Submodular Maximization Subject to Uniform and Partition Matroids: From Theory to Practical Applications and Distributed Solutions

Solmaz S. Kia[1]

[1]University of California Irvine, Mechanical and Aerospace Engineering Dept., 4200 Engineering Gateway, Irvine, CA 92697, USA. email: solmaz@uci.edu; Date: June 2024

This article provides a comprehensive exploration of submodular maximization problems, focusing on those subject to uniform and partition matroids. Crucial for a wide array of applications in fields ranging from computer science to systems engineering, submodular maximization entails selecting elements from a discrete set to optimize a submodular utility function under certain constraints. We explore the foundational aspects of submodular functions and matroids, outlining their core properties and illustrating their application through various optimization scenarios. Central to our exposition is the discussion on algorithmic strategies, particularly the sequential greedy algorithm and its efficacy under matroid constraints. Additionally, we extend our analysis to distributed submodular maximization, highlighting the challenges and solutions for large-scale, distributed optimization problems. This work aims to succinctly bridge the gap between theoretical insights and practical applications in submodular maximization, providing a solid foundation for researchers navigating this intricate domain.

– Submodular Functions and Practical Applications: The chapter highlights how submodular maximization, especially with matroid constraints, serves as a cornerstone for solving various optimization problems in computer science, systems engineering, and beyond. Its applications range from data summarization and sensor placement to network resource management, demonstrating submodularity's wide-ranging impact.

– Algorithmic Strategies for Optimization: A significant portion of the chapter delves into crucial algorithmic approaches like the sequential greedy algorithm and the continuous greedy algorithm. These methods are not only foundational for achieving polynomial-time solutions to submodular maximization problems but also exhibit guaranteed performance bounds, making them vital tools in optimization theory.

– Adaptations for Distributed Environments: The discussion extends to the adaptation of submodular maximization strategies in distributed settings, reflecting the growing need for optimization solutions that cater to privacy concerns and the decentralized nature of modern computational and network systems.

– Innovative Research Directions: Lastly, the chapter concludes with an overview of emerging research areas within the field, such as deep submodular functions, online maximization, and fairness in optimization. These forefront research topics signal the ongoing evolution of submodular maximization theory and its expanding applicability to new, complex problems.

1 Introduction

In the modern landscape of intelligent systems, the demand for optimization algorithms that facilitate optimal decision-making for efficient resource allocation and policy making is at an all-time high. Optimization algorithms enabling these optimal decision-makings are expected to be computationally efficient and come with reasonable communication cost to deliver timely and efficient solutions for in-network operations. Some of the optimal resource allocation and policy-making problems faced for such systems are in the form of a combinatorial optimization problem

$$\max f(\mathcal{S}) \;\text{ subject to } \mathcal{S} \in \mathcal{F}(\mathcal{P}), \tag{1}$$

where the goal is to choose a subset $\mathcal{S}$ of discrete elements from a ground set $\mathcal{P}$ that maximizes a utility, described by function $f : 2^{\mathcal{P}} \to \mathbb{R}_{\geq 0}$ while respecting constraints captured by $\mathcal{F}$, the discrete set of feasible solutions.

Combinatorial optimization problems of the form (1) are often NP-hard, meaning that their complexity class are intrinsically harder than those that can be solved by a nondeterministic Turing machine in polynomial time. The quest thus is focused on finding suboptimal solutions with guaranteed optimality gap that measures the distance between the suboptimal solution and the optimal one. That is, we seek polynomial time suboptimal solutions with well-characterized optimality gap $0 < \alpha < 1$ satisfying

$$\alpha f(\mathcal{S}^\star) \leq f(\bar{\mathcal{S}}) \leq f(\mathcal{S}^\star), \tag{2}$$

where $\mathcal{S}^\star$ is the global maximizer of (1) and $\bar{\mathcal{S}}$ is the decision set delivered by the suboptimal solver.

For a certain class of optimization problem (1), namely problems with monotone submodular utility functions subject to matroid constraints, there is a significant body of work in the literature that provides suboptimal solutions with fully characterized optimality gap when the problem is solved in a centralized manner. In a distributed setting, either the utility function f or the elements of the ground set $\mathcal{P}$ in (1) are fragmented and distributed among the players/agents (devices/subsystems) of the optimization problem. Distributed operations bring about new set of challenges that can exasperate the established optimality gaps and require special considerations to manage the inter-agent communication costs. Additional concerns, such as scalability and the demand for privacy preservation, compound the complexity of distributed algorithms. Distributed algorithms for submodular maximization are not as extensively explored and established as the centralized algorithms.

Submodular maximization theory has been widely acknowledged as an effective approach for solving combinatorial resource allocation problems in the field of computer science. Many problems such as agglomerative clustering, exemplar-based clustering [1], categorical feature compression [2], recommender systems [3], search result diversification [4], data subset selection [5], social networks analysis [6] are cast as submodular maximization problems subject to matroid constraints. Classical combinatorial optimization problems like minimum spanning tree, global minimum cut, maximum matching, traveling salesman problem, max clique, max cut, set cover and knapsack, among the others, are also another set of combinatorial optimization problems that can be modeled as submodular maximization problems. These classical problems often appear as optimal decision-making mechanism in many system and engineering problems. Thus, robust solution to the NP-hard submodular maximization problems can serve as the backbone of optimal decision-making in modern industries such as transportation, supply chain, energy, finance, and scheduling. However, submodular maximization problems may not be as well known in the systems and control field as in computer science field. Submodular maximization generalizes many classic problems in combinatorial optimization and has recently found a wide range of applications in operational planning/task. Some example problems include sensor and actuator placement problems [7, 8, 9, 10, 11, 12], energy storage placement [13, 14], measurement scheduling [15], voltage control in smart grid [16], persistent monitoring via mobile robots [17].

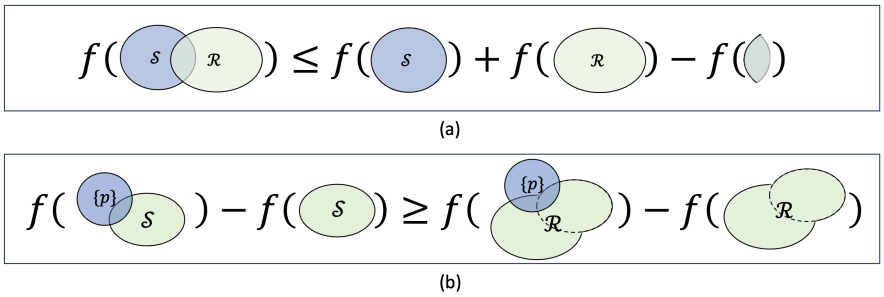

Figure 1: Properties of submodular functions.

This article will focus on solution approaches, application examples and distributed implementation of the two most well-know submodular maximization problems that are selecting elements from a finite discrete ground set $\mathcal{P}$ to maximize a submodular utility function, namely

- *Submodular maximization subject to uniform matroid constraint*

$$\mathcal{S}^\star = \arg\max_{\mathcal{S}\subset\mathcal{P}} f(\mathcal{S}) \;\text{ subject to } |\mathcal{S}| \leq \kappa, \tag{3}$$

  for a given $\kappa \in \mathbb{Z}_{\geq 1}$.

- *Submodular maximization subject to partition matroid constraint*

$$\mathcal{S}^\star = \arg\max_{\mathcal{S}\subset\mathcal{P}} f(\mathcal{S}) \;\text{ subject to } |\mathcal{S}\cap\mathcal{P}_i| \leq \kappa_i,\; i \in \{1,\cdots,N\} \tag{4}$$

  or a given $\kappa \in \mathbb{Z}_{\geq 1}$. Here, $\mathcal{P} \cup_{i=1}^{N} \mathcal{P}_i$ and $\mathcal{P}_i \cap \mathcal{P}_j = \emptyset$ for all $i \neq j$.

The remainder of this article is organized as follows. Section 2 provides a brief overview of the fundamental properties of the submodular functions and matroid constraints. Section 3 provides some example applications of submodular maximization theory. Section 4 reviews the existing centralized optimization algorithms to solve submodular maximization problems. Section 5 reviews various distributed submodular maximization problems, the challenges in solving these problems and some existing algorithms and their design methods. And finally, Section 6 gives our concluding remarks.

2 Submodular functions and matroid constraints

In what follows, we introduce what submodular function and matroid constraints are. We provide an overview of the relevant components of these concepts pertaining to the two submodular maximization problems (3) and (4), which we want to discuss in this article. Interested readers can consult references such as [18] and [19] for a more detailed and comprehensive overview.

Consider a finite ground set $\mathcal{P}$ of n discrete elements, which without loss of generality, is assumed to be $\mathcal{P} = \{1,\cdots,n\}$.

Definition 1 (Normal set function)**.** *A set function $f : 2^{\mathcal{P}} \to \mathbb{R}_{\geq 0}$ is said to be normal if $f(\emptyset) = 0$.*

Definition 2 (Monotone increasing set function)**.** *A set function $f : 2^{\mathcal{P}} \to \mathbb{R}_{\geq 0}$ is said monotone increasing if and only if*

$$f(\mathcal{S}) \geq f(\mathcal{R}), \quad \forall \mathcal{R} \subset \mathcal{S} \subset \mathcal{P}.$$

Definition 3 (Submodular function)**.** *The function* $f : 2^{\mathcal{P}} \to \mathbb{R}_{\geq 0}$ *is submodular if and only if*

$$f(\mathcal{R}) + f(\mathcal{S}) \geq f(\mathcal{R} \cup \mathcal{S}) + f(\mathcal{R} \cap \mathcal{S}), \quad \forall\, \mathcal{S}, \mathcal{R} \in \mathcal{P}. \tag{5}$$

The relationship (6) fundamentally highlights how submodular functions evaluate redundancy or overlap between sets. Specifically, (6) suggests that if a utility function is submodular, the loss of utility at joint evaluation of any two subsets, i.e., at $\mathcal{R} \cup \mathcal{S}$, of the ground set compared to the sum of the utility of the individual subsets $\mathcal{R}$ and $\mathcal{S}$ is more than or equal to the utility of the overlap $\mathcal{R} \cap \mathcal{S}$, see Fig. 2(a). Thus, the relation elegantly penalizes redundancy or overlap within the sets' arguments, quantitatively expressing that the function f attributes a lower incremental value to shared or overlapping elements between $\mathcal{R}$ and $\mathcal{S}$, effectively encouraging diversity within the sets. When this loss is equal to the utility of the overlap the set function f is said to be *modular*.

Definition 4 (Modular set function)**.** *The function* $f : 2^{\mathcal{P}} \to \mathbb{R}_{\geq 0}$ *is submodular if and only if*

$$f(\mathcal{R}) + f(\mathcal{S}) = f(\mathcal{R} \cup \mathcal{S}) + f(\mathcal{R} \cap \mathcal{S}), \quad \forall\, \mathcal{S}, \mathcal{R} \in \mathcal{P}. \tag{6}$$

If f is modular then there is a weight function $w : \mathcal{P} \to \mathbb{R}_{\geq 0}$ such that $f(\mathcal{R}) = \sum_{r \in \mathcal{R}} w(r)$.
For a set value function $f : 2^{\mathcal{P}} \to \mathbb{R}_{\geq 0}$, the *marginal gain*,

$$\Delta_f(p|\mathcal{S}) = f(\mathcal{S} \cup \{p\}) - f(\mathcal{S}), \tag{7}$$

quantifies how much the function f, which maps sets to real numbers, increases as a result of adding an element $p \in \mathcal{P}$ to $\mathcal{S} \subset \mathcal{P}$. Marginal gain often is also called the *discrete derivative* of f at $\mathcal{S}$ with respect to p. In the context of f being a utility function, marginal gain measures the additional value or utility obtained by including the element p into the set $\mathcal{S}$. Marginal gain is a crucial concept in optimization, especially when dealing with set functions, because it helps in understanding the benefit of including an additional element in a set given the current context, i.e., the elements already selected in the set $\mathcal{S}$. Submodular functions are characterized by their diminishing return, i.e., a set function $f : 2^{\mathcal{P}} \to \mathbb{R}_{\geq 0}$ is submodular if and only if

$$f(\mathcal{S} \cup \{p\}) - f(\mathcal{S}) \geq f(\mathcal{R} \cup \{p\}) - f(\mathcal{R}), \quad \forall\, \mathcal{S} \subset \mathcal{R} \subset \mathcal{P},\ p \in \mathcal{P} \backslash \mathcal{R}; \tag{8}$$

see Fig. 2(b). The diminishing return property makes submodular functions particularly appealing for problems where resources are being allocated, tasks are being assigned, or selections are being made from a set, and the goal is to maximize some notion of utility or coverage subject to constraints (like size, budget, or time). This property ensures that as solutions grow, the incremental gains diminish, leading to natural "saturation" points that help in defining "optimal" or "near-optimal" solutions to such problems. For the special case of modular functions, the utility of adding an element to a set is independent of the set's current composition—paralleling linear functions in numerical optimization. That is for modular functions

$$f(\mathcal{S} \cup \{p\}) - f(\mathcal{S}) = f(\mathcal{R} \cup \{p\}) - f(\mathcal{R}), \quad \forall\, \mathcal{S} \subset \mathcal{R} \subset \mathcal{P},\ p \in \mathcal{P} \backslash \mathcal{R}; \tag{9}$$

For submodular functions, the total *curvature* $c \in [0, 1]$ of the function, defined as

$$c = 1 - \min_{\mathcal{S} \subset \mathcal{P},\ p \notin \mathcal{S}} \frac{\Delta_f(p|\mathcal{S})}{\Delta_f(p|\emptyset)}. \tag{10}$$

This definition aims to capture the least proportional increase in the function value when adding an element to any set, relative to adding it to the empty set. For modular functions,(9) implied that $c = 0$. Thus, the total curvature c of a submodular function is a measure of how far the function is from being linear. On the other hand, if $c = 1$, this typically indicates the case where the function experiences the greatest reduction in marginal gains for adding elements to a set. Specifically, it signifies that there is

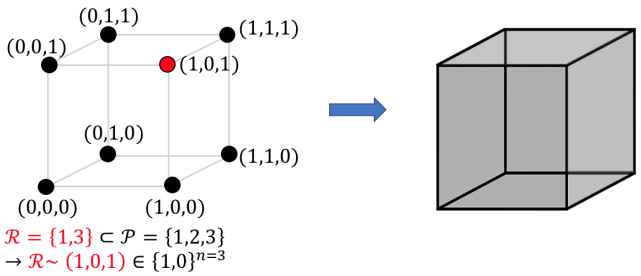

Figure 2: Multilinear extension extends a submodular function $f(\mathcal{R})$ to the continuous space defined on hypercube $[0,1]^n$. $F(\boldsymbol{x})$ agrees with $f(\mathcal{R})$ on the vertices of the hypercube (for integral $\boldsymbol{x}$).

at least one situation where adding an element to a larger set grants no additional value compared to adding it to some smaller set, thus reflecting a scenario with the maximum possible loss in marginal gain as the set size increases.

It is worth noting that submodularity is preserved under taking nonnegative linear combinations, i.e., if $f_1, \cdots, f_m : 2^{\mathcal{P}} \rightarrow \mathbb{R}_{\geq 0}$ is submodular then $f(\mathcal{S}) = \sum_{i=1}^{m} \alpha_i f_i(\mathcal{S})$ is submodular for any $\alpha_1, \cdots, \alpha_m \in \mathbb{R}_{\geq 0}$. Monotone submodularity is also preserved under truncation, i.e., if $f : 2^{\mathcal{P}} \rightarrow \mathbb{R}_{\geq 0}$ is submodular, then so is $f(\mathcal{S}) = \min\{f(\mathcal{S}), c\}$ for any constant $c \in \mathbb{R}$. Submodularity is also preserved when we take the residual, i.e., if $f : 2^{\mathcal{P}} \rightarrow \mathbb{R}_{\geq 0}$ is submodular, and $\mathcal{R}, \mathcal{S} \subset \mathcal{P}$ are any disjoint sets, then the residual $g : 2^{\mathcal{R}} \rightarrow \mathbb{R}_{\geq 0}$ defined as $g(\mathcal{Q}) = f(\mathcal{S} \cup \mathcal{Q}) - f(\mathcal{Q})$ for any $\mathcal{Q} \subset \mathcal{R}$ is submodular.

Set functions with ground set of n elements can be cast as a vector function defined on vertices of hypercube $\{0,1\}^n$ as shown in Fig. 2. [20] proposed to use the *multilinear extension* of submodular set functions to extend the submodular utility function $f : 2^{\mathcal{P}} \rightarrow \mathbb{R}_{\geq 0}$, which is defined on the vertices of the $n-$dimensional hypercube $\{0,1\}^n$, to the continuous multilinear function $F(\boldsymbol{x})$ defined as

$$F(\boldsymbol{x}) = \sum_{\mathcal{R} \subset \mathcal{P}} f(\mathcal{R}) \prod_{p \in \mathcal{R}} [\boldsymbol{x}]_p \prod_{p \notin \mathcal{R}} (1 - [\boldsymbol{x}]_p), \quad \boldsymbol{x} \in [0,1]^n, \tag{11}$$

in the continuous space, where $[\boldsymbol{x}]_p$ is the pth element of $\boldsymbol{x}$. The multilinear extension agrees with $f(\mathcal{S})$ ar the integral values of $\boldsymbol{x}$. More precisely, for any $\mathcal{R} \subset \mathcal{P}$, we have $F(\mathbf{1}_{\mathcal{R}}) = f(\mathcal{R})$ where $\mathbf{1}_{\mathcal{R}}$ is a vector whose elements corresponding to set $\mathcal{R}$ are 1 and the rest are 0. This multilinear extension plays a crucial role in solving submodular maximization problems through a technique known as continuous relaxation, which will be elaborated upon in Section (4).

Multilinear extension essentially allows each element's inclusion in a set to be considered as a probability, smoothing the transition from discrete to continuous space for optimization. More precisely, interpreting $\boldsymbol{x} \in [0,1]^n$ as *the membership probability vector* and positing $\mathcal{R}_{\boldsymbol{x}} \subset \mathcal{P}$ as a random set where each element $p \in \mathcal{P} = \{1, \cdots, n\}$ is included in $\mathcal{R}_{\boldsymbol{x}}$ with probability $[\boldsymbol{x}]_p$, equation (11) translates to

$$F(\boldsymbol{x}) = \mathbb{E}[f(\mathcal{R}_{\boldsymbol{x}})]. \tag{12}$$

Vondrák's work reveals that the first derivative of $F(\boldsymbol{x})$ with respect to $[\boldsymbol{x}]_p$, as shown in

$$\frac{\partial F}{\partial [\boldsymbol{x}]_p}(\boldsymbol{x}) = \mathbb{E}[f(\mathcal{R}_{\boldsymbol{x}} \cup \{p\}) - f(\mathcal{R}_{\boldsymbol{x}} \setminus \{p\})], \tag{13}$$

consistently remains non-negative for monotone increasing submodular functions. Furthermore, the analysis extends to the second derivative of $F(\boldsymbol{x})$, indicating

$$\frac{\partial^2 F}{\partial [\boldsymbol{x}]_p \partial [\boldsymbol{x}]_q}(\boldsymbol{x}) = \mathbb{E}[f(\mathcal{R}_{\boldsymbol{x}} \cup \{p,q\}) - f(\mathcal{R}_{\boldsymbol{x}} \cup \{q\} \setminus \{p\}) - f(\mathcal{R}_{\boldsymbol{x}} \cup \{p\} \setminus \{q\}) + f(\mathcal{R}_{\boldsymbol{x}} \setminus \{p,q\})], \tag{14}$$

which, by virtue of submodularity, guarantees $\frac{\partial^2 F}{\partial [\boldsymbol{x}]_p \partial [\boldsymbol{x}]_q}(\boldsymbol{x}) \leq 0$.

The multilinear extension function, as detailed in equation (11), is positioned within the broader category of continuous Diminishing Returns (DR) submodular functions. These functions constitute a distinct subclass within the spectrum of non-convex/non-concave continuous functions. The problem of optimizing DR-submodular functions over a convex set has attracted considerable interest in both the machine learning and theoretical computer science communities due to its wide applicability in data mining, machine learning, economics, and statistics. Readers keen on a deeper exploration of this domain may find valuable insights in seminal work such as [21] or recent work such as [22].

Having reviewed the submodular functions, we next introduce *matroids*, which play a crucial role in addressing complex problems such as submodular maximization, particularly when constrained by a uniform matroid, a fundamental concept in combinatorial optimization. Matroids provide a powerful abstract framework that generalizes the notion of linear independence from vector spaces to more versatile combinatorial structures. A *matroid* $\mathcal{M}$ over a finite ground set $\mathcal{P}$ is a collection of independent subset of $\mathcal{P}$. When speaking of a matroid, we denote it as a pair $(\mathcal{P}, \mathcal{M})$, where $\mathcal{M} \subset 2^{\mathcal{P}}$.

Definition 5. *We say $\mathcal{M} \subset 2^{\mathcal{P}}$ is a matroid if and only if*

1. *$\mathcal{M} \neq \emptyset$; if $\mathcal{R} \in \mathcal{M}$ and $\mathcal{S} \subset \mathcal{R}$ then $\mathcal{S} \in \mathcal{M}$ (downward closed; independence system property)*
2. *If $\mathcal{S}, \mathcal{R} \in \mathcal{M}$ and $|\mathcal{S}| > |\mathcal{R}|$ then there exists a $p \in \mathcal{S} \backslash \mathcal{R}$ such that $\mathcal{R} \cup \{p\} \in \mathcal{M}$ (augmentation property).*

Among matroids, the *uniform* and *partition* types stand out for their widespread application.

Definition 6 (Uniform Matroid)**.** *For some $\kappa \in \mathbb{Z}_{>0}$, the* uniform matroid *over a ground set $\mathcal{P}$ is defined as $(\mathcal{P}, \mathcal{M})$, where $\mathcal{M} = \left\{\mathcal{S} \subset \mathcal{P} \,\middle|\, |\mathcal{S}| \leq \kappa\right\}$.*

A uniform matroid is intuitively understood as allowing any combination of elements, as long as their number does not exceed a specified limit.

Definition 7 (Partition Matroid)**.** *For some $\kappa_i \in \mathbb{Z}_{>0}$, $i \in \{1, \cdots, N\}$, the* partition matroid *over a ground set $\mathcal{P}$ is defined as $(\mathcal{P}, \mathcal{M})$, where $\mathcal{M} = \left\{\mathcal{S} \subset \mathcal{P} \,\middle|\, |\mathcal{S} \cap \mathcal{P}_i| \leq \kappa_i,\ i \in \{1, \cdots, N\}\right\}$, where $\mathcal{P}_1, \cdots, \mathcal{P}_N$ are non-overlapping partition of $\mathcal{P}$, i.e., $\mathcal{P} = \cup_{i=1}^{N} \mathcal{P}_i$ and $\mathcal{P}_i \cap \mathcal{P}_j = \emptyset$ for any $i \neq j$.*

A partition matroid intuitively groups items into distinct categories, allowing only a specified maximum number of selections from each group.

3 Application examples

Many discrete optimal resource allocation problems involve strategic choices to maximize utility, including goals like maximizing entropy, mutual information, symmetric mutual information, cut capacity, weighted coverage, and facility location. As discussed in [23], these utilities have been characterized as submodular functions, permitting the framing of combinatorial optimization involving these utilities as submodular maximization problems. However, adapting some problems to conform to the standard submodular maximization framework often requires pre-processing steps that adjust the decision variables and the ground set to reconfigure the constraints into a matroid format. In what follows, we present some classic examples of submodular maximization problems, which have broad relevance in various decision-making contexts. We also demonstrate through application examples how problem settings can be restructured to align with one of the standard forms of submodular maximization, particularly those subject to uniform or partition matroids.

Exemplar-based clustering, as introduced by [24], aims to identify a subset of exemplars that optimally represent a large dataset by solving the k-medoid problem. This method seeks to minimize the

cumulative pairwise dissimilarities between chosen exemplars $\mathcal{P}$ and dataset elements $\mathcal{D}$, formulated as

$$L(\mathcal{R}) = \sum\nolimits_{p\in\mathcal{R}} \min_{d\in\mathcal{D}} \text{dist}(p, d), \tag{15}$$

for any subset $\mathcal{R} \subset \mathcal{P}$, where $\text{dist}(p, d) \geq 0$, not necessarily symmetric or obeying the triangle inequality, defines the dissimilarity, or distance, between elements. In exemplar clustering, we seek a subset $\mathcal{R} \subset \mathcal{P}$, subject to constraints like cardinality, that minimizes L. This problem can be posed as a submodular maximization problem by defining the utility function

$$f(\mathcal{R}) = L(\{d_0\}) - L(\mathcal{R} \cup \{d_0\}), \tag{16}$$

where d_0 is an added hypothetical auxiliary element in the exemplar space. This utility function (16) quantifies the reduction in the loss associated with the active set versus the loss with merely the phantom placement location, and maximizing this function corresponds to minimizing the loss (15). The utility function (16) is shown to be submodular and monotonically increasing [25].

Consider now an information harvesting problem where we aim to collect information from a countable set of sources $\mathcal{D}$, distributed across a two-dimensional finite space $\mathcal{W} \subset \mathbb{R}^2$. Suppose we want to deploy $\kappa \in \mathbb{Z}_{>1}$ data harvester devices at a set of pre-specified data retrieval points $\mathcal{B}$ within $\mathcal{W}$, with the condition that $|\mathcal{B}| > \kappa$. We assume the most effective information transfer from an information point $d \in \mathcal{D}$ to a harvester device located at $b \in \mathcal{B}$ occurs when the distance between b and d is minimized. Therefore, for each information point $d \in \mathcal{D}$, the nearest information retrieval point $b \in \mathcal{B}$ with a deployed device is assigned to harvest information. The optimal deployment can be achieved by framing the problem as an exemplar clustering problem with the submodular utility function (16), where the distance $\text{dist}(b, d) = \|b - d\|$ represents the Euclidean distance between point $b \in \mathcal{B}$ and data point $d \in \mathcal{D}$. The resulted problem becomes a submodular maximization subject to a uniform matroid, as we aim to select κ elements from the ground set $\mathcal{B}$ that maximize utility. In a multi-agent variation of this problem, we envision a scenario where a group of N agents each aims to deploy $\kappa_i \in \mathbb{Z}_{\geq 1}$, $i \in \{1, \cdots, N\}$, data harvesters in $\mathcal{W}$. Every agent $i \in \{1, \cdots, N\}$ has access to a subset $\mathcal{B}_i \subset \mathcal{B}$ of retrieval points, where $\kappa_i < |\mathcal{B}_i|$; the sets of potential retrieval points among agents can overlap. In this setup, the optimal deployment strategy still involves maximizing the submodular utility function (16), with $\text{dist}(b, d) = \|b - d\|$. However, the matroid constraint is now a partition matroid, with non-overlapping local ground sets defined for each agent $i \in \{1, \cdots, N\}$ as $\mathcal{P}_i = \{(i, b) \,|\, b \in \mathcal{B}_i\}$.

Next, we shift our focus to another pivotal concept in the landscape of discrete optimization: *the Optimal Welfare problem* [20]. In the Optimal Welfare problem m items in a discrete set $\mathcal{Q} = \{1, \cdots, m\}$ should be distributed among N agents such that the sum of local utilities of the agents is maximized [26], i.e,

$$\max_{\mathcal{S}_1,\cdots,\mathcal{S}_N\subset\mathcal{Q}} f(\cup_{i=1}^{N}\mathcal{S}_i) = \sum\nolimits_{i=1}^{N} f_i(\mathcal{S}_i), \text{ s.t.,}$$
$$\mathcal{S}_i \cap \mathcal{S}_j = \emptyset, \;\; i \neq j, \;\; i, j \in \{1, \cdots, N\};$$

The local cost at each agent, f_i, $i \in \{1, \cdots, N\}$ is, normal, monotone increasing and submodular. We know that the sum of submodular functions is submodular, however, optimal Welfare in its original form does not conform to a standard submodular maximization subject to matroid condition. Luckily, this problem can be cast as submodular maximization subject to partition matroid by defining the subsets $\mathcal{P}_i$'s in (4) as $\mathcal{P}_i = \{(i, j) \,|\, j \in \{1, \cdots, N\}\}$, $i \in \{1, \cdots, m\}$ and setting $\kappa_i = 1$, which limits the assignment of item i to only one agent. In this problem setting, $f^i(\mathcal{R}^i) = f^i(\bar{\mathcal{R}}^i)$ where $\bar{\mathcal{R}}^i = \{j \in \mathcal{Q} \,|\, (i, j) \in \mathcal{R}\}$.

Let us now transition to another compelling area of study within set function maximization: sensor placement problems. These scenarios are especially prevalent when operational constraints dictate predetermined locations for deploying sensors. In these problems, we deal with the task of finding

the best possible locations for sensors such that best performance is achieved by utility maximization via the limited resources/sensors available. A pertinent instance of this type of problem is detailed in [27], focusing specifically on sensor placement within traffic networks in the absence of routing information. The traffic network is represented by a graph $\mathcal{G} = (\mathcal{V}, \mathcal{L})$, where $\mathcal{V}$ is the set of network nodes (e.g., intersections), and $\mathcal{L} \subset \mathcal{V} \times \mathcal{V}$ is the set of network links. The traffic network is modeled from a macroscopic point of view, in which for each link $l \in \mathcal{L}$, f_l is the amount of vehicular flow on link l (vehicles per unit of time). The sensor placement objective is to locate sensors, e.g., loop detectors, such that the maximum number of link flows can be identified from a set of measurements at limited number of the nodes. In such a scenario, the goal is to maximize the identifiability of link flows provided that a set of κ flow measurements are obtained. Let $\mathcal{L}_m$ be the set of links with flow measurement sensors that deliver measurements $y_l = f_l$ for $l \in \mathcal{L}_m$. Furthermore, we know that at each node, flow conservation must hold. Thus the flow equations consist of $|\mathcal{L}_m| + |\mathcal{V}|$ linear equations, which can be formalized as $A\, f = b$, where $A \in \mathbb{R}^{(|\mathcal{L}_m|+|\mathcal{V}|)\times|\mathcal{L}|}$ is the matrix of linear equations, f is the vector flow across the links and b is the concatenate measurement vectors and the vector $0_{|\mathcal{L}|}$, the right hand value of the conservation of flow equations. Flows considered unidentifiable reside within the null space of matrix A. Accordingly, [27] casts the problem of maximizing the number of identifiable flows as reducing the the null space of A, or, equivalently,

$$\max_{\mathcal{L}_m \subset \mathcal{L}} \text{rank}(A), \quad \text{s.t. } |\mathcal{L}_m| \leq \kappa. \tag{17}$$

By showing that rank(A) is a monotone increasing submodular function, [27] establishes that (17) is an incidence of the standard submodular maximization subject to uniform matroid.

In some problem settings, it might not be immediately evident that the problem at hand can be framed as set function optimization, especially in terms of submodular maximization subject to matroid constraints. An illustrative example of such a scenario is provided in [17], which addresses a multi-agent persistent monitoring problem over a connected graph through a submodular maximization framework. Specifically, [17] examines the persistent monitoring of a set of finite $\mathcal{V}$ interconnected geographical nodes by a finite set of mobile sensors/agents $\mathcal{A} = \{1, \cdots, N\}$, where $|\mathcal{V}| \gg |\mathcal{A}|$. The mobile agents are required to navigate through a set of prespecified edges $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$, such as aerial or ground corridors, to visit the nodes. The travel time $t_{\mathsf{a},e}$ for each agent $\mathsf{a} \in \mathcal{A}$ across each edge $e \in \mathcal{E}$ is known. Each node $v \in \mathcal{V}$ is assigned a reward function,

$$R_v(t) = \begin{cases} 0, & t = \bar{t}_v, \\ \psi_v(t - \bar{t}_v), & t > \bar{t}_v, \end{cases} \tag{18}$$

where $\psi_v(t)$ is a nonnegative, concave, and increasing function of time, and $\bar{t}_v$ represents the most recent time node v has been visited by an agent. In scenarios such as data harvesting or health monitoring, $\psi_v(\cdot)$ might represent the weighted idle time of node v, or in event detection, it might indicate the probability of at least one event occurring during the intervals between visits. When any agent $\mathsf{a} \in \mathcal{A}$ arrives at any time $\bar{t} \in \mathbb{R}_{>0}$ at node $v \in \mathcal{V}$, the agent immediately scans the node, and the reward $R_v(\bar{t})$ is scored for the patrolling team $\mathcal{A}$; subsequently, $\bar{t}_v$ of node v in (18) is updated to $\bar{t}$. If more than one agent arrives at node $v \in \mathcal{V}$ and scans it at the same time $\bar{t}$, the reward collected for the team is still $R_v(\bar{t})$. The goal of optimal monitoring involves designing a dispatch policy that determines the tours sequence of interconnected nodes to be visited and the timings of these visits), and the specific agents allocated to each tour to achieve the maximum total reward for the team over the mission horizon T. A dispatch policy for an agent $\mathsf{a} \in \mathcal{A}$ over the given mission time horizon is denoted by the tuple $\mathbf{p}_\mathsf{a} = (\mathbf{V}_\mathsf{a}, \mathbf{T}_\mathsf{a}, \mathsf{a})$, where $\mathbf{V}_\mathsf{a}$ and $\mathbf{T}_\mathsf{a}$ specify the inter-connected nodes and their corresponding visitation times (tour) assigned to agent a. Given the policies for all agents in the team, $\mathcal{S} = \{\mathbf{p}_1, \cdots, \mathbf{p}_N\}$, the collective reward of the agents is computed as

$$\mathsf{R}(\mathcal{S}) = \sum\nolimits_{\forall\, \mathbf{p} \in \mathcal{S}} \sum\nolimits_{l=1}^{\mathsf{n}_\mathsf{p}} R_{\mathbf{V}_\mathsf{p}(l)}(\mathbf{T}_\mathsf{p}(l)), \tag{19}$$

where n_{p} is the number of nodes to be visited when policy $\mathsf{p} \in \mathcal{S}$ is implemented. The optimal dispatch design, then, is to find $\mathcal{S}$ that maximizes the total reward $\mathsf{R}(\mathcal{S})$. Notably, overlapping tours, i.e., arrivals at some node(s) v at the same time $\bar{t}$ by more than one agent, are not desired as only one reward $R_v(\bar{t})$ will be collected for the team. For example, for a group of five agents, given any two sets of policies such as $\mathcal{S}_1 = \{\mathsf{p}_1, \mathsf{p}_2\}$ and $\mathcal{S}_2 = \{\mathsf{p}_3, \mathsf{p}_4, \mathsf{p}_5\}$, we know that

$$\mathsf{R}(\mathcal{S}_1) + \mathsf{R}(\mathcal{S}_2) \geq \mathsf{R}(\mathcal{S}_1 \cup \mathcal{S}_2).$$

Indeed, [17] shows that the collective reward function is a submodular set function. It then proceeds to determine the optimal dispatch policy by solving the submodular set function maximization subject to a partition matroid defined by

$$\max_{\mathcal{S}\subset\mathcal{P}} \mathsf{R}(\mathcal{S}), \quad \text{s.t. } |\mathcal{S} \cap \mathcal{P}_{\mathsf{a}}| \leq 1, \ \forall \mathsf{a} \in \mathcal{A} = \{1, \cdots, N\}, \tag{20}$$

where $\mathcal{P}_{\mathsf{a}}$ is the set of all feasible policies for agent a, and $\mathcal{P} = \cup_{\mathsf{a}\in\mathcal{A}}\mathcal{P}_{\mathsf{a}}$. When T is large, constructing the entire set of feasible policies for each agent can become prohibitively costly. To manage this cost, [17] proposes a moving horizon approach where the problem defined in (20) is considered over a shorter planning horizon. The objective then becomes to identify the optimal policy for this shorter time frame, apply a portion of the policy, and iteratively repeat this process for the remainder of the mission horizon. This strategy enables also a more adaptable and efficient response to the dynamic aspects of the monitoring task.

4 Solving submodular maximization subject to matroid constraints

Research on problems involving the maximization of submodular functions dates back to the work of Nemhauser, Wolsey, and Fisher in the 1970's [28, 29, 30]. For monotone-increasing submodular maximization subject to uniform matroid constraint, the celebrated result by Nemhauser et al. [29] showed that the simple *sequential greedy (SG) algorithm*, which starts at $\mathcal{S}_0 = \emptyset$ and iterates according to

$$\mathcal{S}_i = \mathcal{S}_{i-1} \cup \max_{p\in\mathcal{P}\setminus\mathcal{S}_{i-1}} \Delta_f(p|\mathcal{S}_{i-1}), \quad i \in \{1, \cdots, \kappa\}, \tag{21}$$

yields an optimality gap, as defined in (2), of $\alpha_{\text{uniform}} = 1 - \frac{1}{\mathrm{e}} \approx 0.63$, i.e., $f(\mathcal{S}_{\text{SG}}) \geq (1 - \frac{1}{\mathrm{e}})\, OPT$, where $\mathcal{S}_{\text{SG}} = \mathcal{S}_\kappa$ and $OPT = f(\mathcal{S}^\star)$. This optimality gap is established elegantly by invoking the monotone increasing and submodularity of the utility function as demonstrated below. In what follows let $\mathcal{S}^\star = \{s_1^\star, \cdots, s_\kappa^\star\}$.

$$\begin{aligned}
f(\mathcal{S}^\star) &\leq f(\mathcal{S}^\star \cup \mathcal{S}_i) && \text{(monotone increasing)}\\
&= f(\mathcal{S}_i) + \textstyle\sum_{j=1}^{\kappa} \Delta_f(s_j^\star|\mathcal{S}_i \cup \{s_1^\star, \cdots, s_{j-1}^\star\}) && \text{(telescoping sum)}\\
&\leq f(\mathcal{S}_i) + \textstyle\sum_{j=1}^{\kappa} \Delta_f(s_j^\star|\mathcal{S}_i) && \text{(submodularity)}\\
&\leq f(\mathcal{S}_i) + \textstyle\sum_{j=1}^{\kappa} (f(\mathcal{S}_{i+1}) - f(\mathcal{S}_i)) && \text{(greedy selection to build } \mathcal{S}_{i+1})\\
&= f(\mathcal{S}_i) + \kappa\,(f(\mathcal{S}_{i+1}) - f(\mathcal{S}_i)).
\end{aligned}$$

Now defining $\delta_i = f(\mathcal{S}^\star) - f(\mathcal{S}_i)$ and recalling that $f(\emptyset) = 0$, we can show that

$$f(\mathcal{S}^\star) - f(\mathcal{S}_i) \leq (1 - \frac{1}{\kappa})^i (f(\mathcal{S}^\star) - f(\mathcal{S}_0)) \leq \mathrm{e}^{i/\kappa} f(\mathcal{S}^\star) \rightarrow (1 - \frac{1}{\mathrm{e}}) f(\mathcal{S}^\star) \leq f(\mathcal{S}_{\text{SG}}).$$

If the total curvature c, defined in (10), of the utility function is known, the optimality gap can be precisely adjusted by incorporating the curvature. Detailed analysis of the greedy algorithm, contingent on c, was provided in [31], which established that $\alpha_{\text{uniform}} = \frac{1}{c}\left(1 - \frac{1}{e^c}\right)$. It is important to remember

that curvature c quantifies the diminishing returns characteristic of a set function. A curvature of $c = 0$ signifies a modular function, for which $f(\{p_1, p_2\}) = f(\{p_1\}) + f(\{p_2\})$, where $p_1, p_2 \in \mathcal{P}$. In this case, $\alpha_{\text{uniform}} = 1$, indicating that the sequential greedy algorithm achieves the optimal solution within a finite number of steps for modular functions, a result applicable under any matroid constraint. Conversely, a curvature of $c = 1$ suggests the presence of at least one element that, under certain conditions, does not enhance the function f. In scenarios where the total curvature remains undetermined, adopting a conservative approach by setting $c = 1$ is prudent to prepare for the most challenging case.

Owing to the intrinsic matroid characteristics such as downward-closeness and augmentation, the sequential greedy algorithm, initially developed for problems constrained by uniform matroids, can indeed be adapted to tackle submodular maximization under any matroid constraint. These fundamental properties ensure that the algorithm maintains the feasibility of the solution at each step and guarantees that the provided solution is both suboptimal and feasible, thereby highlighting the broad applicability and efficacy of the greedy method across a multitude of optimization scenarios. It has been demonstrated that, for monotone increasing submodular utility functions within an optimization problem (1) where $\mathcal{F}(\mathcal{P})$ is a matroid, i.e., $\mathcal{F}(\mathcal{P}) = (\mathcal{P}, \mathcal{M})$, the sequential greedy algorithm

$$\mathcal{S}_i = \mathcal{S}_{i-1} \cup \max_{p \notin \mathcal{S}_{\text{SG}}, \mathcal{S}_{i-1} \cup \{p\} \in \mathcal{M}} \Delta_f(p|\mathcal{S}_{i-1}), \tag{22}$$

proceeds until no additional p makes $\mathcal{S}_{i-1} \cup \{p\}$ a feasible selection within $\mathcal{M}$. This process is guaranteed to yield a solution with an optimality gap of $\alpha = \frac{1}{2}$, that is, $f(\mathcal{S}_{\text{SG}}) \geq \frac{1}{2} \cdot OPT$, where $\mathcal{S}_{\text{SG}} = \mathcal{S}_\kappa$ and $OPT = f(\mathcal{S}^\star)$. Specifically, in the context of submodular maximization subject to a partition matroid (4), the guaranteed optimality gap achieved by the sequential greedy algorithm is documented as $\alpha_{\text{partition}} = \frac{1}{2}$.

For submodular maximization subject to a partition matroid (4), the sequential greedy algorithm can be implemented more efficiently by making sequential greedy choices starting from $\mathcal{P}_1$, choosing κ_1 elements from this set, then fixing these choices before proceeding to set $\mathcal{P}_2$ to choose the κ_2 elements, and so on, until the last κ_N choices are made after fixing the first $\kappa_1, \ldots, \kappa_{n-1}$ choices. The process can be explained as starting with $\mathcal{S}_0 = \emptyset$ and $\mathcal{R}_0 = \emptyset$ and iterating over

$$\begin{aligned} \mathcal{S}_j &= \mathcal{S}_{j-1} \cup \mathcal{R}_{\kappa_i}, \quad j = \{1, \cdots, N\}, \\ \mathcal{R}_i &= \mathcal{R}_{i-1} \cup \max_{p \in \mathcal{P}_i \setminus \mathcal{R}_{i-1}} \Delta_f(p|\mathcal{R}_{i-1}), \quad i \in \{1, \cdots, \kappa_i\}; \end{aligned} \tag{23}$$

this process outputs $\mathcal{S}_{\text{SG}} = \mathcal{S}_N$. To illustrate the role of submodularity in setting the optimality gap of $\alpha_{\text{partition}} = \frac{1}{2}$ for the sequential greedy algorithm (23), next, we provide a concise overview of the proof. For simplicity, let us assume $\kappa_i = 1$, indicating our goal to select one element from each subset $\mathcal{P}_i$, for all $i \in \{1, \cdots, N\}$ in (4). Mirroring the steps from the proof of the optimality gap in submodular maximization under a uniform matroid, we can establish, given that the utility function is normal, monotone increasing, and submodular, the following inequality:

$$f(\mathcal{S}^\star) \leq f(\mathcal{S}_N) + \sum\nolimits_{j=1}^{N} \Delta_f(s_j^\star|\mathcal{S}_N), \tag{24}$$

where $s_1^\star, \cdots, s_N^\star$ are the elements of $\mathcal{S}^\star$, that is, $\mathcal{S}^\star = \{s_1^\star, \cdots, s_N^\star\}$. Let $\bar{s}_1^\star, \cdots, \bar{s}_N^\star$ be the elements of $\mathcal{S}_N$, implying $\mathcal{S}_N = \{\bar{s}_1^\star, \cdots, \bar{s}_N^\star\}$. Note that $s_i^\star$ and $\bar{s}_i^\star$ belong to $\mathcal{P}_i$ for each $i \in \{1, \cdots, N\}$. Thus, we can write

$$\begin{aligned} \Delta_f(s_i^\star|\mathcal{S}_N) &= f(\mathcal{S}_{i-1} \cup \{\bar{s}_i, \cdots, \bar{s}_N\} \cup \{s_i^\star\}) - f(\mathcal{S}_{i-1} \cup \{\bar{s}_i, \cdots, \bar{s}_N\}) \\ &\leq f(\mathcal{S}_{i-1} \cup \{s_i^\star\}) - f(\mathcal{S}_{i-1}) && \text{(by submodularity)} \\ &\leq f(\mathcal{S}_i) - f(\mathcal{S}_{i-1}) && \text{(by greedy selection to construct } \mathcal{S}_i). \end{aligned}$$

Therefore, from (24), we conclude that $f(\mathcal{S}^\star) \leq 2f(\mathcal{S}_{\text{SG}})$, thus establishing that $\alpha_{\text{partition}} = \frac{1}{2}$. It is important to note that this optimality gap represents the worst-case scenario and is independent of the

order in which the disjoint components $\mathcal{P}_1, \cdots, \mathcal{P}_N$ of the ground set $\mathcal{P}$ are engaged in the sequential greedy selection process outlined in (23).

It is important to note that while the sequential greedy algorithm delivers an optimality gap of $\alpha = \frac{1}{2}$, more recent advancements in the field have introduced algorithms aiming for a tighter approximation bound. The literature has shown that for monotone submodular function maximization subject to a matroid constraint, it is computationally hard to approximate this problem within a factor better than $1-1/e \approx 0.63\%$ [32]. A suboptimal solution, aiming at such bound is proposed in [20] and explained in further detail in [33]. The approach introduced by [20] utilizes the multilinear extension of submodular set functions, as defined in (11), along with a continuous relaxation of the partition matroid, to solve for a fractional solution of the continuous relaxation of (4). A suboptimal solution for (4) is then recovered through a lossless rounding procedure. In the following, we provide a brief overview of this sophisticated approach.

To simplify our notation, in the remainder of this article, we assume without loss of generality that the ground set is given by $\mathcal{P} = \bigcup_{j=1}^{N} \mathcal{P}_j = \{1, \cdots, n\}$ and for any $i \in \mathcal{A}$ we have $\mathcal{P}_i = \{l, l+1, \cdots, m\} \subset \{1, \cdots, n\}$ and $\mathcal{P}_{i+1} = \{m+1, m+2, \cdots, q\} \subset \{1, \cdots, n\}$.

The *matroid polytope*, which is the continuous extension of the partition matroid described in Definition 7, is

$$P(\mathcal{M}) = \left\{ \boldsymbol{x} \in [0,1]^n \;\middle|\; \sum\nolimits_{p \in \mathcal{P}_i} [\boldsymbol{x}_i]_p \leq \kappa_i, \forall i \in \mathcal{A} \right\}. \tag{25}$$

Let $\boldsymbol{x} = [\boldsymbol{x}_1^\top, ..., \boldsymbol{x}_N^\top]^\top$, where $\boldsymbol{x}_i \in [0,1]^{|\mathcal{P}_i|}$ is the membership probability vector of $\mathcal{P}_i$ that defines the probability of choosing policies from $\mathcal{P}_i$ for each agent $i \in \mathcal{A}$. Then, the matroid polytope (25) is the convex hull of the partition matroid in the space of the membership probability vector.

With the multilinear extension and matroid polytope defined, the continuous relaxation of problem (4) is expressed as

$$\max F(\boldsymbol{x}), \quad \text{s.t.,} \;\; \boldsymbol{x} \in P(\mathcal{M}). \tag{26}$$

[20] has shown that $OPT = \max_{\boldsymbol{x} \in P(\mathcal{M})} F(\boldsymbol{x})$, OPT is the optimal value of the submodular maximization problem (4). Therefore, by finding the maximizer of (26) and employing a lossless rounding, we can determine the submodular maximization problem (4). But, optimization problem (26) is non-convex/non-concave maximizer, which implies that numerical solutions to this problem do not always guarantee convergence to the global maximizer.

The solution proposed by [20] for (26), known as the *continuous greedy* algorithm, involves moving in the direction of maximum ascent within $P(\mathcal{M})$ by following the flow:

$$\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} = \boldsymbol{v}(\boldsymbol{x}) \;\; \text{where} \;\; \boldsymbol{v}(\boldsymbol{x}) = \arg\max_{\boldsymbol{w} \in P(\mathcal{M})} (\boldsymbol{w} \cdot \nabla F(\boldsymbol{x})), \tag{27}$$

over the time interval $[0, 1]$. This approach can be seen as a variation of the so-called Frank-Wolfe algorithm, also referred to as the conditional gradient, introduced by [34]. The Frank-Wolfe algorithm is a straightforward first-order iterative constrained optimization algorithm designed for optimizing smooth functions over closed, bounded convex sets.

[20] showed that $\boldsymbol{x}(t)$, generated by (27), stays within $P(\mathcal{M})$ for $t \in [0, 1]$, as it forms a convex combination of vectors in $P(\mathcal{M})$. Additionally, $\boldsymbol{x}(1)$ reaches the boundary of $P(\mathcal{M})$. [20] further showed that by following (27), we achieve $F(\boldsymbol{x}(1)) \geq (1 - 1/\mathrm{e})\, F(\boldsymbol{x}^\star)$, and thus $F(\boldsymbol{x}(1)) \geq (1 - 1/\mathrm{e})\, OPT$, where $OPT = f(\mathcal{S}^\star)$. Next, by employing the Pipage rounding method proposed by [35] or its stochastic variant as described in, the fractional solution $\boldsymbol{x}(1)$ is transformed into an integral point $\bar{\boldsymbol{x}}$ within $(\mathcal{P}, \mathcal{M})$. This integral solution $\bar{\boldsymbol{x}}$, represented as $[\bar{\boldsymbol{x}}_1^\top, ..., \bar{\boldsymbol{x}}_N^\top]^\top$ and confined to the set $\{0, 1\}^n$, meets the condition $F(\bar{\boldsymbol{x}}) \geq F(\boldsymbol{x}(1))$. As a result, $\bar{\mathcal{S}} = \mathcal{S}_{\bar{\boldsymbol{x}}}$ within $(\mathcal{P}, \mathcal{M})$ constitutes a deterministic feasible set that not only satisfies $f(\bar{\mathcal{S}}) \geq F(\boldsymbol{x}(1))$ but also ensures $f(\bar{\mathcal{S}}) \geq (1 - 1/\mathrm{e})OPT$. While the

specifics of the Pipage rounding method are omitted here for brevity, it is crucial to acknowledge that the ability to reach the boundary of $P(\mathcal{M})$—that is,

$$\boldsymbol{x}(1) \in \left\{ \boldsymbol{x} \in [0,1]^n \;\middle|\; \sum\nolimits_{p \in \mathcal{P}_i} [\boldsymbol{x}_i]_p = \kappa_i, \forall i \in \mathcal{A} \right\},$$

plays a pivotal role in achieving lossless rounding through the Pipage rounding method.

Although elegant, a number of issues need to be addressed regarding the continuous greedy algorithm (27) to make it a practical and computationally reasonable solution. One challenge is computing the conditional gradient $\boldsymbol{v}(\boldsymbol{x})$ in (27), which, at first glance, seems to require solving a linear program. However, given that the gradient $\nabla F(\boldsymbol{x})$ consists of nonnegative elements, one solution for $\boldsymbol{v}(\boldsymbol{x}) = [\boldsymbol{v}_1^\top, ..., \boldsymbol{v}_N^\top]^\top$ is $\boldsymbol{v}_i \in \{0,1\}^{|\mathcal{P}_i|}$ with $\boldsymbol{v}_i = \mathbf{1}_{\boldsymbol{C}_i}$, where $\boldsymbol{C}_i$ is the set of the indexes of κ_i largest elements of $\nabla F(\boldsymbol{x})_i$. Here, we partitioned $\nabla F(\boldsymbol{x})$ as $[\nabla F(\boldsymbol{x})_1^\top, ..., \nabla F(\boldsymbol{x})_N^\top]^\top$.

The next challenge addressed is adopting a more practical solution through the use of a numerical iterative process:

$$\boldsymbol{x}(t+1) = \boldsymbol{x}(t) + \frac{1}{T}\boldsymbol{v}(t), \quad \text{where} \quad \boldsymbol{v}(t) = \underset{\boldsymbol{w} \in P(\mathcal{M})}{\arg\max}(\boldsymbol{w} \cdot \nabla F(\boldsymbol{x})), \tag{28}$$

where $1/T$, with $T \in \mathbb{Z}_{>0}$, is the step size used to discretize $[0,1]$ instead of following the continuous path in (27). However, a more significant issue is constructing the gradient $\nabla F(\boldsymbol{x})$ without the need to compute $f(\mathcal{R})$ for all $\mathcal{R} \in 2^{\mathcal{P}}$. Given that multilinear extension requires this information, computing $\nabla F(\boldsymbol{x})$ becomes computationally intractable as the size of the ground set $\mathcal{P}$ increases. A practical solution is provided by considering the stochastic interpretation (13) of the gradient. Drawing enough set samples according to the membership probability vector $\boldsymbol{x}$ can yield an estimate of $\nabla F(\boldsymbol{x})$ at a reasonable computational cost. The Chernoff-Hoeffding inequality [36] can be utilized to determine the quality of this estimation given the number of samples. [20] shows that the optimality gap for this practical approach is $1 - 1/\mathrm{e} - O(1/T)$ with the probability of $1 - 2Tn\mathrm{e}^{-\frac{1}{8T^2}K}$, where K is the number of samples. Further studies on the use of the multilinear extension for submodular maximization are discussed in [37, 33, 38, 39, 40, 41].

5 Distributed Submodular maximization

In many submodular maximization problem settings, the volume of collected data is not only vast but also expanding rapidly. The widespread deployment of sensors, for example, has resulted in the collection of large quantities of physical world measurements. Similarly, data on mechanical and human activities are being captured and stored at ever-increasing rates and with greater detail. These datasets are often complex and high-dimensional, necessitating distributed storage and processing solutions. Efforts to adapt the sequential greedy algorithm for tackling large-scale submodular maximization problems have included attempts to reduce the problem size through approximations [42]. Given the extensive algorithmic challenges, parallel computing naturally lends itself as a solution, with certain distributed algorithmic solutions for submodular optimization already explored, primarily focusing on submodular maximization subjected to uniform matroid constraints; see e.g., [43, 44, 45, 46, 47, 48]. Many of these efforts utilize the MapReduce method [49], which is arguably one of the most successful models for reliable and efficient parallel computing. Within this method, the ground set is distributed among m independent machines (the map phase), each with limited memory and processing power. These machines then perform computations in parallel on their allotted data subsets. The outcomes of these computations are then collated onto a single machine and further processed in the final round to produce the result. It is during the shuffle phase that these machines can communicate and exchange data.

While parallel computing plays a crucial role in addressing the challenges of large-scale data, the shift towards distributed implementations of submodular maximization extends beyond the pursuit of

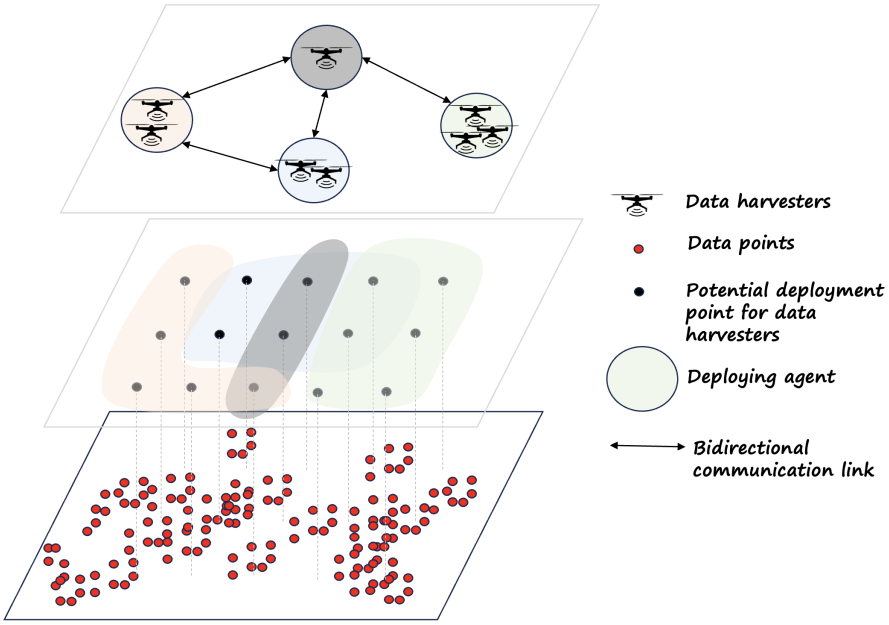

Figure 3: A simple schematic illustration of a distributed sensor deployment for data harvesting involving a group of agents $\mathcal{A}$. Each agent $i \in \mathcal{A}$ possesses κ_i data harvesting drones, which can be deployed at a set of pre-assigned deployment points $\mathcal{B}_i$, indicated by the same color used to depict the agent. The agents communicate over a connected graph to determine the optimal deployment positions for the team, as defined by the submodular maximization problem subject to a partition matroid described in Section 3.

computational efficiency. In some submodular maximization problems especially in cyber-physical system (CPS) and networked system applications, the elements of the optimization problem—such as the elements of the utility function or the discrete decision set—are often distributed across multiple entities (agents), which, despite seeking the optimal solution, might not want to share their information with a central authority. For such applications, the solution is desired in a distributed manner, where agents arrive at the solution using local interactions. Many challenges must be faced when designing distributed submodular maximization solvers. The algorithmic challenges stem from processing the data that is distributed across several machines/agents while using minimal communication and synchronization across the agents/machines. At the same time, it is important to deliver solutions that are competitive with the centralized solution when all fragmented information is available to a single authority.

Due to space limitations, we will elaborate only on the solution approaches for a class of distributed submodular maximization classified as *distributed ground set*, which is an instance of submodular maximization subject to partition matroid. In this class of problems, the disjoint sets $\mathcal{P}_1, \cdots, \mathcal{P}_N$ each respectively belong to an agent $i \in \mathcal{A} = \{1, \cdots, N\}$ and are *not known* to other agents. The number of strategy choices of each agent i, κ_i, is also known *only* to the respective agent. The agents' access to the utility function is through a black box that returns $f(\mathcal{R})$ for any given set $\mathcal{R} \in \mathcal{P}$ (value oracle model). As an example, consider the data harvesting problem discussed in Section 3, where we have a group of agents $\mathcal{A}$, which communicate over a connected undirected graph and want to decide via local interactions where to deploy their data harvester devices to maximize the utility function (16); see Fig. 3.

For distributed ground set problems, the sequential greedy algorithm, described in (23), readily adapts to decentralization, either through sequential message-passing or by sharing messages sequentially via the cloud, see [17]. However, decentralizing the sequential greedy algorithm introduces communication routing overhead. When agents communicate over a connected graph, implement-

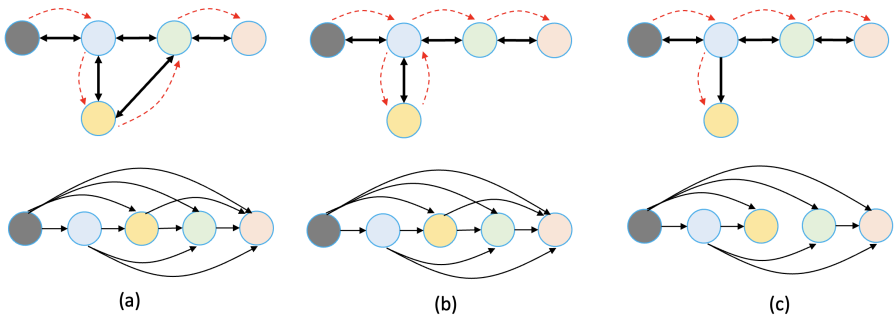

Figure 4: The top figures illustrate the communication topology (solid black arrows) and the message-passing sequence (dashed red arrows). The bottom figures detail the information-sharing graph. In case (a), the communication graph possesses a Hamiltonian path, enabling the shortest sequence for optimal message passing in the sequential greedy algorithm. In case (b), the absence of a Hamiltonian path in the communication graph requires the blue agent to participate twice in the message-passing process to implement the sequential greedy algorithm. In case (c), with the communication graph disconnected, the information-sharing graph becomes incomplete, preventing the precise execution of the sequential greedy algorithm. Consequently, the guaranteed optimality gap decreases from $1/2$.

ing sequential message-passing necessitates identifying a Hamiltonian path—a path that visits each agent on the graph exactly once—which is an NP-hard problem. If a Hamiltonian path does not exist within a graph, an alternative path that visits agents the fewest number of times must be identified to ensure communication-efficient sequential message-passing. This concept is visually represented in Figure 4(a) and (b), demonstrating scenarios with and without a Hamiltonian path, respectively. Moreover, it has been demonstrated that the sequence order affects the actual approximation factor of the solution derived from the sequential greedy algorithm [50]; see also [51] for further numerical examples. The difficulty of pinpointing the sequence that yields the best solution grows exponentially with the size and connectivity of the communication network.

In the implementation of the sequential greedy algorithm through message-passing, an additional key factor to consider is the impact of message drop-offs, or incomplete message-passing sequences, on the optimality gap. Such drop-offs lead to an incomplete information sharing graph, $\mathcal{G}_I$. The lower plots in Fig. 4 illustrate this graph, visualizing the flow of information as a result of the message-passing process. Here, an arrow from agent i to agent j signifies successful information relay from i to j, whether directly or indirectly through preceding agents. Research by [52] reveals that the optimality gap is intricately linked to the clique number, $\mathcal{W}(\mathcal{G}_I)$, of the information graph. Specifically, the clique number represents the size of the largest fully connected component in the information graph's undirected version. The findings demonstrate that the optimality gap expands as communication paths become disjointed, a phenomenon quantified as $f(\bar{\mathcal{S}}_{SG}) \geq \frac{1}{2+n-\mathcal{W}(\mathcal{G}_I)} f(\mathcal{S}^\star)$. Notably, this analysis guarantees reaching the well-known optimality gap of $1/2$ for the problem of submodular maximization under partition matroid constraints when the information graph is complete and $\mathcal{W}(\mathcal{G}_I) = n$, highlighting the critical role of information graph connectivity in algorithm performance.

The distributed implementation of the continuous greedy algorithm (28) over connected graphs has been explored in the literature. However, for the class of problems where the ground set is distributed, results are relatively scarce. A notable contribution in this area was proposed by [53] and [51]. Specifically, for the special class of submodular set functions with curvature $c = 1$, and when each agent is restricted to choosing only a single strategy from its own strategy set (*i.e.*, $\kappa_i = 1$), [53] introduced an average consensus-based distributed algorithm to address the maximization problem over connected graphs. The approach requires a closed-form expression of the multilinear extension function. However, constructing the closed form of the multilinear extension of a submodular function and its derivatives exponentially increases in computational complexity relative to the size of the

strategy set. Moreover, this result also depends on a centralized rounding scheme. An alternative solution employing a max-consensus method for local interaction among agents was proposed in [51]. This method takes advantage of a sampling-based approach to approximate $\nabla F(\boldsymbol{x})$ and employs a distributed stochastic rounding procedure, allowing agents to locally derive their policy selection, thus achieving a practical and fully distributed solution. We omit the technical details of these distributed solutions for brevity. Nevertheless, it is essential to highlight a significant hidden challenge in the distributed continuous greedy algorithms that utilize the multilinear extension of the utility function. In the distributed implementations of the continuous greedy algorithm (28) as seen in [53] and [51], each agent maintains a local copy of the membership probability vector $\boldsymbol{x}$ and employs local interactions through consensus protocols to align these local copies towards a common choice that approximates the central solution. However, an inherent challenge involves the local computation of $\nabla F(.)$, which necessitates access to all elements of any other agent's local ground set for which the corresponding element of the local copy of $\boldsymbol{x}$ is non-zero. This necessity implies that the distributed implementation of the continuous greedy algorithm involves not only the local exchange of $\boldsymbol{x}$ among neighboring agents but also the sharing of elements of the local ground sets of the agents across the network. [51] proposes a mechanism for managing this information sharing. For further details, interested readers are encouraged to refer to [51].

6 Concluding Remarks

In conclusion, this article has systematically explored submodular maximization problems subject to uniform and partition matroids, uncovering their significant applications across various disciplines. By diving into algorithmic solutions like the sequential greedy algorithm and its adaptations for distributed contexts, we highlight the vast potential for key advancements in optimization theory and practical problem-solving. Further attention to the continuous greedy algorithm opens a gateway to addressing the computational intricacies inherent in these problems, offering a refined tool for approaching submodular maximization within a continuous setting. This combined exploration of discrete and continuous methods enriches the field's theoretical base while expanding its applicability to real-world challenges, affirming the pivotal role of matroid constraints in shaping solution strategies for submodular maximization.

Submodular maximization theory, particularly when subjected to matroid constraints, is an expanding field that continues to attract considerable interest. We close this article with a few recommendations for further reading for the interested reader.

6.1 Further reading

Submodularity, a defining characteristic of set functions, provides a theoretical framework that aids in developing systematic approaches to tackle complex optimization challenges. However, not all utility functions inherently display submodular characteristics. This observation has prompted researchers to extend their inquiries beyond classical submodularity, exploring variations such as weakly or proportional submodularity as introduced in [54], along with lattice submodularity detailed in [55]. These lines of investigation have significantly broadened the theoretical landscape, enabling the application of optimization algorithms to a more diverse array of problems where traditional submodularity is not directly applicable. For instance, applications leveraging these variants are discussed in [56] and [57].

Similarly, the concept of k-submodularity extends the principle of submodularity to scenarios where decision-making involves multiple states beyond simple binary choices. Foundational contributions to this topic, such as those by [58] and [59], illustrate how k-submodularity broadens the traditional approach by accounting for functions in which elements can occupy one of $k + 1$ states, not limited to mere presence or absence. This broader perspective supports more nuanced problem modeling, especially in situations where elements engage at varying levels or capacities.

Furthermore, while sequential greedy and sampling-based approximate continuous greedy algorithms offer polynomial-time solutions, their computational demands can escalate with larger ground sets. Here, the "lazy" evaluation strategy, which aims to minimize computational load by circumventing unnecessary function evaluations, plays a critical role in enhancing efficiency. As initially proposed by [60] and further refined in the "Lazier than Lazy Greedy" by [61], these approaches significantly streamline the process of maximizing submodular functions. They cleverly leverage the diminishing returns property, updating marginal gains only for top-priority elements, thereby curbing the requisite number of function evaluations.

Besides computational efficiency, modern applications of submodular maximization, particularly in distributed environments, require privacy preservation. For instance, in distributed operations, agents need assurance that their policy choices will remain confidential. While distributed approaches reduce central data aggregation, communications between agents could still expose distributed network operations to adversarial eavesdroppers. Research into privacy for submodular maximization employs methods ranging from differential privacy as in [62] to novel strategies like the one by [63], which leverages the stochastic rounding in continuous greedy algorithms for privacy guarantees.

The field of submodular maximization is rapidly advancing, continually extending the boundaries of optimization theory. Key emerging topics that exemplify the forefront of research in this domain include Deep Submodular functions [64], Online submodular maximization [65], Mixed-integer programming approaches to generalized submodular optimization [66], Adaptive Submodular maximization [67], Streaming submodular maximization [68], Submodular reinforcement learning [69], and Fairness in submodular maximization [70].

References

[1] P.-J. Honysz, A. Schulze-Struchtrup, S. Buschjäger, and K. Morik, "Providing meaningful data summarizations using exemplar-based clustering in industry 4.0." https://arxiv.org/abs/2105.12026, 2021.

[2] A. Rostamizadeh, H. Esfandiari, L. Chen, MohammadHossein, Bateni, T. Fu, and V. Mirrokni, "Categorical feature compression via submodular optimization," pp. 515–523, 2019.

[3] K. El-Arini and C. Guestrin, "Beyond keyword search: discovering relevant scientific literature," in *17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 439–447, 2011.

[4] A. Borodin, H. C. Lee, and Y. Ye, "Max-sum diversification, monotone submodular functions and dynamic updates," in *ACM SIGMOD-SIGACT-SIGAI symposium on Principles of Database Systems*, pp. 155–166, 2012.

[5] K. Wei, R. Iyer, and J. Bilmes, "Max-sum diversification, monotone submodular functions and dynamic updates," vol. 37, pp. 1954–1963, 2015.

[6] Q. Ni, J. Guo, C. Huang, and W. Wu, "Community-based rumor blocking maximization in social networks: Algorithms and analysis," *Theoretical Computer Science*, vol. 840, pp. 257–269, 2020.

[7] A. Krause, A. Singh, and C. Guestrin, "Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies," *Journal of Machine Learning Research*, vol. 9, no. Feb, pp. 235–284, 2008.

[8] T. Summers, F. Cortesi, and J. Lygeros, "On submodularity and controllability in complex dynamical networks.," vol. 3, no. 1, pp. 91–101, 2016.

[9] Z. Liu, A. Clark, P. Lee, L. Bushnell, D. Kirschen, and R. Poovendran, "Submodular optimization for voltage control," *IEEE Tran. on Power Systems*, vol. 33, no. 1, pp. 502–513, 2018.

[10] A. Clark, P. Lee, B. Alomair, L. Bushnell, and R. Poovendran, "Combinatorial algorithms for control of biological regulatory networks," vol. 5, no. 2, pp. 748–759, 2018.

[11] A. Krause and C. Guestrin, "Near-optimal observation selection using submodular functions," in *American Association for Artificial Intelligence*, vol. 7, pp. 1650–1654, 2007.

[12] A. Clark, L. Bushnell, and R. Poovendran, "A supermodular optimization framework for leader selection under link noise in linear multi-agent systems," vol. 59, no. 2, pp. 283–296, 2014.

[13] J. Qin, I. Yang, and R. Rajagopal, "Submodularity of storage placement optimization in power networks," vol. 64, no. 8, pp. 3268–3283, 2019.

[14] M. Bucciarelli, S. Paoletti, E. Dall'Anese, and A. Vicino, "On the greedy placement of energy storage systems in distribution grids," 2020.

[15] S. T. Jawaid and S. Smith, "Submodularity and greedy algorithms in sensor scheduling for linear dynamical systems," *Automatica*, vol. 61, pp. 282–288, 2015.

[16] Z. Liu, A. Clark, P. Lee, L. Bushnell, D. Kirschen, and R. Poovendran, "Towards scalable voltage control in smart grid: A submodular optimization approach," in *ACM/IEEE 7th International Conference on Cyber-Physical Systems*, 2016.

[17] N. Rezazadeh and S. S. Kia, "A sub-modular receding horizon solution for mobile multi-agent persistent monitoring," *Automatica*, vol. 127, p. 109460, 2021.

[18] C. H. Papadimitriou and K. Steiglitz, *Combinatorial optimization: algorithms and complexity*. Courier Corporation, 1998.

[19] S. Fujishige, *Submodular functions and optimization*. Elsevier, 2005.

[20] J. Vondrák, "Optimal approximation for the submodular welfare problem in the value oracle model," in *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pp. 67–74, 2008.

[21] A. Bian, K. Levy, A. Krause, and J. M. Buhmann, "Continuous dr-submodular maximization: Structure and algorithms," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[22] M. Pedramfar, C. Quinn, and V. Aggarwal, "A unified approach for maximizing continuous dr-submodular functions," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[23] A. Krause and D. Golovin, "Submodular function maximization," in *Tractability: Practical Approaches to Hard Problems* (L. Bordeaux, Y. Hamadi, and P. Kohli, eds.), pp. 71–104, Cambridge, UK: Cambridge University Press, 2014.

[24] L. Kaufman and P. J. Rousseeuw, *Finding groups in data: an introduction to cluster analysis*. John Wiley & Sons, 2009.

[25] R. Gomes and A. Krause, "Budgeted nonparametric learning from data streams," 2010.

[26] D. J. L. B. Lehmann and N. Nisan, "Combinatorial auctions with decreasing marginal utilities," 2006.

[27] N. Mehr and R. Horowitz, "A submodular approach for optimal sensor placement in traffic networks," in *2018 Annual American Control Conference (ACC)*, pp. 6353–6358, IEEE, 2018.

[28] L. Fisher, G. Nemhauser, and L. Wolsey, "An analysis of approximations for maximizing submodular set functions—ii," in *Polyhedral Combinatorics*, pp. 73–87, Springer, 1978.

[29] G. Nemhauser, L. Wolsey, and M. Fisher, "An analysis of approximations for maximizing submodular set functions—i," *Mathematical Programming*, vol. 14, no. 1, pp. 265–294, 1978.

[30] G. L. Nemhauser and L. A. Wolsey, "Best algorithms for approximating the maximum of a submodular set function," *Math. Operations Research*, vol. 3, no. 3, pp. 177–188, 1978.

[31] M. Conforti and G. Cornuéjols, "Submodular set functions, matroids and the greedy algorithm: tight worst-case bounds and some generalizations of the Rado-Edmonds theorem," *Discrete applied mathematics*, vol. 7, no. 3, pp. 251–274, 1984.

[32] U. Feige, "A threshold of ln n for approximating set cover," *Journal of the ACM*, vol. 45, no. 4, pp. 634–652, 1998.

[33] G. Calinescu, C. Chekuri, M. Pal, and J. Vondrak, "Maximizing a monotone submodular function subject to a matroid constraint," vol. 40, no. 6, pp. 1740–1766, 2011.

[34] M. Frank, P. Wolfe, *et al.*, "An algorithm for quadratic programming," *Naval research logistics quarterly*, vol. 3, no. 1-2, pp. 95–110, 1956.

[35] A. Ageev and M. Sviridenko, "Pipage rounding: A new method of constructing algorithms with proven performance guarantee," *Journal of Combinatorial Optimization*, vol. 8, no. 3, pp. 307–328, 2004.

[36] H. W, "Probability inequalities for sums of bounded random variables," in *The Collected Works of Wassily Hoeffding*, pp. 409–426, Springer, 1994.

[37] J. Vondrák, "Submodularity and curvature: The optimal algorithm (combinatorial optimization and discrete algorithms)," 2010.

[38] C. Chekuri, J. Vondrak, and R. Zenklusen, "Submodular function maximization via the multilinear relaxation and contention resolution schemes," vol. 43, no. 6, pp. 1831–1879, 2014.

[39] A. A. Bian, B. Mirzasoleiman, J. Buhmann, and A. Krause, "Guaranteed non-convex optimization: Submodular maximization over continuous domains," in *Artificial Intelligence and Statistics*, pp. 111–120, 2017.

[40] A. Mokhtari, H. Hassani, and A. Karbasi, "Stochastic conditional gradient methods: From convex minimization to submodular maximization," *Journal of Machine Learning Research*, vol. 21, no. 105, pp. 1–49, 2020.

[41] O. Sadeghi and M. Fazel, "Online continuous dr-submodular maximization with long-term budget constraints," in *International Conference on Artificial Intelligence and Statistics*, pp. 4410–4419, 2020.

[42] K. Wei, R. Iyer, and J. Bilmes, "Fast multi-stage submodular maximization," in *International conference on machine learning*, pp. 1494–1502, PMLR, 2014.

[43] B. Mirzasoleiman, A. Karbasi, R. Sarkar, and A. Krause, "Distributed submodular maximization: Identifying representative elements in massive data," in *Advances in Neural Information Processing Systems*, pp. 2049–2057, 2013.

[44] R. Barbosa, A. Ene, H. L. Nguyen, and J. Ward, "The power of randomization: distributed submodular maximization on massive datasets," in *Proceedings of the 32nd International Conference on International Conference on Machine Learning*, vol. 37, pp. 1236–1244, 2015.

[45] B. Mirzasoleiman, A. Karbasi, R. Sarkar, and A. Krause, "Distributed submodular maximization," vol. 17, pp. 1–44, 2016.

[46] B. Mirzasoleiman, M. Zadimoghaddam, and A. Karbasi, "Fast distributed submodular cover: Public-private data summarization," in *Advances in Neural Information Processing Systems*, pp. 3594–3602, 2016.

[47] R. Kumar, B. Moseley, S. Vassilvitskii, and A. Vattani, "Fast greedy algorithms in mapreduce and streaming," *ACM Transactions on Parallel Computing*, vol. 2, no. 3, pp. 1–22, 2015.

[48] P. S. Raut, O. Sadeghi, and M. Fazel, "Online dr-submodular maximization with stochastic cumulative constraints," *arXiv preprint arXiv:2005.14708*, 2020.

[49] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," vol. 51, no. 1, pp. 107–113, 2008.

[50] R. Konda, D. Grimsman, and J. R. Marden, "Execution order matters in greedy algorithms with limited information," in *2022 American Control Conference (ACC)*, pp. 1305–1310, 2022.

[51] N. Rezazadeh and S. S. Kia, "Distributed strategy selection: A submodular set function maximization approach," *Automatica*, vol. 153, p. 111000, 2023.

[52] B. Gharesifard and S. Smith, "Distributed submodular maximization with limited information," vol. 5, no. 4, pp. 1635–1645, 2018.

[53] A. Robey, A. Adibi, B. Schlotfeldt, J. G. Pappas, and H. Hassani, "Optimal algorithms for submodular maximization with distributed constraints," *arXiv preprint arXiv:1909.13676*, 2019.

[54] A. Das and D. Kempe, "Submodular meets spectral: Greedy algorithms for subset selection, sparse approximation and dictionary selection," *arXiv preprint arXiv:1102.3975*, 2011.

[55] T. Soma and Y. Yoshida, "Maximizing monotone submodular functions over the integer lattice," *Mathematical Programming*, vol. 172, pp. 539–563, 2018.

[56] A. Hashemi, M. Ghasemi, H. Vikalo, and U. Topcu, "Randomized greedy sensor selection: Leveraging weak submodularity," *IEEE Transactions on Automatic Control*, vol. 66, no. 1, pp. 199–212, 2020.

[57] Z. Liu, A. Clark, L. Bushnell, D. S. Kirschen, and R. Poovendran, "Controlled islanding via weak submodularity," *IEEE transactions on power systems*, vol. 34, no. 3, pp. 1858–1868, 2018.

[58] A. Huber and V. Kolmogorov, "Towards minimizing k-submodular functions," in *Combinatorial Optimization: Second International Symposium, ISCO 2012, Athens, Greece, April 19-21, 2012, Revised Selected Papers 2*, pp. 451–462, Springer, 2012.

[59] J. Ward and S. Živnỳ, "Maximizing k-submodular functions and beyond," *ACM Transactions on Algorithms (TALG)*, vol. 12, no. 4, pp. 1–26, 2016.

[60] M. Minoux, "Accelerated greedy algorithms for maximizing submodular set functions," in *Optimization Techniques: Proceedings of the 8th IFIP Conference on Optimization Techniques Würzburg, September 5–9, 1977*, pp. 234–243, Springer, 2005.

[61] B. Mirzasoleiman, A. Badanidiyuru, A. Karbasi, J. Vondrák, and A. Krause, "Lazier than lazy greedy," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 29, 2015.

[62] M. R. Fouad, K. Elbassioni, and E. Bertino, "A supermodularity-based differential privacy preserving algorithm for data anonymization," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 7, pp. 1591–1601, 2014.

[63] N. Rezazadeh and S. S. Kia, "Distributed submodular maximization: trading performance for privacy," in *2022 IEEE 61st Conference on Decision and Control (CDC)*, pp. 5953–5958, 2022.

[64] B. W. Dolhansky and J. A. Bilmes, "Deep submodular functions: Definitions and learning," *Advances in Neural Information Processing Systems*, vol. 29, 2016.

[65] Z. Xu, H. Zhou, and V. Tzoumas, "Online submodular coordination with bounded tracking regret: Theory, algorithm, and applications to multi-robot coordination," *IEEE Robotics and Automation Letters*, vol. 8, no. 4, pp. 2261–2268, 2023.

[66] S. Küçükyavuz and Q. Yu, "Mixed-integer programming approaches to generalized submodular optimization and its applications," in *Tutorials in Operations Research: Advancing the Frontiers of OR/MS: From Methodologies to Applications*, pp. 1–30, 2023.

[67] H. Esfandiari, A. Karbasi, and V. Mirrokni, "Adaptivity in adaptive submodularity," in *Conference on Learning Theory*, pp. 1823–1846, PMLR, 2021.

[68] A. Badanidiyuru, B. Mirzasoleiman, A. Karbasi, and A. Krause, "Streaming submodular maximization: Massive data summarization on the fly," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 671–680, 2014.

[69] M. Prajapat, M. Mutnỳ, M. N. Zeilinger, and A. Krause, "Submodular reinforcement learning," *arXiv preprint arXiv:2307.13372*, 2023.

[70] M. El Halabi, J. Tarnawski, A. Norouzi-Fard, and T.-D. Vuong, "Fairness in submodular maximization over a matroid constraint," in *International Conference on Artificial Intelligence and Statistics*, pp. 1027–1035, 2024.