Combinatorial Problems and Scalability in Artificial Intelligence

Francesco Quinzan

Publikationsbasierte Universitätsdissertation
zur Erlangung des akademischen Grades

doctor rerum naturalium
(Dr. rer. nat.)

in der Wissenschaftsdisziplin
Theoretische Informatik

eingereicht an der
Digital-Engineering-Fakultät
der Universität Potsdam

Betreuer

Prof. Dr. Tobias Friedrich
Hasso Plattner Institute, University of Potsdam

Gutachter

Prof. Dr. Jeff A. Bilmes
University of Washington
Prof. Dr. Amin Karbasi
Yale University

Published online on the
Publication Server of the University of Potsdam:
https://doi.org/10.25932/publishup-61111
https://nbn-resolving.org/urn:nbn:de:kobv:517-opus4-611114

Abstract

Modern datasets often exhibit diverse, feature-rich, unstructured data, and they are massive in size. This is the case of social networks, human genome, and e-commerce databases. As Artificial Intelligence (AI) systems are increasingly used to detect pattern in data and predict future outcome, there are growing concerns on their ability to process large amounts of data. Motivated by these concerns, we study the problem of designing AI systems that are *scalable* to very large and heterogeneous data-sets.

Many AI systems require to solve combinatorial optimization problems in their course of action. These optimization problems are typically NP-hard, and they may exhibit additional side constraints. However, the underlying objective functions often exhibit additional properties. These properties can be exploited to design suitable optimization algorithms. One of these properties is the well-studied notion of *submodularity*, which captures diminishing returns. Submodularity is often found in real-world applications. Furthermore, many relevant applications exhibit generalizations of this property.

In this thesis, we propose new scalable optimization algorithms for combinatorial problems with diminishing returns. Specifically, we focus on three problems, the *Maximum Entropy Sampling* problem, *Video Summarization*, and *Feature Selection*. For each problem, we propose new algorithms that work at scale. These algorithms are based on a variety of techniques, such as forward step-wise selection and adaptive sampling. Our proposed algorithms yield strong approximation guarantees, and the perform well experimentally.

We first study the Maximum Entropy Sampling problem. This problem consists of selecting a subset of random variables from a larger

set, that maximize the entropy. By using diminishing return properties, we develop a simple forward step-wise selection optimization algorithm for this problem. Then, we study the problem of selecting a subset of frames, that represent a given video. Again, this problem corresponds to a submodular maximization problem. We provide a new adaptive sampling algorithm for this problem, suitable to handle the complex side constraints imposed by the application. We conclude by studying Feature Selection. In this case, the underlying objective functions generalize the notion of submodularity. We provide a new adaptive sequencing algorithm for this problem, based on the Orthogonal Matching Pursuit paradigm.

Overall, we study practically relevant combinatorial problems, and we propose new algorithms to solve them. We demonstrate that these algorithms are suitable to handle massive datasets. However, our analysis is not problem-specific, and our results can be applied to other domains, if diminishing return properties hold. We hope that the flexibility of our framework inspires further research into scalability in AI.

Zusammenfassung

Moderne Datensätze bestehen oft aus vielfältigen, funktionsreichen und unstrukturierten Daten, die zudem sehr groß sind. Dies gilt insbesondere für soziale Netzwerke, das menschliche Genom und E-Commerce Datenbanken. Mit dem zunehmenden Einsatz von künstlicher Intelligenz (KI) um Muster in den Daten zu erkennen und künftige Ergebnisse vorherzusagen, steigen auch die Bedenken hinsichtlich ihrer Fähigkeit große Datenmengen zu verarbeiten. Aus diesem Grund untersuchen wir das Problem der Entwicklung von KI-Systemen, die auf große und heterogene Datensätze skalieren.

Viele KI-Systeme müssen während ihres Einsatzes kombinatorische Optimierungsprobleme lösen. Diese Optimierungsprobleme sind in der Regel NP-schwer und können zusätzliche Nebeneinschränkungen aufwiesen. Die Zielfunktionen dieser Probleme weisen jedoch oft zusätzliche Eigenschaften auf. Diese Eigenschaften können genutzt werden, um geeignete Optimierungsalgorithmen zu entwickeln. Eine dieser Eigenschaften ist das wohluntersuchte Konzept der Submodularität, das das Konzept des abnehmende Erträge beschreibt. Submodularität findet sich in vielen realen Anwendungen. Darüber hinaus weisen viele relevante Anwendungen Verallgemeinerungen dieser Eigenschaft auf.

In dieser Arbeit schlagen wir neue skalierbare Algorithmen für kombinatorische Probleme mit abnehmenden Erträgen vor. Wir konzentrieren uns hierbei insbesondere auf drei Probleme: dem Maximum-Entropie-Stichproben Problem, der Videozusammenfassung und der Feature Selection. Für jedes dieser Probleme schlagen wir neue Algorithmen vor, die gut skalieren. Diese Algorithmen basieren auf verschiedenen Techniken wie der schrittweisen Vorwärtsauswahl und

dem adaptiven sampling. Die von uns vorgeschlagenen Algorithmen bieten sehr gute Annäherungsgarantien und zeigen auch experimentell gute Leistung.

Zunächst untersuchen wir das Maximum-Entropy-Sampling Problem. Dieses Problem besteht darin, zufällige Variablen aus einer größeren Menge auszuwählen, welche die Entropie maximieren. Durch die Verwendung der Eigenschaften des abnehmenden Ertrags entwickeln wir einen einfachen forward step-wise selection Optimierungsalgorithmus für dieses Problem. Anschließend untersuchen wir das Problem der Auswahl einer Teilmenge von Bildern, die ein bestimmtes Video repräsentieren. Dieses Problem entspricht einem submodularen Maximierungsproblem. Hierfür stellen wir einen neuen adaptiven Sampling-Algorithmus für dieses Problem zur Verfügung, das auch komplexe Nebenbedingungen erfüllen kann, welche durch die Anwendung entstehen. Abschließend untersuchen wir die Feature Selection. In diesem Fall verallgemeinern die zugrundeliegenden Zielfunktionen die Idee der submodularität. Wir stellen einen neuen adaptiven Sequenzierungsalgorithmus für dieses Problem vor, der auf dem Orthogonal Matching Pursuit Paradigma basiert.

Insgesamt untersuchen wir praktisch relevante kombinatorische Probleme und schlagen neue Algorithmen vor, um diese zu lösen. Wir zeigen, dass diese Algorithmen für die Verarbeitung großer Datensätze geeignet sind. Unsere Auswertung ist jedoch nicht problemspezifisch und unsere Ergebnisse lassen sich auch auf andere Bereiche anwenden, sofern die Eigenschaften des abnehmenden Ertrags gelten. Wir hoffen, dass die Flexibilität unseres Frameworks die weitere Forschung im Bereich der Skalierbarkeit im Bereich KI anregt.

Acknowledgments

I would like to that Prof. Tobias Friedrich for his support and mentorship. Tobias was always supportive of my research activities, and he encouraged me to be creative and follow my intuition. As an employee of the Algorithm Engineering group, I had the opportunity to get to know many researchers, and take part in various scientific activities. I am very grateful to Tobias for his support.

I also would like to thank the many students and researchers that I had the opportunity to work with during my studies. Many thanks to Dr. Timo Kötzing and Prof. Andrew Sutton for introducing me to their field of research. Many thanks to Dr. Aneta Neumann, Dr. Markus Wagner and Prof. Frank Neumann for working with me, and hosting me at their institute. I also would like to thank Dr. Andreas Göbel, for helping me in developing the topic of Submodular Optimization. Many thanks also to Dr. Rajiv Khanna and Prof. Michael Mahoney for introducing me to the machine learning community, and for working with me. I also would like to thank Dr. Sarel Cohen, Moshik Hershcovitch, and Dr. Alon Eden for working with me on Feature Selection and Combinatorial Auctions. Many thanks to Dr. Jia-Jie Zhu and Prof. Bernard Shölkopf for working with me on machine learning, and hosting me at their institute.

I would like to thank the many fellow students that I had the opportunity to work with, particularly Vanja Doskoc and Christopher Weyand. Besides them, I am grateful to every member of the Algorithm Engineering group, for their support and fruitful discussions. Many thanks to Dr. Samuel Baguley, Dr. Katrin Casel, Dr. Agnes Cseh, Dr. Davis Issac, Dr. Nikhil Kumar, Dr. Pascal Lenzner, Philipp Fishbeck, Maximilian Katzmann, Simo Krogmann, Gregor Lagodzin-

ski, Anna Melnichenko, Louise Molitor, Stefan Neubert, Aikaterini Niklanovits, Marcus Pappik, Aishwarya Radhakrishnan, Ralf Rothenberger, Martin Schirneck, Karen Seidel, and Ziena Zeif.

Besiders my collaborators and mentors I would like to thank my parents Maria Laura Almini, Gian Paolo Quinzan, and my brother Giovanni Quinzan, for their support and affection.

Contents

1 Introduction

Several fundamental problems pertaining the development of Artificial Intelligence systems can be embedded as combinatorial optimization problems. This is the case of Data Summarization, Feature Selection, and Spread of influence in Social Networks (see Table 1.1-1.2).

On a high level, we consider combinatorial problems that are described as follows. Given a ground set $[n] = \{1, \ldots, n\}$ of discrete items, find a subset $\mathsf{S} \subseteq [n]$ that maximizes a function $f(\mathsf{S})$, expressing the goodness of a given solution. In Data Summarization, for instance, the set set $[n]$ consists of all of the data-points, and the function f measures how well a set of data-points summarizes the entire dataset. For these optimization problems, we typically assume *oracle access* to the function f. That is, the function f is evaluated in a single operation, and its implementation and inner workings are not known. However, oracle valuations are typically time-consuming. Hence, fast algorithms ought to minimize the total number of oracle calls during run time. The total number of oracle calls required by an algorithm largely depends on the underlying properties of f. For example, if f is a linear function, then there exists an algorithm that converges to the optimum after polynomial calls to f. However, on general *monotone* functions there is no algorithm that achieves a constant-factor approximation guarantee after polynomial calls to f, unless P=NP.

A class of functions of intermediate complexity are submodular functions. These functions capture the notion of diminishing returns, i.e. the more we acquire the less our marginal gain will be. Submodular functions are particularly interesting, because they pertain several real-world problems. Examples of real-world submodular problems

Problem	Description
Data Summarization (Iyer and Bilmes 2015)	Create a subset that represents the most relevant information within the original content.
Data Subset Selection (Wei et al. 2015)	Select a subset of big data to train a classifier, while incurring minimal performance loss.
Maximum Welfare in Combinatorial Auctions (Feige and Vondrák 2010)	Given a smart market in which participants place bids on combinations of discrete items, identify an optimal allocation strategy.
Spread of Influence in Social Networks (Karimi et al. 2017)	Identify "influencers" in a social network, i.e., users that are likely to quickly spread information.
Maximum Cut (Feige, Mirrokni, et al. 2011)	Partition the vertices of a graph into two disjoint subsets, such that the number of edges connecting these partitions is maximum.

Table 1.1: Common optimization problems in AI, that consist of maximizing submodular functions. Maximizing submodular functions is NP-hard.

are presented in Table 1.1. Formally, consider a ground set $[n]$ of discrete items and a function f expressing the goodness of fit. We say that f is *submodular* if for all subsets $\mathsf{S}, \mathsf{T} \subseteq [n]$, it holds

$$f(\mathsf{S}) + f(\mathsf{T}) \geq f(\mathsf{S} \cup \mathsf{T}) + f(\mathsf{S} \cap \mathsf{T}).$$

One can easily prove that this property is equivalent to a diminishing returns property.

Submodular minimization can be embedded as a convex optimization problem (Lovász 1982). However, the problem of maximizing a submodular function is more complex. We show that submodular maximization is NP-hard, by considering the following well-known example (Feige, Mirrokni, et al. 2011). Let $\mathcal{G} = (\mathsf{V}, E)$ be a (directed) graph with n vertices and m edges. We consider the problem of finding a subset $\mathsf{U} \subseteq \mathsf{V}$ of nodes such that the sum of the weights on the outgoing edges of U is maximal. This problem is known as the Maximum Directed Cut problem, and it is a well-known NP-hard

Problem	Description
Dictionary Selection (Das and Kempe 2011)	Given a set of predictor variables $\{Z_i\}_i$, select a set of d observation variables optimizing the average loss for each Z_j, using at most k of the selected vectors.
Feature Selection (Krause, A. P. Singh, et al. 2008)	Select a subset of relevant variables or predictors for use in model construction. This problem is a special case of Dictionary Selection.
Compressed Sensing (Das and Kempe 2011)	A signal processing technique to efficiently acquire and reconstruct a signal.
LPs with Combinatorial Constraints (A. A. Bian et al. 2017)	Maximize an LP of the form $\max_{\mathbf{x}\in\mathscr{P}}\langle\mathbf{d},\mathbf{x}\rangle$, where $\mathbf{d}$ is a real-valued vector, and $\mathscr{P}$ is a polytope representing continuous cost constraints.

Table 1.2: Common combinatorial optimization problems in AI. These problems can be approached by optimizing functions that generalize the notion of submodularity.

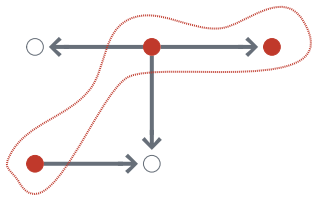


Figure 1.1: A simple, directed graph $\mathscr{G}$. The cut function $f(\mathsf{U})$ for a sets of U returns the number of outgoing edges from that set. In this case $f(\mathsf{U}) = 3$, where U is the set highlighted in red.

problem. To establish a connection with submodularity, we first define the following combinatorial function, called the cut function. For each subset of nodes $\mathsf{U} \subseteq [n]$, consider the set $C(\mathsf{U}) := \{(e_1, e_2) \in E\colon e_1 \in \mathsf{U} \text{ and } e_2 \notin \mathsf{U}\}$. We define the cut function as $f(\mathsf{U}) := |C(\mathsf{U})|$. For an illustration of the cut function see Figure 1.1. The maximum directed cut problem can be approached by maximizing the cut function. As noted by Feige, Mirrokni, et al. 2011, the cut function is a submodular function. Hence, submodular maximization is NP-hard.

Oftentimes, in real-world applications a solution is subject to side constraints. Among the most common constraints are matroid and

knapsack constraints (Iyer and Bilmes 2013; Lee et al. 2009). Other, more complex constraints include p-matchoids, p-extendable systems, and p-systems (Chekuri and Quanrud 2019a; Moran Feldman, Harshaw, et al. 2017; Mestre 2006). Although monotone functions under simple side constraints can be optimized efficiently (Cornuejols et al. 1977; Nemhauser and Wolsey 1978), it is highly non-trivial to maximize non-monotone submodular functions under general side constraints.

1.1 Scope of this Thesis

Motivated by the many applications (see Table 1.1-1.2), we study combinatorial problems for AI and machine learning, that consists of maximizing submodular functions. From a technical prospective, we study submodularity, to design fast approximation algorithms for this problem. Although the notion of submodularity is widely studied, there are several problems and challenges that remain open. In fact, known algorithms to maximize functions that are non-monotone, or under complex side constraints, do not scale to large data-sets very well (Lee et al. 2009; Mirzasoleiman, Jegelka, et al. 2018a). Hence, I am interested in studying the following research question:

How can we efficiently maximize non-monotone submodular functions under complex side constraints?

Another related question pertain the use of submodularity techniques to tackle non-submodular problems. In fact, many interesting combinatorial optimization problems fulfill axioms that generalize the notion of submodularity. This is the case of Feature Selection and Compressed Sensing (see Table 1.2). Hence, we study the following research question:

Can we generalize results for submodular functions to non-submodular problems?

We hope that a systematic study of submodularity and its generalizations will allow for the development of optimization algorithms, suitable to handle massive datasets.

1.2 Literature Overview

One of the earliest work on submodular maximization is the work of Nemhauser and Wolsey 1978. In this paper, Nemhauser and Wolsey 1978 show that no-polynomial time algorithm can achieve a better approximation ratio than $(1-1/e)$. The classical result of Cornuejols et al. 1977 shows that a greedy algorithm achieves a $1/2$ approximation ratio for the problem of maximizing monotone submodular functions under partition matroid constraints. Similar bounds on the approximation guarantee hold for monotone submodular functions under multiple linear constraints (Kulik et al. 2009).

These bounds do not hold if the optimization function is non-monotone. Feige, Mirrokni, et al. 2007 prove that a deterministic local search achieves a $1/3$-approximation guarantee for maximizing a non-monotone submodular function, and that a randomized local search achieves a $2/5$-approximation guarantee. Feige, Mirrokni, et al. 2007 show that there is no polynomial-time $(1/2+\varepsilon)$-approximation algorithm to maximize a non-negative symmetric submodular function, unless P = NP. Buchbinder, Moran Feldman, Naor, et al. 2012 propose a randomized algorithm that achieves a $1/2$-approximation on this problem. Similar bounds can also be achieved with a deterministic algorithm (Buchbinder, Moran Feldman, and Garg 2019).

Another important problem is the study of non-monotone submodular maximization under additional side constraints. If side constraints consist of a simple matroid, then a randomized greedy algorithm achieves an approximation of $1/e$ (Călinescu et al. 2011). This result can be further improved to a 0.5008-approximation ratio by derandomizing a search heuristic (Buchbinder, Moran Feldman,

and Garg 2018). It is possible to maximize a non-monotone submodular function under matroid constraints, with an algorithm based on simulated annealing (Gharan and Vondrák 2011). This algorithm achieves 0.41-approximation for unconstrained submodular maximization, and a 0.325-approximation for submodular maximization subject to a matroid independence constraint. However, it is impossible to achieve a 0.491-approximation, for submodular maximization under a simple cardinality constraint (Gharan and Vondrák 2011). The aforementioned upper bounds can be further improved with variants of the greedy algorithm (Moran Feldman, Naor, et al. 2011a; Filmus and Ward 2012), and using multilinear extensions and content resolution schemes (Călinescu et al. 2007; Vondrák et al. 2011).

The problem of maximizing a non-monotone submodular function under multiple matroids and knapsacks is studied by Lee et al. 2009. This problem is significantly more complex than the case of simple matroid constraints. The problem of maximizing a submodular function under k-column sparse packing constraints is studied by Bansal et al. 2010. They propose an approximation algorithm for this problem, with approximation guarantees parameterized by k. It is possible to maximize submodular functions under general polytopes constraints using multilinear extensions and content-resolution schemes (Chekuri, Vondrák, et al. 2010; Vondrák et al. 2011). Ene and Nguyen 2016 proposed an algorithm that achieves a 0.372-approximation for the problem of maximizing a non-monotone continuous submodular function under a downward closed polytope.

Submodular maximization problems have also been studied in distributed settings. Mirzasoleiman, Karbasi, Sarkar, et al. 2013; Mirzasoleiman, Karbasi, Sarkar, et al. 2016; Mirzasoleiman, Zadimoghaddam, et al. 2016 propose a map-reduce style algorithm for this problem. The solution quality achieved by this algorithm is similar to that of the centralized approach. Mirrokni and Zadimoghaddam 2015 propose a randomized map-reduce approach for submodular maximization under cardinality constraints, that achieves a 0.545-

approximation guarantee. Other map-reduce style algorithms have been proposed to tackle various problems in Machine Learning, such as data summarization (Mirzasoleiman, Karbasi, Badanidiyuru, et al. 2015; Ponte Barbosa et al. 2015) and multi-label feature selection (Ghadiri and Schmidt 2019).

Using an adaptive sampling framework, it is possible to design algorithms to maximize submodular functions that can be efficiently parallelized (Thompson 1990). These algorithms achieve a constant-factor approximation of the optimal solution in poly-logarithmic iterations, by performing polynomial independent oracle queries in each iteration. Specifically, algorithms with poly-logarithmic adaptivity are known for maximizing monotone submodular functions under cardinality constraints (Balkanski and Singer 2018; Fahrbach et al. 2019b), or matroid constraints (Balkanski, Rubinstein, et al. 2019). Efficient algorithms in the adaptive complexity model are also known for non-monotone submodular maximization under cardinality constraints (Balkanski, Breuer, et al. 2018; Fahrbach et al. 2019a). Interestingly, Ene and Nguyen 2019 gives a constant-factor approximation algorithm in the adaptive complexity model, suitable to handle matroid and packing constraints. Lower-bounds for subodular maximization in the adaptive-complexity model are also known (Balkanski and Singer 2018; Li et al. 2020).

Another relevant question that has been studied in recent year is whether submodular functions can be approximated. A relevant work in this direction is the paper by Goemans et al. 2009. In this paper, the authors propose an algorithm that approximates a submodular function at any point, that achieves an approximation factor of $\tilde{\mathcal{O}}(\sqrt{n})$. Goemans et al. 2009 also provide a $\tilde{\Omega}(\sqrt{n})$ approximation on this problem. Balcan and Harvey 2011 study the problem of "learning submodular function" in the PMAC model. That is, we are given a set of polynomial labelled solutions $\mathcal{S} \subseteq 2^{[n]}$ according to an unknown submodular target function f^*, and we want to output a hypothesis function f that approximates f^* with high probability. Balcan and

Harvey 2011 give an algorithm that achieves a $\mathcal{O}(\sqrt{n})$ for this problem, and also a $\tilde{\Omega}(n^{1/3})$ lower bound. They conclude that, although it is possible to learn submodular functions from samples, it is not possible to find an approximate global maximum of f^* by optimizing f instead. Balkanski, Rubinstein, et al. 2017 study the problem of optimizing functions from samples. The define a class of functions that is learnable in the PMAC model, but for which no constant-factor approximation from samples can be achieved.

Due to the success of submodular optimizers, many authors have studied the question of generalizing submodular optimization methodologies to non-submodular functions. Das and Kempe 2011 propose the notion of weak-submodularity, to generalize the work of Krause and Cevher 2010 on the dictionary selection problem. The notion of weak submodularity has been studied in connection with the interpretability of black-box classifiers, such as neural networks (Elenberg, Dimakis, et al. 2017), and feature selection (Khanna et al. 2017). A. A. Bian et al. 2017 extend the forward-stepwise technique to all monotone functions, via the greedy submodularity ratio.

1.3 Contribution and Outline

Our main contribution consists in developing fast algorithms for relevant combinatorial problems in AI and machine learning, that scale to very large datasets. We focus on three major problems: *Maximum Entropy Sampling*, *Video Summarization*, and *Feature Selection*. We consider an axiomatic characterization of each problem based on the notion of submodularity and its generalizations. We then use these axioms to study new algorithms for the aforementioned problems. In doing so, we develop new techniques and algorithms that can be applied to other domains.

In Chapter 2 we study the Maximum Entropy Sampling problem. Given a set of random variables, the Maximum Entropy Sampling

problem consists of selecting a subset of these variables that maximizes the entropy. The Maximum Entropy Sampling problem arises in connection with sensor placement problems. The Maximum Entropy Sampling problem corresponds to a non-monotone submodular maximization problem. We show that a simple forward step-wise selection algorithm finds a good approximation of the optimal solution, if the underlying objective function has bounded curvature.

In Chapter 3 we study Video Summarization. This problem consists of selecting a subset of frames, that "summarize" a given video. This problem is an instance of Data Summarization, which consists of sampling a representative set of points from a large data-set. Video Summarization is a non-monotone submodular maximization problem under additional side constraints. We provide a new algorithm, based on the adaptive sampling paradigm, that finds a good approximation of the optimal solution in poly-logarithmic rounds of adaptivity. This algorithm achieves significant speed-up over previous known techniques, if parallelized.

In Chapter 4 we study the problem of selecting optimal features for model construction. This problem corresponds to the maximization of functions that generalize the notion of submodularity. We provide a new adaptive sequencing algorithm for this problem, based on the Orthogonal Matching Pursuit paradigm. This algorithm converges after poly-logarithmic adaptive rounds, and it outputs a near-optimal solution.

In Chapter 5, we conclude by discussing the main results, their implications, and we discuss possible future research activities.

2 Maximum Entropy Sampling

This chapter is based on a conference paper titled "Greedy Maximization of Functions with Bounded Curvature under Partition Matroid Constraints", by Francesco Quinzan, and Ralf Rothernberger, Andreas Göbel, Frank Neumann, Tobias Friedrich (Quinzan, Göbel, et al. 2019). This chapter is also based on a conference paper titled "Non-Monotone Submodular Maximization with Multiple Knapsacks in Static and Dynamic Settings", by Francesco Quinzan and Vanja Doskoč, Aneta Neumann, Andreas Göbel, Frank Neumann, Tobias Friedrich (Doskoc et al. 2019).

Given a set of random variables, the maximum entropy sampling problem consists of selecting a subset of these variables that maximizes the entropy over all sets of the same size. This problem arises in connection with spatio-temporal dynamics monitoring.

Motivated by this application, we investigate the performance of a deterministic GREEDY algorithm for the problem of maximizing non-monotone submodular functions under a partition matroid constraint. Even though constrained maximization problems of monotone submodular functions have been extensively studied, little is known about greedy maximization of non-monotone submodular functions or monotone subadditive functions.

We give approximation guarantees for GREEDY on these problems, in terms of the curvature. We find that this simple heuristic yields a strong approximation guarantee on a broad class of functions. We perform experiments on a real-world dataset, to show that our algorithm works well in practice.

We conclude that GREEDY is well-suited to approach these problems. Overall, we present evidence to support the idea that, when

dealing with constrained maximization problems with bounded curvature, one needs not search for (approximate) monotonicity to get good approximate solutions.

2.1 Submodularity and Maximum Entropy Sampling

We study the problem of maximizing a non-monotone submodular function, under a partition matroid constraint. Before giving a formal definition of this problem (see Section 2.2), we first discuss an application that motivates the study of this problem. This application is the maximum entropy sampling problem. Given set of random variables $\{X_1, \ldots, X_n\}$, we consider the problem of identifying a small subset of these variables that best explain the model. This problem finds a broad spectrum of applications, from Bayesian experimental design (Sebastiani and Henry P. Wynn 2002), to path planning for robots coordination (A. Singh et al. 2009). A concrete application of this framework is given by sensor placement tasks. Consider a subset of sensors deployed over a surface. The measurements of these sensors can be described as sets of random variables $\{X_1, \ldots, X_n\}$, and searching for an informative subset of the variables corresponds to identifying a subset of sensors that give the most comprehensive overview of the overall measurement.

Many criteria have been proposed for characterizing the amount of information explained by a set of random variables (Cressie 2015; Shewry and Henry P Wynn 1987; Zhu and Stein 2006; Zimmerman 2006). One of the most significant ones is the Shannon entropy (Shannon 1948). The Shannon entropy was first introduced by Claude Shannon as part of his theory of communication. The problem of searching for a set of random variables that maximizes the Shannon entropy is commonly referred to as the maximum entropy sampling problem.

Assuming that the random variables $\{X_1, \ldots, X_n\}$ follow a joint Gaussian distribution with mean vector $\boldsymbol{\sigma} = \mathbf{0}$ and variance $\boldsymbol{\Sigma}$, then the Shannon entropy is given by the formula

$$\text{entropy}(\mathsf{S}) = \frac{1+\ln(2\pi)}{2}|\mathsf{S}| + \frac{1}{2}\ln\det(\boldsymbol{\Sigma}_{\mathsf{S}}). \tag{2.1}$$

Here, S is an indexing set for the random variables, $|\mathsf{S}|$ is the size of this set, and $\boldsymbol{\Sigma}_{\mathsf{S}}$ is the covariance matrix of the random variables $\{X_j\}_{j\in\mathsf{S}}$. It is well-known that the Shannon entropy is a submodular function. However, the Shannon entropy needs not be a monotone function. For instance, consider a set of two random variables $\{X_1, X_2\}$ with covariance matrix

$$\boldsymbol{\Sigma} = \begin{pmatrix} 2+\ln(2\pi)+\varepsilon & \sqrt{1+\ln(2\pi)+\varepsilon} \\ \sqrt{1+\ln(2\pi)+\varepsilon} & 1 \end{pmatrix}$$

for a constant $\varepsilon > 0$. In this case, the entropy attains the following values:

$$\begin{aligned} \text{entropy}(\{1\}) &= \frac{3}{2} + \ln 2\pi + \frac{\varepsilon}{2} \\ \text{entropy}(\{2\}) &= 1 + \frac{\ln 2\pi}{2} \\ \text{entropy}(\{1,2\}) &= \frac{3}{2} + \ln 2\pi, \end{aligned}$$

and it holds $\text{entropy}(\{1\}) > \text{entropy}(\{1,2\})$. Hence, the Shannon entropy is a non-monotone function in general.

2.2 Problem Formulation

We study the problem of maximizing a non-monotone submodular function, under additional side constraints, typically imposed by the underlying application. We consider a kind of side constraints called

partition matroids. Under these side constraints, elements of the ground set are partitioned into different categories, and only a fixed amount of items are allowed from each category. Our problem can be then stated as follows.

Problem 1. *Let* $f\colon 2^{[n]} \longrightarrow \mathbb{R}_{\geq 0}$ *be a non-negative submodular function*[1]*, let* $\mathsf{B}_1,\dots,\mathsf{B}_k$ *be a collection of disjoint subsets of* $[n]$*, and consider* $r_1,\dots,r_k$ *integers such that* $1 \leq r_i \leq |\mathsf{B}_i|$*, for all* $i \in [k]$*. We consider the maximization problem*

$$\arg\max_{\mathsf{S}\subseteq[n]} \{f(\mathsf{S}) : |\mathsf{S}\cap\mathsf{B}_i| \leq r_i,\ \forall i \in [k]\}.$$

We denote with $\mathrm{OPT} \subseteq [n]$ *any solution to this problem, and we define* $\mathtt{OPT} := f(\mathrm{OPT})$.

Note that the problem of maximizing f under a cardinality constraint on the size of the solution is a special case of Problem 1. To simplify the exposition, throughout our analyses, we always assume that the following reduction holds.

Reduction 1. *For Problem 1 we may assume* $cr_i \leq |\mathsf{B}_i|$ *for all* $i \in [k]$*, for an arbitrary constant* $c > 0$*. Moreover, we may assume that there exists a set* $\mathsf{D}_i \subseteq \mathsf{B}_i$ *of size* r_i *s.t.* $f(\mathsf{S}) = f(\mathsf{S}\setminus\mathsf{D}_i)$ *for all* $\mathsf{S} \subseteq [n]$*, for all* $i \in [k]$.

Proof. Fix a constant $c > 0$. We observe that if the condition of the statement does not hold, then it is sufficient to add a set D of $\sum_i cr_i$ "dummy" elements that do not have any effect on the values attained by the function f, and remove them from the output of the algorithm, for all $i \in [k]$. Denote with $\mathsf{D}_1,\dots,\mathsf{D}_k$ a partition of D with $|\mathsf{D}_i| = cr_i$ for all $i \in [k]$. This only increases the size of the instance by a multiplicative constant factor. We define new subsets $\overline{\mathsf{B}}_i = \mathsf{B}_i \cup \mathsf{D}_i$ for all $i \in [k]$. Thus, we can maximize the function f on the newly-defined

1 We always assume that f is normalized, that is $f(\emptyset) = 0$.

partition constraint without affecting neither the global optimum, nor the value of the algorithm's output. ■

2.2.1 The Curvature

In our analysis, we give approximation guarantees in terms of the curvature. Intuitively, the curvature is a parameter that bounds the maximum rate with which a function changes. A function f has curvature α if the value $f(\mathsf{S} \cup \{s\}) - f(\mathsf{S})$ does not change by a factor larger than $1-\alpha$ when varying S. This parameter was first introduced by (Conforti and Cornuéjols 1984) to beat the $(1-e^{-1})$-approximation barrier of monotone submodular functions. Formally we use the following definition of curvature, relaxing the definition of the greedy curvature proposed by (A. A. Bian et al. 2017).

▶ **Definition 2.1 (Curvature).** Consider a non-negative function $f\colon 2^{[n]} \to \mathbb{R}$ as in Problem 1. The curvature is the smallest scalar α s.t.

$$f_{(\mathsf{S}\cup\mathsf{T})\setminus\{s\}}(s) \geq (1-\alpha) f_{\mathsf{S}\setminus\{s\}}(s),$$

for all $\mathsf{S}, \mathsf{T} \subseteq [n]$ and $s \in \mathsf{S} \setminus \mathsf{T}$. ◀

The curvature α as define above is always non-negative. We say that a function f has positive curvature if $\alpha \leq 1$. Otherwise, we say that f has negative curvature. Note that a function is monotone if and only if it has positive curvature.

We remark that the curvature is invariant under multiplication by a positive scalar. In other words, if a function f has curvature α, then any function cf has curvature α, for all $c > 0$. Moreover, the following simple result holds.

▶ **Proposition 2.1.** Let $f, g\colon 2^{[n]} \to \mathbb{R}_{\geq 0}$ be non-negative functions with curvature α_1, α_2 respectively. Then the curvature α of the function $f+g$ is upper-bounded as $\alpha \leq \sup\{\alpha_1, \alpha_2\}$. ◀

Algorithm 1: The GREEDY algorithm.

```
S ← ∅;
while |S| ≤ Σ_{i=1}^{k} r_i do
    let s ∈ [n] maximizing f(S ∪ {s}) − f(S) and s.t.
      |(S ∪ {s}) ∩ B_i| ≤ r_i, ∀i ∈ [k];
    S ← S ∪ {s};
return S;
```

Proof. Fix two subsets $S, T \subseteq [n]$ of size at most d, and a point $s \in S \setminus T$. From the definition of curvature we have

$$f(S \cup T) - f((S \cup T) \setminus \{s\}) \geq (1 - \alpha_1)(f(S) - f(S \setminus \{s\})),$$
$$g(S \cup T) - g((S \cup T) \setminus \{s\}) \geq (1 - \alpha_2)(g(S) - g(S \setminus \{s\})).$$

Thus, we have that it holds

$$\begin{aligned}
&(f+g)(S \cup T) - (f+g)((S \cup T) \setminus \{s\}) \\
&\quad = f(S \cup T) - f((S \cup T) \setminus \{s\}) + g(S \cup T) - g((S \cup T) \setminus \{s\}) \\
&\quad \geq (1 - \alpha_1)(f(S) - f(S \setminus \{s\})) + (1 - \alpha_2)(g(S) - g(S \setminus \{s\})) \\
&\quad \geq (1 - \sup\{\alpha_1, \alpha_2\})\big(f(S) - f(S \setminus \{s\}) + g(S) - g(S \setminus \{s\})\big) \\
&\quad = (1 - \sup\{\alpha_1, \alpha_2\})\big((f+g)(S) - (f+g)(S \setminus \{s\})\big),
\end{aligned}$$

as claimed. ∎

2.3 The Simple Greedy Algorithm

GREEDY is the simple discrete greedy algorithm that appears in Algorithm 5. Starting with the empty set, GREEDY iteratively adds points that maximize the marginal values with respect to the already found solution. This algorithm is a mild generalization of the simple deter-

ministic greedy algorithm due to Nemhauser and Wolsey (Nemhauser and Wolsey 1978).

We give approximation guarantees for GREEDY on Problem 1, when optimizing a non-monotone submodular function with bounded curvature α. Our proof technique generalizes the results of (Conforti and Cornuéjols 1984) to non-monotone functions f by utilizing the notion of curvature. We have the following theorem.

▶ **Theorem 2.1.** Let f be a submodular function with curvature α. Let S^* be the output of Algorithm 5, and let $\mathtt{OPT}$ be the optimal f-value for Problem 1. Then it holds

$$\frac{f(\mathsf{S}^*)}{\mathtt{OPT}} \geq \frac{1}{\alpha}\left(1 - \exp\left\{-\alpha \frac{\sum_{i\in[k]} r_i}{\min_{i\in[k]} r_i}\right\}\right),$$

with $r_1, \ldots, r_k$ as in Problem 1. ◀

Note that in the case of a uniform matroid, the bound of Theorem 2.1 becomes

$$\frac{f(\mathsf{S}^*)}{\mathtt{OPT}} \geq \frac{1}{\alpha}\big(1 - \exp\{-\alpha\}\big).$$

Hence, if f is monotone, then our approximation guarantee matches the approximation guarantee of Conforti and Cornuéjols, which is known to be nearly optimal (Conforti and Cornuéjols 1984; Vondrák 2010), in the uniform matroid case. Furthermore, in the non-monotone case our lower-bound may yield significant improvement over state-of-the-art known bounds (Buchbinder and Moran Feldman 2018; Buchbinder, Moran Feldman, Naor, et al. 2014). Particularly, we beat the $1/e$-approximation barrier on functions with curvature $\alpha \leq 2.5$ and the $1/2$-approximation barrier on functions with curvature $\alpha \leq 1.6$ (see Figure 2.1).

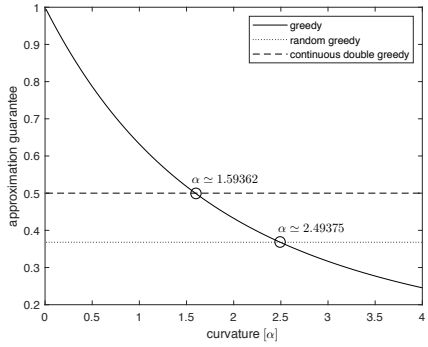


Figure 2.1: Approximation guarantees for a non-monotone submodular function under a cardinality constraint for varying curvature, as in Theorem 2.1. We observe that on functions with sufficiently small curvature we outperform state-of-the-art bounds (Buchbinder and Moran Feldman 2018; Buchbinder, Moran Feldman, Naor, et al. 2014) for this problem.

2.4 Run Time Analysis for the Greedy

In this section, we prove Theorem 2.1. On a high level, this proof essentially consists of showing that the marginal contributions obtained by the GREEDY are always lower-bounded by the solution of an LP. We then prove the desired approximation guarantee, by exploiting the properties of this LP. Before proving Theorem 2.1, we discuss preliminary results pertaining this LPs.

2.4.1 Preliminary Results

In this section, we define and study the LPs that will be used in the proof of Theorem 2.1. These LPs are defined as follows.

▶ **Definition 2.2.** Fix a constant $\alpha \in \mathbb{R}_{>0}$, and let p, r be non-negative integers with $p \leq r$. Fix a set $\mathsf{J} \subseteq [r]$, and define the sets $\mathsf{J}_i \coloneqq \{j \in \mathsf{J} \colon j \leq i\}$ for all $i \in [r]$. Then, we denote with $\mathscr{R}(\alpha, p, r, \mathsf{J})$ the LP is defined as

$$(r - |\mathsf{J}_{t-1}|)x_t + \alpha \sum_{j \in [t-1]} x_j + (1-\alpha) \sum_{j \in \mathsf{J}_{t-1}} x_j \geq 1, \ \forall t \in [p].$$

◀

Note that any LP as in Definition 2.2 is completely determined by α, p, r, and J, and it can be written as

$$\begin{bmatrix} r & & & & \\ \alpha & (r-|\mathsf{J}_1|) & & & \\ \alpha & 1 & & & \\ \alpha & 1 & & & \\ \vdots & \vdots & & & \\ \vdots & \vdots & & & \\ \vdots & \vdots & & & \\ \vdots & \vdots & & & \\ \alpha & 1 & \cdots & (r-|\mathsf{J}_{p-1}|) & \\ \alpha & 1 & \cdots & \alpha & (r-|\mathsf{J}_p|) \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ x_{p-1} \\ x_p \end{bmatrix} \geq \begin{bmatrix} 1 \\ 1 \\ 1 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ 1 \\ 1 \end{bmatrix}$$

We say that an array $(y_1, \ldots, y_p) = \mathbf{y}$ with non-negative coefficients $y_j \geq 0$ is an optimal solution of $\mathscr{R}(\alpha, p, r, \mathsf{J})$ if $\mathbf{y}$ is a solution of $\mathscr{R}(\alpha, p, r, \mathsf{J})$, and if it holds

$$\sum_{j=1}^{p} z_j \geq \sum_{j=1}^{p} y_j,$$

for any solution $(z_1, \ldots, z_p) = \mathbf{z}$ of $\mathscr{R}(\alpha, p, r, \mathsf{J})$ with non-negative coefficients $z_j \geq 0$. Optimal solutions have the following important property.

▶ **Lemma 2.1.** Let $(y_1, \ldots, y_p) = \mathbf{y}$ be an optimal solution of $\mathscr{R}(\alpha, p, r, \mathsf{J})$ with $\mathsf{J} \neq \emptyset$. Then it holds $y_j \leq y_{j+1}$ for all $j \in \mathsf{J}$. ◀

Proof. We proceed ad absurdum, by assuming that there exists a point $i \in \mathsf{J}$ such that $y_i > y_{i+1}$, and then proving that this assump-

tion contradicts the optimality of $\mathbf{y}$. To this end, define the positive constant

$$\varepsilon = \frac{r - |\mathsf{J}_i|}{r - |\mathsf{J}_{i-1}|}(y_i - y_{i+1}).$$

Note that ε is non-negative, under the assumption that $y_i > y_{i+1}$. Consider a vector $(z_1, \ldots, z_r) = \mathbf{z}$, defined as

$$\begin{cases} z_j = y_j & \text{if } 1 \leq j < i; \\ z_j = y_j - \varepsilon & \text{if } j = i; \\ z_j = y_j + \frac{\varepsilon}{r - |J_i|} & \text{if } i < j \leq p; \end{cases}$$

To continue with the proof, we use the following claim.

▶ **Claim 2.1.** The vector $(z_1, \ldots, z_p) = \mathbf{z}$ is a feasible solution of the linear program $\mathscr{R}(\alpha, p, r, \mathsf{J})$. ◀

Proof of Claim 2.1. To prove this claim, we define the following function

$$L(\mathbf{x}, t) := (r - |\mathsf{J}_{t-1}|)x_t + \alpha \sum_{j \in [t-1]} x_j + (1 - \alpha) \sum_{j \in \mathsf{J}_{t-1}} x_j. \tag{2.2}$$

This function takes as input a vector $(x_1, \ldots, x_p) = \mathbf{x}$, an integer t, and it it returns the value obtained by multiplying the t-th row of the constraint matrix of $\mathscr{R}(\alpha, p, r, \mathsf{J})$ with the vector $\mathbf{x}$. Note that, in order to prove Claim 2.1, it is sufficient to show that it holds

$$L(\mathbf{z}, t) \geq 1, \quad \text{for all } t \in [p]. \tag{2.3}$$

We show that (2.3) holds, by considering four cases.

Case 1: $t < i$. This case follows immediately from the definition of $\mathbf{z}$ and the optimality of $\mathbf{y}$, since it holds $L(\mathbf{z}, t) = L(\mathbf{y}, t)$ and $L(\mathbf{y}, t) \geq 1$.

Case 2: $t = i$. We have that it holds

$$\begin{aligned}
L(\mathbf{z}, i) &= (r - |\mathsf{J}_{i-1}|)z_i + \alpha \sum_{j\in[i-1]} z_j + (1-\alpha) \sum_{j\in\mathsf{J}_{i-1}} z_j \\
&= (r - |\mathsf{J}_{i-1}|)(y_i - \varepsilon) + \alpha \sum_{j\in[i-1]} y_j + (1-\alpha) \sum_{j\in\mathsf{J}_{i-1}} y_j \\
&= (r - |\mathsf{J}_{i-1}|)\left(y_i - \frac{r - |\mathsf{J}_i|}{r - |\mathsf{J}_{i-1}|}(y_i - y_{i+1})\right) \\
&\qquad + \alpha \sum_{j\in[i-1]} y_j + (1-\alpha) \sum_{j\in\mathsf{J}_{i-1}} y_j \\
&= (r - |\mathsf{J}_i|)y_{i+1} + (|\mathsf{J}|_i - |\mathsf{J}|_{i-1})y_j \\
&\qquad + \alpha \sum_{j\in[i-1]} y_j + (1-\alpha) \sum_{j\in\mathsf{J}_{i-1}} y_j \\
&= (r - |\mathsf{J}_i|)y_{i+1} + \alpha \sum_{j\in[i]} y_j + (1-\alpha) \sum_{j\in\mathsf{J}_i} y_j \\
&= L(\mathbf{y}, i+1)
\end{aligned}$$

where the third equation uses the definition of ε, and the fifth one uses the fact that $i \in \mathsf{J}$. Using the optimality of $\mathbf{y}$, we have that $L(\mathbf{z}, i) = L(\mathbf{y}, i+1) \geq 1$, and (2.3) holds for $t = i$.

Case 3: $t > i$. In this case, we prove that it holds $L(\mathbf{z}, t) \geq L(\mathbf{y}, t)$, for all $t > i$ with an induction argument on t. For the base case with $t = i+1$, we have that

$$\begin{aligned}
L(\mathbf{z}, i+1) &= (r - |\mathsf{J}_i|)z_{i+1} + \alpha \sum_{j\in[i]} z_j + (1-\alpha) \sum_{j\in\mathsf{J}_i} z_j \\
&= (r - |\mathsf{J}_i|)\left(y_{i+1} + \frac{\varepsilon}{r - |J_i|}\right) + \alpha \sum_{j\in[i]} z_j + (1-\alpha) \sum_{j\in\mathsf{J}_i} z_j \\
&= (r - |\mathsf{J}_i|)y_{i+1} + \varepsilon + \alpha \sum_{j\in[i]} y_j + (1-\alpha) \sum_{j\in\mathsf{J}_i} y_j - \varepsilon \\
&= L(\mathbf{y}, i+1),
\end{aligned}$$

and the base step holds. To prove the inductive step, note that by the definition of the function L, it holds $L(\mathbf{x}, t+1) - L(\mathbf{x}, t) \geq (r - |\mathsf{J}_t|)(x_{t+1} - x_t)$, for any vector $(x_1, \ldots, x_p) = \mathbf{x}$. Hence it holds

$$\begin{aligned} L(\mathbf{z}, t+1) - L(\mathbf{z}, t) - L(\mathbf{y}, t+1) + L(\mathbf{y}, t) \\ &\geq (r - |\mathsf{J}_t|)(z_{t+1} - z_t - y_{t+1} + y_t) \\ &= (r - |\mathsf{J}_t|)\left(\frac{\varepsilon}{r - |J_i|} - \frac{\varepsilon}{r - |J_i|}\right) \\ &= 0. \end{aligned}$$

By rearranging we get $L(\mathbf{z}, t+1) - L(\mathbf{y}, t+1) \geq L(\mathbf{z}, t) - L(\mathbf{y}, t)$. By using the inductive hypothesis on t, it holds $L(\mathbf{z}, t) - L(\mathbf{y}, t) \geq 0$, from which it follows that $L(\mathbf{z}, t+1) \geq L(\mathbf{y}, t+1)$. ■

We conclude the proof of Lemma 2.1, by showing that $\mathbf{z}$ contradicts the minimality of $\mathbf{y}$. To this end, we observe that it holds

$$\sum_{j \in [r]} (z_j - y_j) = -\varepsilon + \sum_{j=i+1}^{p} \frac{\varepsilon}{r - |\mathsf{J}_i|} \leq \varepsilon\left(-1 + \frac{p - |\mathsf{J}_i| - 1}{r - |\mathsf{J}_i|}\right) < 0$$

where the first inequality follows since $|\mathsf{J}_i| \leq i$, the second one uses the assumption that $p \leq r$, and the last one follows since $\varepsilon > 0$. The claim holds. ■

The lemma above is useful, because it allows us to significantly simplify our setting. In fact, using Lemma 2.1 we can prove that the sum of the coefficients of any optimal solution of $\mathscr{R}(\alpha, p, r, \mathsf{J})$ can be lower-bounded using the solutions of a simpler LP. The following lemma holds.

▶ **Lemma 2.2.** Let $(y_1, \ldots, y_p) = \mathbf{y}$ be an optimal solution of $\mathscr{R}(\alpha, p, r, \mathsf{J})$.

Let $(z_1, \dots, z_p) = \mathbf{z}$ be an optimal solution of the following system

$$\begin{bmatrix} r & 0 & \dots & \dots & \dots & 0 \\ \alpha & r & 0 & \dots & \dots & 0 \\ \alpha & \alpha & r & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ \alpha & \dots & \alpha & r & 0 & 0 \\ \alpha & \dots & \dots & \alpha & r & 0 \\ \alpha & \dots & \dots & \dots & \alpha & r \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{p-2} \\ x_{p-1} \\ x_p \end{bmatrix} \geq \begin{bmatrix} 1 \\ 1 \\ 1 \\ \vdots \\ 1 \\ 1 \\ 1 \end{bmatrix} \tag{2.4}$$

which corresponds to $\mathscr{R}(\alpha, p, r, \emptyset)$. Then it holds

$$\sum_{j=1}^{p} y_j \geq \sum_{j=1}^{p} z_j.$$

◀

Proof. Denote with $j_1, \dots, j_k$ the points of J sorted in increasing order. In order to prove the claim, we introduce additional terminology. We say that j_i and j_{i+1} are adjacent if it holds $j_i + 1 = j_{i+1}$. The notion of adjacency is relevant for the proof, because of the following claim.

▶ **Claim 2.2.** Suppose that there exists an index i such that $j_i + 1 \neq j_{i+1}$, and consider the set $\mathsf{J}' := \{j_1, \dots, j_{i-1}, j_i + i, j_{i+1}, j_k\}$. Note that the set J' consists of changing the element j_i in J with the element $j_i + 1$. Then, $\mathbf{y}$ is a solution of $\mathscr{R}(\alpha, p, r, \mathsf{J}')$. ◀

Proof of Claim 2.2. Consider the function $L(\mathbf{x}, t)$ as in (2.2) for the system $\mathscr{R}(\alpha, p, r, \mathsf{J})$. This function takes as input a vector $(x_1, \dots, x_p) = \mathbf{x}$, an integer t, and it it returns the value obtained by multiplying the t-th row of the constraint matrix of $\mathscr{R}(\alpha, p, r, \mathsf{J})$ with the vector $\mathbf{x}$. Similarly, denote with $L'(\mathbf{x}, t)$ the function as in (2.2) for the system $\mathscr{R}(\alpha, p, r, \mathsf{J}')$. We prove the claim by showing that it holds $L'(\mathbf{y}, t) \geq 1$, for all $t \in [p]$. First, observe that by the definition of the sets J and J'

it holds $L'(\mathbf{y}, t) = L(\mathbf{y}, t) \geq 1$, for all $t < j_i$. For $t = j_i$ it holds

$$L'(\mathbf{y}, j_i) = L(\mathbf{y}, j_i) + y_{i_j}, \tag{2.5}$$

since $|\mathsf{J}'_{i_j}| = |\mathsf{J}_{i_j}| - 1$, and it follows that $L'(\mathbf{y}, j_i) \geq 1$. For $t = j_i + 1$ it holds

$$\begin{aligned} L'(\mathbf{y}, j_i + 1) &= (r - \left|\mathsf{J}'_{j_i}\right|) y_{j_i+1} \\ &\quad + \alpha \sum_{j \in [j_i]} y_j + (1-\alpha) \sum_{j \in \mathsf{J}'_{j_i}} y_j - \left|\mathsf{J}'_{j_i} \setminus \mathsf{J}'_{j_i-1}\right| y_{j_i+1} \\ &\geq (r - \left|\mathsf{J}'_{j_i}\right|) y_{j_i} \\ &\quad + \alpha \sum_{j \in [j_i]} y_j + (1-\alpha) \sum_{j \in \mathsf{J}_{j_i}} y_j - \left|\mathsf{J}'_{j_i} \setminus \mathsf{J}'_{j_i-1}\right| y_{j_i+1} \\ &\geq L'(\mathbf{y}, j_i), \end{aligned}$$

where the second inequality uses that $y_{j_i+1} \geq y_{j_i}$ due to Lemma 2.1. Hence, using (2.5) it holds $L'(\mathbf{y}, j_i + 1) \geq L(\mathbf{y}, j_i) \geq 1$. We conclude by showing that it holds $L'(\mathbf{y}, t) \geq 1$, for all $t > j_i + 1$. To this end, we note that it holds

$$L'(\mathbf{y}, t) - L(\mathbf{y}, t) = (\alpha - 1) y_{i_j} + (1 - \alpha) y_{i_j+1} \geq 0,$$

where we have used again that $y_{j_i+1} \geq y_{j_i}$ due to Lemma 2.1. Hence, for $t > j_i + 1$ it holds $L'(\mathbf{y}, t) \geq L(\mathbf{y}, t) \geq 1$, as claimed. ■

Note that, using Claim 2.2 we may assume that all points of J are adjacent, i.e., $j_i + 1 = j_{i+1}$ for all $j_i \in \mathsf{J}$, and that $j_k = p$. Note also that from Lemma 2.1, for any such set J it holds $j_1 \leq j_2 \leq \cdots \leq j_k$. We can use this fact to prove the following claim.

▶ **Claim 2.3.** Suppose that the points $\{j_1, \ldots, j_k\} = \mathsf{J}$ are all adjacent such that $j_k = p$, and let $(y_1, \ldots, y_p) = \mathbf{y}$ be an optimal solution of $\mathscr{R}(\alpha, p, r, \mathsf{J})$. Then, $\mathbf{y}$ is a solution of $\mathscr{R}(\alpha, p, r, \mathsf{J}'')$ with $\mathsf{J}'' = \mathsf{J} \setminus \{j_1\}$. ◀

Proof of Claim 2.3. Denote with $L''(\mathbf{x}, t)$ the function as in (2.2) for the system $\mathscr{R}(\alpha, p, r, \mathsf{J}'')$. We prove that it holds $L''(\mathbf{x}, t) \geq 1$ for all $t \in [p]$. To this end, note that $L''(\mathbf{y}, t) = L(\mathbf{y}, t) \geq 1$ for all $t < j_1$, by the definition of J and J''. For $t = j_i$ it holds

$$\begin{aligned} L''(\mathbf{y}, j_i) &= (r - \left|\mathsf{J}''_{j_i}\right|) y_{j_i} + \alpha \sum_{j \in [j_i - 1]} y_j + (1 - \alpha) \sum_{j \in \mathsf{J}''_{j_i - 1}} y_j \\ &\geq (r - \left|\mathsf{J}_{j_i}\right|) y_{j_i} + \alpha \sum_{j \in [j_i - 1]} y_j + (1 - \alpha) \sum_{j \in \mathsf{J}_{j_i - 1}} y_j \\ &= L(\mathbf{y}, j_i), \end{aligned}$$

and it follows that $L''(\mathbf{y}, j_i) \geq L(\mathbf{y}, j_i) \geq 1$. To continue with the proof, since the points of J are all adjacent, then it holds $j_1 \leq j_2 \leq \cdots \leq j_k = p$, from which it follows that $L''(\mathbf{y}, t) \geq L''(\mathbf{y}, j_i) \geq 1$ for all $j_i < t \leq j_k$. The claim follows. ■

By iteratively applying Claim 2.3 to the set J, we can remove all points $j_i \in \mathsf{J}$, while preserving the feasibility of $\mathbf{y}$. Hence, $\mathbf{y}$ is a feasible solution for $\mathscr{R}(\alpha, p, r, \emptyset)$. ■

We now use Lemma 2.2 to write a lower-bound on the coefficients of any solution of $\mathscr{R}(\alpha, p, r, \mathsf{J})$ in closed form. The following lemma holds.

▶ **Lemma 2.3.** Let $(y_1, \ldots, y_p) = \mathbf{y}$ be a solution of $\mathscr{R}(\alpha, p, r, \mathsf{J})$. Then it holds

$$\sum_{t=1}^{p} y_t \geq \sum_{t=1}^{p} \frac{1}{r} \left(1 - \frac{\alpha}{r}\right)^{t-1}.$$

◀

Proof. Let $(z_1, \ldots, z_p) = \mathbf{z}$ be a solution of the system $\mathscr{R}(\alpha, p, r, \emptyset)$ as in (2.4). By Lemma 2.2 it is sufficient to prove that it holds

$$z_t = \frac{1}{r} \left(1 - \frac{\alpha}{r}\right)^{t-1}, \tag{2.6}$$

for all $i \in [p]$. To prove (2.6), we use an induction argument on t. The base case with $t = 1$ is trivially true. Suppose now that the claim holds for $z_1, \dots, z_t$. Then it holds

$$\begin{aligned} z_{t+1} &= \frac{1}{r}\left(1 - \alpha \sum_{j=1}^{t} z_j\right) \\ &= \frac{1}{r}\left(1 - \alpha \sum_{j=1}^{t} \frac{1}{r}\left(1 - \frac{\alpha}{r}\right)^{j-1}\right) \\ &= \frac{1}{r}\left(1 - \frac{\alpha}{r}\right)^{t}, \end{aligned}$$

and (2.6) holds. The lemma follows. ■

2.4.2 Proof of the Main Theorem

In this section we give a proof the main theorem.

▶ **Theorem 2.1.** Let f be a submodular function with curvature α. Let S^* be the output of Algorithm 5, and let $\mathtt{OPT}$ be the optimal f-value for Problem 1. Then it holds

$$\frac{f(\mathsf{S}^*)}{\mathtt{OPT}} \geq \frac{1}{\alpha}\left(1 - \exp\left\{-\alpha \frac{\sum_{i\in[k]} r_i}{\min_{i\in[k]} r_i}\right\}\right),$$

with $r_1, \dots, r_k$ as in Problem 1. ◀

To prove this theorem, we first show that the value of the current solution obtained by Algorithm 5 at each step fulfills an LP as in Definition (2.2), and then we give a lower-bound on the approximation guarantee using Lemma 2.3.

Proof of Theorem 2.1. We denote with S_t the current solution of Algorithm 5 at time step t, and we denote with OPT the optimal solution, i.e., $f(\mathrm{OPT}) = \mathtt{OPT}$. We assume without loss of generality that f is

non-constant. Moreover, due to Reduction 5, we may assume that

$$\sum_{i\in[k]} r_i \geq \alpha.$$

We perform the analysis until a solution of size $\min_i r_i$ is found. Let D be a set of dummy elements as in Reduction 1. Let M be a set of size

$$|\mathsf{M}| = \sum_{i\in[k]} r_i$$

such that $\mathsf{M} \setminus \mathsf{D} = \text{OPT}$. We have that it holds

$$f_{\mathsf{S}_{t-1}}(\mathsf{S}_t) \geq \frac{1}{|\mathsf{M}\setminus\mathsf{S}_{t-1}|} \sum_{s\in\mathsf{M}\setminus\mathsf{S}_{t-1}} f_{\mathsf{S}_{t-1}}(s) \geq \frac{f_{\mathsf{S}_{t-1}}(\text{OPT})}{|\mathsf{M}\setminus\mathsf{S}_{t-1}|}, \tag{2.7}$$

where we have used that $\mathsf{S}_{t-1}\cup\{s\}$ is always a feasible solution, since $t \leq \min_i t_i$, together submodularity. To continue, we consider the following claim.

▶ **Claim 2.4.** Consider the set $\mathsf{J} = \{t\colon \mathsf{S}_t \setminus \mathsf{S}_{t-1} \in \text{OPT}\}$, and define the sets $\mathsf{J}_i := \{j \in \mathsf{J}\colon j \leq i\}$. Fox any set $\mathsf{T} \subseteq [n]$. Then, it holds

$$f_{\mathsf{S}_{t-1}}(\mathsf{T}) \geq f(\mathsf{T}) + (1-\alpha)\sum_{j\in[t-1]} f_{\mathsf{S}_{t-1}}(\mathsf{S}_t) - (1-\alpha)\sum_{j\in\mathsf{J}_{t-1}} f_{\mathsf{S}_{j-1}}(\mathsf{S}_j),$$

for all t. ◀

Proof. From the definition of curvature we have that $f_{\mathsf{S}_{j-1}\cup\mathsf{T}}(\mathsf{S}_j) \geq (1-\alpha) f_{\mathsf{S}_{j-1}}(\mathsf{S}_t)$ for all $j \in [t]\setminus\mathsf{J}$, and $f_{\mathsf{S}_{j-1}\cup\mathsf{T}}(\mathsf{S}_j) \geq 0$ for all $j \in \mathsf{J}$. Hence, it holds

$$\begin{aligned} f_{\mathsf{S}_{t-1}}(\mathsf{T}) &= f(\mathsf{T}) + \sum_{j\in[t-1]} f_{\mathsf{S}_{j-1}\cup\mathsf{T}}(\mathsf{S}_j) - \sum_{j\in[t-1]} f(\mathsf{S}_j) \\ &\geq f(\mathsf{T}) + (1-\alpha)\sum_{j\in[t]} f_{\mathsf{S}_{t-1}}(\mathsf{S}_t) - (1-\alpha)\sum_{j\in\mathsf{J}_t} f_{\mathsf{S}_{j-1}}(\mathsf{S}_j) - \sum_{j\in[t-1]} f(\mathsf{S}_j) \end{aligned}$$

$$= f(\mathsf{T}) - \alpha \sum_{j\in[t]} f_{\mathsf{S}_{t-1}}(\mathsf{S}_t) - (1-\alpha) \sum_{j\in \mathsf{J}_t} f_{\mathsf{S}_{j-1}}(\mathsf{S}_j),$$

where the first inequality uses the telescopic sum, and the second one follows by the definition of curvature. ■

Note that from Claim 2.4 it follows that

$$f_{\mathsf{S}_{t-1}}(\mathrm{OPT}) \geq \mathtt{OPT} - \alpha \sum_{j\in[t-1]} f_{\mathsf{S}_{t-1}}(\mathsf{S}_t) - (1-\alpha) \sum_{j\in \mathsf{J}_{t-1}} f_{\mathsf{S}_{j-1}}(\mathsf{S}_j),$$

for all t. Combining this inequality with (2.7) and rearranging, it holds

$$\begin{aligned}
&\sum_i (r_i - |\mathsf{J}_{t-1}|) f_{\mathsf{S}_{t-1}}(\mathsf{S}_t) + \alpha \sum_{j\in[t-1]} f_{\mathsf{S}_{j-1}}(\mathsf{S}_j) + (1-\alpha) \sum_{j\in \mathsf{J}_{t-1}} f_{\mathsf{S}_{j-1}}(\mathsf{S}_j) \\
&\geq f_{\mathsf{S}_{t-1}}(\mathrm{OPT}) - (1-\alpha) \sum_{j\in[t-1]} f_{\mathsf{S}_{j-1}}(\mathsf{S}_j) + (1-\alpha) \sum_{j\in \mathsf{J}_{t-1}} f_{\mathsf{S}_{j-1}}(\mathsf{S}_j) \\
&\geq \mathtt{OPT}, \qquad (2.8)
\end{aligned}$$

where the first inequality follows since $\sum_i r_i - |\mathsf{J}_{t-1}| = \mathsf{M} \setminus \mathsf{S}_{t-1}$. Define the quantities

$$y_j := \frac{f_{\mathsf{S}_{j-1}}(\mathsf{S}_j)}{\mathtt{OPT}} \qquad r := \sum_i r_i \qquad p := \min_i r_i. \qquad (2.9)$$

Using this notation, we observe that (2.8) can be rewritten as

$$(r - |\mathsf{J}_{t-1}|) y_t + \alpha \sum_{j\in[t-1]} y_j + (1-\alpha) \sum_{j\in \mathsf{J}_{t-1}} y_j \geq 1.$$

Hence, the vector $\mathbf{y} = (y_1, \ldots, y_p)$ is a solution of the system $\mathscr{R}(\alpha, p, r, \mathsf{J})$ as in Definition 2.2. Then, from Lemma 2.3 we get the following claim.

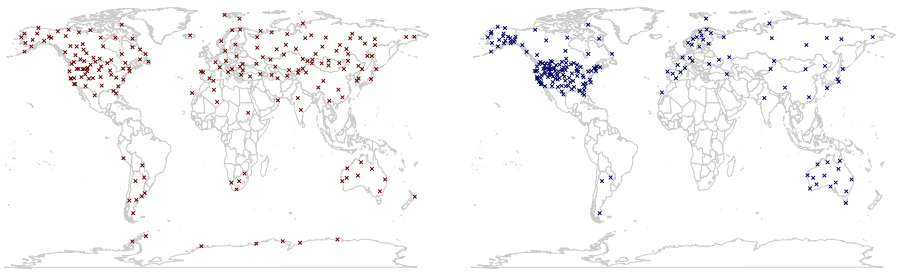


Figure 2.2: A visualization of the solution found by GREEDY for $d = 10\%$ in the case of a uniform constraint (left), and a partition constraint by countries (right). In both case, a solution is obtained by maximizing the entropy as given in (2.1). The covariance matrix $\mathbf{\Sigma}$ for all possible locations is displayed in Figure 2.4. We observe that in the case of a cardinality constraint, the informative stations tend to be spread out, whereas in the partition constraint by countries they tend to be grouped in a few areas. We remark that in the original dataset stations are not distributed uniformly among countries.

▶ **Claim 2.5.** Let $(y_1, \dots, y_p) = \mathbf{y}, r, p$ be as in (2.9). Then it holds

$$\sum_{t=1}^{p} y_t \geq \sum_{t=1}^{p} \frac{1}{r}\left(1 - \frac{\alpha}{r}\right)^{t-1},$$

with α the curvature of f. ◀

Hence, by substituting quantities as in (2.8) it holds

$$\begin{aligned}\frac{f(\mathsf{S}^*)}{\mathsf{OPT}} &\geq \sum_{j=1}^{\min_i r_i} \frac{1}{\sum_i r_i}\left(1 - \frac{\alpha}{\sum_i r_i}\right)^{j-1} \\ &\geq \frac{1}{\alpha}\left(1 - \exp\left\{-\alpha \frac{\sum_{i\in[k]} r_i}{\min_{i\in[k]} r_i}\right\}\right),\end{aligned}$$

where the first inequality holds since the f-values never decrease during the optimization process. The claim follows. ■

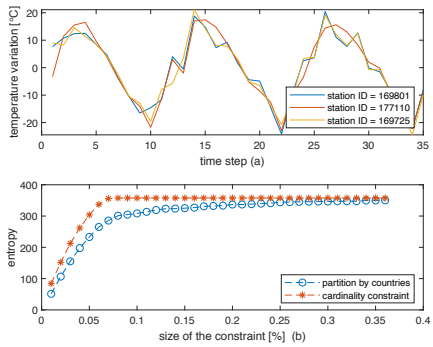


Figure 2.3: (a) A visualization of the monthly temperature variations of three time series, with particularly high variance. Each series corresponds to a unique station ID. We model each variation series as a Gaussian distribution. (b) Optimal solution found by GREEDY for a uniform constraint and a partition matroid constraint by countries. The f-value of each set of stations is the entropy (2.1), with Σ the covariance matrix of variation series as in (a) (see Figure 2.4).

2.5 Experiments

We perform a set of experiments, to test our proposed algorithm on the Maximum Entropy Sampling problem in a real-world scenario. We consider a sensor placement task: Select a subset of stations most suitable to collect new data, based on previous measurements. To this end, we consider the Berkeley Earth climate dataset [2]. This dataset combines 1.6 billion temperature reports from 16 preexisting data archives, for over 39.000 unique stations. For each station, we consider a unique time series for the average monthly temperature. We always consider time series that span between years 2015-2017. This gives us a total of 2736 time series, for unique corresponding stations.

For each time series $\mathbf{X}' = \{X'_t\}_{t\in[m]}$ we study the corresponding variation series $\mathbf{X} = \{X_t\}_{t\in[m]}$ defined as $X_t = X'_t - X'_{t-1}$. A visualization of time series $\mathbf{X}$ is given in Figure 2.3(a). We compute the covariance

2 http://berkeleyearth.org/data/

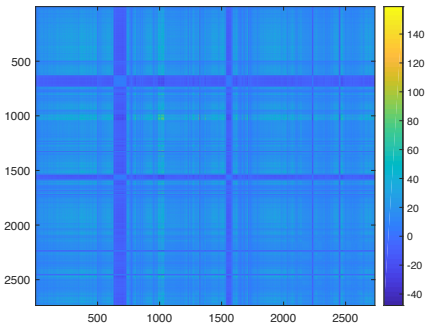


Figure 2.4: A visualization of the covariance matrix Σ of time series available in the Berkley Earth climate dataset. We consider stations that have full available reports between years 2015-2017, for a total of 2736 stations. We consider the variation between average monthly temperatures of each time series. Each entry of this matrix is computed by taking the sample covariance as in (2.10).

matrix $\mathbf{\Sigma}$ between series $\mathbf{X}$, $\mathbf{Y}$, the entries of which are defined as

$$\mathrm{cov}(\mathbf{X},\mathbf{Y}) = \frac{1}{m-1}\sum_{t\in[m]}(X_t - \mathbb{E}[\,\mathbf{X}\,])(Y_t - \mathbb{E}[\,\mathbf{Y}\,]), \tag{2.10}$$

with $m = 35$ the length of each series. A visualization of the covariance matrix $\mathbf{\Sigma}$ is given in Figure 2.4. Assuming that the joint probability distribution is Gaussian, we proceed by maximizing the corresponding entropy, as in (2.1).

We consider two types of constraints. In a first set of experiments we consider the problem of maximizing the entropy under a cardinal constraint only. Specifically, given a parameter d, the goal is to find a subset of time series that maximizes the entropy, of size at most d of all available data. We also consider a more complex set of constraints: Find a subset of time series that maximizes the entropy, and s.t. it contains at most d of all available data of each country. The latter constraint is a partition matroid constraint, where each subset B_i consists of all data series measured by stations in a given country.

A summary of the results is displayed in Figure 2.3(b). We observe that in both cases the entropy quickly evolves to a stationary local optimum, indicating that a relatively small subset of stations is sufficient

to explain the random variations between monthly observations in the model. We observe that the GREEDY reaches similar approximation guarantees in both cases. We remark that the GREEDY finds a nearly optimal solution under a cardinality constraint, assuming that the entropy is (approximately) monotone (Krause, A. P. Singh, et al. 2008).

In Figure 2.2 we display solutions found by GREEDY for the cardinality and partition matroid constraint, with $d = 10\%$. We observe that in the case of a cardinality constraint, the sensors spread across the map; in the case of a partition matroid constraint sensors tend to be placed unevenly. We remark that in the original data set, some countries have a much higher density of stations than others.

2.6 Submodular Maximization under Knapsack Constrains

So far we studied the Maximum Entropy Sampling problem under simple cardinality constraints, and partition matroid constraints. However, practical applications might exhibit other additional cost constraints. For instance, in the sensor placement application, one might have to take into account operational costs, when selecting thermal stations to perform future measurements with.

Motivated by this application, we give approximation guarantees fr the problem of maximizing a submodular function under knapsack constraints, as in the following problem.

Problem 2. *Let $f\colon 2^{[n]} \to \mathbb{R}_{\geq 0}$ be a submodular function.*[3] *Consider linear cost functions $c_i\colon 2^{[n]} \to \mathbb{R}_{\geq 0}$,*[4] *and corresponding weights W_i,*

3 We assume that $f(\emptyset) = 0$, and that f is non-constant.
4 We assume that $\max_j c_j(e) > 0$ for all $e \in [n]$.

Algorithm 2: The k-GREEDY algorithm.

$\mathsf{X} \leftarrow \{e \in [n] : c_j(e) \leq W_j/k \ \forall j \in [k]\}$;
$v^* \leftarrow \arg\max_{e \in \mathsf{X}} f(e)$;
$\mathsf{S} \leftarrow \emptyset$;
while $\mathsf{X} \neq \emptyset$ **do**
    let $e \in \mathsf{X}$ maximizing $f_{\mathsf{S}}(e)/\max_j c_j(e)$;
    $\mathsf{X} \leftarrow \mathsf{X} \setminus e$;
    if $c_j(\mathsf{S} \cup e) \leq W_j, \forall j \in [k]$ **then** $\mathsf{S} \leftarrow \mathsf{S} \cup e$;
return $\arg\max\{f(\mathsf{S}), f(v^*)\}$;

for all $i \in [k]$. *We search for a set* $\mathrm{OPT} \subseteq [n]$, *such that*

$$\mathrm{OPT} \in \arg\max_{\mathsf{S} \subseteq [n]} \{f(\mathsf{S}) : c_i(\mathsf{S}) \leq W_i, \forall i \in [k]\}.$$

In this section, we denote with OPT any solution to Problem 2, and we define $\mathtt{OPT} = f(\mathrm{OPT})$. In this setting, one has k knapsacks and wishes to find an optimal set of items such that its total cost, expressed by the functions c_i, does not violate the capacity of each knapsack. Note that the same set might have different costs for different knapsacks.
We denote with (c, W) the constraint requirements $c_i(\mathsf{S}) \leq W_i$ for all $\mathsf{S} \subseteq [n]$, for all $i \in [k]$.

2.6.1 The Greedy for k-Knapsacks

We approach Problem 2 with a variant of the GREEDY (Khuller et al. 1999; Zhang and Vorobeychik 2016). We refer to this algorithm as the k-GREEDY.The k-GREEDY optimizes f over the set X, containing all singletons $e \in [n]$ such that $c_j(e) \leq W_j/k$ for all $j \in [k]$. After finding a greedy approximate solution S over this set, the k-GREEDY finds the

maximum among the feasible singletons v^*. The k-GREEDY outputs the set with highest f-value among S and v^*.

2.6.2 Approximation Guarantees

We prove that Algorithm 2 yields a strong approximation guarantee, when maximizing a submodular function under k knapsack constraints in the static case. This part of the analysis does not consider dynamic weight updates. We use the notion of curvature as in Definition 2.1.

▶ **Theorem 2.2.** Let f be a submodular function with curvature α. The k-GREEDY is a $(1-e^{-1/k})/(2\max(1,\alpha))$-approximation algorithm for Problem 2. Its run time is $\mathcal{O}(n^2)$. ◀

Note that if the function f is monotone, then the approximation guarantee given in Theorem 2.2 asymptotically matches well-known results (Khuller et al. 1999). In our analysis we always assume that the following reduction holds.

Reduction 2. *For Problem 2 we may assume that there exists a point $e^* \in [n]$ such that $f(\mathsf{S} \cup e^*) = f(\mathsf{S})$ for all $\mathsf{S} \subseteq [n]$, and $c_i(e^*) = W_i$ for all $i \in [k]$. Furthermore, we may assume that $c_i(e) \leq W_i$, for all $e \in [n]$ and for all $i \in [m]$.*

If the conditions of Reduction 2 do not hold, one can remove all points $e \in [n]$ that violate one of the constraints and add a point e^* without altering the function f. Intuitively, Reduction 2 ensures that $\arg\max_{e\in[n]} f(e)$ is always feasible in all constraints, and that the set OPT consists of at least one point. Furthermore, the point $e^* \in [n]$ ensures that the solution quality never decreases throughout a greedy optimization process, until a non-feasible solution is reached.

Proof of Theorem 2.2. For simplicity, without loss of generality we assume that $W_1 = \cdots = W_k$. We denote with W the weight of each

knapsack. We first observe that the k-GREEDY has two phases. During the first phase, k-GREEDY iteratively adds points to the current solution in a greedy fashion. During the second phase, k-GREEDY finds the optimum among single-element sets. The greedy procedure requires at most $\mathcal{O}\left(n^2\right)$ steps, while the second procedure requires at most $\mathcal{O}(n)$ steps. Hence, the resulting run time is $\mathcal{O}\left(n^2\right)$.

We now prove the k-GREEDY yields the desired approximation guarantee.Let S_t be a solution generated during the firs phase of k-GREEDY, at time step t. Furthermore, define $v_t = \mathsf{S}_t \setminus \mathsf{S}_{t-1}$. Let $r \geq 0$ be the smallest index, such that

1. $c_j(\mathsf{S}_{r-1}) \leq W$ for all $j \in [k]$;

2. there exists $j \in [k]$ such that $c_j(\mathsf{S}_r) > W$.

In other words, r is the first point it time such that the new greedy solution does not fulfill all knapsacks at the same time. We first prove that either the solution S_{r-1} or the point $v^* = \arg\max_{e \in \mathsf{X}} f(e)$ yields a good approximation guarantee of OPT. We have that it holds

$$\begin{aligned} f_{\mathsf{S}_{t-1}}(\mathrm{OPT}) &\leq \sum_{e \in \mathrm{OPT} \setminus \mathsf{S}_{t-1}} f_{\mathsf{S}_{t-1}}(e) \\ &= \sum_{e \in \mathrm{OPT} \setminus \mathsf{S}_{t-1}} \max_j c_j(e) \frac{f_{\mathsf{S}_{t-1}}(e)}{\max_j c_j(e)} \\ &\leq \frac{f(\mathsf{S}_t)}{\max_j c(v_t)} \sum_{e \in \mathrm{OPT} \setminus \mathsf{S}_{t-1}} \max_j c_j(e) \\ &\leq \frac{kW}{\max_j c(v_i)} f(\mathsf{S}_t) \end{aligned}$$

where the first inequality follows from the assumption that f is submodular; the second inequality follows due to the greedy choice of Algorithm 2; the last inequality uses the fact that $c(\mathrm{OPT}) \leq W$, together with the fact that $c(e) \leq kW$ for all $c \in \mathsf{X}$, for all $j \in [k]$. Rearranging

yields

$$f(\mathsf{S}_t) \geq \frac{\max_j c_j(v_i)}{kW} f_{\mathsf{S}_{t-1}}(\mathrm{OPT}). \tag{2.11}$$

To continue with the proof, we use Claim 2.4.Following Claim 2.4 it holds

$$\begin{aligned} f_{\mathsf{S}_{t-1}}(\mathrm{OPT}) &\geq f(\mathrm{OPT}) + (1-\alpha) \sum_{j \in [t-1] \setminus \mathsf{J}_{t-1}} f_{\mathsf{S}_{j-1}}(\mathsf{S}_j) \\ &\geq f(\mathrm{OPT}) + (1-\max(1,\alpha)) \sum_{j \in [t-1] \setminus \mathsf{J}_{t-1}} f_{\mathsf{S}_{j-1}}(\mathsf{S}_j) \\ &\geq f(\mathrm{OPT}) + (1-\max(1,\alpha)) \sum_{j \in [t-1]} f_{\mathsf{S}_{j-1}}(\mathsf{S}_j), \end{aligned}$$

where we have used that $f_{\mathsf{S}_{j-1}}(\mathsf{S}_j) \geq 0$ due to Reduction 2. Combining this observation with (2.11) yields

$$\begin{aligned} f(\mathsf{S}_t) \geq{}& \frac{\max_j c_j(v_i)}{kW} f(\mathrm{OPT}) \\ &+ \frac{(1-\max(1,\alpha)) \max_j c_j(v_i)}{kW} \sum_{j \in [t-1]} f_{\mathsf{S}_{j-1}}(\mathsf{S}_j) \end{aligned}$$

from which it follows that

$$\begin{aligned} \max(1,\alpha) f(\mathsf{S}_t) \geq{}& \frac{\max_j c_j(v_i)}{kW} f(\mathrm{OPT}) \\ &- \frac{\max(1,\alpha) \max_j c_j(v_i)}{kW} \sum_{j \in [t-1]} f_{\mathsf{S}_{j-1}}(\mathsf{S}_j). \end{aligned}$$

Defining $x_t = f(\mathsf{S}_t)/f(\mathrm{OPT})$ for all $t \in [r]$ we can write the inequality above as

$$\frac{kW \max(1,\alpha)}{\max_j c_j(v_t)} x_t + \max(1,\alpha) \sum_{i \in [t-1]} x_i \geq 1. \tag{2.12}$$

We conclude the proof by showing that any array of solutions $(x_1, \ldots, x_n)$ with coefficients $x_i \in [0,1]$ that fulfils the LP as in (2.12)

yields

$$\sum_{t\in[r]} x_t \geq \sum_{t\in[r]} \frac{\max_j c_j(v_t)}{kW\max(1,\alpha)} \prod_{i\in[t-1]} \left(1 - \frac{\max_j c_j(v_i)}{kW}\right). \tag{2.13}$$

In order to prove (2.13), since it holds $x_t \in [0,1]$ for all $t \in [r]$, we can simplify our setting, by studying the system

$$\frac{kW\max(1,\alpha)}{\max_j c_j(v_t)} x_t + \max(1,\alpha) \sum_{i\in[t-1]} x_i = 1. \tag{2.14}$$

This is due to the fact that the sum of the coefficients of any solution of (2.14) are upper-bounded by the sum of the coefficients of a solution of (2.13). We conclude the proof, by considering the following claim.

▶ **Claim 2.6.** Let $(x_1,\ldots,x_r)$ be a solution of the LP as in (2.14). Then it holds

$$x_t \geq \frac{\max_j c_j(v_t)}{kW\max(1,\alpha)} \prod_{i\in[t-1]} \left(1 - \frac{\max_j c_j(v_i)}{kW}\right),$$

for all $t \in [r]$. ◀

Proof. Define $c_t = \max_j c_j(v_t)/(W\max(1,\alpha))$ for all $t \in [r]$. Then the LP above can be written as

$$\frac{x_t}{c_t} = 1 - \max(1,\alpha) \sum_{i\in[t-1]} x_t.$$

By defining $y_t = x_t/c_t$ for all $t \in [r]$, we have that

$$y_t - y_{t-1} = -\max(1,\alpha)x_{t-1} = -\max(1,\alpha)c_{t-1}y_{t-1},$$

and we obtain the following recurrent relation $y_t + (\max(1,\alpha)c_t - 1)y_{t-1} = 0$, for all $t > 1$. This is a recurrent linear equation with solutions

$$y_t = \prod_{j\in[t-1]} (1 - c_j\max(1,\alpha)).$$

The claim follows, by substituting y_t and c_j in the equation above. ■

Hence, we have that it holds

$$\begin{aligned} f(\mathsf{S}_r) &= f(\mathrm{OPT}) \sum_t x_t \\ &\geq f(\mathrm{OPT}) \sum_t \frac{\max_j c_j(v_t)}{kW \max(1,\alpha)} \prod_{i\in[t-1]} \left(1 - \frac{\max_j c_j(v_i)}{kW}\right) \\ &\geq \frac{f(\mathrm{OPT})}{\max(1,\alpha)} \left(1 - \prod_{i\in[r]} \left(1 - \frac{\max_j c_j(v_i)}{kW}\right)\right) \\ &\geq \frac{f(\mathrm{OPT})}{\max(1,\alpha)} \left(1 - \exp\left\{-\sum_{i\in[r]} \frac{\max_j c_j(v_i)}{kW}\right\}\right), \end{aligned}$$

where the first inequality follows from Lemma 2.6; the second inequality follows via standard calculations; the last inequality follows because $1 - x \leq e^{-x}$. Consider an index ℓ such that $c_\ell(\mathsf{S}_r) > kW$. We have that it holds

$$\begin{aligned} f(\mathsf{S}_r) &\geq \frac{1}{\max(1,\alpha)} \left(1 - \exp\left\{-\max(1,\alpha) \sum_{i\in[r]} \frac{\max_j c_j(v_i)}{kW}\right\}\right) f(\mathrm{OPT}) \\ &\geq \frac{1}{\max(1,\alpha)} \left(1 - \exp\left\{-\max(1,\alpha) \sum_{i\in[r]} \frac{c_\ell(v_i)}{kW}\right\}\right) f(\mathrm{OPT}) \\ &\geq \frac{1}{\max(1,\alpha)} \left(1 - e^{-1/k}\right) f(\mathrm{OPT}). \end{aligned}$$

To conclude, denote with v^* the point with maximum f-value among the singletons, it follows that

$$\begin{aligned} \arg\max\{f(\mathsf{S}_{r-1}), f(v^*)\} &\geq (f(\mathsf{S}_{r-1}) + f(v^*))/2 \\ &\geq (f(\mathsf{S}_{r-1}) + f(v_r))/2 \\ &\geq f(\mathsf{S}_r)/2 \end{aligned}$$

$$\geq \frac{1}{2\max(1,\alpha)}\left(1-e^{-1/k}\right)f(\mathrm{OPT}),$$

where we have used submodularity. The claim follows. ■

3 Video Summarization

This chapter is based on a conference paper titled "Adaptive Sampling for Fast Constrained Maximization of Submodular Function", by Francesco Quinzan, and Vanja Doskoč, Andreas Göbel, Tobias Friedrich (Quinzan, Doskoc, et al. 2021).

Video summarization tasks consist of selecting a subset of frames that are representative of ta given video. These tasks can often be approached by maximizing submodular functions, under complex side constraints imposed by the underlying application. We develop an algorithm with poly-logarithmic adaptivity for non-monotone submodular maximization under general side constraints. The adaptive complexity of a problem is the minimal number of sequential rounds required to achieve the objective.
Our algorithm is suitable to maximize a non-monotone submodular function under a p-system side constraint, and it achieves a $(p + \mathcal{O}(\sqrt{p}))$-approximation for this problem, after only poly-logarithmic adaptive rounds and polynomial queries to the valuation oracle function. Furthermore, our algorithm achieves a $(p+\mathcal{O}(1))$-approximation when the given side constraint is a p-extendible system.
This algorithm yields an exponential speed-up, with respect to the adaptivity, over any other known constant-factor approximation algorithm for this problem. It also competes with previous known results in terms of the query complexity. We perform various experiments on real-world applications. We find that, in comparison with commonly used heuristics, our algorithm performs better on these instances.

3.1 Determinantal Point Processes and Video Summarization

3.1.1 Determinantal Point Processes

Determinantal Point Processes are probability distributions, first studied in classical physics (Macchi 1975). DPPs have received considerable attention in recent years, since they are commonly used in AI and Machine Learning (Elfeki et al. 2019; Gillenwater et al. 2012; Kang 2013; Kathuria et al. 2016; Kulesza and Taskar 2011; Mirzasoleiman, Jegelka, et al. 2018b; Mirzasoleiman, Karbasi, and Krause 2017). For a set of items $\mathsf{V} = \{1, \ldots, n\}$, a DPP defines a discrete probability distribution $\mathscr{P}$ over all subsets $\mathsf{S} \subseteq \mathsf{V}$ via the following requirement. Fix a set $\mathsf{A} \subseteq \mathsf{V}$. Then, for a set $\mathsf{S} \subseteq V$ chosen at random, i.e., $\mathsf{S} \sim \mathscr{P}$, it holds

$$\Pr(\mathsf{A} \subseteq \mathsf{S}) = \det(K_{\mathsf{A}}). \tag{3.1}$$

Here, $K \in \mathbb{R}^{n \times n}$ is a fixed matrix, called the *marginal kernel*, and K_{A} denotes the sub-matrix indexed by the points of A. Although well-posed, this definition does not give an explicit characterization of the probability $\Pr(\mathsf{S})$.

If the marginal kernel K is invertible, it is possible to define the corresponding DPP via *L-ensembles*. An L-ensemble over subsets S of V is a probability distribution of the form

$$\Pr(\mathsf{S}) = \frac{\det(L_{\mathsf{S}})}{\det(L + I)},$$

where $L \in \mathbb{R}^{n \times n}$ is a positive semidefinite matrix, L_{S} denotes the sub-matrix indexed by the points of S, and I is the $n \times n$-identity matrix. The relationship between L-ensembles and DPPs is clarified by the following result.

▶ **Theorem 3.1.** An L-ensemble is a DPP, and its marginal kernel is $K = L(L+I)^{-1} = I - (L+I)^{-1}$. ◀

For a proof of this theorem see, i.e., (Kulesza and Taskar 2012). Note that from Theorem 3.1 it follows that a DPP with an invertible marginal kernel is an L-ensemble with matrix $L = K(I-K)^{-1}$. Hence, the definition of L-ensembles is more restrictive than the definition of DPP as in (3.1). However, L-ensembles give an explicit characterization of the underlying probability distribution. In this context, we always define DPPs via L-ensembles.

3.1.2 Video Summarization

Video summarization consists of the following task: Given a video, choose a subset of frames that gives a descriptive overview of the video. DPPs are commonly used for sampling subset of frames for video summarization, since they maximize diversity (Mirzasoleiman, Jegelka, et al. 2018b; Mirzasoleiman, Karbasi, and Krause 2017).

When using DPPs for Video Summarization, a key challenge lies in the definition of a suitable marginal kernel. We follow (Gong et al. 2014) to this end, although other methods have been proposed in recent years (Kulesza and Taskar 2010). For each frame i, we compute a feature vector $\mathbf{f}_i$ consisting of visual features, such as color and SIFT, and qualitative information, such as size, colorfulness and luminosity (Kulesza and Taskar 2011). We define the matrix L of the corresponding L-ensemble as

$$L_{i,j} \coloneqq z_i^T W^T W z_j,$$

with $z_i = \tanh(U\mathbf{f}_i)$, and U, W parameter arrays. We learn the parameters U and W using a neural network.

In order to create descriptive summaries, several additional side constraints are imposed on the solution space. A common example is an upper-bound on the maximum number of frames that can be

selected to construct a summary. Another example is the use of *partition matroids*. Here, each video is divided into segments, and only ℓ_j frames in each segment j are selected. Sometimes, more complex constraints are employed. For instance, (Moran Feldman, Karbasi, et al. 2018; Mirzasoleiman, Jegelka, et al. 2018b) use a face-recognition tool to define side constraints. With this tool, they identify actors in each frame, and select summaries containing at most k frames showing each actor.

3.2 Problem Formulation

In this section, we present a high-level description of Video Summarization, which is then used to derive approximation guarantees for our algorithm. We search for video summaries by selecting subsets S that maximize diversity. In order to find a diverse set of frames, we search or a set that has the highest probability of being sampled, according to a DPP with marginal kernel defined as in Section 3.1.1. This task than consists of maximizing the function

$$f(\mathsf{S}) = \frac{\det(L_{\mathsf{S}})}{\det(L+I)}, \tag{3.2}$$

under additional side constraints. Here, the matrix L is defined as in Theorem 3.1. It is well-known that the function $f(\cdot)$ as in (3.2) is log-submodular (Kulesza and Taskar 2012).[5] Hence, Video Summarization can be formulated as a constrained submodular maximization problem. We study this problem under two types of side constraints, to capture several real-world applications.

5 A function $f : 2^{\mathsf{V}} \to \mathbb{R}_{>0}$ is log-submodular if the function $\log f(\cdot)$ is a submodular function.

3.2.1 p-Systems Side Constraints

We study the problem of maximizing a submodular function under additional side constraints, defined as a p-system side constraint. As discussed, i.e., in (A. Gupta et al. 2010; Mirzasoleiman, Badanidiyuru, and Karbasi 2016), these constraints are very general, and they commonly arise in Video Summarization.

Given a collection of feasible solutions $\mathscr{I}$ over a ground set V and a set $T \subseteq V$, we denote with $\mathscr{I}|_T$ a collection consisting of all sets $S \subseteq T$ that are feasible in $\mathscr{I}$. Furthermore, a base for $\mathscr{I}$ is any maximum feasible set $U \in \mathscr{I}$. We define p-systems as follows.

▶ **Definition 3.1.** A p-system $\mathscr{I}$ over a ground set V is a collection of subsets of V fulfilling the following three axioms:

- $\emptyset \in \mathscr{I}$;
- for any two sets $S \subseteq \Omega \subseteq V$, if $\Omega \in \mathscr{I}$ then $S \in \mathscr{I}$;
- for any set $T \subseteq V$ and any bases $S, U \in \mathscr{I}|_T$ it holds $|S| \leq p|U|$.

◀

The second defining axiom is referred to as subset-closure or downward-closed property. With this notation, we study the following problem.

Problem 3. *Given a submodular function* $f : 2^V \to \mathbb{R}_{\geq 0}$ *and a* p*-system* $\mathscr{I}$*, find a set* $S \subseteq V$ *maximizing* $f(S)$ *such that* $S \in \mathscr{I}$*.*

3.2.2 p-Extendable Systems Side Constraints

We also consider a family of side constraints of intermediate generality, commonly referred to as p-extendable systems. These side constraints are strictly less general than p-systems, but they capture various real-world scenarios. We study p-extendable systems

because they admit algorithms that perform very well in practise. p-extendable systems were first defined by (Mestre 2006), as follows.

▶ **Definition 3.2.** A p-extendable system $\mathscr{I}$ over a ground set V is a p-system, that fulfills the following additional axiom: for every pair of sets $\mathsf{S}, \Omega \in \mathscr{I}$ with $\mathsf{S} \subset \Omega$, and for every element $e \notin \mathsf{S}$, there exists a set $\mathsf{U} \subseteq \Omega \setminus \mathsf{S}$ of size $|\mathsf{U}| \leq p$ such that $\Omega \setminus \mathsf{U} \cup \{e\} \in \mathscr{I}$. ◀

Although less general than p-systems, side constraints as in Definition 3.2 generalize many interesting combinatorial structures, such as matroid intersections and p-matchoids (Mestre 2006). Hence, we study the following problem.

Problem 4. *Given a submodular function* $f : 2^{\mathsf{V}} \rightarrow \mathbb{R}_{\geq 0}$ *and a* p*-extendable system* $\mathscr{I}$*, find a set* $\mathsf{S} \subseteq \mathsf{V}$ *maximizing* $f(\mathsf{S})$ *such that* $\mathsf{S} \in \mathscr{I}$*.*

3.2.3 The Adaptivity as a Computational Model

In this work, we assume oracle access to the function $f(\mathsf{S})$ as in (3.2), and we also assume access to the side constraint structure via an oracle. Standard oracle models for the side constraints are: the *independence oracle*, which takes as input a set and returns whether that set is a feasible solution; the *rank oracle*, that returns the maximum cardinality of any feasible solution contained in a given input set; and the *span oracle*, which for an input set S and a point $\{e\}$ it returns whether or not $\mathsf{S} \cup \{e\}$ has a higher rank than S (Chekuri and Quanrud 2019b). Here, we assume access to the independence oracle, which is the most general oracle model of the three.

Oracle valuations via $f(\mathsf{S})$ are typically time consuming, since they require to compute the determinant of a potentially large matrix, and evaluate feasibility with respect to complex side constraints. Hence, a standard approach for the design of optimizers suitable for Video Summarization consists of minimizing the total number of calls to $f(\mathsf{S})$. This approach has lead to the design of efficient algorithms

in recent years (Moran Feldman, Harshaw, et al. 2017; Moran Feldman, Harshaw, et al. 2020; Mirzasoleiman, Badanidiyuru, and Karbasi 2016). A major drawback of these algorithms, however, is that oracle calls are *highly sequential*, and they cannot be parallelized efficiently.

In contrast to previous related work, we propose a new algorithm for Problem 3 and Problem 4, that allows for an *efficient prallelization* of the oracle queries. To this end, we consider a different computational model, and use the adaptivity as a performance measure (Balkanski and Singer 2018). The adaptivity refers to the number of sequential rounds, wherein polynomial number of parallel queries are made in each round. Formally, the adaptivity is defined as follows.

▶ **Definition 3.3 (Adaptivity, (Balkanski and Singer 2018)).** Given an oracle f, an algorithm is r-adaptive if every query q to the oracle f occurs at a round $i \in [r]$ such that q is independent of the answers $f(q')$ to all other queries q' at round i. ◀

The adaptivity is closely related to other classic models such as Parallel Random Access Machines (PRAM). The PRAM model consists of a set of processors, that communicate via a single shared memory, and a memory access unit. The adaptivity extend to PRAM via the notion of depth. The depth is the number of parallel steps in an algorithm or the longest chain of dependencies. However, the PRAM model assumes that the input is loaded in memory, whereas the adaptive complexity model only assumes access to an oracle function.

3.3 Related Work

Several algorithms have been discovered, to maximize a monotone submodular function under general side constraints such as p-systems and multiple knapsacks (Badanidiyuru and Vondrák 2014; Chekuri and Pál 2005). These algorithms include streaming algorithms (Badanidiyuru, Mirzasoleiman, et al. 2014; Chakrabarti and Kale 2015; Chekuri,

S. Gupta, et al. 2015), centralized algorithms (Badanidiyuru and Vondrák 2014; Mirzasoleiman, Badanidiyuru, Karbasi, et al. 2015), and distributed algorithms (Kumar et al. 2015; Mirzasoleiman, Karbasi, Sarkar, et al. 2013).

Many algorithms have been proposed to maximize *non-monotone* submodular functions under a variety of constraints (Buchbinder, Moran Feldman, Naor, et al. 2015; Chekuri, Vondrák, et al. 2014; Feige, Mirrokni, et al. 2011; Moran Feldman, Naor, et al. 2011b; A. Gupta et al. 2010; Lee et al. 2009). These algorithms yield good approximation guarantees, but their run time is polynomial in the number of samples, and polynomial in the number of additional side constraints. Recently, algorithms were discovered to maximize a non-monotone submodular function under very general side constraints (Moran Feldman, Harshaw, et al. 2017; Mirzasoleiman, Badanidiyuru, and Karbasi 2016). These constant-factor approximation algorithms scale polynomially in the number of data-points, but also in the number of additional side constraints.

In some cases, approximation algorithms do not exhibit increasingly worse run time in the number of constraints. This is the case when maximizing a submodular function under p*-extendible systems* or p*-matchoid* side constraints (Chekuri and Quanrud 2019a; Moran Feldman, Harshaw, et al. 2017). These side constraints are strictly less general than those studied by Mirzasoleiman, Badanidiyuru, and Karbasi 2016, but they are general enough to capture a variety of interesting applications.

Our Contribution. We develop the first algorithm with poly-log adaptivity suitable to maximize a non-monotone submodular function under a p-system side constraint and a p-extendable system. In contrast to all previous algorithms with low adaptivity, our algorithm only requires access to the independence oracle for the side constraints. This algorithm achieves strong approximation guarantees and run time, competing with known algorithms for this problem

Algorithm 3: RAND-SEQUENCE

```
A ← ∅; while X ≠ ∅ do
    sort the points {x_i}_i = X randomly;
    η ← max{j : S ∪ A ∪ {x_i}_{i≤j} ∈ 𝒥};
    A ← A ∪ {x_1, ..., x_η};
    X ← {e ∈ X \ (S ∪ A) : S ∪ A ∪ e ∈ 𝒥};
return A;
```

(Moran Feldman, Harshaw, et al. 2017; Moran Feldman, Harshaw, et al. 2020; A. Gupta et al. 2010; Mirzasoleiman, Badanidiyuru, and Karbasi 2016).[6]

3.4 A Fast Algorithm for Video Summarization

Our method consists of three parts (see Algorithms 3-5). We call these parts the RAND-SEQUENCE, the RAND-SAMPLING, and the REP-SAMPLING. These algorithms also uses the BINARY-SEARCH and UNIF-SAMPLING sub-routines. The following is a description of each algorithm and sub-routine.

The RAND-SEQUENCE sub-routine. This sub-routine is based on the work of (Karp et al. 1988). Given as input a ground set X, a current solution S, and a p-system $\mathscr{I}$, this algorithm finds a random set A such that $S \cup A$ is a base for $\mathscr{I}$.

6 The Parallel Greedy algorithm by Moran Feldman, Harshaw, et al. 2020 requires access to the rank oracle for the underlying p-matchoid system. This oracle is strictly less general then the independence oracle required by our algorithm.

Algorithm 4: RAND-SAMPLING

$\mathsf{S} \leftarrow \emptyset$;
$\mathsf{X} \leftarrow \arg\max_e \{f(e) \colon e \in \mathsf{V} \wedge e \in \mathscr{I}\}$;
$\delta \leftarrow f(\mathsf{X}), \delta_0 \leftarrow \lambda f(\mathsf{X})$;
while $\delta \geq \delta_0$ **do**
  while $\mathsf{X} \neq \emptyset$ **do**
    $\{a_j\}_{j \in J} \leftarrow$ RAND-SEQUENCE$(\mathsf{X}, \mathsf{S}, \mathscr{I})$;
    $\eta \leftarrow$ BINARY-SEARCH$(\mathsf{J}, \min\{j \in \mathsf{J} \colon \left|\mathsf{X}_j\right| < (1-\varepsilon)|\mathsf{X}|\})$ with $\mathsf{X}_j = \{e \in \mathsf{X} \colon f_{\mathsf{S} \cup \{a_1, \ldots, a_{j-1}\}}(e) \geq \delta \wedge \mathsf{S} \cup \{a_1, \ldots, a_{j-1}\} \cup e \in \mathscr{I}\}$;
    $\mathsf{A} \leftarrow$ UNIF-SAMPLING$(\{a_1, \ldots, a_{\eta-1}\}, \varphi_1)$;
    $\mathsf{X} \leftarrow \mathsf{X}_\eta$; $\mathsf{S} \leftarrow \mathsf{S} \cup \mathsf{A}$;
  $\delta \leftarrow (1-\varepsilon)\delta$;
  $\mathsf{X} \leftarrow \{e \in \mathsf{V} \colon f_{\mathsf{S}}(e) \geq \delta \wedge \mathsf{S} \cup e \in \mathscr{I}\}$;
return S;

The RAND-SAMPLING algorithm. This algorithms generalizes a sampling algorithm proposed in (Balkanski, Rubinstein, et al. 2019) to non-monotone submodular maximization. This algorithm requires as input an oracle function f, a ground set V, a p-system or p-extendable system $\mathscr{I}$, and parameters $\lambda, \varepsilon, \varphi_1$. The parameter λ determines the total number of iterations for the RAND-SAMPLING, the parameter ε determines the rate with which the variable δ decreases, whereas φ_1 determines the distribution for the UNIF-SAMPLING subroutine. For a constant δ, points are added to the current solution yielding a marginal contribution upper-bounded by δ. Note that at each adaptive step, the RAND-SAMPLING uses the BINARY-SEARCH and the UNIF-SAMPLING sub-routine. If Algorithm 4 reaches an iteration with $\mathsf{X} = \emptyset$, then it decreases the value of δ so that points with lower marginal contribution can be added to the current solution.

Algorithm 5: REP-SAMPLING

$\lambda \leftarrow \varepsilon(p+1)/(r+1)$;
for $j \leq m$ *iterations* **do**
  $\Omega_j \leftarrow$ RAND-SAMPLING$(f, \mathsf{V}, \mathscr{I}, \lambda, \varepsilon, \varphi_1)$;
  $\Lambda_j \leftarrow$ UNIF-SAMPLING(Ω_j, φ_2);
  $\mathsf{V} \leftarrow \mathsf{V} \setminus \Omega_j$;
return $\arg\max_j \{f(\Omega_j), f(\Lambda_j)\}$;

The BINARY-SEARCH sub-routine. This sub-routine is just the standard binary search algorithm. It is used to locate an index η such that $\eta = \min\{j : |\mathsf{X}_j| \leq (1-\varepsilon)|\mathsf{X}|\}$, with X_j, where the index j spans over the set J. This sub-routine uses the fact that, due to submodularity, it holds $|\mathsf{X}_j| \leq |\mathsf{X}_{j+1}|$ for all $j \in \mathsf{J}$.

The UNIF-SAMPLING sub-routine. For a given input set and probability φ, this algorithm samples points of the input set independently, with probability φ.

The REP-SAMPLING algorithm. This algorithm requires as input an oracle function f a ground set V, a p-system or p-extendable system $\mathscr{I}$ and parameters λ, m, ε and φ_1, φ_2. At each step, the REP-SAMPLING calls Algorithm 4 to find a partial solution Ω_j. Then, Algorithm 5 samples a subset of Ω_j, where each point is drawn independently with probability φ_2. Afterwards, the REP-SAMPLING removes all points of Ω_j from the ground set, and it runs the REP-SAMPLING on the resulting ground set. This procedure is iterated m times.

3.5 Run Time Analysis for p-Systems

3.5.1 Overview of the Main Results

In this section, we discuss theoretical run time analysis results for Problem 3. All results concerning the approximation guarantee use the following pivotal lemma.

▶ **Lemma 3.1.** Define the sets Ω_j and Λ_j as in Algorithm 5, and denote with V_j the ground set for Algorithm 4, during the j-th iteration of the for-loop lines 2-5 of Algorithm 5. Furthermore, let δ_0, δ, and λ be as in Algorithm 4 and Algorithm 5. Then, for parameters $\varphi_1 = 1$ and $\varphi_2 = 1/2$ it holds

$$(p+1)\mathbb{E}\left[\, f(\Omega_j) \,\right] + \lambda r \mathbb{E}\left[\, f(\Omega_j) \,\right] \geq (1-\varepsilon)^2 \mathbb{E}\left[\, f(\Omega_j \cup (\mathrm{OPT} \cap V_j) \,\right],$$

for all sets Ω_j. ◀

On a high level, we prove that the constant δ in Algorithm 4 is an upper-bound for the best possible improvement up to a multiplicative factor, and that the marginal contribution of any point added to the current solution does not exceed δ in expected value, up to a multiplicative constant. We then combine this fact with the defining properties of the p-system to prove the claim, which holds for non-monotone functions. The proof of Lemma 3.1 requires additional results, and it is deferred to Section 3.5.2.

Using Lemma 3.1, we can prove strong approximation guarantees for Algorithm 5. These guarantees follow from the following general theorem.

▶ **Theorem 3.2.** Fix constants $\varepsilon \in (0,1)$, $m \geq 2$, $\varphi_1 = 1$, and $\varphi_2 = 1/2$. Denote with Ω^* the output of Algorithm 5. Then,

$$\mathrm{OPT} \leq m\left(\frac{(1+\varepsilon)(p+1)}{(1-\varepsilon)^2(m-1)} + 2\right)\mathbb{E}\left[\, f(\Omega^*) \,\right].$$

◀

A proof of this theorem is given in Section 3.5.3. We estimate the number of adaptive rounds until Algorithm 5 reaches the desired approximation guarantee. The following lemma holds.

▶ **Lemma 3.2.** Fix constants $\varepsilon \in (0,1)$, $\varphi_1, \varphi_2 \in [0,1]$ and $m \geq 0$. Then Algorithm 5 terminates after $\mathcal{O}\left(\frac{m}{\varepsilon^2}\log\left(\frac{r}{p\varepsilon}\right)\log r \log n\right)$ rounds of adaptivity. Furthermore, the query complexity of Algorithm 5 is upper-bounded as $\mathcal{O}\left(\frac{mn}{\varepsilon^2}\log\left(\frac{r}{p\varepsilon}\right)\log r \log n\right)$. ◀

A proof of this result is given in Section 3.5.4. The following lemma follows from Theorem 3.2 and Lemma 3.2.

▶ **Lemma 3.3.** Fix a constant $\varepsilon \in (0,1)$, and define parameters $m = 1 + \lceil\sqrt{(p+1)/2}\rceil$, $\varphi_1 = 1$, and $\varphi_2 = 1/2$. Denote with Ω^* the optimal solution found by Algorithm 5. Then,

$$\texttt{OPT} \leq \frac{1-\varepsilon}{(1-\varepsilon)^2}\left(p + 2\sqrt{2(p+1)} + 5\right)\mathbb{E}\left[\, f(\Omega^*)\,\right].$$

Furthermore, with this parameter choice Algorithm 5 terminates after $\mathcal{O}\left(\frac{\sqrt{p}}{\varepsilon^2}\log n \log\left(\frac{r}{p\varepsilon}\right)\log r\right)$ rounds of adaptivity.
Its query complexity is $\mathcal{O}\left(\frac{\sqrt{p}n}{\varepsilon^2}\log n \log\left(\frac{r}{p\varepsilon}\right)\log r\right)$. ◀

A proof is given in Section 3.5.5. We remark that there exists an algorithm with constant adaptivity for unconstrained non-monotone submodular maximization that achieves an approximation guarantee arbitrarily close to 1/2 (Chen et al. 2019). Using this algorithm as a sub-routine, in line 4 of Algorithm 5, yields a constant-factor improvement over the approximation guarantee of Lemma 3.3, without affecting the upper-bound on the adaptivity. However, this algorithm requires access to a continuous extension of the value oracle f, whereas Algorithm 5 only requires access to f.

3.5.2 Proof of Lemma 3.1

In this section, we prove Lemma 3.1, which is useful to prove the desired approximation guarantee. This lemma is restated here.

▶ **Lemma 3.1.** Define the sets Ω_j and Λ_j as in Algorithm 5, and denote with V_j the ground set for Algorithm 4, during the j-th iteration of the for-loop lines 2-5 of Algorithm 5. Furthermore, let δ_0, δ, and λ be as in Algorithm 4 and Algorithm 5. Then, for parameters $\varphi_1 = 1$ and $\varphi_2 = 1/2$ it holds

$$(p+1)\mathbb{E}\left[\, f(\Omega_j) \,\right] + \lambda r \mathbb{E}\left[\, f(\Omega_j) \,\right] \geq (1-\varepsilon)^2 \mathbb{E}\left[\, f(\Omega_j \cup (\mathrm{OPT} \cap \mathsf{V}_j) \,\right],$$

for all sets Ω_j. ◀

Throughout this section, we use the notation introduced in Lemma 3.1. We also always assume that parameters φ_1, φ_2 are set as $\varphi_1 = 1$ and $\varphi_2 = 1/2$. In order to prove Lemma 3.1, we need the following technical proposition.

▶ **Proposition 3.1 (Proposition 2.2 in (M. Fisher et al. 1978)).** Let $\{x_1, \ldots, x_m\}$,
$\{y_1, \ldots, y_m\}$ be two sequences of non-negative real numbers. Suppose that it holds

$$\sum_{j=1}^{i} x_i \leq i,$$

for all $i \in [m]$ and $y_i \geq y_{i+1}$ for all $i \in [m-1]$. Then

$$\sum_{j=1}^{m} y_i \geq \sum_{j=1}^{m} x_i y_i.$$

◀

In addition to Proposition 3.1, we prove additional lemma, which are used in the proof of Lemma 3.1. We first prove the following result, which follows from Lemma 2 in (Balkanski, Rubinstein, et al. 2019).

▶ **Lemma 3.4.** Fix an index j, and let δ, Ω be as in Algorithm 4, as it runs over the set V_j. Then it holds

$$\delta \geq (1-\varepsilon) \sup_{\{e \in \mathsf{V}_j \setminus \Omega \colon \Omega \cup e \in \mathscr{I}\}} f_\Omega(e).$$

◀

Proof. We prove the claim by induction on the iterations of Algorithm 4. The base case trivially holds, due to the definition of δ. Suppose that at some point the current solution Ω is updated to $\Omega \cup \{a_1, \ldots, a_\eta\}$. We have that

$$\begin{aligned}
&\sup_{\{e \in V_j \setminus (\Omega \cup \{a_1,\ldots,a_\eta\}) \colon \Omega \cup \{a_1,\ldots,a_\eta\} \cup e \in \mathscr{I}\}} f_{\Omega \cup \{a_1,\ldots,a_\eta\}}(e) \\
&\leq \sup_{\{e \in V_j \setminus (\Omega \cup \{a_1,\ldots,a_\eta\}) \colon \Omega \cup \{a_1,\ldots,a_\eta\} \cup e \in \mathscr{I}\}} f_\Omega(e) \\
&\leq \sup_{\{e \in V_j \setminus \Omega \colon \Omega \cup e \in \mathscr{I}\}} f_\Omega(e) \\
&\leq \delta,
\end{aligned}$$

where the first inequality uses the submodularity property of f; the second inequality holds since $\{e \in \mathsf{V}_j \setminus (\Omega \cup \{a_1, \ldots, a_\eta\}) \colon \Omega \cup \{a_1, \ldots, a_\eta\} \cup e \in \mathscr{I}\} \subseteq \{e \in \mathsf{V}_j \setminus \Omega \colon \Omega \cup e \in \mathscr{I}\}$; the last inequality follows due to the inductive hypothesis.

We now show that the claim holds when δ is updated to $\delta' = (1-\varepsilon)\delta$. At this point, it holds $\mathsf{X} = \emptyset$, and each point $e^* \in \mathsf{V}_j \setminus \Omega$ such that $\Omega \cup e^* \in \mathscr{I}$ was discarded during a previous iteration. Denote with Ω' the solution at the iteration when e^* was discarded, and let $\{a_1, \ldots, a_\eta\}$ be the next set of points added to Ω'. Since e^* was discarded, then one of the following two conditions must hold:

1. $\Omega' \cup \{a_1, \ldots, a_\eta\} \cup e^* \notin \mathscr{I}$;
2. $f_{\Omega' \cup \{a_1,\ldots,a_\eta\}}(e^*) \leq \delta$.

In the first case, due to the downward-closed property of $\mathscr{I}$ it holds $\Omega \cup e^* \notin \mathscr{I}$, which contradicts the definition of e^*. In the latter case it holds

$$f_{\Omega}(e^*) \leq f_{\Omega' \cup \{a_1,\dots,a_\eta\}}(e^*) \leq \delta = \frac{\delta'}{1-\varepsilon}$$

where the first inequality uses the fact that f is submodular. The claim follows. ■

We also need an additional lemma, to prove that Algorithm 5 gives a constant-factor approximation for Problem 3.

▶ **Lemma 3.5.** Fix an index j, and let δ, X be as in Algorithm 4, as it optimizes the function f over the set V_j. Denote with $\{a_i\}_i$ the points of Ω_j, sorted as they were added to it. It holds $\mathbb{E}_{a_i}\left[\, f_{\{a_1,\dots,a_{i-1}\}}(a_i) \,\right] \geq (1-\varepsilon)\delta$. ◀

Proof. First, suppose that the constant δ is updated after the point a_{i-1} is added to the current solution. In that case, by definition of the set X, every point $e \in \mathsf{X} \setminus \{a_1,\dots,a_{i-1}\}$ such that $\{a_1,\dots,a_{i-1}\} \cup e \in \mathscr{I}$ yields $f_{\{a_1,\dots,a_{i-1}\}}(e) > \delta$. Hence, the claim holds.

Suppose now that the constant δ is not updated after the point a_{i-1} is added to the current solution. Define the set $\mathsf{X}_{i-1}^{\mathscr{I}} := \{e \in \mathsf{X} \colon \{a_1,\dots,a_{i-1}\} \cup e \in \mathscr{I}\}$. We first claim that the point a_i is chosen uniformly at random over the set $\mathsf{X}_{i-1}^{\mathscr{I}}$. In fact, if a_i is added to the current solution, then there exists a set $\{x_1,\dots,x_\eta\}$ as in Line 4 of Algorithm 3 such that $a_i \in \{x_1,\dots,x_\eta\}$. Let $j \leq \eta$ be an index such that $a_i = x_j$. Due to the downward-closed property of $\mathscr{I}$ it holds $\{e \colon A \cup \{x_1,\dots,x_j,e\} \in \mathscr{I}\} \subseteq \mathsf{X} \setminus \{x_1,\dots,x_j\}$. Hence, a_i is chosen uniformly at random among all points e such that $\mathsf{A} \cup \{x_1,\dots,x_{j-1},e\} = \{a_1,\dots,a_{j-1},e\} \in \mathscr{I}$.

Then

$$\mathsf{Pr}_{a_i}\left(f_{\{a_1,\dots,a_{i-1}\}}(a_i) > \delta\right) \geq \frac{|\mathsf{X}_{i-1}|}{|\mathsf{X}_{i-1}^{\mathscr{I}}|} \geq \frac{|\mathsf{X}_{i-1}|}{|\mathsf{X}|},$$

where we have used that $\mathsf{X}_{i-1}^{\mathscr{I}} \subseteq \mathsf{X}_{i-1}$ and that the point e is chosen uniformly at random over the set $\mathsf{X}_{i-1}^{\mathscr{I}}$.

We now prove that $|\mathsf{X}_{i-1}|/|\mathsf{X}| \geq (1-\varepsilon)$. To this end, we first note that $|\mathsf{X}_i| \geq |\mathsf{X}_{i+1}|$, for all indices i. Fix a point $e \in \mathsf{X}_{i+1}$. Then, this point yields $\{a_1,\ldots,a_i\} \cup a \in \mathscr{I}$ and $f_{\{a_1,\ldots,a_i\}}(a) \geq \delta$. By the downward-closed property of $\mathscr{I}$ we get $\{a_1,\ldots,a_{i-1}\} \cup a \in \mathscr{I}$, and by submodularity we get $f_{\{a_1,\ldots,a_{i-1}\}}(a) \geq \delta$. Hence, $\mathsf{X}_{i+1} \subseteq \mathsf{X}_i$ and $|\mathsf{X}_i| \geq |\mathsf{X}_{i+1}|$ as claimed. It follows that the BINARY-SEARCH sub-routine of Algorithm 4 terminates whenever $\eta = \min\{i\colon |\mathsf{X}_i| \leq (1-\varepsilon)|\mathsf{X}|\}$, which implies that $|\mathsf{X}_{i-1}| > (1-\varepsilon)|\mathsf{X}|$. The claim follows since it holds $\mathbb{E}_{a_i}\left[\, f_{\{a_1,\ldots,a_{i-1}\}}(a_i)\,\right] \geq \mathsf{Pr}_{a_i}\big(f_{\{a_1,\ldots,a_{i-1}\}}(a_i) > \delta\big)\delta$. ■

Using Proposition 3.1, Lemma 3.4-3.5, we can prove Lemma 3.1. This proof uses ideas proposed by (A. Gupta et al. 2010).

Proof of Lemma 3.1. Denote with $\{a_i\}_i$ the points of Ω_j in the order that they were added to Ω_j. Define the set

$$\mathsf{W}_{\delta_0} \coloneqq \{e \in \textsc{opt} \cap \mathsf{V}_j\colon f_{\Omega_j}(e) \geq \delta_0\}.$$

Note that this set consists of all points of $\textsc{opt} \cap \mathsf{V}_j$ such that their marginal contribution is above δ_0, when added to the current solution. First, fix a set Ω_j, and suppose that $f_{\{a_1,\ldots,a_{i-1}\}}(a_i) \geq \delta$ for all indices i. Define the sets $\mathsf{A}_i \coloneqq \big\{e \in \mathsf{W}_{\delta_0} \setminus \{a_1,\ldots,a_i\}\colon \{a_1,\ldots,a_i\} \cup e \in \mathscr{I}\big\}$. Since the system $\mathscr{I}$ is downward-closed, then $\mathsf{A}_i \subseteq \mathsf{A}_{i-1}$. Define the sets $\mathsf{D}_i \coloneqq \mathsf{A}_{i-1} \setminus \mathsf{A}_i$. Note that these sets consist of all points in W_{δ_0} that yield a feasible solution when added to $\{a_1,\ldots,a_{i-1}\}$, but that violate side constraints when added to $\{a_1,\ldots,a_i\}$.

We now claim that the set $\{a_1,\ldots,a_i\}$ is a maximal independent set for

$$\{a_1,\ldots,a_i\} \cup (\mathsf{D}_1 \cup \cdots \cup \mathsf{D}_i) = \{a_1,\ldots,a_i\} \cup \big(\mathsf{W}_{\delta_0} \setminus \mathsf{A}_i\big).$$

To this end, note that the set $\{a_1,\ldots,a_i\}$ is independent by definition, and that any point $e \in \big(\mathsf{W}_{\delta_0} \setminus \mathsf{A}_i\big) \setminus \{a_1,\ldots,a_i\}$ is such that $\{a_1,\ldots,a_i\} \cup$

$e \notin \mathscr{I}$. Hence $\{a_1, \ldots, a_i\}$ is maximal as claimed. Note also that $\mathsf{D}_1 \cup \cdots \cup \mathsf{D}_i \subseteq \mathsf{W}_{\delta_0}$ is an independent set, due to the subset-closure of $\mathscr{I}$. Since $\mathscr{I}$ is a p-system, then it holds

$$|\mathsf{D}_1| + \cdots + |\mathsf{D}_i| = |\mathsf{D}_1 \cup \cdots \cup \mathsf{D}_i| \leq p|\{a_1, \ldots, a_i\}| = pi. \tag{3.3}$$

Furthermore, using submodularity and Lemma 3.4 it holds

$$\begin{aligned} |\mathsf{D}_i|\delta &\geq (1-\varepsilon)|\mathsf{D}_i| \sup_e f_{\{a_1,\ldots,a_{i-1}\}}(e) \\ &\geq (1-\varepsilon) f_{\{a_1,\ldots,a_{i-1}\}}(\mathsf{D}_i) \\ &\geq (1-\varepsilon) f_{\Omega_j}(\mathsf{D}_i), \end{aligned}$$

where the last inequality follows from submodularity. Combining this with (3.3) and Proposition 3.1, we get

$$\begin{aligned} p \sum_i f_{\{a_1,\ldots,a_{i-1}\}}(a_i) &\geq \sum_i |\mathsf{D}_i|\delta \\ &\geq (1-\varepsilon) \sum_i f_{\Omega_j}(\mathsf{D}_i) \\ &\geq f_{\Omega_j}\big(\mathsf{W}_{\delta_0}\big), \end{aligned}$$

where the last inequality uses submodularity. Hence, rearranging yields $pf(\Omega_j) \geq f(\Omega_j \cup \mathsf{W}_{\delta_0}) - f(\Omega_j) = f_{\Omega_j}(\mathsf{W}_{\delta_0})$. If we unfix the set Ω_j and take the expected value, and using Lemma 3.5 we get

$$p\mathbb{E}\big[\, f(\Omega_j) \,\big] \geq (1-\varepsilon)^2 \mathbb{E}\Big[\, f_{\Omega_j}(\mathsf{W}_{\delta_0}) \,\Big]. \tag{3.4}$$

Using Lemma 3.5 again, we have that $\mathbb{E}_{a_i}\big[\, f_{\{a_1,\ldots,a_{i-1}\}}(a_i) \,\big] \geq 0$, for all points a_i added to Ω_j. It follows that $\mathbb{E}\big[\, f(\Omega_j) \,\big] \geq \mathbb{E}\big[\, f(a_0) \,\big]$, with a_0 the first point added to Ω_j. Then, from the definition of δ_0 it follows that $\lambda\mathbb{E}\big[\, f(\Omega_j) \,\big] \geq \lambda\mathbb{E}\big[\, f(a_0) \,\big] = \mathbb{E}[\, \delta_0 \,]$. Hence, using submodularity

and the linearity of the expected value, we get

$$\lambda r\mathbb{E}\left[\, f(\Omega_j)\,\right] \geq r\mathbb{E}\left[\,\delta_0\,\right] \geq \mathbb{E}\left[\sum_{e\in(\mathrm{OPT}\cap \mathsf{V}_j)\setminus \mathsf{W}_{\delta_0}} f_{\Omega_j}(e)\right] \geq \mathbb{E}\Big[\, f_{\Omega_j}((\mathrm{OPT}\cap \mathsf{V}_j)\setminus \mathsf{W}_{\delta_0})\,\Big], \tag{3.5}$$

where we have used submodularity.

Combining (3.4) with (3.5) and using submodularity again we get $p\mathbb{E}\left[\, f(\Omega_j)\,\right] + \lambda r\mathbb{E}\left[\, f(\Omega_j)\,\right] \geq (1-\varepsilon)^2\mathbb{E}\Big[\, f_{\Omega_j}(\mathrm{OPT}\cap \mathsf{V}_j)\,\Big]$. The claim follows by rearranging. ■

3.5.3 Proof of Theorem 3.2

In this section, we prove the following theorem.

▶ **Theorem 3.2.** Fix constants $\varepsilon \in (0,1)$, $m \geq 2$, $\varphi_1 = 1$, and $\varphi_2 = 1/2$. Denote with Ω^* the output of Algorithm 5. Then,

$$\mathtt{OPT} \leq m\left(\frac{(1+\varepsilon)(p+1)}{(1-\varepsilon)^2(m-1)} + 2\right)\mathbb{E}\left[\, f(\Omega^*)\,\right].$$

◀

In our analysis we consider the following well-known result.

▶ **Lemma 3.6 (Theorem 2.1 in (Feige, Mirrokni, et al. 2011)).** Let $\mathsf{U} \subseteq \mathsf{V}$ be a set chosen uniformly at random. Then it holds $\mathbb{E}\left[\, f(\mathsf{U})\,\right] \geq f(\mathsf{O})/4$, with $\mathsf{O} \subseteq \mathsf{V}$ the subset attaining the maximum f-value. ◀

Furthermore, we also consider the following properties of submodular functions.

▶ **Lemma 3.7 (Lemma 10 in (Moran Feldman, Harshaw, et al. 2017)).** For any fixed m-tuple of mutually disjoint sets Ω_j it holds $(m-1)\mathtt{OPT} \leq \sum_{j\leq m} f(\Omega_j \cup \mathrm{OPT})$. ◀

▶ **Lemma 3.8 (Lemma 11 in (Moran Feldman, Harshaw, et al. 2017)).** Let $f\colon 2^{\mathsf{V}} \to \mathbb{R}_{\geq 0}$ be a non-negative submodular function. For every three sets $\mathsf{A}, \mathsf{B}, \mathsf{C} \subseteq \mathsf{V}$ it holds $f(\mathsf{A} \cup (\mathsf{B} \cap \mathsf{C})) + f(\mathsf{B} \setminus \mathsf{C}) \geq f(\mathsf{A} \cup \mathsf{B})$. ◀

Using Lemma 3.6-3.8 together with Lemma 3.1, we can prove Theorem 3.2.

Proof of Theorem 3.2. Fix an m-tuple of sets Ω_j for Algorithm 5, and consider the sets $\textsc{opt} \setminus \Omega_j$, for all $j \leq m$. Note that it holds $\mathsf{V}_j = \mathsf{V} \setminus \cup_{i \leq j} \Omega_i$. Hence, $f(\textsc{opt} \setminus \mathsf{V}_i) = f(\textsc{opt} \setminus (\mathsf{V} \setminus \cup_{i \leq j} \Omega_i)) = f(\cup_{i \leq j} (\textsc{opt} \cap \Omega_j)) \leq \sum_{i \leq j} f(\textsc{opt} \cap \Omega_j)$, where the last inequality uses submodularity.

Unfixing the sets Ω_j and taking the expected value yields

$$\mathbb{E}\left[\, f(\textsc{opt} \setminus \mathsf{V}_i) \,\right] \leq \sum_{i \leq j} \mathbb{E}\left[\, f(\textsc{opt} \cap \Omega_j) \,\right], \tag{3.6}$$

for all indices $j \leq m$. We have that it holds

$$\begin{aligned}
(m-1)\mathtt{OPT} &\leq \sum_{j \leq m} \mathbb{E}\left[\, f(\textsc{opt} \cup \Omega_j) \,\right] \\
&\leq \sum_{j \leq m} \mathbb{E}\left[\, f(\Omega_j \cup (\textsc{opt} \cap \mathsf{V}_j)) \,\right] + \sum_{j \leq m} \mathbb{E}\left[\, f(\textsc{opt} \setminus \mathsf{V}_j) \,\right] \\
&\leq m \frac{p+1}{(1-\varepsilon)^2} \mathbb{E}\left[\, f(\Omega_j) \,\right] + m \frac{\lambda r}{(1-\varepsilon)^2} \mathbb{E}\left[\, f(\Omega_j) \,\right] \\
&\qquad + \sum_{j \leq m} \sum_{i \leq j} \mathbb{E}\left[\, f(\textsc{opt} \cap \Omega_j) \,\right] \\
&\leq m(p+1) \frac{1+\varepsilon}{(1-\varepsilon)^2} \mathbb{E}\left[\, f(\Omega_j) \,\right] + \sum_{j \leq m} \sum_{i \leq j} \mathbb{E}\left[\, f(\textsc{opt} \cap \Omega_j) \,\right] \\
&\leq m(p+1) \frac{1+\varepsilon}{(1-\varepsilon)^2} \mathbb{E}\left[\, f(\Omega_j) \,\right] + 4 \sum_{j \leq m} \sum_{i \leq j} \mathbb{E}\left[\, f(\Lambda_j) \,\right], \\
&\leq m(p+1) \frac{1+\varepsilon}{(1-\varepsilon)^2} \mathbb{E}\left[\, f(\Omega^*) \,\right] + 2m(m-1) \mathbb{E}\left[\, f(\Omega^*) \,\right],
\end{aligned}$$

where the first inequality follows from Lemma 3.7; the second inequality follows from Lemma 3.8; the third inequality follows from (3.6) and Lemma 3.1; the fourth inequality follows from Lemma 3.6; the last inequality follows since $f(\Omega^*)$ is maximum over $f(\Omega_j)$ and $f(\Lambda_j)$. The claim follows by rearranging the inequality above. ■

3.5.4 Proof of Lemma 3.2

We now prove an upper-bound on the run time for Algorithm 5, which we restate here.

▶ **Lemma 3.2.** Fix constants $\varepsilon \in (0,1)$, $\varphi_1, \varphi_2 \in [0,1]$ and $m \geq 0$. Then Algorithm 5 terminates after $\mathcal{O}\left(\frac{m}{\varepsilon^2}\log\left(\frac{r}{p\varepsilon}\right)\log r \log n\right)$ rounds of adaptivity. Furthermore, the query complexity of Algorithm 5 is upper-bounded as $\mathcal{O}\left(\frac{mn}{\varepsilon^2}\log\left(\frac{r}{p\varepsilon}\right)\log r \log n\right)$. ◀

Proof. First, note that Algorithm 3 requires no function evaluation, and it always return a sequence $\{a_i\}_i$ of length at most r.

At each step of Algorithm 4, the BINARY-SEARCH sub-routine requires $\mathcal{O}\big(\log(r)\big)$ iterations, sine the set J has size at most r. Each iteration of this sub-routine requires $\mathcal{O}(1)$ rounds of adaptivity, and $\mathcal{O}\big(\log(r)\big)$ function evaluations. Note also that the while-loop, lines 5-9 of Algorithm 4 terminates after at most $\mathcal{O}\big(\varepsilon^{-1}\log n\big)$ iterations. In fact, we have that $|\mathsf{X}| \leq n$, and that at each iteration the size of the new set X decreases of a multiplicative factor of $(1-\varepsilon)$. Similarly, the outer while-loop, lines 4-11 of Algorithm 4 terminates after at most $\mathcal{O}\big(\varepsilon^{-1}\log(r/p\varepsilon)\big)$ iterations.

Hence, each call of Algorithm 4 requires $\mathcal{O}\left(m\varepsilon^{-2}\log(\frac{r}{p\varepsilon})\log(\varepsilon^{-1}\log r)\log n)\right)$ rounds of adaptivity. Similarly, since the BINARY-SEARCH sub-routine requires $\mathcal{O}\big(n\log(r)\big)$ function evaluations, then the query complexity is as claimed. ■

3.5.5 Proof of Lemma 3.3

We perform the run time analysis for an optimal choice of the parameter m. We have that the following lemma holds.

▶ **Lemma 3.3.** Fix a constant $\varepsilon \in (0,1)$, and define parameters $m = 1 + \lceil\sqrt{(p+1)/2}\rceil$, $\varphi_1 = 1$, and $\varphi_2 = 1/2$. Denote with Ω^* the optimal solution found by Algorithm 5. Then,

$$\texttt{OPT} \leq \frac{1-\varepsilon}{(1-\varepsilon)^2}\left(p + 2\sqrt{2(p+1)} + 5\right)\mathbb{E}[\, f(\Omega^*)\,].$$

Furthermore, with this parameter choice Algorithm 5 terminates after $\mathcal{O}\left(\frac{\sqrt{p}}{\varepsilon^2}\log n \log\left(\frac{r}{p\varepsilon}\right)\log r\right)$ rounds of adaptivity.
Its query complexity is $\mathcal{O}\left(\frac{\sqrt{p}n}{\varepsilon^2}\log n\log\left(\frac{r}{p\varepsilon}\right)\log r\right)$. ◀

Proof. We start with the approximation guarantee. Denote with Ω^* an approximate solution found by Algorithm 5, and let OPT be the optimal solution for Problem 3. Then, from Theorem 3.2 we get

$$\texttt{OPT} \leq m\left(\frac{(1+\varepsilon)(p+1)}{(1-\varepsilon)^2(m-1)} + 2\right)\mathbb{E}[\, f(\Omega^*)\,]$$

Substituting $m = 1 + \lceil\sqrt{(p+1)/2}\rceil$ and rearranging yields

$$\begin{aligned}
\texttt{OPT} &\leq \frac{1+\varepsilon}{(1-\varepsilon)^2}\left(p + 2\left\lceil\sqrt{\frac{p+1}{2}}\right\rceil\right)\mathbb{E}[\, f(\Omega^*)\,] \\
&\quad + \frac{1+\varepsilon}{(1-\varepsilon)^2}\left((p+1)\left\lceil\sqrt{\frac{p+1}{2}}\right\rceil^{-1} + 3\right)\mathbb{E}[\, f(\Omega^*)\,] \\
&\leq \frac{1+\varepsilon}{(1-\varepsilon)^2}\left(p + 2\left(\sqrt{\frac{p+1}{2}} + 1\right)\right)\mathbb{E}[\, f(\Omega^*)\,] \\
&\quad + \frac{1+\varepsilon}{(1-\varepsilon)^2}\left((p+1)\left(\sqrt{\frac{p+1}{2}}\right)^{-1} + 3\right)\mathbb{E}[\, f(\Omega^*)\,]
\end{aligned}$$

$$= \frac{(1+\varepsilon)(p+2\sqrt{2(p+1)}+5)}{(1-\varepsilon)^2}\mathbb{E}\left[\, f(\Omega^*)\,\right].$$

Hence, the claim on the approximation guarantee follows. The upper-bounds on the adaptivity, depth, and total number of calls to the valuation oracle function follow directly from Lemma 3.2. ■

3.6 Run Time Analysis for p-Extendable Systems

3.6.1 Overview of the Main Results

In this section, we perform the theoretical analysis for the REP-SAMPLING, when maximizing a non-monotone submodular function under a p-extendable system side constraint, as in Problem 4. We prove that, with different sets of input parameters, our algorithm has adaptivity and query complexity that is not dependent on p. Again, all proofs are deferred to the appendix. The following theorem holds.

▶ **Theorem 3.3.** Fix parameters $\varepsilon \in (0,1)$, $m = 1$, $\varphi_1 = (p+1)^{-1}$, and $\varphi_2 \in [0,1]$. Denote with Ω^* the output of Algorithm 5. Then,

$$\texttt{OPT} \leq \frac{(1+\varepsilon)(p+1)^2}{p(1-\varepsilon)^2}\mathbb{E}\left[\, f(\Omega^*)\,\right].$$

With this parameter choice, Algorithm 5 terminates after $\mathcal{O}\left(\varepsilon^{-2}\log n \log\left(\frac{r}{\varepsilon}\right)\log r\right)$ rounds of adaptivity, and it requires $\mathcal{O}\left(\frac{n}{\varepsilon^2}\log n \log\left(\frac{r}{\varepsilon}\right)\log r\right)$ function evaluations. ◀

We give a proof of this theorem later on in this section. Before discussing the proof of Theorem 3.3, we introduce additional notation and lemmas. The proof of Theorem 3.3 is based on the work of (Moran Feldman, Harshaw, et al. 2017), together with the fact that Algorithm 4

1. $O_0 := \emptyset$;
2. if $v_i \in \Omega$, then $O_i \subseteq \text{OPT} \setminus (T_i \cup_{j=0}^{i-1} O_j)$ is a set of minimum size such that $(\text{OPT} \setminus (\cup_{j=0}^{i} O_j)) \cup T_i \in \mathscr{I}$;
3. if $v_i \notin \Omega$, $X_{v_i} = 1$, and $v_i \in \text{OPT} \setminus (\cup_{j=0}^{i-1} O_j)$, then $O_i = \{v_i\}$;
4. if $v_i \notin \Omega$ and $v_i \notin \text{OPT} \setminus (\cup_{j=0}^{i-1} O_j)$, or if $X_{v_i} = 0$, then $O_i = \emptyset$.

Figure 3.1: A recursive definition of the sets $\{O_j\}$, used for the run time analysis of Algorithm 5, in the case of p-extendable systems (see Theorem 3.3).

yields expected marginal increase lower-bounded by the best possible greedy improvement, up to a multiplicative constant. We remark that Lemma 3.3 also holds when side constraints are p-matchoids and the intersections of matroids, since p-extendable systems are a generalization of both.

We introduce additional notation, that will be used to prove Theorem 3.3, as well as all preliminary lemmas. First of all, since $m = 1$, we need not specify the index of the input search space V_i and solution Ω_i of Algorithm 4, and we simply use the notation $V = V_1$ and $\Omega = \Omega_1$. Again we define $|V| = n$. Furthermore, we define an ordering of the points $\{v_i\}_i = V$, with v_i the i-th point sampled by Algorithm 4 during run time. All points of V that are not sampled during run time, are placed at the end of the sequence $\{v_i\}_i$ in random order. We also define the sets $T_i = \{v_1, \dots, v_i\} \cap \Omega$, and we define the sequence $\{v_i^*\}$ as

$$v_i^* := \max_{\{v \in V \setminus T_{i-1} \colon T_{i-1} \cup v \in \mathscr{I}\}} f_{T_{i-1}}(v).$$

For each point $v \in V$, denote with X_v an indicator function such that $X_v = 1$ if v is sampled as part of any random feasible sequence $\{a_1, \ldots, a_\eta\}$ during run time, and $X_v = 0$ otherwise. We also consider a sequence $\{O_i\}_{i=0}^n$ of sets $O_i \subseteq V$ defined recursively as in Figure 3.1. We define the set $O := (\text{OPT} \setminus (\cup_{i=0}^n O_i)) \cup T_n = (\text{OPT} \setminus (\cup_{i=0}^n O_i)) \cup \Omega$. It can be proven that the set O as defined above, can be iteratively defined with a greedy algorithm that outputs a constant-factor approximation as in Theorem 3.3 (Moran Feldman, Harshaw, et al. 2017). Hence, the proof of Theorem 3.3 extends an argument based on properties of a greedy algorithm for Problem 4 to adaptive sampling algorithms.

In order to prove Theorem 3.3, we use the following well-known result.

▶ **Lemma 3.9 (Lemma 2.2 in (Buchbinder, Moran Feldman, Naor, et al. 2014)).** Let $\Omega \subseteq V$ be a set such that each element appears in Ω with probability at most k. Then it holds $\mathbb{E}[\, f(U) \,] \geq (1-k) f(\emptyset)$. ◀

In order to prove Theorem 3.3 we use three additional lemmas. These lemmas use the notation introduced earlier in this section.

▶ **Lemma 3.10.** Fix all random decisions of Algorithm 4. Then it holds

$$f(\Omega) + |O \setminus \Omega| \delta_0 \geq f(\Omega \cup \text{OPT}) - \sum_{i=0}^{n} |O_i \setminus \Omega| f_{T_{i-1}}(v_i^*).$$

◀

▶ **Lemma 3.11.** It holds

$$\mathbb{E}_{v_i}[\, |O_i \setminus \Omega| f_{T_{i-1}}(v_i^*) \,] \leq (1-\varepsilon)^2 \frac{p}{p+1} \mathbb{E}_{v_i}[\, \mathcal{X}_{v_i} f_{T_{i-1}}(v_i) \,].$$

◀

▶ **Lemma 3.12.** It holds

$$\sum_{i=1}^{n} \mathbb{E}\left[\, |\mathsf{O}_i \setminus \Omega| f_{\mathsf{T}_{i-1}}(v_i^*) \,\right] \leq p(1-\varepsilon)^2 \mathbb{E}\left[\, f(\Omega) \,\right].$$

◀

We prove these lemmas in Section 3.6.2. We now have all necessary tools to prove the main guarantee for p-extendable systems.

Proof of Theorem 3.3. Combining Lemma 3.10, taking the expected value, and combining with Lemma 3.12 we get

$$\frac{p+1}{(1-\varepsilon)^2}\mathbb{E}\left[\, f(\Omega) \,\right] + \mathbb{E}[\, |\mathsf{O} \setminus \Omega| \delta_0 \,] \geq \mathbb{E}\left[\, f(\Omega \cup \mathrm{OPT}) \,\right]. \tag{3.7}$$

Denote with $\{a_i\}_i$ the points of Ω in the order that they were added to Ω. From Lemma 3.5 and submodularity, we have that $\mathbb{E}_{a_i}\left[\, f_{\{a_1,\ldots,a_{i-1}\}}(a_i) \,\right] \geq 0$, for all points a_i added to Ω. It follows that $\mathbb{E}\left[\, f(\Omega) \,\right] \geq \mathbb{E}\left[\, f(a_0) \,\right]$, with a_0 the first point added to Ω. Hence, from the definition of δ_0, and since $|\mathsf{O} \setminus \Omega| \leq r$ due to feasibility, we get $\varepsilon(p+1)\mathbb{E}\left[\, f(\Omega) \,\right] \geq \mathbb{E}[\, |\mathsf{O} \setminus \Omega| \delta_0 \,]$. Substituting in (3.7) we get

$$\frac{(p+1)(1+\varepsilon)}{(1-\varepsilon)^2}\mathbb{E}\left[\, f(\Omega) \,\right] \geq \mathbb{E}\left[\, f(\Omega \cup \mathrm{OPT}) \,\right].$$

To conclude the proof, we observe that the function $g(\mathsf{S}) = f(\mathsf{S} \cup \mathrm{OPT})$ is a submodular function. Since each element of V appears in Ω w.p. at most $(p+1)^{-1}$, then by Lemma 3.9 we get

$$\mathbb{E}\left[\, f(\Omega \cup \mathrm{OPT}) \,\right] = \mathbb{E}\left[\, g(\Omega) \,\right] \geq \frac{p}{p+1} g(\emptyset) = \frac{p}{p+1}\mathtt{OPT}.$$

The claim follows. ■

3.6.2 Proof of the Preliminary Lemmas

We prove all additional lemmas, that are then used to prove Theorem 3.3. In this section, we use the notation introduced in the previous section. We first prove the following result.

▶ **Lemma 3.10.** Fix all random decisions of Algorithm 4. Then it holds

$$f(\Omega) + |\mathsf{O} \setminus \Omega|\delta_0 \geq f(\Omega \cup \mathrm{OPT}) - \sum_{i=0}^{n} |\mathsf{O}_i \setminus \Omega| f_{\mathsf{T}_{i-1}}(v_i^*).$$

◀

Proof. First, we prove that it holds

$$f(\Omega) + |\mathsf{O} \setminus \mathsf{S}|\delta_0 \geq f(\mathsf{O}). \tag{3.8}$$

To this end, note that since $(\mathrm{OPT} \setminus (\cup_{i=0}^{n} \mathsf{O}_i)) \cup \Omega \in \mathscr{I}$, then it holds $\{v\} \cup \Omega \in \mathscr{I}$ for all $v \in \mathsf{O} \setminus \Omega$. Hence, by the termination criterion of Algorithm 4, we have that $f_\Omega(v) \leq \delta_0$. Hence,

$$f(\mathsf{O}) \leq f(\Omega) + \sum_{v \in \mathsf{O} \setminus \Omega} f_\Omega(v) \leq f(\Omega) + |\mathsf{O} \setminus \Omega|\delta_0,$$

where we have used submodularity. Then (3.8) follows.

Next, we prove that it holds

$$f(\mathsf{O}) \geq f(\Omega \cup \mathrm{OPT}) - \sum_{i=1}^{n} |\mathsf{O}_i \setminus \Omega| f_{\mathsf{T}_{i-1}}(v_i^*). \tag{3.9}$$

Note that the claim of this lemma follows by combining (3.8) and (3.9).

To prove (3.9), we first observe that the sets $\mathsf{O}_i \setminus \Omega$ are mutually disjoint, and we can write $\mathsf{O} = (\Omega \cup \mathrm{OPT}) \setminus (\cup_{i=1}^{n} (\mathsf{O}_i \setminus \Omega))$. Using this

equality, we have that

$$\begin{aligned} f(\mathsf{O}) &\geq f(\Omega \cup \mathrm{OPT}) - \sum_{i=1}^{n} f_{(\Omega \cup \mathrm{OPT}) \setminus (\cup_{j=0}^{i-1} (\mathsf{O}_j \setminus \Omega))}(\mathsf{O}_i \setminus \Omega) \\ &\geq f(\Omega \cup \mathrm{OPT}) - \sum_{i=1}^{n} f_{\mathsf{T}_{i-1}}(\mathsf{O}_i \setminus \Omega) \\ &\geq f(\Omega \cup \mathrm{OPT}) - \sum_{i=1}^{n} \sum_{v \in \mathsf{O} \setminus \Omega} f_{\mathsf{T}_{i-1}}(v), \end{aligned}$$

where the first equation is the telescopic sum, the second one uses submodularity, together with the fact that $\mathsf{T}_{i-1} \subseteq \Omega$, and the third one use submodularity again. Then (3.9) follows from the definition of v_i^*. ■

Next, using Lemma 3.10 we prove the following result.

▶ **Lemma 3.11.** It holds

$$\mathbb{E}_{v_i}\left[\, |\mathsf{O}_i \setminus \Omega| f_{\mathsf{T}_{i-1}}(v_i^*) \,\right] \leq (1-\varepsilon)^2 \frac{p}{p+1} \mathbb{E}_{v_i}\left[\, \mathscr{X}_{v_i} f_{\mathsf{T}_{i-1}}(v_i) \,\right].$$

◀

Proof. We first observe that if $\mathscr{X}_i = 0$, then the claim holds since $|\mathsf{O}_i \setminus \Omega| = \emptyset$. Hence, we prove the claim by conditioning on the event $\{\mathscr{X}_{v_i} = 1\}$. In this case, the point v_i is added to Ω w.p. $(p+1)^{-1}$.

If v_i is not added to the current solution, then $|\mathsf{O}_i| \leq 1$, and conditioning on the event $\{v_i \notin \Omega\}$ we get

$$\begin{aligned} &\mathbb{E}_{a_i}\left[\, |\mathsf{O}_i \setminus \Omega| f_{\mathsf{T}_{i-1}}(v_i^*) \mid \mathscr{X}_{v_i} = 1, v_i \notin \Omega \,\right] \\ &\qquad \leq \mathbb{E}_{a_i}\left[\, \mathscr{X}_{v_i} f_{\mathsf{T}_{i-1}}(v_i^*) \mid \mathscr{X}_{v_i} = 1, v_i \notin \Omega \,\right] \\ &\qquad \leq \frac{p}{p+1} \mathbb{E}_{a_i}\left[\, \mathscr{X}_{v_i} f_{\mathsf{T}_{i-1}}(v_i^*) \mid \mathscr{X}_{v_i} = 1 \,\right], \end{aligned}$$

where we have used that $\mathsf{Pr}(a_i \notin \mathsf{S}) = 1 - (p+1)^{-1}$. Furthermore, if the

solution v_i is added to Ω, then the set O_i has size at most $|O_i| \leq p$, since $\mathscr{I}$ is a p-extendable system. Hence,

$$\begin{aligned}\mathbb{E}_{a_i}\left[\, |O_i \setminus \Omega| f_{T_{i-1}}(v_i^*) \mid \mathscr{X}_{v_i} = 1, v_i \in \Omega \,\right] &\leq p\mathbb{E}_{a_i}\left[\, \mathscr{X}_{v_i} f_{T_{i-1}}(v_i^*) \mid \mathscr{X}_{v_i} = 1, v_i \in \Omega \,\right] \\ &\leq \frac{p}{p+1}\mathbb{E}_{a_i}\left[\, \mathscr{X}_{v_i} f_{T_{i-1}}(v_i^*) \mid \mathscr{X}_{v_i} = 1 \,\right].\end{aligned}$$

The claim follows combining the two chains of inequalities above, together with Lemma 3.4 and Lemma 3.5. ■

We also need the following lemma, to prove the main theorem.

▶ **Lemma 3.12.** It holds

$$\sum_{i=1}^{n} \mathbb{E}\left[\, |O_i \setminus \Omega| f_{T_{i-1}}(v_i^*) \,\right] \leq p(1-\varepsilon)^2 \mathbb{E}\left[\, f(\Omega) \,\right].$$

◀

Proof. For each $v_i \in V$, let $\mathscr{G}_{v_i}$ be a random variable whose value is equal to in the increase in the value of the current solution when v_i is added to it. Note that if v_i yields $\mathscr{X}_{v_i} = 0$, then $\mathscr{G}_{v_i} = 0$ because it cannot be added to the current solution. Hence, it holds $\sum_{i=1}^{n} \mathbb{E}\left[\, \mathscr{G}_{v_i} \,\right] = \mathbb{E}\left[\, f(\Omega) \,\right]$. Then,

$$\begin{aligned}\mathbb{E}_{v_i}\left[\, \mathscr{G}_{v_i} \,\right] &= \Pr(v_i \in \Omega)\mathbb{E}_{v_i}\left[\, \mathscr{X}_{v_i} f_{T_{i-1}}(v_i) \,\right] \\ &= \frac{1}{p+1}\mathbb{E}_{a_i}\left[\, \mathscr{X}_{v_i} f_{T_{i-1}}(v_i) \,\right].\end{aligned}$$

The claim follows using Lemma 3.11, and using the law of total probability and linearity of the expected value. ■

3.7 Complexity and Adaptivity of the Independence Oracle

We conclude our analysis with a general discussion on the performance of Algorithm 5 in the number of calls to the independence oracle for the p-system constraint. As discussed in Section 3.4, at each adaptive round our algorithm calls Algorithm 3 to build a random feasible sequence. This procedure requires to find a maximum independent set.

Finding a maximum independent set requires calls to the independence oracle, which takes as input a set S, and returns as output a Boolean value, true if the given set is independent in $\mathscr{I}$ and false otherwise. These oracle evaluations are typically time-consuming, since they require to evaluate feasibility with respect to complex side constraints. Hence, a fast algorithm for this problem has good performance in terms of rounds of independence calls to the oracle function. The following lemma holds.

▶ **Lemma 3.13.** Fix parameters $\varepsilon \in (0,1)$, $m \geq 1$, and $\varphi_1, \varphi_2 \in [0,1]$. Then Algorithm 5 requires expected $\mathcal{O}\left(\frac{m\sqrt{n}}{\varepsilon^2}\log\left(\frac{r}{p\varepsilon}\right)\log r \log n\right)$ rounds of independent calls to the oracle for the p-system constraint. Furthermore, the total number of calls to the independence system is $\mathcal{O}\left(\frac{mn^{3/2}}{\varepsilon^2}\log\left(\frac{r}{p\varepsilon}\right)\log r \log n\right)$. ◀

In order to prove this Lemma, we use the following well-known result.

▶ **Theorem 3.4 (Theorem 6 in (Karp et al. 1988)).** Algorithm 3 terminates after $\mathcal{O}(\sqrt{r})$ steps. ◀

Combining Theorem 3.4 with Lemma 3.2, we prove Lemma 3.13 as follows.

Proof of Lemma 3.13. We first observed that the independence oracle for the p-system is called by Algorithm 3, and also by Algorithm

4. Since at each iteration of Algorithm 3, queries to the oracle for the p-system are independent, the from Theorem 3.4 it follows that Algorithm 3 requires $\mathcal{O}\left(\sqrt{r}\right)$ rounds of independent calls to the oracle for the p-system. Furthermore, all calls to the independence oracle for the p-system in Algorithm 4 are independent. Combining these observations with Lemma 3.2 it follows that Algorithm 5 requires $\mathcal{O}\left(m\sqrt{n}/\varepsilon^2 \log(n)\log(r/\varepsilon)\right)$ rounds of independent calls to the oracle for the p-system. The claim follows, since at each round at most $\mathcal{O}(n)$ calls to the oracle for the p-system are executed in parallel. ■

Note that the bound on the rounds of independent calls to the independence oracle, as in Lemma 3.13 is sub-linear, but not poly-logarithmic in the problem size. The reason is that Algorithm 3 requires $\mathcal{O}\left(\sqrt{n}\right)$ rounds of independent calls to the oracle for the p-system. We are not aware of any algorithm that finds a base in less than $\mathcal{O}\left(\sqrt{n}\right)$ rounds. Furthermore, it is well-known that there is no algorithm that obtains an approximation guarantee that is constant in the problem size for Problem 3, than $\tilde{\Omega}(n^{1/3})$ steps of independent calls to the oracle for the p-system constraint (see (Balkanski, Rubinstein, et al. 2019; Karp et al. 1988)).

For a p-system $\mathcal{I}$, the rank of a set S is the maximum cardinality of its intersection with a maximum independent set in $\mathcal{I}$. Given access to an oracle that returns the rank of a set in $\mathcal{I}$, it is possible to design an algorithm that finds a maximum independent set of a p-system in $\mathcal{O}\left(\log n^2\right)$ rounds of independent calls to the rank oracle (see (Karp et al. 1988)). However, this work focuses on general constraints where the rank of a set is not known.

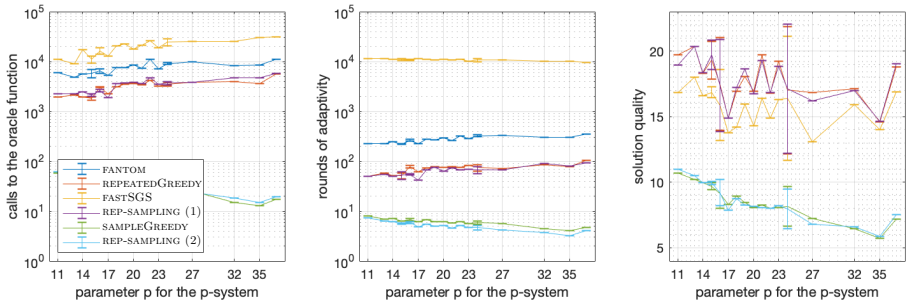


Figure 3.2: Results for the experiments on Video Summarization on movie segments taken from FLIC (Sapp and Taskar 2013). Each plot shows the average performance over segments with fixed p. Error bars correspond to the best and worst case. Note that the y-axis in the two leftmost plots uses a logarithmic scale. The FASTSGS uses parameters $\ell = \lfloor 2 + \sqrt{p+1} \rfloor$ and $\varepsilon = 0.1$; the REP-SAMPLING (1) uses parameters $\varepsilon = 0.1$, $m = 1 + \lceil \sqrt{(p+1)/2} \rceil$, $\varphi_1 = 1$, $\varphi_2 = 0.5$; the REP-SAMPLING (2) uses parameters $\varepsilon = 0.01$, $m = 1$, $\varphi_1 = (1+p)^{-1}$, $\varphi_2 = 1$.

3.8 Experiments

3.8.1 Benchmarks

In this section, we demonstrate experimentally that the REP-SAMPLING has superior performance that several other heuristics, suitable for Video Summarization. As discussed in Section 3.1, Video Summarization consists of choosing a sequence of frames that gives a descriptive overview of a given video. In our set of experiments, we implement the REP-SAMPLING as describe in Algorithm 5. We always test our algorithm against these algorithms:

The FANTOM algorithm. This algorithm, which iterates a greedy algorithm multiple times, is studied in (A. Gupta et al. 2010) and (Mirzasoleiman, Badanidiyuru, and Karbasi 2016).

The REPEATEDGREEDY algorithm. This algorithm consists of iterating a greedy algorithm multiple times (Moran Feldman, Harshaw, et al. 2017). It uses Algorithm 1 in (Buchbinder, Moran Feldman, Naor, et al. 2015) as a sub-routine.

The FASTSGS algorithm. This algorithm is studied in (Moran Feldman, Harshaw, et al. 2020), and it is essentially a fast implementation of the SIMULTANEOUSGREEDYS (Moran Feldman, Harshaw, et al. 2020). This algorithm updates multiple solutions concurrently, and it picks the best of them.

The SAMPLEGREEDY algorithm. This algorithm is specifically designed to handle p-extendable systems (see (Moran Feldman, Harshaw, et al. 2017)). This algorithm samples points independently at random, and then it builds a greedy solution over the resulting set.

We test the REP-SAMPLING against these algorithms, since they can be considered the state-of-the-art for Video Summarization.

3.8.2 Results

We select a representative summary by maximizing the function $\log\det_L(S)$, which is non-monotone and submodular. We impose the following additional side constraints. First, we impose an upper-bound on the maximum number of frames of each summary. Then, we partition each video into segments, and define a partition matroid to select at most ℓ_j frames in each segment j. Following (Moran Feldman, Karbasi, et al. 2018; Mirzasoleiman, Jegelka, et al. 2018b), we also use a face-recognition tool to identify actors in each movie, and select a summary containing at most k_i frames showing face i. This additional constraint corresponds to a p-system $\mathcal{I} = \{S \subseteq V : |S \cap V_i| \leq k_i\}$, with V_i all frames containing face i. The parameter p is estimated by counting the total number of distinct faces i that appear in more than k_i frames. In our experiments, the parameters

k_i are always set to a fixed constant for all videos. Hence, the only variable that affects p is the total number of distinct faces in each movie.

For our experimental investigation, we use movies from the Frames Labeled In Cinema (FLIC) data-set (Sapp and Taskar 2013). We consider all movies in this data-set with at least 200 frames, as to highlight performance when dealing with large problem size. We consider various parameter choices for the REP-SAMPLING. These parameter choices are as in Lemma 3.3 and Theorem 3.3.

The results are displayed in Figure 3.2, where we describe the parameter choice for each algorithm. For each non-deterministic algorithm, results are the sample mean of 100 independent runs. We observe that, for different parameter choice, our algorithm outperforms FANTOM and the FASTSGS, and it has better adaptivity than the greedy algorithms. The solution quality for the SAMPLEGREEDY and REP-SAMPLING, with parameters as in Lemma 3.3, is worse on these instances.

4 Feature Selection

This chapter is based on a conference paper titled "Fast Feature Selection with Fairness Constraints", by Francesco Quinzan, and Rajiv Khanna, Sarel Cohen, Moshik Hershcovitch, Daniel G. Waddington, Tobias Friedrich, Michael W. Mahoney (Quinzan, Khanna, et al. 2022).

We study the fundamental problem of selecting a few features for a given modeling problem, while satisfying additional side constraints.[7] This problem is computationally challenging on large datasets, even with the use of greedy algorithm variants. To address this challenge, we extend the adaptive query model, recently proposed for the greedy forward selection for submodular functions, to the faster paradigm of Orthogonal Matching Pursuit for non-submodular functions. Our extension also allows the use of downward-closed constraints, which can be used to encode certain fairness criteria into the feature selection process. The proposed algorithm achieves exponentially fast parallel run time in the adaptive query model, scaling much better than prior work. The proposed algorithm also handles certain fairness constraints by design. We prove strong approximation guarantees for the algorithm based on standard assumptions. These guarantees are applicable to many parametric models, including Generalized Linear Models. Finally, we demonstrate empirically that the proposed

7 We recognize that ensuring fairness in any machine learning model is a complicated problem. Here, we consider some of the fairness constraints proposed in the literature. How these and related results fit within applied data science pipelines is a complicated and open challenge. Our theoretical results are applicable more generally, and there may also be other notions of fairness which do not fall within our theoretical framework.

algorithm competes favorably with state-of-the-art techniques for feature selection, on real-world and synthetic datasets.

4.1 Feature Selection as an Optimization Problem

Feature Selection can be framed as follows. Given a function $l : \mathbb{R}^p \to \mathbb{R}_{>0}$ expressing the goodness of fit for a given model, we search for a set of features S maximizing the function

$$f(\mathsf{S}) := l(\boldsymbol{\beta}^{(\mathsf{S})}) - l(\mathbf{0}). \tag{4.1}$$

Here, $\mathbf{0}$ represents the n-dimensional zero vector, and $\boldsymbol{\beta}^{(\mathsf{S})}$ is a vector maximizing $l(\cdot)$ with non-zero entries corresponding to the features in S. If we denote by $\mathscr{I}$ the set of all acceptable solutions that satisfy the side constraints, then the feature selection optimization problem with side constraints can be formalized as

$$\underset{\mathsf{S}\subseteq[n]\,:\,\mathsf{S}\in\mathscr{I}}{\operatorname{argmax}} f(\mathsf{S}) = \underset{\mathsf{S}\subseteq[n]\,:\,\mathsf{S}\in\mathscr{I}}{\operatorname{argmax}} l(\boldsymbol{\beta}^{(\mathsf{S})}) - l(\mathbf{0}), \tag{4.2}$$

where $[n]$ is the index set of all the features.

Typically, $\mathscr{I}$ corresponds to an r-sparsity constraint, i.e., a solution S is feasible if it contains at most r features. Several algorithms have been proposed for feature selection under sparsity constraints. Examples include the Lasso, algorithms based on the forward stepwise selection (Elenberg, Khanna, et al. 2018; S. Qian and Singer 2019), algorithms based on the Orthogonal Matching Pursuit (Elenberg, Khanna, et al. 2018; Needell and Tropp 2010; Sakaue 2020), forward-backward methods (Jalali et al. 2011; Liu et al. 2014), Pareto optimization (C. Qian, C. Bian, et al. 2020; C. Qian, Y. Yu, et al. 2015), Exponential Screening (Rigollet and Tsybakov 2010), and gradient-based methods (Jain et al. 2014; Yuan et al. 2017).

The aforementioned algorithms, however, do not take into account additional side concerns such as fairness, which is crucial when deploying machine learning systems in the real world. Recently, there has been a growing effort towards developing fair algorithms for several fundamental problems, such as regression and classification (Agarwal, Beygelzimer, et al. 2018; Michael Feldman et al. 2015; Grgic-Hlaca, Zafar, K. P. Gummadi, et al. 2018; Kim et al. 2018; Zafar, Valera, Gomez-Rodriguez, and K. P. Gummadi 2017b), matching (Chierichetti et al. 2019), and summarization (Celis et al. 2018; Halabi et al. 2020). Motivated by this line of research, we consider the following research question:

How can we efficiently perform feature selection,
while taking into account additional constraints such as fairness?

We address this question by studying the optimization problem (4.2) under general side constraints $\mathscr{I}$, that can be used to enumerate notions of fairness. Our approach is motivated by the observation that several definitions of fairness can be incorporated in the learning process as additional side constraints (Agarwal, Beygelzimer, et al. 2018; Agarwal, Dudík, et al. 2019; Chierichetti et al. 2019; Donini et al. 2018; Grgic-Hlaca, Redmiles, et al. 2018; Grgic-Hlaca, Zafar, K. Gummadi, et al. 2016; Grgic-Hlaca, Zafar, K. P. Gummadi, et al. 2018; Halabi et al. 2020; Woodworth et al. 2017; Zafar, Valera, Gomez-Rodriguez, and K. P. Gummadi 2017b).

To this end, we provide a novel algorithmic result based on the paradigm of matching pursuit for Problem (4.2), which is much faster than previously known techniques in the adaptive query model. Our framework also allows for certain notions of fairness in the learning process, via a reduction to the p-system side constraints (see Section 4.2.4). We analyze our approach theoretically, and also present convincing empirical evidence in favor of our approach in the sequel. Our theoretical guarantees are applicable more generally, and we use the fairness constraints as an illustrative real-world application. At

the same time, we also recognize that quantifying the full meaning of the notion of fairness with its societal context as understood by human beings may be hard to achieve through mathematical formalism (Selbst et al. 2019). As such, p-systems may not explicate all current or future codifications of fairness. Our claim is not to *solve* the problem of fairness itself, but rather providing a theoretically sound algorithm and analysis for some of its current manifestations, with the hope of serving as a blueprint for encouraging further developments along similar lines in the future.

Optimization problems as in (4.2) under general side constraints are computationally challenging in practise. A major bottleneck lies in the evaluation of $\beta^{(S)}$ for a given set S, since this operation requires to re-train the model onto every candidate set S. Another challenge is enforcing the side constraint $\mathscr{I}$ that encodes the selection criteria, except in certain trivial cases where the protected classes are known a priori (Beutel et al. 2019; Lahoti et al. 2020). In this work, we propose an algorithm suitable for Problem (4.2), that is efficient with respect to these computational challenges. We propose a novel matching pursuit algorithm for the constrained feature selection problem (4.2). This algorithm uses oracle access to the gradient $\nabla l(\beta^{(S)})$, and to an oracle for the evaluation of the feasibility of an input solution set. This oracle model is well-aligned with previous related work (Elenberg, Khanna, et al. 2018; Sakaue 2020). Our algorithm converges after poly-logarithmic rounds in the recently proposed adaptive query model. In each round, calls to the oracle functions can be performed in parallel, resulting into a dramatic speed-up.

Our algorithm is based on a general technique called adaptive sequencing, that was recently proposed for submodular functions (Balkanski, Rubinstein, et al. 2019; Breuer et al. 2020). However, previous adaptive sequencing algorithms fail on Problem (4.2).

The main theoretical contributions of this paper are two-fold: (a) We extend the adaptive query model to a class of non-submodular objectives using the paradigm of gradient based pursuit algorithms,

(b) we incorporate general downward-closed constraints in the optimization process beyond the standard sparsity constraints. Our main result can be stated as follows.

▶ **Theorem 4.1 (main result, informal).** Denote with OPT the global maximum of a function $l(\cdot)$ as in (4.2). There exists a randomized algorithm that outputs a set of features S^* such that

$$\frac{\mathbb{E}[l(\boldsymbol{\beta}^{(\mathsf{S}^*)})] - l(\mathbf{0})}{\mathtt{OPT}} \geq \frac{1}{1+p}\big(1 - \exp\{-\alpha(1-\varepsilon)^2\}\big),$$

for all $0 < \varepsilon < 1$. Here, p is a parameter that depends on $\mathscr{I}$, and α is a parameter that depends on l. For n total number of features, this algorithm uses $\mathcal{O}\big(\varepsilon^{-2}\log n\big)$ rounds of calls to $\nabla l(\boldsymbol{\beta}^{(\mathsf{S})})$. Furthermore, this algorithm uses expected $\mathcal{O}\big(\varepsilon^{-2}\sqrt{r}\log n\big)$ rounds of calls to the oracle for the feasibility of an input solution set, where r is the largest size of a feasible solution. ◀

In this section, we always denote with OPT the global maximum of the optimization problem, and with OPT any solution set such that $\mathtt{OPT} = \mathbb{E}[l(\boldsymbol{\beta}^{(\mathsf{OPT})})] - l(\mathbf{0})$. To the best of our knowledge, our algorithm is the fastest known algorithm for the general setting of maximizing non-submodular functions with side constraints, with provable guarantees and strong empirical performance (see Section 4.7). (S. Qian and Singer 2019) propose another algorithm for maximizing non-submodular functions that converges after poly-logarithmic rounds. However, their algorithm cannot handle general side constraints $\mathscr{I}$ by design. Hence, it is unsuitable for settings when notions of fairness are to be incorporated in the feature selection process. Furthermore, for the standard r-sparsity constraint, their approximation guarantee is worse then ours (see Theorem 4.4).

4.1.1 Technical Overview

In our analysis, we face two major technical challenges. The first challenge is that of re-purposing the adaptive sequencing for functions that are not submodular, without a significant loss in the approximation guarantee. Adaptive sequencing has been so far employed only for maximizing submodular functions. Interestingly, it is known that standard adaptive sampling techniques do not guarantee constant factor approximation for weakly or differentially submodular functions (S. Qian and Singer 2019), which are typically invoked for feature selection theoretical studies (Elenberg, Khanna, et al. 2018).

The second challenge consists of integrating a constrained selection process based on orthogonal projections in the above adaptive sequencing framework. Common objective functions for feature selection do not have certain desirable properties (e.g. an antitone gradient) and the standard analysis for adaptive sequencing fails in our setting. To resolve these issues, we prove a new connection between gradient evaluations of functions that are restricted strong concave, and their marginal contributions, which may be of independent interest. This connection allows us to bound the gradient evaluations in terms of a discrete function, which is in turn used in the analysis to obtain the desired approximation guarantees. See Theorem 4.4 for the formal result.

4.1.2 Motivating Example

In this section, we motivate our analysis by showing that previous algorithms based on adaptive sequencing do not work for feature selection.

We illustrate this with an example, showing that adaptive sampling fails. To this end, we consider the idealized algorithm proposed by (Breuer et al. 2020), as presented in Algorithm 6. This algorithm is based on (Balkanski, Rubinstein, et al. 2019). Following (Breuer et

al. 2020), we refer to this algorithm as the FAST. Starting from the empty set, the FAST generates at every iteration a uniformly random sequence $\{a_1, \ldots, a_k\}$ of the elements X not yet discarded. This sequence can be sampled uniformly at random, or using the RNDSEQ sub-routine. Afterwards, the FAST filters elements that have a high marginal contribution, when added to S, and it determines the prefix of $\{a_1, \ldots, a_k\}$ that is added to the current solution S. This prefix has the property that there is a large fraction of elements in X with high contribution to the current solution S. To find this prefix, the FAST uses a standard binary search sub-routine, which we refer to as B-SEARCH. Note that the B-SEARCH sub-routine only evaluates contributions with respect to a set R of size m, sampled uniformly at random from X, using the SAMPLE sub-routine.

Following and example provided by (Elenberg, Khanna, et al. 2018), we show that the FAST fails on a simple linear regression task with three features. For a fixed parameter $z > 0$, consider the following variables:

$$\begin{aligned}
\mathbf{y} &= [1,0,0]^T \\
\mathbf{x}_1 &= [0,1,0]^T \\
\mathbf{x}_2 &= [z, \sqrt{1-z^2}, 0]^T \\
\mathbf{x}_3 &= [2z, 0, \sqrt{1-4z^2}]^T
\end{aligned}$$

Note that all variables have unit norm. Our goal is to choose two of the three variables $\{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3\}$ that best estimate $\mathbf{y}$, with respect to the R^2 objective. To this end, we introduce additional notation. For a given index set $S \subseteq [3]$, we denote with $\mathbf{X}_S$ the matrix whose columns consists of the features indexed by S. For instance, for $S = \{1,3\}$ it holds

$$\mathbf{X}_{\{1,3\}} = \begin{bmatrix} 0 & 2z \\ 1 & 0 \\ 0 & \sqrt{1-4z^2} \end{bmatrix}.$$

With this notation, we define the objective function for our problem as

$$f(\mathsf{S}) = R^2(\beta^{(\mathsf{S})}) - R^2(\mathbf{0}) = (\mathbf{y}^T\mathbf{X}_\mathsf{S})(\mathbf{X}_\mathsf{S}^T\mathbf{X}_\mathsf{S})^{-1}(\mathbf{X}_\mathsf{S}^T\mathbf{y}), \tag{4.3}$$

with $R^2(\cdot)$ the R^2 objective evaluated on the model for an input parameter vector $\beta^{(\mathsf{S})}$. Using this formula, one can easily see that it holds

$$\begin{aligned} f(\{1\}) &= 0 & f(\{1,2\}) &= 1 \\ f(\{2\}) &= z^2 & f(\{1,3\}) &= 4z^2 \\ f(\{3\}) &= 4z^2 & f(\{2,3\}) &= (5z^2 - 8z^4)(1-4z^4)^{-1} \end{aligned}$$

Clearly, the optimal solution is $f(\{1,2\})$. However, on this instance the FAST outputs the solution $\mathsf{S} = \emptyset$, attaining an f-value of $f(\mathsf{S}) = 0$. The following lemma holds.

▶ **Lemma 4.1.** Consider the FAST optimizing the function $f(\mathsf{S})$ as in (4.3), over sets $\mathsf{S} \subseteq [3]$ of size at most $|\mathsf{S}| \leq 2$. Then, for any constant $\varepsilon > 0$ and parameter $z > 0$ sufficiently small, the FAST outputs the solution $\mathsf{S} = \emptyset$. ◀

Proof. We first observe that as long as $\mathsf{S} = \emptyset$, the FAST sets $t \leftarrow (1-\varepsilon)/2$, since OPT $= 1$ (see Line 4 of Algorithm 6). Assuming that it holds $z^2 < (1-\varepsilon)/2$, then each singleton $s \in [3]$ yields $t > f(\{s\})$. Hence, with that choice of t no point s can be added to the current solution, and the FAST terminates after ε^{-1} iteration, by returning the empty set. ■

4.2 Preliminaries

We denote with n the number of features, i.e., the dimension of the domain of $l(\cdot)$, and we define $[n] := \{1, 2, \ldots, n\}$. For any $s \in [n]$, we denote with $\mathbf{e}_s$ the unit vector, with a 1 for the coefficient indexed by s, and 0 otherwise. Feature sets are represented by sans script fonts,

i.e., S, T. Vectors are represented by lower-case bold letters as $\boldsymbol{x}$, $\boldsymbol{y}$, and matrices are represented by upper-case bold letters, i.e., $\boldsymbol{X}$, $\boldsymbol{Y}$. Similarly, feature vectors are represented by Greek lower-case bold letters, and covariance matrices are represented by Greek upper-case bold letters, as $\boldsymbol{\beta}$ and $\boldsymbol{\Sigma}$ respectively. For a feature set S, we denote with $\boldsymbol{\beta}^{(\mathsf{S})}$ a vector maximizing $l(\cdot)$ with non-zero entries indexed by the set S. For a feature set T and a parameter vector $\boldsymbol{\beta}$, we define $\nabla l(\boldsymbol{\beta})_{\mathsf{T}} := \langle \nabla l(\boldsymbol{\beta}), \sum_{s\in\mathsf{T}} \mathbf{e}_s \rangle$. We denote with $\texttt{OPT}$ the optimal value attained by the function $f(\cdot)$ as in (4.2). We denote with $\mathscr{I}$ the p-system side constraint, and with r its rank, as defined in Section 4.2.1. The notation $\mathrm{Cond}(\mathsf{T})$ denotes the set $\{s \in [n] \setminus \mathsf{T} \colon \mathsf{T} \cup \{s\} \in \mathscr{I}\}$.

4.2.1 Setup

We study optimization tasks as in Problem (4.2) under some additional assumptions on $l(\cdot)$ which are often satisfied in practical applications (see Section 4.2.3). These assumptions hold on r-sparse subdomains.

▶ **Definition 4.1 (Sparse Subdomain).** A r-sparse subdomain is a set of the form $\Omega_r := \{(\boldsymbol{x}, \boldsymbol{y}) \in \mathbb{R}^n \times \mathbb{R}^n \colon \|\boldsymbol{x}\|_0 \le r, \|\boldsymbol{y}\|_0 \le r, \|\boldsymbol{x}-\boldsymbol{y}\|_0 \le r\}$. ◀

Using this definition, we can define the notions of strong concavity and smoothness, as follows.

▶ **Definition 4.2 (Restricted Strong Concavity, Restricted Smoothness).** A function $l(\cdot)$ is said to be restricted strong concave with parameter m and restricted smooth with parameter M on a subdomain Ω_r iff, for all $(\boldsymbol{x}, \boldsymbol{y}) \in \Omega_r$ it holds that

$$-\frac{m}{2}\|\boldsymbol{y}-\boldsymbol{x}\|_2 \ge l(\boldsymbol{y}) - l(\boldsymbol{x}) - \langle \nabla l(\boldsymbol{x}), \boldsymbol{y}-\boldsymbol{x}\rangle \ge -\frac{M}{2}\|\boldsymbol{y}-\boldsymbol{x}\|_2.$$

◀

We say that l is (M, m)-(Smooth, Restricted Concave), if it fulfills the conditions as in Definition 4.2 with parameters M and m.

We are specifically interested in the problem of maximizing a function $l(\cdot)$ that is restricted strong concave and restricted smooth under additional side constraints, called p-systems. These side constraints are very general, and they can be used to encode several notions of fairness (see Section 4.2.4 for details). p-systems are defined in Section 3.2.1. However, we re-state the definition here. p-systems are collections of subsets of the ground set $[n]$, that fulfill additional properties. In order to give an axiomatic definition of these properties, we introduce additional terminology. Given a collection of feasible solutions $\mathscr{I}$ over a ground set V and a set $T \subseteq V$, we denote with $\mathscr{I}|_T$ a collection consisting of all sets $S \subseteq T$ that are feasible in $\mathscr{I}$. Furthermore, a base for $\mathscr{I}$ is any maximum feasible set $U \in \mathscr{I}$.

▶ **Definition 4.3 (p-Systems).** A p-system $\mathscr{I}$ over $[n]$ is a collection of subsets of $[n]$ such that

- $\emptyset \in \mathscr{I}$;
- for any two sets $\mathsf{S} \subseteq \mathsf{T} \subseteq [n]$, if $\mathsf{T} \in \mathscr{I}$ then $\mathsf{S} \in \mathscr{I}$;
- for any set $\mathsf{T} \subseteq [n]$ and any bases $\mathsf{S}, \mathsf{U} \in \mathscr{I}|_{\mathsf{T}}$ it holds $|\mathsf{S}| \leq p|\mathsf{U}|$.

◀

The second defining axiom is referred to as subset-closure or downward-closed property. We define the rank r of a p-system $\mathscr{I}$ as the maximum cardinality of any feasible solution $\mathsf{T} \in \mathscr{I}$. We define $\mathrm{Cond}(\mathsf{T})$ as the set of all points $s \in [n] \setminus \mathsf{T}$ such that $\mathsf{T} \cup \{s\} \in \mathscr{I}$. Note that a set T is a maximum independent set if it holds $\mathrm{Cond}(\mathsf{T}) = \emptyset$.

Armed with these definitions, we can re-visit Problem 4.2, where the set of feasible solutions $\mathscr{I}$ is a p-system, and $l(\cdot)$ is restricted strong concave and smooth under a *r-sparsity constraint*, is a special case of Problem 4.2, by setting $\mathscr{I} := \{\mathsf{T} \subseteq [n] : |\mathsf{T}| \leq r\}$. Hence, our work

generalizes previous related works (Chierichetti et al. 2019; Elenberg, Khanna, et al. 2018; Sakaue 2020).

We assume access to an oracle that returns $\nabla l(\beta^{(\mathsf{T})})$, for a given input set T, and we assume access to the *independence oracle* of the underlying p-system $\mathscr{I}$. The independence oracle takes as input a set T, and returns as output a Boolean value, true if $\mathsf{T} \in \mathscr{I}$ and false otherwise.

4.2.2 Relationship to Generalized Submodularity

Consider a function l that is restricted smooth and restricted strong concave . It is well known that the corresponding function f as in (4.1) has weak diminishing return properties. The first of these diminishing returns properties is called *weak submodularity*, and it was first introduced by (Das and Kempe 2011), to study statistical subset selection problems. The second property is called *differential submodularity*. Differential submodularity was introduced by (S. Qian and Singer 2019) to study a fast algorithm for statistical subset selection in the adaptive query model. Differential submodularity generalizes the notion of submodularity, but it is less general than the notion of weak submodularity.

Weak submodularity. Functions that exhibit weak submodularity, i.e., weakly submodular functions, are defined in terms of the submodularity ratio, as follows.

▶ **Definition 4.4 (Weak submodularity, Definition 2.3 of (Das and Kempe 2011)).** Let $\mathsf{L}, \mathsf{S} \subseteq [n]$ be two disjoint set, and consider a function $f\colon 2^{[n]} \to \mathbb{R}_{\geq 0}$. The submodularity ratio of L w.r.t S is defined as

$$\gamma_{\mathsf{L},\mathsf{S}} := \frac{\sum_{j\in\mathsf{S}}[f(\mathsf{L}\cup\{j\}) - f(\mathsf{L})]}{f(\mathsf{L}\cup\mathsf{S}) - f(\mathsf{L})}.$$

The submodularity ratio $\gamma_{\mathsf{U},k}$ of a set $\mathsf{U} \subseteq [n]$ with respect to k is

defined as

$$\gamma_{U,k} := \min_{\{L,S:\, L\cap S\neq\emptyset, L\subseteq U, |S|\leq k\}} \gamma_{L,S}.$$

We say that f is γ- weakly submodular if it holds $\gamma_{U,k} \geq \gamma$ for all sets $U \subseteq [n]$. ◀

There is a well-known connection between weak submodularity and Problem 4.2. This connection was discovered by (Elenberg, Khanna, et al. 2018), and it can be formalized as follows.

▶ **Theorem 4.2 (Theorem 1 by (Elenberg, Khanna, et al. 2018)).** Define f as in (4.1), with a function l that is $(m_{|U|+k}, M_{|U|+k})$-(strongly concave, smooth) on $\Omega_{|U|+k}$, and $\tilde{M}_{|U|+1}$ smooth on $\Omega_{|U|+1}$. Then, the submodularity ratio $\gamma_{U,k}$ of f is lower-bounded as

$$\gamma_{|U|,k} \geq \frac{m_{|U|+k}}{\tilde{M}_{|U|+1}} \geq \frac{m_{|U|+k}}{M_{|U|+k}}.$$

◀

Differential submodularity. Differential submodularity is a notion of intermediate generality. That is, all submodular functions are differentially submodular, but not all weakly submodular functions are differentially submodular. This definition is useful for studying algorithms in the adaptive query model. Differential submodularity is defined as follows.

▶ **Definition 4.5 (Differential submodularity, Definition 1 by (S. Qian and Singer 2019)).** A function $f\colon 2^{[n]} \to \mathbb{R}_{\geq 0}$ is α-differentially submodular, if there exists two submodular functions h, g s.t. for any subsets $L, S \subseteq [n]$ it holds $g_S(L) \leq f_S(L) \leq h_S(L)$, and $g_S(L) \geq \alpha h_S(L)$. ◀

Essentially, a function f is differentially submodular, if its marginal contributions can be bounded by submodular functions. It is well-known that restricted strong concavity and smoothness imply differential submodularity. The following theorem holds.

▶ **Theorem 4.3 (Theorem 6 by (S. Qian and Singer 2019)).** Define f as in (4.1), with a function l that is (m, M)-(strongly concave, smooth) on Ω_{2r}. Then, the objective f is differentially submodular, such that

$$\frac{m_{|S|+1}}{M_{|S|+k}}\overline{f}_S(T) \leq f_S(T) \leq \frac{M_{|S|+1}}{m_{|S|+k}}\overline{f}_S(T),$$

with $\overline{f}_S(T) = \sum_{s\in T} f_S(T)$. ◀

4.2.3 Feature Selection for Generalized Linear Models

Given a set of observations and a parametric family of distributions $\{p(\cdot\,;\beta) \mid \beta \in \Omega\}$ with $\Omega \subseteq \mathbb{R}^n$, we wish to identify a vector of parameters β maximizing the goodness of fit for these observation, according to a chosen measure l. For generalized linear models, common measures for feature selection are restricted strong concave and restricted smooth. We study the log-likelihood and the coefficient of determination, although analogous results hold for other similar statistics (Das and Kempe 2011; S. Qian and Singer 2019).

Maximizing the log-conditional. We first discuss results concerning the (M, m)-(smotheness, strong concavity) of the log-conditional for generalized linear models. Consider a feature matrix $\mathbf{X} \in \mathbb{R}^{n\times d}$ and response variable $\mathbf{y}$. Assuming that the response follows a distribution in an exponential family, the log-conditional can be written as

$$\log p(\mathbf{y} \mid \mathbf{X}; \beta) = h^{-1}(\tau) - Z(\mathbf{X}, \beta) + g(\mathbf{y}, \tau) \tag{4.4}$$

with Z the log-partition function, and τ the dispersion parameter. We give approximation guarantees for the objective

$$l(\beta) = \log p(\mathbf{y} \mid \mathbf{X}; \beta) - \eta\|\beta\|_2^2, \tag{4.5}$$

for some parameter $\eta \geq 0$. We show that the function l is restricted smooth and restricted strong concave under various assumptions. We start discussing the simplest case of a logistic regression. We then study the general case as in (4.4), under the assumption that l has a non-zero regularization term. We then conclude with the general case, i.e., no assumptions on the regularization term.

The log-likelihood function for the logistic regression is defined as

$$l(\boldsymbol{\beta}^{(\mathsf{S})}) := \sum_i y_i \langle \mathbf{X}_\mathsf{S}, \boldsymbol{\beta} \rangle - \log\left(1 - e^{\langle \mathbf{X}_\mathsf{S}, \boldsymbol{\beta} \rangle}\right), \tag{4.6}$$

with $\mathbf{X}_\mathsf{S}$ the matrix of all features indexed by S and y_i the i-th observation, i.e., the i-th coefficient of the response $\boldsymbol{y}$. For this class of log-likelihood functions, the following result holds.

▶ **Proposition 4.1.** The log-likelihood function l for the logistic regression as in (4.6), is (m, M)-(restricted smooth, restricted strong concave) with parameters

$$m := \min_\mathsf{S} \lambda_{\min}(\mathbf{X}_\mathsf{S}^T \mathbf{X}_\mathsf{S}) \text{ and } M := \max_\mathsf{S} \lambda_{\max}(\mathbf{X}_\mathsf{S}^T \mathbf{X}_\mathsf{S}).$$

◀

We now study log-conditional functions as in (4.4), under the assumption that the corresponding objective l has a regularization term with parameter $\eta > 0$. We introduce additional notation to this end. For any feature set $\mathsf{T} \subseteq [n]$, denote with $\mathscr{P}_\mathsf{T}$ an operator that takes as input vectors $\mathbf{x} \in \mathbb{R}^n$, and it replaces all indices in $[n] \setminus \mathsf{T}$ of $\mathbf{x}$ by 0. For any vector $\mathbf{x} \in \mathbb{R}^p$, we define $\mathbf{x}_\mathsf{T} := \mathscr{P}_\mathsf{T}(\mathbf{x})$. We consider the following assumption on the distribution of the features (see (Bahmani et al. 2013)).

For fixed constants $r, R > 0$, we make the following assumption on the feature matrix $\mathbf{X} \in \mathbb{R}^{n \times p}$. The rows $\mathbf{x}$ of $\mathbf{X}$ are generated $i.i.d.$, such that the following additional conditions hold. For any set $\mathsf{T} \subseteq [p]$ of size $|\mathsf{T}| \leq r$,

- $\|\mathbf{x}_\mathsf{T}\|_2 \leq R$;
- none of the matrices $\mathscr{P}_\mathsf{T}\mathbb{E}\left[\,\mathbf{x}\mathbf{x}^T\,\right]\mathscr{P}_\mathsf{T}$ are the zero matrix.

Following this notation, define

$$\phi_{\max} := \max_{|\mathsf{T}|\leq k} \lambda_{\max}(\mathscr{P}_\mathsf{T}\mathbf{C}\mathscr{P}_\mathsf{T}) \qquad \phi_{\min} := \min_{|\mathsf{T}|\leq k} \lambda_{\max}(\mathscr{P}_\mathsf{T}\mathbf{C}\mathscr{P}_\mathsf{T}),$$

with $\lambda_{\max}(\cdot)$ the largest eigenvalue. The following corollary holds.

▶ **Proposition 4.2 (Corollary 4 by (Elenberg, Khanna, et al. 2018)).** Consider a function l as in (4.5), and suppose that $\eta > 0$. Suppose that the aforementioned assumption holds with parameters r, R, and suppose that the number of samples s is lower-bounded as

$$s > \frac{R(\log r + r(1 + \log\frac{n}{k} - \log\delta))}{\phi_{\min}(1+\varepsilon)\log(1+\varepsilon) - \varepsilon}.$$

Then, with probability at least $1-\delta$ the function l is (m, M)-(smooth, restricted concave), for all β with at most r non-zero coefficients. The parameters m and M are defined as $m = q(1+\varepsilon)\phi_{\max} + \eta$ and $M = \eta$, with q a constant fulfilling $q \geq \max_i h^{-1}(\tau)Z''(\beta, \mathbf{x}_i)$ for $h^{-1}(\cdot), Z(\cdot,\cdot)$ as in (4.4). ◀

We conclude by studying log-conditional functions as in (4.4), with no additional assumption on the regularization term. For simplicity, we consider functions l as in (4.4) with $\eta = 0$. However, these results can easily be extended to the general case. The following lemma holds.

▶ **Lemma 4.2 (Corollary 2 by (Elenberg, Khanna, et al. 2018)).** Denote with r an upper-bound on the sparsity of the feature sets. Following the notation introduced above, suppose that the feature matrix $\mathbf{X}$ consists of samples drawn from a sub-Gaussian distribution with parameter σ^2 and covariance matrix Σ. Then, for $\eta = 0$ the function l

as in (4.5) is (M, m)-(smooth, restricted concave) with parameters

$$M = \alpha_u \lambda_{\max}(\Sigma)\left(3 + \frac{2nr}{s}\right) \quad \text{and} \quad m = \alpha_\ell - \frac{c^2\sigma^2}{\alpha_\ell}\frac{k\log n}{s},$$

with high probability, for $s > 0$ sufficiently large. The constant α_ℓ depends on (σ^2, Σ) and k; the constant α_u yields

$$\alpha_u \geq \max_i h^{-1}(\tau)^{-1} Z''(\beta, \mathbf{x}_i),$$

with $h^{-1}(\cdot), Z(\cdot,\cdot)$ as in (4.4). ◀

Maximizing the R^2 objective. Consider a linear model with a normalized response variable $\mathbf{y}$. The R^2 objective is defined as

$$R^2_{\mathbf{X},\mathbf{y}}(\beta) := 1 - \mathbb{E}\left[\ (\mathbf{y} - \langle \mathbf{X}, \beta\rangle)^2\ \right]. \tag{4.7}$$

This function is a popular measure for the goodness of fit. The function $R^2_{\mathbf{X},\mathbf{y}}(\beta)$ is restricted strong concave and restricted smooth, with parameters depending on the properties of the matrix $\boldsymbol{X}$. Specifically, given a feature matrix consisting of n features and k observations, the R^2 objective with regularization can be written as

$$l(\beta) = R^2_{\mathbf{X},\mathbf{y}}(\beta) = 1 - \frac{1}{k}\|(\mathbf{y} - g(\langle \mathbf{X}, \beta\rangle)\|_2^2. \tag{4.8}$$

The following lemma, similar to Lemma 4.1, holds.

▶ **Proposition 4.3.** The R^2 objective l for the linear regression as in (4.8), is (m, M)-(restricted smooth, restricted strong concave) with parameters

$$m := \min_S \lambda_{\min}(\mathbf{X}_S^T \mathbf{X}_S) \text{ and } M := \max_S \lambda_{\max}(\mathbf{X}_S^T \mathbf{X}_S).$$

◀

This lemma can easily be extended to the case of an R^2 objective with regularization term.

4.2.4 Embedding Fairness via p-Systems

In this section, we describe how certain notions of fairness can be embedded as p-Systems.We consider *procedural fairness* to this end. Procedural fairness selects input features that are used in the decision process based on factors such as volitionality, reliability, privacy, and relevance (Beahrs 1991; Kilbertus et al. 2017; Kusner et al. 2017; Trankell 1973). In this work, we consider measures for procedural fairness studied by (Grgic-Hlaca, Zafar, K. Gummadi, et al. 2016) and (Grgic-Hlaca, Zafar, K. P. Gummadi, et al. 2018). However, our analysis is not specific to these definitions. For a given model, features are selected based on the moral judgement of a set of users $\mathcal{U}$.

The moral judgement of these users defines measures of unfairness as follows. Denote with $\mathcal{U}_s^A$ the set of all users that considers feature s fair to use, if it increases the accuracy of a model. The feature-accuracy unfairness is defined as

$$\mathcal{I}_{\text{acc}}^{\delta} := \left\{ \mathsf{T} \in [n] \colon 1 - \frac{\left| \bigcap_{s \in \mathsf{T}} g(\mathcal{U}_s^A, \mathcal{U}_s) \right|}{|\mathcal{U}|} \leq \delta \right\} \tag{4.9}$$

Here, $\delta > 0$ is a user-defined parameter, which determines the trade-off between fairness and accuracy, and $g(\mathcal{U}_s^A, \mathcal{U}_s)$ is a function that outputs $\mathcal{U}_s^A$ if s increases accuracy, and $\mathcal{U}_s$ otherwise. Note that this collection of sets is a p-system as in Definition 4.3. Note also that evaluations of g require to train the model. We remark that there exist other definitions for procedural fairness in the literature (Grgic-Hlaca, Zafar, K. Gummadi, et al. 2016; Grgic-Hlaca, Zafar, K. P. Gummadi, et al. 2018).

Several other quantitative definitions of fairness can be embedded as p-Systems, via the reduction-based approach proposed by (Agar-

wal, Beygelzimer, et al. 2018; Agarwal, Dudík, et al. 2019). This is true, for example, for Statistical Parity (Agarwal, Beygelzimer, et al. 2018), Equalized Odds (Hardt et al. 2016), and Equality of Opportunity (Hardt et al. 2016; Kleinberg et al. 2017). However, these definitions can be incorporated into the learning model directly (Donini et al. 2018; Woodworth et al. 2017; Zafar, Valera, Gomez-Rodriguez, and K. P. Gummadi 2017b), or by implementing wrappers around the classifier (Hardt et al. 2016; Pin Calmon et al. 2017).

4.2.5 Adaptivity

We evaluate performance using the notion of adaptivity. This notion was defined in Section 3.2.3. However, we discuss this notion again in this section. Formally, we can define adaptivity as follows.

▶ **Definition 4.6 (Adaptivity (Balkanski and Singer 2018)).** Given an oracle f, an algorithm is r-adaptive if every query q to the oracle f occurs at a round $i \in [r]$ such that q is independent of the answers $f(q')$ to all other queries q' at round i. ◀

We evaluate empirical speedup by the adaptivity of the oracle to evaluate $\nabla l(\beta^{(\mathsf{T})})$. We also evaluate the adaptivity of the independence oracle for the p-system $\mathscr{I}$.

4.3 Algorithmic Overview

Our algorithm, which we call FASTOMP, is displayed in Algorithm 6. This algorithm is based on a technique called adaptive sequencing (Balkanski, Rubinstein, et al. 2019; Breuer et al. 2020), which was recently proposed for maximizing submodular functions.

Say X is the complete set of candidate features, and S is the current solution. Starting from $\mathsf{S} \leftarrow \emptyset$, the FASTOMP iteratively generates a

Algorithm 6: FASTOMP

$\mathsf{S} \leftarrow \emptyset$;
while *the number of iterations is less than* ε^{-1} *and Cond*$(\mathsf{S}) \neq \emptyset$ **do**
  $\mathsf{X} \leftarrow \{s \in [n] : \{s\} \in \mathscr{I}\}$;
  $t \leftarrow (1-\varepsilon)\frac{m}{|\mathsf{T}|M}\|\nabla l(\beta^{(\mathsf{S})})_{\mathsf{T}}\|_2^2$ with $\mathsf{T} \subseteq \mathsf{X}$ maximizing $\|\nabla l(\beta^{(\mathsf{S})})_{\mathsf{T}}\|_2^2$ s.t. $|\mathsf{T}| \leq r$;
  while $\mathsf{X} \neq \emptyset$ *and Cond*$(\mathsf{S}) \neq \emptyset$ **do**
    $\{a_1, a_2, \ldots, a_k\} \leftarrow$ RNDSEQ(X, S) and define $\mathsf{S}_j \leftarrow \mathsf{S} \cup \{a_1, \ldots, a_j\}$;
    observe $\mathsf{X}_j \leftarrow \{s \in \mathsf{X} : \langle \nabla l(\beta^{(\mathsf{S}_j)}), \mathbf{e}_s\rangle^2 \geq t \text{ and } s \in \mathrm{Cond}(\mathsf{S}_j)\}$;
    $j^* \leftarrow \min_j\{|\mathsf{X}_j| < (1-\varepsilon)|\mathsf{X}|\}$;
    $\mathsf{X} \leftarrow \mathsf{X}_{j^*}$ and $\mathsf{S} \leftarrow \mathsf{S}_{j^*}$;
return S;

random sequence of features $\{a_1, a_2, \ldots, a_k\}$ with the RNDSEQ sub-routine, such that the set $\{a_1, a_2, \ldots, a_k\} \cup \mathsf{S}$ is a maximum independent set of $\mathscr{I}$. After a sequence is generated, the FASTOMP identifies a prefix $\{a_1, \ldots, a_{j^*}\}$ that is added to the current solution. The index j^* defining this prefix is chosen such that it holds $|\mathsf{X}_{j^*}| \geq (1-\varepsilon)|\mathsf{X}|$, for all $j < j^*$. This inequality ensures that any point added to the current solution yields $\langle \nabla l(\beta^{(\mathsf{S})}), \mathbf{e}_s\rangle^2 \geq t$ in expected value. Finally, the ground set X is updated as to include only those points that yield a good improvement to the new solution. The RNDSEQ sub-routine used to generate $\{a_1, a_2, \ldots, a_k\}$ corresponds to Algorithm A by (Karp et al. 1988), and it is presented in Algorithm 7. Here, k is the size of the independent set returned in the current iteration.

Given as input a ground set X, a current solution S, and a p-system $\mathscr{I}$, this algorithm finds a random set A such that $\mathsf{S} \cup \mathsf{A}$ is a maximum

Algorithm 7: RNDSEQ(X, S)

```
A ← ∅;
while X ≠ ∅ do
    sort the points {x_i}_i = X uniformly at random;
    j* ← max{j : S ∪ A ∪ {x_i}_{i≤j} ∈ 𝓘};
    A ← A ∪ {x_1, ..., x_{j*}};
    X ← {e ∈ X \ (S ∪ A) : S ∪ A ∪ e ∈ 𝓘};
return A;
```

independent set for $\mathscr{I}$. This algorithm iteratively shuffles the set X, and then it identifies the longest prefix of this sequence that can be added to S, without violating side constraints. This prefix is then added to A. This algorithm terminates when $S \cup A \in \mathscr{I}$ is a maximal independent set.

This algorithm uses parallel calls to the independence oracle of $\mathscr{I}$, since the evaluations for the feasibility of prefixes of A can be preformed in parallel. Hence, with this algorithm the adaptivity of the independence oracle corresponds to the number of iterations until convergence. It is well-known that this algorithm converges after expected $\mathcal{O}(\sqrt{r})$ iterations, with r the rank of $\mathscr{I}$ (see Theorem 6 by (Karp et al. 1988)). This implies that the adaptivity of the independence oracle is $\mathcal{O}(\sqrt{r})$. Although it is not known if this upper-bound on the adaptivity is tight, it is known that there is no algorithm that finds a maximum independent set of $\mathscr{I}$ with less then $\tilde{\Omega}(n^{1/3})$ rounds (see Theorem 7 by (Karp et al. 1988)).

Adaptive sequencing via matching projections. Our proposed algorithm differs from previous related work on adaptive sequencing, since it does not use queries to the function f as in (4.1) when selecting features. Instead, our algorithm uses oracle access to the function $\nabla l(\mathcal{B}^{(S)})$, and features $s \in [n]$ are added to the current solution if it

holds $\langle \nabla l(\boldsymbol{\beta}^{(\mathsf{S}_j)}), \mathbf{e}_s \rangle \geq t$ in expected value, with t a threshold updated during run time. As such, our approach extends the applicability of adaptive sequencing to gradient based pursuit methods as opposed to the standard value-oracle based methods in earlier works. Finding optimal solutions with this technique requires much less computation than the previous approach (Balkanski, Rubinstein, et al. 2019), by which points s are selected if it holds $f(\mathsf{S}_j \cup \{s\}) \geq t$. It is usually much faster to compute an inner product, than the regression score expressed by the function $f(\cdot)$.

Implicit estimates of $\mathtt{OPT}$. All algorithms based on adaptive sampling techniques require an estimate of $\mathtt{OPT}$, which is typically not known a priori. To circumvent this problem, usually multiple passes of the same algorithm are performed for various guesses of $\mathtt{OPT}$ (Breuer et al. 2020), or an additional preprocessing steps are performed (Fahrbach et al. 2019b). Our algorithm has the significant advantage that $\mathtt{OPT}$ is estimated implicitly. Specifically, for a function $l(\cdot)$ that is (m, M)-(smooth, restricted concave), we have

$$\max_{\{\mathsf{T}\colon |\mathsf{T}| \leq k\}} \|\nabla l(\boldsymbol{\beta}^{(\mathsf{S})})_{\mathsf{T}}\|_2^2 \geq 2m\big(\mathtt{OPT} - f(\mathsf{S})\big),$$

with f as in (4.1). A proof of this result is deferred to Appendix 4.5. Hence, the FAST$_{\text{OMP}}$ estimates $\mathtt{OPT}$ with a single oracle valuation, and no multiple runs or preprocessing are required to achieve a good solution quality.

Finding the set X_{j^*}. Although monotone, common optimization functions $l(\cdot)$ for feature selection do not have certain desirable properties, such as an antitone gradient, which were helpful in previous analyses of similar algorithms. Hence, the sequence $\{|\mathsf{X}_j|\}_j$ as in Line 7 of Algorithm 6 is not monotonic, in general. For this reason, it is not possible to estimate the optimal index j^* with a binary search, as

other adaptive sequencing algorithms do (Breuer et al. 2020). However, our algorithm requires $\mathcal{O}(r)$ total phases of re-training to estimate j^*, whereas the general adaptive sequencing technique would require $\mathcal{O}(r\,n)$ oracle queries to evaluate j^* (Balkanski, Rubinstein, et al. 2019). Here, r is the rank of the side constraint for the features, and n is the problem size.

Sampling random sequences of features. The RAND-SEQUENCE algorithm correspond to Algorithm A by (Karp et al. 1988). To our knowledge, no other algorithm is known for this problem, with better adaptivity than Algorithm 7. Here, k is the size of the independent set returned in the current iteration.

Parameter tuning. If l is the log-likelihood of a linear model, or the R^2 objective of a linear model with normalized response variable, then m and M can be estimated in terms of the design of the feature matrix. These estimates need not be tight, and certifying bounds for m and M is NP-hard (Bandeira et al. 2013). In general, the constant m/M in Line 4 of Algorithm 6 requires tuning by making multiple runs of the algorithm. We remark that other known parallel algorithms for feature selection also require estimates of these parameters (S. Qian and Singer 2019).

Parallel algorithms that estimate the rank are known for several p-systems. For instance, the rank of a graphic matroid can be estimated with parallel algorithms that compute spanning trees. Furthermore, parallel rank oracles are known for matroids that can be represented as independent sets of vectors in a given field (Borodin et al. 1982; Chistov 1985; Ibarra et al. 1980; Mulmuley 1987). These algorithms can also be used to estimate the rank of more complex constraints, such as the intersection of matroids or p-matchoids.

4.4 Approximation Guarantees

In this section, we study the approximation guarantees for Algorithm 6, when solving the Problem (4.2).

▶ **Theorem 4.4.** Define the support selection function $f(\cdot)$ as in (4.1), for the given function $l(\cdot)$ that is (M,m)-(restricted smooth, restricted strong concave), on the sparse sub-domain Ω_{2r}. Consider a p-system $\mathscr{I}$ of rank r over $[n]$, and let S^* be the output of Algorithm 6 while $\mathtt{OPT}$ is the optimum solution set for the Problem 4.2. Then,

$$\frac{\mathbb{E}\big[\, f(\mathsf{S}^*)\,\big]}{\mathtt{OPT}} \geq \frac{1}{1+p}\left(1-\exp\left\{-(1-\varepsilon)^2\frac{m^3}{M^3}\right\}\right),$$

for all $0<\varepsilon<1$. Furthermore, in the specific case when $\mathscr{I}$ is r-sparsity constraint over $[n]$, then,

$$\frac{\mathbb{E}\big[\, f(\mathsf{S}^*)\,\big]}{\mathtt{OPT}} \geq \left(1-\exp\left\{-(1-\varepsilon)^2\frac{m^2}{M^2}\right\}\right).$$

◀

A full proof of this theorem is deferred to Section 4.5. We remark that, if $\mathscr{I}$ is a r-sparsity constraint, then the approximation guarantee of Theorem 4.4 is asymptotically better than the guarantee attained by other parallel algorithms for this problem, such as the DASH (S. Qian and Singer 2019). Specifically, as proven in Theorem 1 by (S. Qian and Singer 2019), the DASH yields an approximation of $1-\exp\{m^4/M^4\}-\varepsilon$ on this problem. Furthermore, the DASH cannot handle general side constrains as in Problem 4.2.

We also provide bounds for the run time of the FAST$_{\text{OMP}}$ as follows.

▶ **Theorem 4.5.** Algorithm 6 terminates after $\mathcal{O}\big(\varepsilon^{-2}\log n\big)$ rounds of calls to the oracle function, and it uses at most $\mathcal{O}\big(\varepsilon^{-2} r\log n\big)$ oracle queries. Furthermore, Algorithm 6 requires expected $\mathcal{O}\big(\varepsilon^{-2}\sqrt{r}\log n\big)$

independent calls to the oracle for the p-system $\mathscr{I}$, and the total expected calls to the oracle for the p-system is $\mathcal{O}\left(\varepsilon^{-2} n r \log n\right)$. ◀

The proof of Theorem 4.5 is deferred to Appendix 4.6. The estimates on the rounds of adaptivity extends to the PRAM model. If we denote with d_l the depth required to evaluate the oracle function on a set, then the FASTOMP has $\mathcal{O}\left(\varepsilon^{-2} d_l \log n\right)$ depth. Note that the rounds of independent calls to the oracle are sub-linear, but not poly-logarithmic in the problem size. The reason is that the RNDSEQ sub-routine requires expected $\mathcal{O}\left(\sqrt{n}\right)$ rounds of independent calls to the oracle for the p-system.

If $\mathscr{I}$ is an r-sparsity constraint, then there exist oblivious algorithms for Problem 4.2, that require $\mathcal{O}(1)$ sequential oracle calls (Elenberg, Khanna, et al. 2018; Sakaue 2020). These algorithms select r best points $s \in [n]$ independently, according to the values $f(\{s\})$, or the value of the inner product $\langle \nabla l(\mathbf{0}), \mathbf{e}_s \rangle$. However, any extension of these techniques to general p-system side constraints would require $\Omega(r)$ sequential calls to the independence oracle.

4.5 Proof of Theorem 4.4

We prove the following theorem.

▶ **Theorem 4.4.** Define the support selection function $f(\cdot)$ as in (4.1), for the given function $l(\cdot)$ that is (M,m)-(restricted smooth, restricted strong concave), on the sparse sub-domain Ω_{2r}. Consider a p-system $\mathscr{I}$ of rank r over $[n]$, and let S^* be the output of Algorithm 6 while $\mathtt{OPT}$ is the optimum solution set for the Problem 4.2. Then,

$$\frac{\mathbb{E}\left[\, f(\mathsf{S}^*) \,\right]}{\mathtt{OPT}} \geq \frac{1}{1+p}\left(1 - \exp\left\{-(1-\varepsilon)^2 \frac{m^3}{M^3}\right\}\right),$$

for all $0 < \varepsilon < 1$. Furthermore, in the specific case when $\mathscr{I}$ is r-sparsity

constraint over $[n]$, then,

$$\frac{\mathbb{E}\left[\, f(\mathsf{S}^*)\,\right]}{\mathtt{OPT}} \geq \left(1 - \exp\left\{-(1-\varepsilon)^2 \frac{m^2}{M^2}\right\}\right).$$

◀

The proof of this theorem is based on a few lemmas and propositions, which we discuss in Appendix 4.5.1 before proving Theorem 4.4. On a high level, the proof of Theorem 4.4 is split into two separate cases. First we prove that Theorem 4.4 holds when Algorithm 6 terminates after ε^{-1} iterations of the outer While-loop of Algorithm 6. Then, we prove Theorem 4.4 under the assumption that Algorithm 6 finds a solution of size k. The first part of the proof is discussed in Appendix 4.5.3 (see Theorem 4.8), and the second case is discussed in Appendix 4.5.2 (see Theorem 4.7).

4.5.1 Preliminary Results

Our analysis is based on a few preliminary result, which we discuss in this section.

▶ **Theorem 4.6.** Suppose the l is (M,m)-(smooth, strongly concave) on Ω_{2k}. Then, for each subsets $\mathsf{S},\mathsf{T} \subseteq [p]$ of size at most k it holds

$$2M \sum_{j\in\mathsf{T}} f_{\mathsf{S}}(j) \geq \|\nabla l(\boldsymbol{\beta}^{(\mathsf{S})})_{\mathsf{T}}\|_2^2 \geq 2m \sum_{j\in\mathsf{T}} f_{\mathsf{S}}(j).$$

◀

Proof. We start proving the first inequality. Fix a point $j \in \mathsf{T}$. Then, for any scalar α it holds

$$\begin{aligned} f_{\mathsf{S}}(j) = l(\boldsymbol{\beta}^{(\mathsf{S}\cup\{j\})}) - l(\boldsymbol{\beta}^{(\mathsf{S})}) &\geq l(\boldsymbol{\beta}^{(\mathsf{S})} + \alpha\mathbf{e}_j) - l(\boldsymbol{\beta}^{(\mathsf{S})}) \\ &\geq \langle \nabla l(\boldsymbol{\beta}^{(\mathsf{S})}), \alpha\mathbf{e}_j\rangle - \frac{M}{2}\alpha^2, \end{aligned} \tag{4.10}$$

where the first inequality uses the maximality of $\beta^{(S\cup\{j\})}$, and the second on uses the restricted smoothness. By substituting $\alpha = \frac{1}{M}\langle \nabla l(\beta^{(S)}), \mathbf{e}_j\rangle$ in (4.10), we get

$$2M \sum_{j\in T} f_S(j) \geq \sum_{j\in T} \left\langle \nabla l(\beta^{(S)}), \mathbf{e}_j \right\rangle^2 = \|\nabla l(\beta^{(S)})_T\|_2^2,$$

ant the first inequality follows. To conclude the proof, note that it holds

$$\begin{aligned} f_S(j) &= l(\beta^{(S\cup\{j\})}) - l(\beta^{(S)}) \\ &\leq \langle \nabla l(\beta^{(S)}), \beta^{(S\cup\{j\})} - \beta^{(S)}\rangle - \frac{m}{2}\|\beta^{(S\cup\{j\})} - \beta^{(S)}\|_2^2 \\ &\leq \max_{\mathbf{v}:\, \mathbf{v}_{(S\cup\{j\})=0}} \langle \nabla l(\beta^{(S)}), \mathbf{v} - \beta^{(S)}\rangle - \frac{m}{2}\|\mathbf{v} - \beta^{(S)}\|_2^2, \end{aligned} \tag{4.11}$$

where the first inequality uses the restricted strong concavity, and the second one uses the maximality of $\mathbf{v}$. By setting $\mathbf{v} = \beta^{(S)} + \frac{1}{m}\langle \nabla l(\beta^{(S)}), \mathbf{e}_j\rangle$ in (4.11) we get

$$\frac{\|\nabla l(\beta^{(S)})_{S^*}\|_2^2}{2m} \geq l(\beta^{(S\cup\{j\})}) - l(\beta^{(S)}) = f_S(j).$$

By taking the sum over all $j \in T$ and rearranging we get

$$\|\nabla l(\beta^{(S)})_T\|_2^2 = \sum_{j\in T} \frac{\|\nabla l(\beta^{(S)})_{S^*}\|_2^2}{2m} \geq \sum_{j\in T} f_S(j),$$

and the claim follows. ■

In order to perform the analysis, we also use the differential submodularity property of f, which we discussed in Appendix 4.2.2.

▶ **Theorem 4.3 (Theorem 6 by (S. Qian and Singer 2019)).** Define f as in (4.1), with a function l that is (m, M)-(strongly concave, smooth)

on Ω_{2r}. Then, the objective f is differentially submodular, such that

$$\frac{m_{|S|+1}}{M_{|S|+k}}\overline{f}_S(T) \le f_S(T) \le \frac{M_{|S|+1}}{m_{|S|+k}}\overline{f}_S(T),$$

with $\overline{f}_S(T) = \sum_{s\in T} f_S(T)$. ◀

Using Theorem 4.6 and Theorem 4.3, we get the following corollary.

▶ **Corollary 4.1.** Suppose the l is (M,m)-(smooth, strongly con-cave) on Ω_{2k}. Then, there exists a monotone submodular function $\bar{f}$ such that

$$\frac{M}{m}\bar{f}_S(T) \ge f_S(T) \ge \frac{m}{M}\bar{f}_S(T),$$

and

$$2M\bar{f}_S(T) \ge \|\nabla l(\beta^{(S)})_T\|_2^2 \ge 2m\bar{f}_S(T),$$

for each subsets $S, T \subseteq [p]$ of size at most k. ◀

In our analysis, we use the following technical proposition.

▶ **Proposition 4.4 (Proposition 2.2. by (Nemhauser, Wolsey, and M. L. Fisher 1978)).** Consider two sequences of non-negative real numbers $\{x_1,\dots,x_m\}$ and $\{y_1,\dots,y_m\}$. Suppose that it holds $\sum_j x_j \le j$ for all $j \in [m]$, and $y_j \ge y_{j+1}$ for all $j \in [m-1]$. Then,

$$\sum_{j=1}^{m} y_j \ge \sum_{j=1}^{m} x_j y_j.$$

◀

4.5.2 If Algorithm 6 Outputs a Maximum Independent Set

We now prove Theorem 4.4, assuming that Algorithm 6 outputs a solution S of maximum size, before performing ε^{-1} iterations of the

outer While-loop of Algorithm 6. Formally, we prove the following theorem.

▶ **Theorem 4.7.** Define the function f as in (4.1), with a log-likelihood function that is (M, m)-(smooth, strongly concave) on Ω_{2r}. Suppose that Algorithm 6 outputs a solution S^* such that $\mathrm{Cond}(\mathsf{S}^*) = \emptyset$. Then,

$$\frac{\mathbb{E}\left[\, f(\mathsf{S}^*)\,\right]}{\mathtt{OPT}} \geq \frac{1}{1+p}\left(1 - \exp\left\{-(1-\varepsilon)^2 \frac{m^3}{M^3}\right\}\right),$$

for all $0 < \varepsilon < 1$. Furthermore, in the specific case when $\mathscr{I}$ is r-sparsity constraint over $[n]$, then,

$$\frac{\mathbb{E}\left[\, f(\mathsf{S}^*)\,\right]}{\mathtt{OPT}} \geq 1 - \exp\left\{-(1-\varepsilon)^2 \frac{m^2}{M^2}\right\}.$$

◀

The proof of Theorem 4.7 is based on the following two additional lemma.

▶ **Lemma 4.3.** At any point during the optimization process it holds

$$(1-\varepsilon)^{-1}\frac{rM}{m}t \geq 2m\big(\mathtt{OPT} - f(\mathsf{S})\big),$$

with S the current solution. ◀

Proof. Denote with $\bar{\mathsf{S}}$ a solution of size at most $|\bar{\mathsf{S}}| \leq r$ maximizing $f(\mathsf{S} \cup \bar{\mathsf{S}})$, and let $\mathsf{T} \subseteq [n]$ be a set maximizing $\|\nabla l(\boldsymbol{\beta}^{(\mathsf{S})})_{\mathsf{T}}\|_2^2$, such that $|\mathsf{T}| \leq r$ and $\mathsf{S} \cup \{s\} \in \mathscr{I}$ for all $s \in \mathsf{T}$. Note that it holds

$$(1-\varepsilon)^{-1}\frac{rM}{m}t = \frac{r}{|\mathsf{T}|}\|\nabla l(\boldsymbol{\beta}^{(\mathsf{S})})_{\mathsf{T}}\|_2^2 \geq \|\nabla l(\boldsymbol{\beta}^{(\mathsf{S})})_{\mathsf{T}}\|_2^2, \tag{4.12}$$

where the first inequality follows by the definition of t, and the second one follows since $|\mathsf{T}| \leq r$. We first prove the claim when t is updated at

the beginning of each iteration of the outer While-loop of Algorithm 6. It holds

$$\begin{aligned} l(\boldsymbol{\beta}^{(\mathsf{S}\cup\bar{\mathsf{S}})}) - l(\boldsymbol{\beta}^{(\mathsf{S})}) \\ &\leq \langle \nabla l(\boldsymbol{\beta}^{(\mathsf{S})}), \boldsymbol{\beta}^{(\mathsf{S}\cup\bar{\mathsf{S}})} - \boldsymbol{\beta}^{(\mathsf{S})} \rangle - \frac{m}{2} \|\boldsymbol{\beta}^{(\mathsf{S}\cup\bar{\mathsf{S}})} - \boldsymbol{\beta}^{(\mathsf{S})}\|_2^2 \\ &\leq \max_{\mathbf{v}:\, \mathbf{v}_{(\mathsf{S}\cup\bar{\mathsf{S}})=0}} \langle \nabla l(\boldsymbol{\beta}^{(\mathsf{S})}), \mathbf{v} - \boldsymbol{\beta}^{(\mathsf{S})} \rangle - \frac{m}{2} \|\mathbf{v} - \boldsymbol{\beta}^{(\mathsf{S})}\|_2^2, \end{aligned} \tag{4.13}$$

where the first inequality uses the restricted strong concavity, and the second one uses the maximality of $\mathbf{v}$. By setting $\mathbf{v} = \frac{1}{m}\boldsymbol{\beta}^{(\mathsf{S})} + \nabla l(\boldsymbol{\beta}^{(\mathsf{S})})_{\bar{\mathsf{S}}}$ in the inequality above we get

$$\begin{aligned} & f(\mathtt{OPT}) - f(\mathsf{S}) \\ & = l(\boldsymbol{\beta}^{(\mathsf{S}\cup\bar{\mathsf{S}})}) - l(\boldsymbol{\beta}^{(\mathsf{S})}) \quad \text{(maximality of } \bar{\mathsf{S}} \text{ and monotonicity)} \\ & \leq \frac{\|\nabla l(\boldsymbol{\beta}^{(\mathsf{S})})_{\bar{\mathsf{S}}}\|_2^2}{2m} \qquad \text{(substituting } \mathbf{v} \text{ in (4.13))} \\ & \leq \frac{\|\nabla l(\boldsymbol{\beta}^{(\mathsf{S})})_{\mathsf{T}}\|_2^2}{2m}. \qquad \text{(maximality of } \mathsf{T}) \end{aligned} \tag{4.14}$$

The claim follows by combining (4.14) and (4.12).

Suppose now that the current solution S is updated to S' during the inner While-loop of Algorithm 6. Then, $f(\mathsf{S}) \geq f(\mathsf{S}')$ due to monotonicity, and the claim holds. ■

▶ **Lemma 4.4.** At any given time step, suppose that the current solution S is updated to $\mathsf{S} \cup \{a_1, \ldots, a_{j^*}\}$, and define $\mathsf{S}_j = \mathsf{S} \cup \{a_1, \ldots, a_j\}$ for all $j \in [j^*]$. Then it holds

$$\mathbb{E}\Big[\, f_{\mathsf{S}_{j-1}}(\mathsf{S}_j) \,\Big] \geq (1-\varepsilon)^2 \frac{m^2}{rM^2} \big(\mathtt{OPT} - \mathbb{E}[\, f(\mathsf{S}_{j-1}) \,]\big).$$

◀

Proof. Fix all random decisions of Algorithm 6 until the point a_j is

added to the current solution. Define the sets $\mathsf{X}_j^{\mathscr{I}} := \{e \in \mathsf{X} \setminus \{a_1, \dots, a_{j-1}\} \colon \{a_1, \dots, a_{j-1}\} \cup \{a\} \in \mathscr{I}\}$. Then, it holds

$$\begin{aligned}
&\mathbb{E}_{a_j}\left[\, f_{\mathsf{S}_{j-1}}(a_j) \,\right] \\
&\geq \frac{1}{2M}\mathbb{E}_{a_j}\left[\, \langle \nabla l(\beta^{(\mathsf{S}_{j-1})}), \mathbf{e}_{a_j}\rangle^2 \,\right] && \text{(Theorem 4.6)} \\
&\geq \frac{1}{2M}\mathsf{Pr}_{a_j}\Big(\langle \nabla l(\beta^{(\mathsf{S}_{j-1})}), \mathbf{e}_{a_j}\rangle^2 \geq t\Big)\, t && \text{(Markov's inequality)} \quad (4.15)
\end{aligned}$$

By design of the RNDSEQ subroutine, each point a_j is sampled uniformly at random from the set $\mathsf{X}_j^{\mathscr{I}}$, i.e. $a_j \sim \mathscr{U}(\mathsf{X}_j^{\mathscr{I}})$. Hence, it holds $\mathsf{Pr}_{a_j}\Big(\langle \nabla l(\beta^{(\mathsf{S}_{j-1})}), \mathbf{e}_{a_j}\rangle^2 \geq t\Big) = \big|\mathsf{X}_{j-1}\big| / \big|\mathsf{X}_j^{\mathscr{I}}\big|$. Combining this observation with (4.15) we get

$$\begin{aligned}
&\mathbb{E}_{a_j}\left[\, f_{\mathsf{S}_{j-1}}(a_j) \,\right] \\
&\quad \geq \frac{1}{2M}\frac{|\mathsf{X}_{j-1}|}{\big|\mathsf{X}_j^{\mathscr{I}}\big|}\, t, \\
&\quad \geq \frac{1}{2M}\frac{|\mathsf{X}_{j-1}|}{|\mathsf{X}|}\, t, && (\mathsf{X}_j^{\mathscr{I}} \subseteq \mathsf{X}) \\
&\quad \geq (1-\varepsilon)\frac{1}{2M}\, t && (\big|\mathsf{X}_{j-1}\big| \geq (1-\varepsilon)|\mathsf{X}|) \\
&\quad \geq (1-\varepsilon)^2 \frac{m^2}{rM^2}\big(\mathtt{OPT} - f(\mathsf{S}_{j-1})\big), && \text{(Lemma 4.3)}
\end{aligned}$$

The claim follows by taking the expectation on both sides. ■

Using this lemma, we can now prove Theorem 4.7.

Proof of Theorem 4.7. Denote with $\{a_1, \dots, a_j\}$ the first j points added to the solution S^*, sorted in the order that they were added to it, and define the constant $c := (1-\varepsilon)^2 m^2 / rM^2$. Using an induction

argument on j, we prove that it holds

$$\frac{\mathbb{E}\left[\, f(\{a_1,\ldots,a_j\})\,\right]}{\mathtt{OPT}} \geq \Big(1-(1-c)^j\Big). \tag{4.16}$$

The base case with $j = 0$ holds, due to the non-negativity of the function f. For the inductive case, we have that it holds

$$\begin{aligned}
&\mathbb{E}\left[\, f(\{a_1,\ldots,a_j\})\,\right] \\
&\quad\geq \mathbb{E}\left[\, f(\{a_1,\ldots,a_{j-1}\})\,\right] + c\big(\mathtt{OPT} - f(\{a_1,\ldots,a_{j-1}\})\big) \\
&\quad\geq (1-c)\Big(1-(1-c)^{j-1}\Big)\mathtt{OPT} + c\mathtt{OPT} \\
&\quad\geq \Big(1-(1-c)^j\Big)\mathtt{OPT},
\end{aligned}$$

where the first inequality follows by Lemma 4.4, and the second one follows by induction. Then, (4.16) holds. It follows that for $j = |\mathsf{S}^*|$ we have

$$\frac{\mathbb{E}\left[\, f(\mathsf{S}^*)\,\right]}{\mathtt{OPT}} \geq 1-(1-c)^{|\mathsf{S}|} \geq 1-\exp\left\{-\frac{|\mathsf{S}^*|}{k}c\right\}.$$

In the case of a r-sparsity constraint we have that $|\mathsf{S}^*| = r$, and the claim follows. In the case of a p-system constraint, we have that $|\mathsf{S}^*| \leq pr$, hence

$$\begin{aligned}
\frac{\mathbb{E}\left[\, f(\mathsf{S}^*)\,\right]}{\mathtt{OPT}} &\geq 1-\exp\left\{-(1-\varepsilon)^2 p\frac{m^2}{M^2}\right\} \\
&\geq \frac{1}{1+p}\left(1-\exp\left\{-(1-\varepsilon)^2\frac{m^3}{M^3}\right\}\right),
\end{aligned}$$

and the claim also holds. ■

4.5.3 If Algorithm 6 Terminates after ε^{-1}terations

We now prove Theorem 4.4, assuming that the FASTOMP terminates after ε^{-1} iterations of the outer While-loop of Algorithm 6. Specifically, we prove the following theorem.

▶ **Theorem 4.8.** Define the function f as in (4.1), with a log-likelihood function that is (M,m)-(smooth, strongly concave) on Ω_{2r}. Suppose that Algorithm 6 terminates after ε^{-1} iterations of the outer-While loop. Then,

$$\frac{\mathbb{E}\left[\, f(\mathsf{S}^*)\,\right]}{\mathtt{OPT}} \geq \frac{1}{1+p}\left(1-\exp\left\{-(1-\varepsilon)^2\frac{m^3}{M^3}\right\}\right),$$

for all $0<\varepsilon<1$. Furthermore, in the specific case when $\mathscr{I}$ is r-sparsity constraint over $[n]$, then,

$$\frac{\mathbb{E}\left[\, f(\mathsf{S}^*)\,\right]}{\mathtt{OPT}} \geq 1-\exp\left\{-(1-\varepsilon)^2\frac{m^2}{M^2}\right\}.$$

◀

In order to prove this theorem, we introduce additional notation. We denote with S_i the current solution at the beginning of the i-th iteration of the outer While-loop of Algorithm 6. Furthermore, denote with $\bar{\mathsf{S}}$ a feasible set, such that $f(\bar{\mathsf{S}}) = \mathtt{OPT}$, and denote with a_j the j-th element added to the solution S. For each element a_j, define the set $\mathsf{D}_j := (\bar{\mathsf{S}} \cap \mathrm{Cond}(\{a_1,\ldots,a_{j-1}\})) \setminus (\bar{\mathsf{S}} \cap \mathrm{Cond}(\{a_1,\ldots,a_j\}))$. Note that these sets consist of all points in $\bar{\mathsf{S}}$ that yield a feasible solution when added to $\{a_1,\ldots,a_{j-1}\}$, but that violate side constraints when added to $\{a_1,\ldots,a_j\}$. Note also that it holds $\mathsf{D}_1 \cup \cdots \cup \mathsf{D}_j = \mathsf{T}^* \setminus \mathrm{Cond}(\{a_1,\ldots,a_j\})$.

The proof of this theorem is based on the following lemma.

▶ **Lemma 4.5.** It holds

$$\frac{M}{m} f_{\mathsf{S}_i}(\mathsf{S}_{i+1}) + \varepsilon \bar{f}_{\mathsf{S}_i}(\mathsf{D}_1 \cup \cdots \cup \mathsf{D}_{|\mathsf{S}_i|}) \geq \varepsilon \frac{m}{M} f_{\mathsf{S}_i}(\bar{\mathsf{S}}).$$

◀

Proof. Fix all random decision of Algorithm 6, up to the $(i+1)$-th iteration of the outer While-loop of Algorithm 6. Let $\mathsf{T} \subseteq [n]$ be a set maximizing $\|\nabla l(\boldsymbol{\beta}^{(\mathsf{S}_i)})_{\mathsf{T}}\|_2^2$, such that $|\mathsf{T}| \leq r$ and $\mathsf{T} \subseteq \mathrm{Cond}(\mathsf{S}_i)$.

Due to the assumption on the stopping criterion, it holds $\mathsf{X} = \emptyset$ at the end of iteration i. This means that each point $j \in (\mathsf{T} \setminus \mathsf{S}_i) \cap \mathrm{Cond}(\mathsf{S}_i)$ was discarded at some point during the previous iteration. Denote with U_j the current solution when j was discarded. Then, it holds

$$(1-\varepsilon)\frac{2m}{r} \bar{f}_{\mathsf{S}_i}(\mathsf{T}) \geq (1-\varepsilon)\frac{m}{rM} \|\nabla l(\boldsymbol{\beta}^{(\mathsf{S}_i)})_{\mathsf{T}}\|_2^2 = t \tag{4.17}$$

where the first inequality follows by Corollary 4.1, and the second one follows by the definition of t. Since the point j was discarded and since $j \in \mathrm{Cond}(\mathsf{S}_i)$, then it must hold $t \geq \langle \nabla l(\boldsymbol{\beta}_j^{(\mathsf{U}_j)}), \mathbf{e}_j \rangle^2$. Combining this observation with (4.17) we get

$$\begin{aligned}(1-\varepsilon)\frac{2m}{r} \bar{f}_{\mathsf{S}_i}(\mathsf{T}) &\geq \langle \nabla l(\boldsymbol{\beta}_j^{(\mathsf{U}_j)}), \mathbf{e}_j \rangle^2 \\ &\geq 2m \bar{f}_{\mathsf{U}_j}(j) \geq 2m \bar{f}_{\mathsf{S}_{i+1}}(j),\end{aligned} \tag{4.18}$$

where the second inequality follows by Corollary 4.1, and the last one follows from the submodularity of $\bar{f}$. By taking the sum over all points $j \in (\mathsf{T} \setminus \mathsf{S}_i) \cap \mathrm{Cond}(\mathsf{S}_i)$ and rearranging, we get

$$\begin{aligned}&(1-\varepsilon)\bar{f}_{\mathsf{S}_i}(\mathsf{T}) \\ &\geq (1-\varepsilon)|(\mathsf{T} \setminus \mathsf{S}_i) \cap \mathrm{Cond}(\mathsf{S}_i)| \frac{\bar{f}_{\mathsf{S}_i}(\mathsf{T})}{r} \qquad (r \geq |\mathsf{T}^*|)\end{aligned}$$

$$\geq \sum_{j \in (T \setminus S_i) \cap \mathrm{Cond}(S_i)} \bar{f}_{S_{i+1}}(j) \qquad \text{(by (4.18))}$$

$$\geq \bar{f}_{S_{i+1}}((T \setminus S_i) \cap \mathrm{Cond}(S_i)). \qquad \text{(submodularity)} \qquad (4.19)$$

By rearranging (4.19) we get

$$\begin{aligned} \bar{f}_{S_i}(S_{i+1}) &\geq \varepsilon \bar{f}_{S_i}((T \setminus S_i) \cap \mathrm{Cond}(S_i)) \\ &= \varepsilon \bar{f}_{S_i}(\bar{S} \cap \mathrm{Cond}(S_i)) \\ &\geq \varepsilon \bar{f}_{S_i}(\bar{S}) - \varepsilon \bar{f}_{S_i}(\bar{S} \setminus \mathrm{Cond}(S_i)) \\ &\geq \varepsilon \bar{f}_{S_i}(\bar{S}) - \varepsilon \bar{f}_{S_i}(D_1 \cup \cdots \cup D_{|S_i|}) \end{aligned} \qquad (4.20)$$

Hence, it holds

$$\begin{aligned} &\frac{M}{m} f_{S_i}(S_{i+1}) \geq \bar{f}_{S_i}(S_{i+1}) && \text{(Corollary 4.1)} \\ &\geq \varepsilon \bar{f}_{S_i}(\bar{S}) - \varepsilon \bar{f}_{S_i}(D_1 \cup \cdots \cup D_{|S_i|}) && \text{(it follows from (4.20))} \\ &\geq \varepsilon \frac{m}{M} f_{S_i}(\bar{S}) - \varepsilon \bar{f}_{S_i}(D_1 \cup \cdots \cup D_{|S_i|}). && \text{(Corollary 4.1)} \end{aligned}$$

The claim follows by rearranging. ■

In order to continue with the proof, we also use the following lemma.

▶ **Lemma 4.6.** It holds

$$p f(S_i) \geq (1-\varepsilon) \frac{m^2}{M^2} \bar{f}_{S_i}(D_1 \cup \cdots \cup D_{|S_i|}).$$

◀

Proof. Fix an index $j \leq |S_i|$, and fix all random decisions of Algorithm 6 until the point a_j is added to the current solution. Define the sets

$\mathsf{X}_j^{\mathscr{I}} := \{e \in \mathsf{X} \setminus \{a_1, \ldots, a_{j-1}\} \colon \{a_1, \ldots, a_{j-1}\} \cup \{a\} \in \mathscr{I}\}$. Then, it holds

$$\begin{aligned}
&\mathbb{E}_{a_j}\Big[\, f_{\{a_1,\ldots,a_{j-1}\}}(a_j) \,\Big] \\
&\quad \geq \frac{1}{2M}\mathbb{E}_{a_j}\Big[\, \langle \nabla l(\boldsymbol{\beta}^{(\{a_1,\ldots,a_{j-1}\})}), \mathbf{e}_{a_j}\rangle^2 \,\Big] \\
&\quad \geq \frac{1}{2M}\mathsf{Pr}_{a_j}\Big(\langle \nabla l(\boldsymbol{\beta}^{(\{a_1,\ldots,a_{j-1}\})}), \mathbf{e}_{a_j}\rangle^2 \geq t\Big)\, t
\end{aligned} \tag{4.21}$$

where the first inequality uses Lemma 4.6, and the second one uses Markov's inequality. Note that the RNDSEQ subroutine samples points a_j uniformly at random $a_j \sim \mathscr{U}(\mathsf{X}_j^{\mathscr{I}})$. Hence,

$$\mathsf{Pr}_{a_j}\Big(\langle \nabla l(\boldsymbol{\beta}^{(\mathsf{S}_{j-1})}), \mathbf{e}_{a_j}\rangle^2 \geq t\Big) = \frac{|\mathsf{X}_{j-1}|}{\left|\mathsf{X}_j^{\mathscr{I}}\right|} \geq \frac{|\mathsf{X}_{j-1}|}{|\mathsf{X}|}, \tag{4.22}$$

where the last inequality holds, since $\mathsf{X}_j^{\mathscr{I}} \subseteq \mathsf{X}$. Combining this observation with (4.21) we get

$$\begin{aligned}
&\mathbb{E}_{a_j}\Big[\, f_{\{a_1,\ldots,a_{j-1}\}}(a_j) \,\Big] \\
&\quad \geq \frac{1}{2M}\frac{|\mathsf{X}_j|}{|\mathsf{X}|}\, t && \text{(it follows by (4.22))} \\
&\quad \geq \frac{1-\varepsilon}{2M}\, t && (|\mathsf{X}_{j-1}| \geq (1-\varepsilon)|\mathsf{X}|) \\
&\quad = \frac{1-\varepsilon}{2M^2}\frac{m}{|\mathsf{T}|}\|\nabla l(\boldsymbol{\beta}^{(\{a_1,\ldots,a_{j-1}\})})_{\mathsf{T}}\|_2^2 && \text{(by the definition of } t) \\
&\quad \geq \frac{1-\varepsilon}{2M^2}\frac{m}{|\mathsf{D}_j|}\|\nabla l(\boldsymbol{\beta}^{(\{a_1,\ldots,a_{j-1}\})})_{\mathsf{D}_j}\|_2^2, && (\mathsf{T} \text{ is maximal})
\end{aligned}$$

By taking the expected value on both sides in the chain of inequalities

above, we get

$$\sum_{j\leq|\mathsf{S}_i|}\big|\mathsf{D}_j\big|\mathbb{E}_{a_j}\Big[\,\bar{f}_{\{a_1,\ldots,a_{j-1}\}}(a_j)\,\Big] \geq (1-\varepsilon)\frac{m}{2M^2}\sum_{j\leq|\mathsf{S}_i|}\mathbb{E}\Big[\,\|\nabla l(\beta^{(\mathsf{S}_i)})_{\mathsf{D}_j}\|_2^2\,\Big]. \tag{4.23}$$

In order to continue with the proof, we give an upper-bound on the size of the sum $\sum_j\big|\mathsf{D}_j\big|$. To this end, note that the set S_i is a maximum independent set over the ground set

$$\mathsf{S}_i\cup\big(\mathsf{D}_1\cup\cdots\cup\mathsf{D}_{|\mathsf{S}_i|}\big)=\mathsf{S}_i\cup\big(\mathsf{T}^*\setminus\mathrm{Cond}(\mathsf{S}_i)\big).$$

In fact, the set S_i is independent by definition, and that any point $s\in(\mathsf{T}^*\setminus\mathrm{Cond}(\mathsf{S}_i))\setminus\mathsf{S}_i$ yields $\mathsf{S}_i\cup\{s\}\notin\mathscr{I}$. Hence S_i is a maximum independent set as claimed. Note also that $\mathsf{D}_1\cup\cdots\cup\mathsf{D}_{|\mathsf{S}_i|}\subseteq\mathsf{T}^*$ is an independent set, due to the subset-closure of $\mathscr{I}$. Since $\mathscr{I}$ is a p-system, then it holds

$$|\mathsf{D}_1|+\cdots+\big|\mathsf{D}_{|\mathsf{S}_i|}\big|=\big|\mathsf{D}_1\cup\cdots\cup\mathsf{D}_{|\mathsf{S}_i|}\big|\leq p|\mathsf{S}_i|. \tag{4.24}$$

Hence, it holds

$$\begin{aligned}
&p\sum_{j\leq|\mathsf{S}_i|}\mathbb{E}_{a_j}\Big[\,f_{\{a_1,\ldots,a_{j-1}\}}(a_j)\,\Big]\\
&\geq\sum_{j\leq|\mathsf{S}_i|}\big|\mathsf{D}_j\big|\mathbb{E}_{a_j}\Big[\,\bar{f}_{\{a_1,\ldots,a_{j-1}\}}(a_j)\,\Big] && \text{(by (4.24) and Prop. 4.4)}\\
&\geq(1-\varepsilon)\frac{m}{2M^2}\sum_{j\leq|\mathsf{S}_i|}\mathbb{E}\Big[\,\|\nabla l(\beta^{(\mathsf{S}_i)})_{\mathsf{D}_j}\|_2^2\,\Big] && \text{(it follows by (4.23))}\\
&\geq(1-\varepsilon)\frac{m^2}{M^2}\sum_{j\leq|\mathsf{S}_i|}\bar{f}_{\mathsf{S}_i}(\mathsf{D}_j) && \text{(Theorem 4.6)}\\
&\geq(1-\varepsilon)\frac{m^2}{M^2}\sum_{j\leq|\mathsf{S}_i|}\bar{f}_{\mathsf{S}_i}(\mathsf{D}_1\cup\cdots\cup\mathsf{D}_{|\mathsf{S}_i|}), && \text{(submodularity)}
\end{aligned}$$

as claimed ■

We now have all necessary tools to prove Theorem 4.8.

Proof of Theorem 4.8. We first prove the claim, assuming that $\mathscr{I}$ is a general p-system. In this case, by combining Lemma 4.5 with Lemma 4.6 it holds

$$\frac{M}{m}\mathbb{E}\left[\, f_{\mathsf{S}_i}(\mathsf{S}_{i+1}) \,\right] + \varepsilon p \frac{M^2}{m^2}\mathbb{E}\left[\, f(\mathsf{S}_i) \,\right] \geq \varepsilon(1-\varepsilon)\frac{m}{M}\mathbb{E}\left[\, f_{\mathsf{S}_i}(\bar{\mathsf{S}}) \,\right]. \tag{4.25}$$

To continue, define the constant $c = (1-\varepsilon)m^3/M^3$. We prove by induction on i that it holds

$$(1+\varepsilon i p)\mathbb{E}\left[\, f(\mathsf{S}_i) \,\right] \geq (1-(1-\varepsilon c)^i)\mathtt{OPT}. \tag{4.26}$$

The base case with $\mathsf{S}_0 = \emptyset$ trivially follows, since the function f is non-negative. For the inductive case, suppose that the claim holds for $\mathbb{E}\left[\, f(\mathsf{S}_{i-1}) \,\right]$. Then,

$$\begin{aligned}
&(1+\varepsilon i p)\mathbb{E}\left[\, f(\mathsf{S}_i) \,\right] \\
&\quad \geq \mathbb{E}\left[\, f(\mathsf{S}_i) \,\right] + \varepsilon i p \mathbb{E}\left[\, f(\mathsf{S}_{i-1}) \,\right] \\
&\quad \geq \mathbb{E}\left[\, f(\mathsf{S}_{i-1}) \,\right] + \varepsilon c \mathbb{E}\left[\, f_{\bar{\mathsf{S}}}(\mathsf{S}^*) \,\right] + \varepsilon(i-1)p\mathbb{E}\left[\, f(\mathsf{S}_{i-1}) \,\right] \\
&\quad \geq (1-\varepsilon c)\mathbb{E}\left[\, f(\mathsf{S}_{i-1}) \,\right] + \varepsilon c \mathtt{OPT} + \varepsilon(i-1)p\mathbb{E}\left[\, f(\mathsf{S}_{i-1}) \,\right] \\
&\quad \geq (1-\varepsilon c)(1-(1-\varepsilon c)^{i-1})\mathtt{OPT} + \varepsilon c \mathtt{OPT} \\
&\quad \geq (1-(1-\varepsilon c)^i)\mathtt{OPT},
\end{aligned}$$

where the first inequality uses monotonicity; the second one follows by (4.25); the third inequality uses monotonicity again, and the fourth one follows by induction. Hence, (4.26) holds. It follows that

$$\begin{aligned}
\mathbb{E}\left[\, f(\mathsf{S}^*) \,\right] &= \mathbb{E}\left[\, f(\mathsf{S}_{\lfloor 1/\varepsilon \rfloor}) \,\right] \\
&\geq \frac{1}{1+\varepsilon\lfloor 1/\varepsilon \rfloor p}\left(1-\left(1-\varepsilon(1-\varepsilon)\frac{m^3}{M^3}\right)^{\lfloor 1/\varepsilon \rfloor}\right)\mathtt{OPT}
\end{aligned}$$

$$\geq \frac{1}{1+p}\left(1-\exp\left\{-(1-\varepsilon)\frac{m^3}{M^3}\right\}\right)\mathtt{OPT},$$

where the first equation follows by the stopping criterion, and the first inequality follows by (4.26).

We conclude by proving the claim in the special case that $\mathscr{I}$ is a r-sparsity constraint. Since the algorithm terminates before a solution of size r is found, we have that $\mathrm{Cond}(\mathsf{S}_i) = [n] \setminus \mathsf{S}_i$ for all iterations i. Hence, $\mathsf{D}_1 \cup \cdots \cup \mathsf{D}_i = \emptyset$ and Lemma 4.5 yields

$$\mathbb{E}\left[\, f_{\mathsf{S}_i}(\mathsf{S}_{i+1}) \,\right] \geq \varepsilon \frac{m^2}{M^2} \mathbb{E}\left[\, f_{\mathsf{S}_i}(\mathsf{S}^*) \,\right].$$

With this inequality, we can use an inductive argument similar to the proof for the general case, and obtain an improved lower-bound on the solution quality. ■

4.6 Proof of Theorem 4.5

We conclude by giving upper-bounds on the run time and adaptivity for Algorithm 6. Recall that the notion of adaptivity is given in Definition 4.6. The following theorem holds.

▶ **Theorem 4.5.** Algorithm 6 terminates after $\mathcal{O}\left(\varepsilon^{-2}\log n\right)$ rounds of calls to the oracle function, and it uses at most $\mathcal{O}\left(\varepsilon^{-2} r \log n\right)$ oracle queries. Furthermore, Algorithm 6 requires expected $\mathcal{O}\left(\varepsilon^{-2}\sqrt{r}\log n\right)$ independent calls to the oracle for the p-system $\mathscr{I}$, and the total expected calls to the oracle for the p-system is $\mathcal{O}\left(\varepsilon^{-2} n r \log n\right)$. ◀

In order to prove this result, we use the following well-known estimate on the number of adaptive rounds of the RNDSEQ sub-routine (see Algorithm 7).

▶ **Theorem 4.9 (Theorem 6 by (Karp et al. 1988)).** Algorithm 7 terminates after expected $\mathcal{O}(\sqrt{r})$ steps, with r the rank of the independent system $\mathcal{I}$. ◀

Note that this theorem implies that the number of adaptive rounds of the independence oracle for Algorithm 7 is $\mathcal{O}(\sqrt{r})$ in expected value. In fact, in each step of Algorithm 7, queries to the independence oracle can be performed in parallel. Using this result, we can now prove Theorem 4.5.

of Theorem 4.5. We first give upper-bounds for the oracle function that accesses $\nabla l(\cdot)$. To this end, observe that there are two While-loops in Algorithm 6. The outer while-loop terminates after at most ε^{-1} iteration. The inner While-loop terminates after $\mathcal{O}(\varepsilon^{-1}\log n)$ iterations, since at each iteration the size of X decreases at least of a multiplicative factor of $1-\varepsilon$. Hence, the rounds of calls to the oracle function is $\mathcal{O}(\varepsilon^{-2}\log n)$. Furthermore, at each iteration of the inner While-loop, at most r parallel calls to $\nabla l(\cdot)$ are preformed. It follows that the total number of oracle calls is $\mathcal{O}(\varepsilon^{-2} r\log n)$.

We now estimate the number of adaptive rounds and run time for the calls to the independence oracle. To this end, note that is oracle is called by the RNDSEQ sub-routine,and it is also evaluated nr times in parallel during the inner While-loop of Algorithm 6. From Theorem 4.9 it follows that the number of adaptive rounds is $\mathcal{O}(\varepsilon^{-2}\sqrt{r}\log n)$, and that the total number of calls to the oracle function is $\mathcal{O}(\varepsilon^{-2} nr\log n)$ as claimed. ■

4.7 Experiments

In this section, we provide extensive experiments on both synthetic and real world datasets to illustrate the superior performance of the proposed methods. All experiments are performed on Python 3 on

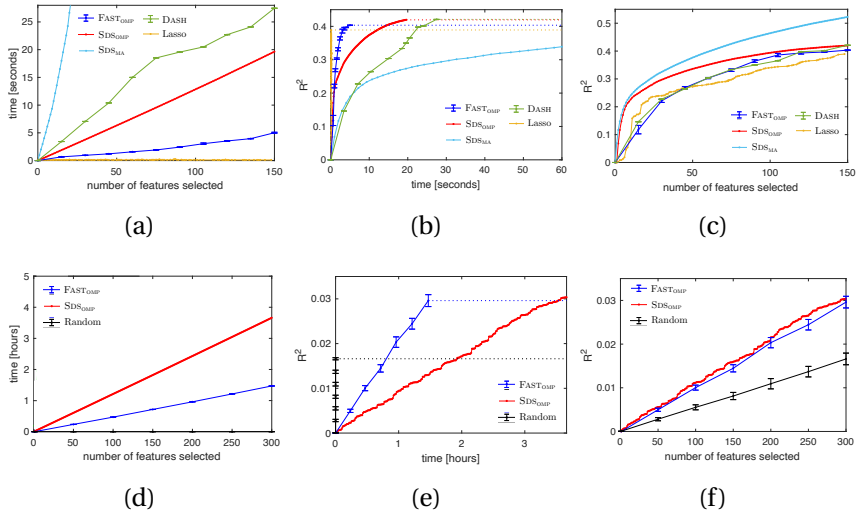
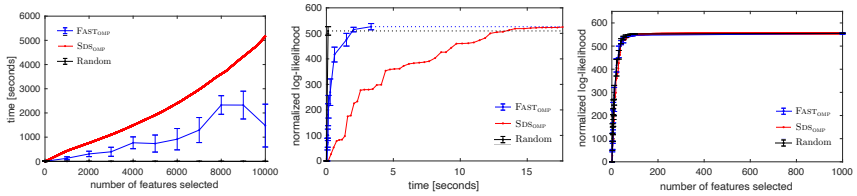


Figure 4.1: Results on the Synthetic Unconstrained Dataset (top row), and the Synthetic Dataset with Constraints (bottom row), as described in Section 4.7. We observe that the FAST$_{\text{OMP}}$ is faster in selecting features and is better in selecting the *correct* in terms of R^2.

a server that runs Linux with Intel Xeon E5-2630 v4 with 40 CPUs at 2.2GHz.

4.7.1 Datasets

Unconstrained Synthetic Dataset. We use an artificially generated data-set consisting of 500 features, 1000 observations, and it has size approximately 10MB. We generated the dataset using a joint Gaussian distribution with mean vector $\boldsymbol{\mu} = \mathbf{0}$, and covariance matrix $\boldsymbol{\Sigma}$. We define $\boldsymbol{\Sigma}$ such that 10% of the features are highly correlated with the response, and the remaining features have low correlation with the response variable. We then add posterior uniform noise, and normalize the observations.



(a) (b) (c)

Figure 4.2: Results on the Pan-Cancer Data-Set, as described in Section 4.7. The FASTOMP outperforms baselines in selecting the useful features much faster.

Synthetic Dataset with Constraints. This data-set consists of 1×10^6 features, 1×10^4 observations, and with size approximately 19GB. We generated this dataset method described for the Unconstrained Synthetic Data-set. We randomly generate a p-system side constraints, that determines the feasibility of a set of features.

CGARN Pan-Cancer Data-set. We use a bio-medical dataset of approximately 2×10^4 features and 804 observation, with size approximately 201MB (Cancer Genome Atlas Research Network et al. 2013). In this dataset, features embed information on the genome sequences of 802 patients affected with cancer, and the observations consists of a pseudo-Boolean array, with 1 if the corresponding patient has PRAD cancer and 0 otherwise. We randomly generate a p-system side constraints, that determines the feasibility of a set of features.

ProPublica COMPAS Data-Set. We consider the well-known ProPublica COMPAS dataset, and perform feature selection with procedural fairness constraints on this dataset (Grgic-Hlaca, Redmiles, et al. 2018; Grgic-Hlaca, Zafar, K. P. Gummadi, et al. 2018). The ProPublica COMPAS dataset is a pretrial risk assessment instrument (Larson et al. 2016). The ProPublica COMPAS dataset was constructed in

2016, using data of defendants from Broward County, FL, who had been arrested in 2013 or 2014 and assessed with the COMPAS risk screening system. ProPublica then collected data on future arrests for these defendants through the end of March 2016, in order to study how the COMPAS score predicted recidivism (Angwin et al. 2016). Based on its analysis, ProPublica concluded that the COMPAS risk score was racially biased (Berk et al. 2021). The ProPublica COMPAS data has become one of the key bench-marking datasets for testing algorithmic fairness definitions and procedures (Chouldechova 2017; Corbett-Davies et al. 2017a; Corbett-Davies et al. 2017b; Cowgill and Tucker 2019; Rudin et al. 2018; Zafar, Valera, Gomez-Rodriguez, and K. P. Gummadi 2017a; Zafar, Valera, Gomez-Rodriguez, K. P. Gummadi, and Weller 2017). However, (Bao et al. 2021) notes that there are inaccuracies in the COMPAS dataset. For instance, COMPAS race categories lack Native Hawaiian or Other Pacific Islander, and it redefines Hispanic as race instead of ethnicity.

The ProPublica COMPAS dataset consists of the following features: "number of prior criminal offenses", "arrest charge description", "charge degree", "number of juvenile felony offenses", "juvenile misdemeanor offenses", "other juvenile offenses", "age" of the defendant, "sex" of the defendant and "race" of the defendant. The dataset also contains information on whether the defendant recidivated or not.

Following (Grgic-Hlaca, Redmiles, et al. 2018; Grgic-Hlaca, Zafar, K. P. Gummadi, et al. 2018), we use the COMPAS dataset to predict if a defendant faces risks of recidivism, and study the trade-off between fairness and accuracy achieved by some of the benchmarks on this instance. The measures of unfairness for the constraints follow the definitions of Section 4.2.4. Consensus is based on a survey performed in 2018 with the Amazon Mechanical Turk (AMT) platform (Grgic-Hlaca, Redmiles, et al. 2018). In this survey, each user was asked to assess if a feature is fair to use for model construction. This survey gathered responses to the above questions from 200 different AMT master workers. Although the results of the survey were

qualitative similar for all the AMT workers, "very liberal" and "very conservative" workers responded differently (Grgic-Hlaca, Redmiles, et al. 2018). We remark that the (Grgic-Hlaca, Redmiles, et al. 2018; Grgic-Hlaca, Zafar, K. P. Gummadi, et al. 2018) do not claim that AMT users are the right group to obtain consensus for estimating fairness.

4.7.2 Benchmarks

GREEDY. Starting from the empty set, this algorithm adds feasible points to the current solution in a greedy fashion (Elenberg, Khanna, et al. 2018; Krause and Cevher 2010). This algorithm uses oracle access to the function f as in (4.1). In our experiments, we implement a version of GREEDY with parallel queries to the oracles for a fair comparison.

SDS$_{\text{OMP}}$. Starting from the empty set, this algorithm iteratively add features s to the current solution S if they maximize the dot product $\langle \nabla l(\beta^{(\mathsf{S})}), \mathbf{e}_s \rangle$ (Elenberg, Khanna, et al. 2018; Krause and Cevher 2010). Our implementation of SDS$_{\text{OMP}}$ handles side constrains, by adding a new feature s to the current solution S if it holds $s \in \text{Cond}(\mathsf{S})$. We implement calls to the independence oracle in parallel, for fair comparison.

DASH. This algorithm achieves strong approximation guarantees on the subset selection problem, under the RSC/RSM assumption (S. Qian and Singer 2019). The DASH uses oracle access to the function f as in (4.1). Calls to the function f can be paralleilzed efficiently. This algorithm can only handle r-sparsity constraints.

ISK. This algorithm is the iterated submodular-cost knapsack algorithm proposed by (Iyer and Bilmes 2013). This algorithm can be used to solve the submodular cost submodular knapsack (SCSK) problem.

This algorithm was used by (Grgic-Hlaca, Zafar, K. P. Gummadi, et al. 2018) on to perform feature selection on the ProPublica COMPAS dataset. However, the ISK can only handle r-sparsity constraints, and it has no known guarantees on Problem (4.2).

LASSO. Lasso is a popular algorithm useful for regression, and classification. The LASSO optimizes the ℓ_1-objective with regularization. Finding a regularizer to obtain a solution of size k is intractable (Mairal and B. Yu 2012). In our experiments we vary the regularizer manually and benchmark against the resulting solution size. Note that the LASSOT can only handle r-sparsity constraints.

RANDOM. This simple algorithm outputs a maximum independent set of $\mathscr{I}$ chosen uniformly at random. We use the RAND-SEQUENCE sub-routine to generate this set. This sub-routine corresponds to Algorithm A by (Karp et al. 1988).

4.7.3 Results

Unconstrained Synthetic Dataset. We consider a linear regression task on this dataset, and we search for a set of features maximizing the R^2 objective. The results are presented in Figure 4.1. Figure 4.1(a) showcases the run time of each algorithm for a fixed number of features selected. We observe that our algorithm significantly outperforms baselines in performance. In Figure 4.1(b), we fix an upper-bound of $k = 150$ on the number of features selected, and we compare the solution accuracy of each algorithm, for a given time budget. We observe that the FAST_{OMP} outperforms the other algorithms. Furthermore, we observe that the GREEDY yields significantly worse performance than baselines. In Figure 4.1(c) we display the solution quality vs. the number of features selected. We observe that the FAST_{OMP}, the SDS_{OMP}, the DASH, the LASSO achieve similar

solution quality. The GREEDY yields best performance, but it is much slower than the other algorithms.

Synthetic Dataset with Constraints We search for a set of features maximizing the R^2 objective, for the Synthetic Dataset with Constraints. The results for this set of experiments are presented in Figure 4.1. We do not report on the results for the GREEDY, due to its bad performance. We do not test the DASH, the ISK, and the LASSO since they cannot handle p-system side constraints by design.

In Figure 4.1(d), we observe that our algorithm significantly outperforms the SDS_{OMP} and GREEDY in performance. In Figure 4.1(e), we compare the solution accuracy of each algorithm, for a given time budget, and $k = 350$ features selected. We observe that the FAST_{OMP} outperforms both the SDS_{OMP} and the RANDOM. In Figure 4.1(f) we observe that the FAST_{OMP} and the SDS_{OMP} achieve similar solution quality, and that they both outperform the RANDOM.

CGARN Pan-Cancer Dataset We use the logistic regression to predict if patients have PRAD cancer, or other types of cancer. To this end, we search for features maximizing the normalized log-likelihood. We do not report on the results for the GREEDY, due to its bad performance. We do not test the DASH, the ISK, and the LASSO since they cannot handle p-system side constraints by design.

In Figure 4.2(a), we observe the run time of each algorithm for a fixed number of features selected. Our algorithm significantly outperforms the baselines in performance. In Figure 4.2(b), we compare the solution accuracy, expressed by the normalized log-likelihood, as a function of time, until each algorithm reaches a solution of $k = 50$ features. We observe that the FAST_{OMP} outperforms the SDS_{OMP} and the RANDOM. In Figure 4.2(c) we display the solution quality achieved by each algorithms for increasing number of features selected. We observe that all algorithms achieve similar performance. However, our

algorithm mildly outproeforms the other algorothms, with respect to the "otcome fairness" indicator.

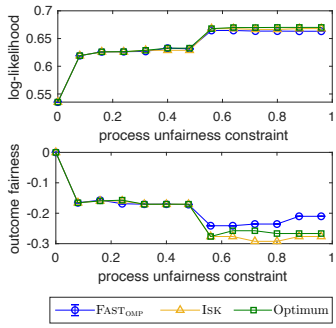


Figure 4.3: Experiments on fairness on the ProPublica COMPAS Dataset.

ProPublica COMPAS Dataset. Following (Grgic-Hlaca, Redmiles, et al. 2018), we use a logistic regression to predict recidivism risk. We train the model using the log-likelihood with regularization, and considering fairness constraints as in (4.9). We report on the outcome fairness of each output feature set, by estimating the racial bias of the corresponding classifier. Following (Grgic-Hlaca, Zafar, K. P. Gummadi, et al. 2018; Kleinberg et al. 2017; Zafar, Valera, Gomez-Rodriguez, and K. P. Gummadi 2017b), we examine the false positive (FPR) and false negative (FNR) rates for whites (w) and non-whites (nw) as

$$-|\text{FPR}_w - \text{FPR}_{nw}| - |\text{FNR}_w - \text{FNR}_{nw}|. \tag{4.27}$$

This measure of fairness varies between -2 and 0, with -2 corresponding to maximum unfairness and 0 to maximum fairness.

The results for this set of experiments are displayed in Figure 4.3. We compare the solution quality and fairness of the FAST$_{\text{OMP}}$, against the ISK, and the solution with best possible accuracy (optimum). The optimum is obtained by training classifiers with all possible combinations of the features of the dataset dataset. In Figure 4.3, we display the results for process unfairness constraints defined as in (4.9). We observe that the FAST$_{\text{OMP}}$ achieves nearly optimal solution. Although the outcome fairness inevitably decreases for increasing process un-

fairness, the FAST$_{\text{OMP}}$ maintains better outcome fairness than the other algorithms.

5 Conclusion

We studied the problem of achieving scalability in AI. We identified three relevant optimization problems, and we propose new algorithms for these problems, that can handle very large datasets. Our proposed algorithms, however, can be used to tackle other combinatorial problems, as long as the underlying axioms characterizing these are fulfilled.

Specifically, we showed that non-monotone submodular maximization under a partition matroid and knapsack constraints can be approached with a simple greedy algorithm. This algorithm finds a near-optimal solution, if the underlying optimization function exhibits bounded curvature. This algorithmic result is particularly useful to solve the Maximum Entropy Sampling problem. Furthermore, we showed that adaptive sampling techniques can be used to maximize a non-monotone submodular function under p-system side constraints. This is practically relevant, since Video Summarization, and other data summarization tasks, can be framed as constrained submodular problems. We conclude by showing that adaptive sampling techniques can be extended to non-submodular problems. As a consequence, we show that adaptive sampling can be used to tackle the central problem of selecting optimal features for model construction.

We provide tools to optimize non-monotone submodular functions under complex side constraints. Furthermore, we show that adaptive sampling techniques can be generalized to non-monotone submodular functions. These tools can be incorporated in the workflow of AI systems, to improve the ability of these systems to handle massive datasets.

5.1 Outlook

As AI systems increasingly become part of our lives, there are growing concerns on potential anomalies and harmful behavior in their course of action. Hence, it is important to study "*how to build trustworthy AI systems*". There are several characteristics that make AI systems trustworthy (see Figure 5.1). Relevant future work consists of studying *robustness* and *fairness*, with the use of optimization and machine learning (ML) techniques.

Figure 5.1: Relevant features of trustworthy AI systems. I am interested in studying *robustness* and *fairness*.

Trustworthy AI systems ought to be robust to errors and changing environments. It is possible to fortify AI systems with the aid of machine learning techniques, such as distributional robustness. These techniques mitigate the variation in accuracy of the output, in the presence of input and system anomalies. Incorporating these techniques into existing systems typically results in additional computational challenges. I am interested in studying robust learning, to make AI systems more reliable. Another important related problem consists of studying fairness. Smart machines can make ethically questionable decisions. For instance, machine learning models could make predictions that are racially or gender-biased. It is crucial to take into account these problems, when building AI systems. For future work, it would be interesting to study new techniques that incorporate notions of fairness into the learning processes. Studying fairness is challenging, since fairness criteria should always take into account the societal context, and the pipeline of the system operating within.

A common problem in ML models is the presence of a bias in the training samples. Distributional robustness provides an effective framework to fortify optimization techniques for Machine Learning. In this setting, the optimization is performed over an entire family of functions, with the goal of optimizing the worst-case over all of these functions. Interestingly, Staib et al. 2019 study distributional robustness in the context of submodular optimization. For future work, it would be interesting to further study this framework, possibly in connection with fairness and privacy concerns.

Fairness in AI and machine learning has been receiving increasing attention in recent years. A common way to address fairness concerns is to incorporate quantitative notions of fairness in the learning process, via additional side constraints. Hence, fairness in AI and machine learning motivates further research in the problem of optimizing combinatorial functions with side constraints.

Bibliography

Agarwal, Alekh, Alina Beygelzimer, et al. (2018). **A Reductions Approach to Fair Classification**. In: *Proc. of ICML*. Vol. 80, 60–69.

Agarwal, Alekh, Miroslav Dudík, and Zhiwei Steven Wu (2019). **Fair Regression: Quantitative Definitions and Reduction-Based Algorithms**. In: *Proc. of ICML*. Vol. 97, 120–129.

Angwin, J. et al. (2016). *There's software used across the country to predict future criminals. And it's biased against blacks.*

Badanidiyuru, Ashwinkumar, Baharan Mirzasoleiman, et al. (2014). **Streaming submodular maximization: massive data summarization on the fly**. In: *Proc. of KDD*, 671–680.

Badanidiyuru, Ashwinkumar and Jan Vondrák (2014). **Fast algorithms for maximizing submodular functions**. In: *proc. of SODA*, 1497–1514.

Bahmani, Sohail, Bhiksha Raj, and Petros T. Boufounos (2013). **Greedy sparsity-constrained optimization**. *Journal of Machine Learning Research* 14:1, 807–841.

Balcan, Maria-Florina and Nicholas J. A. Harvey (2011). **Learning submodular functions**. In: *proc. of STOC*, 793–802.

Balkanski, Eric, Adam Breuer, and Yaron Singer (2018). **Non-monotone Submodular Maximization in Exponentially Fewer Iterations**. In: *Proc. of NeurIPS*, 2359–2370.

Balkanski, Eric, Aviad Rubinstein, and Yaron Singer (2017). **The limitations of optimization from samples**. In: *Proc. of STOC*, 1016–1027.

– (2019). **An optimal approximation for submodular maximization under a matroid constraint in the adaptive complexity model**. In: *proc. of STOC*, 66–77.

Balkanski, Eric and Yaron Singer (2018). **The adaptive complexity of maximizing a submodular function**. In: *proc. of STOC*, 1138–1151.

Bandeira, Afonso S. et al. (2013). **Certifying the Restricted Isometry Property is Hard**. *IEEE Transactions of Information Theory* 59:6, 3448–3450.

Bansal, Nikhil et al. (2010). **On k-Column Sparse Packing Programs**. In: *Proc. of IPCO*, 369–382.

Bao, Michelle et al. (2021). **It's COMPASlicated: The Messy Relationship between RAI Datasets and Algorithmic Fairness Benchmarks**. *CoRR* abs/2106.05498.

Beahrs, John O. (1991). **Volition, Deception, and the Evolution of Justice**. *Journal of the American Academy of Psychiatry and the Law Online* 19:1, 81–93.

Berk, Richard et al. (2021). **Fairness in Criminal Justice Risk Assessments: The State of the Art**. *Sociological Methods & Research* 50:1, 3–44.

Beutel, Alex et al. (2019). **Fairness in Recommendation Ranking through Pairwise Comparisons**. In: *Proc. of SIGKDD*, 2212–2220.

Bian, Andrew An et al. (2017). **Guarantees for Greedy Maximization of Non-submodular Functions with Applications**. In: *Proc. of ICML*, 498–507.

Borodin, Allan, Joachim von zur Gathen, and John E. Hopcroft (1982). **Fast Parallel Matrix and GCD Computations**. *Information and Control* 52:3, 241–256.

Breuer, Adam, Eric Balkanski, and Yaron Singer (2020). **The FAST Algorithm for Submodular Maximization**. In: *Proc. of ICML*, 1134–1143.

Buchbinder, Niv and Moran Feldman (2018). **Deterministic Algorithms for Submodular Maximization Problems**. *ACM Transactions on Algorithms* 14:3, 32:1–32:20.

Buchbinder, Niv, Moran Feldman, and Mohit Garg (2018). **Deterministic $(1/2 + \epsilon)$-Approximation for Submodular Maximization over a Matroid**. *CoRR* abs/1807.05532.

– (2019). **Deterministic (½ + ϵ)-Approximation for Submodular Maximization over a Matroid**. In: *Proc. of SODA*, 241–254.
Buchbinder, Niv, Moran Feldman, Joseph Naor, et al. (2012). **A Tight Linear Time (1/2)-Approximation for Unconstrained Submodular Maximization**. In: *Proc. of FOCS*, 649–658.
– (2014). **Submodular Maximization with Cardinality Constraints**. In: *proc. of SODA*, 1433–1452.
– (2015). **A Tight Linear Time (1/2)-Approximation for Unconstrained Submodular Maximization**. *SIAM Journal of Computing* 44:5, 1384–1402.
Călinescu, Gruia et al. (2007). **Maximizing a Submodular Set Function Subject to a Matroid Constraint**. In: *Proc. of IPCO*, 182–196.
– (2011). **Maximizing a Monotone Submodular Function Subject to a Matroid Constraint**. *SIAM Journal of Computing* 40:6, 1740–1766.
Cancer Genome Atlas Research Network et al. (2013). **The Cancer Genome Atlas Pan-Cancer analysis project.** *Nature Genetics* 45:10, 1113–20.
Celis, L. Elisa et al. (2018). **Fair and Diverse DPP-Based Data Summarization**. In: *Proc. of ICML*, 715–724.
Chakrabarti, Amit and Sagar Kale (2015). **Submodular maximization meets streaming: matchings, matroids, and more**. *Mathematical Programming* 154:1-2, 225–247.
Chekuri, Chandra, Shalmoli Gupta, and Kent Quanrud (2015). **Streaming Algorithms for Submodular Function Maximization**. In: *Proc. of ICALP*, 318–330.
Chekuri, Chandra and Martin Pál (2005). **A Recursive Greedy Algorithm for Walks in Directed Graphs**. In: *Proc. of FOCS*, 245–253.
Chekuri, Chandra and Kent Quanrud (2019a). **Parallelizing greedy for submodular set function maximization in matroids and beyond**. In: *Proc. of STOC*, 78–89.
– (2019b). **Submodular Function Maximization in Parallel via the Multilinear Relaxation**. In: *proc. of SODA*, 303–322.

Chekuri, Chandra, Jan Vondrák, and Rico Zenklusen (2010). **Dependent Randomized Rounding via Exchange Properties of Combinatorial Structures**. In: *Proc. of FOCS*, 575–584.
– (2014). **Submodular Function Maximization via the Multilinear Relaxation and Contention Resolution Schemes**. *SIAM Journal of Computing* 43:6, 1831–1879.
Chen, Lin, Moran Feldman, and Amin Karbasi (2019). **Unconstrained submodular maximization with constant adaptive complexity**. In: *proc. of STOC*, 102–113.
Chierichetti, Flavio et al. (2019). **Matroids, Matchings, and Fairness**. In: *Proc. of AISTATS*, 2212–2220.
Chistov, Alexander L. (1985). **Fast parallel calculation of the rank of matrices over a field of arbitrary characteristic**. In: *Proc. of FCT*, 63–69.
Chouldechova, Alexandra (2017). **Fair prediction with disparate impact: A study of bias in recidivism prediction instruments**. *Big data* 5:2, 153–163.
Conforti, Michele and Gérard Cornuéjols (1984). **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.
Corbett-Davies, Sam et al. (2017a). **Algorithmic Decision Making and the Cost of Fairness**. In: *Proc. of KDD*, 797–806.
– (2017b). **Algorithmic decision making and the cost of fairness**. In: *Proc. of KDD*, 797–806.
Cornuejols, Gerard, Marshall L. Fisher, and George L. Nemhauser (1977). **Location of Bank Accounts to Optimize Float: An Analytic Study of Exact and Approximate Algorithms**. *Management Science* 23:8, 789–810.
Cowgill, Bo and Catherine E Tucker (2019). **Economics, fairness and algorithmic bias**. *preparation for: Journal of Economic Perspectives*.
Cressie, Noel (2015). **Statistics for spatial data**. John Wiley & Sons.

Das, Abhimanyu and David Kempe (2011). **Submodular meets Spectral: Greedy Algorithms for Subset Selection, Sparse Approximation and Dictionary Selection**. In: *Proc. of ICML*, 1057–1064.
Donini, Michele et al. (2018). **Empirical Risk Minimization Under Fairness Constraints**. In: *Proc. of NeurIPS*, 2796–2806.
Doskoc, Vanja et al. (2019). **Non-Monotone Submodular Maximization with Multiple Knapsacks in Static and Dynamic Settings**. *CoRR* abs/1911.06791.
Elenberg, Ethan R., Alexandros G. Dimakis, et al. (2017). **Streaming Weak Submodularity: Interpreting Neural Networks on the Fly**. In: *Proc. of NeurIPS*, 4044–4054.
Elenberg, Ethan R., Rajiv Khanna, et al. (2018). **Restricted Strong Convexity Implies Weak Submodularity**. *The Annals of Statistics* 46:6B, 3539–3568.
Elfeki, Mohamed et al. (2019). **GDPP: Learning Diverse Generations using Determinantal Point Processes**. In: *Proc. of ICML*. Vol. 97, 1774–1783.
Ene, Alina and Huy L. Nguyen (2016). **Constrained Submodular Maximization: Beyond 1/e**. In: *Proc. of FOCS*, 248–257.
– (2019). **Submodular Maximization with Nearly-optimal Approximation and Adaptivity in Nearly-linear Time**. In: *Proc. of SODA*, 274–282.
Fahrbach, Matthew, Vahab S. Mirrokni, and Morteza Zadimoghaddam (2019a). **Non-monotone Submodular Maximization with Nearly Optimal Adaptivity and Query Complexity**. In: *Proc. of ICML*, 1833–1842.
– (2019b). **Submodular Maximization with Nearly Optimal Approximation, Adaptivity and Query Complexity**. In: *Proc. of SODA*, 255–273.
Feige, Uriel, Vahab S. Mirrokni, and Jan Vondrák (2007). **Maximizing Non-Monotone Submodular Functions**. In: *Proc. of FOCS*, 461–471.

Feige, Uriel, Vahab S. Mirrokni, and Jan Vondrák (2011). **Maximizing Non-monotone Submodular Functions**. *SIAM Journal of Computing* 40:4, 1133–1153.
Feige, Uriel and Jan Vondrák (2010). **The Submodular Welfare Problem with Demand Queries**. *Theory of Computing* 6:1, 247–290.
Feldman, Michael et al. (2015). **Certifying and Removing Disparate Impact**. In: *Proc. of SIGKDD*, 259–268.
Feldman, Moran, Christopher Harshaw, and Amin Karbasi (2017). **Greed Is Good: Near-Optimal Submodular Maximization via Greedy Optimization**. In: *Proc. of COLT*, 758–784.
– (2020). **Simultaneous Greedys: A Swiss Army Knife for Constrained Submodular Maximization**. *CoRR* abs/2009.13998.
Feldman, Moran, Amin Karbasi, and Ehsan Kazemi (2018). **Do Less, Get More: Streaming Submodular Maximization with Subsampling**. In: *Proc. of NeurIPS*, 730–740.
Feldman, Moran, Joseph Naor, and Roy Schwartz (2011a). **A Unified Continuous Greedy Algorithm for Submodular Maximization**. In: *Proc. of FOCS*, 570–579.
– (2011b). **Nonmonotone Submodular Maximization via a Structural Continuous Greedy Algorithm - (Extended Abstract)**. In: *Proc. of ICALP*, 342–353.
Filmus, Yuval and Justin Ward (2012). **A Tight Combinatorial Algorithm for Submodular Maximization Subject to a Matroid Constraint**. In: *Proc. of FOCS*, 659–668.
Fisher, M., George Nemhauser, and Laurence Wolsey (Mar. 1978). **An analysis of approximations for maximizing submodular set functions - II**. *Mathematical Programming* 14, 73–87.
Ghadiri, Mehrdad and Mark Schmidt (2019). **Distributed Maximization of "Submodular plus Diversity" Functions for Multi-label Feature Selection on Huge Datasets**. In: *Proc. of AISTATS*, 2077–2086.
Gharan, Shayan Oveis and Jan Vondrák (2011). **Submodular Maximization by Simulated Annealing**. In: *Proc. of SODA*, 1098–1116.

Gillenwater, Jennifer, Alex Kulesza, and Ben Taskar (2012). **Near-Optimal MAP Inference for Determinantal Point Processes**. In: *Proc. of NIPS*, 2744–2752.
Goemans, Michel X. et al. (2009). **Approximating submodular functions everywhere**. In: *Proc. of SODA*, 535–544.
Gong, Boqing et al. (2014). **Diverse Sequential Subset Selection for Supervised Video Summarization**. In: *Proc. of NIPS*, 2069–2077.
Grgic-Hlaca, Nina, Elissa M. Redmiles, et al. (2018). **Human Perceptions of Fairness in Algorithmic Decision Making: A Case Study of Criminal Risk Prediction**. In: *Proc. of WWW*, 903–912.
Grgic-Hlaca, Nina, Muhammad Bilal Zafar, K. Gummadi, et al. (2016). **The Case for Process Fairness in Learning: Feature Selection for Fair Decision Making**. In: *NeurIPS Symposium on Machine Learning and the Law*.
Grgic-Hlaca, Nina, Muhammad Bilal Zafar, Krishna P. Gummadi, et al. (2018). **Beyond Distributive Fairness in Algorithmic Decision Making: Feature Selection for Procedurally Fair Learning**. In: *Proc. of AAAI*, 51–60.
Gupta, Anupam et al. (2010). **Constrained Non-monotone Submodular Maximization: Offline and Secretary Algorithms**. In: *proc. of WINE*, 246–257.
Halabi, Marwa El et al. (2020). **Fairness in Streaming Submodular Maximization: Algorithms and Hardness**. In: *Proc. of NeurIPS*.
Hardt, Moritz, Eric Price, and Nati Srebro (2016). **Equality of Opportunity in Supervised Learning**. In: *Proc. of NIPS*, 3315–3323.
Ibarra, Oscar H., Shlomo Moran, and Louis E. Rosier (1980). **A Note on the Parallel Complexity of Computing the Rank of Order n Matrices**. *Information Processing Letters* 11:4/5, 162.
Iyer, Rishabh K. and Jeff A. Bilmes (2013). **Submodular Optimization with Submodular Cover and Submodular Knapsack Constraints**. In: *Proc. of NIPS*, 2436–2444.

Iyer, Rishabh K. and Jeff A. Bilmes (2015). **Submodular Point Processes with Applications to Machine learning**. In: *Proc. of AISTATS*.

Jain, Prateek, Ambuj Tewari, and Purushottam Kar (2014). **On Iterative Hard Thresholding Methods for High-dimensional M-Estimation**. In: *Proc. of NIPS*, 685–693.

Jalali, Ali, Christopher C. Johnson, and Pradeep Ravikumar (2011). **On Learning Discrete Graphical Models using Greedy Methods**. In: *Proc. of NIPS*, 1935–1943.

Kang, Byungkon (2013). **Fast determinantal point process sampling with application to clustering**. In: *Proc. of NIPS*, 2319–2327.

Karimi, Mohammad Reza et al. (2017). **Stochastic Submodular Maximization: The Case of Coverage Functions**. In: *proc. of NIPS*, 6853–6863.

Karp, Richard M., Eli Upfal, and Avi Wigderson (1988). **The Complexity of Parallel Search**. *Journal of Computer and System Science* 36:2, 225–253.

Kathuria, Tarun, Amit Deshpande, and Pushmeet Kohli (2016). **Batched Gaussian Process Bandit Optimization via Determinantal Point Processes**. In: *Proc. of NIPS*, 4206–4214.

Khanna, Rajiv et al. (2017). **Scalable Greedy Feature Selection via Weak Submodularity**. In: *Proc. of AISTATS*, 1560–1568.

Khuller, Samir, Anna Moss, and Joseph Naor (1999). **The Budgeted Maximum Coverage Problem**. *Information Processing Letters* 70:1, 39–45.

Kilbertus, Niki et al. (2017). **Avoiding Discrimination through Causal Reasoning**. In: *Proc. of NIPS*, 656–666.

Kim, Michael P., Omer Reingold, and Guy N. Rothblum (2018). **Fairness Through Computationally-Bounded Awareness**. In: *Proc. of NeurIPS*, 4847–4857.

Kleinberg, Jon M., Sendhil Mullainathan, and Manish Raghavan (2017). **Inherent Trade-Offs in the Fair Determination of Risk Scores**. In: *Proc. of ITCS*. Vol. 67, 43:1–43:23.

Krause, Andreas and Volkan Cevher (2010). **Submodular Dictionary Selection for Sparse Representation**. In: *Proc.of ICML*, 567–574.
Krause, Andreas, Ajit Paul Singh, and Carlos Guestrin (2008). **Near-Optimal Sensor Placements in Gaussian Processes: Theory, Efficient Algorithms and Empirical Studies**. *Journal of Machine Learning Research* 9, 235–284.
Kulesza, Alex and Ben Taskar (2010). **Structured Determinantal Point Processes**. In: *Proc. of NIPS*, 1171–1179.
– (2011). **k-DPPs: Fixed-Size Determinantal Point Processes**. In: *Proc. of ICML*, 1193–1200.
– (2012). **Determinantal Point Processes for Machine Learning**. Now Publishers.
Kulik, Ariel, Hadas Shachnai, and Tami Tamir (2009). **Maximizing submodular set functions subject to multiple linear constraints**. In: *Proc. of SODA*, 545–554.
Kumar, Ravi et al. (2015). **Fast Greedy Algorithms in MapReduce and Streaming**. *ACM Transactions on Parallel Computing* 2:3, 14:1–14:22.
Kusner, Matt J. et al. (2017). **Counterfactual Fairness**. In: *Proc. of NIPS*, 4066–4076.
Lahoti, Preethi et al. (2020). **Fairness without Demographics through Adversarially Reweighted Learning**. In: *Proc. of NeurIPS*.
Larson, Jeff, Marjorie Roswell, and Vaggelis Atlidakis (2016). *Data and Analysis for 'How We Analyzed the COMPAS Recidivism Algorithm'*. https://github.com/propublica/compas-analysis.
Lee, Jon et al. (2009). **Non-monotone submodular maximization under matroid and knapsack constraints**. In: *Proc. of STOC*. Ed. by Michael Mitzenmacher, 323–332.
Li, Wenzheng, Paul Liu, and Jan Vondrák (2020). **A polynomial lower bound on adaptive complexity of submodular maximization**. *CoRR* abs/2002.09130.

Liu, Ji, Jieping Ye, and Ryohei Fujimaki (2014). **Forward-Backward Greedy Algorithms for General Convex Smooth Functions over A Cardinality Constraint**. In: *Proc. of ICML*. Vol. 32, 503–511.

Lovász, László (1982). **Submodular functions and convexity**. In: *Proc. of ISMP*. Ed. by Achim Bachem, Bernhard Korte, and Martin Grötschel, 235–257.

Macchi, Odile (1975). **The coincidence approach to stochastic point processes**. *Advances in Applied Probability* 7:1, 83–122. DOI: 10.2307/1425855.

Mairal, Julien and Bin Yu (2012). **Complexity Analysis of the Lasso Regularization Path**. In: *Proc. of ICML*.

Mestre, Julián (2006). **Greedy in Approximation Algorithms**. In: *Proc. of ESA*. Vol. 4168. Lecture Notes in Computer Science, 528–539.

Mirrokni, Vahab S. and Morteza Zadimoghaddam (2015). **Randomized Composable Core-sets for Distributed Submodular Maximization**. In: *Proc. of STOC*, 153–162.

Mirzasoleiman, Baharan, Ashwinkumar Badanidiyuru, and Amin Karbasi (2016). **Fast Constrained Submodular Maximization: Personalized Data Summarization**. In: *Proc. of ICML*. Vol. 48, 1358–1367.

Mirzasoleiman, Baharan, Ashwinkumar Badanidiyuru, Amin Karbasi, et al. (2015). **Lazier Than Lazy Greedy**. In: *Proc. of AAAI*, 1812–1818.

Mirzasoleiman, Baharan, Stefanie Jegelka, and Andreas Krause (2018a). **Streaming Non-Monotone Submodular Maximization: Personalized Video Summarization on the Fly**. In: *Proc. of AAAI*, 1379–1386.

– (2018b). **Streaming Non-Monotone Submodular Maximization: Personalized Video Summarization on the Fly**. In: *proc. of AAAI*, 1379–1386.

Mirzasoleiman, Baharan, Amin Karbasi, Ashwinkumar Badanidiyuru, et al. (2015). **Distributed Submodular Cover: Succinctly Summarizing Massive Data**. In: *Proc. of NeurIPS*, 2881–2889.

Mirzasoleiman, Baharan, Amin Karbasi, and Andreas Krause (2017). **Deletion-Robust Submodular Maximization: Data Summarization with "the Right to be Forgotten"**. In: *Proc. of ICML*. Vol. 70, 2449–2458.

Mirzasoleiman, Baharan, Amin Karbasi, Rik Sarkar, et al. (2013). **Distributed Submodular Maximization: Identifying Representative Elements in Massive Data**. In: *Proc. of NIPS*, 2049–2057.

– (2016). **Distributed Submodular Maximization**. *Journal of Machine Learning Research* 17, 238:1–238:44.

Mirzasoleiman, Baharan, Morteza Zadimoghaddam, and Amin Karbasi (2016). **Fast Distributed Submodular Cover: Public-Private Data Summarization**. In: *Proc. of NeurIPS*, 3594–3602.

Mulmuley, Ketan (1987). **A fast parallel algorithm to compute the rank of a matrix over an arbitrary field**. *Combinatorica* 7:1, 101–104.

Needell, Deanna and Joel A. Tropp (2010). **CoSaMP: iterative signal recovery from incomplete and inaccurate samples**. *Communication of the ACM* 53:12, 93–100.

Nemhauser, George L. and Laurence A. Wolsey (1978). **Best Algorithms for Approximating the Maximum of a Submodular Set Function**. *Mathematics of Operation Research* 3:3, 177–188.

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

Pin Calmon, Flávio du et al. (2017). **Optimized Pre-Processing for Discrimination Prevention**. In: *Proc. of NIPS*, 3992–4001.

Ponte Barbosa, Rafael da et al. (2015). **The Power of Randomization: Distributed Submodular Maximization on Massive Datasets**. In: *Proc. of ICML*, 1236–1244.

Qian, Chao, Chao Bian, and Chao Feng (2020). **Subset Selection by Pareto Optimization with Recombination**. In: *Proc. of AAAI*, 2408–2415.

Qian, Chao, Yang Yu, and Zhi-Hua Zhou (2015). **Subset Selection by Pareto Optimization**. In: *Proc. of NIPS*, 1774–1782.
Qian, Sharon and Yaron Singer (2019). **Fast Parallel Algorithms for Statistical Subset Selection Problems**. In: *Proc. of NeurIPS*, 5073–5082.
Quinzan, Francesco, Vanja Doskoc, et al. (2021). **Adaptive Sampling for Fast Constrained Maximization of Submodular Functions**. In: *Proc. of AISTATS*, 964–972.
Quinzan, Francesco, Andreas Göbel, et al. (2019). **Greedy Maximization of Functions with Bounded Curvature under Partition Matroid Constraints**. In: *Proc. of AAAI*, 2272–2279.
Quinzan, Francesco, Rajiv Khanna, et al. (2022). **Fast Feature Selection with Fairness Constraints**. *CoRR* abs/2202.13718.
Rigollet, Philippe and Alexandre Tsybakov (2010). **Exponential Screening and optimal rates of sparse estimation**. *The Annals of Statistics* 39, 731–771.
Rudin, Cynthia, Caroline Wang, and Beau Coker (2018). **The age of secrecy and unfairness in recidivism prediction**. *arXiv:1811.00731*.
Sakaue, Shinsaku (2020). **On Maximization of Weakly Modular Functions: Guarantees of Multi-stage Algorithms, Tractability, and Hardness**. In: *Proc. of AISTATS*, 22–33.
Sapp, Benjamin and Ben Taskar (2013). **MODEC: Multimodal Decomposable Models for Human Pose Estimation**. In: *Proc. of CVPR*.
Sebastiani, Paolo and Henry P. Wynn (2002). **Maximum entropy sampling and optimal Bayesian experimental design**. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 62:1, 145–157.
Selbst, Andrew D. et al. (2019). **Fairness and Abstraction in Sociotechnical Systems**. *Proc. of FAccT*.
Shannon, Claude E (1948). **A mathematical theory of communication**. *The Bell system technical journal* 27:3, 379–423.
Shewry, Michael C and Henry P Wynn (1987). **Maximum entropy sampling**. *Journal of applied statistics* 14:2, 165–170.

Singh, Amarjeet et al. (2009). **Efficient Informative Sensing using Multiple Robots**. *Journal of Artificial Intelligence Research* 34, 707–755.

Staib, Matthew, Bryan Wilder, and Stefanie Jegelka (2019). **Distributionally Robust Submodular Maximization**. In: *Proc. of AISTATS*, 506–516.

Thompson, Steven K. (1990). **Adaptive Cluster Sampling**. *Journal of the American Statistical Association* 85:412, 1050–1059.

Trankell, Arne (1973). **Review: Reliability of Evidence. Methods for Analysing and Assessing Witness Statements**. *The Journal of Criminal Law* 37:2, 152–152. DOI: 10.1177/002201837303700214.

Vondrák, Jan (2010). **Submodularity and curvature: the optimal algorithm**. *RIMS Kokyuroku Bessatsu* B23, 253–266.

Vondrák, Jan, Chandra Chekuri, and Rico Zenklusen (2011). **Submodular function maximization via the multilinear relaxation and contention resolution schemes**. In: *Proc. of STOC*, 783–792.

Wei, Kai, Rishabh K. Iyer, and Jeff A. Bilmes (2015). **Submodularity in Data Subset Selection and Active Learning**. In: *Proc. of ICML*. Vol. 37, 1954–1963.

Woodworth, Blake E. et al. (2017). **Learning Non-Discriminatory Predictors**. In: *Proc. of COLT*. Vol. 65, 1920–1953.

Yuan, Xiao-Tong, Ping Li, and Tong Zhang (2017). **Gradient Hard Thresholding Pursuit**. *Journal of Machine Learning Research* 18, 166:1–166:43.

Zafar, Muhammad Bilal, Isabel Valera, Manuel Gomez-Rodriguez, and Krishna P. Gummadi (2017a). **Fairness Beyond Disparate Treatment & Disparate Impact: Learning Classification without Disparate Mistreatment**. In: *Proc. of WWW*, 1171–1180.

– (2017b). **Fairness Constraints: Mechanisms for Fair Classification**. In: *Proc. of AISTATS*. Vol. 54, 962–970.

Zafar, Muhammad Bilal, Isabel Valera, Manuel Gomez-Rodriguez, Krishna P. Gummadi, and Adrian Weller (2017). **From Parity to**

Preference-based Notions of Fairness in Classification. In: *Proc. of NIPS*, 229–239.

Zhang, Haifeng and Yevgeniy Vorobeychik (2016). **Submodular Optimization with Routing Constraints**. In: *Proc. of AAAI*. Ed. by Dale Schuurmans and Michael P. Wellman, 819–826.

Zhu, Zhengyuan and Michael L Stein (2006). **Spatial sampling design for prediction with estimated parameters**. *Journal of agricultural, biological, and environmental statistics* 11:1, 24.

Zimmerman, Dale L (2006). **Optimal network design for spatial prediction, covariance parameter estimation, and empirical prediction**. *Environmetrics: The official journal of the International Environmetrics Society* 17:6, 635–652.

List of Publications

Articles in Refereed Journals

Evolutionary algorithms and submodular functions: benefits of heavy-tailed mutations. *Natural Computing* 20:3, 561–575. Quinzan, Francesco, Andreas Göbel, Markus Wagner, et al.

Articles in Refereed Conference Proceedings

Adaptive Sampling for Fast Constrained Maximization of Submodular Functions. In: *Proc. of AISTATS*, 964–972. Quinzan, Francesco, Vanja Doskoc, et al.

Non-Monotone Submodular Maximization with Multiple Knapsacks in Static and Dynamic Settings. In: *Proc. of ECAI*, 435–442. Quinzan, Francesco, Vanja Dostoč, et al.

Greedy Maximization of Functions with Bounded Curvature under Partition Matroid Constraints. In: *Proc. of AAAI*, 2272–2279. Quinzan, Francesco, Andreas Göbel, Ralf Rothenberger, et al.