Submodular Maximization Subject to Information Constraints

by

Andrew Downie

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2022

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

In general, submodular maximization is relevant in many problems in controls, robotics and machine learning, because it models many computationally difficult problems. A simple greedy strategy can provide strong approximation guarantees for many of these problems. We wish to expand the set of scenarios where submodular maximization can be applied. More specifically, in this thesis we study submodular maximization problems where decision-makers are subject to information constraints.

The first type of information constraint we explore is when decision-makers can only partially observe the submodular objective function. This scenario can arise when an objective function is expensive to compute or physical constraints prevent the evaluation of the objective function. We formalize the problem and then show that in general, strong performance cannot be guaranteed. We then present two different greedy strategies that provide strong approximation guarantees when only having limited access to the objective under additional assumptions about the submodular objective.

The second information constraint we explore is in the context of distributed submodular maximization. In these scenarios, a team of agents wishes to maximize a submodular objective collaboratively but are constrained to make decisions based on a subset of the other agents' actions. We are interested in what types of functions are challenging for agents to optimize given their information structure. We explore how submodular functions that exhibit worst-case performance can be formulated through a linear program. This approach provides a means to numerically compute worst-case performance bounds as well as functions where a team of agents will exhibit their worst-case performance. Using this technique, we provide theoretical performance bounds based on their information structure that are tighter than the known bounds in the literature.

Acknowledgements

I would like to thank both my supervisors, Professors Stephen L. Smith and Bahman Gharesifard for their guidance during my time at the University of Waterloo. I would also like to thank my colleagues in the Smith Autonomy Lab for the great discussions and insights. I am grateful for my friends Matt Cann, Daniel Sola and Aidan Keavney, who I lived with during my time in Waterloo, for keeping me motivated throughout the pandemic. Finally, I would like to thank my family, for all the support they have provided me during my journey.

Dedication

This thesis is dedicated to my parents and grandmother.

Table of Contents

List of Figures

Chapter 1

Introduction

Submodular maximization has recently generated interest in many decision-making problems, as it can provide strong performance guarantees for computationally difficult problems. Submodular functions are set functions that exhibit the property of diminishing returns. Submodular optimization is a well-studied subject, as these function model many real-world problems in controls [32, 41], robotics [42, 26, 55], data processing and machine learning [45, 44, 40]. It is well known that maximizing a submodular function subject to a cardinality constraints is NP-hard, but if the function is normalized and monotone, then a greedy algorithm provides an approximation factor of $(1 - 1/e)$ in polynomial time [37]. A significant portion of the research in this domain involves expanding the set of applications where submodular maximization can applied to solve hard problems. This work, primarily focuses on submodular maximization problems where the decision-makers are subject to information constraints.

One practical difficulty in implementing algorithms for submodular maximization in complex settings is that the required function evaluations are computationally expensive. This can be attributed to the large-scale characteristics of the system [1], application-specific constraints such as communication constraints [5], or the type of data the objective function is evaluating [27]. In its most common form, a submodular function is treated as a *value oracle*, which is repeatedly queried by a greedy strategy to maximize the function. Therefore, it is inherently assumed that one can evaluate the function for sets of any size. In practice, however, it may only be possible to evaluate the function on smaller set sizes due to computational costs or limitations imposed. Consider the setting where a company is selecting locations for several new retail stores. The total revenue received by a set of store locations can be modelled as a submodular function: As more stores are added, the marginal benefit of adding a new store is reduced. In the classical greedy algorithm

for submodular maximization, we assume we have access to a value oracle to evaluate subsets of store locations. Armed with this oracle, we iteratively add a new store s_k to existing set $\{s_1, \ldots, s_{k-1}\}$ by selecting the location s_k that maximizes the marginal benefit $f(s_1, \ldots, s_{k-1}, s) - f(s_1, \ldots, s_{k-1})$. To evaluate the quantity, the oracle must accurately model the revenue of k stores, which can be challenging in practice due to their complex interactions: for example s_k may reduce the revenue at some store s_i, which then may affect some other store's revenue.

Motivated by the lack of access to the full value oracle in practical settings, we seek to determine how well we can approximate the maximum value of a submodular function when we have access to a limited set of function values. For the main chapter of this thesis, we focus on the case where we can access function values for single elements $f(s_i)$ and for pairs of elements $f(s_i, s_j)$. We refer to this as *pairwise information*. In the motivating example, this corresponds to knowing the total revenue for a single store and the total revenue for any two stories together and nothing more. Note that this restriction on information is severe. A submodular function on a base set of N elements can be represented as a look up table with 2^N values. If only pairwise information is available, this means we have access to only $N(N+1)/2$ values. While we focus on pairwise information, we also extend most results to the case of k-wise information, where we can evaluate the value of any set with size at most k.

Information constraints also arrive in the context of distributed systems problems. Suppose there are a team of agents that wish to maximize a collaborative objective. Often, there are communication constraints, that limit the information available to each of the agents. We focus on the scenario where the agents' objective function is submodular. This problem has been in explored in [12, 15, 16, 5]. The main focus of these works are to provide performance guarantees for agents executing an adapted greedy strategy. In the second main contribution of this thesis, we take an alternative approach to this problem.

A large portion of the theoretical research in the domain of submodular maximization is concerned with providing approximation guarantees. It is common that the greedy strategies for maximization perform better in practice than the theoretical guarantees. With this in mind, we are interested in answering the following question: Is it possible to find a submodular function f such that the solution produced by the greedy strategy exhibits its worst-case approximation ratio? Typically, it is challenging to find such examples, and we want to know if such examples can be programmatically found. In this work, we establish a connection between submodular functions and feasible regions of linear constraints. We use linear programming to find these worst-case function examples. A similar linear programming technique was used to provide approximation guarantees in the foundational paper by Nemhauser et. al [37]. The authors upper bound the optimal value of the problem

and then find the minimum value of the greedy solution by utilizing a linear program. In our work, we are interested in generalizing these methods as well as finding the particular functions that exhibit the worst-case values.

We directly apply this approach to provide performance guarantees to the distributed submodular maximization problem. To elaborate further on the distributed settings we have in mind, we consider the problem where each agent must a select an action from their own set of available actions, to maximize the objective. The agents make choices based on a subset of actions selected by other agents. Since the agents cannot observe all of the other agents actions they are forced to make decisions under partial information. The agents make their choices by greedily maximizing their own marginal objective value given the information available to them.

This problem is studied in [12, 15], and is analogous to maximizing a submodular function over the partition matroid, with an additional information constraint. When the agents share a common action set to select from, the problem becomes analogous to maximizing over the uniform matroid. For the uniform matroid, the centralized greedy strategy provides an approximation guarantee of $(1-1/e)$ of optimal compared to the partition matroid, where the same strategy provides an approximation guarantee of $1/2$ [25]. In [12, 15], the authors primarily focus on providing guarantees for the scenario where agents each have their own action set. We are interested in knowing, if the agents share an action set, can a tighter approximation bound be achieved. We are also interested in how the performance is characterized in terms of the agents' information structure.

1.1 Thesis Contributions

The following are the key contributions of this thesis:

- In Chapter 4, we introduce the problem of submodular maximization with limited function access. We provided inapproximability results for the problem. In light of this, we propose two main algorithms to provide approximation guarantees to the problem by leveraging additional assumptions about the underlying objective function. The work in Chapter 4 is under review for a journal publication and can be found in [8].
- In Chapter 5, we establish a connection between linear programming and submodular functions, and apply this connection to provide approximation guarantees for the distributed submodular maximization problem. The work in Chapter 5 is under review for a journal publication.

1.2 Thesis Outline

The remainder of the thesis is structured as follows:

Chapter 2 reviews the relevant research literature that relates to the contributions presented in this thesis.

Chapter 3 reviews topics and notation that are used throughout Chapters 4 and 5.

In Chapter 4, we introduce the problem of maximizing a submodular function with limited function access. During this chapter, we provide inapproximability results for the problem. We then propose two algorithms which have limited access to the objective, where under a set of assumptions both provide approximation guarantees for the problem. We also discuss a method to measure approximation performance using limited information about the objective. The pairwise algorithm can be computed efficiently which allow for design trade-offs between approximation performance and time efficiency. We end the chapter with an experiment using real-world data, which highlights the effectiveness of the algorithms.

In Chapter 5, we establish a connection between linear programming and submodular maximization. We then utilize a linear programming approach to provide performance guarantees for the distributed submodular maximization problem. We finally, provide empirical evidence that the new guarantees are stronger than guarantees found in the previous literature.

Chapter 6 summarizes the work presented in thesis, as well as discusses future work.

Chapter 2

Literature Review

2.1 Submodular Maximization

2.1.1 Approximate Value Oracles and Surrogate Function Maximization

In some applications it is challenging to compute the submodular objective function. In these scenarios, the submodular objective function can be substituted with a surrogate objective, which is maximized instead [53]. The goal would be to select a surrogate function that can be efficiently computed. This method can still provide strong approximation guarantees if the surrogate function sufficiently represents the original objective function. The surrogate function can be treated as an approximate value oracle. An approximate value oracle is a black-box that takes as input a set of elements and outputs an approximation of the original function evaluated on the input set. The ratio between the oracle's values for greedily selected elements and the original objective values for the same elements can be used to derive approximation bounds [14, 53]. The ratios used to derive the bounds are called approximation factors.

2.1.2 Time Complexity of Greedy Strategy

An important aspect of submodular maximization is the computational efficiency of the greedy strategy [35, 1, 57, 36]. Under the assumption that the strategy has access to a value oracle for the submodular function, that can be computed in *constant* time, the

time complexity of the greedy strategy is $\mathcal{O}(|X| \cdot n)$ [35]. Where X is the base set of elements optimized over and n the number of element selected. Even though the time complexity is polynomial, for X with large cardinality, the greedy strategy can become prohibitively expensive to execute. Consequently, alternative implementations are available for the greedy strategy, which improves computational efficiency to handle situations where X is large.

A common technique used to improve efficiency is utilizing lazy evaluation of the function combined with sampling of the base set. Together these techniques reduce the overall number of computations required to execute the greedy strategy [28, 35]. Parallel and steaming algorithms have been developed for applications where the X is too large to be stored in memory. The parallel algorithm utilizes a Map-Reduce style computation where X is split up across multiple machines, and each machine computes a partial solution to the problem. A central machine then combines the partial solutions to produce the final solution [36]. The streaming algorithms use elaborate thresholding techniques to select elements while passing over X once [1, 23]. Alternative algorithms realize the set X as a tree. Nodes of the tree are then pruned to reduce the number of elements that are optimized over [57]. These techniques come at the cost of reduced approximation performance, ultimately providing trade-offs between performance and efficiency. In practical applications, there exist a computational cost in computing the submodular function. Most of these techniques reduce the search space being optimized over but do not address the computational cost of computing the objective.

2.1.3 Effects of Curvature on Approximation Performance

An important submodular function property for optimization purposes is curvature. The curvature characterizes the maximum rate at which the marginal returns diminish as the size of the subset taken with respect to grows. Curvature essentially describes how "submodular" a function is. A foundational result from [4] shows that the greedy strategy provides stronger approximation guarantees than originally presented in [37] when assumptions about the curvature are made. It was also shown that any algorithm that can guarantee an approximation bound better than the one found in [4] would require an exponential number of queries to the submodular function [51]. This means that the best approximation bound that one can hope for without additional assumptions are the bounds provided in [51].

In general, the curvature is useful when analyzing the performance bounds of greedy algorithms in various submodular maximization problems. Curvature is used to provide

bounds for maximizing specific submodular objectives as well as in guarantees for more complex submodular maximization problems. For example, curvature for functions that arise in sensor selection for Kalman filtering [17] and entropy minimization [44] have been thoroughly studied. An example of a more complex submodular maximization problem is robust and sequential submodular maximization [39, 49]. In these problems, a submodular maximization problem is iteratively solved where after each iteration, an adversary removes a subset of the elements to minimize the value of the solution. The curvature of the objective has a strong effect on the approximation guarantees for an adapted greedy strategy in this scenario.

2.1.4 Information Constrained Greedy Strategies

Recently, information constraints are considered in the context of distributed submodular maximization. In these scenarios, a team of agents are attempting to collaboratively maximize a submodular objective function. Each agent has access to their own set of actions and can observe a limited number of decisions made by other agents [12, 5, 15, 16, 46]. In these settings, the agents sequentially make decisions maximizing their marginal contribution with respect to a subset of decisions made by previous agents. In [12, 15, 5], the information available to the agents is encoded in a directed acyclic graph which is called the communication graph. In [12, 15], the team's performance is bounded by properties of the communication graph such as the *clique* number and the *fractional independence* number. Alternatively in [5], the team's performance is characterized by the maximum redundancy between pairs of agents' that do not share an edge in the communication graph.

In [16], a similar scenario is explored where agents can also pass a subset of their observations to agents that appear later in the communication graph. The authors showed that message passing could be exploited to improve the team's overall performance. Authors have also explored how the performance is affected when subsets of agents have to make decisions in parallel [46]. In all of the above distributed scenarios, the approximation performance of the agents degrade as communication between the agents is reduced.

2.2 Applications of Submodular Maximization

2.2.1 Classic Submodular Maximization Problems

There is a large number of problems that can be modeled by submodular functions. Some classical submodular functions that have been extensively explored include, graph cut [29],

entropy [44], mutual information [25], set cover [9], and facility location [25]. Some common constraints that are imposed on the problems include, cardinality [37], knapsack [20], budget [43, 29], and matroid constraints [2, 18, 11]. In many of the following problems involve combining the above objectives and constraints to solve problems in real-world applications.

2.2.2 Sensor Placement and Coverage

Submodular objectives arise in many sensing and coverage problems. Given a set of sensors, with each sensor having its own sensing footprint, the goal is to place a set of sensors such that their combined sensing area is maximized. Area coverage exhibits the diminishing returns property because when a sensor is placed, its footprint could overlap with another sensor's footprint, reducing its marginal gain. Coverage objectives have been used in planning UAV trajectories for 3D scanning of large structures [42], wireless sensor networks [34, 47], video surveillance, and sensor scheduling [52].

Sensing objectives can also be modelled from an information-theoretic perspective [28]. Suppose we have a set of random variables we wish to observe with a finite number of sensors. For example, suppose we wish to measure temperatures at various locations in a building with a set of sensors. To choose a set of locations to place sensors, we can select a configuration that maximizes the *mutual information* between the random variables at the locations where the sensors are placed and random variables at locations where sensors are not placed [28, 24]. This particular mutual information objective exhibits submodularity, and therefore the greedy strategy can be used to optimize it.

2.2.3 Data Summarizing and Data Processing

Submodular maximization has become useful in data summarization and prepossessing. When working with large text documents or data sets, it is useful to filter the data into smaller subsets that are easier to work with [54]. Submodular functions are good models for summarization and diversity measures [40, 36, 31, 29, 30]. One application of the summarization objectives is to perform automatic summarization on large text documents and multi-document libraries [31, 29]. In these problems, the sentences in the documents are modeled as a graph where the nodes are sentences and the edges represent the similarity between pairs of sentences. A modified graph cut function is then greedily maximized to generate a summary of the documents. The modified graph cut function is submodular and maximizing it produces a subset of sentences that summarize the data set. The graph

cut objective measures the similarity between a selected set of sentences and the sentences that are not selected, which is effectively a measure of summarization.

Similarly, submodular maximization is useful for machine learning applications, especially when working with large data sets. In [36], the authors want to perform exemplar-based clustering where they wish to select a subset of the data that is representative of the entire data set. This objective can be formulated as a submodular function, and as previously mentioned, the authors proposed a parallel greedy strategy to accomplish their goal. The greedy strategy is often much more efficient than other methods which makes it applicable for problems with massive data sets. The greedy strategy has also been used in data set prepossessing to select the best data points to train machine learning models on [54]. Some other machine learning applications of submodular maximization is in recommender systems [45], image segmentation [40], and clustering [21].

2.2.4 Controls and Path planning

Submodularity has many applications in controls. As shown in [3, 17], the mean squared error of a Kalman filter is approximately submodular with respect to the set of sensors observing the system. The marginal gain of adding an individual sensor for state estimation diminishes as more sensors are added. Even though the mean squared error for Kalman filtering is only approximately submodular, the authors in [17] provide approximation guarantees for a greedy sensor selection strategy. In a related problem, the author of [50] use greedy optimization of a submodular objective to select actuators to ensure controlability of a system. These greedy techniques have also been extended for robust target tracking applications [49, 48, 56].

Submodularity has also played a role in informative path planning. As described in [22, 6], suppose an agent is attempting to plan a path to maximize the information retrieved from an environment. The authors present strategies where the agents utilize a submodular objective that describe the information captured by a path to guide path planning. Submodular functions model these systems well and using greedy optimization techniques, provide guarantees for the quality of the paths planned.

Chapter 3

Background

3.1 Preliminaries

3.1.1 Submodular Functions

Let X be a set of base elements, and let 2^X be the power set of the base set X which is the set containing all of the subsets of X.

Definition 3.1.1 (Submodularity)**.** *A set function $f : 2^X \to \mathbb{R}$ is submodular if for all $A \subseteq B \subseteq X$ and $x \in X \backslash B$, we have*

$$f(A \cup \{x\}) - f(A) \geq f(B \cup \{x\}) - f(B). \tag{3.1}$$

An alternative but, equivalent definition of a submodularity is provided in the following lemma.

Lemma 3.1.2. *A set function $f : 2^X \to \mathbb{R}$ is submodular if and only if $\forall A, B \subseteq X$, we have*

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B).$$

These definitions are used interchangeably in the literature.

For $x, y \in X$ and $A \subseteq X$, we refer to $f(A \cup \{x\}) - f(A)$ as the marginal return of x given A, denoted by $f(x|A)$. We denote the objective value of a singleton $f(\{x\})$ by $f(x)$. We also denote the pairwise marginal return of x given y as $f(x|y)$.

We will also consider submodular functions which possess the two following properties.

Definition 3.1.3 (Monotonicity)**.** *A set function $f : 2^X \to \mathbb{R}$ is monotonic if for all $A \subseteq B \subseteq X$, we have*

$$f(A) \leq f(B).$$

Definition 3.1.4 (Normalized)**.** *A set function $f : 2^X \to \mathbb{R}$ is normalized if we have,*

$$f(\emptyset) = 0.$$

We can write the values of a submodular function in terms of a telescoping sum of the marginal returns. Let $f : 2^X \to \mathbb{R}$ and $S = \{x_1, \ldots, x_n\} \subset X$ and $S_i = \{x_1, \ldots, x_i\}$, then

$$f(S) = \sum_{i=1}^{n} f(x_i|S_{i-1}).$$

This identity is a useful when working with submodular functions. We also have the following inequality for submodular functions,

$$\sum_{i=1}^{n} f(x_i|S_{i-1}) \leq \sum_{i=1}^{n} f(x_i). \tag{3.2}$$

A set function is consider to be modular if (3.2) holds with equality.

Another property of a submodular function that is relevant to this work is *curvature*.

Definition 3.1.5 (Curvature)**.** *Let f be a submodular function then, the curvature c of f is defined as*

$$c = 1 - \min_{S \subseteq X, x \in X \setminus S} \frac{f(x|S)}{f(x)}. \tag{3.3}$$

The curvature of a submodular function reflects how much the marginal values of $f(x|S)$ can decrease as a function of S.

3.1.2 Linear Programming

Linear programs are optimization problems where the objective function and constraints are linear. A general linear program in standard form can be expressed as

$$\begin{aligned} &\min_{x \in \mathbb{R}^n} c^\top x, \\ \text{s.t.} \quad &Ax \leq b, \\ &x \geq 0. \end{aligned} \tag{3.4}$$

where $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$. By [7], if the program has an optimal solution with finite objective value, then $x \in \mathbb{R}^n$ can be found using a polynomial algorithm. Given a linear program in standard form, a black-box solver can be used to efficiently find a solution.

3.1.3 Graph Theory

Let $G = (V, E)$ be a directed graph with vertices V and edges E. A directed acyclic graph (DAG), is a directed graph, where no cycles exist. Every DAG can be topologically sorted such that there exist a labeling of vertices such that $i < j$ for all $(i, j) \in E$. The following are three definitions that this work will utilize:

Definition 3.1.6 (Complete)**.** *A directed acyclic graph $G = (V, E)$ is complete if no edge can be added to E without introducing a cycle into the graph.*

Definition 3.1.7 (Clique)**.** *A clique of a graph $G = (V, E)$ is a sub-graph that is complete.*

Definition 3.1.8 (Clique Number)**.** *Given a graph $G = (V, E)$ the clique number of the graph $\omega(G)$ is the size of the largest clique.*

In general, finding the maximum clique is difficult. The decision version of the problem called the *clique problem* is NP-complete [7]. Note, the number of cliques in a graph can be exponential in the number of vertices.

3.2 Submodular Maximization Problem

3.2.1 Basic Problem

Suppose that you have a base set X and a monotone, normalized and submodular function $f : 2^X \to \mathbb{R}$. We wish to solve the following problem.

$$\begin{aligned} &\max_{S \subseteq X} f(S) \\ &\text{s.t. } |S| \leq n \end{aligned} \tag{3.5}$$

This is the classic problem of maximizing a submodular function subject to a cardinality constraint. We generalize the constraints using the following definition, which is a generalization of *independence*.

Definition 3.2.1 (Matroid)**.** *Let X be a finite set and $\mathcal{I}$ a non-empty collection of subsets of X called the independent sets. The system $M = (X, \mathcal{I})$ is called a matroid if:*

1. *$A \subseteq B \subseteq X$ and $B \in \mathcal{I} \implies A \in \mathcal{I}$*
2. *$\forall A, B \in \mathcal{I}$ and $|A| < |B| \implies \exists\, x \in B \backslash A$ such that $A \cup \{x\} \in \mathcal{I}$.*

Let $\mathcal{M} = (X, \mathcal{I})$ be a matroid, then a generalized version of Problem 3.5 can be stated as follows.

$$\begin{aligned} &\max_{S \subseteq \mathcal{I}} f(S) \\ &\text{s.t. } |S| \leq n \end{aligned} \tag{3.6}$$

Problem 3.5 maximizes f over the *uniform* matroid. Another common matroid constraint as seen in [12, 36, 5] is the maximizing over the *partition* matroid. Let $\{X_i\}_{i=1}^n$ be a sequence of disjoints sets and $X = \cup_{i=1}^n X_i$. Then we wish to solve the following program.

$$\begin{aligned} &\max_{S \subseteq X} f(S) \\ &\text{s.t. } |S \cap X_i| \leq k \\ &\forall i \in \{1, \ldots, n\} \end{aligned} \tag{3.7}$$

In general maximizing a submodular function subject to a cardinality constraint is a NP-hard problem [33]. On the other hand, minimizing a submodular function can be done in polynomial time [19].

3.2.2 The Greedy Algorithm

The primary algorithm that this work is based off is the greedy algorithm for submodular maximization. To produce an approximate solution $S = \{x_1, \ldots, x_n\} \subseteq X$ to Problem 3.5, we utilized the following algorithm.

Algorithm 1: Greedy Algorithm for Submodular Maximization

Input: Base set X, submodular function f and cardinality constraint n
Result: Approximate solution $S \subseteq X$

$S_i \leftarrow \emptyset$ for all $i \in \{0, \ldots, n\}$;
for $i \leftarrow 1, \ldots, n$ **do**
    $x_i \leftarrow \arg\max_{x \in X \setminus S_{i-1}} f(x|S_{i-1})$;
    $S_i \leftarrow S_{i-1} \cup \{x_i\}$;
end
$S \leftarrow S_n$

This algorithm is basis for much of the research in submodular maximization, including this work.

Approximation Performance

We recall, the result that we build off throughout this work [37].

Theorem 3.2.2. *(Nemhauser et. al 1978) Let $f : 2^X \to \mathbb{R}$ be a normalized monotone submodular function, $S \subseteq X$ be the solution produced by Algorithm 1 and $S^* \subseteq X$ be the solution to Problem 3.5, then*

$$f(S) \geq (1 - \tfrac{1}{e}) f(S^*).$$

In general, this is the best approximation we can achieve for maximizing an arbitrary submodular function subject to a cardinality constraint. In fact, the authors in [9] show that no algorithm that uses a polynomial number of calls to the value oracle f can achieve a better approximation factor than $(1-\frac{1}{e})$ unless $P = NP$. To achieve tighter approximation results we must make stronger assumptions about the function f. For example, if we assume the function f has a curvature c, then we can achieve an approximation ratio of $\frac{1}{c}(1 - e^{-c})$ using the greedy strategy [51]. In general, for Problem 3.6 the greedy strategy provides a $1/2$ approximation ratio and an approximation ratio of $1/(1+c)$ given the curvature of f.

Time Complexity

Assuming that we have a value oracle for the function f which takes $\mathcal{O}(1)$ time to compute, the classical greedy strategy takes $\mathcal{O}(|X| \cdot n)$ time to execute. This is because during each iteration of the algorithm, $f(x|S_{i-1})$ must be computed for each $x \in X \setminus S_{i-1}$ which takes

$\mathcal{O}(|X|)$ time. This set of operations is done n times to generate the solution, giving an overall time complexity of $\mathcal{O}(|X| \cdot n)$. As described in [28] and [35], the total number of operations can be reduced by exploiting submodularity and lazy evaluations. Although, it is faster to run, the algorithm still can have a worst-case time complexity of $\mathcal{O}(|X| \cdot n)$ for particular problem instances.

In practical applications, it is not possible to assume that f is a value oracle and requires an algorithm to compute. We let $T_{\text{eval}} : \mathbb{N} \to \mathbb{R}_+$ be the cost of computing f on $S \subseteq X$ of size $n \in \mathbb{N}$. Then, the total worst case time complexity of the greedy strategy is $\mathcal{O}(|X| \cdot n \cdot T_{\text{eval}}(n))$.

Chapter 4

Submodular Maximization with Limited Function Access

In this chapter we introduce the new problem of submodular maximization with limited function access. We highlight the challenges the problem pose as well as present properties of submodular functions that can be utilized to make the problem tractable. We purpose multiple algorithms that prove to be effective from a theoretical and experimental perspective. The main results for this chapter were submitted for publication and can be found in [8].

4.1 Problem Definition

A key focus is to understand the limitations of algorithms that only have access to partial information about the objective function. We make this precise in the next definition.

Definition 4.1.1 (k-wise Information)**.** *Given a submodular function $f : 2^X \to \mathbb{R}$, the k-wise information is defined as the set of tuples $\{(S, f(S))|S \subseteq X, |S| \leq k\}$, where $k \in \mathbb{N}$ and $k \leq |X|$. When $k = 2$, we refer to this set as pairwise information.*

An algorithm that has access to k-wise information can use evaluations of f on sets of size k or less to form decisions. We denote the class of such algorithms by $\Pi_{\text{k-wise}}$, or Π_{pairwise} when $k = 2$. The main objective that we have in mind is to study the Problem 3.5 with such limitations. More formally, we want to solve the following problem.

Problem 4.1.2 (k-wise Submodular Maximization Problem)**.** *Given a normalized, monotone and submodular function $f : 2^X \to \mathbb{R}$, we wish to solve the following program,*

$$\max_{S \subseteq X} f(S)$$
$$s.t.\ |S| \leq n.$$

Using an algorithm π that only has access to k-wise information about the function f, i.e., $\pi \in \Pi_{\text{k-wise}}$.

For most of this chapter we will be focusing on the pairwise case when $k = 2$, but we also show generalized results when k is larger. As previously described in the introduction, in the case when k is small such as $k = 2$, the information limitation is severe. Let $N = |X|$, then considering that there are 2^N subsets that can be evaluated, we are restricting our algorithms to only have access to evaluations for $N(N+1)/2$ of those subsets. When $N = 10$, the algorithm only has access to 4.4% of the possible function evaluations.

4.2 Inapproximability with k-wise Information

We begin with a negative result that addresses the inapproximability of Problem 4.1.2.

Theorem 4.2.1 (Inapproximability of Problem 4.1.2)**.** *Consider Problem 4.1.2 with k-wise information. Then for every algorithm $\pi \in \Pi_{\text{k-wise}}$, there exists a normalized, monotone and submodular function $f : 2^X \to \mathbb{R}$ such that*

$$f(S^\pi) \leq \frac{k}{n} f(S^*),$$

where S^π is the solution constructed by π and S^* is the optimal solution.

Proof. We begin by constructing a normalized, monotone and submodular function f. Consider a set X partitioned into two disjoint sets $X = V \cup V^*$, where $|V^*| = n$ and $|V| \geq n$. We define the function $f : 2^X \to \mathbb{R}_{\geq 0}$ as:

$$f(S) = \min\{|S \cap V|, k\} + |S \cap V^*|.$$

This function, given k it assigns a value of k to all sets of size k. The set V can be thought of as the general set and V^* as a special set where you are guaranteed to get value if you

select an element from V^*. However, the value for all sets S where $|S| \leq k$, get mapped their cardinality.

We now show that f is submodular. Consider any $A \subseteq B \subseteq X$ and $x \in X \backslash B$, we show that

$$f(x|A) \geq f(x|B).$$

First notice that $f(x|A)$ and $f(x|B)$ are each either 0 or 1, since adding an element can increase the function by at most one. There are two cases to consider.

Case 1 ($x \in V^*$): In this case $f(x|A) = 1$, since $A \cup \{x\}$ has one more element in V^* than A. Since $f(x|B) \leq 1$, the result follows.

Case 2 ($x \in V$): We assume that $f(x|B) = 1$ as otherwise the result holds. Since $f(x|B) = 1$ and $x \in V$, we must have $|B \cap V| \leq k$. This implies that $|A \cap V| \leq k$ since $A \subseteq B$. Thus $f(x|A) = 1$ and the result holds.

We now show that f is normalized. If $S = \emptyset$ then

$$|S \cap V| = |S \cap V^*| = 0$$

which implies that $f(\emptyset) = 0 + 0$ and therefore is normalized. Finally, we show that f is monotone. Let $A \subset B \subseteq X$, then

$$f(B) = f(A) + f(B \backslash A|A) \tag{4.1}$$

$$= f(A) + \sum_{i=1}^{|B\backslash A|} f(x_i|A \cup \{x_{i-1}, \ldots, x_1\}). \tag{4.2}$$

Where $B \backslash A = \{x_1, \ldots, x_{|B\backslash A|}\}$. By our original observation that $f(x|S)$ can only take on values 0 or 1 implies that

$$\sum_{i=1}^{|B\backslash A|} f(x_i|A \cup \{x_{i-1}, \ldots, x_1\}) \geq 0.$$

Therefore we have,

$$f(A) + \sum_{i=1}^{|B\backslash A|} f(x_i|A \cup \{x_{i-1}, \ldots, x_1\}) \geq f(A), \tag{4.3}$$

which proves that f is monotone.

Since f is normalized, monotone and submodular it is now a valid function for Problem 4.1.2. For any set S with $|S| \leq k$ we have that $f(S) = |S|$, which reveals no information on which elements of S are in V or V^*. Hence, from the perspective of an algorithm in $\Pi_{\text{k-wise}}$, the elements in X are indistinguishable. Given any algorithm $\pi \in \Pi_{\text{k-wise}}$, there exists an assignment of the elements of X to V and V^* such that $f(S^\pi) = k$. Since the optimal solution is $S^* = V^*$ achieving a value of $f(S^*) = n$, we obtain the desired result. □

This result highlights the challenges that arise under k-wise information constraints. As shown in the proof, there exist functions where their marginal returns with respect to sets of size k or less, tell you nothing about the marginals with respect to the sets with sizes greater than k.

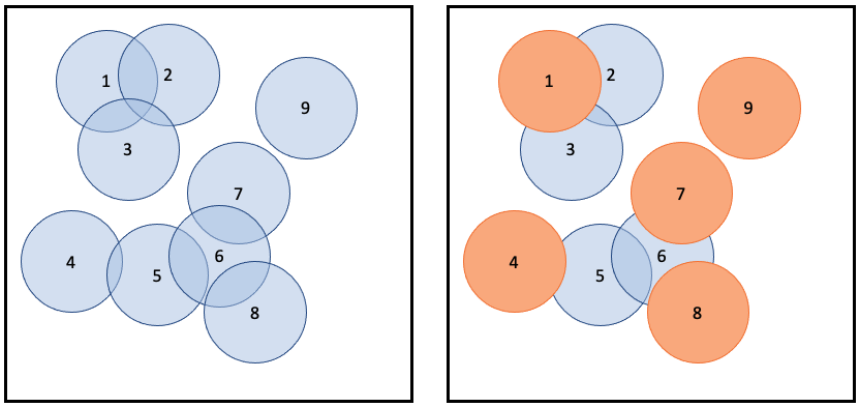

Figure 4.1: Left: Set of sensor footprints that an algorithm could potentially select. The objective is to select five sensors that maximize the area covered by the union of their footprints. Right: A set of 5 sensors that maximize the objective.

The sensor coverage example depicted in Figure 4.1 can provide intuition into scenarios where given pairwise information, an approximation algorithm could be derived. In Figure 4.1, the marginal returns of a single sensor given a subset of the other sensors can be approximated by taking its area and subtracting the pairwise overlaps between itself and other sensors footprints. The pairwise overlaps can be derived using only pairwise information, therefore the pairwise information can be used to infer the values of the higher order marginals. For example if the pairwise overlaps were large then we should expect $f(x|S) \approx 0$ but if the pairwise overlaps were small then we should expect $f(x|S) \approx f(x)$. Notice in Figure 4.1, for the subset of sensors that maximize the area covered by their union, there are no pairwise overlaps between the sensor footprints.

In this work, we focus primarily on the $k = 2$ case where algorithms only have access to pairwise information, but we do provide some results for the general case. In the following sections, we characterize submodular functions where the marginals returns of elements with respect to sets of size k or smaller are informative of the higher order marginals. We accomplish this by using new notions of curvature.

4.3 Pairwise Algorithms

4.3.1 Optimistic Algorithm

A natural strategy to tackle Problem 4.1.2 with a pairwise information constraint is to greedily select elements that maximize an *estimate* of the marginal returns, similarly to [53]. First, note that

$$\min_{x_j \in A} f(x|x_j) \geq f(x|A), \tag{4.4}$$

which holds by submodularity of f, because for all $x_j \in A$, we have $f(x|x_j) \geq f(x|A)$. We define a simple estimate of the marginal returns of f as the left hand side of (4.4)

$$\bar{f}(x|A) := \min_{x_j \in A} f(x|x_j). \tag{4.5}$$

The pairwise marginal for all $x_j \in A$ upper bounds $f(x|A)$ and hence we choose the minimum as it is the best available estimate of the true value of $f(x|A)$. In a nearly identical style to classical greedy strategy in Algorithm 1, we now define a new greedy strategy.

Algorithm 2: Optimistic Greedy Algorithm

Input: Base set X, submodular function f and cardinality constraint n
Result: Approximate solution $S \subseteq X$

$S_i \leftarrow \emptyset$ for all $i \in \{0, \ldots, n\}$;
for $i \leftarrow 1, \ldots, n$ **do**
  $x_i \leftarrow \arg\max_{x \in X \setminus S_{i-1}} \bar{f}(x|S_{i-1})$;
  $S_i \leftarrow S_{i-1} \cup \{x_i\}$;
end
$S \leftarrow S_n$

Throughout this work we will refer to Algorithm 2 as the *optimistic* algorithm. In essence, the optimistic algorithm aims to greedily select elements with the maximum *potential* marginal returns.

4.3.2 Approximate Value Oracles

To characterize the performance of the optimistic algorithm, we consider the problem through the lens of maximizing a submodular objective function via a surrogate objective

function. Following [53], we will discuss how to determine performance guarantees when using such surrogates.

Let $\{x_1, \ldots, x_n\} \subseteq X$ be the choices made by some algorithm. We denote the choices selected after the i-th iteration of the algorithm by $S_i = \{x_1, \ldots, x_i\}$. Let $\{x_1^g, \ldots, x_n^g\} \subseteq X$ be the such that each x_i^g maximize the marginal return of f given S_{i-1}. More formally defined as:

$$x_i^g = \arg\max_{x \in X \setminus S_{i-1}} f(x|S_{i-1}). \tag{4.6}$$

The set $\{x_1^g, \ldots, x_n^g\} \subseteq X$ represents the elements that a greedy strategy with full information about the objective f would have selected, given that it had previous selected S_{i-1}. Using these values, we can measure the quality of a given algorithm's choices compared to that of an algorithm with full information about the objective. We do this by finding $\alpha_i \in \mathbb{R}_+$, for $i \in \{1, \ldots, n\}$ such that

$$\alpha_i f(x_i|S_{i-1}) \geq f(x_i^g|S_{i-1}). \tag{4.7}$$

By the greedy choice property of x_i^g, we have that

$$f(x_i|S_{i-1}) \leq f(x_i^g|S_{i-1}).$$

Hence, $\alpha_i \geq 1$ for all $i \in \{1, \ldots, n\}$. From this point on, we call each α_i the approximation factor associated with x_i.

In the general framework proposed in [53], the objective is to greedily maximize multiple surrogate objective functions to produce approximate solutions. For our problem where we are constrained to use only pairwise information and we simply greedily maximize using a single surrogate function $\bar{f}(x|S)$. We provide a simplified version of [53, Theorem 1] as follows.

Theorem 4.3.1. *Suppose that $S = \{x_1, \ldots, x_n\} \subseteq X$ is the set of elements selected by an algorithm and α_i for $i \in \{1, \ldots, n\}$ is the set of approximation factors corresponding S that satisfy (4.7). Let S^* be the optimal solution to Problem 3.5. Then,*

$$f(S) \geq \left(1 - e^{-\frac{1}{n}\sum_{i=1}^{n}\frac{1}{\alpha_i}}\right) f(S^*). \tag{4.8}$$

Given that we only maximize one surrogate function and in order to keep this work self contained, we provide a simplified proof of this result.

Proof. Let S^* be the solution to Problem 3.5. Recall Proposition 2.1 from [37] that f is a monotone submodular set function on X if and only if $f(T) \leq f(S) + \sum_{x_j \in T \setminus S} f(x_j|S)$ for all $S, T \subseteq X$. If S is empty, then

$$f(S^*) \leq \sum_{x_i^* \in S^*} f(x_i^*) \leq n f(x_1^*) \leq n\alpha_1 f(x_1). \tag{4.9}$$

Then if we apply the lemma again with S_j, we have

$$f(S^*) \leq f(S_j) + \sum_{x_j^* \in S^* \setminus S_j} f(x_j^*|S_j). \tag{4.10}$$

We also know that

$$\alpha_{j+1} f(x_{j+1}|S_j) \geq \max_{x \in X \setminus S_j} f(x|S_j) \geq f(x_j^*|S_j). \tag{4.11}$$

We now substitute (4.11) into (4.10) and write $f(S_j)$ as sum of its marginals to get the following:

$$\begin{aligned} f(S^*) &\leq f(S_j) + \sum_{x_j^* \in S^* \setminus S_j} \alpha_{j+1} f(x_{j+1}|S_j) \\ &\leq \sum_{i=1}^{j} f(x_i|S_{i-1}) + n\alpha_{j+1} f(x_{j+1}|S_j). \end{aligned} \tag{4.12}$$

Equation (4.12) holds since $|S^* \setminus S_j| \leq n$. We will now rearrange (4.12) to get,

$$f(x_{j+1}|S_j) \geq \frac{1}{\alpha_{j+1} n} f(S^*) - \frac{1}{\alpha_{j+1} n} \sum_{i=1}^{j} f(x_i|S_{i-1}). \tag{4.13}$$

Now we will add $\sum_{i=1}^{j} f(x_i|S_{i-1})$ to both sides of (4.13) and simplify,

$$\sum_{i=1}^{j+1} f(x_i|S_{i-1}) \geq \frac{1}{\alpha_{j+1} k} f(S^*) + \frac{\alpha_{j+1} k - 1}{\alpha_{j+1} k} \sum_{i=1}^{j} f(x_i|S_{i-1}). \tag{4.14}$$

We will now prove by induction on j that

$$\sum_{i=1}^{j} f(x_i|S_{i-1}) \geq \frac{\prod_{i=1}^{j}(\alpha_i n) - \prod_{i=1}^{j}(\alpha_i n - 1)}{\prod_{i=1}^{j}(\alpha_i n)} f(S^*).$$

For base case $j = 1$ we will use (4.9) to get

$$f(x_1) \geq \frac{1}{\alpha_1 n} f(S^*).$$

Therefore proving the base case. Now assuming the claim holds for $j-1$. We will apply the inductive hypothesis to equation (4.14),

$$\sum_{i=1}^{j} f(x_i|S_{i-1}) \geq \frac{1}{\alpha_j n} f(S^*) + \frac{\alpha_j n - 1}{\alpha_j} \cdot \frac{\prod_{i=1}^{j-1}(\alpha_i n) - \prod_{i=1}^{j-1}(\alpha_i n - 1)}{\prod_{i=1}^{j-1}(\alpha_i n)} f(S^*). \tag{4.15}$$

Then after rearranging, we arrive at

$$f(S_j) \geq \frac{\prod_{i=1}^{j}(\alpha_i n) - \prod_{i=1}^{j}(\alpha_i n - 1)}{\prod_{i=1}^{j}(\alpha_i n)} f(S^*),$$

proving the inductive hypothesis. If we take $j = n$, we arrive at

$$f(S_n) \geq \frac{\prod_{i=1}^{n}(\alpha_i n) - \prod_{i=1}^{n}(\alpha_i n - 1)}{\prod_{i=1}^{n}(\alpha_i n)} f(S^*).$$

We will now lower bound right coefficients on $f(S^*)$ to simplify the bound. We can now cancel out the denominator of the coefficient to get,

$$\frac{\prod_{i=1}^{n}(\alpha_i n) - \prod_{i=1}^{n}(\alpha_i n - 1)}{\prod_{i=1}^{n}(\alpha_i n)} = 1 - \prod_{i=1}^{n} \frac{\alpha_i n - 1}{\alpha_i n} \tag{4.16}$$

$$= 1 - \prod_{i=1}^{n} \left(1 - \frac{1}{\alpha_i n}\right). \tag{4.17}$$

We can now upper bound each term in the product using $1 + x \leq e^x$ with $x = \frac{1}{\alpha_i n}$ to get a lower bound,

$$1 - \prod_{i=1}^{n} \left(1 - \frac{1}{\alpha_i n}\right) \geq 1 - \prod_{i=1}^{n} e^{-\frac{1}{\alpha_i n}} \tag{4.18}$$

$$= 1 - e^{-\frac{1}{n}\sum_{i=1}^{n} \frac{1}{\alpha_i}}. \tag{4.19}$$

Using this lower bound yields our result. □

Note that Theorem 4.3.1 relies on f being a normalized, monotone, and submodular function. This result can be applied to any algorithm for Problem 3.5, not just algorithms that only have access to pairwise information. An interesting remark about Theorem 4.3.1, is that the performance depends essentially on the average of the approximation factors. Some of these factors could be large compared to the others, but as long as most of them are small, adequate performance is maintained.

4.3.3 Optimistic Algorithm Approximation Performance

We aim to provide approximation guarantees for the optimistic algorithm. To give intuition for what we are about to present, we consider the following example.

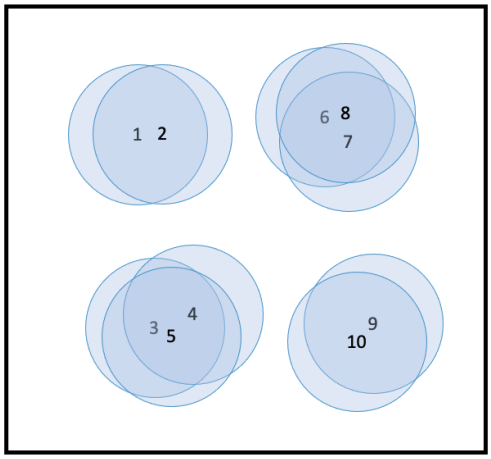

Figure 4.2: Example sensor coverage configuration where the optimistic algorithm performs better than uninformed greedy strategy.

Example 4.3.2. Consider the scenario depicted in Figure 4.2. Here, we wish to select four sensors that maximize the area of their combined footprints. One of the simplest algorithms that satisfies the pairwise information constraint is the *uninformed* greedy strategy which is described in Algorithm 3.

Algorithm 3: Uninformed Greedy Algorithm

Input: Base set X, submodular function f and cardinality constraint n
Result: Approximate solution $S \subseteq X$

$S_i \leftarrow \emptyset$ for all $i \in \{0, \ldots, n\}$;
for $i \leftarrow 1, \ldots, n$ **do**
    $x_i \leftarrow \arg\max_{x \in X \setminus S_{i-1}} f(x)$;
    $S_i \leftarrow S_{i-1} \cup \{x_i\}$;
end
$S \leftarrow S_n$

We refer to Algorithm 3 as *uninformed* because it only uses the most basic information about f, which is its evaluation of single elements. Using the uniformed greedy strategy in the scenario described in Figure 4.2 could potentially lead to poor performance. This strategy cannot distinguish between its choices and therefore could select four sensors that almost perfectly overlap with each other(i.e., in the same pile), resulting in a low objective value. Alternatively, if we had used the optimistic algorithm, once one element is selected

from a pile, the pairwise upper bound (4.5) for the other elements in the same pile would be low. In later iterations, the optimistic algorithm would avoid selecting elements from piles where previous elements have been selected from. Interestingly, we see that for each i, the difference between $f(x_i|S_{i-1})$ and $\bar{f}(x_i|S_{i-1})$ is small for elements selected by the optimistic algorithm. We notice that in these scenarios, the value of $\bar{f}(x_i|S_{i-1})$ provides accurate information about the value $f(x_i|S_{i-1})$.

This is the idea we wish to capture in the following result.

Theorem 4.3.3. *Let $S_{i-1} = \{x_1, \ldots, x_{i-1}\} \subseteq X$ be the partial solution of optimistic algorithm after $(i-1)$ iterations, and let $x_i \in X$ be the element selected during the i-th iteration. Then we have that,*

$$\alpha_i^{\text{opt}} = \begin{cases} 1 & i \in \{1, 2\} \\ \frac{\bar{f}(x_i|S_{i-1})}{f(x_i|S_{i-1})} & i > 2 \end{cases}, \tag{4.20}$$

satisfy (4.7) for all $i \leq n$.

Proof. Let x_i^g be the true greedy choice at iteration i given S_{i-1} given by (4.6). For $i \in \{1, 2\}$, we have that

$$\bar{f}(x|S_{i-1}) = f(x|S_{i-1}).$$

Hence, $x_i = x_i^g$ and therefore, we can let $\alpha_1^{\text{opt}} = \alpha_2^{\text{opt}} = 1$. For $i > 2$, based on from (4.7) let $\alpha_i^{\min}$ be the smallest value such that (4.7) holds, which can be written as

$$\alpha_i^{\min} = \frac{f(x_i^g|S_{i-1})}{f(x_i|S_{i-1})}. \tag{4.21}$$

Any approximation factor α_i such that $\alpha_i \geq \alpha_i^{\min}$ will satisfy (4.7). We will now upper bound $\alpha_i^{\min}$ as follows:

$$\begin{aligned} \alpha_i^{\min} &= \frac{f(x_i^g|S_{i-1})}{f(x_i|S_{i-1})} \\ &\leq \frac{\bar{f}(x_i^g|S_{i-1})}{f(x_i|S_{i-1})} && (4.22) \\ &\leq \frac{\bar{f}(x_i|S_{i-1})}{f(x_i|S_{i-1})}, && (4.23) \end{aligned}$$

where (4.22) holds by definition and (4.23) holds by the greedy choice property of the optimistic algorithm. Setting α_i^{opt} to be the right hand side of (4.23) for $i > 2$, we conclude the proof. □

The following corollary is an immediate consequence of Theorem 4.3.3.

Corollary 4.3.4. *Let $S \subseteq X$ be the solution produced by the optimistic algorithm and S_{i-1} be the partial solution after $(i-1)$ iterations of the optimistic algorithm and let $x_i \in X$ be the element selected during the i-th iteration, then we have*

$$f(S) \geq \left(1 - e^{-\frac{1}{n}\left(2+\sum_{i=3}^{n} \frac{f(x_i|S_{i-1})}{\bar{f}(x_i|S_{i-1})}\right)}\right) f(S^*), \tag{4.24}$$

where S^* is the solution to Problem 4.1.2 with $k = 2$.

The result immediately follows combining Theorem 4.3.1 and Theorem 4.3.3 together.

We see that the approximation performance of the algorithm is dictated by the sum in the exponent of (4.24). We can interpret the exponent as the mean of the set,

$$\left\{1, 1, \frac{f(x_3|S_2)}{\bar{f}(x_3|S_2)}, \ldots, \frac{f(x_n|S_{n-1})}{\bar{f}(x_n|S_{n-1})}\right\}.$$

This implies that, to get adequate performance from the optimistic algorithm, we need the value of $\bar{f}(x_i|S_{i-1})$ to be close to $f(x_i|S_{i-1})$ on *average*.

We also see that $\frac{f(x_i|S_{i-1})}{\bar{f}(x_i|S_{i-1})}$ is closely related to the tradition notion of curvature described in Definition 3.1.5. Let us define a new notion of curvature.

Definition 4.3.5 (k-Marginal Curvature)**.** *The k-marginal curvature of a submodular function f given $S \subseteq X$ and $x \in X \backslash S$ is defined as*

$$c_k(x|S) = 1 - \max_{A \subseteq S, |A| < k} \frac{f(x|S)}{f(x|A)}. \tag{4.25}$$

To analyze the optimistic algorithm that only has access to pairwise information, we will work with the 2-marginal curvature, which can be written as

$$c_2(x|S) = 1 - \frac{f(x|S)}{\bar{f}(x|S)}.$$

This allows us to rewrite (4.24) as follows,

$$f(S) \geq \left(1 - e^{-\frac{1}{n}\left(2+\sum_{i=3}^{n} 1 - c_2(x_i|S_{i-1})\right)}\right) f(S^*). \tag{4.26}$$

We can characterize the worst case performance of the optimistic algorithm in terms of the average of the 2-marginal curvatures, which capture the intuition in Example 4.3.2. In Figure 4.2, the elements will have 2-marginal curvatures close to zero, resulting in a strong approximation bound.

Remark 4.3.6. This 2-marginal curvature is similar to the traditional notion of curvature in Definition 3.1.5, but characterizes the relationship between the values of the pairwise upper bounds $\bar{f}(x|S)$ and true values of $f(x|S)$. A key difference between the two definitions, is there exists situations where the values of the 2-marginal curvatures can be close to 0 even though the value of the traditional curvature is close to 1. The sensor coverage function, described in Figure 4.2, is an example of a function where the traditional curvature is close to 1 and the values of the 2-marginal curvatures are close to 0.

4.3.4 Extension to k-wise Information

The analysis from Subsection 4.3.3 can be naturally extended to the problem with k-wise information. Suppose that we wish to solve Problem 4.1.2 using an algorithm with k-wise information. We extend the pairwise optimistic algorithm to the k-wise optimistic as follows. Let us define a new upper bound on the marginal returns using k-wise information. Let $x \in X$ and $S \subseteq X$, then we have the following upper bound

$$\min_{A \subseteq S, |A| < k} f(x|A) \geq f(x|S), \tag{4.27}$$

which holds by submodularity of f. We will denote the left hand side of (4.27) as

$$\bar{f}_k(x|S) := \min_{A \subseteq S, |A| < k} f(x|A).$$

Let us define the k-wise optimistic algorithm.

Algorithm 4: k-wise Optimistic Greedy Algorithm

Input: Base set X, submodular function f and cardinality constraint n
Result: Approximate solution $S \subseteq X$

$S_i \leftarrow \emptyset$ for all $i \in \{0, \ldots, n\}$;
for $i \leftarrow 1, \ldots, n$ **do**
  $x_i \leftarrow \arg\max_{x \in X \setminus S_{i-1}} \bar{f}_k(x|S_{i-1})$;
  $S_i \leftarrow S_{i-1} \cup \{x_i\}$;
end
$S \leftarrow S_n$

By submodularity we have that $\bar{f}(x|S) \geq \bar{f}_k(x|S) \geq f(x|S)$ for $k > 1$. Our result is stated next.

Theorem 4.3.7. *Let $S_{i-1} \subseteq X$ be the partial solution of k-wise optimistic algorithm after $(i-1)$ iterations and $x_i \in X$ be the element selected during the i-th iteration. Then, we have*

$$\alpha_i^{\text{opt,k}} = \begin{cases} 1 & i \leq k \\ \frac{\bar{f}_k(x_i|S_{i-1}))}{f(x_i|S_{i-1})} & i > k \end{cases}, \tag{4.28}$$

satisfy (4.7) for all $i \leq n$.

The proof follows similarly to the proof of Theorem 4.3.3.

Proof. Let x_i^g be the true greedy choice at iteration i given S_{i-1}. For $i \leq k$ we have that $x_i = x_i^g$ by the definition of $\bar{f}_k(x_i|S_{i-1})$. Therefore we have, $\alpha_1^{\text{opt,k}} = \cdots = \alpha_k^{\text{opt,k}} = 1$. The minimum possible approximation factor we have can be written as

$$\alpha_i^{\min} = \frac{f(x_i^g|S_{i-1})}{f(x_i|S_{i-1})}. \tag{4.29}$$

Any approximation factor α_i such that $\alpha_i \geq \alpha_i^{\min}$ will satisfy equation (4.7). We will now upper bound $\alpha_i^{\min}$ as follows,

$$\begin{aligned} \alpha_i^{\min} &= \frac{f(x_i^g|S_{i-1})}{f(x_i|S_{i-1})} \\ &\leq \frac{\bar{f}_k(x_i^g|S_{i-1})}{f(x_i|S_{i-1})} && (4.30) \\ &\leq \frac{\bar{f}_k(x_i|S_{i-1})}{f(x_i|S_{i-1})}, && (4.31) \end{aligned}$$

where (4.30) holds by the definition of the upper bound (4.27). Equation (4.31) holds by the greedy choice property of the k-wise optimistic algorithm. Therefore the right hand side of (4.31) is a valid approximation factor. We set $\alpha_i^{\text{opt,k}}$ to the right hand side of (4.31) for $i > k$, we conclude our proof. □

The following corollary is an immediate consequence of Theorem 4.3.7.

Corollary 4.3.8. *Let $S \subseteq X$ be the solution produced by the k-wise optimistic algorithm and $S_{i-1} \subseteq S$ be the partial solution after $(i-1)$ iterations of the algorithm and let $x_i \in S$ be the element selected at the i-th iteration, then we have*

$$f(S) \geq \left(1 - e^{-\frac{1}{n}\left(k + \sum_{i=k+1}^{n} \frac{f(x_i|S_{i-1})}{\bar{f}_k(x_i|S_{i-1})}\right)}\right) f(S^*). \tag{4.32}$$

Where S^* is the solution to Problem 4.1.2.

Note that using this result and definition 4.3.5, we can write (4.32) as

$$f(S) \geq \left(1 - e^{-\frac{1}{n}\left(k+\sum_{i=k+1}^{n} 1-c_k(x_i|S_{i-1})\right)}\right) f(S^*). \tag{4.33}$$

Comparing this to the scenario with pairwise information, we see that access to more information improves approximation guarantees. In particular, since $c_2(x|S) \geq c_k(x|S)$ for all $S \subseteq X$ and $x \in X \backslash S$, we can guarantee that

$$\frac{1}{n}\left(k + \sum_{i=k+1}^{n} 1 - c_k(x_i|S_{i-1})\right) \geq \frac{1}{n}\left(2 + \sum_{i=3}^{n} 1 - c_2(x_i|S_{i-1})\right). \tag{4.34}$$

This implies that the approximation bound in Corollary 4.3.8 is tighter than Corollary 4.3.4.

Having access to k-wise information provide us with stronger approximation bounds, but we trade-off computation performance. We are required to compute the minimum marginal over all subsets $A \subseteq S_{i-1}$, where $|A| \leq k$ for each $x \in X$ during each iteration of Algorithm 4. When $k \leq |S_{i-1}|$, we need to search $\binom{|S_{i-1}|}{k-1}$ subsets of S_{i-1} to find the minimum. This becomes computationally expensive as S_{i-1} grows larger. If $k = 3$, the computation for each marginal estimate is quadratic in $|S_{i-1}|$. From a practical perspective, we can actually compute the pairwise optimistic algorithm efficiently; We will discuss this in Section 4.5.

4.3.5 General Performance Bound for k-wise Optimistic Algorithm

From the analysis in Sections 4.3.3 and 4.3.4 we notice that for the bound in Corollary 4.3.8 to be computed, the function evaluations of selected elements must be known. This may not be desirable in some applications due to the fact that the user will have to run the algorithm (which could potentially be expensive) to know the performance bounds. We can provide a bound that is potentially weaker but does not require us to know the selected elements $x_1, \ldots, x_n \in X$ before computing.

Let us define similar a notion of curvature to the k-marginal curvature but is closer to the tradition notion in Definition 3.1.5.

Definition 4.3.9 (Total k-Marginal Curvature)**.** *Let f be a submodular function then, the curvature $\bar{c}_k$ of f is defined as,*

$$\bar{c}_k = 1 - \min_{S \subseteq X, x \in X \setminus S} \frac{f(x|S)}{\bar{f}_k(x|S)}. \tag{4.35}$$

Using this definition we can follow a similar process to arrive at a approximation bound for the k-wise optimistic algorithm.

Theorem 4.3.10. *Let $S \subseteq X$ be the solution produced by the k-wise optimistic algorithm, and $S^* \subseteq X$ be the solution to Problem 4.1.2, then we have*

$$f(S) \geq \left(1 - e^{-\left(1 - \frac{n-k}{n}\bar{c}_k\right)}\right) f(S^*) \geq \left(1 - e^{-(1-\bar{c}_k)}\right) f(S^*). \tag{4.36}$$

Proof. Let $S = \{x_1, \ldots, x_n\}$ be the solution produced by the k-wise optimistic algorithm, and $S_i = \{x_1, \ldots, x_i\}$. Let x_i^g be the true greedy choice at iteration i given S_{i-1}. Let $\alpha_1, \ldots, \alpha_n$ be the approximation factors for the solution S. For $i \leq k$ we have that $x_i = x_i^g$ by the definition of $\bar{f}_k(x_i|S_{i-1})$. Therefore we have, $\alpha_1 = \cdots = \alpha_k = 1$. The minimum possible approximation factor, can be written as

$$\alpha_i^{\min} = \frac{f(x_i^g|S_{i-1})}{f(x_i|S_{i-1})}. \tag{4.37}$$

Any approximation factor α_i such that $\alpha_i \geq \alpha_i^{\min}$ will satisfy equation (4.7). We will now upper bound $\alpha_i^{\min}$ as follows for $i > k$,

$$\begin{aligned}
\alpha_i^{\min} &= \frac{f(x_i^g|S_{i-1})}{f(x_i|S_{i-1})} \\
&\leq \frac{\bar{f}_k(x_i^g|S_{i-1})}{f(x_i|S_{i-1})} && (4.38) \\
&\leq \frac{\bar{f}_k(x_i|S_{i-1})}{f(x_i|S_{i-1})} && (4.39) \\
&\leq \max_{\bar{S} \subseteq X, x \in X \setminus \bar{S}} \frac{\bar{f}_k(x|\bar{S})}{f(x|\bar{S})},
\end{aligned}$$

where (4.38) holds by the definition of the k-wise upper bound, and (4.39) holds by the greedy choice property of the k-wise optimistic algorithm. To use Theorem 4.3.1 to produce an approximation bound we need to use the inverse of the approximation factor,

$$\begin{aligned}\frac{1}{\alpha_i} &= \frac{1}{\max_{\bar{S}\subseteq X, x\in X\setminus\bar{S}} \frac{\bar{f}_k(x|\bar{S})}{f(x|\bar{S})}} \\ &= \min_{\bar{S}\subseteq X, x\in X\setminus\bar{S}} \frac{f(x|\bar{S})}{\bar{f}_k(x|\bar{S})} \qquad (4.40) \\ &= 1-\bar{c}_k, \qquad (4.41)\end{aligned}$$

where (4.40) holds since $\bar{f}_k(x|\bar{S}) \geq f(x|\bar{S})$ for all $\bar{S} \subseteq X, x \in X\setminus\bar{S}$ and (4.41) holds by the definition of total k-marginal curvature. We will now substitute in $\frac{1}{\alpha_1}, \ldots, \frac{1}{\alpha_n}$ into Theorem 4.3.1 and simplify to arrive at the approximation bound,

$$\begin{aligned}f(S) &\geq \left(1 - e^{-\left(1-\frac{n-k}{n}\bar{c}_k\right)}\right) f(S^*) \\ &\geq \left(1 - e^{-(1-\bar{c}_k)}\right) f(S^*), \qquad (4.42)\end{aligned}$$

where (4.42) holds since $1 - \frac{n-k}{n}\bar{c}_k \geq 1 - \bar{c}_k$, concluding the proof. □

Theorem 4.3.10 shows that the k-wise optimistic algorithm provides a constant factor approximation for Problem 4.1.2 which is dependent on this new notion of curvature. Although, the performance bounds are not as tight as Corollary 4.3.8, it does show that the fundamental quantity that underlies the performance of the k-wise optimistic algorithm is the total k-marginal curvature. We do not provide experiments showing the approximation bounds produced by Theorem 4.3.10 since the bound produced by Corollary 4.3.8 is stronger.

4.4 Pairwise Algorithms Utilizing Supermodularity of Conditioning

In this section we introduce an additional property that a submodular function can possess which is useful when we only have access to pairwise information. The property is called *supermodularity of conditioning* and is related to monotonicity and allows us to compute performance bounds for algorithms "post-hoc" using only pairwise information.

4.4.1 Post-Hoc Approximation Performance Bounds

To characterize the approximation performance of an algorithm $\pi \in \Pi_{\text{pairwise}}$ using Theorem 4.3.3, we are required to compute the full marginal of the function f, which may not be available in practice. Alternatively we can execute an algorithm π and produce a solution

$$S^\pi = \{x_1^\pi, \dots, x_n^\pi\},$$

which can be used determine an approximation ratio $\gamma \in [0, 1]$ such that

$$f(S^\pi) \geq \gamma f(S^*),$$

where S^* is the optimal solution to Problem 4.1.2. This is done by bounding α_i in (4.7) using only pairwise information and applying Theorem 4.3.1. As described in the proof of Theorem 4.3.3, the smallest value of α_i that will satisfy (4.7) is $\alpha_i^{\min}$.

Let $S_i^\pi = \{x_1^\pi, \dots, x_i^\pi\} \subseteq X$, be the partial solution of S^π. Let $\{x_1^g, \dots, x_n^g\} \subseteq X$ be the set of elements defined by (4.6) with $S_i = S_i^\pi$. Then, we have that

$$\alpha_i^{\min} = \frac{f(x_i^g | S_{i-1}^\pi)}{f(x_i^\pi | S_{i-1}^\pi)} \leq \frac{\max_{x \in X \setminus S_{i-1}^\pi} \bar{f}(x | S_{i-1}^\pi)}{f(x_i^\pi | S_{i-1}^\pi)}. \tag{4.43}$$

By lower bounding $f(x_i^\pi | S_{i-1}^\pi)$ using only pairwise information, we obtain an α_i that satisfies (4.7), and therefore Theorem 4.3.1 allows us to find an approximation ratio γ.

If we impose an additional monotonicity property on f called *supermodularity of conditioning*, then we are able to find a lower bound on the marginal returns of f using only pairwise information.

Definition 4.4.1 (Supermodularity of Conditioning)**.** *A submodular function f possess the property of supermodularity of conditioning if for all $S \subseteq X$, $A \subseteq B \subseteq X$ and $C \subseteq X \setminus B$, we have that*

$$f(S|A) - f(S|A, C) \geq f(S|B) - f(S|B, C). \tag{4.44}$$

Supermodularity of conditioning is a higher order monotonicity property which describes how the *redundancy* between two sets are affected by conditioning. Suppose that $A = \emptyset$, the redundancy between S and C is $f(S) - f(S|C)$. If both terms are further conditioned by a set B, then the difference between the terms will be reduced. Meaning that

$$f(S) - f(S|C) \geq f(S|B) - f(S|B, C).$$

Supermodularity of conditioning has been used in the context of distributed submodular maximization in [5]. Some notable examples of functions that exhibit supermodularity of conditioning are weighted set coverage, area coverage and probabilistic set coverage. We recall the following result from [5].

Lemma 4.4.2 (Pairwise Redundancy Bound)**.** *Let $f : 2^X \to \mathbb{R}$ be a submodular function, that exhibits supermodularity of conditioning and let $A, B, C \subseteq X$ be disjoint subsets. Then*

$$f(A|B) - f(A|B, C) \leq \sum_{c \in C} f(c) - f(c|A). \tag{4.45}$$

We can now state the following result which establishes a lower bound on the marginal returns of f.

Theorem 4.4.3 (Pairwise Marginal Lower Bound)**.** *Let f be a submodular function that exhibits supermodularity of conditioning. Then for $S \subseteq X$ and $x \in X \backslash S$, we have*

$$f(x|S) \geq f(x) - \sum_{x_j \in S} f(x) - f(x|x_j). \tag{4.46}$$

Proof. Since f exhibits supermodularity of conditioning, applying Lemma 4.4.2 with $A = \{x\}$, $B = \emptyset$ and $C = S$, we have

$$\begin{aligned} f(x) - f(x|S) &\leq \sum_{x_j \in S} f(x_j) - f(x_j|x) \\ &= \sum_{x_j \in S} f(x) - f(x|x_j), \end{aligned} \tag{4.47}$$

where the last equality hold by the definition of the marginal return, yielding the result. □

For $x \in X$ and $S \subseteq X$ we define

$$\underline{f}(x|S) := f(x) - \sum_{x_j \in S} f(x) - f(x|x_j).$$

We can now directly use the lower bound on the marginal returns to bound $\alpha_i^{\min}$. Now we have that $\alpha_i^{\min} \leq \alpha_i^{\text{pairwise}}$, where

$$\alpha_i^{\text{pairwise}} = \begin{cases} \frac{\max_{x \in X \backslash S_{i-1}^\pi} \bar{f}(x|S_{i-1}^\pi)}{\underline{f}(x_i^\pi|S_{i-1}^\pi)} & \underline{f}(x_i^\pi|S_{i-1}^\pi) > 0 \\ \infty & \underline{f}(x_i^\pi|S_{i-1}^\pi) \leq 0 \end{cases}. \tag{4.48}$$

Now $\alpha_i^{\text{pairwise}}$ satisfies (4.7) and is computable using only pairwise information. Note that we need to set $\alpha_i^{\text{pairwise}} = \infty$ when $f(x_i^\pi | S_{i-1}^\pi) \leq 0$ as otherwise, the resulting $\alpha_i^{\text{pairwise}}$ would not upper bound $\alpha_i^{\min}$. We now present an Algorithm 5 that given $\pi \in \Pi_{\text{pairwise}}$ and pairwise information about f, produces an approximation bound γ such that the solution S^π satisfies $f(S^\pi) \geq \gamma f(S^*)$.

Algorithm 5: Pairwise Information Post-Hoc Bound

Input: Base set X and $S^\pi \subseteq X$
Result: γ such that $f(S) \geq \gamma f(S^*)$

$S_0^\pi \leftarrow \emptyset$;
for $i \leftarrow 1, \ldots, n$ **do**
  select x_i^π from $S^\pi \backslash S_{i-1}^\pi$;
  if $\underline{f}(x_i^\pi | S_{i-1}^\pi) > 0$ **then**
    $\alpha_i \leftarrow \frac{\max_{x \in X \backslash S_{i-1}^\pi} \bar{f}(x | S_{i-1}^\pi)}{\underline{f}(x_i^\pi | S_{i-1}^\pi)}$;
  else
    $\alpha_i \leftarrow \infty$;
  $S_i^\pi \leftarrow S_{i-1}^\pi \cup \{x_i^\pi\}$;
end
$\gamma \leftarrow 1 - e^{-\frac{1}{n} \sum_{i=1}^n \frac{1}{\alpha_i}}$;

Algorithm 5 provides us a means to find performance bounds for an arbitrary algorithm π given only pairwise information about f. This does not guarantee the performance before execution, but it does provide a method to verify performance of an algorithm without having to explicitly compute $f(S^\pi)$ and $f(S^*)$.

We end this section with a few remarks about supermodularity of conditioning and the lower bound.

Remark 4.4.4. (On Supermodularity of Conditioning) Note that assuming f possess supermodularity of conditioning does not affect the hardness results for submodular maximization. As shown in [5], the weighted set cover problem objective possesses supermodularity of conditioning and hardness results still exist for the problem.

Remark 4.4.5. (On Lower Bound Interpretation) We used the intuition from area and set coverage functions to find the pairwise lower bound. For example, let f be the set coverage function

$$f(S) = \left| \cup_{x_s \in S} A_{x_s} \right|.$$

Where $|\cdot|$ denotes the cardinality of a set and each A_x is a set that uniquely correspond to an element $x \in X$. We can interpret the marginal gain of adding $x \in X$ to $S \subseteq X$ as

$$f(x|S) = |A_x| - |A_x \cap (\cup_{x_s \in S} A_{x_s})|\,.$$

If we expand the last term using the inclusion-exclusion principle and then truncate the sum to the first order terms, we yield a lower bound [13] as follows,

$$f(x|S) \geq |A_x| - \sum_{x_s \in S} |A_x \cap A_{x_s}|.$$

Which can be rewritten as,

$$f(x|S) \geq f(x) - \sum_{x_s \in S} f(x) - f(x|x_s),$$

which is the same form as the lower bound in (4.47). We utilized supermodularity of conditioning to generalize this idea to a larger class of submodular functions that are not necessarily coverage functions.

4.4.2 Pessimistic Algorithm

Next, we propose another pairwise algorithm which we call the *pessimistic* algorithm.

Algorithm 6: Pessimistic Greedy Algorithm

Input: Base set X, submodular function f and cardinality constraint n
Result: Approximate solution $S \subseteq X$

$S_i \leftarrow \emptyset$ for all $i \in \{0, \ldots, n\}$;
for $i \leftarrow 1, \ldots, n$ **do**
  $x_i \leftarrow \arg\max_{x \in X \setminus S_{i-1}} \underline{f}(x|S_{i-1})$;
  $S_i \leftarrow S_{i-1} \cup \{x_i\}$;
end
$S \leftarrow S_n$

Similar to the optimistic algorithm, we greedily select elements with the highest *guaranteed* value. This is equivalent to greedily minimizing the approximation factors $\alpha_i^{\text{pairwise}}$ in Algorithm 5. What differs from the optimistic algorithm is that we require the additional assumption of supermodularity of conditioning on the objective function. This algorithm

enjoys similar guarantees as the optimistic algorithm when curvature assumptions are made but can outperform in certain scenarios. Later, in our experimental results we show the effectiveness of the pessimistic algorithm for a probabilistic coverage problem.

Using another alternative definition of curvature, we can produce a similar performance bound as the optimistic algorithm. Let us define the k-cardinality curvature as follows.

Definition 4.4.6 (k-Cardinality Curvature)**.** *We define the k-cardinality curvature τ_k as*

$$\tau_k = 1 - \min_{A \subseteq X, |A| < k, x \in X \setminus A} \frac{f(x|A)}{f(x)}. \tag{4.49}$$

What differs between this notion of curvature and the k-marginal curvature, is that it compares the values of the marginal of x with respect to sets with size less than k to the values of f evaluated on single elements. The 2-cardinality curvature can be written as

$$\tau_2 = 1 - \min_{y \in X, x \in X \setminus \{y\}} \frac{f(x|y)}{f(x)}.$$

We will use this definition to prove the following result for the pessimistic algorithm.

Theorem 4.4.7. *Let f be a normalized, monotone and submodular function that possesses supermodularity of conditioning. For the solution produced by the pessimistic algorithm the approximation factors,*

$$\alpha_i^{\text{pes}} = \begin{cases} 1 & i \leq 2 \\ \frac{1}{1 - \min\{(i-1)\tau_2, 1\}} & i > 2 \end{cases}, \tag{4.50}$$

satisfy (4.7) for all $i \leq n$.

Proof. Let $S = \{x_1, \ldots, x_n\}$ be the solution produced by the pessimistic algorithm and $S_i = \{x_1, \ldots, x_i\}$. Let x_i^g be the true greedy choice at iteration i given S_{i-1}. For $i = \{1, 2\}$, we have that

$$\underline{f}(x|S_{i-1}) = f(x|S_{i-1}).$$

Hence, $x_i = x_i^g$ and therefore, we can let $\alpha_1^{\text{pes}} = \alpha_2^{\text{pes}} = 1$. Using the definition of the 2-cardinality curvature we have the following inequalities. First,

$$\tau_2 \geq 1 - \frac{f(x|y)}{f(x)}, \tag{4.51}$$

for all $x, y \in X$, which holds by definition of τ_2. Next using the lower bound (4.46), we have

$$\begin{aligned} f(x|S) &\geq f(x) - \sum_{x_j \in S} f(x) - f(x|x_j) \\ &= f(x) \left(1 - \sum_{x_j \in S} 1 - \frac{f(x|x_j)}{f(x)} \right) \\ &\geq f(x)(1 - |S|\tau_2). \end{aligned}$$

Since we know that $f(x|S) \geq 0$, then we have

$$f(x|S) \geq \max\{\underline{f}(x|S), 0\} \geq f(x)(1 - \min\{|S|\tau_2, 1\}). \tag{4.52}$$

Using these inequalities we can bound the minimum achievable approximation factor as follows:

$$\begin{aligned} \alpha_i^{\min} &= \frac{f(x_i^g|S_{i-1})}{f(x_i|S_{i-1})} \\ &\leq \frac{f(x_i^g|S_{i-1})}{\max\{\underline{f}(x_i|S_{i-1}), 0\}} && (4.53) \\ &\leq \frac{f(x_i^g|S_{i-1})}{\max\{\underline{f}(x_i^g|S_{i-1}), 0\}} && (4.54) \\ &\leq \frac{f(x_i^g|S_{i-1})}{f(x_i^g)(1 - \min\{(i-1)\tau_2, 1\})} && (4.55) \\ &\leq \frac{1}{1 - \min\{(i-1)\tau_2, 1\}}. && (4.56) \end{aligned}$$

Where (4.53) holds by the definition of the lower bound, (4.54) holds by the greedy choice property of the pessimistic strategy and finally (4.55) holds by (4.52). Therefore if we set α_i^{pes} to the right hand side of equation (4.56) for $i > 2$, then α_i^{pes} is a valid approximation factor for Theorem 4.3.1. □

The following corollary immediately follows.

Corollary 4.4.8. *Let $S \subseteq X$ be the solution produced by the pessimistic algorithm, then we have*

$$f(S) \geq \left(1 - e^{-\frac{1}{n}\left(2 + \sum_{i=3}^{n}(1 - \min\{(i-1)\tau_2, 1\})\right)}\right) f(S^*). \tag{4.57}$$

Similar to the optimistic algorithm, we see that if τ_2 is small, $\underline{f}(x|S)$ closely represents the true value of $f(x|S)$. This bound on performance can be loose relative to the bound produced by Algorithm 5 due to (4.52) being coarse. The post-hoc bound produced by Algorithm 5 will provide a tighter bound on performance than Corollary 4.4.8.

Remark 4.4.9. The new notions of curvature defined in Definitions 4.3.5 and 4.4.6 are related to the traditional definition of curvature. Let c be the traditional curvature as described in Definition 3.1.5, let $S \subseteq X$ and $x \in X \backslash S$, both the k-marginal and k-cardinality curvature have similar inequalities.

$$c \geq c_k(x|S) \text{ and } c \geq \tau_k.$$

There are scenarios where c can be 1 and either $c_2(x|S)$ or τ_2 can be small. In Figure 4.1, we see that τ_2 will be small but $c(x|S)$ can be large. In Figure 4.1, let us denote the disks by their numbers d_i and let the area of each disk be equal to 1. Let $x = d_6$ and let $S = \{d_5, d_7, d_8\}$. Then $c_2(x|S)$ will be large because $\bar{f}(x|S) \approx 2/3$ and $f(x|S) \approx 0$. As previously described, τ_2 is small in this example because for the two disks, $x = d_5$ and $y = d_6$ with the most overlap, we have $f(x|y) \approx 2/3$ and $f(x) = 1$. Figure 4.2, describes the opposite case where $\tau_2 \approx 1$ and $c_2(x|S) \approx 0$ for any $S \in X$ and $x \in X \backslash S$.

Remark 4.4.10. It turns out that τ_k and $\bar{c}_k$ are closely related. We can rewrite the definition of τ_k as follow:

$$\begin{aligned}
\tau_k &= 1 - \min_{A \subseteq X, |A| < k, x \in X \backslash A} \frac{f(x|A)}{f(x)} \\
&= 1 - \min_{S \subseteq X, x \in X \backslash S, A \subseteq S, |A| < k} \frac{f(x|A)}{f(x)} \\
&= 1 - \min_{S \subseteq X, x \in X \backslash S} \frac{\min_{A \subseteq S, |A| < k} f(x|A)}{f(x)} \\
&= 1 - \min_{S \subseteq X, x \in X \backslash S} \frac{\bar{f}_k(x|S)}{f(x)}.
\end{aligned}$$

This means that τ_k and $\bar{c}_k$ only differ in weather the upper bound on the marginal is in the numerator or the denominator. The traditional curvature is written as

$$c = 1 - \min_{S \subseteq X, x \in X \backslash S} \frac{f(x|S)}{f(x)}.$$

We can interpret both $\bar{c}_k$ and τ_k as truncating the fraction in two different ways, both of which result in,

$$c \geq \bar{c}_k \text{ and } c \geq \tau_k.$$

If we truncate the numerator of c we provide performance bounds for the pessimistic algorithm and if we truncate the denominator we provide performance bounds for the optimistic algorithm.

4.4.3 Comparison of Pairwise Algorithms

Tightness of Bounds

We can compare the performances of the optimistic and pessimistic algorithms by comparing the exponents in (4.26) and (4.57). Note that the algorithms will have the best approximation bounds if the exponents evaluate to -1. The performance of each algorithm is dependent on the corresponding notions of curvatures. For the optimistic algorithm, we wish that the 2-marginal curvatures are close to zero for each x_i and S_{i-1}. For the pessimistic algorithm we instead wish that the 2-cardinality curvature is close to zero. One downside that the pessimistic algorithm has is that each term of the sum has the 2-cardinality curvature multiplied by $(i-1)$. This means that when the cardinality constraint n is large, the $\min\{(i-1)\tau_2, 1\}$ will saturate, resulting in the later terms of the sum to evaluate to 0, hindering the guaranteed performance of the pessimistic algorithm.

Assumptions Required

It is important to note that the pessimistic algorithm requires that the function f possesses the property of supermodularity of conditioning. This is a strong assumption on the function and limits the number of applications where the pessimistic algorithm can be applied. The optimistic algorithm on the other hand can be applied to any submodular function. An advantage of the pessimistic algorithm, is that the performance bound is computable only using pairwise information. The performance bounds of the optimistic algorithm requires the ability to compute the objective function on sets of arbitrary size.

Empirical Results

As we show in our experimental results Section 4.6, for the particular problem we explore, the pessimistic algorithm tends to out perform the optimistic algorithm in terms of the approximation performance. Our experiments by no means show how the algorithms perform in every situation but highlights the potential performance of the two pairwise algorithms.

4.5 Time Complexity Analysis

A practical issue with the classical greedy strategy is that for large problems, it can be expensive to compute [28, 35]. As previously described, it is a common assumption in the literature that we have a *value oracle* for the submodular objective, that is computable in constant time. This leads to a time complexity that is linear in both the size of X and magnitude of n. As described in Subsection 3.2.2, the time complexity of the traditional greedy strategy, is $\mathcal{O}(|X| \cdot n \cdot T_{\text{eval}}(n))$, where T_{eval} describes the cost of evaluating f given a the size of the input set S.

The pairwise algorithms can exploit the evaluations of $f(A)$ on sets $A \subseteq X$ where $|A| \leq 2$, to achieve significant time complexity improvements. The cost of computing $f(x|y)$ for $x, y \in X$ can be considered constant for algorithms in Π_{pairwise}. This is because and algorithm $\pi \in \Pi_{\text{pairwise}}$ can only evaluate f on sets of size 1 and 2 which is independent of the size of the inputs $|X|$ and n.

Although, the optimistic and pessimistic algorithms are able to provide approximation guarantees based on properties of f, we see that they can present computational challenges. If they are naively implemented as in Algorithm 2 and 6, the cost of computing each $\bar{f}(x|X)$ or $\underline{f}(x|S)$ in the $\arg\max$ is $\mathcal{O}(|S|)$. In optimistic algorithm, for each $\bar{f}(x|S)$, we are computing the minimum $f(x|y)$ over $y \in S$. In the pessimistic algorithm, for each $\underline{f}(x|S)$, we are computing a sum of $f(x) - f(x|y)$ for $y \in S$. Essentially if we treat $\bar{f}(x|S)$ and $\underline{f}(x|S)$ as surrogates for $f(x|S)$, we are executing a greedy strategy with evaluation cost $T_{\text{eval}}(|S|) = |S|$, resulting in an overall time complexity of $\mathcal{O}(|X| \cdot n \cdot n)$ for both algorithms.

The naive implementation of the optimistic and pessimistic algorithms would only provide time complexity improvements if, the cost of executing f on sets S had time complexity larger than $\mathcal{O}(n)$. Thankfully, we are able to avoid the linear computation cost of $\bar{f}(x|S)$ and $\underline{f}(x|S)$ by utilizing the following two recursive definitions.

Let $S = \{x_1, \ldots, x_n\} \subseteq X$ be the set selected by a pairwise algorithm and $S_i = \{x_1, \ldots, x_i\}$. Both $\bar{f}(x|S)$ and $\underline{f}(x|S)$ have the recursive definitions. For the upper bound we have that for each $x \in X$,

$$\begin{aligned}
\bar{f}(x|S_i) &= \min_{x_j \in S_i} \{f(x|x_j)\} \\
&= \min\{\min_{x_j \in S_{i-1}} \{f(x|x_j)\}, f(x|x_i)\} \\
&= \min\{\bar{f}(x|S_{i-1}), f(x|x_i)\}. \qquad (4.58)
\end{aligned}$$

Similarly, for the lower bound have that for each $x \in X$,

$$\begin{aligned}\underline{f}(x|S_i) &= f(x) - \sum_{x_j \in S_i} f(x) - f(x|x_j) \\ &= f(x) - \sum_{x_j \in S_{i-1}} f(x) - f(x|x_j) - (f(x) - f(x|x_i)) \\ &= \underline{f}(x|S_{i-1}) - (f(x) - f(x|x_i)). \end{aligned} \tag{4.59}$$

We present a generalized pairwise algorithm, which efficiently implements both the optimistic and pessimistic algorithms.

Algorithm 7: Fast Pairwise Greedy Algorithm

Input: Base set X, submodular function f and cardinality constraint n
Result: Approximate solution $S \subseteq X$

1 $S_i \leftarrow \emptyset$ for all $i \in \{0, \dots, n\}$;
2 $f_{\text{est}}(x|S_0) \leftarrow f(x)$ for all $x \in X$;
3 **for** $i \leftarrow 1, \dots, n$ **do**
4 $\quad x_i \leftarrow \arg\max_{x \in X \setminus S_{i-1}} f_{\text{est}}(x|S_{i-1})$;
5 $\quad S_i \leftarrow S_{i-1} \cup \{x_i\}$;
6 $\quad f_{\text{est}}(x|S_i) \leftarrow \text{BOUND}(f_{\text{est}}(x|S_{i-1}), x, x_i)$ for all $x \in X \setminus S_i$;
7 **end**
8 $S \leftarrow S_n$

In Algorithm 7, $f_{\text{est}}(x|S_{i-1})$ represents the marginal estimate $\bar{f}(x|S_{i-1})$ or $\underline{f}(x|S_{i-1})$ depending if the optimistic or pessimistic algorithm is executed. BOUND is a subroutine which updates the current marginal estimate for x given the previous estimate and the element x_i, using (4.58) or (4.59). The updates for the upper and lower bounds take constant time. We can realize the sets of marginal estimates as a map where x is the key and the value is the marginal estimate of x given the current set S_{i-1}.

This means the arg max in Line 4, can be computed by iterating over $x \in X$ which takes $\mathcal{O}(|X|)$. Also, line 6 can be computed in $\mathcal{O}(|X|)$ time by iterating over x and running the BOUND subroutine which take constant time. Therefore, the execution time of each iteration of the For-Loop is $\mathcal{O}(|X|)$ and is executed n times which results in a time complexity of exactly $\mathcal{O}(|X| \cdot n)$.

This is significant because this algorithm essentially removes the cost of computing the function f, comparatively to the traditional greedy algorithm. Note, that this algorithm,

requires a map of size $|X|$ which maintains the values of the marginal estimates. This may not be feasible for some applications [23, 36, 57]. In these scenarios, other algorithms would have to be considered to solve the problem.

The efficiency of the pairwise algorithms introduces new trade-offs for practical applications of submodular maximization. We can trade-off approximation performance guarantees for time complexity improvements. This can be useful in scenarios where a user needs to repeatedly obtain approximate solutions to a submodular maximization problem quickly, but is not as sensitive to the quality of the solution. This trade-off can be critical in scenarios where there are real time computing constraints.

For example suppose that f was the set coverage function described in Remark 4.4.5, then given S then the cost of computing $f(S)$ is $\mathcal{O}(|S| * l)$ where l is the size of the largest set A_x, assuming that the sets are implemented using hash tables. Given large A_x sets and large S then $f(S)$ can become computationally expensive. For practical problems such as contaminant detection in large sensor networks, evaluating the objective can require parsing though gigabytes of data [27].

As we will show in Section 4.6, the execution time improvements in using the pairwise algorithms can be significant while simultaneously still providing relatively strong approximation performance. If the function has favourable curvature conditions, then the losses in the guaranteed performance from using the pairwise algorithms can be minimal.

4.6 Simulation Results

In this section, we benchmark the proposed algorithms in a simulated application of providing autonomous ride service in New York City utilizing electric vehicles. We focus on a coverage problem of selecting a set of charging locations for vehicles, for which they can best respond to customer demand after charging. From a historical data set provided by the NYC Taxi & Limousine Commission [38], we know that throughout the day the geographical distribution of customer demand is changing.

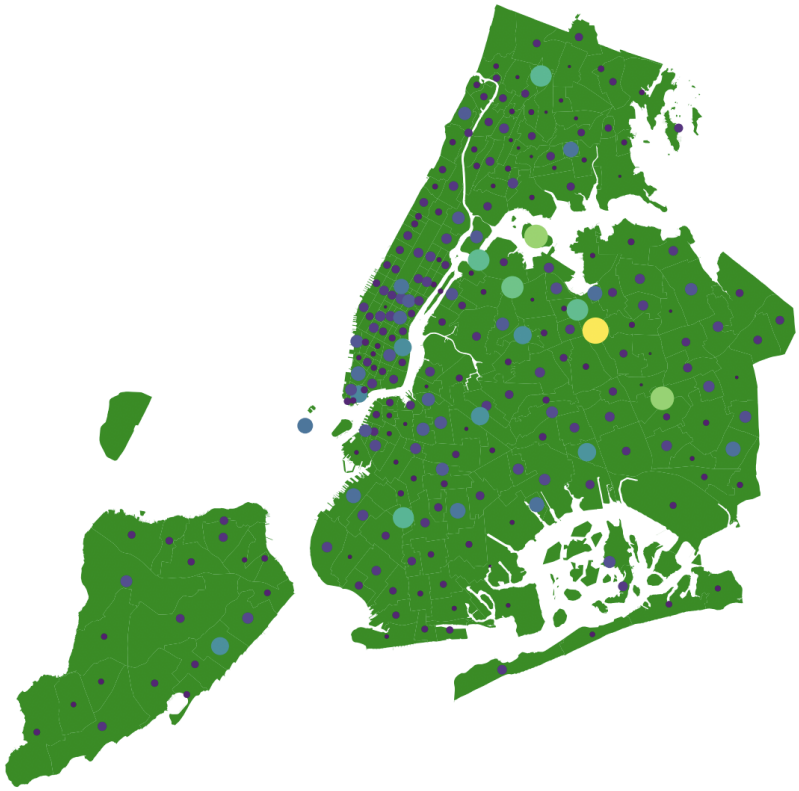

Figure 4.3: Geographical visualization of taxi customer data collected on January 1, 2020. There are 263 districts and size of the dots in each districts are proportional to the number of pick ups that occurred in the district.

New York City is split into 263 Taxi districts and we assume that the charging stations are located at the centroids of each of theses districts. We wish to select a subset of charging locations that maximize the expected customer demand that can be efficiently serviced from these locations. We say a customer can be efficiently serviced if it can be picked up with a delay of at most t minutes. Let $\mathcal{E}$ be the set of districts and let X be the set of stations. Let p_x^e be the probability that a vehicle deployed from station $x \in X$ can pick up a passenger in district $e \in \mathcal{E}$ in t-minutes. For simplicity we assume that the ride requests originate from the centriods of the districts. Finally, let v_e be the demand in district e, which is modeled as the estimated number of pick up requests in district e in a specified time interval, based on historical data.

Let $S \subseteq X$ be a set of stations. then the objective we want to maximize which we will call the *hidden* objective function f_h, is written as follows:

$$f_h(S) = \sum_{e \in \mathcal{E}} \left(\left(1 - \prod_{x \in S} (1 - p_x^e) \right) v_e \right). \tag{4.60}$$

The hidden objective function is more formally known as the probabilistic coverage function and was used for a related sensor coverage problem in [5].

Suppose we do not have access to the entire hidden objective function due to computational and/or modelling challenges, and thus the optimization must be solved using only pairwise information. Given the pairwise information constraint, we can compute the expected demand that can be serviced by a single station and a pair of stations as,

$$f(x) = \sum_{e \in \mathcal{E}} p_x^e v_e$$

and

$$f(x, y) = \sum_{e \in \mathcal{E}} (1 - (1 - p_x^e)(1 - p_y^e)) v_e.$$

We will model p_x^e using a Gaussian Kernel function

$$p_x^e = e^{-\frac{d(x,e)^2}{r^2}},$$

where $d : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}_+$ is a distance metric, and r is a tune-able parameter that dictates the range of distances where a vehicle could pick up a passenger in under t-minutes. For our experiments, we used the Euclidean distance metric for simplicity but the metric could be changed to better model the real system.

The objective function is a normalized, monotone and submodular function that exhibits supermodularity of conditioning [5], which allows us to apply all of our results. To decide which stations should be selected during different time intervals throughout the day, we estimate v_e for each $e \in \mathcal{E}$ from the historical data and then attempt to solve

$$S^* \in \underset{S \subseteq X, |S| \leq n}{\arg\max} \; f_h(S).$$

Under the pairwise information constraint this is exactly an instance of Problem 4.1.2 when $k = 2$.

We compare the optimistic and pessimistic algorithms' ability to maximize f_h while only given access to $f(x)$ and $f(x, y)$ for $x, y \in X$, to the full information greedy algorithm with full access to f_h. We compute the hidden objective value $f_h(S)$ for solutions S of the pairwise algorithms to compare against the solutions produced by the full information greedy algorithm.

4.6.1 Experimental Approximation Performance Results

To determine how the strategies of the optimistic and pessimistic strategies perform we used historical data for ride services in New York City. The data set included the pick-up times and locations for all the "For Hire Vehicle" rides in the month of January 2020 [38]. We tested the performance of algorithms on varying distributions, we split up the data by pick-up time. We made twelve sections, each corresponding to a unique two-hour window of the day. For each of the subsets, we estimate values v_e for each district by taking the average number of rides in each time interval over all the days in the data set. We executed the three algorithms on each of the twelve sets to test performance, which is summarized in Figure 4.4.

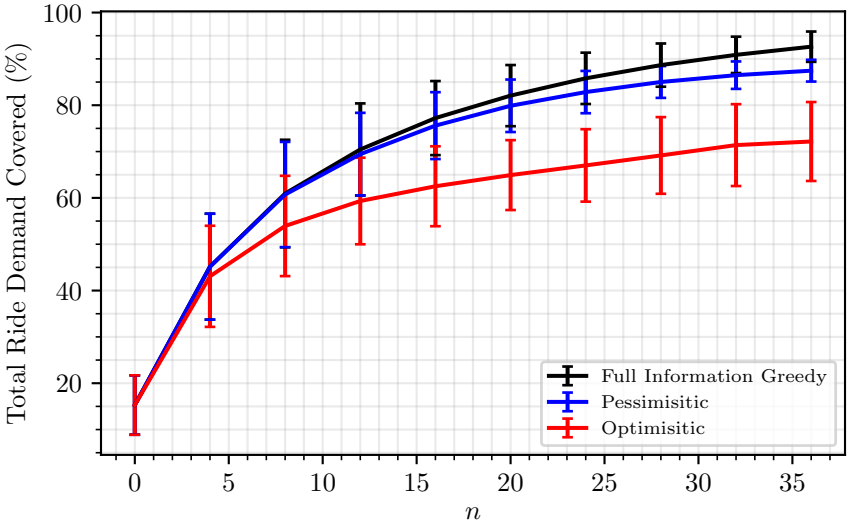

Figure 4.4: Experimental results comparing the performance of the three algorithms. For each algorithm, the average percentage of total demand covered by the selected stations was plotted against the number of stations in the set. The percentage covered was averaged over the 12 experiments. The error bars enclose one standard deviation above and below the average.

Figure 4.4 shows the performance for different numbers of charging stations selected. We see that all three algorithms have similar performance for low values of n, but then begin to diverge after 5 stations are selected. The total ride demand covered is computed by taking the demand covered by a set of station selected by an algorithm $f_h(S)$ and dividing by $f_h(X)$ during a two hour time interval. This was done to compare the performance of

each time interval as the ride demand changes over the day. The pessimistic algorithm's performance is significantly better than the optimistic algorithm's. The pessimistic algorithm yielded a value no worst than 90% of the full information greedy algorithm's value and the optimistic algorithm yielded a value no worst than 67% across all trials.

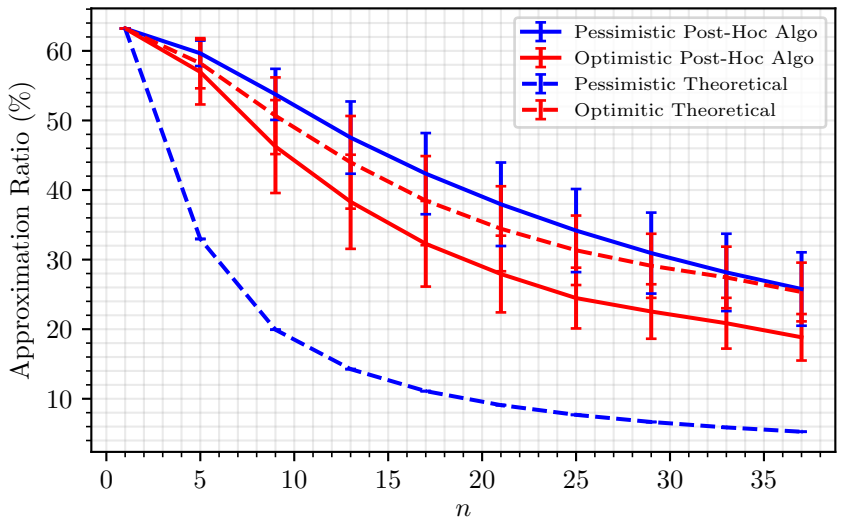

Figure 4.5: Experimental results comparing the average worst case lower bounds produced by each algorithm as a function of the number of stations selected n. The bounds with solid lines were computed using Algorithm 5. The bound for marked "Pessimistic Theoretical" was computed using Corollary 4.4.8 and line marked "Optimistic Theoretical" was computed using Corollary 4.3.4.The error bars represent one standard devation above and below the averages.

Figure 4.5 compares the worst-case lower bounds of the optimistic and pessimistic algorithms computed by Algorithm 5 on the same trials as used in Figure 4.4 as well as the worst case performance bounds produced by Corollaries 4.3.4 and 4.4.8. As the number of stations increases, the lower bounds on performance degrades in all cases. For the pessimistic algorithm, this is due to the fact the lower bounds on the marginals also exhibit diminishing returns and continually selecting elements maximizing the lower bound drives the denominator of (4.48) down. This causes high values of the estimated $\alpha_1, \ldots, \alpha_n$ and decreasing approximation bounds. The optimistic algorithm does not actively minimize the estimated approximation factors, which are reflected in both the percentage of ride demand covered and computed approximation bounds.

As observed in Figure 4.4, both pairwise algorithms are performing similarly to the full information greedy strategy even through the lower bounds in Figure 4.5 suggest otherwise. We also see that the theoretical performance bound for the optimistic algorithm is relatively close to the bound produced by Algorithm 5 for the pessimistic algorithm. The pessimistic algorithm's theoretical bound is much lower than the rest of the bounds, due to the fact that the average τ_2 for each trial was near 1 with a value of 0.89. This highlights the benefits of utilizing Algorithm 5 when computing performance bounds for the pessimistic algorithm.

This experiment reveals that the bound produced by Algorithm 5 becomes less accurate as n increases. This is due to the fact that the approximation factors measure the multiplicative difference between the true greedy choices and the pairwise algorithms' choices. The true marginals for elements selected near the end of execution tend to be smaller. Thus, the difference in the overall objective values could be small, but the multiplicative difference could be large which is reflected in a lower bound on performance.

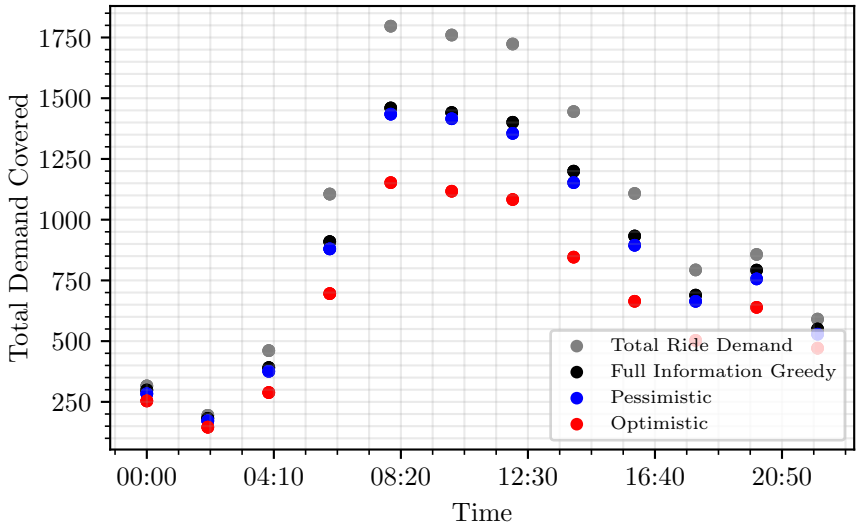

Figure 4.6: Experimental results comparing the demand covered by each algorithm for each two hour time interval of the day with $n = 25$. The grey dots represent to maximum possible demand that could be covered during that time interval.

Figure 4.6 looks at the performance of the algorithms over different subsets of historical data. We plotted the objective value for 25 stations using each of the algorithms for

each of the twelve subsets of data. We also plotted the maximum possible value of the objective function each of the algorithms could possibly achieve in the time frame. From Figure 4.6, we see that for each time frame, the pessimistic algorithm is essentially as effective as the full information greedy algorithm, but the optimistic greedy algorithm is less effective. The performance of both the optimistic and pessimistic algorithms seem to be robust to the changing of the demand distribution through a 24-hour period. For time intervals where there is low total demand, the pairwise algorithms are as effective as the full information greedy strategy. The gap in performance between the optimistic and pessimistic algorithms is more apparent when there is higher total ride demand. On the other hand, the performance of the optimistic algorithm is consistent.

Remark 4.6.1. We can reason why the pessimistic algorithm is out performing the optimistic strategy for this problem by comparing the estimates of the marginals they are maximizing. We can rewrite the optimistic strategy's objective as,

$$\bar{f}(x|S) = \min_{x_j \in S} f(x|x_j) = f(x) - \max_{x_j \in S}\{f(x) - f(x|x_j)\}. \tag{4.61}$$

If we compare this to the objective maximized in the pessimistic strategy,

$$\underline{f}(x|S) = f(x) - \sum_{x_j \in S}(f(x) - f(x|x_j)). \tag{4.62}$$

We see that the upper bound and lower bound on the marginal return are related quantities. The $f(x) - f(x|x_j)$ term in both (4.61) and (4.62) represents the maximum decrease of $f(x|S)$ from $f(x)$ contributed by x_j. By estimating the marginal returns using $\bar{f}(x|S)$ we are assuming a single element in S impacts the value of $f(x|S)$. By estimating the marginal return using $\underline{f}(x|S)$, we are assuming each element in S independently impacts the value of $f(x|S)$.

In this scenario, because the elements are spread out geographically, the marginal return of adding a station, is affected by all previously selected stations that surround the new station being added. It is unlikely that only a single previously selected station impacts the marginal return of the newly added station. Therefore, the estimates $\underline{f}(x|S)$ are likely more accurate than $\bar{f}(x|S)$, resulting in better performance of the pessimistic algorithm.

4.6.2 Experimental Time Complexity Results

Using the same data used for the performance experiments, we measured the time efficiency of the three algorithms. We measured the execution time of the algorithms for each of the 12

data subsets and plotted the average execution time in terms of n. Each of the experiments was computed using Python 3.7 on a 2017 Macbook Pro with a 3.1 GHz Dual-Core Intel i5 and 8 GB 2133 MHz LPDDR3 RAM.

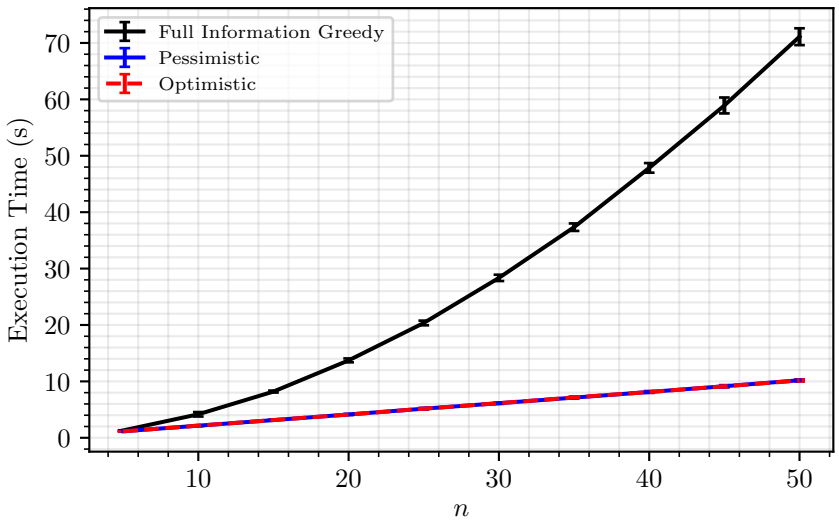

Figure 4.7: Experimental result showing the execution time for each algorithm as the number of stations selected increases. For each n, the algorithms execution times were recorded for each of the 12 subsets of data and then average over 5 trials. The error bars represent one standard deviation around the averages.

Figure 4.7 summarizes the results from the execution time experiment. For each trial, the size of $|X|$ was the same. The relationship between the execution times of the pairwise algorithms and the value of n is linear. This relationship is as expected given our time complexity analysis in Section 4.5. Using the pairwise greedy strategies and the implementation in Algorithm 7, the time complexity is reduced from quadratic to linear in terms of n. The pessimistic and optimistic algorithms share similar execution times which resulted in the two line overlapping.

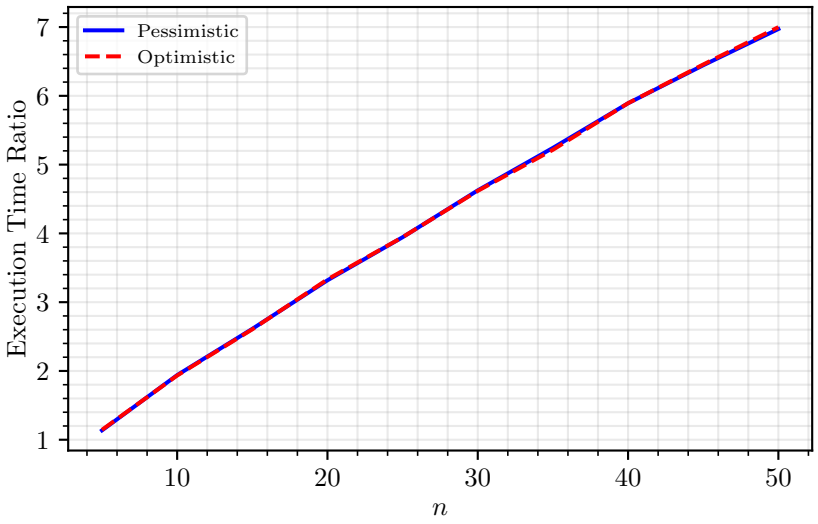

Figure 4.8: Plot of the ratios of execution times of the greedy strategy to the execution times of the pairwise algorithms as the number of stations selected increases. For each n the average execution time of the greedy algorithm was divided the average execution of each algorithm.

In Figure 4.8, we see that for both pairwise algorithms the ratio of the execution time of the full information greedy algorithm and the pairwise algorithms is essentially linear. This verifies that the pairwise algorithms result in a reduction in time complexity by a factor of n for this particular objective function.

4.7 Summary

In this chapter, we introduced the submodular maximization problem with limited function access and established inapproximability results for the general problem. We then introduced the optimistic strategy, which provides a constant factor approximation for general problem given the value of the total k-marginal curvature. With an addition assumption of submodularity of conditioning on the objective, we provide a method to measure the performance of an arbitrary greedy strategy using only pairwise information. We then provide an additional pairwise strategy called the pessimistic algorithm. We present performance bounds for the pessimistic algorithm that are in terms of the k-cardinality curvature. We

then show that both optimistic and pessimistic algorithms can be computed efficiently. Finally, we present empirical evidence showing the effectiveness of the pairwise algorithm.

Chapter 5

Linear Programming for Distributed Submodular Maximization

In this chapter we will discuss work related to representations of submodular functions and how these representations can be used in the context of distributed submodular maximization. We begin by establishing a connection between submodular functions and feasible regions of linear constraints. We then formulate a general linear program that can be used to find these worst-case function examples. We are able to directly apply the linear programming approach to the distributed submodular maximization problem.

5.1 Linear Programming Representation

We start this chapter by introducing a method to realize submodular functions as vectors in the feasible region of a set of linear constraints. This bears resemblance with the techniques used in the original formulation of the submodular maximization problem in the classical work of [37]. To wit, let X be a base set of elements. We represent a set function defined on X as a 2^N dimensional real-valued vector, where $N = |X|$. Let $v \in \mathbb{R}^{2^N}$ be a vector (or lookup table) where each component of v gives the function value for a corresponding subset $S \subseteq X$, we denote this component by v_S. The indexing of subsets for the vectors is fixed i.e., for any two vectors $v, \hat{v} \in \mathbb{R}^{2^N}$, the values of v_S and $\hat{v}_S$ are located at the same index for the two vectors. Given $v \in \mathbb{R}^{2^N}$, we define a set function $f : 2^X \to \mathbb{R}$ by

$$f(S) = v_S.$$

We can enforce properties on the function f by imposing constraints on v. We can realize v as

$$v = \begin{bmatrix} f(\emptyset) \\ f(x_1) \\ \vdots \\ f(x_n) \\ f(x_1, x_2) \\ \vdots \\ f(x_{N-1}, x_N) \\ \vdots \\ f(x_1, \ldots, x_N) \end{bmatrix} \in \mathbb{R}^{2^N}. \tag{5.1}$$

We now define a matrix $A_{\text{submodular}}$, which if $A_{\text{submodular}} v \geq 0$ then, the function f defined by v is submodular. For a function f to be submodular we require that

$$f(A) + f(B) \geq f(A \cap B) + f(A \cup B), \tag{5.2}$$

for all $A, B \subseteq X$. Let us rewrite (5.2), in terms of the components of v as

$$v_A + v_B - v_{A \cap B} - v_{A \cup B} \geq 0. \tag{5.3}$$

Let $A_{\text{submodular}} \in \mathbb{R}^{2^N(2^N-1) \times 2^N}$ where each row corresponds to the constraint in (5.3) for a pair of subsets A and B.

We now define another matrix to ensure the function represented by v is also monotone. For f to be monotone we need to ensure that $f(x|A) \geq 0$, for all $x \in X$ and $A \subseteq X$. If we assume that $A_{\text{submodular}} v \geq 0$ holds, then we can enforce monotonicity on v by adding N more constraints.

In particular, we ensure that $f(x|X \backslash \{x\}) \geq 0$ for all $x \in X$. If f is submodular then $f(x|A) \geq f(x|X \backslash \{x\})$ for $A \subseteq X \backslash \{x\}$, then enforcing $f(x|X \backslash \{x\}) \geq 0$ implies $f(x|A) \geq 0$. Written in terms of the components v, we impose the condition

$$v_X - v_{X \backslash \{x\}} \geq 0. \tag{5.4}$$

Let $A_{\text{monotone}} \in \mathbb{R}^{N \times 2^N}$, where each row encodes the monotonicity constraint imposed by x in (5.4) for all $x \in X$. If both

$$A_{\text{submodular}} v \geq 0 \quad \text{and} \quad A_{\text{monotone}} v \geq 0,$$

then the function represented by v will be both monotone and submodular. Finally to ensure that f is normalized we simply impose an equality constraint $v_\emptyset = 0$.

A key fact about these constraints is that they are all linear inequalities in terms of the components of v. We formulate a linear program in order to search for monotone, normalized and submodular functions with certain properties:

$$\begin{aligned} &\max_{v \in \mathbb{R}^{2^N}} c^T v, \\ \text{s.t.} &\begin{bmatrix} A_{\text{submodular}} \\ A_{\text{monotone}} \end{bmatrix} v \geq 0, \\ &v_\emptyset = 0, \quad \text{and} \quad Mv \geq b, \end{aligned}$$

where $c \in \mathbb{R}^{2^N}$ is a general cost vector, $M \in \mathbb{R}^{l \times 2^N}$ and $b \in \mathbb{R}^l$. Here, M and b are general constraints that can be used to enforce additional properties on the submodular function produced by the optimal solution v. The cost vector defines which values of the submodular function should be maximized. In Section 5.3, we specify c, M and b to provide performance guarantees for the adapted greedy strategy for distributed submodular maximization.

5.2 Distributed Submodular Maximization Problem

5.2.1 Problem Statement

We now introduce the distributed submodular maximization problem [12] and how we can apply our linear programming approach. Suppose we are given n agents $V = \{1, \ldots, n\}$. Each agent i has access to the an action set X_i and must choose one action $x_i \in X_i$. The agents follow a greedy strategy and we want study the impact of information on their performance. We consider the scenario where the agents select their actions sequentially. Each agent $i \in V$ has access to a subset of actions chosen by agents $\{1, \ldots, i-1\}$. We encode this information in a directed acyclic graph (DAG) $G = (V, E)$, where the agent indices form a topological order. There is an edge $(i, j) \in E$ if agent j has access to the action of agent i. We refer to this graph as the agents' communication graph.

The in-neighbor set of agent i in G is defined as

$$\mathcal{N}(i, G) = \{j \in V \mid (j, i) \in E\}.$$

The information available to this agent is given by

$$X_{\text{in}}(i, G) = \{x_j \mid j \in \mathcal{N}(i, G)\}.$$

We assume the agents select their actions by greedily maximizing their own marginal return given the information available to them, i.e.,

$$x_i \in \arg\max_{x \in X_i} f(x|X_{\text{in}}(i, G)). \tag{5.5}$$

A greedy solution $S_G = \{x_1, \ldots, x_n\}$ is one for which each x_i satisfies (5.5). We then let $\mathbb{S}_G$ be the set of all greedy solutions.

We study the same problem described in [12, 15, 5] and is defined as follows:

$$\begin{aligned} &\max_{S \subseteq X, |S| \leq n} f(S) \\ &\text{s.t. } |S \cap X_i| \leq 1 \\ &\text{for } i \in \{1, \ldots, n\}, \end{aligned} \tag{5.6}$$

where $X = \cup_{i \in n} X_i$ and each X_i is disjoint. This is formally known as maximizing a submodular function over the partition matroid. This is the exactly Problem 3.7, when $k = 1$.

Suppose we have a normalized, monotone and submodular function f, a group of agents, a DAG G, and the strategy given by (5.5). We want to study the worst-case sub-optimality of the greedy strategy (5.5) for the problem defined in (5.6). Note that this will depend on the information structure G, and thus we want to study how the performance depends on G.

As described in [12], if the sets X_i are not disjoint, then any derived bounds for disjoint sets still hold for non-disjoint action sets. Therefore, we can apply any previously known performance bounds for greedy strategy (5.5) from [12, 15] to the special case of the problem where each agent shares the same action set $X_i = X$. More formally, we are interested in the following problem:

$$\max_{S \subseteq X, |S| \leq n} f(S). \tag{5.7}$$

This is exactly the Problem 3.5. When the agents share action sets we want to study the worst-case sub-optimally of the solution produced by greedy strategy (5.5) for the problem defined in (5.7). As described in the introduction, the centralized greedy strategy provides a tighter approximation guarantee, when maximizing over the uniform matroid compared to the partition matroid. We are interested in understanding if the approximation bounds can be improved for the distributed problem in the same way.

5.2.2 Previous Performance Guarantees

We define the competitive ratio for a normalized monotone submodular function f and a DAG $G = (V, E)$ as

$$\gamma(f, X, G) = \min_{S_G \in \mathbb{S}_G} \frac{f(S_G)}{f(S^*)},$$

where $\mathbb{S}_G$ is the set of all greedy solutions given by (5.5), and thus $\min_{S_G \in \mathbb{S}_G} f(S_G)$ is worst possible greedy solution.

Let us denote the worst-case competitive ratio for a given graph G by

$$\underline{\gamma}(X, G) = \min_{f} \gamma(f, X, G).$$

Both [12] and [15] provide performance bounds in terms of properties of the graph G.

The tightest known bound for this problem is provided in [15] and is in terms of the *fractional independence number* of the graph $G = (V, E)$ denoted by $\alpha^*(G)$. The fractional independence number is defined as

$$\begin{aligned} \alpha^*(G) = & \max_{x \in \mathbb{R}^n} \sum_{v \in V} x_v, \\ & \text{s.t.} \\ & x_v \geq 0 \ \forall \ v \in V, \\ & \sum_{v \in C} x_v \leq 1 \ \forall \text{ Cliques } C \subseteq V. \end{aligned} \tag{5.8}$$

The independence number is the solution to an integer program, and the fractional independence is the solution obtained by relaxing the integer constraints of the program, see [15].

Theorem 5.2.1 (Theorem 1 of [15])**.** *Let $\underline{\gamma}(X, G)$ be the worst case competitive ratio of (5.6) for agents following (5.5). We have,*

$$\underline{\gamma}(X, G) \geq \frac{1}{\alpha^*(G) + 1}. \tag{5.9}$$

To best of our knowledge, this is the tightest known bound for problem (5.6) but also the tightest for the special case when each $X_i = X$, and comparing to problem (5.7). We seek to provide a tighter bound for the special case when using the greedy strategy to approximate the solution to problem (5.7).

5.2.3 Linear Programming Approach

We will now present how to incorporate the adapted greedy strategy in (5.5) into our linear programming model. Suppose we are given a DAG G with n nodes and a set $B = \{x_1, \ldots, x_n\}$ that satisfies the partition matroid constraint. We can add constraints to the linear program such that the feasible set contains all the submodular functions for which B is a greedy solution from (5.5) i.e, $B \in \mathbb{S}_G$.

Given a fixed set $B = \{x_1, \ldots, x_n\} \subseteq X$, let

$$X_{\text{in}}(i, G) = \{x_j \in B \mid j \in \mathcal{N}(i, G)\},$$

for a function $f : 2^X \to \mathbb{R}$ we need the following inequalities to hold

$$f(x_i | X_{\text{in}}(i, G)) \geq f(x | X_{\text{in}}(i, G)) \text{ for all } x \in X_i,$$

for each $x_i \in B$. We can rewrite these in terms of the components of v by

$$v_{\{x_i\} \cup X_{\text{in}}(i,G)} - v_{\{x\} \cup X_{\text{in}}(i,G)} \geq 0 \text{ for all } x \in X_i.$$

Let us define a $A_{\text{greedy},G} \in \mathbb{R}^{(|X_1|+\cdots+|X_n|) \times 2^N}$, which encodes each of the greedy constraints for all $x_i \in B$. If $A_{\text{greedy},G} v \geq 0$ then the feasible region of v describes all the set functions where B is a greedy solution.

5.3 Worst-Case Studies using Linear Programs

We begin this section, with a formulation of a linear program where the solution is a function with minimum competitive ratio, out of all the submodular functions where S_G is selected by agents following (5.5). Given the sets $A, B \subseteq X$ that each satisfy the partition matroid constraint, the following program produces a submodular function where A is an optimal solution to (5.6), and B is a greedy solution produced by algorithm (5.5) with minimum competitive ratio. The constant C fixes the value of $f(B)$ for the resulting function. We define the following program:

$$\begin{aligned} &\max_{v \in \mathbb{R}^{2^N}} \quad v_A, \\ \text{s.t.} \quad &\begin{bmatrix} A_{\text{submodular}} \\ A_{\text{monotone}} \\ A_{\text{greedy},G} \end{bmatrix} v \geq 0, \\ &v_\emptyset = 0 \quad \text{and} \quad v_B = C. \end{aligned} \tag{5.10}$$

Let f be the function produced by the optimal solution v^*. The program finds the maximum value that the set A can take on given $f(B) = C$ and $B \in \mathbb{S}_G$. Since $f(B)$ is fixed, maximizing $f(A)$ will produce the largest value for $f(A)$, and therefore will produce the function with minimum competitive ratio.

Using an off-the-shelf black box optimizer, linear program (5.10) can be solved numerically to produce the worst-case function examples. However, the drawback is that the constraints in the linear programs scale exponentially in N. The encoding of the general linear program requires $\mathcal{O}(2^N(2^N - 1))$ space using sparse matrix representations. For small enough sets X, black-box optimizer are able to solve the linear programs in a tolerable amount of time.

5.3.1 Main Results

Using the linear programming approach we show that by removing edges from the communication graph, we degrade worst-case performance. The result is suggested by [12] and [15] but is not explicitly proven.

Theorem 5.3.1. *Let $f : 2^X \to \mathbb{R}$ be a normalized, monotone and submodular function and $G = (V, E)$ be a DAG. Let $\bar{G} = (V, E \backslash \{e\})$ where $e \in E$. Then, there exists a normalized, monotone, and submodular function $\bar{f} : 2^X \to \mathbb{R}$ with competitive ratio such that,*

$$\gamma(\bar{f}, X, \bar{G}) \leq \gamma(f, X, G).$$

This theorem essentially states that if we remove edges from G, then the approximation guarantees for the greedy strategy degrades. We formalized this in the following corollary.

Corollary 5.3.2. *Consider two DAGs $G = (V, E)$ and $\bar{G} = (V, \bar{E})$ with $\bar{E} \subseteq E$, Then*

$$\underline{\gamma}(X, G) \geq \underline{\gamma}(X, \bar{G}).$$

We will now present the proof of Theorem 5.3.1.

Proof of Theorem 5.3.1. We begin by introducing two linear programs which will compute functions with the minimum competitive ratio given G and $\bar{G}$.

We define our first linear program, referred to as *linear program 1*, as an instance of (5.10), with a fixed set $A = \{x_1^a, \ldots, x_n^a\} \subseteq X$ and $B = \{x_1, \ldots, x_n\} \subseteq X$ such that $|A \cap X_i| = 1$, and $|B \cap X_i| = 1$, with greedy constraints defined using G. Let S_G be a

greedy solution that achieves value $\min_{S\in\mathbb{S}_{\mathbb{G}}} f(S)$ and let $C = f(S_G)$. Let $\hat{f}$ be the function produced by the optimal solution $\hat{v}^* \in \mathbb{R}^{2^N}$ of linear program 1. The second linear program, *linear program 2*, is identical to linear program 1 but we replace the greedy constraints with the greedy constraints defined by $\bar{G}$. Let $\bar{f}$ be the function produced by the optimal solution $\bar{v}^* \in \mathbb{R}^{2^N}$ of linear program 2. The set B is a greedy solution for both $\hat{f}$ and $\bar{f}$, therefore we have that $\hat{f}(B) = \bar{f}(B) = f(S_G)$.

We begin by noting that, since $\bar{G}$ has one less edge than G, only one agent has less information when using $\bar{G}$ instead of G. Let $(j,k) \in E$ be the edge that is removed from G to produce $\bar{G}$. Therefore, we have that $X_{\text{in}}(i,G) = X_{\text{in}}(i,\bar{G})$ for $i \in \{1,\dots,n\}\backslash\{k\}$, and for agent k we have $X_{\text{in}}(k,G) = X_{\text{in}}(k,\bar{G}) \cup \{x_j\}$. Note that $X_{\text{in}}(i,G) \subseteq B$ and $X_{\text{in}}(i,\bar{G}) \subseteq B$ for each i. Let $\hat{v} \in 2^N$ be a vector in the feasible region of the constraints of linear program 1 and $g : 2^X \to \mathbb{R}$ be the submodular function corresponding to $\hat{v}$. Next, note that the constraints in both linear programs are identical except for the constraints imposed by the information structure, $A_{\text{greedy},G}$ and $A_{\text{greedy},\bar{G}}$. Let $\bar{v} \in \mathbb{R}^{2^N}$ be in the feasible region of linear program 2. Next we show that the constraints on the value of $\hat{v}_A$ are tighter than $\bar{v}_A$ for both the partition matriod and uniform matroid scenarios. The tightest constraints that we can impose on maximum value of $g(A)$ that involve the greedy constraints are described as follows:

$$\begin{aligned} g(A) &\leq g(B) + \sum_{x_i^a \in A} g(x_i^a | \{x_{i-1}^a, \dots, x_1^a\} \cup X_{\text{in}}(i,G)) \\ &\leq g(B) + \sum_{x_i^a \in A} g(x_i^a | X_{\text{in}}(i,G)), \end{aligned}$$

where the first inequality holds by monotonicity and the second by submodularity. As a result,

$$\begin{aligned} g(A) \leq & g(B) + \sum_{x_i \in B} g(x_i | X_{\text{in}}(i,G)), \\ \leq & g(B) + \sum_{x_i \in B\backslash\{x_k\}} g(x_i | X_{\text{in}}(i,G)) \\ & + g(x_k | X_{\text{in}}(k,\bar{G}) \cup \{x_j\}), \end{aligned} \tag{5.11}$$

where we have used the greedy constraints in the first inequality. Next, note that by submodularity

$$g(x_k | X_{\text{in}}(k,\bar{G}) \cup \{x_j\}) \leq g(x_k | X_{\text{in}}(i,\bar{G})),$$

and hence

$$g(A) \leq g(B) + \sum_{x_i \in B} g(x_i | X_{\text{in}}(i, \bar{G})). \tag{5.12}$$

Here, the tightest constraints imposed by G in (5.11) are upper bounded by the ones imposed by $\bar{G}$ in (5.12). However, for the case when $X_i = X$, the above constraints are not the tightest possible constraints on the value of $g(A)$; hence, for the uniform matroid scenario, we provide a different set of inequalities. Let $B_i = \{x_1, \ldots, x_i\}$ for all $i \in \{1, \ldots, n\}$ the tightest constraints given G for each B_i are as follows:

$$\begin{aligned} g(A) &\leq g(B_i) + \sum_{x_i^a \in A} g(x_i^a | \{x_{i-1}^a, \ldots, x_1^a\} \cup B_i) \\ &\leq g(B_i) + \sum_{x_i^a \in A} g(x_i^a | X_{\text{in}}(i, G)) \\ &\leq g(B_i) + n \cdot g(x_i | X_{\text{in}}(i, G)), \end{aligned} \tag{5.13}$$

where (5.13), holds since $g(x_i | X_{\text{in}}(i, G)) \geq g(x | X_{\text{in}}(i, G))$ for all $x \in X$. Since G and $\bar{G}$ differ by a single edge, a procedure similar to the one done in (5.12) yields that

$$g(A) \leq g(B_k) + n \cdot g(x_k | X_{\text{in}}(k, \bar{G})).$$

For both the partition and uniform matroid scenarios, each term in the above line of inequalities can be represented as constraints on the decision vectors $\hat{v}$ and $\bar{v}$. By the above inequalities, the constraints on $\hat{v}_A$ of linear program 1 are upper bounded by the constraints on $\bar{v}_A$ of linear program 2. By the properties of linear programming, the maximum values $\hat{v}_A^* \leq \bar{v}_A^*$. Therefore we have that $f(S^*) \leq \hat{f}(A)$ by construction of linear program 1, and $\hat{f}(A) \leq \bar{f}(A)$ by our constraints argument, yielding

$$\frac{f(S_G)}{f(S^*)} \geq \frac{\hat{f}(B)}{\hat{f}(A)} \geq \frac{\bar{f}(B)}{\bar{f}(A)}.$$

Since, $\frac{\hat{f}(B)}{\hat{f}(A)}$ and $\frac{\bar{f}(B)}{\bar{f}(A)}$ are the minimum competitive ratios given G and $\bar{G}$, we have

$$\gamma(f, X, G) \geq \gamma(\bar{f}, X, \bar{G}).$$

□

Using this theorem we can immediately prove Corollary 5.3.2.

Proof of Corollary 5.3.2. First, we let $G_1 = G$ be given by $G_1 = (V, E_1)$. By definition we have,

$$\underline{\gamma}(X, G_1) = \min_f \gamma(f, X, G_1).$$

Let $\bar{E}_1 = E_1 \backslash \{e\}$ and $\bar{G}_1 = (V, \bar{E}_1)$. Let $\hat{f} = \arg\min_f \gamma(f, X, G_1)$. By Theorem 5.3.1, there exist a normalized monotone submodular function $\bar{f}$ such that $\gamma(\hat{f}, X, G_1) \geq \gamma(\bar{f}, X, \bar{G}_1)$. By definition of $\underline{\gamma}(X, \bar{G}_1)$, we have that $\gamma(\bar{f}, X, \bar{G}_1) \geq \underline{\gamma}(X, \bar{G}_1)$. Combining the two inequalities we arrive at

$$\underline{\gamma}(X, G_1) \geq \gamma(\bar{f}, X, \bar{G}_1) \geq \underline{\gamma}(X, \bar{G}_1). \tag{5.14}$$

Let now $\bar{E}_l = E_1 \backslash \{e_1, \ldots, e_l\}$ and $\bar{G}_l = (V, \bar{E}_l)$. Let $\{\bar{G}_i\}_{i=1}^{|l|}$ be a sequence of graphs, where each $\bar{G}_i = (V, E \backslash \{e_1, \ldots, e_i\}))$. Here $\bar{G}_i$ has exactly one less edge than $\bar{G}_{i-1}$. Therefore, we can iteratively apply (5.14) on the sequence of graphs to get,

$$\underline{\gamma}(X, G_1) \geq \underline{\gamma}(X, \bar{G}_1) \geq \cdots \geq \underline{\gamma}(X, \bar{G}_l), \tag{5.15}$$

yielding our result. □

The results stated so far, hold for the special case when $X_i = X$ for each $i \in \{1, \ldots, n\}$, leading to a result that we present next. For this, let $\omega(G)$ be the clique number of the graph G, that is the size of the largest clique.

Theorem 5.3.3. *Let $\underline{\gamma}(X, G)$ be the worst case competitive ratio of (5.7) for agents following (5.5), with $X_i = X$ for all $i \in \{1, \ldots, n\}$, then we have,*

$$\underline{\gamma}(X, G) \geq \left(1 - \left(1 - \tfrac{1}{n}\right)^{\omega(G)}\right) \geq \left(1 - e^{-\frac{\omega(G)}{n}}\right).$$

Before we begin the proof, we need the following classical result from [37], taken here from [25]. Let the classical greedy strategy be defined by

$$S_i = S_{i-1} \cup \{\arg\max_{x \in X} f(x | S_{i-1})\}. \tag{5.16}$$

Theorem 5.3.4 (Theorem 1.5 of [25])**.** *Fix a non-negative monotone submodular function $f : 2^X \to \mathbb{R}_+$ and let $\{S_i\}_{i \geq 0}$ be the greedily selected sets defined by (5.16). Then for all positive integers n and l, we have*

$$f(S_l) \geq \left(1 - \left(1 - \tfrac{1}{n}\right)^l\right) f(S^*) \geq (1 - e^{-l/n}) f(S^*), \tag{5.17}$$

where $f(S^*)$ is the maximum value of Problem (5.7).

We now present the proof of Theorem 5.3.3.

Proof of Theorem 5.3.3. First, we show that $f(S_G) \geq (1 - e^{-\frac{p}{n}}) f(S^*)$, where G is a graph that contains a single clique of size p with no additional edges and S_G is an arbitrary set in $\mathbb{S}_G$. We combine this result with Corollary 5.3.2 to yield our result.

Let $G = (V, E)$ be a communication graph, such that there is a subset vertices that form a clique V_{clique}. Suppose that E contained the minimum number of edges such that V_{clique} forms a clique. Let p be the size of the clique. Given the greedy solution S_G, let $S_{\text{clique}} \subseteq S_G$ be the actions selected by the agents in V_{clique}. We will now show by induction that there exists a set $S_{\text{greedy},p}$ produced by (5.16) such that $S_{\text{clique}} = S_{\text{greedy},p}$. Let agent k be the first agent in the clique, $S_{\text{clique}} = \{x_k, \ldots, x_{k+p}\}$, and $S_{\text{greedy},p} = \{x_1^g, \ldots, x_p^g\}$ be a set that could be selected by the first p iterations of the traditional greedy strategy (5.16). Note that $E = \{(i, j) : i, j \in V_{\text{clique}} \text{ and } i < j\}$ and hence, $X_{\text{in}}(k+i, G) = \{x_k, \ldots, x_{k+i-1}\}$, for $i \geq 0$ and $k + i \in V_{\text{clique}}$.

Base case ($\{x_k\} = S_{\text{greedy},1}$) : We have that $S_{\text{greedy},1} = \{x_1^g\}$, by the definition we have that

$$x_k \in \arg\max_{x \in X} f(x | X_{\text{in}}(k, G)).$$

Since k is the first agent in the clique, we have that $X_{\text{in}}(k, G) = \emptyset$. Therefore, this is exactly the first iteration of the classical greedy strategy, i.e., we can have $x_k = x_1^g$.

Inductive step: Let us assume that $\{x_k, \ldots, x_{k+i-1}\} = S_{\text{greedy},i-1}$. We have $X_{\text{in}}(i, G) = \{x_k, \ldots, x_{k+i-1}\}$, and so

$$x_{k+i} \in \arg\max_{x \in X} f(x | S_{\text{greedy},i-1}).$$

Which implies we can let $x_{k+i} = x_i^g$. As a result we have,

$$\begin{aligned} \{x_k, \ldots, x_{k+i}\} &= S_{\text{greedy},i-1} \cup \{x_{k+i}\} \\ &= S_{\text{greedy}i-1} \cup \{x_i^g\} = S_{\text{greedy},i}. \end{aligned}$$

Proving the inductive hypothesis.

We now have that $S_{\text{clique}} = \{x_k, \ldots, x_{k+p}\} = S_{\text{greedy},p}$, we can apply Theorem 5.3.4. We hence conclude that

$$f(S_{\text{clique}}) = f(S_{\text{greedy},p}) \geq \left(1 - \left(1 - \tfrac{1}{n}\right)^p\right) f(S^*).$$

By monotonicity, we have $f(S_G) \geq f(S_{\text{clique}})$. We now arrive at

$$\frac{f(S_G)}{f(S^*)} \geq \left(1 - \left(1 - \tfrac{1}{n}\right)^p\right).$$

Since this holds for any normalized, monotone and submodular function and for any greedy solution $S_G \in \mathbb{S}_G$, we conclude that

$$\underline{\gamma}(X, G) = \min_f \gamma(f, X, G) \geq \left(1 - \left(1 - \tfrac{1}{n}\right)^p\right). \tag{5.18}$$

Finally, we conclude the proof by combining (5.18) with Corollary 5.3.2. Given a graph G, let V_{clique} be the set of nodes in the graph that form the largest clique in G. Let $E_{\text{clique}} = \{(i, j) : i, j \in V_{\text{clique}} \text{ and } i < j\}$. Let $\hat{G} = (V, E_{\text{clique}})$, which is a sub-graph of G that only has the edges to form the largest clique. By the definition of $\hat{G}$ we have that $p = \omega(G)$. Using Corollary 5.3.2 we obtain

$$\begin{aligned} \underline{\gamma}(X, G) \geq \underline{\gamma}(X, \hat{G}) &\geq \left(1 - \left(1 - \tfrac{1}{n}\right)^{\omega(G)}\right) \\ &\geq \left(1 - e^{-\frac{\omega(G)}{n}}\right), \end{aligned} \tag{5.19}$$

where (5.19) holds by Corollary 5.3.2 and (5.18). □

Similar to the work in [12], we have the performance guarantees for the case where $X_i = X$ are dependent on the size of the largest clique.

5.3.2 Comparison To Previous Results

A result from [15] states the graph G that maximizes the performance guarantees for Theorem 5.2.1, is the complement Turán graph with minimum independence number given n vertices and m or less edges [15]. A Turán graph is a complete multipartite graph, built by partitioning n vertices into r subsets that are as evenly sized as possible, the edges connects each vertex to all the other vertices that are not in its own subset [15]. A property of the complement Turán graph is the independence number is equal to the number of subsets r.

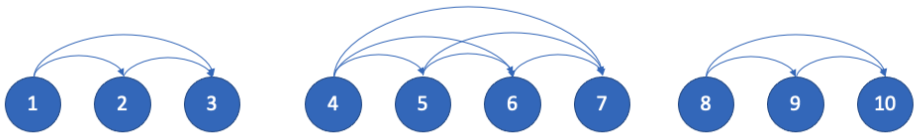

Figure 5.1: Example complement Turán graph $\overline{T(n,r)}$ with $n = 10$ and $r = 3$.

Let $\overline{T(n,r)}$ be the complement Turán graph with n vertices and independence number r, see [15]. The graph that maximizes the bound in Theorem 5.2.1 is defined as

$$\hat{T}(n,m) := \underset{\{\overline{T(n,r)}:|E|\leq m\}}{\arg\min} \quad r.$$

We now provide the following result.

Theorem 5.3.5. *Let $\hat{G} = \hat{T}(n,m)$. Then*

$$1 - e^{-\frac{\omega(\hat{G})}{n}} \geq \frac{1}{\alpha^*(\hat{G}) + 1}.$$

Proof of Theorem 5.3.5. Let $\hat{r}$ be the independence number of the graph $\hat{G} = \hat{T}(n,m)$, we also have the fact that a complement Turán graph with independence number $\alpha(G) = r$ has a maximum clique size of $\lceil \frac{n}{r} \rceil$ [15]. Therefore, we have $\omega(\hat{G}) = \lceil \frac{n}{\hat{r}} \rceil$.

Using this fact we have the following line of inequalities,

$$\begin{aligned} \frac{1}{\alpha^*(\hat{G}) + 1} = \frac{1}{\hat{r} + 1} &\leq \frac{\lceil \frac{n}{\hat{r}} \rceil}{\lceil \frac{n}{\hat{r}} \rceil + n} \\ &= 1 - \frac{1}{1 + \frac{\omega(\hat{G})}{n}} \leq 1 - e^{-\frac{\omega(\hat{G})}{n}}, \end{aligned} \tag{5.20}$$

where the equality in (5.20) holds by the fact $\alpha^*(G) = \alpha(G)$ for compliment Turán graphs [15]. □

We compare the performance guarantees provided by Theorem 5.2.1 to Theorem 5.3.3. In light of Theorem 5.3.5, we know the Turán graph provides a tighter bound in Theorem 5.3.3 than Theorem 5.2.1. Yet it is still unknown if Theorem 5.3.3 is stronger than

Theorem 5.2.1 for every graph. To show that for every graph G that our bound is stronger, we would require an algebraic proof, but to disprove we can find a counter example.

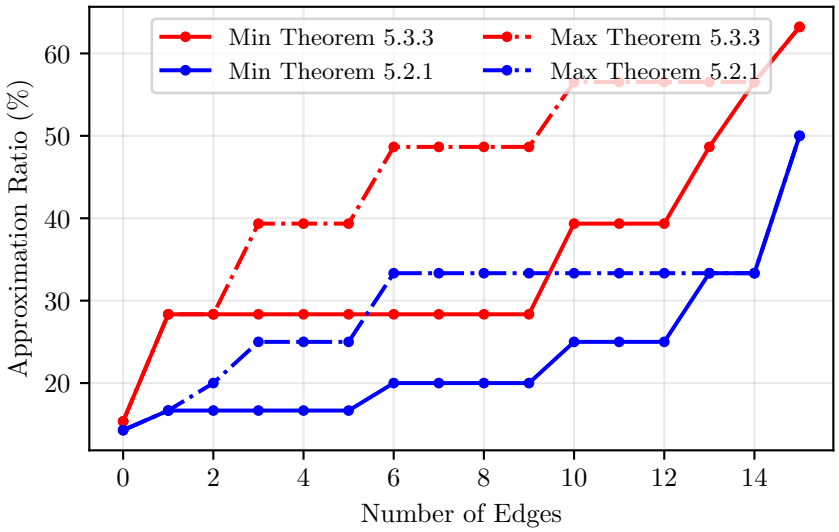

Figure 5.2: Maximum and minimum approximation bounds produced by Theorem 5.3.3 and Theorem 5.2.1 over all graphs with a fixed number of edges.

As a simple experiment, for graphs with 6 vertices or less, we computed both $\omega(G)$ and $\alpha^*(G)$ as well as their corresponding performance bounds. In general, it is not tractable to compute the values of $\omega(G)$ or $\alpha^*(G)$ for an arbitrary graph, but for small graphs the values can be computed in a short period of time. The results from the experiment with 6 vertices is summarized in Figure 5.2. We see that Theorem 5.3.3 had stronger maximum and minimum approximation ratios than Theorem 5.2.1. During this experiment we did not find any graphs G such that the bound in Theorem 5.2.1 was tighter than the bound in Theorem 5.3.3. This does not prove that the bound in Theorem 5.3.3 is tighter in general but we have yet to find any counter examples.

5.4 Discussion

The submodular function used in the proof of Theorem 4.2, used intuition from a solution to a similar linear program. We were searching for examples where imposing pairwise

information constraints on a decision-makers can lead arbitrarily poor results. We wanted to find a function such each of the elements were indistinguishable given only pairwise information, then determine the worst-case approximation ratio that was possible. We formulated the following linear program:

$$\begin{aligned} \max_{v \in \mathbb{R}^{2^N}} \quad & v_{S^*}, \\ \text{s.t.} \quad & \begin{bmatrix} A_{\text{submodular}} \\ A_{\text{monotone}} \end{bmatrix} v \geq 0, \\ & v_S = C, \\ & v_{\{x_i\}} = v_{\{x_j\}} \ \forall \ x_i, x_j \in X, \\ & v_A = v_B \ \forall \ A, B \subseteq X, \ |A| = |B| = 2. \end{aligned} \tag{5.21}$$

To ensure that the elements were indistinguishable by their pairwise values, we enforced constraints such that all the values for single elements are equal and all the values of pairs of elements are equal. We solved this program with $X = S \cup S^*$ and $|S| = |S^*| = 5$. The function that was produced by the optimal solution, had a structure nearly identical to the structure of the function found in the proof of Theorem 4.2. The function had the property that for any $x \in S^*$, $f(x|A) = 1$ for all subsets $A \subseteq X \backslash \{x\}$. For all $x \in S$ marginal return of $f(x|A) = 0$ if $|A| \geq 2$ and $f(x|A) = 1$ if $|A| < 2$. The function also had a maximum value of $\frac{5}{2}C$. Since, all the elements in X are indistinguishable by their pairwise marginals, a strategy could arbitrary select elements all the elements in S, when the optimal solution is to selected all the elements in S^*. This yielded a competitive ratio of $\frac{f(S)}{f(S^*)} = 2/5$. Using the intuition provided by the linear program, we generalized the solution function to prove Theorem 4.2.

Although, the linear programs are only computable for small base sets X, this method can be very effective for disproving conjectures about worst case performance bounds for algorithms executing greedy strategies. One of the caveats though is that the greedy strategy must be representable by a set of linear constraints like the traditional greedy strategy or the pessimistic strategy. The optimistic strategy cannot be written as set of linear constraints. The inequalities we need to encode would have the following form,

$$\bar{f}(x_i|S_{i-1}) = \min_{x_j \in S_{i-1}} f(x_i|x_j) \geq \min_{x_j \in S_{i-1}} f(x|x) \text{ for all } x \in X.$$

Since there are minimums in the expression, we are unable to generate linear constraints. On the other hand, constraints on the curvature of the function can be encoded into linear constraints. It is also possible to encode linear constraints on v enforcing that f possesses supermodularity of conditioning.

5.5 Summary

In this chapter we established a relationship between submodular functions and linear programming. We apply this technique to the distributed submodular maximization problem. Using a linear program we show that removing an edge from the agents' communication graph degrades their worst-case performance. Using this result, we then provide a new performance bound for the scenario where agents share a common action set. We show that the agents' performance is dictated by the size of the largest clique in the communication graph. We provide evidence that the new performance bound is tighter than previous bounds in the literature.

Chapter 6

Conclusion

In this thesis, we investigate sub-problems of submodular maximization where decision-makers are subject to information constraints. In Chapter 4, we explored a submodular maximization problem in which decision-makers have limited access to the objective function. We first showed for maximizing a general submodular function, a decision-maker with k-wise information cannot guarantee a solution with a value greater than k/n of optimal. For simplicity we mainly focus on the case where pairwise information is available. In light of this, we explore different properties that submodular functions can possess to allow greedy algorithms to provide approximation guarantees when limited to k-wise information. We present a simple adapted greedy algorithm called the *optimistic* algorithm, which greedy maximizes an upper bound on the marginal returns. We use a new notion of curvature called the total k-marginal curvature to describe the algorithm's performance. The algorithm can be generalized to take advantage of k-wise information and provide approximation guarantees for the general problem.

We also work with an additional property called supermodularity of conditioning. The property allows for marginal returns of a submodular function to be lower bounded in terms of only pairwise information. With the additional assumption, performance bounds for any greedy algorithm can be computed after execution, using only pairwise information. This leads to another pairwise algorithm called the *pessimistic* algorithm. The pessimistic algorithm greedily maximizes the lower bound on the marginal returns. The performance bound for the pessimistic algorithm is provided in terms of another curvature notion called the k-cardinality curvature, which is closely related to the total k-marginal curvature. The two notions of curvature present unique scenarios where each algorithm performs effectively. We show that both the pairwise algorithms can be computed faster than the traditional greedy strategy, which allows for performance trade-offs between approximation guarantees

and computational efficiency. Finally, we present an experiment using real-world data that show the effectiveness of both pairwise algorithm in terms of approximation performance and time efficiency.

In Chapter 5, we explore the connection between submodular functions and linear programming. We formulate a general linear program to find worst-case submodular function examples. The we directly apply the formulation to provide performance guarantees for the distributed submodular maximization problem. We show using the properties of linear programs, that removing edges from the agents' communication graph degrades the worst-case performance of the greedy strategy. We also provide improved bounds for a special case of the problem where agents share a common action set. The linear programming representation, in general, is intractable to solve because of the large number of constraints imposed to ensure submodularity. For small problems, the programs are solvable and can provide examples of functions where the worst-case performance of a greedy strategy is exhibited. These examples are useful for providing intuition into underlying submodular maximization problems.

6.1 Future Work

6.1.1 Pessimistic Algorithm Generalization

The intuition for the pessimistic algorithm came from coverage functions and utilizing the inclusion-exclusion principle. In [21], the authors introduce submodular combinatorial information measures, which generalize mutual information to submodular functions. They define a measure of mutual information in terms of a submodular function and the inclusion-exclusion principle. The definition provides a means to write any submodular function in terms of the submodular mutual information measure. Using this construction we can generalize the lower bound on the marginal returns of a function defined in Theorem 4.4.3 to utilize k-wise information. If we constrain the submodular functions to be a measure such as cardinality or area, then we can create bounds on the $f(S)$ using sums of $f(A)$ where $A \subseteq X$ and $|A| < k$. This method extends the pessimistic algorithm to utilize k-wise information with stronger performance bounds. From preliminary work, this method seems to be more effective than the k-wise optimistic algorithm because it is able to better utilize the information available and estimate the marginal returns more accurately.

6.1.2 Linear Programming Extension to Approximate Value Oracles

Throughout Chapter 4 we take advantage of results for approximate value oracles for submodular functions to provide approximation guarantees for our proposed algorithms. We can encode approximate greedy algorithms into constraints in the same way we do in Chapter 5. The constraints then can be used to provided performance bounds for an extension of the distributed submodular maximization problem where the agents only have an approximate value oracle to make decisions. In a simple experiment using a small based set X and and a few different communication graphs with 5 nodes, the worst case performance bounds were numerically computed using the linear programming formulation. The results seemed to show that the worst case performance was dictated by the clique with the smallest set of approximation factors. This type of analysis could potentially act as a bridge between the results presented in Chapters 4 and 5.

References

[1] Ashwinkumar Badanidiyuru, Baharan Mirzasoleiman, Amin Karbasi, and Andreas 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*, KDD '14, page 671–680, New York, NY, USA, 2014. Association for Computing Machinery.

[2] 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.

[3] Luiz F. O. Chamon, George J. Pappas, and Alejandro Ribeiro. The mean square error in kalman filtering sensor selection is approximately supermodular. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 343–350, 2017.

[4] Michele Conforti and Gérard Cornuéjols. Submodular set functions, matroids and the greedy algorithm: Tight worst-case bounds and some generalizations of the rado-edmonds theorem. *Discrete Applied Mathematics*, 7(3):251–274, 1984.

[5] M. Corah and N. Michael. Distributed submodular maximization on partition matroids for planning on large sensor networks. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 6792–6799, 2018.

[6] Micah Corah and Nathan Michael. Efficient online multi-robot exploration via distributed sequential greedy assignment. In *Proceedings of Robotics: Science and Systems*, Cambridge, Massachusetts, July 2017.

[7] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009.

[8] Andrew Downie, Bahman Gharesifard, and Stephen L. Smith. Submodular maximization with limited function access. *arXiv preprint arXiv:2201.00724*, 2022.

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

[10] Uriel Feige, Vahab S. Mirrokni, and Jan Vondrák. Maximizing non-monotone submodular functions. *SIAM Journal on Computing*, 40(4):1133–1153, 2011.

[11] Tobias Friedrich, Andreas Göbel, Frank Neumann, Francesco Quinzan, and Ralf Rothenberger. Greedy maximization of functions with bounded curvature under partition matroid constraints. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):2272–2279, Jul. 2019.

[12] B. Gharesifard and S. L. Smith. Distributed submodular maximization with limited information. *IEEE Transactions on Control of Network Systems*, 5(4):1635–1645, 2018.

[13] Xavier Goaoc, Jiří Matoušek, Pavel Paták, Zuzana Safernová, and Martin Tancer. Simplifying inclusion–exclusion formulas. *Combinatorics, Probability and Computing*, 24(2):438–456, 2015.

[14] Pranava R Goundan and Andreas S Schulz. Revisiting the greedy approach to submodular set function maximization. *Optimization online*, pages 1–25, 2007.

[15] D. Grimsman, M. S. Ali, J. P. Hespanha, and J. R. Marden. The impact of information in distributed submodular maximization. *IEEE Transactions on Control of Network Systems*, 6(4):1334–1343, 2019.

[16] D. Grimsman, M. R. Kirchner, J. P. Hespanha, and J. R. Marden. The impact of message passing in agent-based submodular maximization. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 530–535, 2020.

[17] A. Hashemi, M. Ghasemi, H. Vikalo, and U. Topcu. Randomized greedy sensor selection: Leveraging weak submodularity. *IEEE Transactions on Automatic Control*, 66(1):199–212, 2021.

[18] Q. Hou and A. Clark. Robust maximization of correlated submodular functions under cardinality and matroid constraints. *IEEE Transactions on Automatic Control*, pages 1–1, 2021.

[19] Satoru Iwata, Lisa Fleischer, and Satoru Fujishige. A combinatorial strongly polynomial algorithm for minimizing submodular functions. *J. ACM*, 48(4):761–777, July 2001.

[20] Rishabh Iyer and Jeff Bilmes. Submodular optimization with submodular cover and submodular knapsack constraints. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, page 2436–2444, Red Hook, NY, USA, 2013. Curran Associates Inc.

[21] Rishabh Iyer, Ninad Khargoankar, Jeff Bilmes, and Himanshu Asanani. Submodular combinatorial information measures with applications in machine learning. In Vitaly Feldman, Katrina Ligett, and Sivan Sabato, editors, *Proceedings of the 32nd International Conference on Algorithmic Learning Theory*, volume 132 of *Proceedings of Machine Learning Research*, pages 722–754. PMLR, 16–19 Mar 2021.

[22] Syed Talha Jawaid and Stephen L. Smith. Informative path planning as a maximum traveling salesman problem with submodular rewards. *Discrete Appl. Math.*, 186(C):112–127, may 2015.

[23] Ehsan Kazemi, Marko Mitrovic, Morteza Zadimoghaddam, Silvio Lattanzi, and Amin Karbasi. Submodular streaming in all its glory: Tight approximation, minimum memory and low adaptive complexity. In *International Conference on Machine Learning*, pages 3311–3320. PMLR, 2019.

[24] A. Krause, C. Guestrin, A. Gupta, and J. Kleinberg. Near-optimal sensor placements: maximizing information while minimizing communication cost. In *2006 5th International Conference on Information Processing in Sensor Networks*, pages 2–10, 2006.

[25] Andreas Krause and Daniel Golovin. Submodular function maximization. *Tractability*, 3:71–104, 2014.

[26] Andreas Krause and Carlos Guestrin. Submodularity and its applications in optimized information gathering. *ACM Trans. Intell. Syst. Technol.*, 2(4), July 2011.

[27] Andreas Krause, Jure Leskovec, Carlos Guestrin, Jeanne VanBriesen, and Christos Faloutsos. Efficient sensor placement optimization for securing large water distribution networks. *Journal of Water Resources Planning and Management*, 134(6):516–526, 2008.

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

[29] 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*, pages 912–920, Los Angeles, California, June 2010. Association for Computational Linguistics.

[30] Hui Lin and Jeff Bilmes. A class of submodular functions for document summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, page 510–520, USA, 2011. Association for Computational Linguistics.

[31] Hui Lin, Jeff Bilmes, and Shasha Xie. Graph-based submodular selection for extractive summarization. In *2009 IEEE Workshop on Automatic Speech Recognition Understanding*, pages 381–386, 2009.

[32] Z. Liu, A. Clark, P. Lee, L. Bushnell, D. Kirschen, and R. Poovendran. Submodular optimization for voltage control. *IEEE Transactions on Power Systems*, 33(1):502–513, 2018.

[33] L. Lovasz. *Submodular functions and convexity*, pages 235–257. Springer Berlin Heidelberg, Berlin, Heidelberg, 1983.

[34] S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M.B. Srivastava. Coverage problems in wireless ad-hoc sensor networks. In *Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No.01CH37213)*, volume 3, pages 1380–1387 vol.3, 2001.

[35] Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, Amin Karbasi, Jan Vondrak, and Andreas Krause. Lazier than lazy greedy. *Proceedings of the AAAI Conference on Artificial Intelligence*, 29(1), Feb. 2015.

[36] Baharan Mirzasoleiman, Amin Karbasi, Rik Sarkar, and Andreas Krause. Distributed submodular maximization. *Journal of Machine Learning Research*, 17(235):1–44, 2016.

[37] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions—i. *Mathematical Programming*, 14(1):265–294, 1978.

[38] New York Taxi and Limousine Commission. Tlc trip record data, 2020.

[39] James B. Orlin, Andreas S. Schulz, and Rajan Udwani. Robust monotone submodular function maximization. *Mathematical Programming*, 172(1):505–537, Nov 2018.

[40] Adarsh Prasad, Stefanie Jegelka, and Dhruv Batra. Submodular meets structured: Finding diverse subsets in exponentially-large structured item sets. *Advances in Neural Information Processing Systems*, 27:2645–2653, 2014.

[41] J. Qin, I. Yang, and R. Rajagopal. Submodularity of storage placement optimization in power networks. *IEEE Transactions on Automatic Control*, 64(8):3268–3283, 2019.

[42] M. Roberts, S. Shah, D. Dey, A. Truong, S. Sinha, A. Kapoor, P. Hanrahan, and N. Joshi. Submodular trajectory optimization for aerial 3d scanning. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5334–5343, 2017.

[43] Omid Sadeghi and Maryam Fazel. Online continuous dr-submodular maximization with long-term budget constraints. In *International Conference on Artificial Intelligence and Statistics*, pages 4410–4419. PMLR, 2020.

[44] Dravyansh Sharma, Ashish Kapoor, and Amit Deshpande. On greedy maximization of entropy. In *International Conference on Machine Learning*, pages 1330–1338, 2015.

[45] Serban Stan, Morteza Zadimoghaddam, Andreas Krause, and Amin Karbasi. Probabilistic submodular maximization in sub-linear time. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3241–3250, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

[46] H. Sun, D. Grimsman, and J. R. Marden. Distributed submodular maximization with parallel execution. In *2020 American Control Conference (ACC)*, pages 1477–1482, 2020.

[47] Xinmiao Sun, Christos G. Cassandras, and Xiangyu Meng. A submodularity-based approach for multi-agent optimal coverage problems. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 4082–4087, 2017.

[48] V. Tzoumas, K. Gatsis, A. Jadbabaie, and G. J. Pappas. Resilient monotone submodular function maximization. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 1362–1367, 2017.

[49] V. Tzoumas, A. Jadbabaie, and G. J. Pappas. Robust and adaptive sequential submodular optimization. *IEEE Transactions on Automatic Control*, pages 1–1, 2020.

[50] V. Tzoumas, M. A. Rahimian, G. J. Pappas, and A. Jadbabaie. Minimal actuator placement with bounds on control effort. *IEEE Transactions on Control of Network Systems*, 3(1):67–78, 2016.

[51] Jan Vondrak. Submodularity and curvature: the optimal algorithm. *RIMS Kôkyûroku Bessatsu*, 01 2010.

[52] Bang Wang. Coverage problems in sensor networks: A survey. *ACM Comput. Surv.*, 43(4), oct 2011.

[53] Kai Wei, Rishabh Iyer, and Jeff Bilmes. Fast multi-stage submodular maximization. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, page II–1494–II–1502. JMLR.org, 2014.

[54] Kai Wei, Rishabh Iyer, and Jeff Bilmes. Submodularity in data subset selection and active learning. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1954–1963, Lille, France, 07–09 Jul 2015. PMLR.

[55] Ji-Jie Wu and Kuo-Shih Tseng. Adaptive submodular inverse reinforcement learning for spatial search and map exploration. *Autonomous Robots*, 46(2):321–347, 2022.

[56] L. Zhou, V. Tzoumas, G. J. Pappas, and P. Tokekar. Distributed attack-robust submodular maximization for multi-robot planning. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2479–2485, 2020.

[57] Tianyi Zhou, Hua Ouyang, Jeff Bilmes, Yi Chang, and Carlos Guestrin. Scaling Submodular Maximization via Pruned Submodularity Graphs. In Aarti Singh and Jerry Zhu, editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 316–324, Fort Lauderdale, FL, USA, 20–22 Apr 2017. PMLR.