which we combine the MWU-based greedy [2] algorithm with a threshold decreasing procedure on $E\setminus(E_\Gamma\cup S^\sharp)$, where $E_\Gamma = \{e\in E|\exists i\in\Gamma$ such that $c_i(e)\neq 0\}$ and $\Gamma = \{i\in[d]|b_i - c_i(S^\sharp) \leq W = \frac{2\log d}{\delta^2}\}\subseteq[d]$.

Performance analysis. The proof of Lemma 18 and Proposition 19 are presented in Appendix and D.7 and Appendix D.8 respectively.

Lemma 18. *$S^\sharp$ is a $(1/2-\varepsilon, 1+\varepsilon)$-shadow set of $\mathrm{OPT}_{i_\varepsilon}$,* i.e., *$f(\mathrm{OPT}) - f((\mathrm{OPT}\cup S^\sharp)\setminus\mathrm{OPT}_{i_\varepsilon}) \leq (1+\varepsilon)f(S^\sharp)$ and $f(S^\sharp) \geq (1/2-\varepsilon)f(\mathrm{OPT}_{i_\varepsilon})$.*

Proposition 19. *Algorithm 8 returns a solution set S_o in $O_\varepsilon(n\log\log n)$ time, for which we have $f(S_o) \geq (\frac{1}{2}-\varepsilon)f(\mathrm{OPT})$.*

Acknowledgement

The authors would like to thank Moran Feldman for valuable suggestions on this paper.

References

[1] Shipra Agrawal, Mohammad Shadravan, and Cliff Stein. Submodular secretary problem with shortlists. In *ITCS*, pages 1:1–1:19, 2019.

[2] Yossi Azar and Iftah Gamzu. Efficient submodular function maximization under linear packing constraints. In *ICALP*, pages 38–50, 2012.

[3] Ashwinkumar Badanidiyuru, Baharan Mirzasoleiman, Amin Karbasi, and Andreas Krause. Streaming submodular maximization: Massive data summarization on the fly. In *KDD*, pages 671–680, 2014.

[4] Ashwinkumar Badanidiyuru and Jan Vondrák. Fast algorithms for maximizing submodular functions. In *SODA*, pages 1497–1514, 2014.

[5] Rafael da Ponte Barbosa, Alina Ene, Huy L Nguyen, and Justin Ward. A new framework for distributed submodular maximization. In *FOCS*, pages 645–654, 2016.

[6] Sébastien Bubeck et al. Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 8(3-4):231–357, 2015.

[7] Niv Buchbinder and Moran Feldman. Deterministic algorithms for submodular maximization problems. In *SODA*, pages 392–403, 2016.

[8] Niv Buchbinder, Moran Feldman, and Mohit Garg. Deterministic $(1/2+ \varepsilon)$-approximation for submodular maximization over a matroid. In *SODA*, pages 241–254, 2019.

[9] Niv Buchbinder, Moran Feldman, Joseph Seffi Naor, and Roy Schwartz. A tight linear time (1/2)-approximation for unconstrained submodular maximization. In *FOCS*, pages 649–658, 2012.

[10] Niv Buchbinder, Moran Feldman, and Roy Schwartz. Comparing apples and oranges: query tradeoff in submodular maximization. In *SODA*, pages 1149–1168, 2015.

[11] Niv Buchbinder, Moran Feldman, and Roy Schwartz. Online submodular maximization with preemption. In *SODA*, pages 1202–1216, 2015.

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

[13] Amit Chakrabarti and Sagar Kale. Submodular maximization meets streaming: Matchings, matroids, and more. *Mathematical Programming*, 154(1-2):225–247, 2015.

[14] T-H. Hubert Chan, Zhiyi Huang, Shaofeng H.-C. Jiang, Ning Kang, and Zhihao Gavin Tang. Online submodular maximization with free disposal: Randomization beats 1/4 for partition matroids. In *SODA*, pages 1204–1223, 2017.

[15] Chandra Chekuri, Shalmoli Gupta, and Kent Quanrud. Streaming algorithms for submodular function maximization. In *ICALP*, pages 318–330, 2015.

[16] Chandra Chekuri, TS Jayram, and Jan Vondrák. On multiplicative weight updates for concave and submodular function maximization. In *ITCS*, pages 201–210. ACM, 2015.

[17] Chandra Chekuri, Jan Vondrak, and Rico Zenklusen. Dependent randomized rounding via exchange properties of combinatorial structures. In *FOCS*, pages 575–584, 2010.

[18] Brian C Dean, Michel X Goemans, and Jan Vondrák. Adaptivity and approximation for stochastic packing problems. In *SODA*, pages 395–404, 2005.

[19] Alina Ene and Huy L. Nguyen. A Nearly-Linear Time Algorithm for Submodular Maximization with a Knapsack Constraint. In *ICALP 2019*, pages 53:1–53:12, 2019.

[20] Alina Ene and Huy L. Nguyen. Towards Nearly-Linear Time Algorithms for Submodular Maximization with a Matroid Constraint. In *ICALP 2019*, pages 54:1–54:14, 2019.

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

[22] Uriel Feige, Vahab S. Mirrokni, and Jan Vondrak. Maximizing non-monotone submodular functions. In *FOCS*, pages 323–332, 2007.

[23] Moran Feldman. Personal communication. 2018.

[24] Moran Feldman, Christopher Harshaw, and Amin Karbasi. Greed is good: Near-optimal submodular maximization via greedy optimization. In *COLT*, pages 758–784, 2017.

[25] Moran Feldman, Amin Karbasi, and Ehsan Kazemi. Do less, get more: Streaming submodular maximization with subsampling. In *NeurIPS*, pages 730–740, 2018.

[26] Moran Feldman, Joseph Naor, and Roy Schwartz. A unified continuous greedy algorithm for submodular maximization. In *FOCS*, pages 570–579, 2011.

[27] Yuval Filmus and Justin Ward. A tight combinatorial algorithm for submodular maximization subject to a matroid constraint. In *FOCS*, pages 659–668, 2012.

[28] Shayan Oveis Gharan and Jan Vondrák. Submodular maximization by simulated annealing. In *SODA*, pages 1098–1116, 2011.

[29] Andrei Graur, Tristan Pollner, Vidhya Ramaswamy, and S Matthew Weinberg. New query lower bounds for submodular function minimization. In *ITCS*, 2020.

[30] Michael Gygli, Helmut Grabner, and Luc Van Gool. Video summarization by learning submodular mixtures of objectives. In *CVPR*, pages 3090–3098, 2015.

[31] Chien-Chung Huang and Naonori Kakimura. Multi-pass streaming algorithms for monotone submodular function maximization. *arXiv preprint arXiv:1802.06212*, 2018.

[32] Chien Chung Huang, Naonori Kakimura, and Yuichi Yoshida. Streaming algorithms for maximizing monotone submodular functions under a knapsack constraint. In *APPROX*, page 11, 2017.

[33] Chien-Chung Huang, Naonori Kakimura, and Yuichi Yoshida. Streaming algorithms for maximizing monotone submodular functions under a knapsack constraint. *Algorithmica*, pages 1–27, 2019.

[34] Stratis Ioannidis and Edmund Yeh. Adaptive caching networks with optimality guarantees. *ACM SIGMETRICS Performance Evaluation Review*, 44(1):113–124, 2016.

[35] David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *KDD*, pages 137–146, 2003.

[36] David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. *Theory of Computing*, 11(1):105–147, 2015.

[37] Samir Khuller, Anna Moss, and Joseph Seffi Naor. The budgeted maximum coverage problem. *Information Processing Letters*, 70(1):39–45, 1999.

[38] Nitish Korula, Vahab S. Mirrokni, and Morteza Zadimoghaddam. Online submodular welfare maximization: Greedy beats 1/2 in random order. *SIAM J. Comput.*, 47(3):1056–1086, 2018.

[39] Andreas Krause, Ajit Singh, and Carlos Guestrin. Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research*, 9:235–284, 2008.

[40] Ariel Kulik, Hadas Shachnai, and Tami Tamir. Maximizing submodular set functions subject to multiple linear constraints. In *SODA*, pages 545–554, 2009.

[41] Ravi Kumar, Benjamin Moseley, Sergei Vassilvitskii, and Andrea Vattani. Fast greedy algorithms in mapreduce and streaming. In *SPAA*, pages 1–10, 2013.

[42] Joohyun Lee, Kyunghan Lee, Euijin Jeong, Jaemin Jo, and Ness B. Shroff. Context-aware application scheduling in mobile systems: What will users do and not do next? In *UbiComp*, pages 1235–1246, 2016.

[43] Joohyun Lee, Kyunghan Lee, Euijin Jeong, Jaemin Jo, and Ness B. Shroff. Cas: Context-aware background application scheduling in interactive mobile systems. *IEEE Journal on Selected Areas in Communications*, 35(5):1013–1029, 2017.

[44] Vsevolod F Lev and Raphael Yuster. On the size of dissociated bases. *The Electronic Journal of Combinatorics*, 18(1):117, 2011.

[45] Wenxin Li and Ness Shroff. Towards practical constrained monotone submodular maximization. *arXiv preprint arXiv:1804.08178*, 2018.

[46] Hui Lin and Jeff Bilmes. Multi-document summarization via budgeted maximization of submodular functions. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics(NAACL/HLT-2010)*, pages 912–920, 2010.

[47] Hui Lin and Jeff Bilmes. A class of submodular functions for document summarization. In *HLT*, pages 510–520, 2011.

[48] Yajing Liu, Edwin K. P. Chong, and Ali Pezeshki. Improved bounds for the greedy strategy in optimization problems with curvature. *J. Comb. Optim.*, 37(4):1126–1149, 2019.

[49] Yajing Liu, Edwin K. P. Chong, Ali Pezeshki, and William Moran. Bounds for approximate dynamic programming based on string optimization and curvature. In *CDC*, pages 6653–6658, 2014.

[50] Yajing Liu, Zhenliang Zhang, Edwin K. P. Chong, and Ali Pezeshki. Performance bounds with curvature for batched greedy optimization. *J. Optimization Theory and Applications*, 177(2):535–562, 2018.

[51] Michel Minoux. Accelerated greedy algorithms for maximizing submodular set functions. *Optimization techniques*, pages 234–243, 1978.

[52] Vahab S. Mirrokni and Morteza Zadimoghaddam. Randomized composable core-sets for distributed submodular maximization. In *STOC*, pages 153–162, 2015.

[53] Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, and Amin Karbasi. Fast constrained submodular maximization: Personalized data summarization. In *ICML*, pages 1358–1367, 2016.

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

[55] Baharan Mirzasoleiman, Amin Karbasi, Rik Sarkar, and Andreas Krause. Distributed submodular maximization. *Journal of Machine Learning Research.*, 17(1):8330–8373, 2016.

[56] Eyal Mizrachi, Roy Schwartz, Joachim Spoerhase, and Sumedha Uniyal. A tight approximation for submodular maximization with mixed packing and covering constraints. In *ICALP 2019*, pages 85:1–85:15, 2019.

[57] George L. Nemhauser and Laurence A. Wolsey. Best algorithms for approximating the maximum of a submodular set function. *Mathematics of operations research*, 3(3):177–188, 1978.

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

[59] Arkadii S. Nemirovsky and David B. Yudin. Problem complexity and method efficiency in optimization. 1983.

[60] Yurii Nesterov. *Lectures on convex optimization*, volume 137. Springer, 2018.

[61] Ashkan Norouzi-Fard, Jakub Tarnawski, Slobodan Mitrovic, Amir Zandieh, Aidasadat Mousavifar, and Ola Svensson. Beyond 1/2-approximation for submodular maximization on massive data streams. In *ICML*, pages 3826–3835, 2018.

[62] Tasuku Soma and Yuichi Yoshida. Maximizing monotone submodular functions over the integer lattice. *Mathematical Programming*, 172(1-2):539–563, 2018.

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

[64] Kai Wei, Rishabh Iyer, and Jeff Bilmes. Fast multi-stage submodular maximization. In *ICML*, pages 1494–1502, 2014.

[65] Laurence A. Wolsey. Maximising real-valued submodular functions: Primal and dual heuristics for location problems. *Mathematics of Operations Research*, 7(3):410–425, 1982.

[66] Andrew Chi-Chin Yao. Probabilistic computations: Toward a unified measure of complexity. In *FOCS*, pages 222–227, 1977.

[67] Yuichi Yoshida. Maximizing a monotone submodular function with a bounded curvature under a knapsack constraint. *SIAM Journal on Discrete Mathematics*, 33(3):1452–1471, 2019.

[68] Yisong Yue and Carlos Guestrin. Linear submodular bandits and their application to diversified retrieval. In *NeurIPS*, pages 2483–2491, 2011.

[69] Zhenliang Zhang, Edwin K. P. Chong, Ali Pezeshki, William Moran, and Stephen D. Howard. Submodularity and optimality of fusion rules in balanced binary relay trees. In *CDC*, pages 3802–3807.

[70] Zhenliang Zhang, Edwin K.P. Chong, Ali Pezeshki, and William Moran. String submodular functions with curvature constraints. *IEEE Transactions on Automatic Control*, 61(3):601–616, 2016.

[71] Zhenliang Zhang, Zengfu Wang, Edwin K. P. Chong, Ali Pezeshki, and William Moran. Near optimality of greedy strategies for string submodular functions with forward and backward curvature constraints. In *CDC*, pages 5156–5161, 2013.

[72] Zizhan Zheng and Ness B. Shroff. Submodular utility maximization for deadline constrained data collection in sensor networks. *IEEE Transactions on Automatic Control*, 59(9):2400–2412, 2014.

A Additional Related Work

There is a large body of literature on submodular maximization [54, 10, 48, 50, 49, 71, 69, 65], here we mention only a few which is most relevant to our work. Besides the aforementioned results, there are also some other well known results towards more practical algorithm design. For the simple cardinality constraint, Mirzasoleiman et al. [54] and Buchbinder et al. [10] proposed the stochastic greedy algorithm, a randomized algorithm achieving the $(1 - 1/e - \varepsilon)$ approximation ratio using $O(n \log(1/\varepsilon))$ value queries in total. For the general matroid constraint, Badanidiyuru and Vondrák [4] proposed an accelerated continuous greedy algorithm which uses $O((nr/\varepsilon^4) \cdot \log(n/\varepsilon))$ value oracle queries, together with $O((\log(n/\varepsilon))/\varepsilon^2 + (r^2/\varepsilon))$ matroid independence queries. Buchbinder et al. [10] designed a faster algorithm that achieves the approximation ratio of $(1 - 1/e - \varepsilon)$ by performing $O(k\lambda + \frac{kn}{\lambda\varepsilon^5} \log^2(\frac{n}{\varepsilon}))$ value oracle queries and $O(\frac{k^2}{\varepsilon} + \frac{\lambda n}{\varepsilon^2} \log(\frac{n}{\varepsilon}))$ independence oracle queries, where parameter $\lambda \in [1, k]$ characterizes the tradeoff between the number of value oracle query and independence query. A $(1/e - \varepsilon)$-approximate algorithm for cardinality constrained non-monotone submodular maximizing problem is presented [10], which requires $O((n/\varepsilon) \cdot \log(1/\varepsilon))$ function evaluations. For a single knapsack constraint, Wolsey [65]

proposed a modified greedy algorithm with an approximation ratio of $1 - e^{-\beta} \approx 0.35$, where β denotes the root of equation $e^x = 2 - x$.

B Supplementary Materials of Section 2

B.1 Proof of Theorem 1

Proof: Let set $U = \{u_1, u_2, \ldots, u_{|U|}\}$ be the output of Algorithm 1 and $U^{(i)} = \{u_1, u_2, \ldots, u_i\}$ ($U^{(0)} = \varnothing$), where u_i represents the i-th element being added into U and $|U| \leq k$. We first make the observation that

$$f(U^{(1)}) \geq (1 - \varepsilon) \cdot \frac{f(\mathrm{OPT})}{k}. \tag{2}$$

Since τ is initialized to $\tau_0 = \frac{e\underline{w}_\ell}{k} \geq \frac{f(\mathrm{OPT})}{k}$, there must exist one iteration in which $\tau \in [\frac{f(\mathrm{OPT})}{k} - \frac{\varepsilon\underline{\omega}_\ell}{k}, \frac{f(\mathrm{OPT})}{k}]$. We argue that set U in this iteration cannot be empty, otherwise the singleton with maximum function value should be added in the candidate solution no later than this iteration. Therefore inequality (2) holds.

We use $\mathcal{G}$ to represent the set of possible values of threshold τ in the refined threshold decreasing procedure, according to line 12-16 of Algorithm 1, we have

$$\mathcal{G} = \Big\{\frac{e\underline{w}_\ell}{k} - i \cdot \frac{\varepsilon\underline{\omega}_\ell}{k} \Big| i \in \mathbb{Z}_{\geq 0}\Big\} \bigcap \Big[\frac{\underline{w}_\ell}{ek}, \frac{e\underline{w}_\ell}{k}\Big]. \tag{3}$$

Let $\tau_{\min} = \min\{\tau \in \mathcal{G} : \tau \geq \frac{f(\mathrm{OPT})}{ek}\}$ be the minimum element in $\mathcal{G}$ that is no less than $\frac{f(\mathrm{OPT}*)}{ek}$, then based on the definition of $\mathcal{G}$ and $\tau_{\min}$, we have

$$\frac{f(\mathrm{OPT})}{ek} \leq \tau_{\min} \leq \frac{f(\mathrm{OPT})}{ek} + \frac{\varepsilon\underline{\omega}_\ell}{k}.$$

Without loss of generality, we can assume that at the end of the iteration when $\tau = \tau_{\min}$, there are k elements in U. Otherwise for the solution $U' \subseteq U$ obtained at the end of this iteration, we have $f(U' + e) - f(U') < \tau_{\min}$ for $\forall e \in E \setminus U'$. Consequently we have $f(\mathrm{OPT}) - f(U') \leq \sum_{e \in \mathrm{OPT}} [f(U' + e) - f(U')] \leq |\mathrm{OPT}| \cdot \tau_{\min} \leq \frac{f(\mathrm{OPT})}{e} + \varepsilon\underline{w}_\ell$. and $f(U) \geq f(U') \geq (1 - 1/e) \cdot f(\mathrm{OPT}) - \varepsilon\underline{w}_\ell \geq (1 - 1/e - \varepsilon) \cdot f(\mathrm{OPT})$, which already implies the approximation ratio of $(1 - 1/e - \varepsilon)$.

Now we are ready to bound the function value of $f(U) = f(U^{(k)})$. We first claim the following recursion inequality,

$$f(U^{(i+1)}) - f(U^{(i)}) \geq \frac{f(\mathrm{OPT}) - f(U^{(i)})}{k} - \frac{\varepsilon}{k} f(\mathrm{OPT}). \tag{4}$$

To see this fact, observe that elements in $E \setminus U^{(i)}$ are not added into the candidate set when the threshold is equal to $(\tau^{(i)} + \frac{\varepsilon\underline{\omega}_\ell}{k})$. Here $\tau^{(i)}$ denotes the corresponding threshold value when u_i is selected. Hence (4) holds for $i \geq 1$ since

$$\begin{aligned} f(\mathrm{OPT}) - f(U^{(i)}) &\leq \sum_{e \in \mathrm{OPT}} [f(U^{(i)} + e) - f(U^{(i)})] \leq k \cdot \Big(\tau^{(i)} + \frac{\varepsilon\underline{\omega}_\ell}{k}\Big) \\ &\leq k \cdot [f(U^{(i)} + u_i) - f(U^{(i)})] + \varepsilon f(\mathrm{OPT}). \qquad \text{(Lemma 7)} \end{aligned}$$

Inequality (4) also holds for $i = 0$, as it is essentially equivalent to (2). Similar to the analysis of standard greedy algorithm, solving the recursion inequality (4) with initial condition given by (2), we have $f(U^{(i)}) \geq [(1-(1-1/k)^i) - \varepsilon] \cdot f(\mathrm{OPT})$. As there are k elements in the final solution, then $f(U) = f(U^{(k)}) \geq (1 - 1/e - \varepsilon) \cdot f(\mathrm{OPT})$. □

B.2 Review of lazy greedy and stochastic greedy algorithm

- *Lazy Greedy Heuristic [51].* The Lazy greedy heuristic speeds up the classic greedy algorithm in practice, even though its theoretical (worst case) performance is exactly the same as greedy. For each element, the algorithm maintains an upper bound for its marginal gain, and the upper bound is updated to the exact value of the marginal increment with respect to current candidate set when the element is queried. Elements are sorted in a non-increasing value of its upper bound. In each iteration, lazy greedy continuously evaluates each element along the sorted list, until identifying an element with marginal gain that is larger than the upper bounds of remaining elements. The key idea of lazy greedy is to exploit the fact that the marginal gain of an element with respect to the candidate set is non-increasing during the iteration of the algorithm, to reduce unnecessary queries and comparisons.

- *Stochastic greedy [10, 54].* All the algorithms discussed above are deterministic, while there is also a fast randomized algorithm termed stochastic greedy. Stochastic greedy algorithm is indeed a folklore result [10] and provides an approximation guarantee only in the *expected* sense, *i.e.*, it only ensures that the average objective value of its random solution is no less than $(1 - 1/e - \varepsilon)$ times optimal, rather than being able to obtain a solution of value that is always within $(1 - 1/e - \varepsilon)$ optimal. In each iteration of the stochastic greedy, the algorithm takes the maximum marginal value in a uniformly random sampled subset, instead of the whole ground set as classic greedy. It was shown that a random set of size $O((n/k) \cdot \log(1/\varepsilon))$ is sufficient to provide the desired expected approximation ratio, which results in $O(n \cdot \log(1/\varepsilon))$ queries in total.

  On the other hand, it is also important to point out that it still remains a fundamental problem [7] to derandomize algorithms for submodular optimization. As the function is accessed via a value oracle, which makes it hard to apply standard derandomization techniques. There exists several studies working towards obtaining better performance guarantees [7, 8].

B.3 Warm up—A query lower bound for maximizing monotone objective function subject to cardinality constraint

Without loss of generality, we assume that n is even. Consider a non-negative sequence $\{a_j\}_{j\in[n]}$, where $a_j = 0$ for $j \in [\frac{n}{2}]$ and $a_j = 1$ otherwise. Among all the $N = n!$ permutations of $\{a_j\}_{j\in[n]}$, let $\sigma^{(i)}$ be the i-th ($i \in [N]$) permutation and $\sigma^{(i)}(j)$ be the value of the j-th ($j \in [n]$) element in permutation $\sigma^{(i)}$. We define a linear function $f_i(\cdot) : 2^E \to \mathbb{Z}^+$ as $f_i(S) = \sum_{j\in S} \sigma^{(i)}(j)$. The size of the codomain of f is in the order of $O(n)$, as its function value of which always lies in the interval of $[0, n/2]$.

The adversary maintains a list $\mathcal{L}$ of functions $\{f_i(\cdot)\}_{i\in[N]}$ and updates the list according to their function values on the queried set. Initially all the N functions appear in list $\mathcal{L}$. After the algorithm makes query S_i, functions with identical objective value on set S_i will be assigned to the same

sub-list. The adversary always keeps the longest sub-list and returns the associated function value as the answer to query S_i. Let $N^{(i)}$ be the number of functions after the i-th query, we have $N^{(i+1)} \geq \frac{N^{(i)}}{n}$, as there are no more than n distinct function values. Hence, the number of functions remaining on the list after Q queries is at least

$$N^{(Q)} \geq \frac{N}{n^Q} = \frac{n!}{n^Q}.$$

Suppose that the algorithm chooses set $R \subseteq E$ as its output, without loss of generality we assume that $|R| = n/2$, otherwise we can obtain the same conclusion by applying the arguments on set R', which contains R and has a size of $|R'| = n/2$. To achieve an approximation ratio of $\alpha = \Theta(1)$, the objective value of set R must satisfy $f(R) \geq \alpha \cdot f(\mathrm{OPT}) = \alpha n/2$. Furthermore, the number of such functions satisfies that

$$\Big|\Big\{f_i\Big|f_i(R) \geq \alpha n/2,\ i \in [N]\Big\}\Big| = \sum_{r=\frac{\alpha n}{2}}^{n/2} \binom{n/2}{r} \cdot \binom{n/2}{n/2-r} \cdot (n/2)! \cdot (n/2)! \geq N^{(Q)} \geq \frac{n!}{n^Q}, \tag{5}$$

from which we know that

$$n^{2Q} \geq \frac{\binom{n}{n/2}}{\sum_{r=\frac{\alpha n}{2}}^{n/2} \binom{n/2}{r}^2} \geq \frac{\binom{n}{n/2}}{n \cdot \binom{n/2}{\alpha n/2}^2}. \qquad (\alpha = 1/2 + \Theta(1))$$

This implies that $Q \geq \frac{n}{\log n}$, otherwise the algorithm will fail to distinguish the functions in the list and thus cannot provide a solution with desired approximation guarantee.

B.4 Proof of Theorem 2

Proof of Theorem 2: Let set $\mathcal{I}_s$ be a *scale-free* input instance of problem $\mathcal{P}$ and $\mathcal{A}$ be a finite set of deterministic approximation algorithms. We use $\mathcal{R}^{(\alpha)}$ to represent the set of randomized algorithms with expected approximation ratio no less than $\alpha \in [0, 1]$. Note that every randomized algorithm R could be represented as a distribution λ_R over the set of deterministic algorithms, then $\mathcal{R}(\mathcal{P}, \alpha)$, the *randomized complexity* of problem $\mathcal{P}$ over $\mathcal{R}^{(\alpha)}$, could be lower bounded as,

$$\mathcal{R}(\mathcal{P}, \alpha) = \min_{R \in \mathcal{R}^{(\alpha)}} \max_{I \in \mathcal{I}} \mathbb{E}_{\mathrm{A} \sim \lambda_R} T(\mathrm{A}, I) \tag{6}$$

$$\geq \min_{R \in \mathcal{R}^{(\alpha)}} \max_{I \in \mathcal{I}_s} \mathbb{E}_{\mathrm{A} \sim \lambda_R} T(\mathrm{A}, I), \tag{7}$$

where $\mathcal{I}$ denote the entire input set and the inequality holds because $\mathcal{I}_s \subseteq \mathcal{I}$.

Let $C_R = \max_I \mathbb{E}_{\mathrm{A} \sim \lambda_R} T(\mathrm{A}, I)$ and $S(\mathrm{A}, I)$ be the solution computed by algorithm A on instance I. According to the definition of approximation ratio, we know that the expected objective function value of the approximate solution $S(\mathrm{A}, I)$, will be no less than a factor of $(\alpha + \delta)$ times the optimal value OPT_I,

$$\mathbb{E}_{\mathrm{A} \sim \lambda_R}[f(S(\mathrm{A}, I))] \geq (\alpha + \delta) \cdot \mathrm{OPT}_I, \quad \forall I \in \mathcal{I}. \tag{8}$$

Notice that

$$\mathbb{E}_{\mathrm{A} \sim \lambda_R}[f(S(\mathrm{A}, I))] \leq \mathbb{P}\Big(f(S(\mathrm{A}, I)) \leq \alpha \mathrm{OPT}_I\Big) \cdot \Big(\alpha \mathrm{OPT}_I\Big) + \Big[1 - \mathbb{P}\Big(f(S(\mathrm{A}, I)) \leq \alpha \mathrm{OPT}_I\Big)\Big] \cdot \mathrm{OPT}_I.$$

Combining with (8), it follows that

$$\mathbb{P}_{\mathrm{A}\sim\lambda_R}\Big(f(S(\mathrm{A},I))\le\alpha\mathrm{OPT}_I\Big)\le 1-\frac{\delta}{1-\alpha},\quad\forall I\in\mathcal{I}. \tag{9}$$

We could further know that

$$\begin{aligned}
&\mathbb{E}_{\mathrm{A}\sim\lambda_R}\Big[\mathbb{1}_{\{f(S(\mathrm{A},I))\le\alpha\mathrm{OPT}_I\}\cup\{T(A,I)>\frac{2(1-\alpha)}{\delta}C_R\}}\Big]\\
\le&\mathbb{E}_{\mathrm{A}\sim\lambda_R}\Big[\mathbb{1}_{\{f(S(\mathrm{A},I))\le\alpha\mathrm{OPT}_I\}}\Big]+\mathbb{E}_{\mathrm{A}\sim\lambda_R}\Big[\mathbb{1}_{\{T(\mathrm{A},I)>\frac{2(1-\alpha)}{\delta}C_R\}}\Big]\\
\le&\Big(1-\frac{\delta}{1-\alpha}\Big)+\frac{\delta}{2(1-\alpha)}=1-\frac{\delta}{2(1-\alpha)}.
\end{aligned} \tag{10}$$

The first inequality follows from union bound; The second inequality is based on (9) and the following consequence from *Markov's inequality*,

$$\begin{aligned}
\mathbb{E}_{\mathrm{A}\sim\lambda_R}\Big[\mathbb{1}_{\{T(\mathrm{A},I)>\frac{2(1-\alpha)}{\delta}C_R\}}\Big]=&\mathbb{P}_{\mathrm{A}\sim\lambda_R}\Big[T(\mathrm{A},I)>\frac{2(1-\alpha)}{\delta}C_R\Big]\\
\le&\frac{\delta\cdot\mathbb{E}_{\mathrm{A}\sim\lambda_R}[T(A,I)]}{2(1-\alpha)C_R}\\
\le&\frac{\delta}{2(1-\alpha)},
\end{aligned}$$

where the last inequality is based on the definition of C_R. Applying *Yao's Minimax Principle* [66], we know that for $\forall\mu$, there exists a deterministic algorithm A^*_μ such that

$$\mathbb{E}_{I\sim\mu}\Big[\mathbb{1}_{\{f(S(\mathrm{A}^*_\mu,I))\le\alpha\mathrm{OPT}_I\}\cup\{T(\mathrm{A}^*_\mu,I)>\frac{2(1-\alpha)}{\delta}C_R\}}\Big]\le 1-\frac{\delta}{2(1-\alpha)}. \tag{11}$$

Now we define a new input instance set $\mathcal{I}^*_\mu\subseteq\mathcal{I}_s$ as

$$\mathcal{I}^*_\mu=\Big\{I\in\mathcal{I}_s\Big|f(S(\mathrm{A}^*_\mu,I))>\alpha\mathrm{OPT}_I,T(\mathrm{A}^*_\mu,I)\le\frac{2(1-\alpha)}{\delta}C_R\Big\}. \tag{12}$$

Based on (11), it can be seen that under probability measure μ,

$$\mathbb{P}_{I\sim\mu}\Big(I\in\mathcal{I}^*_\mu\Big)>\frac{\delta}{2(1-\alpha)}>\frac{\delta}{2}=\Theta(1). \tag{13}$$

Recall that for any deterministic algorithm A, its worst-case time complexity on $\mathcal{I}_s$ is at least $\mathcal{T}(\mathcal{P})$, combined with the fact that $\mathcal{I}_s$ is a *scale-free* set and inequality (13), we know that

$$\max_{I\in\mathcal{I}_s}T(\mathrm{A}^*_\mu,I)\in\Omega(\mathcal{T}(\mathcal{P})). \tag{14}$$

On the other hand, from the definition of $\mathcal{I}^*_\mu$, we know that every input instance in $\mathcal{I}_s$ satisfies

$$T(\mathrm{A}^*_\mu,I)\le\frac{2(1-\alpha)}{\delta}C_R,\quad\forall I\in\mathcal{I}_s. \tag{15}$$

Combining (15) and (14), C_R is then proven to be in the order of $\mathcal{T}(\mathcal{P})$. Notice that the arguments above holds for arbitrary randomized algorithm in $\mathcal{R}^{(\alpha+\delta)}$, it follows that

$$\mathcal{R}(\mathcal{P},\alpha+\delta)\in\Omega(\mathcal{T}(\mathcal{P})).$$

□

B.5 Proof of Theorem 3

Proof of the $O(n/\log n)$ lower bound in Theorem 3: Consider the following class of functions [22], which correspond to cuts in a complete bipartite directed graph on (C, D):

$$f_C(S) = |S \cap C| \cdot |\bar{S} \cap D|, \ (C \in \mathcal{C}) \tag{16}$$

where $\mathcal{C} = \{C \subseteq E || C| = n/2\}$. It can be verified that f_C is submodular for any given set $C \in \mathcal{C}$.

Suppose that the algorithm makes Q queries to $f(\cdot)$ and returns T as its solution set. We first make the observation that for $\forall T \subseteq E$, there exists a set $T' \subseteq E$ with size $\frac{n}{2}$, such that $f_C(T') \geq f_C(T)$. To see this fact, we can construct T' by combining T with set $C' \subseteq C \setminus T$ with size $|C'| = \frac{n}{2} - |T|$ if $|T| \leq \frac{n}{2}$, otherwise removing elements in $T \cap D'$, *i.e.*, let $T' = T \setminus D'$. It can be verified that under this construction $f_C(T') \geq f_C(T)$ and

$$f_C(T') = |T' \cap C|^2, \tag{17}$$

which is due to the fact that $|T'| = \frac{n}{2}$.

Let μ be the uniform distribution over $\mathcal{C}$, then we have $|\mathcal{C}'| = \Theta(\binom{n}{\frac{n}{2}})$ for any $\mathcal{C}' \subseteq \mathcal{C}$ such that $\mu(\mathcal{C}') = \Theta(1)$. Observe that if $f(T) \geq (\frac{1}{4} + \delta) \cdot \mathrm{OPT} = \frac{(1+\delta)\cdot n^2}{16}$, then we have the following lower bound on $|T' \cap C|$,

$$|T' \cap C| = \sqrt{f_C(T')} \geq \sqrt{f_C(T)} = \frac{(1 + \Theta(\delta)) \cdot n}{4} = n'.$$

Via similar arguments in Appendix B.3, we know that

$$\frac{|\mathcal{C}'|}{n^Q} \leq \sum_{i=n'}^{\frac{n}{2}} \binom{\frac{n}{2}}{i}^2,$$

which implies that the number of queries $Q \geq \frac{n}{\log n}$. It is straightforward to verify that $\{f_C(\cdot)\}_{C \in \mathcal{C}}$ is a scale free set, as the aforementioned arguments can be also applied for any instance set $\{f_C(\cdot)\}_{C \subseteq \mathcal{C}'}$, if $\mathcal{C}' \subseteq \mathcal{C}$ and $|\mathcal{C}'| = \Theta(|\mathcal{C}|)$. The proof is complete. □

Fact 20. *[44] There exists a binary matrix $\mathbf{Q} \in \{0,1\}^{q \times n}$, such that for every vector $\mathbf{s} \in \mathbb{R}^n$, equation $\mathbf{Q} \cdot \mathbf{x} = \mathbf{s}$ has at most one binary solution $\mathbf{x} \in \{0,1\}^n$, if*

$$q > (2 \log 3 + o(1))(n/\log n). \tag{18}$$

It suffices to show that $\mathbf{Q} \cdot \Delta\mathbf{x} \neq 0$ holds for every non-zero vector $\Delta\mathbf{x} \in \{-1, 0, 1\}^n$. Lev et al. [44] presented a non-constructive proof via the classic probabilistic method. Consider constructing the n binary columns of $\mathbf{Q}$ at random and independently from uniformly distribution on $\{0,1\}^q$, it can be shown that for any fixed $\Delta\mathbf{x}$, the probability that all the row vectors in $\mathbf{Q}$ are orthogonal to $\Delta\mathbf{x}$ is sufficiently small to ensure that, the summation of probabilities over all vectors $\Delta\mathbf{x} \in \{-1, 0, 1\}^n$ is strictly less than 1. The facts above imply the existence of $\mathbf{Q}$ according to union bound.

Proof of the $O(n/\log n)$ upper bound in Theorem 3: For the complete bipartite directed graph instance considered in USM above, first note that we can obtain the value of $|S \cap C|$ and

$|\bar{S} \cap D|$ according to the following equations,

$$|S \cap C| \cdot |\bar{S} \cap D| = f_C(S), \tag{19}$$

$$|\bar{S} \cap D| + |S| = |S \cap C| + |D|, \tag{20}$$

in which the solution is unique. It can be seen that as long as we can identify set C, we are able to obtain the solution for maximizing $f_C(S)$.

Let $\mathbf{x} = \mathbf{1}_C$ be the characteristic vector of set C, using the matrix implied by [44], we make queries $\{\mathbf{1}_{\mathbf{Q}^{(i)}}\}_{i\in[q]}$ to function f, where $\mathbf{Q}^{(i)}$ represents the i-th row of $\mathbf{Q}$, and we abuse the notation by using $\mathbf{1}_{\mathbf{Q}^{(i)}}$ to denote the corresponding set. Then we can obtain the product vector $\mathbf{s} = \mathbf{Q} \cdot \mathbf{1}_C$ by plugging $\mathbf{1}_{\mathbf{Q}^{(i)}} (i \in [q])$ into (19)-(20) separately, based on which we can recover the value of C. □

C Supplementary Materials of Section 4

C.1 Details of Algorithm 3

Algorithm 3: Main Algorithm for p-system+d knapsack constraints

Input: β returned by Algorithm 5, approximation parameter δ
Output: Approximate solution S_o
Initialization: $T \leftarrow \varnothing$, $\lambda \leftarrow 7(p + d + 1)\beta$
while $\lambda \geq \frac{\beta}{p+7/4d+1}$ **do**
    $T' \leftarrow$ set returned by BT($\lambda, \frac{\delta}{p+1}$)
    $T \leftarrow T \cup \{T'\}$
    $\lambda \leftarrow \lambda/(1 + \delta)$
Return $S_o \leftarrow \text{argmax}_{S\in T} f(S)$

C.2 Details of Algorithm 4

Algorithm 4: Backtracking Threshold (BT) Algorithm (θ, ε) (BT(θ, ε))

Input: Submodular function $f(\cdot)$, weight function $c_i(\cdot)$ ($i \in [d]$), threshold θ, parameter ε
Initialization: $S_0 \leftarrow \{\text{argmax}_{e \in E} f(e)\}$, $B \leftarrow \{e \in E \mid \exists i \in [d] \text{ such that } c_i(e) \geq 1/2\}$, $\Delta \leftarrow f(S_0)$, $S \leftarrow \varnothing$
while $\Delta \geq \frac{\varepsilon f(S_0)}{n}$ **do**
  for $e \in E \setminus B$ **do**
    if $f_S(e) \geq \max\{\theta \cdot \sum_{i=1}^{d} c_i(e), \Delta\}$ *and* $S + e \in \mathcal{I}_p$ **then**
      if $S + e \in \mathcal{I}$ **then**
        $S \leftarrow S + e$
      else
        $\widetilde{S} \leftarrow \{e, \text{argmax}_{e \in S} \sum_{i=1}^{d} c_i(e)\}$
        for $e \in S$ **do**
          if $\widetilde{S} + e \in \mathcal{I}$ **then**
            $\widetilde{S} \leftarrow \widetilde{S} + e$
        break
  $\Delta \leftarrow (1 - \varepsilon) \cdot \Delta$
return $S^* \leftarrow \text{argmax}_{T \in \{S, \tilde{S}, S_0\}} f(T)$

C.3 Details of Algorithm 5

Combine backtracking threshold with adaptive decreasing threshold. Similar to our treatments for cardinality constraint, we use the adaptive decreasing threshold algorithm to approximate the value of $f(\text{OPT})$ in Algorithm 5.

Algorithm 5: ADT for $\mathcal{I} = (\cap_{i=1}^{d} \mathcal{K}_i) \cap \mathcal{I}_p$

Input: Submodular function $f(\cdot) : 2^E \to \mathbb{R}^+$, Algorithm BT (θ, ε).
Output: $\beta = \underline{\omega}_\ell$
Initialization: $\underline{\omega}_1 \leftarrow \frac{3 \max_{e \in E} f(e)}{7(p+d+1)}$, $\bar{\omega}_1 \leftarrow n \cdot \max_{e \in E} f(e)$, $\ell \leftarrow \lceil \log \log n \rceil$, $\epsilon = \frac{2}{7(p+1)(p+d+1)}$
for $i = 1 : \ell$ **do**
  $\alpha_i = \exp(\log n \cdot e^{-i}) - 1$, $\lambda = \underline{\omega}_i$
  while $\lambda \leq \bar{\omega}_i$ **do**
    $S_\lambda^{(i)} \leftarrow \text{BT}(\lambda, \epsilon)$, $\lambda \leftarrow \lambda(1 + \alpha_i)$
  $\underline{\omega}_{i+1} \leftarrow \frac{3 \max_\lambda f(S_\lambda^{(i)})}{2}$, $\bar{\omega}_{i+1} \leftarrow \underline{\omega}_{i+1}(1 + \alpha_i)$
Return $\beta = \underline{\omega}_\ell$.

C.4 Proof of Corollary 11

Proof: Suppose that there are more than d large elements in OPT. From the pigeonhole principle we know that, there exists at least one index $i \in [d]$ such that $c_i(\text{OPT}) > 1$. However, this contradicts the fact that OPT is a feasible solution set, the proof is complete. □

C.5 Proof of Proposition 12

Note that we always assume that every singleton in E is a feasible solution, otherwise we can apply our algorithms on the ground set consisting of feasible singletons. Suppose that the candidate solution set S is equal to $S^\flat$ before considering $e^\flat$, hence there exists $i \in [d]$ such that $c_i(S^\flat + e^\flat) > 1$. We use $\hat{e}$ to represent the element with the largest total cost in $S^\flat$, *i.e.*, $\hat{e} = \text{argmax}_{e \in S^\flat} \sum_{i=1}^{d} c_i(e)$. Let $\widetilde{S}^{(1)} = \{e^\flat, \hat{e}\}$ and $\widetilde{S} = \widetilde{S}^{(1)} \cup \widetilde{S}^{(2)}$. Before proving Proposition 12, we first show the feasibility of the candidate solution sets involved.

Fact 21. *Both $\widetilde{S}$ and $\widetilde{S}^{(1)}$ are feasible solution sets.*

Proof: Note that $\widetilde{S}$ is initialized to $\widetilde{S}^{(1)}$ and is always a feasible set after each update, it suffices to show that $\widetilde{S}^{(1)} \in \mathcal{I}$. To this end, we first remark that $S^\flat \cup \{e^\flat\} \in \mathcal{I}_p$, since $e^\flat$ satisfies all conditions required in line 8 of Algorithm 4. According to the *down-closed* property of $\mathcal{I}_p$ and the fact that $\widetilde{S}^{(2)} \subseteq S^\flat \cup \{e^\flat\}$, we conclude that $\widetilde{S}^{(2)} \in \mathcal{I}_p$. It remains to show that $\widetilde{S}^{(2)}$ belongs to $\cap_{i=1}^{d} \mathcal{K}_i$. Recall that both $e^\flat$ and $\hat{e}$ are small elements, then for $\forall j \in [d]$ we have

$$c_j(\widetilde{S}^{(2)}) \leq c_j(e^\flat) + c_j(\hat{e}) \leq 1,$$

which implies that $\widetilde{S}^{(2)} \in \cap_{i=1}^{d} \mathcal{K}_i$ and the proof is complete. □

Proof of Proposition 12: Without loss of generality we assume that $S^\flat = \{e_1, e_2, \ldots, e_{|S^\flat|}\}$, where e_i denotes the i-th element added into $S^\flat$. Since $\widetilde{S}^{(2)}$ is also a subset of $S^\flat$, we denote it as $\widetilde{S}^{(2)} = \{e_{i_1}, e_{i_2}, \ldots, e_{i_{|\widetilde{S}^{(2)}|}}\}$, where $i_t \in [|S^\flat|]$ for $\forall t \leq |\widetilde{S}^{(2)}| \leq |S^\flat|$. We further let $S_i^\flat = \{e_1, e_2, \ldots, e_i\}$ ($i \in [|S^\flat|]$) be the first i elements in $S^\flat$, and $\widetilde{S}_i^{(2)}$ is defined in an analogous manner.

According to the density threshold rule, we have

$$f(S_{i+1}^\flat) - f(S_i^\flat) \geq \theta \cdot \sum_{j \in [d]} c_j(e_{i+1}). \tag{21}$$

We can now lower bound the objective value of $S^\flat$ as

$$\begin{aligned} f(\widetilde{S}^{(2)}) &= \sum_{j=1}^{|\widetilde{S}^{(2)}|} [f(\widetilde{S}_j^{(2)}) - f(\widetilde{S}_{j-1}^{(2)})] \\ &\geq \sum_{j=1}^{|\widetilde{S}^{(2)}|} [f(S_{i_j}^\flat) - f(S_{i_j-1}^\flat)] && \text{(submodularity)} \\ &\geq \theta \cdot \sum_{j=1}^{|\widetilde{S}^{(2)}|} \sum_{t=1}^{d} c_t(e_{i_j}) = \theta \cdot \sum_{t=1}^{d} c_j(\widetilde{S}^{(2)}). && (22) \end{aligned}$$

where the last inequality is due to (21). Similarly we can obtain

$$f(\widetilde{S}) \geq \theta \cdot \sum_{t=1}^{d} c_j(\widetilde{S}). \tag{23}$$

We next claim that $S^\flat \setminus (\widetilde{S}^{(2)} + \hat{e}) \neq \varnothing$, otherwise $\widetilde{S}$ will be equal to $S^\flat \cup \{e^\flat\}$ and this contradicts the fact that $\widetilde{S}$ is a feasible solution. Hence there exists at least one element $\bar{e} \in S^\flat \setminus (\widetilde{S}^{(2)} + \hat{e})$. We further claim that there exists at least one index $i^\dagger$, such that $c_{i^\dagger}(\widetilde{S} + \bar{e}) > 1$, otherwise we conclude that $\widetilde{S} + \bar{e} \in \cap_{i=1}^d \mathcal{K}_i$. Moreover, the fact that $\widetilde{S} + \bar{e} \subseteq S^\flat + e^\flat$ and $S^\flat + e^\flat \in \mathcal{I}_p$ implies that $\widetilde{S} + \bar{e}$ also belongs to $\mathcal{I}_p$. As a consequence, $\bar{e}$ will be added into $\tilde{S}_{\tilde{e}}$, this contradicts the fact that $\bar{e} \in S^\flat \setminus (\widetilde{S}^{(2)} + \hat{e})$. Therefore

$$\sum\nolimits_{j=1}^d c_j(\widetilde{S} + \bar{e}) \geq c_{i^+}(\widetilde{S} + \bar{e}) > 1. \tag{24}$$

By combining (22) with (24), and interchanging the order of the summation, we know that

$$f(\widetilde{S}) \geq \theta \cdot \sum_{j=1}^d c_j(\widetilde{S}) \geq \theta \cdot \Big(1 - \sum_{j=1}^d c_j(\bar{e})\Big) \geq \theta \cdot \Big(1 - \sum_{j=1}^d c_j(\hat{e})\Big), \tag{25}$$

where the first and second inequality follow from (23) and (24) respectively. The last inequality is based on the definition of $\hat{e}$. On the other hand, using similar arguments to (22) and the monotonicity of f, we know that

$$f(\widetilde{S}) \geq f(\widetilde{S}^{(1)}) \geq \theta \cdot \Big(\sum\nolimits_{j=1}^d c_j(e^\flat) + \sum\nolimits_{j=1}^d c_j(\hat{e})\Big). \tag{26}$$

Moreover, we have

$$f(S^\flat) \geq \theta \cdot \Big(\sum_{e \in S} \sum\nolimits_{j=1}^d c_j(e)\Big) \geq \theta \cdot \Big(1 - \sum\nolimits_{j=1}^d c_j(e^\flat)\Big), \tag{27}$$

where the last inequality holds because $S^\flat + e^\flat$ belongs to $\mathcal{I}_p$ but $S^\flat + e^\flat \notin \mathcal{I}$, from which we know that the total costs of $S^\flat + e^\flat$ in all the d dimensions is larger than 1. Combining (25)–(27), we are able to derive the following lower bound on the quality of output set S^*,

$$\begin{aligned} f(S^*) =& \max\{f(\widetilde{S}), f(S^\flat)\} \geq \frac{f(S^\flat) + 2f(\widetilde{S})}{3} \\ \geq& \frac{1}{3}\Big[\theta \cdot \Big(1 - \sum_{j=1}^d c_j(\hat{e})\Big) + \theta \cdot \Big(\sum_{j=1}^d c_j(\hat{e}) + \sum_{j=1}^d c_j(e^\flat)\Big) + \theta \cdot \Big(1 - \sum_{j=1}^d c_j(e^\flat)\Big)\Big] \\ =& \frac{2}{3}\theta. \end{aligned} \tag{28}$$

The proof is complete. □

C.6 Proof of Proposition 13

Proof: To prove this conclusion, we partition the optimal solution set as

$$\mathrm{OPT} = \mathrm{OPT}_1 \cup \mathrm{OPT}_2 \cup (\mathrm{OPT} \cap B),$$

where set

$$\mathrm{OPT}_1 = \Big\{e \in \mathrm{OPT} \setminus B \Big| f_{S^\flat}(e) < \theta \cdot \sum\nolimits_{j=1}^d c_i(e)\Big\} \tag{29}$$

represents the set of small elements in OPT whose marginal increment with respect to set $S^\flat$ is less than the threshold θ, set $\mathrm{OPT}_2 = \mathrm{OPT} \setminus (B \cup \mathrm{OPT}_1)$ denotes the remaining small elements in OPT. Based on this partition, we are able to lower bound $f(S)$ in the following manner,

$$\begin{aligned} f(\mathrm{OPT}) - f(S) \overset{(a)}{\leq} & f(S \cup \mathrm{OPT}) - f(S) \\ = & [f(S \cup (\mathrm{OPT} \cap B)) - f(S)] \\ & + [f(S \cup (\mathrm{OPT} \cap B) \cup \mathrm{OPT}_1) - f(S \cup (\mathrm{OPT} \cap B))] \\ & + [f(S \cup \mathrm{OPT}) - f(S \cup (\mathrm{OPT} \cap B) \cup \mathrm{OPT}_1)] \\ \overset{(b)}{\leq} & \underbrace{[f[S \cup (\mathrm{OPT} \cap B)] - f(S)]}_{R_b} + \underbrace{[f(S \cup \mathrm{OPT}_1) - f(S)]}_{R_1} \\ & + \underbrace{[f(S \cup \mathrm{OPT}_2) - f(S)]}_{R_2} \end{aligned} \tag{30}$$

where (a) is based on monotonicity of f and (b) follows from submodularity of f. We finish the proof by establishing upper bounds on the three aforementioned terms R_b, R_1, R_2 respectively.

Firstly, a direct consequence of submodularity and the definition of S^* is,

$$R_b = f(S \cup (\mathrm{OPT} \cap B)) - f(S) \leq \sum_{e \in \mathrm{OPT} \cap B} f(e) \leq |\mathrm{OPT} \cap B| \cdot f(S^*). \tag{31}$$

As for the second term R_1, it can be upper bounded in the following manner:

$$\begin{aligned} R_1 = f(S \cup \mathrm{OPT}_1) - f(S) = \sum_{e \in \mathrm{OPT}_1} [f(S+e) - f(S)] \overset{(a)}{\leq} & \theta \cdot \sum_{j=1}^{d} c_j(\mathrm{OPT}_1) \\ \overset{(b)}{\leq} & \theta \cdot \Big(d - \sum_{j=1}^{d} c_j(\mathrm{OPT} \cap B) \Big) \\ \overset{(c)}{\leq} & \theta \cdot \Big(d - \frac{|\mathrm{OPT} \cap B|}{2} \Big), \end{aligned} \tag{32}$$

where (a) is based on the definition of OPT_1 in (29), which indicates that the profit density of elements in OPT_1, its profit density is less than θ. As OPT is a feasible solution, for any $j \in [d]$, we have $c_j(\mathrm{OPT}) \leq 1$ and $c_j(\mathrm{OPT}_1) \leq c_j(\mathrm{OPT}) - c_j(\mathrm{OPT} \cap B) \leq 1 - c_j(\mathrm{OPT} \cap B)$ and (b) follows. (c) holds because the total cost of each large element is no less than $1/2$.

Assuming Proposition 22 below, which will be proved in Appendix C.7, we are able to lower bound the objective value of S.

Proposition 22. $f(S \cup \mathrm{OPT}_2) - f(S) \leq [p + (p+1)\varepsilon] \cdot f(S)$.

More specifically, by plugging inequalities (31)–(32) and Proposition 22 into (30), we have

$$\begin{aligned} f(S) \geq & f(\mathrm{OPT}) - (R_b + R_1 + R_2) \\ \geq & f(\mathrm{OPT}) - |\mathrm{OPT} \cap B| \cdot f(S^*) - \theta \cdot (d - |\mathrm{OPT} \cap B|/2) - (p + O(\varepsilon)) f(S). \end{aligned} \tag{33}$$

Rearranging the terms, we get

$$\begin{aligned}(p+1+|\mathrm{OPT}\cap B|+O(\varepsilon))f(S^*) \geq & (p+1+O(\varepsilon))f(S)+|\mathrm{OPT}\cap B|\cdot f(S^*)\\ \geq & f(\mathrm{OPT})-\theta\cdot(d-|\mathrm{OPT}\cap B|/2).\end{aligned} \tag{34}$$

The proof is complete. □

C.7 Proof of Proposition 22

Proof: An important note is that, for any element $e \in \mathrm{OPT}_2$, the reason that it cannot be added into S is either the marginal increment of e is less than $(\varepsilon \max_e f(e))/n$, or $S+e$ is not a feasible set in $\mathcal{I}_p$. Owing to this observation, we are able to bound R_2 via same arguments for the analysis of the standard greedy for p-system constraint [12, 4]. Here we provide the proof for completeness.

For $S=\{e_1,e_2,\ldots,e_s\}$ and $\forall i\in[s]$, we define set $C_i \subseteq \mathrm{OPT}_2 \setminus S$ as,

$$C_i=\Big\{e\in \mathrm{OPT}_2\setminus S\Big|S^{(i-1)}\cup\{e\}\in\mathcal{I}\Big\}, \tag{35}$$

which consists of the elements in OPT_2 that are able to be added into the candidate solution set in the i-th step. According to the down-closed property of the independent system, we know that $C_{i+1}\subseteq C_i$ and we have $C_1=\mathrm{OPT}_2\setminus S$. Consider set $Q_i=S^{(i)}\cup(C_1\setminus C_{i+1})$. On the one hand, we have $C_1\setminus C_{i+1}\in\mathcal{I}$, since it is a subset of $\mathrm{OPT}\in\mathcal{I}$, which implies that Q_i has a base of size no less than $|C_1\setminus C_{i+1}|$. On the other hand, $S^{(i)}$ is a base of Q_i since no elements in $Q_i\setminus S^{(i)}$ can be added into $S^{(i)}$ according to the definition of C_i. Then based on the definition of p-system, we know that

$$|C_1\setminus C_{i+1}|\leq p\cdot|S^{(i)}|=pi. \tag{36}$$

Now consider the procedure of decreasing threshold Δ. For $1\leq i\leq s$, we let Δ_i be the value of Δ when element e_i is added into S, then we have

$$f(S^{(i)})-f(S^{(i-1)})\geq\Delta_i. \tag{37}$$

And we further claim that

$$f(S^{(i-1)}+e)-f(S^{(i-1)})\leq\frac{\Delta_i}{1-\varepsilon}=\Delta_{i-1},\ \forall e\in C_i. \tag{38}$$

Otherwise $e\in C_i$ satisfying (38) will already be included into the candidate solution set in previous iteration since

- $S^{(i-1)}+e\in\mathcal{I}$ is a feasible set according to the definition of C_i;
- $f(S^{(i-1)}+e)-f(S^{(i-1)})\geq\theta\cdot[\sum_{i=1}^{d}c_i(e)]$ holds, which is based on the submodularity and the definition of OPT_2;
- The marginal increment of e with respect to set $S^{(i-1)}$ is no less than the threshold Δ_{i-1} according to (38).

However $e \notin S$, thus (38) is true. Using similar arguments, we obtain

$$f(S+e) - f(S) \leq \frac{\varepsilon}{n} M, \ \forall e \in C_{s+1}. \tag{39}$$

Hence we are able to show that

$$\begin{aligned}
R_2 =& f(S \cup \mathrm{OPT}_2) - f(S) && (40)\\
\leq& \sum_{e \in \mathrm{OPT}_2 \setminus S} \Big[f(S+e) - f(S) \Big] && \text{(submodularity)}\\
\overset{(a)}{=}& \sum_{i=1}^{s} \sum_{e \in C_i \setminus C_{i+1}} \Big[f(S+e) - f(S) \Big] + \sum_{e \in C_{s+1}} \Big[f(S+e) - f(S) \Big] && (41)\\
\overset{(b)}{\leq}& \frac{1}{1-\varepsilon} \sum_{i=1}^{s} |C_i \setminus C_{i+1}| \cdot \Delta_i + |C_{r+1}| \cdot \frac{\varepsilon}{n} M, && (42)
\end{aligned}$$

where (a) is based on the definition of A_i and (b) follows from inequalities (38)-(39). Observe that

- $\{\Delta_i\}_{i \in [s]}$ is a decreasing sequence;
- The total sum of sequence $\{|C_i \setminus C_{i+1}|\}_{i \in [s]}$ are fixed;
- $|C_i \setminus C_{i+1}| \leq p$, according to (36).

Hence $\sum_{i=1}^{s} |C_i \setminus C_{i+1}| \cdot \Delta_i$ achieves its maximum when $|C_i \setminus C_{i+1}| = p$. As a consequence, the following upper bound holds for the first term in (42),

$$\sum_{i=1}^{s} |C_i \setminus C_{i+1}| \cdot \Delta_i \leq \sum_{i=1}^{s} p \cdot \Delta_i \leq p \cdot f(S). \tag{43}$$

Now plugging (43) and inequality $|C_{r+1}| \cdot \frac{\varepsilon}{n} M \leq M \leq f(\mathrm{OPT})$ into (42), the proof is complete. □

C.8 Proof of Proposition 23

Algorithm 5 returns $\underline{w}_\ell$, which is shown to be a good approximation of $f(\mathrm{OPT})$ in Proposition 23.

Proposition 23. *For any $i \geq 0$, we have*

$$\frac{3f(\mathrm{OPT})}{7(p+d+1)(1+\alpha_i)} \leq \underline{w}_{i+1} \leq \frac{3}{2} f(\mathrm{OPT}), \tag{44}$$

which implies that $\underline{w}_\ell \in [\frac{f(\mathrm{OPT})}{7(p+d+1)(1+c)}, \frac{3f(\mathrm{OPT})}{2}]$.

Proof: We first note the function value of set S^* returned by Algorithm 4 is no less than

$$f(S^*) \geq \min \Big\{ \frac{2\theta}{3}, \frac{f(\mathrm{OPT}) - \theta(d - |\mathrm{OPT} \cap B|/2)}{p + |\mathrm{OPT} \cap B| + 1} - (p\varepsilon + \varepsilon) \cdot f(\mathrm{OPT}) \Big\}. \tag{45}$$

Inequality (45) directly follows from Proposition 12 and 13, by taking the minimum of the two lower bounds.

The RHS of (44) directly follows from the fact that

$$\underline{w}_{i+1} = \frac{3}{2}\max_{\lambda} f(S_\lambda^{(i)}) \leq \frac{3}{2}f(\mathrm{OPT}).$$

We finish the proof of the LHS by induction. For the base case when $i = 0$, inequality (44) is equivalent to the statement that $\underline{w}_1 \geq \frac{3OPT}{7(p+d+1)n}$, which is true based on the definition of $\underline{w}_1$. Now suppose that (44) holds for $i = s-1$, *i.e.*,

$$\underline{w}_s \geq \frac{3f(\mathrm{OPT})}{7(p+d+1)(1+\alpha_{s-1})}. \tag{46}$$

Following from (46), we have $\bar{w}_s = (1+\alpha_{s-1})\underline{w}_s \geq \frac{3f(\mathrm{OPT})}{7(p+d+1)}$. As a consequence, there must exist an integer z_s during the s-th iteration, such that

$$\lambda_s^* = \underline{\omega}_s(1+\alpha_s)^{z_s} \in \Big[\frac{3f(\mathrm{OPT})}{7(p+d+1)(1+\alpha_s)}, \frac{3f(\mathrm{OPT})}{7(p+d+1)}\Big], \tag{47}$$

which implies that $\underline{w}_{s+1}$ can be lower bounded as follows when $\varepsilon = \frac{2}{7(p+1)(p+d+1)}$,

$$\begin{aligned}
\underline{w}_{s+1} &\overset{(a)}{\geq} \frac{3f(BT(\lambda_s^*))}{2} \\
&\overset{(b)}{\geq} \frac{3}{2}\cdot\min\Big\{\frac{2}{3}\cdot\frac{3f(\mathrm{OPT})}{7(1+\alpha_s)(p+d+1)}, \frac{f(\mathrm{OPT}) - \frac{3(d-|\mathrm{OPT}\cap B|/2)OPT}{7(p+d+1)}}{p+|\mathrm{OPT}\cap B|+1} - \frac{2f(\mathrm{OPT})}{7(p+d+1)}\Big\} \\
&\overset{(c)}{\geq} \frac{3}{7(p+d+1)(1+\alpha_s)}f(\mathrm{OPT})
\end{aligned}$$

where (a) is based on the definition of $\underline{w}_{s+1}$. Plugging (47) into (45), we can obtain (b). (c) follows from Fact 11. Hence we have

$$\bar{w}_{s+1} = (1+\alpha_s)\cdot\underline{w}_{s+1} \geq \frac{3}{7(p+d+1)}f(\mathrm{OPT}),$$

which indicates that (44) also holds for $i = s$. The proof is complete. □

C.9 Proof of Theorem 4

Proof: Consider the optimal threshold θ^* defined as

$$\theta^* = \frac{f(\mathrm{OPT})}{d + 2/3 + |\mathrm{OPT}\cap B|/6 + \frac{2}{3}p},$$

according to Proposition 23, it is easy to see that

$$\theta^* \in \Big[\frac{3}{2}\cdot\frac{f(\mathrm{OPT})}{p+\frac{7}{4}d+1}, f(\mathrm{OPT})\Big] \subseteq \Big[\frac{w_\ell}{p+7/4d+1}, 7(p+d+1)(1+c)\underline{w}_\ell\Big].$$

Hence there exist an iteration in which $\lambda \in [(1-\delta)\theta^*, \theta^*]$, from which we know that

$$\begin{aligned}
f(S_o) &\geq \min\Big\{\frac{2(1-\delta)\theta^*}{3}, \frac{f(\mathrm{OPT}) - \theta^*(d - |\mathrm{OPT}\cap B|/2)}{p+|\mathrm{OPT}\cap B|+1} - \delta OPT\Big\} \\
&\geq \Big(\frac{1}{p+7d/4+1} - 2\delta\Big)f(\mathrm{OPT}).
\end{aligned}$$

The proof is complete. □

Time Complexity. To analyze the time complexity of obtaining an $1/(p+\frac{7}{4}d+1)-\Theta(1)$-approximate solution, we let $\delta=\Theta(1)$. Note that using our adaptive decreasing threshold algorithm, we are able to obtain a constant approximation of $f(\mathrm{OPT})$ in $\log\log n$ rounds, while the time complexity in each round is $n\log n$, thus the total time complexity is $O(n\log n\cdot\log\log n)$.

D Supplementary Materials of Section 5

D.1 An Improved Algorithm for Multiple Knapsack Constraints

Algorithm overview. We first remark that the constant approximation of $f(\mathrm{OPT})$ required by Algorithm 6 can be obtained via a similar approach in Section 2. Utilizing this input value in the initialization, we maintain a threshold τ on the profit density—For each fixed value of threshold, elements with profit density exceeding τ will be picked into the our first candidate solution S, if the corresponding new set satisfies the knapsack constraints. At the end of each iteration, we decrease the threshold τ by a multiplicative factor, until τ reaches the preset boundary value. If there exists an element violates some knapsack constraints, we construct a new candidate solution $\tilde{S}$ in the same way as our backtracking threshold algorithm in Section 4. Finally we choose the best solution among all candidate sets.

The performance of Algorithm 6 is summarized in the following lemma.

Algorithm 6: Backtracking+Threshold Decreasing Algorithm for d Knapsack Constraints

Input: submodular function $f(\cdot):2^E\to\mathbb{R}^+$, knapsack function $c_i(\cdot):2^E\to\mathbb{R}^+$ $(i\in[d])$.
Output: Set $S^*\in\cap_{i=1}^d\mathcal{K}_i$.
Initialization: $\lambda\leftarrow$ c-approximation of $f(\mathrm{OPT})$, $e^*\leftarrow\mathrm{argmax}_{e\in E}f(e)$, $S,\tilde{S}\leftarrow\varnothing$, $\tau=\frac{\lambda}{cd}$
while $\tau\le\frac{3\lambda}{7d\cdot c}$ **do**
  for $e\in E\setminus B$ **do**
    if $f_S(e)\ge\tau\sum_{i=1}^d c_i(e)$ **then**
      if $c_i(S+e)\le 1$ $(\forall i\in[d])$ **then**
        $S\leftarrow S+e$;
      else
        $\tilde{S}\leftarrow\{e,\mathrm{argmax}_{e\in S}\sum_{i=1}^d c_i(e)\}$;
        for $e\in S\setminus\tilde{S}$ **do**
          if $c_i(\tilde{S}+e)\le 1$ $(\forall i\in[d])$ **then**
            $\tilde{S}\leftarrow\tilde{S}+e$;
  $\tau\leftarrow(1-\varepsilon)\tau$
return $S^*\leftarrow\mathrm{argmax}_{T\in\{S^*,S,\{e^*\}\}}f(T)$

Lemma 24. *Algorithm 6 returns a feasible solution S^* such that*

$$f(S^*)\ge\Big(\frac{1}{\frac{7}{4}d+\frac{9}{10}+\frac{9}{120d+16}}-O(\varepsilon)\Big)f(\mathrm{OPT}). \tag{48}$$

while computing S^* requires $O(\max\{\varepsilon^{-1},\log\log n\})$ queries per element, i.e., *$O(n\cdot\max\{\varepsilon^{-1},\log\log n\})$ queries in total.*

Proof: In Algorithm 6, the threshold is initialized as $\tau = \tau_0 = \frac{\lambda}{cd}$ and proceeds with the update rule $\tau = \tau_i = \tau_0(1-\varepsilon)^i$. Suppose that we obtain set $S^{(i)}$ after the iteration in which $\tau = \tau_i$, and there are s iterations in total. If $\tau_s \leq \frac{3}{7d} \cdot f(\mathrm{OPT})$, then

$$f(\mathrm{OPT}) - f(S) \leq \sum_{e \in \mathrm{OPT}} [f(S+e) - f(S)] \leq \tau_s \sum_{e \in \mathrm{OPT}} \sum_{j=1}^{d} c_j(e) \leq \frac{3}{7} \cdot f(\mathrm{OPT}).$$

Rearranging the terms we can know that (48) holds. Therefore we can assume that τ_s is strictly larger than the minimum threshold, which indeed implies that $\tilde{S} \neq \varnothing$, *i.e.*, there must exist $\tilde{e}_1$ such that $c_i(S+e) > 1$ for some $i \in [d]$. Let $\tilde{e}_1$ denote the last candidate element being added into S and $\tilde{e}_2$ represents the element with maximum total cost in S. We can establish the following lower bounds on the objective value of set $\tilde{S}$ and $\{\tilde{e}_1, \tilde{e}_2\}$,

$$f(\tilde{S}) \geq \tau_s \cdot \Big[1 - \sum_{j=1}^{d} c_j(\tilde{e}_2)\Big], \tag{49}$$

$$f(\{\tilde{e}_1, \tilde{e}_2\}) \geq \tau_s \cdot \sum_{i=1}^{2} \sum_{j=1}^{d} c_j(\tilde{e}_i). \tag{50}$$

While $f(S)$ is no less than

$$f(S) \geq \tau_s \cdot \sum_{j=1}^{d} c_j(S) \geq \tau_s \cdot \Big[1 - \sum_{j=1}^{d} c_j(\tilde{e}_1)\Big]. \tag{51}$$

Adding inequalities (49) and (50) together, we can get

$$f(S^*) \geq \max\{f(\tilde{S}), f(\{\tilde{e}_1, \tilde{e}_2\})\} \geq \tau_s \cdot \frac{1 + \sum_{j=1}^{d} c_j(\tilde{e}_1)}{2}. \tag{52}$$

Considering the iteration in which $\tau = \tau_i$, elements in $\mathrm{OPT} \setminus (B \cap S^{(i-1)})$ are discarded since

$$f(S^{(i-1)} + e) - f(S^{(i-1)}) \leq \tau_{i-1} \cdot \sum_{j=1}^{d} c_j(e), \tag{53}$$

which implies that

$$\begin{aligned}
f(\mathrm{OPT} \setminus B) - f(S^{(i-1)}) \leq & \sum_{e \in \mathrm{OPT} \setminus (B \cup S^{(i-1)})} [f(S^{(i-1)} + e) - f(S^{(i-1)})] && \text{(submodularity)} \\
\leq & \frac{\tau_i}{1-\varepsilon} \sum_{j=1}^{d} c_j(\mathrm{OPT} \setminus (B \cup S^{(i-1)})) && \text{(inequality (53))} \\
\leq & \frac{[f(S^{(i)}) - f(S^{(i-1)})]}{(1-\varepsilon)\sum_{j=1}^{d} c_j(e^{(i)})} \Big(\sum_{j=1}^{d} [c_j(\mathrm{OPT}) - c_j(\mathrm{OPT} \cap B)] \Big) \\
\leq & \frac{(d - |\mathrm{OPT} \cap B|/2) \cdot [f(S^{(i)}) - f(S^{(i-1)})]}{(1-\varepsilon)\sum_{j=1}^{d} c_j(e^{(i)})}. && (54)
\end{aligned}$$

Rearranging this inequality, we get

$$f(S^{(i)}) - f(\mathrm{OPT} \setminus B) \geq \Big(1 - \frac{(1-\varepsilon)\sum_{j=1}^{d} c_j(e^{(i)})}{d - |\mathrm{OPT} \cap B|/2}\Big) \cdot [f(S^{(i-1)}) - f(\mathrm{OPT} \setminus B)]. \tag{55}$$

Solving this recursive inequality and using the fact that $1 - y \leq e^{-y}$, we have

$$f(S + \tilde{e}_1) = f(S^{(s)}) \geq \Big(1 - \exp\Big(-\frac{(1-\varepsilon)\sum_{j=1}^{d} c_j(S + \tilde{e}_1)}{d - |\mathrm{OPT} \cap B|/2}\Big)\Big) \cdot f(\mathrm{OPT} \setminus B), \tag{56}$$

based on which we have

$$f(S) \geq \frac{1 - \exp\Big(-\frac{1-\sum_{j=1}^{d} c_j(\tilde{e}_1)}{d-|\mathrm{OPT}\cap B|/2}\Big)}{1 + |\mathrm{OPT} \cap B| \cdot \Big[1 - \exp\Big(-\frac{1-\sum_{j=1}^{d} c_j(\tilde{e}_1)}{d-|\mathrm{OPT}\cap B|/2}\Big)\Big]} f(\mathrm{OPT}) - O(\varepsilon) f(\mathrm{OPT}). \tag{57}$$

On the other hand, plugging $i = s$ into (54), we can get

$$f(\mathrm{OPT} \setminus B) - f(S^{(s)}) \leq f(\mathrm{OPT} \setminus B) - f(S^{(s-1)}) \leq \frac{\tau_s}{1-\varepsilon} \cdot [d - |\mathrm{OPT} \cap B|/2]. \tag{58}$$

Rearranging this inequality,

$$f(S^*) \geq \frac{f(\mathrm{OPT}) - (d - |\mathrm{OPT} \cap B|/2) \cdot \tau_s}{|\mathrm{OPT} \cap B| + 1} - O(\varepsilon) f(\mathrm{OPT}). \tag{59}$$

Here we remark that s can assumed to be strictly positive, otherwise τ_s will be equal to τ_0, and it is easy to see that (48) holds in this case since

$$f(S^*) \geq \frac{f(S) + f(\tilde{S}) + f(\{\tilde{e}_1, \tilde{e}_2\})}{3} \geq \frac{2\tau_0}{3} \geq \frac{2f(\mathrm{OPT})}{3d}. \tag{60}$$

Now by combining (51), (52) and (59) together, we can obtain

$$\begin{aligned} f(S^*) \geq \min_{\tau_s, c_j(\tilde{e}_1)} \max \Big\{ & \Big(1 - \sum_{j=1}^{d} c_j(\tilde{e}_1)\Big) \cdot \tau_s, \frac{1 + \sum_{i=1}^{d} c_i(\tilde{e}_1)}{2} \cdot \tau_s, \\ & \frac{OPT - (d - |\mathrm{OPT} \cap B|/2)\tau_s}{|\mathrm{OPT} \cap B| + 1} \Big\} - O(\varepsilon) f(\mathrm{OPT}) \\ \geq \min_{c_j(\tilde{e}_1)} \max \Big\{ & \frac{(1 - \sum_{j=1}^{d} c_j(\tilde{e}_1))}{(|\mathrm{OPT} \cap B| + 1)(1 - \sum_{j=1}^{d} c_j(\tilde{e}_1)) + (d - |\mathrm{OPT} \cap B|/2)} \cdot f(\mathrm{OPT}), \\ & \frac{(1 + \sum_{j=1}^{d} c_j(\tilde{e}_1))}{(|\mathrm{OPT} \cap B| + 1)(1 + \sum_{j=1}^{d} c_j(\tilde{e}_1)) + (2d - |\mathrm{OPT} \cap B|)} \cdot f(\mathrm{OPT}) \Big\} - O(\varepsilon) f(\mathrm{OPT}). \end{aligned} \tag{61}$$

Observe that if the number of large elements in the optimal solution is strictly less than d, then we can directly derive the following approximation result based on (61):

$$f(S^*) \geq \frac{4}{6d + |\mathrm{OPT} \cap B| + 4} \geq \frac{4}{7d + 3}, \tag{62}$$

then (48) follows. Hence it suffices to consider the case when there are exactly d large elements in the optimal solution set, *i.e.*, $|\mathrm{OPT} \cap B| = d$. Combining (61) and (57), we can obtain that

$$f(S^*) \geq \min_{c_j(\tilde{e}_1)} \max \Big\{ \underbrace{\frac{1}{1\Big/\Big[1-\exp\Big(-\frac{2-2\sum_{j=1}^{d} c_j(\tilde{e}_1)}{d}\Big)\Big]+d}}_{\Sigma_1}, \underbrace{\frac{1}{(d+1)+d/[\sum_{j=1}^{d} c_j(\tilde{e}_1)+1]}}_{\Sigma_2} \Big\} f(\mathrm{OPT}) - O(\varepsilon)OPT \tag{63}$$

Observe that Σ_1 is monotone decreasing with respect to the total weight of elements of $\tilde{e}_1$ while Σ_2 is monotone increasing. Therefore when $\sum_{j=1}^{d} c_j(\tilde{e}_1) \geq \gamma = \frac{1}{3} + \frac{8}{45d}$,

$$f(S^*) \geq \Sigma_2 \geq \frac{1}{(d+1)+d/[\sum_{j=1}^{d} c_j(\tilde{e}_1)+1]} \tag{64}$$

$$\geq \frac{1}{\frac{7}{4}d + \frac{9}{10} + \frac{9}{120d+16}}. \tag{65}$$

We remark that

$$\Sigma_1 \geq \Sigma_2 \ \ \Big(\forall \sum_{j=1}^{d} c_j(\tilde{e}_1) \leq \gamma\Big). \tag{66}$$

Then by the monotonicity of Σ_1, we know that (48) holds. To prove (66), it is sufficient to show the following conclusion, owing to the monotonicity of Σ_2,

$$g = \exp\Big(\frac{2-2\gamma}{d}\Big) - \Big(1 + \frac{\gamma+1}{d}\Big) \geq 0.$$

Note that

$$\frac{\partial g}{\partial(\frac{1}{d})} = \exp\Big(\frac{4}{3d} - \frac{16}{45d^2}\Big)\Big(\frac{4}{3} - \frac{32}{45d}\Big) - \Big(\frac{4}{3} + \frac{16}{45d}\Big), \tag{67}$$

utilizing the inequality $e^x \geq 1 + x$, it can be verified that $\frac{\partial g}{\partial(\frac{1}{d})} \geq 0$ for $d \geq 2$ and $g > 0$ for $d = 1$, hence

$$g \geq \lim_{d\to\infty} \Big[\exp\Big(\frac{2-2\gamma}{d}\Big) - \Big(1 + \frac{\gamma+1}{d}\Big)\Big] = 0.$$

Time Complexity. The time complexity follows from the following facts:

- Obtaining a constant approximation of $f(\mathrm{OPT})$ requires $O(\log\log n)$ queries per element.
- There are $O(\varepsilon^{-1})$ iterations in the *while* loop, and each iteration requires a constant number of queries per element to add new elements into S. In addition, the number of queries used to construct set $\tilde{S}$ is in a lower order term.

To summarize, the number of queries performed per element in Algorithm 6 is in the order of $O(\max\{\varepsilon^{-1}, \log\log n\})$. The proof is complete. □

D.2 Details of Algorithm 7

Algorithm 7: Main Algorithm

Input: $\bar{\zeta}, \underline{\zeta}$, approximation parameter δ, Algorithm 3 and its solution S^* on ground set E
Initialization: $\zeta \leftarrow \bar{\zeta}$
while $\zeta \geq \underline{\zeta}$ **do**
  $e_\zeta \leftarrow \text{argmin}\{c(e) | \zeta \leq f(e) \leq \frac{\zeta}{1-\delta}, e \in E\}$
  Apply the backtracking threshold algorithm (Algorithm 3) on function $g_\zeta(\cdot) : 2^{E \setminus \{e_\zeta\}} \to \mathbb{R}^+$ under budget $1 - c(e_\zeta)$, where for $\forall S \subseteq E \setminus \{e_\zeta\}$, $g(S) = f(S + e_\zeta) - f(e_\zeta)$ and obtain solution S_ζ
  $\zeta \leftarrow \zeta(1-\delta)$
$\zeta^* \leftarrow \text{argmax}_\zeta \, f(S_\zeta + e_\zeta)$
$S'_o \leftarrow S_{\zeta^*} + e_{\zeta^*}$
Return $S_o \leftarrow \text{argmax}_{S \in \{S^*, S'_o\}} f(S)$

D.3 Proof of Lemma 14

Proof: In thread i of Algorithm 2, two thresholds $\tau_1^{(i)}$ and $\tau_2^{(i)}$ are utilized to select elements. For a clean presentation, we omit the index of the thread and lower bound the quality of solution obtained by two sequential threshold τ_1 and τ_2 in double threshold backtracking algorithm.

In the following we use T_i $(1 \leq i \leq 2)$ to denote the collection of elements obtained by threshold τ_i. If there exists element $\hat{e}_i$ that has a marginal increment no less than τ_i but violates the knapsack constraint, T_i represents the value of candidate set before element $\hat{e}_i$ arrives. We further let $\text{OPT}' = \text{OPT} \setminus B$ and e_1, e_2 be the element with largest cost in T_1 and $T_2 \setminus T_1$ respectively, *i.e.*, $\tilde{e}_1 = \text{argmax}_{e \in T_1} c(e)$ and $\tilde{e}_2 = \text{argmax}_{e \in T_2 \setminus T_1} c(e)$.

We divide our analysis into three cases, according to the existence of budget violation in each iteration.

Case 1: Algorithm 2 stops at τ_1. In this case, there exists $\hat{e}_1$ such that $c(T_1 + \hat{e}_1) > 1$, and the marginal increment of $\hat{e}_1$ is no less than $f(T_1 + \hat{e}_1) - f(T_1) \geq \tau_1$. Via similar arguments as that for (28), we can obtain that

$$f(S^*) \geq \max\{f(T_1), f(\hat{e}_1, \tilde{e}_1), f(\widetilde{T}_1)\} \geq \frac{2}{3}\tau_1 \geq \frac{\tau_1 + \tau_2}{3}. \tag{68}$$

Case 2: Algorithm 2 stops at τ_2. Without loss of generality, we can assume that $c(T_1) \leq \frac{2}{3}$, otherwise we can immediately obtain the same lower bound as (68), *i.e.*,

$$f(S^*) \geq f(T_1) \geq c(T_1) \cdot \tau_1 \geq \frac{\tau_1 + \tau_2}{3}.$$

With the condition that $c(T_1) > \frac{2}{3}$, the weight of $\hat{e}_2$ satisfies that $c(\hat{e}_2) \geq 1 - c(T_2) \geq \frac{1}{3}$, since $\hat{e}_2$ cannot be added into T_2 due to the budget constraint. Recall that $\widetilde{T}_2$ is obtained by adding $\tilde{e}_2$ and then elements in T_2, until the total weight exceeds the budget, we have

$$c(\widetilde{T}_2 - \hat{e}_2) \geq [1 - c(\hat{e}_2) - c(\tilde{e}_1)] \cdot \mathbb{1}_{c(T_1 + \hat{e}_2) > 1} + [1 - c(\hat{e}_2) - c(\tilde{e}_2)] \cdot \mathbb{1}_{c(T_1 + \hat{e}_2) \leq 1}, \tag{69}$$

based on which we can obtain the following lower bound on the objective value of $\widetilde{T}_2$,

$$\begin{aligned} f(\widetilde{T}_2) =& [f(\widetilde{T}_2) - f(\widetilde{T}_2 - \hat{e}_2)] + f(\widetilde{T}_2 - \hat{e}_2) \\ \geq& [\tau_2 \cdot c(\hat{e}_2) + \tau_1 \cdot c(\widetilde{T}_2 - \hat{e}_2)] \cdot \mathbb{1}_{c(T_1+\hat{e}_2)>1} \\ \geq& [\tau_2 \cdot c(\hat{e}_2) + \tau_1 \cdot (1 - c(\hat{e}_2) - c(\tilde{e}_1))] \cdot \mathbb{1}_{c(T_1+\hat{e}_2)>1}. \end{aligned} \tag{70}$$

Notice that

$$\begin{aligned} f(\{\hat{e}_2, \tilde{e}_1\}) =& [f(\{\hat{e}_2, \tilde{e}_1\}) - f(e_1)] + f(e_1) \\ \geq& [f(\widetilde{T}_2 + \hat{e}_2) - f(\widetilde{T}_2)] + f(e_1) \\ \geq& \tau_2 \cdot c(\hat{e}_2) + \tau_1 \cdot c(\tilde{e}_1). \end{aligned} \tag{71}$$

Combining (70) and (71) together, we have

$$\begin{aligned} f(S^*) \geq& \frac{f(\widetilde{T}_2) + f(\{\hat{e}_2, \tilde{e}_1\})}{2} \\ \geq& \Big[\tau_2 \cdot c(\hat{e}_2) + \frac{\tau_1}{2} \cdot [1 - c(\hat{e}_2)]\Big] \cdot \mathbb{1}_{c(T_1+\hat{e}_2)>1} \qquad (72) \\ \geq& \frac{\tau_1 + \tau_2}{3} \cdot \mathbb{1}_{c(T_1+\hat{e}_2)>1, 2\tau_2 \geq \tau_1}, \qquad (73) \end{aligned}$$

where the last inequality holds due to the the monotonicity of (72) with respect to $c(\hat{e}_2)$, together with the fact that $c(\hat{e}_2) \geq \frac{1}{3}$.

Now we consider the case when $c(T_1 + \hat{e}_2) \leq 1$. Due to the definition of $\hat{e}_2$, we have $c(T_2 + \hat{e}_2) = c(T_1) + c(T_2 \setminus T_1) + c(\hat{e}_2) > 1$, which implies that $\max\{c(T_2 \setminus T_1), c(\hat{e}_2)\} \geq \frac{1-c(T_1)}{2}$. In addition,

$$\begin{aligned} f(S^*) \geq& \max\{f(T_2), f(\widetilde{T}_2)\} \geq \max\{f(T_2), f(T_1 + \hat{e}_2)\} \\ \geq& f(T_1) + \max\{f(T_2) - f(T_1), f(T_1 + \hat{e}_2) - f(T_1)\} \\ \geq& f(T_1) + \tau_2 \cdot \max\{c(T_2 \setminus T_1), c(\hat{e}_2)\} \\ \geq& \tau_1 \cdot c(T_1) + \tau_2 \cdot \frac{1 - c(T_1)}{2}. \end{aligned} \tag{74}$$

Notice that the lower bound in RHS of (74) is monotonically increasing with respect to the total weights of T_1, we have

$$f(S^*) \geq \frac{\tau_1 + \tau_2}{3} \cdot \mathbb{1}_{c(T_1+\hat{e}_2)\leq 1, c(T_1)\geq \frac{1}{3}}.$$

For the case when $c(T_1) \leq \frac{1}{3}$, we first argue that $T_2 \neq T_1$. Because the total weights of elements selected in the second iteration is no less than $c(T_2 \setminus T_1) > 1 - c(T_1) - c(\hat{e}_2) \geq \frac{1}{6} > 0$, hence element $\tilde{e}_2$ must exist. We next claim the following lower bound on $f(S^*)$,

$$\begin{aligned} f(S^*) \geq& \max\{f(T_2), f(\widetilde{T}_2)\} \\ \geq& \max\{f(T_1 + \tilde{e}_2), f(T_1 + \hat{e}_2)\} \qquad \text{(monotonicity of } f(\cdot) \text{ and } T_1 + \hat{e}_2 \text{ is feasible)} \\ =& f(T_1) + \max\{f(T_1 + \tilde{e}_2) - f(T_1), f(T_1 + \hat{e}_2) - f(T_1)\} \\ \geq& f(T_1) + \tau_2 \cdot \max\{c(\tilde{e}_2), c(\hat{e}_2)\}. \end{aligned} \tag{75}$$

Plugging the fact $f(T_1) \geq f(\mathrm{OPT}') - c(\mathrm{OPT}') \cdot \tau_1$ into (75), we have

$$f(S^*) \geq \Big(f(\mathrm{OPT}') - c(\mathrm{OPT}') \cdot \tau_1 + \frac{\tau_2}{3}\Big) \cdot \mathbb{1}_{\max\{c(\tilde{e}_2), c(\hat{e}_2)\} \geq \frac{1}{3}}. \tag{76}$$

If $\max\{c(\tilde{e}_2), c(\hat{e}_2)\} \leq \frac{1}{3}$, we have $c(T_1 \cup \{\hat{e}_2, \tilde{e}_2\}) \leq 1$, *i.e.*, $T_1 \cup \{\hat{e}_2, \tilde{e}_2\}$ is a feasible set. Consequently we have

$$\begin{aligned}
f(S^*) \geq & \frac{f(T_2) + f(\widetilde{T}_2) + f(T_1 \cup \{\hat{e}_2, \tilde{e}_2\})}{3} \\
\geq & f(T_1) + \frac{[f(T_2) - f(T_1)] + [f(\widetilde{T}_2) - f(T_1)] + [f(T_1 \cup \{\hat{e}_2, \tilde{e}_2\}) - f(T_1)]}{3} \\
\geq & f(T_1) + \tau_2 \cdot \frac{c(T_2 \setminus T_1) + c(\widetilde{T}_2 \setminus T_1) + c(\hat{e}_2) + c(\tilde{e}_2)}{3} \\
\geq & [f(\mathrm{OPT}') - c(\mathrm{OPT}') \cdot \tau_1] \cdot \mathbb{1}_{c(T_1) \leq \frac{1}{3}} + (c(T_1) \cdot \tau_1) \cdot \mathbb{1}_{c(T_1) \geq \frac{1}{3}} + \frac{2(1 - c(T_1))}{3} \cdot \tau_2 \\
\geq & \Big(f(\mathrm{OPT}') - c(\mathrm{OPT}') \cdot \tau_1 + \frac{4}{9}\tau_2\Big) \cdot \mathbb{1}_{c(T_1) \leq \frac{1}{3}} + \Big(\frac{\tau_1}{3} + \frac{4}{9}\tau_2\Big) \cdot \mathbb{1}_{c(T_1) \geq \frac{1}{3}}.
\end{aligned} \tag{77}$$

Case 3: Algorithm 2 stops without exceeding the budget. In this case, we have

$$\begin{aligned}
f(S^*) \geq & f(\mathrm{OPT}') - \sum_{e \in \mathrm{OPT}' \setminus S^*} [f(S^* + e) - f(S^*)] \\
\geq & f(\mathrm{OPT}') - \tau_2 \cdot c(\mathrm{OPT}').
\end{aligned} \tag{78}$$

Now we are ready to combine our analyses in the aforementioned three cases, *i.e.*, inequalities (68), (73) and (77)-(78),

$$\begin{aligned}
f(S^*) \geq & \min\Big\{\frac{\tau_1 + \tau_2}{3}, f(\mathrm{OPT}') - \tau_1 \cdot c(\mathrm{OPT}') + \frac{\tau_2}{3}, f(\mathrm{OPT}') - \tau_2 \cdot c(\mathrm{OPT}')\Big\} \\
\overset{(a)}{\geq} & \Big(\frac{6c(\mathrm{OPT}') + 1}{[3c(\mathrm{OPT}') + 1]^2} - \varepsilon\Big) \cdot f(\mathrm{OPT}') \qquad (79) \\
\geq & \Big(\frac{7}{16} - \varepsilon\Big) f(\mathrm{OPT}) \cdot \mathbb{1}_{\mathrm{OPT} \cap B = \varnothing} + \frac{16}{25}[f(\mathrm{OPT}) - f(\mathrm{OPT} \cap B)] \cdot \mathbb{1}_{\mathrm{OPT} \cap B \neq \varnothing}. \qquad (80)
\end{aligned}$$

where (a) holds with equality when

$$\tau_1 = \frac{3}{3c(\mathrm{OPT}') + 1} f(\mathrm{OPT}') - O(\varepsilon) f(\mathrm{OPT})$$

and

$$\tau_2 = \frac{9c(\mathrm{OPT}')}{[3c(\mathrm{OPT}') + 1]^2} f(\mathrm{OPT}') - O(\varepsilon) f(\mathrm{OPT}).$$

Observe that the coefficient of $f(\mathrm{OPT}')$ in (79) decreases with respect to the weight of OPT', thus (80) follows from the facts that $c(\mathrm{OPT}') \leq 1 - \frac{1}{2} \cdot \mathbb{1}_{\mathrm{OPT} \cap B \neq \varnothing}$ and $f(\mathrm{OPT}') \geq f(\mathrm{OPT}) - f(\mathrm{OPT} \cap B) \cdot \mathbb{1}_{\mathrm{OPT} \cap B \neq \varnothing}$. □

D.4 Proof of Theorem 5

Proof: For set S'_o obtained in the 8-th line of Algorithm 7, we have

$$f(S'_o) \geq \frac{4}{11} f(\mathrm{OPT}) + \Big(\frac{3}{11} - \delta\Big) f(\mathrm{OPT} \cap B). \tag{81}$$

Consider the iteration when

$$(1-\delta) f(\mathrm{OPT} \cap B) \leq \zeta = \hat{\zeta} \leq f(\mathrm{OPT} \cap B),$$

we claim that

$$c(e_{\hat{\zeta}}) \leq c(\mathrm{OPT} \cap B),$$

since $\mathrm{OPT} \cap B$ is a candidate element when selecting element $e_{\hat{\zeta}}$. Moreover, $\mathrm{OPT} \setminus B$ is a feasible solution for the residual problem induced by $e_{\hat{\zeta}}$. If we apply an β-approximation algorithm on the corresponding residual problem, the following inequality holds for $S_{\hat{\zeta}}$,

$$g(S_{\hat{\zeta}}) \geq \beta \cdot g(\mathrm{OPT} \setminus B). \tag{82}$$

Plugging the definition of the residual function g into (82),

$$\begin{aligned} f(S^*) \geq f(S_{\hat{\zeta}} + e_{\hat{\zeta}}) &\overset{(a)}{\geq} \beta \cdot f(\mathrm{OPT} \setminus B) + (1-\beta) \cdot f(e_{\hat{\zeta}}) \\ &\overset{(b)}{\geq} \beta \cdot f(\mathrm{OPT}) + (1-2\beta) \cdot f(\mathrm{OPT} \cap B) - \delta \cdot f(\mathrm{OPT} \cap B), \end{aligned} \tag{83}$$

where (a) we use the fact that $g(\mathrm{OPT} \cap B) = f(\mathrm{OPT} \cap B + e_{\hat{\zeta}}) - f(e_{\hat{\zeta}}) \geq f(\mathrm{OPT} \cap B) - f(e_{\hat{\zeta}})$; (b) holds since $f(\mathrm{OPT} \setminus B) \geq f(\mathrm{OPT}) - f(\mathrm{OPT} \cap B)$ and $f(e_{\hat{\zeta}}) \geq \hat{\zeta} \geq (1-\delta) f(\mathrm{OPT} \cap B)$.

Recall that our Algorithm 3 provides an approximation ratio of $\beta = \frac{4}{11}$. Taken together with Lemma 14, we have

$$\begin{aligned} f(S^*) \geq & \min_B \max \Big\{ \frac{7}{16} f(\mathrm{OPT}) \cdot \mathbb{1}_{\mathrm{OPT} \cap B = \varnothing} + \frac{16}{25} [f(\mathrm{OPT}) - f(\mathrm{OPT} \cap B)] \cdot \mathbb{1}_{\mathrm{OPT} \cap B \neq \varnothing}, \\ & \qquad \frac{4}{11} f(\mathrm{OPT}) + \Big(\frac{3}{11} - \delta\Big) \cdot f(\mathrm{OPT} \cap B) \Big\} - O(\varepsilon) f(\mathrm{OPT}) \\ \geq & \Big(\frac{7}{16} - \varepsilon\Big) \cdot f(\mathrm{OPT}). \end{aligned}$$

The proof of approximation guarantee is complete.

Time complexity of offline algorithm. Both Algorithm 2 and Algorithm 7 require a constant upper and lower approximations of $f(\mathrm{OPT})$ as inputs, which can be obtained by our backtracking threshold algorithm in $O(n \log\log n)$ time in the offline case. Notice that in Algorithm 2, there are $O(\varepsilon^{-1})$ different values of λ and it runs in $O(n)$ time for each fixed λ, from which we know that the time complexity of Algorithm 2 is $O(n \cdot \max\{\varepsilon^{-1}, \log\log n\})$. And Algorithm 7 can be accomplished within the same order of time, as it requires $O(\delta^{-1})$ calls to Backtracking Threshold Algorithm and $\delta = O(1)$.

Complexity in streaming setting. In the streaming model, we run Algorithm 2 and Algorithm 7 in parallel. The main difference with the offline algorithm lies in the approach used to obtain a constant approximation of $f(\mathrm{OPT})$. Since $\frac{f(\mathrm{OPT})}{\max_e f(e)}$ lies in the range of $[1, n]$, we can maintain $O(\frac{\log n}{\varepsilon})$ copies of solutions in parallel for each possible approximation value of OPT, which implies a total time complexity of $O(\frac{n \log n}{\varepsilon})$ and space complexity of $O(\frac{n \log n}{\varepsilon})$. □

D.5 Proof of Claim 17

Proof: Utilizing the γ-approximate algorithm for $U^\sharp$-residual problem, we can obtain a set U' such that

$$f_{U^\sharp}(U') \geq \gamma \cdot \max_{S \subseteq E \setminus U^\sharp} f_{U^\sharp}(S) \geq \gamma f_{U^\sharp}(\mathrm{OPT} \setminus (U \cup U^\sharp)), \tag{84}$$

where the last inequality follows from the fact that $\mathrm{OPT} \setminus (U \cup U^\sharp)$ is a feasible solution to $U^\sharp$-residual problem. Plugging the definition of $U^\sharp$-residual function into (84), we can obtain

$$\begin{aligned} f(U' \cup U^\sharp) \geq & \gamma f((\mathrm{OPT} \cup U^\sharp) \setminus U) + (1-\gamma) f(U^\sharp) \\ \geq & \gamma f(\mathrm{OPT}) + (1 - \gamma\beta - \gamma) f(U^\sharp). \end{aligned}$$

If $\gamma + \gamma\beta \leq 1$, then we have $f(U' \cup U^\sharp) \geq \gamma f(\mathrm{OPT})$. Otherwise utilizing the simple fact that $f(U^\sharp) \leq f(U' \cup U^\sharp)$, we can obtain $f(U' \cup U^\sharp) \geq \frac{f(\mathrm{OPT})}{1+\beta}$. □

D.6 Details of Algorithm 8

Algorithm 8: An $O_\varepsilon(n \log \log n)$ Time Deterministic Algorithm for $\mathbf{A} \in \{0,1\}^{O(1) \times n}$

Construct a $(\frac{1}{2} - \varepsilon, 1 + \varepsilon)$-shadow set of $\mathrm{OPT}_{i_\varepsilon}$, denoted by $S^\sharp$, by (1) guessing the marginal increments of elements in $\mathrm{OPT}_{i_\varepsilon}$ via set $\mathcal{G}_\varepsilon = \{0, \frac{\varepsilon}{|O_{i_\varepsilon}|} f(\mathrm{OPT}), \frac{2\varepsilon}{|O_{i_\varepsilon}|} f(\mathrm{OPT}), \cdots, f(\mathrm{OPT})\}$; (2) guessing the cost vector of elements in O_{i_ε}, which belongs to $\{0,1\}^d$.
$b'_i \leftarrow b_i - c_i(S^\sharp)$ $(\forall i \in [d])$
$w_i \leftarrow \frac{1}{b'_i}$ $(\forall i \in [d])$
$\bar{\lambda}, \underline{\lambda} \leftarrow$ Upper and lower bounds on the ratio of marginal gain and weight
$T \leftarrow \varnothing$, $\lambda \leftarrow \bar{\lambda}$
while $\lambda \geq \underline{\lambda}$ **do**
  for $e \in E \setminus (E_\Gamma \cup S^\sharp \cup T)$ **do**
    if $\sum_{i=1}^d b'_i w_i \leq 1 + \delta W + (\delta W)^2$ *and* $f_{T \cup S^\sharp}(e) \geq \lambda \cdot \sum_{i=1}^d w_i c_i(e)$ **then**
      $w_i \leftarrow \left[1 + \frac{\delta W c_i(e)}{b'_i} + \left(\frac{\delta W c_i(e)}{b'_i}\right)^2\right] \cdot w_i$ $(\forall i \in [d])$
      $T \leftarrow T + e$
Return $S_o = S_\varepsilon^\sharp \cup T$

D.7 Proof of Lemma 18

Proof: We denote set $S^\sharp = \{e_1^\sharp, e_2^\sharp, \ldots, e_{|S^\sharp|}^\sharp\}$ and $S_i^\sharp = \{e_1^\sharp, e_2^\sharp, \ldots, e_i^\sharp\}$ $(i \in [|S^\sharp|])$, where $e_i^\sharp$ is the element selected at the i-th step of guessing. According to the definition of $\mathcal{G}_\varepsilon$ in Algorithm 8, we

know that the increment of $e_i^\sharp$ with respect to set $S_{i-1}^\sharp$ is similar as that of element $o_i^\sharp$, *i.e.*,

$$f(S_i^\sharp) - f(S_{i-1}^\sharp) = f(S_{i-1}^\sharp + e_i^\sharp) - f(S_{i-1}^\sharp) \tag{85}$$

$$\geq (1-\varepsilon)[f(S_{i-1}^\sharp + o_i) - f(S_{i-1}^\sharp)] - \frac{\varepsilon}{|O_{i_\varepsilon}|} f(\mathrm{OPT}). \tag{86}$$

The increment of $o_i^\natural$ in (85) can be lower bounded as

$$f(S_{i-1}^\sharp + o_i) - f(S_{i-1}^\sharp) \overset{(a)}{\geq} f(S_{i-1}^\sharp \cup (\mathrm{OPT} \setminus \mathrm{OPT}_{i-1})) - f(S_{i-1}^\sharp \cup (\mathrm{OPT} \setminus \mathrm{OPT}_i)) \tag{87}$$

$$\overset{(b)}{\geq} f(S_{i-1}^\sharp \cup (\mathrm{OPT} \setminus \mathrm{OPT}_{i-1})) - f(S_i^\sharp \cup (\mathrm{OPT} \setminus \mathrm{OPT}_i)) \tag{88}$$

where in (a) we use submodularity of f and the fact that $S_{i-1}^\sharp \cup (\mathrm{OPT} \setminus \mathrm{OPT}_i) + o_i = S_{i-1}^\sharp \cup (\mathrm{OPT} \setminus \mathrm{OPT}_{i-1})$. (b) follows from monotonicity of f. Take summarization from $i = 1$ to $|S^\sharp|$, we can obtain

$$\begin{aligned}
f(S^\sharp) =& \sum_{i=1}^{|S^\sharp|} \left[f(S_i^\sharp) - f(S_{i-1}^\sharp) \right] \\
\geq& (1-\varepsilon) \sum_{i=1}^{|S^\sharp|} \left[f(S_{i-1}^\sharp + o_i) - f(S_{i-1}^\sharp) \right] - \varepsilon f(\mathrm{OPT}) \\
\geq& (1-\varepsilon) \sum_{i=1}^{|S^\sharp|} \left[f(S_{i-1}^\sharp \cup (\mathrm{OPT} \setminus \mathrm{OPT}_{i-1})) - f(S_i^\sharp \cup (\mathrm{OPT} \setminus \mathrm{OPT}_i)) \right] - \varepsilon f(\mathrm{OPT}) \\
=& (1-\varepsilon)[f(\mathrm{OPT}) - f((\mathrm{OPT} \cup S^\natural) \setminus \mathrm{OPT}_{i_\varepsilon})] - \varepsilon f(\mathrm{OPT}).
\end{aligned}$$

Rearranging the inequality above, we can obtain the first inequality. For the second inequality, notice that

$$\begin{aligned}
f(\mathrm{OPT}_{i_\varepsilon}) - f(S^\sharp) \leq& \sum_{e \in O} [f(S^\sharp + e) - f(S^\sharp)] && \text{(submodularity)} \\
=& \sum_{i=1}^{|S^\sharp|} [f(S_{i-1}^\sharp + o_i) - f(S_{i-1}^\sharp)] \\
\leq& \frac{1}{1-\varepsilon} \sum_{i=1}^{|S^\sharp|} [f(S_{i-1}^\sharp + e_i^\sharp) - f(S_{i-1}^\sharp)] && \text{(selection rule of our algorithm)} \\
=& \frac{f(S_{i-1}^\sharp)}{1-\varepsilon}.
\end{aligned}$$

Rearranging the terms, the proof is complete. □

D.8 Proof of Proposition 19

Proof: We first note that $|\mathrm{OPT} \cap E_\Gamma| \leq dW$, since for each $i \in \Gamma$, there are at most W elements in $\mathrm{OPT} \cap E_\Gamma$ whose cost in the i-th dimension is non-zero and $|\Gamma| \leq d$. We next claim the following proposition about the performance of Proposition 25.

Proposition 25. *Set T is an $(1-1/e-\delta)$-approximate solution for the $S^\sharp$–residual problem,* i.e.*,*

$$f(T\cup S^\sharp)-f(S^\sharp)\geq(1-1/e-O(\delta))[f(\mathrm{OPT}\cup S^\sharp\setminus(\mathrm{OPT}_{i_\varepsilon}\cup E_\gamma))-f(S^\sharp)]. \tag{89}$$

Assuming the correctness of Proposition 25, we are ready to finish the proof. According to Proposition 18, we have

$$f(\mathrm{OPT}\setminus E_\gamma)-f((\mathrm{OPT}\cup S^\sharp)\setminus(\mathrm{OPT}_{i_\varepsilon}\cup E_\gamma))\leq f(S^\sharp). \tag{90}$$

Combining with Proposition 25 in which we let $\delta=\frac{1}{2}-\frac{1}{e}$, we know that

$$f(T\cup S^\sharp)\geq\frac{f(\mathrm{OPT}\setminus E_\Gamma)}{2}\geq\Big(\frac{1}{2}-\varepsilon\Big)f(\mathrm{OPT}), \tag{91}$$

which follows from the definition of E_Γ, as removing each element in $O\cap E_\Gamma$ will incur a loss no more than $\frac{\varepsilon}{dW}f(\mathrm{OPT})$ and $|\mathrm{OPT}\cap E_\Gamma|\leq dW$.

□

Proof of Proposition 25: In this proof we shall depart from the previous notation, for any set $S\subseteq E$, we let $S'=S\setminus(S^\sharp\cup E_\Gamma)$. Suppose that element $e^{(t+1)}$ is selected at the $(t+1)$-th iteration of the *while* loop, then for $\forall e\in E'\setminus T^{(t)}$, we have

$$f_{T^{(t)}\cup S^\sharp}(e)\leq\frac{\sum_{i=1}^d w_i^{(t)}c_i(e)}{\sum_{i=1}^d w_i^{(t)}c_i(e^{(t+1)})}f_{T^{(t)}\cup S^\sharp}(e^{(t+1)}), \tag{92}$$

which implies that

$$\sum_{e\in\mathrm{OPT}'}f_{T^{(t)}\cup S^\sharp}(e)\leq\frac{\sum_{i=1}^d w_i^{(t)}c_i(\mathrm{OPT}')}{\sum_{i=1}^d w_i^{(t)}c_i(e^{(t+1)})}f_{T^{(t)}\cup S^\sharp}(e^{(t+1)}).$$

For the LHS, based on the submodularity of f, we have

$$\sum_{e\in\mathrm{OPT}'}f_{T^{(t)}\cup S^\sharp}(e)\geq f(\mathrm{OPT}'\cup S^\sharp)-f(T^{(t)}\cup S^\sharp). \tag{93}$$

Combining (92) and (93),

$$\begin{aligned}\sum_{i=1}^d w_i^{(t)}c_i(e^{(t+1)})\leq&\frac{f_{T^{(t)}\cup S^\sharp}(e^{(t+1)})}{f(\mathrm{OPT}'\cup S^\sharp)-f(T^{(t)}\cup S^\sharp)}\cdot\sum_{i=1}^d w_i^{(t)}c_i(\mathrm{OPT}')\\ \leq&\frac{f_{T^{(t)}\cup S^\sharp}(e^{(t+1)})}{f(\mathrm{OPT}'\cup S^\sharp)-f(T^{(t)}\cup S^\sharp)}\cdot\sum_{i=1}^d b_i'w_i^{(t)}.\end{aligned} \tag{94}$$

Observe that

$$\sum_{i=1}^d b_i'w_i^{(t+1)}=\sum_{i=1}^d b_i'w_i^{(t)}\cdot\Big[1+\frac{\delta Wc_i(e^{(t+1)})}{b_i'}+\Big(\frac{\delta Wc_i(e^{(t+1)})}{b_i'}\Big)^2\Big] \tag{95}$$

$$\leq\sum_{i=1}^d b_i'w_i^{(t)}+(\delta W+\delta^2W)\cdot\sum_{i=1}^d w_i^{(t)}c_i(e^{(t+1)}) \tag{96}$$

where we use $Wc_i(e^{(t)}) \leq b'_i$ as $W = \min\{\frac{b'_i}{c_i(e)} | e \in E \setminus (E_\Gamma \cup S^\sharp), i \in [d]\}$. Let $\Phi(t) = \sum_{i=1}^d b'_i w_i^{(t)}$, then $\Phi(0) = d$, and

$$\Phi(t+1) \leq (1 + \delta + \delta^2) \cdot \Phi(t), \tag{97}$$

which follows from the update rule of $w_i^{(t)}$ in Algorithm 8 and definition of W. Further we can obtain the following bound on the increment of Φ by utilizing (94),

$$\begin{aligned}\frac{\Phi(t+1)}{\Phi(t)} \leq & 1 + (\delta W + \delta^2 W) \cdot \frac{f_{T^{(t)} \cup S^\sharp}(e^{(t+1)})}{f(\mathrm{OPT}' \cup S^\sharp) - f(T^{(t)} \cup S^\sharp)} \\ \leq & \exp\Big\{(\delta W + \delta^2 W) \cdot \frac{f(T^{(t+1)} \cup S^\sharp) - f(T^{(t)} \cup S^\sharp)}{f(\mathrm{OPT}' \cup S^\sharp) - f(T^{(t)} \cup S^\sharp)}\Big\},\end{aligned} \tag{98}$$

which further implies that

$$\begin{aligned}\frac{\Phi(\bar{t})}{\Phi(0)} \leq & \exp\Big\{(\delta W + \delta^2 W) \cdot \sum_{t=0}^{\bar{t}-1} \frac{f(T^{(t+1)} \cup S^\sharp) - f(T^{(t)} \cup S^\sharp)}{f(\mathrm{OPT}' \cup S^\sharp) - f(T^{(t)} \cup S^\sharp)}\Big\} \\ \leq & \Big(\frac{f(\mathrm{OPT}' \cup S^\sharp)}{f(\mathrm{OPT}' \cup S^\sharp) - f(T^{(\bar{t})} \cup S^\sharp)}\Big)^{(\delta W + \delta^2 W)}\end{aligned} \tag{99}$$

Rearranging the terms, we have

$$\begin{aligned}f(T^{(\bar{t})} \cup S^\sharp) \geq & \Big(1 - \Big(\frac{\Phi(0)}{\Phi(\bar{t})}\Big)^{\frac{1}{\delta W + \delta^2 W}}\Big) f(\mathrm{OPT}' \cup S^\sharp) \\ \geq & \Big(1 - \Big(\frac{d[1 + \delta + \delta^2]}{e^{\delta W}}\Big)^{\frac{1}{\delta W + \delta^2 W}}\Big) f(\mathrm{OPT}' \cup S^\sharp),\end{aligned} \tag{100}$$

For the second inequality, we utilize the fact that $\Phi(\bar{t}+1) \geq e^{\delta W}$ and $\Phi(\bar{t}+1) \leq \Phi(\bar{t}) \cdot (1 + \delta + \delta^2)$. Note that

$$\begin{aligned}\Big(\frac{d[1 + \delta + \delta^2]}{e^{\delta W}}\Big)^{\frac{1}{\delta W + \delta^2 W}} = & e^{\frac{-1}{1+\delta}} \cdot d^{\frac{1}{\delta W + \delta^2 W}} \cdot (1 + \delta + \delta^2)^{\frac{1}{\delta W + \delta^2 W}} \\ \leq & \frac{(1 + 2\delta)(1 + \frac{2}{W})(1 + \frac{2 \log d}{\delta W})}{e} \leq \frac{1 + 15\delta}{e},\end{aligned}$$

where we use $e^x \leq 1 + 2x$ for $\forall x \in [0, 1]$, and $W = \frac{2 \log d}{\delta^2}$. The proof is complete. □