Sparsification of Submodular Functions

J Kudla

Mathematical Institute

University of Oxford

A dissertation submitted for the degree of

Master of Science
Mathematics and Foundations of Computer Science

August 31, 2022

Contents

Chapter 1

Introduction

Ambitions to cast algorithmic problems as instances of more general problems are perhaps as old as the discipline of combinatorial optimisation itself. A prominent example is the hierarchy of problems depicted in Fig. 1.1. Finding maximum bipartite matchings in unweighted graphs is possible efficiently by the Hopcroft-Karp algorithm [HK73]. The maximum bipartite matching problem turns out to be a special case of maximum flow, which can be solved a bit less efficiently by numerous algorithms, the Edmonds-Karp algorithm being just one example [Cor+09]. Maximum flow, in turn, appears as a special case of minimum-cost flow, which again is a special case of linear programming [AMO93]. As one moves further up the line of generalisation, running time guarantees are generally sacrificed. However, a huge advantage becomes evident: More general algorithms solve a broader class of problems and can be used in a "blackbox" fashion, as opposed to specialised algorithms which are only applicable to a very specific problem.

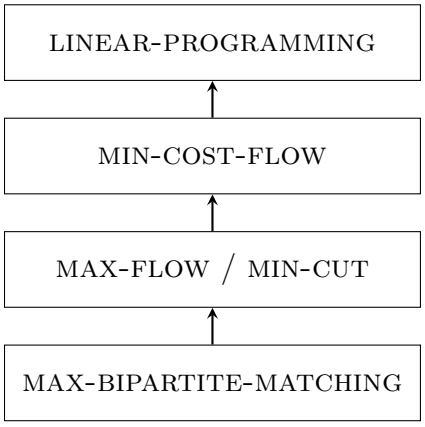


Figure 1.1: Hierarchy of increasingly more general problems with increasingly worse known running time bounds

In this spirit, if we focus on graph cuts, coverage and facility location problems, we discover that they all boil down to the minimisation or maximisation of *submodular functions*. This common generalisation is well-studied and still a hot research topic,

investigated from different perspectives. While there is a lot of work on the general minimisation and maximisation problems, including hardness results and approximation algorithms, the use of *sparsification* as a technique to approximately represent a decomposable submodular function seems rather new. It has been described by Rafiey and Yoshida [RY22] and applied as a preprocessing step to improve previously known algorithms for various combinatorial optimisation problems.

However, it is worth noting that "sparsifying" objects is a celebrated technique in algorithm design that dates back more than 25 years and has been successfully applied to many problems: From the "Sparsification Lemma" for k-SAT by Impagliazzo and Paturi [IP01] over the sparsification of cuts in graphs [BK96; LS18] and hypergraphs [SY19; KK15; CKN20] to the sparsification of submodular functions [RY22].

The objective of this dissertation is pushing the research towards efficient sparsifier constructions for important classes of submodular functions and beyond. We constantly ask the question whether and how a known-to-exist sparsifier of a certain size can be constructed efficiently, giving the computational aspect a key role in this work.

Existence Proofs and Efficient Algorithms

The general sparsification algorithm introduced in Chapter 3, referred to as the *core algorithm* in this work, has a linear running time dependence on the size of the domain, which is exponential for submodular functions. No significant speed-up is known for the general submodular case. In fact, even for monotone submodular functions there a hardness result by Bai et al. [Bai+16] that rules out a polynomial-time approximation of the ratio of such functions up to a factor better than $\sqrt{n}$ under $\mathsf{P} \neq \mathsf{NP}$, implying that our core algorithm most likely cannot be implemented in polynomial time.

This motivates the study of special cases that do admit a sparsifier of small size and its construction in polynomial time at the same time. In this work, we demonstrate that important classes of submodular functions appearing in applications such as (hyper-) graph cuts and sensor placement are instances of these special cases.

Structure and Highlighted Results

After giving background on submodular functions and sparsification, we present our results in three technical chapters. Since the central definitions are only introduced in the next chapter, we concisely summarise our contributions in the beginning of

each corresponding chapter rather than listing them here in full detail. Therefore, the reader is advised to read the beginnings of Chapter 3, 4 and 5 for an overview of all original results in this work. Among those, we want to highlight:

- Fixes in response to the work by Rafiey and Yoshida [RY22]: We disprove their size bounds by a counterexample and show that they are indeed correct under the additional assumption of monotonicity. Moreover, we extend their algorithm beyond submodular functions and fix an invalid step in their proof.
- New polynomial-time constructions of small sparsifiers for the important classes of submodular functions of low curvature and bounded arity. The low curvature class can perhaps be tackled under knapsack constraints as well, which a paper by Perrault et al. [Per+21] suggests. However, their algorithm and its analysis are flawed; we disprove this part of their paper and work towards a correction.
- Polynomial-time construction of small sparsifiers for classes of set functions that are efficiently minimisable and *arity reducible*, the letter being a notion we introduce to capture the essential requirements of our sparsifier construction. Specifically, we show that k-submodular of bounded arity are of this type.

Finally, we conclude with open questions and possible directions for future research.

Chapter 2

Preliminaries

In this chapter, we define terms and introduce notations for the technical part of the dissertation. This also involves some background information and motivation where appropriate. Terms and notations specific to a particular section are introduced where needed, hence they are not part of this chapter. Furthermore, this chapter is intended to make the dissertation more self-contained and accessible to a broader audience of readers who are not familiar with the concepts of submodularity and sparsification.

2.1 Submodularity

The most important objects studied in this work are submodular functions. Given a *ground set* E, a function $f : 2^E \to \mathbb{R}$ is referred to as *set function*. If f satisfies

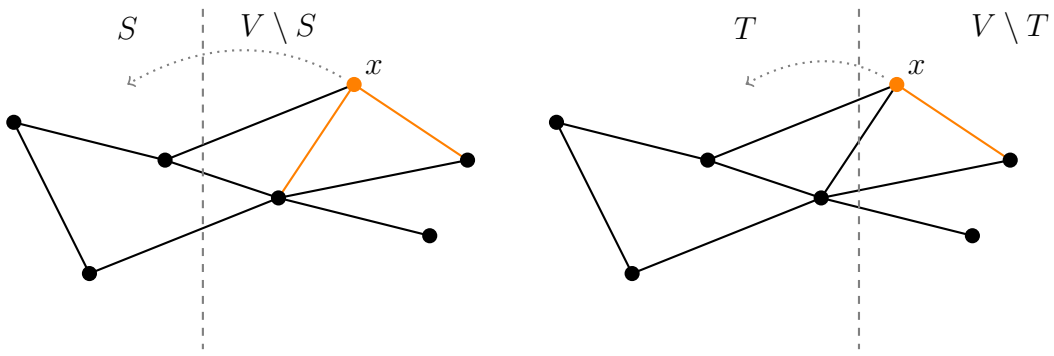
$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B) \tag{2.1}$$

for all $A, B \subseteq E$, we say that f is *submodular*. If equality holds for arguments, we f is *modular*. This is restrictive in the sense that it implies $f(S) = \sum_{e \in S} w_e$ for some weights $(w_e)_{e \in E}$, which is why modular functions are sometimes called *linear*.

Given a set function $f : 2^E \to \mathbb{R}$, we define the *marginal gain* of adding an element $e \in E$ to a set $S \subseteq E$ as

$$\Delta_e (f \mid S) := f(S \cup \{e\}) - f(S).$$

We also define $\Delta_T (f \mid T) := f(S \cup \{T\}) - f(S)$, the marginal gain of augmenting a set S with an entire set T of items. Marginal gains play in important role in the study of submodular functions, which our work reflects. More specifically, we make use of the *diminishing returns property* – an equivalent characterisation of submodular functions. It states that the gain of adding an element does not increase as we enlargen the set it is added to.



(a) Gain of adding x to the cut S (b) Gain of adding x to the cut T

Figure 2.1: Diminishing Returns Property in Graph Cuts

Fact 2.1 (Diminishing Returns Property)**.** *Let $f : 2^E \to \mathbb{R}$ be a set function. Then f is submodular if and only if*

$$\Delta_e(f \mid T) \leq \Delta_e(f \mid S)$$

for all $T \supseteq S$ and $e \notin T$.

This fact is well-known in the field and easy to prove yet a non-trivial second perspective on submodular functions. Viewing them through the lens of the diminishing returns property is insightful in many proofs and applications.

Example 2.2. Given an undirected graph $G = (V, E)$, we define the *cut function*

$$F : 2^V \to \mathbb{R}, \quad S \mapsto |E(S, V \setminus S)|$$

where $E(S, V \setminus S)$ denotes the set of edges $\{u, v\}$ such that $u \in S$ and $v \in V \setminus S$. An elegant way of showing that F is submodular is by establishing Fact 2.1. To this end, consider $S \subseteq T \subseteq V$ and $x \in V \setminus T$. We show that $\Delta_x(F \mid T) \leq \Delta_x(F \mid S)$. Observe that $\Delta_x(F \mid S)$ is the number of edges incident to x that are cut when moving x from $V \setminus S$ to S, see the orange edges in Fig. 2.1a. The same reasoning applies to T. However, all endpoints of non-orange edges belong to T as well (since $T \supseteq S$), plus possibly a few more (see Fig. 2.1b), thereby reducing the number of orange edges. We conclude that F satisfies the diminishing returns property, hence it is submodular.

We say that a set function $f : 2^E \to \mathbb{R}$ is *monotone* if $f(T) \geq f(S)$ for all $T \supseteq S$ and *normalised* if $f(\varnothing) = 0$. Note that f is monotone if and only if all marginal gains are non-negative, i. e., $\Delta_e(f \mid S) \geq 0$ for all $S \subseteq E$ and $e \notin S$.

For a submodular function $f : 2^E \to \mathbb{R}$, the set

$$\mathcal{B}(f) := \left\{x \in \mathbb{R}^E \mid \forall S \subseteq E : x(S) \leq f(S) \text{ and } x(E) = f(E)\right\}$$

can be shown to be a polyhedron and is called the *base polyhedron* of f. In this definition, $x(S) := \sum_{e \in S} x_e$. Moreover, we let $\mathbb{1}_e \in \mathbb{R}^E$ denote the unit vector w. r. t. the coordinate $e \in E$. This extends to subsets $S \subseteq E$ via $\mathbb{1}_S := \sum_{e \in S} \mathbb{1}_e$. For any polyhedron P, we let $\mathsf{EX}(P)$ denote the set of extreme points of P. In particular, $\mathsf{EX}(\mathcal{B}(f))$ is the set of extreme points of the base polyhedron of f. It can be shown that $\mathsf{EX}(\mathcal{B}(f))$ is finite and non-empty for any submodular function f [Fuj91].

Given a ground set E of size n, the space required to represent a (submodular) function $f : 2^E \to \mathbb{R}$ explicitly is generally exponential in n. Therefore, submodular functions are often studied in the *evaluation oracle model*: We assume an oracle that, given any set $S \subseteq E$, returns the value $f(S)$ in time $\mathcal{O}(\mathrm{EO})$. It is not only sensible to choose this model to cope with memory limitations but also for the reason of flexibility, since this model helps abstracting away from the concrete implementation of the evaluation oracle (there might be different ways to implement it).

In the evaluation oracle model, it is sensible to consider the minimisation and maximisation problems, i. e., the problems of computing sets $S \subseteq E$ attaining $\min_{S \subseteq E} f(S)$ and $\max_{S \subseteq E} f(S)$, The maximisation problem is known to be NP-hard, for instance by a trivial reduction from MAX-CUT. The following algorithms are notable:

- Polynomial-time algorithm to compute $\min_{S \subseteq E} f(S)$ exactly. This is a celebrated result by Grötschel, Lovász and Schrijver [GLS81]. Their algorithm uses the ellipsoid method. Later on, fully combinatorial polynomial-time algorithms have been described [Sch00; IFF01; Orl07], the fastest we know being the algorithm by Orlin [Orl07] running in time $\mathcal{O}\left((n^6 + n^5 \cdot \mathrm{EO}) \log n\right)$.
- Greedy $(1 - 1/e)$-approximation for $\max_{S \subseteq E, |S| \leq k} f(S)$ by Nemhauser, Wolsey and Fisher [NWF78], i. e., the maximisation of f subject to a cardinality constraint $|S| \leq k$ (for any $k \leq n$, so $k = n$ yields the unconstraint problem).

Polynomial-time submodular minimisation is paramount to many of the algorithms in this work. We frequently use submodular minimisation as a subroutine when designing sparsification algorithms. The ideas behind the second algorithm become relevant in Section 4.1 as they can be exploited to approximately maximise the ratio of two submodular functions with special properties.

2.2 Sparsification

Given a domain $\mathcal{D}$, we call a function $F : \mathcal{D} \to \mathbb{R}$ *decomposable* if F can be written as the point-wise sum $F = f_1 + \cdots + f_N$, that is, $F(S) = \sum_{i=1}^{N} f_i(S)$ for all $S \in \mathcal{D}$, of "smaller" functions $f_1, \ldots, f_N$. The f_i's are referred to as *constituents* of the decomposition $F = f_1 + \cdots + f_N$. In many applications, F is submodular and can be written as a sum of submodular f_i's that often have "simpler" evaluation oracles and smaller representations, as well as stronger properties (e. g. codomain $\{0, 1\}$).

The idea of sparsification is to select a small number $M \ll N$ of the f_i's, say $f_{i_1}, \ldots, f_{i_M}$, that approximately represent F on $\mathcal{D}$ or a subdomain $\mathcal{D}' \subseteq \mathcal{D}$ of our interest. Simply taking $F = f_{i_1} + \cdots + f_{i_M}$ is obviously insufficient as it leads to an underestimation of F that may be very small and not within a reasonable bound of F, no matter how cleverly we choose $f_{i_1}, \ldots, f_{i_M}$. This observation and the desire for flexibility and generality suggest a *weighted* approach that turns out to allow for many interesting results treated in this dissertation. Perhaps most importantly, the notion of sparsifier coinincides with notions previously defined in the literature, e. g. with cut sparsifiers when specialising to graph cuts.

Guided by these observations, we call a vector $w = (w_1, \ldots, w_N) \in \mathbb{R}^N$ an *ε-sparsifier* for F if the inequality

$$(1 - \varepsilon)F(S) \leq F'(S) \leq (1 + \varepsilon)F(S) \tag{2.2}$$

holds for all $S \in \mathcal{D}$, where $F'(S) := \sum_{i=1}^{N} w_i f_i(S)$. The *size* of w

$$\mathsf{size}(w) := |\{i \in \{1, \ldots, N\}\} \mid w_i \neq 0\}|$$

is the number of non-zero entries in w.

Chapter 3

The Core Algorithm

In this chapter, we describe the core of all sparsification algorithms – a randomised sampling rountine initially described by Rafiey and Yoshida [RY22] for submodular functions. We observe that it is largely independent of submodularity, leading to a more general sparsification algorithm for decomposable functions. The part where submodularity becomes relevant is carefully analysed in this chapter. Most of it closely follows Section 3 of [RY22].

Contributions:

- Generalised sparsification algorithm and analysis beyond submodular functions, fixing an invalid step in the analysis by Rafiey and Yoshida [RY22].
- Concentration bound: In addition to the correctness proof and bound on the expected size of the resulting sparsifier, we establish a concentration bound revealing that it is very unlikely to exceed 3/2-times the expected size.
- Peak contributions counterexample: Rafiey and Yoshida [RY22] claim that $\sum_{i=1}^{N} p_i \leq Bn$ for any function $F = f_1 + \cdots + f_N$ with submodular f_i's, where $B = \max_{1 \leq i \leq N} |\mathsf{EX}(\mathcal{B}(f_i))|$. We disprove this claim by giving a counterexample, rendering most size bounds in the paper invalid. For monotone f_i's, however, we show that $\sum_{i=1}^{N} p_i \leq Bn$ indeed holds.

We have to stress here that the core algorithm forms the fundamental building block of our sparsification results presented in the succeeding parts of this work.

3.1 Sparsifying Any Decomposable Function

Consider a domain $\mathcal{D}$, which is the power set $\mathcal{D} = 2^E$ in the case of set functions. Further suppose $F : \mathcal{D} \to \mathbb{R}_{\geq 0}$ is decomposable as $F = \sum_{i=1}^{N} f_i$, where $f_i : \mathcal{D} \to \mathbb{R}_{\geq 0}$ for each $1 \leq i \leq N$. Since $\mathcal{D}$ might be very large, we assume that each f_i is represented by an evaluation oracle that takes time $\mathcal{O}(\mathrm{EO}_i)$ to respond on a single element $A \in \mathcal{D}$.

The algorithm we present here constructs an ε-sparsifier for any decomposable function $F = \sum_{i=1}^{N} f_i : \mathcal{D} \to \mathbb{R}_{\geq 0}$ probabilistically. It crucially relies on sampling functions with probabilities proportional to the ratios

$$p_i = \max_{\substack{A \in \mathcal{D} \\ F(A) \neq 0}} \frac{f_i(A)}{F(A)} \tag{3.1}$$

for each $1 \leq i \leq N$. Computing and in many interesting cases even approximating the p_i's is by far the hardest step on the way to constructing a sparsifier. Therefore, major parts of this work revolve around the p_i's. For better recognition and more intuitive understanding, we refer to them as *peak contributions* – since p_i describes, on a scale from 0 to 1, the maximum contribution of f_i to F when an element $A \in \mathcal{D}$ is chosen in favour of f_i.

As suggested above, the algorithmic idea is to sample each function f_i with a certain probability κ_i that is proportional to p_i. If f_i is sampled, i. e., it is decided that f_i shall be part of the sparsifier, it is included in the sparsifier with weight $1/\kappa_i$, making its expected weight equal to $\mathbb{E}[w_i] = \kappa \cdot 1/\kappa_i = 1$ – its weight in the initial decomposition.

The procedure with all technical details is outlined in Algorithm 1. We will prove that it is correct and establish guarantees on size and running time in the following, before turning to the special case of submodular functions.

Algorithm 1 The Core Sparsification Algorithm

Input: Function $F = f_1 + \cdots + f_N$ with $f_i : \mathcal{D} \to \mathbb{R}_{\geq 0}$ given by evaluation oracles; error tolerance parameters $\varepsilon, \delta \in (0, 1)$

Output: Vector $w \in \mathbb{R}^N$ such that

- $\mathbb{P}[w \text{ is an } \varepsilon\text{-sparsifier}] \geq 1 - \delta$,
- $\mathbb{E}[\mathsf{size}(w)] = \mathcal{O}\left(\frac{\log|\mathcal{D}| + \log\frac{1}{\delta}}{\varepsilon^2} \sum_{i=1}^{N} p_i\right)$ where $p_i = \max_{\substack{A \in \mathcal{D} \\ F(A) \neq 0}} \frac{f_i(A)}{F(A)}$,
- $\mathbb{P}\left[\mathsf{size}(w) \leq \frac{3}{2}\mathbb{E}[\mathsf{size}(w)]\right] \geq 1 - 4\varepsilon^2$.

$w \leftarrow (0, \ldots, 0)$
$\kappa \leftarrow 3 \log\left(\frac{2|\mathcal{D}|}{\delta}\right)/\varepsilon^2$
for $i = 1, \ldots, N$ **do**
  $p_i \leftarrow \max_{\substack{A \in \mathcal{D} \\ F(A) \neq 0}} \frac{f_i(A)}{F(A)}$ ▷ compute peak contribution (here: naively)
  $\kappa_i \leftarrow \min\{1, \kappa p_i\}$ ▷ cap at 1 as κ_i is a probability
  $w_i \leftarrow \begin{cases} 1/\kappa_i & \text{with probability } \kappa_i \\ 0 & \text{with probability } 1 - \kappa_i \end{cases}$ ▷ sample weight of f_i
end for
return w

Theorem 3.1. *Algorithm 1 outputs a vector $w \in \mathbb{R}^N$ such that*

(i) $\mathbb{P}[w \text{ is an } \varepsilon\text{-sparsifier}] \geq 1 - \delta$,

(ii) $\mathbb{E}[\mathsf{size}(w)] = \mathcal{O}\left(\frac{\log|\mathcal{D}| + \log\frac{1}{\delta}}{\varepsilon^2} \sum_{i=1}^{N} p_i\right)$ *where* $p_i = \max_{\substack{A \in \mathcal{D} \\ F(A) \neq 0}} \frac{f_i(A)}{F(A)}$,

(iii) $\mathbb{P}\left[\mathsf{size}(w) \leq \frac{3}{2}\mathbb{E}[\mathsf{size}(w)]\right] \geq 1 - 4\varepsilon^2$.

Proof. We show part (i) first by showing that each $A \in \mathcal{D}$ satisfies

$$\mathbb{P}\left[(1-\varepsilon)F(A) \leq F'(A) \leq (1+\varepsilon)F(A)\right] \geq 1 - \frac{\delta}{|\mathcal{D}|}. \tag{3.2}$$

An application of the union bound over all $|\mathcal{D}|$ elements[1] of the domain then proves the claim. Note that

$$\mathbb{E}[F'(A)] = \mathbb{E}\left[\sum_{i=1}^{N} w_i f_i(A)\right] = \sum_{i=1}^{N} \mathbb{E}[w_i]\, f_i(A) = \sum_{i=1}^{N} f_i(A) = F(A) \tag{3.3}$$

since $\mathbb{E}[w_i] = \kappa_i \cdot 1/\kappa_i + (1-\kappa_i)\cdot 0 = 1$. Therefore, the inequality indicating "success" above is equivalent to $(1-\varepsilon)\mathbb{E}[F'(A)] \leq F'(A) \leq (1+\varepsilon)\mathbb{E}[F'(A)]$, i. e., that $F'(A)$ lies within a factor of $1 \pm \varepsilon$ of its expectation. The next step is to establish

$$\mathbb{P}\left[|F'(A) - \mathbb{E}[F'(A)]| \geq \varepsilon \cdot \mathbb{E}[F'(A)]\right] \leq 2e^{-\varepsilon^2\kappa/3}. \tag{3.4}$$

This is essentially done by recognising that the terms $w_i f_i(A)$ can be regarded as independent random variables with a bounded range $[0, a]$ for some $a \leq F(A)/\kappa$. However, we cannot simply mimic the proof by Rafiey and Yoshida [RY22] here as it includes an invalid step. They claim that $\max_{1\leq i\leq N} w_i f_i(A) = \max_{1\leq i\leq N} \frac{f_i(A)}{\kappa p_i}$, which is not necessarily true when $\kappa_i = 1$ (remember that $\kappa_i = \min\{1, \kappa p_i\}$). In this case, $w_i = 1/\kappa_i = 1$, so $w_i f_i(A) = f_i(A)$, which exceeds $\frac{f_i(A)}{\kappa p_i}$ for $\kappa p_i > 1$. This in fact shows that their inequality goes in the wrong direction.

Fotunately, this issue can be circumvented by considering the indices i with $\kappa p_i > 1$ separately. Let $I := \{i \in \{1,\dots,N\} \mid \kappa p_i > 1\}$ and $\bar{I} := \{1,\dots,N\} \setminus I$. Moreover, define $Z_i := w_i f_i(A)$ for all $1 \leq i \leq N$, $Z := \sum_{i\in I} Z_i$ and $\bar{Z} := \sum_{i\in\bar{I}} Z_i$. We have

$$F'(A) = \sum_{i=1}^{N} Z_i = \sum_{i\in I} Z_i + \sum_{i\in\bar{I}} Z_i = Z + \bar{Z}$$

[1] Over all elements $A \in \mathcal{D}$ such that $F(A) \neq 0$ to be precise. If $F(A) = 0$, we know that $f_i(A) = 0$ for all $1 \leq i \leq N$, so Eq. (3.3) and Eq. (3.4) trivially hold. We do not consider such sets A in our analysis and encourage the reader to think of $A \in \mathcal{D}$ with $F(A) > 0$ for the rest of the analysis.

where the first part $\sum_{i\in I} Z_i = \sum_{i\in I} f_i(A)$ is deterministic as we have $w_i = 1$ with probability $\kappa_i = \min\{1, \kappa p_i\} = 1$ for each $i \in I$. Thus,

$$\begin{aligned}
F'(A) - \mathbb{E}\left[F'(A)\right] &= \sum_{i=1}^{N} w_i f_i(A) - \sum_{i=1}^{N} f_i(A) \\
&= \sum_{i\in I} w_i f_i(A) + \sum_{i\in \bar{I}} w_i f_i(A) - \sum_{i\in I} f_i(A) - \sum_{i\in \bar{I}} f_i(A) \\
&= Z + \sum_{i\in \bar{I}} w_i f_i(A) - Z - \sum_{i\in \bar{I}} f_i(A) \\
&= \sum_{i\in \bar{I}} w_i f_i(A) - \sum_{i\in \bar{I}} f_i(A) \\
&= \sum_{i\in \bar{I}} Z_i - \sum_{i\in \bar{I}} \mathbb{E}\left[Z_i\right] \\
&= \bar{Z} - \mathbb{E}\left[\bar{Z}\right]
\end{aligned}$$

by Eq. (3.3), the partitioning $\{1, \ldots, N\} = I \cup \bar{I}$, definition of the Z_i's, Z and $\bar{Z}$, and linearity of expectation. We conclude that Eq. (3.4) is equivalent to

$$\mathbb{P}\left[\left|\bar{Z} - \mathbb{E}\left[\bar{Z}\right]\right| \geq \varepsilon \cdot \mathbb{E}\left[F'(A)\right]\right] \leq 2e^{-\varepsilon^2 \kappa/3}. \tag{3.5}$$

Since $f_i(A) \geq 0$ for all $1 \leq i \leq N$, observe that

$$\mathbb{E}\left[\bar{Z}\right] = \sum_{i\in \bar{I}} f_i(A) \leq \sum_{i\in \bar{I}} f_i(A) + \sum_{i\in I} f_i(A) = \sum_{i=1}^{N} f_i(A) = F(A) = \mathbb{E}\left[F'(A)\right], \tag{3.6}$$

where the last step is by Eq. (3.3). We now bound $\mathbb{P}\left[\left|\bar{Z} - \mathbb{E}\left[\bar{Z}\right]\right| \geq \varepsilon \cdot \mathbb{E}\left[F'(A)\right]\right]$ by an application of the Chernoff-Hoeffding bound [MR95] in the version of Theorem 2.2 in [RY22]. A check of the preconditions reveals that the Z_i's are independent random variables (since the w_i's are sampled independently). Letting $a := \max_{i\in\bar{I}} Z_i$, the Z_i's with $i \in \bar{I}$ all have range $[0, a]$. Choosing $\mu := \mathbb{E}\left[F'(A)\right] \geq \mathbb{E}\left[\bar{Z}\right]$, we conclude

$$\begin{aligned}
&\mathbb{P}\left[\left|\bar{Z} - \mathbb{E}\left[\bar{Z}\right]\right| \geq \varepsilon\mu\right] \leq 2e^{-\frac{\varepsilon^2\mu}{3a}} \\
\iff &\mathbb{P}\left[\left|\bar{Z} - \mathbb{E}\left[\bar{Z}\right]\right| \geq \varepsilon \cdot \mathbb{E}\left[F'(A)\right]\right] \leq 2e^{-\frac{\varepsilon^2 F(A)}{3a}}.
\end{aligned} \tag{3.7}$$

This is very close to Eq. (3.5). Indeed, the last we need is an upper bound on a. To accomplish this, note that

$$Z_i = w_i f_i(A) = \frac{f_i(A)}{\kappa p_i} \leq \frac{f_i(A)}{\kappa \max_{S\subseteq E} \frac{f_i(S)}{F(S)}} \leq \frac{f_i(A)}{\kappa \frac{f_i(A)}{F(A)}} = \frac{F(A)}{\kappa}$$

for all $i \in \bar{I}$. Thus, maximising over all $i \in \bar{I}$ reveals $a = \max_{i \in \bar{I}} Z_i \leq F(A)/\kappa$. Substituting this into Eq. (3.7) yields Eq. (3.5). The RHS $2e^{-\varepsilon^2 \kappa/3}$ becomes $\leq \frac{\delta}{|\mathcal{D}|}$ when choosing $\kappa = 3 \log\left(\frac{2|\mathcal{D}|}{\delta}\right)/\varepsilon^2$ as in Algorithm 1 above, proving Eq. (3.2).

Part (ii) is a simple computation. The events $\{w_i \neq 0\}$ occur with probability κ_i, so

$$\mathbb{E}\left[\mathsf{size}(w)\right] = \mathbb{E}\left[\sum_{i=1}^{N} [w_i \neq 0]\right] = \sum_{i=1}^{N} \mathbb{P}\left[w_i \neq 0\right] = \sum_{i=1}^{N} \kappa_i \leq \sum_{i=1}^{N} \kappa p_i = \kappa \sum_{i=1}^{N} p_i,$$

where $\kappa = 3 \log\left(\frac{2|\mathcal{D}|}{\delta}\right)/\varepsilon^2 = \mathcal{O}\left(\frac{\log|\mathcal{D}| + \log\frac{1}{\delta}}{\varepsilon^2}\right)$ by the choice in Algorithm 1. Hence the overall bound $\mathbb{E}\left[\mathsf{size}(w)\right] = \mathcal{O}\left(\frac{\log|\mathcal{D}| + \log\frac{1}{\delta}}{\varepsilon^2} \sum_{i=1}^{N} p_i\right)$.

Part (iii) is shown as follows. Let $[w_i \neq 0]$ be an indicator random variable for the event $\{w_i \neq 0\}$, i. e., $[w_i \neq 0] = 1$ if $w_i \neq 0$ and $[w_i \neq 0] = 0$ if $w_i = 0$. The variance of a sum of independent random variables is the sum of their variances. Thus,

$$\mathbb{V}\mathrm{ar}\left[\mathsf{size}(w)\right] = \mathbb{V}\mathrm{ar}\left[\sum_{i=1}^{N} [w_i \neq 0]\right] = \sum_{i=1}^{N} \mathbb{V}\mathrm{ar}\left[[w_i \neq 0]\right]$$

where $\mathbb{V}\mathrm{ar}\left[[w_i \neq 0]\right] = \mathbb{E}\left[[w_i \neq 0]^2\right] - \mathbb{E}\left[[w_i \neq 0]\right]^2 = \kappa_i - \kappa_i^2$ for each $1 \leq i \leq N$. It follows that $\mathbb{V}\mathrm{ar}\left[\mathsf{size}(w)\right] = \sum_{i=1}^{N} (\kappa_i - \kappa_i^2) \leq \sum_{i=1}^{N} \kappa_i = \mathbb{E}\left[\mathsf{size}(w)\right]$.

Thus, by Chebyshev's inequality,

$$\mathbb{P}\left[\left|\mathsf{size}(w) - \mathbb{E}\left[\mathsf{size}(w)\right]\right| \geq \frac{1}{2}\mathbb{E}\left[\mathsf{size}(w)\right]\right] \leq \frac{\mathbb{V}\mathrm{ar}\left[\mathsf{size}(w)\right]}{\left(\frac{1}{2}\mathbb{E}\left[\mathsf{size}(w)\right]\right)^2} \leq \frac{4}{\mathbb{E}\left[\mathsf{size}(w)\right]},$$

which is $\leq 4\varepsilon^2$ because $\mathbb{E}\left[\mathsf{size}(w)\right] \geq 1/\varepsilon^2$: If $\kappa_i = 1$ for some i (i. e. $I \neq \varnothing$), we know that $\mathbb{E}\left[\mathsf{size}(w)\right] = \sum_{i=1}^{N} \kappa_i \geq 1 \geq 1/\varepsilon^2$ (recall $\varepsilon \in (0,1)$). If $\kappa_i < 1$ for all i (i. e. $I = \varnothing$), we have $\mathbb{E}\left[\mathsf{size}(w)\right] = \sum_{i=1}^{N} \kappa_i = \kappa \sum_{i=1}^{N} p_i \overset{(\star)}{\geq} \kappa \geq 1/\varepsilon^2$, where $(\star)$ is justified by the following bound.

Claim: We always have $\sum_{i=1}^{N} p_i \geq 1$.

Fix an arbitrary element $A^* \in \mathcal{D}$ with $F(A) \neq 0$. Then

$$\sum_{i=1}^{N} p_i = \sum_{i=1}^{N} \max_{\substack{A \in \mathcal{D} \\ F(A) \neq 0}} \frac{f_i(A)}{F(A)} \geq \sum_{i=1}^{N} \frac{f_i(A^*)}{F(A^*)} = \frac{1}{F(A^*)} \sum_{i=1}^{N} f_i(A^*) = \frac{1}{F(A^*)} \cdot F(A^*) = 1.$$

Noting $\mathbb{P}\left[\mathsf{size}(w) \geq \frac{3}{2}\mathbb{E}\left[\mathsf{size}(w)\right]\right] \leq \mathbb{P}\left[\left|\mathsf{size}(w) - \mathbb{E}\left[\mathsf{size}(w)\right]\right| \geq \frac{1}{2}\mathbb{E}\left[\mathsf{size}(w)\right]\right] \leq 4\varepsilon^2$ (as the former implies the latter) completes the proof. □

Remark 3.2. Algorithm 1 can be invoked with $\delta = \mathcal{O}(1/n^c)$ so that it yields an ε-sparsifier with high probability. This only influences the running time by a constant factor c because of the dependence on $\log \frac{1}{\delta}$.

Remark 3.3. If the size of the sparsifier is of primary interest, running Algorithm 1 a couple of times and taking the smallest vector w (w. r. t. $\mathsf{size}(w)$) leads to a procedure that, for any fixed $\varepsilon > 0$, returns a sparsifier of size $\mathcal{O}\left(\frac{\log|\mathcal{D}|+\log\frac{1}{\delta}}{\varepsilon^2}\sum_{i=1}^N p_i\right)$ after a logarithmic number of iterations. This is a consequence of part (iii). Notice that it might be necessary to choose δ appropriately to also guarantee that the solution indeed is an ε-sparsifier with high probability.

Corollary 3.4. *In the setting of Algorithm 1, let $\widehat{p}_1, \dots, \widehat{p}_N \in \mathbb{R}_{\geq 0}$ satisfy $\widehat{p}_i \geq p_i$ for all $1 \leq i \leq N$. If Algorithm 1 is executed with the $\widehat{p}_i$'s instead of $p_i = \max_{\substack{A\in\mathcal{D}\\F(A)\neq 0}} \frac{f_i(A)}{F(A)}$ in line 6, it returns a vector $w \in \mathbb{R}^N$ such that*

(i) $\mathbb{P}\left[w \text{ is an } \varepsilon\text{-sparsifier}\right] \geq 1 - \delta$,

(ii) $\mathbb{E}\left[\mathsf{size}(w)\right] = \mathcal{O}\left(\frac{\log|\mathcal{D}|+\log\frac{1}{\delta}}{\varepsilon^2}\sum_{i=1}^N \widehat{p}_i\right)$,

(iii) $\mathbb{P}\left[\mathsf{size}(w) \leq \frac{3}{2}\mathbb{E}\left[\mathsf{size}(w)\right]\right] \geq 1 - 4\varepsilon^2$.

Proof. All steps in the proof of Theorem 3.1 can be mimicked. For part (i), the magic happens when estimating the quantity a, where

$$\frac{f_i(A)}{\kappa \cdot \max_{S\in\mathcal{D}} \frac{f_i(S)}{F(S)}} \leq \frac{f_i(A)}{\kappa \cdot \frac{f_i(A)}{F(A)}} = \frac{F(A)}{\kappa}$$

is established for all $i \in \bar{I}$. Since $\widehat{p}_i \geq p_i$ for each i, we have

$$w_i f_i(A) = \frac{f_i(A)}{\kappa \cdot \widehat{p}_i} \leq \frac{f_i(A)}{\kappa \cdot p_i},$$

for each i with $\kappa\widehat{p}_i \leq 1$, hence the same bound $a \leq F(A)/\kappa$ follows. The indices i with $\kappa\widehat{p}_i < 1$ can be handled without any modifications. Part (ii) is exactly the same. Part (iii) is also largely the same, except for the claim $\sum_{i=1}^N p_i \geq 1$ that needs to be checked for the $\widehat{p}_i$'s. However, this is trivial, as $\sum_{i=1}^N \widehat{p}_i \geq \sum_{i=1}^N p_i \geq 1$. □

At a first glance, this result may seem amazing – it implies that any constant-factor approximation of the p_i's will do the job, leading to the same asymptotic bounds. However, in the general setting of functions $f_1, \dots, f_N : \mathcal{D} \to \mathbb{R}_{\geq 0}$, there is no way of obtaining this much faster than the exact p_i's. Even in the case where all f_i's are submodular, the p_i's are hard to approximate. It is a major part of this dissertation to

investigate classes of submodular functions that do admit an efficient approximation of the peak contributions.

Remark 3.5 (Trivial Execution)**.** In general, the best upper bound we know on the peak contributions is $p_i \leq 1$. Thus, Corollary 3.4 tells us that it is correct to invoke Algorithm 1 with $\widehat{p}_i = 1$ for all $1 \leq i \leq N$. Indeed, in Section 3.2 we will see an example where $\widehat{p}_i = 1$ for every $1 \leq i \leq N$. Since $\kappa > 1$ for $\varepsilon \in (0, 1)$, we then have $\kappa_i = \min\{1, \kappa p_i\} = 1$. This results in the initial decomposition, i. e., Algorithm 1 essentially computes nothing – a sparsifier is not for free!

There are various ways to implement Algorithm 1, leading to different running time bounds. In the most general scenario where no further assumptions are made, two models seem reasonable:

- Individual Oracles: An evaluation oracle for each f_i with response time $\mathcal{O}(\mathsf{EO}_i)$.
- Individual Oracles + Sum Oracle: Each f_i is given by an evaluation oracle with response time $\mathcal{O}(\mathsf{EO}_i)$ and there is an additional evaluation oracle for F with response time $\mathcal{O}(\mathsf{EO}_\Sigma)$.

Observe how the second model captures the first: If we have no additional information, we can assume $\mathsf{EO}_\Sigma = \sum_{i=1}^{N} \mathcal{O}(\mathsf{EO}_i)$ since $F(A)$ can be computed as $\sum_{i=1}^{N} f_i(A)$ after obtaining $f_i(A)$ for each $1 \leq i \leq N$ from the respective oracle.

The main work to be done for Algorithm 1 to successfully construct a (small) sparsifier is the computation or approximation of the peak contributions. A large part of this dissertation is devoted to this problem for various special cases and under different assumptions. In the most general setting investigated in this chapter, we are required to compute p_i by iterating through all $A \in \mathcal{D}$, which takes time at least $\Omega(|\mathcal{D}|)$.

The running time of a naive implementation is in $\mathcal{O}\left(N|\mathcal{D}| \sum_{i=1}^{N} \mathsf{EO}_i\right)$.

3.2 The Submodular Case

We introduced the core algorithm as a general tool for the sparsification of decomposable functions, extending far beyond submodular functions. However, if the functions involved are submodular, Rafiey and Yoshida [RY22] prove a more precise size bound by bounding the sum $\sum_{i=1}^{N} p_i$ of the peak contributions. Unfortunately, this proof seems wrong and we will provide a counterexample. The good news is that it turns out to be correct if monotonicity is given – an important special case that is shown

computationally tractable by an $\mathcal{O}(\sqrt{n})$-approximation of the peak contributions using the ellipsoid method [RY22; Bai+16]. Together with the bound $\sum_{i=1}^{N} p_i \leq Bn$ (where B is a quantity described later that is independent of N) we establish here, Theorem 3.1 part (ii) simplifies to $\mathbb{E}[\mathsf{size}(w)] = \mathcal{O}\left(B\frac{n^2+n\log\frac{1}{\delta}}{\varepsilon^2}\right)$ if all f_i's are monotone, which is $\mathcal{O}\left(\frac{Bn^2}{\varepsilon^2}\right)$ for fixed δ. This section takes a deeper look into the structure of submodular functions, mainly involving extreme points of base polyhedra.

We begin by citing a result that characterises the extreme points of the base polyhedron of a submodular function.

Theorem 3.6 (Extreme Point Theorem [Fuj91])**.** *Let $f : 2^E \to \mathbb{R}$ be submodular. A point $y \in \mathcal{B}(f)$ is an extreme point of $\mathcal{B}(f)$ iff for a maximal chain (w. r. t. inclusion)*

$$\mathcal{C} : \varnothing = S_0 \subset S_1 \subset \cdots \subset S_n = E$$

of 2^E we have $y_{e_i} = f(S_i) - f(S_{i-1})$ for $i = 1, \ldots, n$, where $\{e_i\} = S_i \setminus S_{i-1}$.

For convenience, we introduce the notation $\mathsf{EX}(P)$ to denote the set of extreme points of a polyhedron P. In particular, $\mathsf{EX}(\mathcal{B}(f))$ is the set of extreme points of the base polyhedron of a submodular function f.

Corollary 3.7. *Let $f : 2^E \to \mathbb{R}$ be submodular and monotone. Then $\mathsf{EX}(\mathcal{B}(f)) \subseteq \mathbb{R}^E_{\geq 0}$, i. e., any extreme point of the base polyhedron has only non-negative coordinates.*

Proof. Suppose $y \in \mathsf{EX}(\mathcal{B}(f))$ is an extreme point of the base polyhedron. By Theorem 3.6, there is a maximal chain $\varnothing = S_0 \subset S_1 \subset \cdots \subset S_n = E$ such that $y_{e_i} = f(S_i) - f(S_{i-1})$ for $1 \leq i \leq n$, where $\{e_i\} = S_i \setminus S_{i-1}$. By monotonicity,

$$y_{e_i} = f(S_i) - f(S_{i-1}) = f(S_{i-1} \cup \{e_i\}) - f(S_{i-1}) \geq f(S_{i-1}) - f(S_{i-1}) = 0$$

for each $1 \leq i \leq n$. Since the chain is maximal, each element $e \in E$ occurs as some e_i, implying $y_e \geq 0$ for all $e \in E$. □

The next statement expresses the values $f(A)$ of a submodular function in terms of a maximisation over the base polyhedron.

Lemma 3.8. *Let $f : 2^E \to \mathbb{R}$ be normalised submodular. Then*

$$f(A) = \max_{y \in \mathsf{EX}(\mathcal{B}(f))} \langle y, \mathbb{1}_A \rangle$$

for all $A \subseteq E$.

Proof. The inequality $\max_{y \in \mathsf{EX}(\mathcal{B}(f))} \langle y, \mathbb{1}_A \rangle \leq f(A)$ is immediate since any $y \in \mathcal{B}(f)$ satisfies $y(S) \leq f(S)$ for all $S \subseteq E$, including $S = A$. Hence

$$\langle y, \mathbb{1}_A \rangle = \sum_{e \in E} y_e \cdot \mathbb{1}_A(e) = \sum_{e \in A} y_e = y(A) \leq f(A).$$

To see that the value $f(A)$ is actually attained, we construct a suitable extreme point using Theorem 3.6. Suppose $A = \{e_1, \ldots, e_k\}$. Order the remaining elements arbitrarily, so $E \setminus A = \{e_{k+1}, \ldots, e_n\}$. Now define a maximal chain

$$\mathcal{C} : \varnothing = S_0 \subset S_1 \subset \cdots \subset S_n = E$$

by letting $S_i := \{e_1, \ldots, e_i\}$ for $0 \leq i \leq n$. Next, we set

$$y_{e_i} := f(S_i) - f(S_{i-1})$$

for $1 \leq i \leq n$, which defines a vector $y \in \mathbb{R}^E$ that – by construction – happens to be an extreme point of $\mathcal{B}(f)$ by Theorem 3.6. Moreover, it satisfies

$$y(A) = \sum_{e \in A} y_e = \sum_{i=1}^{k} y_{e_i} = \sum_{i=1}^{k} \left(f(S_i) - f(S_{i-1}) \right) = f(S_k) - f(\varnothing) = f(S_k) = f(A)$$

as f is normalised. Thus, y witnesses that $\max_{y \in \mathsf{EX}(\mathcal{B}(f))} \langle y, \mathbb{1}_A \rangle \geq f(A)$. □

Now we have all tools required to establish an upper bound on $\sum_{i=1}^{N} p_i$, making the size bound in Theorem 3.1 part (ii) simpler and more expressive. However, we first disprove the bound $\sum_{i=1}^{N} p_i \leq Bn$ claimed by Rafiey and Yoshida (c. f. Claim 3.3 in [RY22], proof in the appendix) by counterexample. Afterwards, we show that the bound holds if all constituent functions are monotone.

The counterexample is inspired by Cohen et al. [Coh+17], who use it to establish a lower bound on the size of a cut sparsifier. This is exactly what we need to disprove $\sum_{i=1}^{N} p_i \leq Bn$: The ability to isolate edges in order to have peak contributions of 1. Consider the directed complete bipartite graph $G = (V, E)$ with bipartition $V = L \cup R$ where $L = \{u_1, \ldots, u_5\}$, $R = \{v_1, \ldots, v_5\}$ and $E = L \times R$ as depicted in Fig. 3.1. For each edge $e = (u, v) \in E$, we define a cut function

$$f_e : 2^V \to \mathbb{R}, \quad S \mapsto \begin{cases} 1 & \text{if } u \in S \text{ and } v \notin S, \\ 0 & \text{otherwise.} \end{cases}$$

It is well-known (and not hard to see) that the f_e's are submodular. In fact, we will later prove this in the more general setting of r-uniform hypergraphs, see Fact 4.16.

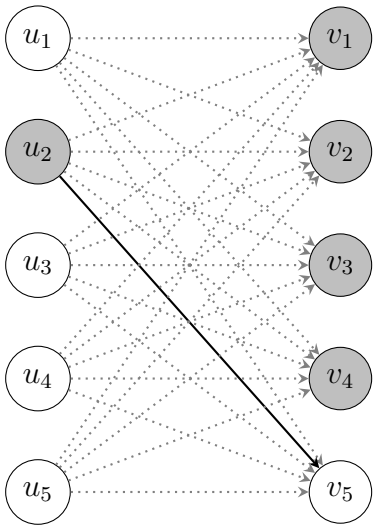


Figure 3.1: Graph G where each edge can be isolated by a cut

The sum $F := \sum_{e \in E} f_e$ is known as the (directed) cut function of G, since $F(S)$ is the number of edges cut connecting a vertex in S to a vertex outside S. Let $p_e = \max_{S \subseteq V} \frac{f_e(S)}{F(S)}$ denote the peak contributions of the f_e's.

We will now establish how this example violates $\sum_{e \in E} p_e \leq Bn$. First,

$$B = \max_{e \in E} |\mathsf{EX}(\mathcal{B}(f_e))| = 2$$

as each base polyhedron $\mathcal{B}(f_i)$ has two extreme points. An easy way to see this goes by Theorem 3.6. For any maximal chain $\varnothing = V_0 \subset V_1 \subset \cdots \subset V_n = V$, we have

$$f_e(V_{i+1}) - f_e(V_i) = \begin{cases} 1 & \text{if } V_{i+1} \setminus V_i = \{u\} \text{ and } v \notin V_i, \\ -1 & \text{if } V_{i+1} \setminus V_i = \{v\} \text{ and } u \in V_i, \\ 0 & \text{otherwise.} \end{cases}$$

Thus, the only extreme points are y and y', where $y_e = 0$ for all $e \in E$, and $y'_u = 1$, $y'_v = -1$, $y'_e = 0$ for all $e \in E \setminus \{u, v\}$. Since G has 10 vertices and the vertex set $V = L \cup R$ is the ground set, Claim 3.3 in [RY22] asserts that $\sum_{e \in E} p_e \leq Bn = 20$. By showing $p_e = 1$ for each $e \in E$, we establish

$$\sum_{e \in E} p_e = |E| = |L| \cdot |R| = 5 \cdot 5 = 25 > 20,$$

contradicting the assertion. Fix an edge $e = \{u, v\}$. Letting $S := \{u\} \cup (R \setminus \{v\})$, we get $f_e(S) = 1$ and $f_{e'}(S) = 0$ for each edge $e' \neq e$. The latter is because S either

- contains the right endpoint of e' or

- does not contain the left endpoint of e',

both cases leading to $f_{e'}(S) = 0$ by definition of $f_{e'}$. Thus,

$$p_e = \max_{A \subseteq V} \frac{f_e(A)}{F(A)} \geq \frac{f_e(S)}{F(S)} = \frac{f_e(S)}{f_e(S) + \sum_{e' \neq e} f_{e'}(S)} = \frac{1}{1+0} = 1,$$

implying $p_e = 1$, as desired.

The following statement appears as Claim 3.3 in [RY22] with a very similar proof. However, this proof only works if the coordinates of the extreme points of the base polyhedra are non-negative. We give greater detail and explanations of the individual steps and point out where exactly the non-negativity is needed, hence where the proof in [RY22] fails. Corollary 3.7 ensures non-negativity when all f_i's are monotone. For the following lemma, we specialise to $\mathcal{D} = 2^E$ for some ground set E of size n.

Lemma 3.9. *Let normalised, monotone submodular functions $f_1, \dots, f_N : 2^E \to \mathbb{R}$ be given with peak contributions defined w. r. t. $F = f_1 + \cdots + f_N$. Then $\sum_{i=1}^N p_i \leq Bn$, where $B = \max_{1 \leq i \leq N} |\mathsf{EX}(\mathcal{B}(f_i))|$ denotes the maximum number of extreme points of the base polyhedra of the f_i's.*

Proof. The key ingredient is Lemma 3.8. It gives us

$$\sum_{i=1}^N p_i = \sum_{i=1}^N \max_{A \subseteq E} \frac{f_i(A)}{F(A)} = \sum_{i=1}^N \max_{A \subseteq E} \frac{\max_{y \in \mathsf{EX}(\mathcal{B}(f_i))} \langle y, \mathbb{1}_A \rangle}{\sum_{j=1}^N \max_{y \in \mathsf{EX}(\mathcal{B}(f_j))} \langle y, \mathbb{1}_A \rangle} \tag{3.8}$$

using the definition of the p_i's. Since all f_i's are monotone, we know that $y \in \mathbb{R}^E_{\geq 0}$ for all $y \in \mathcal{B}(f_i)$ for every $1 \leq i \leq N$, so all terms $\langle y, \mathbb{1}_A \rangle$ involved in the above equation are non-negative. For non-negative numbers $x_1, \dots, x_n \in \mathbb{R}_{\geq 0}$, we always have

$$\frac{x_1 + \cdots + x_n}{n} \overset{(1)}{\leq} \max_{1 \leq i \leq n} x_i \overset{(2)}{\leq} x_1 + \cdots + x_n, \tag{3.9}$$

i. e., the maximum lies between the average and the sum. This is a trick used in the proof by Rafiey and Yoshida [RY22] that does not hold for general submodular functions. Non-negativity plays an important role in a couple of steps in the remainder of this proof. Applying (1) to the denominator and (2) to the numerator in Eq. (3.8), we obtain

$$\frac{\max_{y \in \mathsf{EX}(\mathcal{B}(f_i))} \langle y, \mathbb{1}_A \rangle}{\sum_{j=1}^N \max_{y \in \mathsf{EX}(\mathcal{B}(f_j))} \langle y, \mathbb{1}_A \rangle} \leq \frac{\sum_{y \in \mathsf{EX}(\mathcal{B}(f_i))} \langle y, \mathbb{1}_A \rangle}{\sum_{j=1}^N \frac{1}{|\mathsf{EX}(\mathcal{B}(f_j))|} \sum_{y \in \mathsf{EX}(\mathcal{B}(f_j))} \langle y, \mathbb{1}_A \rangle} \tag{3.10}$$

$$= \frac{\sum_{e \in A} \sum_{y \in \mathsf{EX}(\mathcal{B}(f_i))} y_e}{\sum_{e \in A} \sum_{j=1}^N \frac{1}{|\mathsf{EX}(\mathcal{B}(f_j))|} \sum_{y \in \mathsf{EX}(\mathcal{B}(f_j))} y_e}, \tag{3.11}$$

where the equality step is by $\langle y, \mathbb{1}_A \rangle = \sum_{e \in A} y_e$. Next, note that $\sum_{y \in \mathsf{EX}(\mathcal{B}(f_i))} y_e \geq 0$ and $\sum_{j=1}^{N} \frac{1}{|\mathsf{EX}(\mathcal{B}(f_j))|} \sum_{y \in \mathsf{EX}(\mathcal{B}(f_j))} y_e \geq 0$ for all $e \in A$ because $y_e \geq 0$ for all $y \in \mathcal{B}(f_i)$ and $e \in E$, so we can employ the inequality

$$\frac{x_1 + \cdots + x_n}{q_1 + \cdots + q_n} \leq \max_{1 \leq i \leq n} \frac{x_i}{q_i} \tag{3.12}$$

that applies to all non-negative numbers $x_1, \ldots, x_n, q_1, \ldots, q_n \in \mathbb{R}_{\geq 0}$, giving

$$\frac{\sum_{e \in A} \sum_{y \in \mathsf{EX}(\mathcal{B}(f_i))} y_e}{\sum_{e \in A} \sum_{j=1}^{N} \frac{1}{|\mathsf{EX}(\mathcal{B}(f_j))|} \sum_{y \in \mathsf{EX}(\mathcal{B}(f_j))} y_e} \leq \max_{e \in A} \frac{\sum_{y \in \mathsf{EX}(\mathcal{B}(f_i))} y_e}{\sum_{j=1}^{N} \frac{1}{|\mathsf{EX}(\mathcal{B}(f_j))|} \sum_{y \in \mathsf{EX}(\mathcal{B}(f_j))} y_e}. \tag{3.13}$$

Combining this with Eq. (3.10) and Eq. (3.11), we can extend Eq. (3.8) to

$$\sum_{i=1}^{N} p_i \leq \sum_{i=1}^{N} \max_{A \subseteq E} \max_{e \in A} \frac{\sum_{y \in \mathsf{EX}(\mathcal{B}(f_i))} y_e}{\sum_{j=1}^{N} \frac{1}{|\mathsf{EX}(\mathcal{B}(f_j))|} \sum_{y \in \mathsf{EX}(\mathcal{B}(f_j))} y_e} \tag{3.14}$$

$$= \sum_{i=1}^{N} \max_{e \in E} \frac{\sum_{y \in \mathsf{EX}(\mathcal{B}(f_i))} y_e}{\sum_{j=1}^{N} \frac{1}{|\mathsf{EX}(\mathcal{B}(f_j))|} \sum_{y \in \mathsf{EX}(\mathcal{B}(f_j))} y_e} \tag{3.15}$$

$$\leq \sum_{i=1}^{N} \sum_{e \in E} \frac{\sum_{y \in \mathsf{EX}(\mathcal{B}(f_i))} y_e}{\sum_{j=1}^{N} \frac{1}{|\mathsf{EX}(\mathcal{B}(f_j))|} \sum_{y \in \mathsf{EX}(\mathcal{B}(f_j))} y_e}, \tag{3.16}$$

where Eq. (3.15) follows because maximising over all $A \subseteq E$ and $e \in A$ is the same as maximising over all $e \in E$ here, since the RHS only depends on e. Eq. (3.16) follows from Eq. (3.9) again – with all quotients being non-negative because the numerators and denominators are. A few algebraic steps get us to the claim now. Swapping $\sum_{i=1}^{N} \ldots$ with $\sum_{e \in E} \ldots$ and using $|\mathsf{EX}(\mathcal{B}(f_j))| \leq \max_{1 \leq \ell \leq n} |\mathsf{EX}(\mathcal{B}(f_\ell))| = B$, we get

$$\sum_{i=1}^{N} \sum_{e \in E} \frac{\sum_{y \in \mathsf{EX}(\mathcal{B}(f_i))} y_e}{\sum_{j=1}^{N} \frac{1}{|\mathsf{EX}(\mathcal{B}(f_j))|} \sum_{y \in \mathsf{EX}(\mathcal{B}(f_j))} y_e} \tag{3.17}$$

$$= \sum_{e \in E} \sum_{i=1}^{N} \frac{\sum_{y \in \mathsf{EX}(\mathcal{B}(f_i))} y_e}{\sum_{j=1}^{N} \frac{1}{|\mathsf{EX}(\mathcal{B}(f_j))|} \sum_{y \in \mathsf{EX}(\mathcal{B}(f_j))} y_e} \tag{3.18}$$

$$\leq \sum_{e \in E} \sum_{i=1}^{N} \frac{\sum_{y \in \mathsf{EX}(\mathcal{B}(f_i))} y_e}{\frac{1}{B} \sum_{j=1}^{N} \sum_{y \in \mathsf{EX}(\mathcal{B}(f_j))} y_e} \tag{3.19}$$

$$= \sum_{e \in E} B = Bn, \tag{3.20}$$

leading to the claimed $\sum_{i=1}^{N} p_i \leq Bn$ via Eq. (3.14), Eq. (3.15) and Eq. (3.16). □

What makes this lemma interesting is that the bound $\sum_{i=1}^{N} p_i \leq Bn$ does not depend on N. In cases where sparsification is interesting, we usually have $N \gg n$.

Remark 3.10. If all f_i's are monotone, we can at least say the following, which Rafiey and Yoshida [RY22] already observed. There is an algorithm based on the ellipsoid method that, given two monotone submodular functions $f, g : 2^E \to \mathbb{R}_{\geq 0}$, approximates $\max_{A \subseteq E} \frac{f(A)}{g(A)}$ up to a factor of $\mathcal{O}(\sqrt{n} \log n)$ in polynomial time (see Theorem 3.4 in [RY22] and [Bai+16]). It can be used to obtain approximations $\widehat{p}_1, \ldots, \widehat{p}_N$ such that $p_i \leq \widehat{p}_i \leq \mathcal{O}(\sqrt{n} \log n) p_i$ for $1 \leq i \leq N$. By Corollary 3.4, we can execute Algorithm 1 with the $\widehat{p}_i$'s and obtain an ε-sparsifier of expected size $\mathcal{O}\left(\frac{n+\log \frac{1}{\delta}}{\varepsilon^2} \sum_{i=1}^{N} \widehat{p}_i\right)$ in polynomial time. Since the f_i's are monotone, Lemma 3.9 applies and the bound becomes $\mathcal{O}\left(\frac{Bn^{2.5} \log n}{\varepsilon^2}\right)$ for fixed δ.

Chapter 4

Improved Sparsifier Constructions

This chapter treats efficient constructions of small sparsifiers in three scenarios. Section 4.1 concerns decomposable submodular functions where the constituents are of low curvature. In Section 4.2, we investigate a similar setting where the constituents are of bounded arity. Lastly, we introduce a relaxed notion of sparsification in Section 4.3 and show that such relaxed sparsifiers can be efficiently constructed for any submodular function. All results are obtained by finding an efficient way to compute or approximate the peak contributions, which are then fed into the core algorithm for a sparsifier.

Contributions:

- Sparsifier of expected size $\mathcal{O}\left(\frac{n+\log\frac{1}{\delta}}{\left(1-e^{c_F{}^{-1}}\right)\varepsilon^2}\sum_{i=1}^{N} p_i\right)$ in polynomial time if all f_i's have low curvature, which becomes $\mathcal{O}\left(\frac{Bn^2}{\left(1-e^{c_F{}^{-1}}\right)\varepsilon^2}\right)$ for fixed δ if all f_i's are monotone. In this case, a sparsifier of even smaller size can be constructed in the knapsack-constrained setting where $F(S) \leq B$ for some budget $B \geq 0$.
- As part of the knapsack-constraint sparsification, we show that Algorithm 2 in [Per+21] is flawed and disprove the analysis by providing a counterexample. We propose a way to bypass this issue for most instances of knapsack-constrained ratio maximisation, making the aforementioned sparsifier construction in the knapsack-constrained setting possible for most instances.
- Sparsifier of expected size $\mathcal{O}\left(\frac{n+\log\frac{1}{\delta}}{\varepsilon^2}\sum_{i=1}^{N} p_i\right)$ in polynomial time if all f_i's have bounded arity and their effective supports are known. The bound becomes $\mathcal{O}\left(\frac{Bn^2}{\varepsilon^2}\right)$ for fixed δ if all f_i's are monotone, matching a lower bound known for submodular functions, see Remark 1.3 in [RY22] and Bai et al. [Bai+16].
- Sparsifier of expected size $\mathcal{O}\left(\frac{n+\log\frac{1}{\delta}}{\varepsilon^2}\sum_{i=1}^{N} p_i\right)$ under a relaxed notion of sparsification that in a way converges to the notion we have used so far as $n \to \infty$.

4.1 Low Curvature

For a monotone, non-negative submodular function $f : 2^E \to \mathbb{R}_{\geq 0}$, we define its *curvature* (sometimes called *total curvature* [Von10]) as

$$\begin{aligned} c_f &:= 1 - \min_{S \subseteq E} \min_{e \in E \setminus S} \frac{\Delta_e(f \mid S)}{\Delta_e(f \mid \varnothing)} \\ &= 1 - \min_{S \subseteq E} \min_{e \in E \setminus S} \frac{f(S \cup \{e\}) - f(S)}{f(\{e\}) - f(\varnothing)}. \end{aligned}$$

Notice how the quotient compares the marginal gains of adding an element e to the sets S and $\varnothing$, respectively. The curvature always satisfies $0 \leq c_f \leq 1$, with the upper bound being justified by the diminishing return property $\Delta_e(f \mid S) \leq \Delta_e(f \mid \varnothing)$. Also note that $c_f = 0$ precisely in the case where f is modular – c_f can be seen as a measure of how close f is to a modular function.

Remark 4.1. One may raise concerns about division by zero if $f(\{e\}) = 0$. Although this is indeed possible, such elements behave trivially: Submodularity of f implies $f(S \cup \{e\}) - f(S) \leq f(\{e\}) - f(\varnothing) = 0$, hence $f(S \cup \{e\}) \leq f(S)$ for any set S. By monotonocity, it also holds $f(S \cup \{e\}) \geq f(S)$, hence $f(S \cup \{e\}) = f(S)$. Thus, whenever we add element e, the value of f is not influenced. In other words, we might restrict ourselves to the subset $E_{>0} = \{e \in E \mid f(\{e\}) > 0\} \subseteq E$.

The curvature c_f can take values in the interval $[0, 1]$. We say that f has *low curvature* if $c_f < 1$. This is sometimes referred to as *bounded curvature* because the quotients of marginal gains are bounded away from 1. It turns out that low curvature is a powerful property that allows us to efficiently approximate the peak contributions up to a constant factor, leading to an efficient execution of the core algorithm.

4.1.1 Approximate Ψ-Maximisation

Given non-negative, monotone submodular functions f and g, Perrault et al. [Per+21] show that $\max_{S \subseteq E} f(S)/g(S)$ can be approximated by a simple greedy algorithm within a constant factor if $c_g < 1$. In fact, this greedy algorithm works not only for the quotient f/g but for any quasiconvex operation $\Psi(\cdot, \cdot)$ that is non-decreasing in the first argument.

We note that a function $f : \mathbb{R}^2 \to \mathbb{R}$ is *quasiconvex* if

$$f(\alpha x + (1-\alpha)x', \alpha y + (1-\alpha)y') \leq \max\{f(x, y), f(x', y')\}$$

for all $x, y, x', y' \in \mathbb{R}$ and $\alpha \in [0,1]$. Since convexity requires the same inequality with $\alpha \cdot f(x,y) + (1-\alpha) \cdot f(x',y')$ (a weighted average!) on the RHS, we see that quasiconvexity is a weaker notion (as the weighted average is at most the maximum).

Why is it useful to be able to approximate $\max_{S \subseteq E} \Psi(f(S), g(S))$ instead of just the quotient? Apart from challenge of seeking the most general result, quasiconvex 2-variables functions that are non-decreasing in the first argument cover interesting choices such as the difference function $\Psi = f - g$ or $\Psi = f - \sqrt{g}$.

We refer to the problem of maximising $\Psi(f(S), g(S))$ over $S \subseteq E$ for a quasiconvex function $\Psi : \mathbb{R}^2 \to \mathbb{R}$ that is non-decreasing in its first argument as Ψ-*maximisation*.

It is instructive to sketch the greedy algorithm by Perrault et al. [Per+21] here. The idea is to start with the empty set $S_0 = \varnothing$ and successively build a (maximal) chain $\varnothing = S_0 \subset S_1 \subset \cdots \subset S_n = E$ by adding the elements of E in an order chosen by a greedy policy. At any step k, this policy chooses an element $e_k \in E \setminus S_{k-1}$ not chosen before such that the quotient of marginal gains on f and g w. r. t. S_{k-1} is maximal. After building the chain, the best set S_k w. r. t. the function $S \mapsto \Psi(f(S), g(S))$ is selected as output. See Algorithm 2 for a formal description of this algorithm.

Algorithm 2 Approximate Ψ-maximisation in the submodular case

Input: Monotone submodular functions $f, g : 2^E \to \mathbb{R}_{\geq 0}$; quasiconvex 2-variables function Ψ that is non-decreasing in the first argument

Output: $S \subseteq E$ with $\Psi((1 - e^{c_g - 1}) f(S^*), g(S^*)) \leq \Psi(f(S), g(S))$ for any $S^* \subseteq E$

1: $S_0 \leftarrow \varnothing$
2: **for** $k = 1$ **to** n **do**
3: $\quad e_k \leftarrow \arg\max_{e \in E \setminus S_{k-1}} \frac{\Delta_e(f|S_{k-1})}{\Delta_e(g|S_{k-1})}$
4: $\quad S_k \leftarrow S_{k-1} \cup \{e_k\}$
5: **end for**
6: $S \leftarrow \arg\max_{0 \leq k \leq n} \Psi(f(S_k), g(S_k))$
7: **return** S

The following guarantee is known.

Theorem 4.2 ([Per+21])**.** *Let $f, g : 2^E \to \mathbb{R}_{\geq 0}$ be non-negative, monotone submodular functions. Let Ψ be a quasiconvex 2-variables function that is non-decreasing in the first argument. Then, for any $S^* \subseteq E$, the output S of Algorithm 2 satisfies*

$$\Psi(\left(1 - e^{c_g - 1}\right) f(S^*), g(S^*)) \leq \Psi(f(S), g(S)).$$

We note that $0 \leq 1 - e^{c_g - 1} \leq 1 - 1/e$, where $1 - e^{c_g - 1} \to 0$ as $c_g \to 1$, i. e., the approximation factor tends to 0 as g approaches the end of the low curvature regime. On the other side, the best case is a $(1 - 1/e)$-approximation if g is modular.

Two prominent choices for such a function Ψ are the quotient $\Psi(f, g) = f/g$ and the difference $\Psi(f, g) = f - g$, both defined pointwise. We are particularly interested in the quotient, for which Theorem 4.2 guarantees

$$\left(1 - e^{c_g - 1}\right) \frac{f(S^*)}{g(S^*)} = \frac{\left(1 - e^{c_g - 1}\right) f(S^*)}{g(S^*)} \leq \frac{f(S)}{g(S)},$$

hence S gives an $(1 - e^{c_g - 1})$-approximation of the maximum quotient (by choosing $S^* = \arg\max_{A \subseteq E} f(A)/g(A)$). Before we apply Algorithm 2 to construct sparsifiers, we turn our focus onto the second algorithm Perrault et al. [Per+21] describe in their paper. We reveal a flaw and suggest a new algorithm that fixes the issue for most instances. We apply it in Section 4.1.3 to sparsification under knapsack constraints.

4.1.2 Knapsack-Constrained Ψ-Maximisation

The algorithm and its analysis by Perrault et al. (Algorithm 2 in [Per+21]) are flawed. This section consists of two parts:

(A) We present a counterexample revealing an invalid step in the analysis and giving evidence that the algorithm might fail to achieve the claimed approximation guarantee. Perrault confirmed[1] that their algorithm is indeed flawed.

(B) PTAS-style $(1 - \varepsilon)\,(1 - e^{c_g - 1})$-approximation running in polynomial time for fixed ε and c_g for knapsack-constrained ratio maximisation that achieves the approximation guarantee under an additional assumption.

Remark 4.3. Part (B) can be considered a partial fix of the flawed algorithm by Perrault et al. [Per+21]. Personal communications[2] make it seem challenging to fix the approach. Our algorithm in part (B) stands in the light of these difficulties and is the only algorithm for ratio maximisation under knapsack constraints that we know.

(A) Let $E = \{e_1, \ldots, e_5\}$ be a ground set of five elements. We define corresponding "base values" $e_1^f, \ldots, e_5^f$ and $e_1^g, \ldots, e_5^g$ as in Table 4.1. The base values govern the

[1] Personal communications on August 25, 2022 and August 28, 2022

[2] Pierre Perrault on August 25, 2022 and August 28, 2022; Julien Codsi on August 29, 2022

definitions of submodular functions $f, g : 2^E \to \mathbb{R}_{\geq 0}$ as follows:

$$\begin{aligned} f(S) &:= \sum_{e \in S} f(e^f) \\ g(S) &:= -\frac{|S|^2}{30} + \sum_{e \in S} g(e^g) \end{aligned} \tag{4.1}$$

Next, we demonstrate that f and g fulfill all assumptions made by Perrault et al. in their paper [Per+21].

	e_1	e_2	e_3	e_4	e_5
e_i^f	2	2	2	2	10
e_i^g	1	1	1	1	41

Table 4.1: Definition of the counterexample's base values

Fact 4.4. *The functions f and g defined in Eq. (4.1) are both submodular and monotone. Moreover, the curvature of g is $c_g = 3/10 < 1$.*

Proof. The submodularity of f is immediate as it is a sum of weights e_i^f of elements e_i in S, i. e., f is modular. For g, we check the diminishing returns property. Let $T \supseteq S$ and $e \notin T$. Then

$$\begin{aligned} \Delta_e(g \mid T) &= g(T \cup \{e\}) - g(T) \\ &= \left(-\frac{|T \cup \{e\}|^2}{30} + \sum_{h \in T} g(h^g) + g(e^g) \right) - \left(-\frac{|T|^2}{30} + \sum_{h \in T} g(h^g) \right) \\ &= \frac{|T|^2}{30} - \frac{(|T|+1)^2}{30} + g(e^g) \end{aligned}$$

and $\Delta_e(g \mid S) \leq \frac{|S|^2}{30} - \frac{(|S|+1)^2}{30} + g(e^g)$ by the same steps. By $S \subseteq T$, we have $|S| \leq |T|$, hence $\frac{|T|^2}{30} - \frac{(|T|+1)^2}{30} \leq \frac{|S|^2}{30} - \frac{(|S|+1)^2}{30}$, implying $\Delta_e(g \mid T) \leq \Delta_e(g \mid S)$.

For monotonicity, we see that $e_i^f \geq 0$ for $i = 1, \ldots, 5$, hence f is monotone as the sum of e_i^f's. For g, we have already derived an expression for $\Delta_e(g \mid T)$ with $T \subseteq E$ and $e \notin T$ above. Using it, the fact that $e_i^g \geq 1$ for $i = 1, \ldots, 5$ and $|T| \leq 4$, we conclude

$$\Delta_e(g \mid T) = \frac{|T|^2}{30} - \frac{(|T|+1)^2}{30} + g(e^g) \geq \frac{|T|^2}{30} - \frac{(|T|+1)^2}{30} + 1 \geq \frac{4^2}{30} - \frac{5^2}{30} + 1 \geq 0,$$

so g is monotone. Lastly, the curvature of g is by definition

$$c_g = 1 - \min_{T \subseteq E} \min_{e \in E \setminus T} \frac{\Delta_e(g \mid T)}{\Delta_e(g \mid \varnothing)} = 1 - \min_{T \subseteq E} \min_{e \in E \setminus T} \frac{g(e^g) - \frac{2|T|+1}{30}}{g(e^g) - \frac{1}{30}}$$

after simplifying, in particular $|T \cup \{e\}|^2 - |T|^2 = (|T|+1)^2 - |T|^2 = 2|T| + 1$. One may now work out c_g by substituting in all $2^5 = 32$ subsets T and all candidates $e \notin T$, amounting to more ~ 100 expressions to evaluate. However, the bound

$$\frac{g(e^g) - \frac{2|T|+1}{30}}{g(e^g) - \frac{1}{30}} \geq \frac{g(e^g) - \frac{9}{30}}{g(e^g) - \frac{1}{30}} \geq \frac{1 - \frac{9}{30}}{1 - \frac{1}{30}} = \frac{7}{10}$$

immediately reveals that $c_g \leq 1 - 7/10 = 3/10$. Moreover, in the above inequality chain, equality is attained for $T = \{e_2, e_3, e_4, e_5\}$ and $e = e_1$, hence $c_g = 3/10$. □

Let's work out the sets $S_2, S_3, \ldots$ the algorithm by Perrault et al. constructs for the knapsack-constrained ratio maximisation instance given by f, g, $\Psi = f/g$ and the budget $B = 100$. It first computes

$$S_2 = \arg \max_{|S| \leq 3, g(S) \leq B} \Psi\left(f(S), g(S)\right),$$

i. e. an optimal solution with at most three elements. It is not hard to see that exactly the 3-element subsets of $\{e_1, e_2, e_3, e_4\}$ are optimal, all giving an objective value of

$$\frac{f(\{e_1, e_2, e_3\})}{g(\{e_1, e_2, e_3\})} = \frac{2+2+2}{1+1+1-\frac{3^2}{30}} = \frac{6}{3 - \frac{9}{30}} = \frac{20}{9}.$$

Next, independent of S_2, the algorithm computes S_3 as the greedy-maximisation of f under $|S| \leq 3$. This is easily seen to be e_5 plus two other arbitrary elements, say e_1 and e_2, giving $f(\{e_1, e_2, e_5\}) = 2+2+10 = 14$ and $g(\{e_1, e_2, e_5\}) = 1+1+41-3^2/30 > 42$. Finally, the algorithm enters its **for**-loop for $k \geq 4$ and sets $S_4 = \{e_1, e_2, e_3, e_5\}$ as well as $S_5 = \{e_1, e_2, e_3, e_4, e_5\}$.

Now turn the focus onto the analysis, see Appendix D in [Per+21]. Right in the beginning, the assumption $|S^*| > 3$ is made, where S^* is an optimal solution. This is sound because the algorithm would output an optimal solution if $|S^*| \leq 3$ (because of S_2). Our instance in fact corresponds to the $|S^*| > 3$ case, as $\{e_1, e_2, e_3, e_4\}$ is the unique optimal solution. In the next step, the wrong claim is made: According to Perrault et al., there is an index $\ell \geq 3$ such that $g(S_\ell) \leq g(S^*) \leq g(S_{\ell+1})$. Since g is monotone, this means $g(S^*) \geq g(S_3)$ in particular. However,

$$g(S^*) = g(\{e_1, e_2, e_3, e_4\}) = 1+1+1+1-\frac{4^2}{30} < 41 - \frac{1^2}{30} = g(\{e_5\}) = g(S_3),$$

rendering this step invalid. We note that this cannot even be fixed by scaling $g(S^*)$ by some constant, as the discrepancy between $g(S^*)$ and $g(S_3)$ can be made arbitrarily

large. Later in the analysis, the facts that $g(S^*)$ is squeezed between $g(S_\ell)$ and $g(S_{\ell+1})$ and that $\ell \geq 3$ are required in several steps.

In conclusion, the analysis is flawed and makes it at least questionable whether the algorithm achieves the claimed approximation factor of $1 - e^{c_g - 1}$. Although this is the case in our example ($1 - e^{c_g - 1}$ is slightly more than $1/2$, while S_2 provides a $26/27$-approximation), we leave disproving or verifying this for all instances to further investigations.

(B) We solve the knapsack-constrained ratio maximisation problem by a PTAS-style algorithm that combines the unconstrained Ψ-maximisation by Perrault et al. [Per+21] with a partial enumeration wrapper and some combinatorial observations. The overall algorithm and its analysis are quite different from the proposed but flawed algorithm Perrault et al. provide. Refer to Algorithm 4 for a formal description of the complete algorithm.

Theorem 4.5. *Let $f, g : 2^E \to \mathbb{R}_{\geq 0}$ be monotone submodular functions with g of low curvature. Further let $B \geq 0$ be a non-negative budget and $\varepsilon > 0$. Algorithm 4 outputs a set $S \subseteq E$ always satisfying $g(S) \leq B$ and also satisfying*

$$(1 - \varepsilon)\left(1 - e^{c_g - 1}\right) \frac{f(S^*)}{g(S^*)} \leq \frac{f(S)}{g(S)}$$

for any $S^* \subseteq E$ with $g(S^*) \leq B$ for which the following condition $(\star)$ holds true.

$(\star)$ For some constant[3] $0 \leq c < 1$, there does not exist any subset $S^*_+ \subseteq S^*$ with $\Delta_e(g \mid \varnothing) > \frac{B(1-c)}{\max\left\{\frac{2+\varepsilon}{\varepsilon}, 5\right\}}$ for all $e \in S^*_+$ and $g(S^*_+) > cB$.

The running time is polynomial in n for fixed ε, c_g and c.

Our proof strategy involves three steps. First, we introduce the notions of *heavy* and *light* elements. We then proceed to showing that any feasible solution can only contain a small number of heavy elements, allowing us to brute-force over the set of heavy elements contained in an optimal solution. In this light, Lemma 4.6 is the underlying reason why partial enumeration works on this problem. Next, we show the rather technical Lemma 4.7 that allows us to reduce a solution S (satisfying certain properties) to a solution $S^L \subseteq S$ that is almost as good but guaranteed to be found by Algorithm 4. After proving both lemmas, we show that Algorithm 4 is correct if all elements are light, which is captured by Theorem 4.8. This already proves the

[3]The running time is not polynomial as a function of c.

special case $c = 0$ of Theorem 4.5. We will then put everything together for a full proof of Theorem 4.5.

Given the function g, budget B and a parameter $L \geq 1$, we call an element $e \in E$ (g, B, L)-*light* if

$$\Delta_e(g \mid \varnothing) \leq \frac{B}{L}.$$

Otherwise, e is (g, B, L)-*heavy*. The distinction between heavy and light elements drives the partial enumeration step that brute-forces over all possible heavy elements an optimal solution could contain.

Lemma 4.6. *Any set $S \subseteq E$ with $g(S) \leq B$ contains at most $L/(1 - c_g)$ many (g, B, L)-heavy elements.*

Proof. Fix a set $S \subseteq E$ with $g(S) \leq B$ and let $S^+ \subseteq S$ denote the subset of (g, B, L)-heavy elements in S. By definition, it holds

$$\Delta_e(g \mid \varnothing) > \frac{B}{L}$$

for all $e \in S^+$. Label the elements of S^+ as $S^+ = \{e_1^+, \ldots, e_k^+\}$. We then have

$$\begin{aligned} g(S) \geq g(S^+) \geq g(S^+) - g(\varnothing) &= \sum_{i=1}^{k} \Delta_{e_i^+}\left(g \mid \{e_1^+, \ldots, e_{i-1}^+\}\right) \\ &\geq \sum_{i=1}^{k} (1 - c_g)\, \Delta_{e_i^+}(g \mid \varnothing) \\ &\geq \sum_{i=1}^{k} \frac{(1 - c_g)B}{L} = \frac{(1 - c_g)B|S^+|}{L}. \end{aligned}$$

Since we also have $g(S) \leq B$, it follows that $\frac{(1-c_g)B|S^+|}{L} \leq B$, hence $|S^+| \leq \frac{L}{1-c_g}$. □

Lemma 4.7. *Suppose $g(\varnothing) \leq \lambda B$ for a constant $0 \leq \lambda < 1$ and $L \geq 5/(1 - \lambda)$. Let $S \subseteq E^-$ be a set of (g, B, L)-light elements satsifying $\left(1 - \frac{1}{L}\right) B < g(S) \leq B$. Then there exists a subset $S^L \subseteq S$ such that $g(S) \leq \left(1 - \frac{1}{L}\right) B$ and*

$$\left(1 - \frac{2}{(1 - \lambda)L - 1}\right) \frac{f(S)}{g(S)} \leq \frac{f(S^L)}{g(S^L)}. \tag{4.2}$$

Proof. The idea is partitioning the elements of S into k groups $S = G_1 \cup \cdots \cup G_k$ such that $\frac{1}{L}B < g(G_i) - g(\varnothing) < \frac{2}{L}B$ for each $1 \leq i \leq k$. Such a partition is constructed according to Algorithm 3. We establish the following properties:

(i) It holds $\frac{1}{L}B \leq g(G_i) - g(\varnothing) < \frac{2}{L}B$ for each $1 \leq i \leq k$.

(ii) We always have $k \geq \frac{1}{2}\left((1-\lambda)L - 1\right)$ groups.

Combining both properties will then allow us to prove the lemma.

Algorithm 3 Partitioning S into groups $G_1, \ldots, G_k$

```
1: procedure PARTITION(S, B, L)
2:     k ← 0
3:     while G_1 ∪ ··· ∪ G_k ≠ S do
4:         k ← k + 1
5:         G_k ← ∅
6:         while g(G_k) − g(∅) < (1/L)B do
7:             Select e ∈ S \ (G_1 ∪ ··· ∪ G_k) arbitrarily
8:             G_k ← G_k ∪ {e}
9:         end while
10:    end while
11:    return G_1, ..., G_k
12: end procedure
```

For (i), it is crucial to remember that all elements $e \in S$ are (g, B, L)-light, so $\Delta_e(g \mid \varnothing) \leq \frac{1}{L}B$ for all $e \in S$. Since PARTITION in Algorithm 3 stops adding elements to G_k once $g(G_k) - g(\varnothing) \geq \frac{1}{L}B$, we trivially have the lower bound but also obtain the upper bound by considering the last element $e \in S$ that is added to G_k. It satisfies

$$\begin{aligned} g(G_k) - g(\varnothing) &= g((G_k \setminus \{e\}) \cup \{e\}) - g(G_k \setminus \{e\}) + g(G_k \setminus \{e\}) - g(\varnothing) \\ &= \Delta_e(g \mid G_k \setminus \{e\}) + g(G_k \setminus \{e\}) - g(\varnothing) \\ &\leq \underbrace{\Delta_e(g \mid \varnothing)}_{\leq \frac{1}{L}B} + \underbrace{g(G_k \setminus \{e\}) - g(\varnothing)}_{< \frac{1}{L}B} < \frac{2}{L}B. \end{aligned}$$

For (ii), we examine the "way" from $g(\varnothing)$ to $g(S)$ and use (i) to bound the number of groups necessary to get to $g(S)$ from below. To this end, we express $g(G_1 \cup \cdots \cup G_k)$ as a telescoping sum and use the submodularity of g to obtain an upper bound:

$$\begin{aligned} g(G_1 \cup \cdots \cup G_k) - g(\varnothing) &= \sum_{i=1}^{k} g(G_1 \cup \cdots \cup G_i) - g(G_1 \cup \cdots \cup G_{i-1}) \\ &\leq \sum_{i=1}^{k} g(G_i) - g(\varnothing) \leq \sum_{i=1}^{k} \frac{2}{L}B = \frac{2kB}{L}. \end{aligned}$$

The third step is by (i) and the second step follows from the submodularity of g applied to $G_{\leq i-1} := G_1 \cup \cdots \cup G_{i-1}$ (we also set $G_{\leq i} := G_1 \cup \cdots \cup G_i$) and G_i:

$$\begin{aligned} & g(G_{\leq i}) + g(\varnothing) = g(G_{\leq i-1} \cup G_i) + g(G_{\leq i-1} \cap G_i) \geq g(G_{\leq i-1}) + g(G_i) \\ \iff & g(G_{\leq i}) - g(G_{\leq i-1}) \geq g(G_i) - g(\varnothing) \end{aligned}$$

Since $g(S) = g(G_1 \cup \cdots \cup G_k)$, we also have the lower bound

$$g(G_1 \cup \cdots \cup G_k) - g(\varnothing) = g(S) - g(\varnothing) \geq \left(1 - \frac{1}{L}\right) B - \lambda B = \left(1 - \frac{1}{L} - \lambda\right) B$$

by the assumptions $g(S) \geq \left(1 - \frac{1}{L}\right)$ and $g(\varnothing) \leq \lambda B$. Putting upper bound and lower bound together, we conclude that

$$\left(1 - \frac{1}{L}\right) B - \lambda B \leq \frac{2kB}{L},$$

which implies (ii) after a few algebraic transformations.

We are now ready to show the existence of $S^L \subseteq S$ satisfying Eq. (4.2). Set

$$S^L := \arg\max_{S \in \mathcal{S}} \frac{f(S)}{g(S)} \qquad \text{where} \qquad \mathcal{S} := \left\{ \bigcup_{i \in \{1,\ldots,k\} \setminus \{j\}} G_i \mid j \in \{1, \ldots, n\} \right\}.$$

In words, we select S^L among the candidates $\bigcup_{i \in \{1,\ldots,k\} \setminus \{j\}} G_i$, i. e., uniting all groups except for one and maximising over the group not included. There are two cases.

Case 1: $\Delta_{G_j}(f \mid \varnothing) \leq \frac{1}{k} f(S)$ for some group G_j.
In this case, $S^L = \bigcup_{i \in \{1,\ldots,k\} \setminus \{j\}} G_i$ does the job. This is because

$$\begin{aligned} f(S) = f(G_1 \cup \cdots \cup G_k) &= \Delta_{G_j}\left(f \mid \bigcup_{i \neq j} G_i\right) + f\left(\bigcup_{i \neq j} G_j\right) \\ &\leq \Delta_{G_j}(f \mid \varnothing) + f\left(\bigcup_{i \neq j} G_j\right) \\ &\leq \frac{1}{k} f(S) + f\left(\bigcup_{i \neq j} G_j\right), \end{aligned}$$

implying $f(S^L) \geq f\left(\bigcup_{i \neq j} G_j\right) \geq \left(1 - \frac{1}{k}\right) f(S)$.

Case 2: $\Delta_{G_j}(f \mid \varnothing) > \frac{1}{k} f(S)$ for all groups G_j.
For the sake of contradiction, assume $f\left(\bigcup_{i \neq j} G_j\right) < \left(1 - \frac{1}{k}\right) f(S)$ for all j. Then

$$\begin{aligned} f(S) - f(\varnothing) &= f(G_1 \cup \cdots \cup G_k) - f(\varnothing) \\ &= \sum_{j=1}^{k} \Delta_{G_j}\left(f \mid \bigcup_{i \neq j} G_j\right) \\ &= \sum_{j=1}^{k} f(G_1 \cup \cdots \cup G_k) - f\left(\bigcup_{i \neq j} G_j\right) \\ &> \sum_{j=1}^{k} f(S) - \left(1 - \frac{1}{k}\right) f(S) \\ &= f(S), \end{aligned}$$

which is clearly a contradiction as $f(S) - f(\varnothing) \leq f(S)$. Consequently, there must be a group G_j such that $f\left(\bigcup_{i \neq j} G_j\right) \geq \left(1 - \frac{1}{k}\right) f(S)$.

It remains to put things together. By monotonicity, we have $g(S^L) \leq g(S)$ and thus

$$\frac{f(S^L)}{g(S^L)} \geq \frac{\left(1 - \frac{1}{k}\right) f(S)}{g(S)} = \left(1 - \frac{1}{k}\right) \frac{f(S)}{g(S)},$$

proving the lemma as $k \geq \frac{1}{2}\left((1 - \lambda)L - 1\right)$ by (ii). □

Theorem 4.8 (Knapsack-Constrained Ratio Maximisation Lite)**.** *Let $f, g : 2^E \to \mathbb{R}_{\geq 0}$ be monotone submodular functions with g of low curvature. Let $B \geq 0$ be a non-negative budget, $\varepsilon > 0$ and $0 \leq c < 1$ such that $g(\varnothing) \leq cB$. Define $L := \frac{\max\left\{\frac{2+\varepsilon}{\varepsilon}, 5\right\}}{1-c}$. Further suppose all elements $e \in E$ are (g, B, L)-light.* MAX-RATIO-LIGHT*(E, f, g, B) in Algorithm 4 outputs a set $S \subseteq E$ such that $g(S) \leq B$ and*

$$(1 - \varepsilon)\left(1 - e^{c_g - 1}\right) \frac{f(S^*)}{g(S^*)} \leq \frac{f(S)}{g(S)}$$

for all $S^* \subseteq E$ with $g(S^*) \leq B$.

Proof. Fix $S^* \subseteq E$ with $g(S^*) \leq B$. Since all elements are (g, B, L)-light, we know that MAX-RATIO-LIGHT behaves as if there was no knapsack constraint as long as $g(S_\ell) \leq \left(1 - \frac{1}{L}\right) B$, as each element e_ℓ can add no more than $\frac{1}{L} B$ to the value of g. In this case, MAX-RATIO-LIGHT coincides with Algorithm 2. Therefore, if we have $g(S^*) \leq \left(1 - \frac{1}{L}\right) B$, we know that MAX-RATIO-LIGHT will output an $(1 - e^{c_g - 1})$-approximation to S^* by Theorem 4.2 as if there was no knapsack constraint involved.

Algorithm 4 Approximate Ratio Maximisation under Knapsack Constraints

Input: Monotone submodular functions $f, g : 2^E \to \mathbb{R}_{\geq 0}$ with g of low curvature; budget $B \geq 0$; error parameter $\varepsilon > 0$; $0 \leq c < 1$

Output: $S \subseteq E$ such that $g(S) \leq B$ and $(1-\varepsilon)\,(1 - e^{c_g - 1})\,\frac{f(S^*)}{g(S^*)} \leq \frac{f(S)}{g(S)}$ for all $S^* \subseteq E$ with $g(S^*) \leq B$.

$S \leftarrow \varnothing$
$L \leftarrow \frac{\max\left\{\frac{2+\varepsilon}{\varepsilon}, 5\right\}}{1-c}$
$E^+ \leftarrow \{e \in E \mid e \text{ is } (g, B, L)\text{-heavy}\}$
$E^- \leftarrow \{e \in E \mid e \text{ is } (g, B, L)\text{-light}\}$
for all $H \subseteq E^+$ with $|H| \leq \dfrac{L}{1 - c_g}$ **do**
    Define $\widehat{f} : 2^{E^-} \to \mathbb{R}_{\geq 0}, A \mapsto f(A \cup H)$
    Define $\widehat{g} : 2^{E^-} \to \mathbb{R}_{\geq 0}, A \mapsto g(A \cup H)$
    $S^- \leftarrow$MAX-RATIO-LIGHT$(E^-, \widehat{f}, \widehat{g}, B)$
    if $\frac{f(S^- \cup H)}{g(S^- \cup H)} > \frac{f(S)}{g(S)}$ **then**
        $S \leftarrow S^- \cup H$
    end if
end for
return S
procedure MAX-RATIO-LIGHT(E, f, g, B)
    $S_0 \leftarrow \varnothing$
    $U \leftarrow E$
    for $\ell = 1$ **to** n **do**
        $e_\ell \leftarrow \arg\max_{e \in U} \frac{\Delta_e(f \mid S_{\ell-1})}{\Delta_e(g \mid S_{\ell-1})}$
        if $g(S_{\ell-1} \cup \{e_\ell\}) \leq B$ **then**
            $S_\ell \leftarrow S_{\ell-1} \cup \{e_\ell\}$
        else
            $S_\ell \leftarrow S_{\ell-1}$
        end if
        $U \leftarrow U \setminus \{e_\ell\}$
    end for
    return $\arg\max_{0 \leq \ell \leq n} \frac{f(S_\ell)}{g(S_\ell)}$
end procedure

Let's turn to the more interesting case $g(S^*) > \left(1 - \frac{1}{L}\right) B$. Since $g(\varnothing) \leq cB$, $L \geq \frac{5}{1-c}$ by definition and the fact that all elements in $S^* \subseteq E$ are (g, B, L)-light, we observe that the requirements of Lemma 4.7 are satisfied with $\lambda := c$ and S^* (in the role of S). As a consequence, we get the existence of $S^L \subseteq S$ with $g(S^L) \leq \left(1 - \frac{1}{L}\right) B$ and

$$\left(1 - \frac{2}{(1-c)L - 1}\right) \frac{f(S^*)}{g(S^*)} \leq \frac{f(S^L)}{g(S^L)}.$$

The factor on the LHS is

$$1 - \frac{2}{(1-c)L - 1} \geq 1 - \frac{2}{(1-c)\frac{2+\varepsilon}{\varepsilon(1-c)} - 1} = 1 - \frac{2}{\frac{2+\varepsilon}{\varepsilon} - 1} = 1 - \varepsilon,$$

hence $(1-\varepsilon)\frac{f(S^*)}{g(S^*)} \geq \frac{f(S^L)}{g(S^L)}$. Now, we apply the same argument to S^L as we have applied in the $g(S^*) \leq \left(1 - \frac{1}{L}\right)$ case above. MAX-RATIO-LIGHT will behave exactly the same as Algorithm 2 until $g(S_\ell) > \left(1 - \frac{1}{L}\right) B$. Before this happens, however, an $(1 - e^{c_g - 1})$-approximation of S^L has been found according to Theorem 4.2. Thus, the output S of MAX-RATIO-LIGHT is an $(1 - e^{c_g - 1})$-approximation of S^L, which in turn is a $(1-\varepsilon)$-approximation of S^*. Thus, S is a $(1-\varepsilon)\,(1 - e^{c_g - 1})$-approximation of S^*, as desired. □

Remark 4.9. If $c = 0$ in the statement of Theorem 4.5, we see that the assumption $(\star)$ asserts that there is no set $S^*_+ \subseteq S^*$ with $\Delta_e(g \mid \varnothing) > \frac{B}{\max\left\{\frac{2+\varepsilon}{\varepsilon}, 5\right\}}$ for all $e \in S^*_+$ such that $g(S^*_+) > 0$. This simplifies to the equivalent assertion: There is no element $e \in S^*$ with $\Delta_e(g \mid \varnothing) > \frac{B}{\max\left\{\frac{2+\varepsilon}{\varepsilon}, 5\right\}}$. If all elements $e \in E$ are (g, B, L)-light for $L = \max\left\{\frac{2+\varepsilon}{\varepsilon}, 5\right\}$ (which is the choice of L when $c = 0$), $(\star)$ is always satisfied. Hence, Theorem 4.8 proves Theorem 4.5 if $c = 0$.

We will now give a proof of Theorem 4.5 for general values of $0 \leq c < 1$. This is the point where the partial enumeration wrapper in Algorithm 4 comes into play.

Proof of Theorem 4.5. Fix $S^* \subseteq E$ with $g(S^*) \leq B$. The idea of the whole partial enumeration technique is the ability to split S^* into a heavy and a light part. Let $S^*_+ := S^* \cap E^+$ and $S^*_- := S^* \cap E^-$, where

$$E^+ = \{e \in E \mid e \text{ is } (g, B, L)\text{-heavy}\}$$
$$E^- = \{e \in E \mid e \text{ is } (g, B, L)\text{-light}\}$$

as in Algorithm 4. By the extra assumption, $g(S^*_+) \leq cB$. Moreover, monotonicity implies $g(S^*_+) \leq g(S^*) \leq B$, so $|S^*_+| \leq L/(1 - c_g)$ by Lemma 4.6. Therefore, the

for-loop iterating over all $H \subseteq E^+$ with $|H| \leq L/(1-c_g)$ will eventually hit $H = S_+^*$. In this iteration, MAX-RATIO-LIGHT$(E^-, \widehat{f}, \widehat{g}, B)$ is invoked, where $\widehat{f}(A) = f(A \cup S_+^*)$ and $\widehat{g}(A) = g(A \cup S_+^*)$ for all $A \subseteq E^-$. In particular, $\widehat{g}(\varnothing) = g(S_+^*) \leq cB$. Moreover, all items in E^- are $(\widehat{g}, B, L)$-light:

$$\Delta_e(\widehat{g} \mid \varnothing) = \widehat{g}(\{e\}) - \widehat{g}(\varnothing) = g(S_+^* \cup \{e\}) - g(S_+^*) = \Delta_e\left(g \mid S_+^*\right) \leq \Delta_e(g \mid \varnothing) \leq \frac{B}{L}.$$

Here, the last step follows because $e \in E^-$ is (g, B, L)-light by construction. Finally, $\widehat{g}$ is monotone as well and also has low curvature because

$$\frac{\Delta_e(\widehat{g} \mid A)}{\Delta_e(\widehat{g} \mid \varnothing)} = \frac{g(A \cup S_+^* \cup \{e\}) - g(A \cup S_+^*)}{g(S_+^* \cup \{e\}) - g(S_+^*)} \geq \frac{g(A \cup S_+^* \cup \{e\}) - g(A \cup S_+^*)}{g(\{e\}) - g(\varnothing)} \geq 1 - c_g$$

where the first inequality follows by the diminishing returns property and the second step by the fact that g has curvature c_g. We conclude that $\widehat{g}$ has curvature $c_{\widehat{g}} \leq c_g$.

Putting all these observations together, we see that Theorem 4.8 applies. Thus, the set S^- returned by MAX-RATIO-LIGHT$(E^-, \widehat{f}, \widehat{g}, B)$ satisfies

$$(1-\varepsilon)\left(1 - e^{c_{\widehat{g}}-1}\right)\frac{\widehat{f}(S_-^*)}{\widehat{g}(S_-^*)} \leq \frac{\widehat{f}(S^-)}{\widehat{g}(S^-)} \tag{4.3}$$

and $\widehat{g}(S^-) \leq B$ for S_-^* in particular (as $\widehat{g}(S_-^*) = g(S_+^* \cup S_-^*) = g(S^*) \leq B$). Putting S_+^* and S^- together will result in a feasible, near-optimal solution: The feasiblity is easily checked via $g(S_+^* \cup S^-) = \widehat{g}(S^-) \leq B$. For the approximation ratio, note that

$$\frac{f(S_+^* \cup S^-)}{g(S_+^* \cup S^-)} = \frac{\widehat{f}(S^-)}{\widehat{g}(S^-)}$$

as well as $1 - e^{c_{\widehat{g}}-1} \geq 1 - e^{c_g-1}$ and combine it with Eq. (4.3). This shows that $S_+^* \cup S^-$ is indeed a $(1-\varepsilon)(1-e^{c_g-1})$-approximation of S^*. We finally remark that Algorithm 4 optimises over all $S^- \cup H$ and thus outputs a solution of value at least $\frac{f(S_+^* \cup S^-)}{g(S_+^* \cup S^-)}$, thereby achieving the bound stated in Theorem 4.5. □

4.1.3 Assembling a Sparsifier

Algorithm 2 and Algorithm 4 come in useful when approximating peak contributions.

We first treat the case of monotone submodular functions $f_1, \ldots, f_N : 2^E \to \mathbb{R}_{\geq 0}$ of low curvature without any knapsack constraints. The aim is to approximate the peak contributions $p_i = \max_{A \subseteq E} \frac{f_i(A)}{F(A)}$ for all $1 \leq i \leq N$. If all f_i's have curvatures $c_{f_i} < 1$,

so has F by Lemma 4.12 shown at the end of this section. Consequently, we can apply Algorithm 2 to find sets $A_1, \ldots, A_N$ such that

$$\left(1 - e^{c_F - 1}\right) p_i = \left(1 - e^{c_F - 1}\right) \frac{f_i(A_i^*)}{F(A_i^*)} \leq \frac{f_i(A_i)}{F(A_i)} \leq \frac{f_i(A_i^*)}{F(A_i^*)} = p_i$$

where $A_i^* = \arg\max_{S \subseteq E} f_i(S)/F(A)$ for $1 \leq i \leq N$. If we use the quantities

$$\widehat{p_i} := \frac{1}{1 - e^{c_F - 1}} \frac{f_i(A_i)}{F(A_i)}$$

instead of the exact p_i's in Algorithm 1, it does still work (as $\widehat{p_i} \geq p_i$, so Corollary 3.4 applies) and, noting that $\widehat{p_i} \leq p_i / \left(1 - e^{c_F - 1}\right)$, the size bound becomes

$$\begin{aligned}
\mathbb{E}\left[\mathsf{size}(w)\right] &= \mathcal{O}\left(\frac{n + \log\frac{1}{\delta}}{\varepsilon^2} \sum_{i=1}^{N} \widehat{p_i}\right) \\
&= \mathcal{O}\left(\frac{n + \log\frac{1}{\delta}}{\varepsilon^2} \sum_{i=1}^{N} \frac{p_i}{1 - e^{c_F - 1}}\right) \\
&= \mathcal{O}\left(\frac{n + \log\frac{1}{\delta}}{\left(1 - e^{c_F - 1}\right)\varepsilon^2} \sum_{i=1}^{N} p_i\right) \\
&= \mathcal{O}\left(\frac{Bn\left(n + \log\frac{1}{\delta}\right)}{\left(1 - e^{c_F - 1}\right)\varepsilon^2}\right)
\end{aligned}$$

which is the original bound with an additional $1/\left(1 - e^{c_F - 1}\right)$ factor. Here, the first few steps are using the general bound stated in Theorem 3.1 part (ii), while the last step specialises to the monotone submodular case using Lemma 3.9. For constant δ, we get the simplified bound $\mathbb{E}\left[\mathsf{size}(w)\right] = \mathcal{O}\left(\frac{Bn^2}{\left(1 - e^{c_F - 1}\right)\varepsilon^2}\right)$, which is almost as good as the existence result stated by Theorem 3.1 and in [RY22], both specialised to monotone functions. Furthermore, if we are interested in polynomial-time computability, this bound is much better than what we get in other special cases such as the $\mathcal{O}\left(\frac{Bn^{2.5}\log n}{\varepsilon^2}\right)$ size bound for monotone functions (see Remark 3.10), which approximates the peak contributions up to a $\mathcal{O}\left(\sqrt{n}\log n\right)$ factor by the ellipsoid method [Bai+16], while our algorithm in the low curvature case uses a simple greedy algorithm.

Remark 4.10. When defining the $\widehat{p_i}$'s, it becomes evident (again) why we require $c_F < 1$. Low curvature is necessary for the $\widehat{p_i}$'s to be well-defined.

Remark 4.11. There is a good reason to not disregard $1/\left(1 - e^{c_F - 1}\right)$ as a constant in the size bound. It is mainly because we consider N as part of the input. Thus, as N grows, we add more functions f_i to the decomposition whose curvatures define

the curvature of F. If the sequence $(c_{f_i})_{i\geq 1}$ converges to 1 (but never hits it), the curvature of $F = f_1 + \cdots + f_N$ can be seen to converge to 1 as well, making the factor $1/\left(1 - e^{c_F - 1}\right)$ approach $+\infty$. From this perspective, it is by no means constant.

Now turn to the knapsack-constrained setting, where monotone submodular functions $f_1, \ldots, f_N : 2^E \to \mathbb{R}_{\geq 0}$ of low curvature and a budget $B \geq 0$ are given. The goal is to sparsify $F = f_1 + \cdots + f_N$ only the subdomain

$$\mathcal{D}_{\leq B} := \{A \subseteq E \mid F(A) \leq B\}$$

where the value of F does not exceed the budget B. To make it clear, given $\varepsilon > 0$, we want a sparsifier $w = (w_1, \ldots, w_N)$ such that

$$(1 - \varepsilon)F(A) \leq F'(A) \leq (1 + \varepsilon)F(A)$$

for all $A \in \mathcal{D}_{\leq B}$. Such a sparsifier on $\mathcal{D}_{\leq B}$ is clearly superior w. r. t. size. We have

$$p_i = \max_{A \in \mathcal{D}_{\leq B}} \frac{f_i(A)}{F(A)} \leq \max_{A \subseteq E} \frac{f_i(A)}{F(A)},$$

so the expected size $\mathcal{O}\left(\frac{n + \log \frac{1}{\delta}}{\varepsilon^2} \sum_{i=1}^N p_i\right)$ from Theorem 3.1 is smaller (or equal, in the worst case) than it would be for a sparsifier over the full domain 2^E.

To realise such a construction, our goal is approximating the values $\max_{A \in \mathcal{D}_{\leq B}} \frac{f_i(A)}{F(A)}$ up to a factor of $\mathcal{O}\left(1/\left(1 - e^{c_F - 1}\right)\right)$. Since we do not care about the additional constant factor, we choose $\varepsilon = 1/2$ to make $\frac{B \max\left\{\frac{2+\varepsilon}{\varepsilon}, 5\right\}}{1-c}$ as small as possible, i. e. equal to $\frac{5B}{1-c}$, in order to get the most out of Theorem 4.5. Now we see that Algorithm 4 guarantees a $\frac{1}{2}\left(1 - e^{c_F - 1}\right)$-approximation if the condition

> ($\star$) There is a constant $0 \leq c < 1$ for which there exists an optimal solution $S^* \in \arg\max_{S \in \mathcal{D}_{\leq B}} \frac{f_i(S)}{F(S)}$ such that the elements $e \in S^*$ with $\Delta_e(g \mid \varnothing) \leq \frac{5B}{1-c}$ form a set S^*_+ with $g(S^*_+) \leq cB$

is satisfied. In particular, ($\star$) holds if $\Delta_e(g \mid \varnothing) \leq B/5$ for all $e \in E$. Given ($\star$), we can run Algorithm 4, obtain outputs $A_1, \ldots, A_N \in \mathcal{D}_{\leq B}$ for $i = 1, \ldots, N$, and use them to define

$$\widehat{p}_i := \frac{1}{\frac{1}{2}\left(1 - e^{c_F - 1}\right)} \frac{f_i(A_i)}{F(A_i)}$$

for all $1 \leq i \leq N$. Then, as before in the unconstrained case, $p_i \leq \widehat{p}_i \leq \frac{1}{2}\left(1 - e^{c_F - 1}\right) p_i$ for all $1 \leq i \leq N$. Applying Algorithm 1 and Corollary 3.4, we get an ε-sparsifier

of expected size $\mathcal{O}\left(\frac{n+\log\frac{1}{\delta}}{\left(1-e^{c_F-1}\right)\varepsilon^2}\sum_{i=1}^N p_i\right)$, which is asymptotically the same as in the unconstrained case but with p_i's only defined over the subdomain $\mathcal{D}_{\leq B}$.

Finally, as promised earlier, we are left to prove that $c_F < 1$ if $F = f_1 + \cdots + f_N$ and all f_i's have low curvature. It can be seen that the curvature c_F of F is at most the maximum of the curvatures c_{f_i} of the f_i's.

Lemma 4.12. *Let $f_1, \ldots, f_N : 2^E \to \mathbb{R}_{\geq 0}$ be monotone, non-negative submodular functions with curvatures $c_{f_1}, \ldots, c_{f_N} < 1$. Then the sum $F = f_1 + \cdots + f_N$ is a monotone, non-negative submodular function with curvature $c_F \leq \max\{c_{f_1}, \ldots, c_{f_N}\} < 1$.*

Proof. Let $c_{\max} := \max\{c_{f_1}, \ldots, c_{f_N}\}$ be the maximum curvature among $f_1, \ldots, f_N$. We consider the quotient $\Delta_e(F \mid S)/\Delta_e(F \mid \varnothing)$ to get a bound on the curvature c_F of F. Using $F = f_1 + \cdots + f_N$, it can be rewritten and estimated as

$$\begin{aligned}
\frac{\Delta_e(F \mid S)}{\Delta_e(F \mid \varnothing)} &= \frac{\sum_{i=1}^N f_i(S \cup \{e\}) - \sum_{i=1}^N f_i(S)}{\sum_{i=1}^N f_i(\{e\}) - \sum_{i=1}^N f_i(\varnothing)} \\
&= \frac{\sum_{i=1}^N \frac{f_i(S\cup\{e\})-f(S)}{f_i(\{e\})-f_i(\varnothing)}\left(f_i(\{e\}) - f_i(\varnothing)\right)}{\sum_{i=1}^N f_i(\{e\}) - f_i(\varnothing)} \\
&\geq \frac{\sum_{i=1}^N \min_{A\subseteq E}\min_{e'\notin A}\left(\frac{f_i(A\cup\{e'\})-f(A)}{f_i(\{e'\})}\right)\left(f_i(\{e\} - f(\varnothing))\right)}{\sum_{i=1}^N f_i(\{e\}) - f_i(\varnothing)} \\
&= \frac{\sum_{i=1}^N (1-c_{f_i})\left(f_i(\{e\}) - f_i(\varnothing)\right)}{\sum_{i=1}^N \left(f_i(\{e\}) - f_i(\varnothing)\right)} \\
&\geq \frac{\sum_{i=1}^N (1-c_{\max})\left(f_i(\{e\}) - f_i(\varnothing)\right)}{\sum_{i=1}^N \left(f_i(\{e\}) - f_i(\varnothing)\right)} \\
&= 1 - c_{\max}.
\end{aligned}$$

Taking the minimum over all $S \subseteq E, e \notin S$, we still end up with a positive number $\geq 1 - c_{\max}$. Thus, $c_F \leq 1 - (1 - c_{\max}) = c_{\max} < 1$. Lastly, as a sum of monotone set functions, F is monotone. □

4.2 Bounded Arity

4.2.1 The Arity of a Submodular Function

An improvement in both size and construction time of the ε-sparsifier is possible if the submodular function F in question can be decomposed as $F = f_1 + \cdots + f_N$ into f_i's of *bounded arity*. This covers a broad range of submodular functions such as cut functions. We will first motivate the bounded arity special case and then present our algorithm that computes the peak contributions in a brute-force fashion.

To motivate the bounded arity property, consider a set function $f : 2^E \to \mathbb{R}$. Instead of writing $f(S)$ for $S \subseteq E$ to describe the values of f, we may encode S as a binary vector of $|E|$ coordinates. Let $E = \{e_1, \ldots, e_n\}$ and set the i^{th} coordinate of the vector corresponding to S to 1 iff $e_i \in S$. This standard identification allows us to write $f(x_1, \ldots, x_n)$ for $(x_1, \ldots, x_n) \in \{0, 1\}^n$ to describe the values of f. From this perspective, it is intuitive to say that f has *arity* a if its values only depend on a arguments. It is worth noting that this representation aligns with how one would represent f in the context of constraint satisfaction problems, as part of a valued constraint language [KTŽ15]. Tieing this back to the initial notation $f(S)$, we recognise that f having arity a means there exists a set $C \subseteq E$ of size $|C| = a$ such that $f(S) = f(S \cap C)$ for all $S \subseteq E$. That is exactly the definition we choose to work with. In this setting, we call C the *effective support* of f.

4.2.2 Computing The Peak Contributions

Turning back to sparsification, we start from a decomposition $F = f_1 + \cdots + f_N$ of a submodular function F into submodular functions $f_1, \ldots, f_N$ of arity $\leq a$ for some constant a. Let $C_1, \ldots, C_N$ denote the effective supports of $f_1, \ldots, f_N$, respectively. In order to construct a sparsifier, we want to apply Algorithm 1. To do so efficiently, we need to be able to compute the peak contributions, i. e., maximise the quotients $\frac{f_i(A)}{F(A)}$ over all $A \subseteq E$ for each $1 \leq i \leq N$. Since the numerators only depend on how A intersects the C_i's – which are of constant size – we might brute-force this intersection and minimise the denominator w. r. t. an extra constraint.

Remark 4.13. To make this idea work, it is vital for us to *know* the C_i's. There might be scenarios where all f_i's are known to have bounded arity while the elements they are supported on are not computationally available. However, if f is monotone and non-negative, the effective support can be found by testing each element $e \in E$

for $f(\{e\}) > f(\varnothing)$. If the test fails, we know that e does not influence the values of f and hence does not belong to its effective support.

Algorithm 5 Computing the peak contributions for f_i's of bounded arity

Input: Functions $f_1, \ldots, f_N$ with effective supports $C_1, \ldots, C_N$ of size $\leq a$
Output: $p_i = \max_{A \subseteq E} \frac{f_i(A)}{F(A)}$ for each $1 \leq i \leq N$

```
for i = 1, ..., N do
    p_i ← 0
    for all H ⊆ C_i do
        Compute a minimiser A* of F̂ : 2^{E\C_i} → ℝ, A ↦ F(A ∪ H)
        p_i ← max{ p_i, f_i(H) / F̂(A*) }
    end for
end for
return p_1, ..., p_N
```

Refer to Algorithm 5 for a formal description of how the peak contributions are computed. For each $1 \leq i \leq N$, the quantity p_i is computed as follows. We first choose a set $H \subseteq C_i$ and then restrict ourselves to sets $A \subseteq E$ with $A \cap C_i = H$. This implies $f_i(A) = f_i(A \cap C_i) = f_i(H)$ as C_i is the effective support of f_i. Hence, we are left with the quotient $f_i(H)/F(A)$ that has to be maximised over all $A \subseteq E$ with $A \cap C_i = H$. Since the numerator no longer depends on A, the objective becomes minimising $F(A)$ subject to $A \cap C_i = H$. It turns out that this is just a submodular minimisation problem of an auxiliary function $\widehat{F}$. The following statement is key to the correctness of this approach.

Lemma 4.14. *Let $f, F : 2^E \to \mathbb{R}$ be submodular functions and let $C \subseteq E$ denote the effective support of f. Then*

$$\max_{A \subseteq E} \frac{f(A)}{F(A)} = \max_{H \subseteq C} \frac{f(H)}{\min_{A \subseteq E \setminus C} \widehat{F}(A)}$$

where $\widehat{F} : 2^{E \setminus C} \to \mathbb{R}, A \mapsto F(A \cup H)$.

Proof. "$\leq$" Let A^* be a maximiser of the LHS. Set $H := A^* \cap C$ and $A := A^* \setminus C$. We now have

$$\frac{f(A^*)}{F(A^*)} = \frac{f\left(A^* \cap C\right)}{F\left((A^* \setminus C) \cup (A^* \cap C)\right)} = \frac{f(H)}{F\left((A^* \setminus C) \cup H\right)} = \frac{f(H)}{\widehat{F}(A)}$$

with $H \subseteq C$ and $A \subseteq E \setminus C$, so this is certainly $\leq$ the RHS above.

"$\geq$" Let (H^*, A^*) a maximising pair of the RHS. Letting $A := A^* \cup H^*$, we get

$$\frac{f(H^*)}{\widehat{F}(A^*)} = \frac{f(A \cap C)}{F(A^* \cup H)} = \frac{f(A)}{F(A)},$$

which is always $\leq$ the LHS as it maximises over all A. □

One more observation allows us to give a more precise size bound on the sparsifier in the bounded arity case. By Theorem 3.1, the ε-sparsifier produced by Algorithm 1 is of expected size $\mathcal{O}\left(\frac{\log|\mathcal{E}|+\log\frac{1}{\delta}}{\varepsilon^2}\sum_{i=1}^{N} p_i\right)$, where $\sum_{i=1}^{N} p_i \leq Bn$ by Lemma 3.9, if the f_i's are all monotone. In general, the quantity B might be huge. However, if the functions all have bounded arity, we can show that $B = \mathcal{O}(1)$, allowing for an ε-sparsifier of expected size $\mathcal{O}(n^2/\varepsilon^2)$ in case of monotone constituent functions – assuming δ is constant. The following statement claimed but not shown in [RY22] shows a bound on B that is independent of n or N but just depends on the arity.

Lemma 4.15. *Suppose $f : 2^E \to \mathbb{R}$ is submodular with arity $\leq a$. Then the base polyhedron $\mathcal{B}(f)$ has at most 2^{a^2} extreme points, i. e., $|\mathsf{EX}(\mathcal{B}(f))| \leq 2^{a^2}$.*

Proof. By Theorem 3.6, a point $x \in \mathcal{B}(f)$ is an extreme point iff there is a maximal chain $\varnothing = S_0 \subset S_1 \subset \cdots \subset S_n = E$ with $x(e_i) = f(S_i) - f(S_{i-1})$ for $1 \leq i \leq n$, where $\{e_i\} = S_i \setminus S_{i-1}$. Letting C denote the effective support of f, these equations become $x(e_i) = f(S_i \cap C) - f(S_{i-1} \cap C)$. Now $e_i \notin C$ implies

$$S_i \cap C = (S_{i-1} \cup \{e_i\}) \cap C = S_{i-1} \cap C,$$

hence $x(e_i) = f(S_i \cap C) - f(S_{i-1} \cap C) = f(S_{i-1} \cap C) - f(S_{i-1} \cap C) = 0$. Therefore, only the coordinates corresponding to elements $e \in C$ can be non-zero. For any such coordinate $e \in C$, we have

$$x(e) = f(S \cup \{e\}) - f(S) = f\left((S \cup \{e\}) \cap C\right) - f(S \cap C)$$

where S is the (unique) set in the maximal chain e is added to. Now observe that the RHS can attain at most 2^a possible values: One for each subset $S \subseteq C$. This is because the value is completely determined by how S intersects C.

Putting it all together, we have at most a non-zero coordinates with at most 2^a different values for each of them. Counting all combinations, we retrieve an upper bound of $(2^a)^a = 2^{a^2}$ candidates for x. Thus, $\mathcal{B}(f)$ has at most 2^{a^2} extreme points. □

4.2.3 Application to Hypergraph Cut Sparsification

The motivation of submodular sparsification partly arises from the sparsification of graph cuts, which have been extensively studied not only for graphs but also for hypergraphs [SY19; KK15; CKN20]. In the following, we use the results derived in Section 4.2.2 to construct a hypergraph cut sparsifier in polynomial time.

Given an r-uniform hypergraph $G = (V, E)$, we define an individual cut function

$$f_e : 2^V \to \mathbb{R}, \quad S \mapsto \begin{cases} 1 & \text{if } 0 < |S \cap e| < r, \\ 0 & \text{otherwise} \end{cases} \tag{4.4}$$

for each hyperedge $e \in E$. Letting $F := \sum_{e \in E} f_e$, it is not hard to see that $F(S)$ equals the number of hyperedges "cut" by S, i. e., whose endpoints are non-trivially split among S and $E \setminus S$. We remark that F is submodular as a sum of submodular functions, with the f_e's being submodular by Fact 4.16.

Fact 4.16. *Let $G = (V, E)$ be an r-uniform hypergraph and let $e \in E$. The function f_e defined in Eq. (4.4) is submodular.*

Proof. We establish the diminishing returns property $\Delta_v(f_e \mid T) \leq \Delta_v(f_e \mid S)$ for all $T \supseteq S$ and $v \notin T$. To this end, observe that

$$\Delta_v(f_e \mid A) = \begin{cases} 1 & \text{if } |e \cap A| = 0, \\ 0 & \text{if } 0 < |e \cap A| < r, \\ -1 & \text{if } |e \cap A| = r \end{cases}$$

for any set $A \subseteq V$ and $v \in e \setminus A$. Observe that $\Delta_v(f_e \mid A)$ is non-increasing as $|e \cap A|$ increases. Since $|e \cap T| \geq |e \cap S|$ by $T \supseteq S$, we get $\Delta_v(f_e \mid T) \leq \Delta_v(f_e \mid S)$ if $v \in e \setminus T$ (hence $v \in e \setminus S$). If $v \in V \setminus (T \cup e)$, i. e., $v \notin e$, the quantities $|e \cap S|$ and $|e \cap T|$ do not change upon adding v, so $\Delta_v(f_e \mid T) = \Delta_v(f_e \mid S) = 0$, which also satisfies the diminishing returns property. We conclude that f_e is submodular by Fact 2.1. □

In order to apply the results from Section 4.2.2 and efficiently construct a small sparsifier, we need that the f_e's have bounded arity and we need to know their effective supports. Fix a hyperedge $e \in E$. We claim that f_e has effective support e. This is best seen by Eq. (4.4): The value f_e assigns to S is 1 if $0 < |S \cap e| < r$ and 0 otherwise. In particular, it depends only on $S \cap e$ (in fact, only the size of it), hence $f_e(S) = f_e(S \cap e)$ for any $S \subseteq V$. Since G is r-uniform, we know that $|e| = r$, so f_e has arity r. If the uniformness of G is regarded as a constant, the f_e's have bounded arity and we can apply Algorithm 5 to compute the peak contributions of the f_e's w. r. t.

$F = \sum_{e \in E} f_e$. This, in a last step, allows us to compute an ε-sparsifier of expected size $\mathcal{O}\left(\frac{n+\log \frac{1}{\delta}}{\varepsilon^2} \sum_{e \in E} p_e\right)$ via Algorithm 1. Unfortunately, it is not clear how to get a non-trivial and expressive bound on the sum $\sum_{e \in E} p_e$ of peak contributions, since the result by Rafiey and Yoshida [RY22] has been disproven in Section 3.2 and only holds for monotone functions, which the hypergraph cut functions f_e do not belong to.

4.3 Sparsifier in the Limit

We have seen how to construct (in exponential time) an ε-sparsifier of expected size $\mathcal{O}\left(\frac{n+\log\frac{1}{\delta}}{\varepsilon^2}\sum_{i=1}^{N} p_i\right)$ in the submodular case. In this section, we address the question of achieving this size bound in polynomial time under a slightly weaker notion of ε-sparsifier. Since all ideas work not only on submodular functions but apply to the more general setting of a decomposable set function $F : 2^E \to \mathbb{R}$, we present all steps in this general setting and finally return to submodular functions for a more precise size bound. Given a decomposable set function $F : 2^E \to \mathbb{R}$ with $F = f_1 + \cdots + f_N$, define a *relaxed ε-sparsifier* as a vector $w \in \mathbb{R}^N$ such that

$$(1-\varepsilon)F(S) \leq F'(S) \leq (1+\varepsilon)F(S) \tag{4.5}$$

for all $S \in \mathcal{S}$, where $\mathcal{S} \subseteq 2^E$ is a set of $|\mathcal{S}| \geq \left(1-\frac{1}{n}\right)2^n$ elements, $n = |E|$, and $F' = \sum_{i=1}^{N} w_i f_i$. As before, we let $\mathsf{size}(w)$ denote the number of non-zero entries in w. The notion of a relaxed ε-sparsifier follows the idea of satisfying the guarantee Eq. (4.5) for *almost* all subsets $S \subseteq E$. More precisely, we only allow an $1/n$-fraction of subsets to violate it. Although $2^n/n \to \infty$ as $n \to \infty$, the fraction of subsets satisfying Eq. (4.5) is $\frac{|\mathcal{S}|}{|2^E|} \geq \frac{\left(1-\frac{1}{n}\right)2^n}{2^n} \to 1$ as $n \to \infty$, which explains the term "sparsifier in the limit".

As in Section 4.1 and 4.2, the approach comes down to estimating peak contributions. For each $1 \leq i \leq N$ and $S \subseteq E$, we define the ratio

$$\varrho_i(S) := \frac{f_i(S)}{F(S)}. \tag{4.6}$$

In light of this definition, we notice that $p_i = \max_{S \subseteq E} \varrho_i(S)$. The *rank* of a set $S \subseteq E$ w. r. t. f_i is then defined via

$$r_i(S) := \frac{1}{2^n}\left|\{A \subseteq E \mid \varrho_i(A) < \varrho_i(S)\}\right|, \tag{4.7}$$

i. e. the fraction of sets ranking strictly lower than S in the ratio $\varrho_i(\cdot)$.

The estimation of the p_i's is done by a simple randomised procedure:

- Sample $A_1, \ldots, A_m \subseteq E$ independently uniformly at random.
- For $1 \leq j \leq m$, compute the ratios $\varrho_i(A_j)$.
- Take $\widehat{p_i} := \max_{1 \leq j \leq m} \varrho_i(A_j)$ as an estimate of p_i.

Unlike in the situation of Corollary 3.4, the inequality $\widehat{p}_i \geq p_i$ does *not* hold for the $\widehat{p}_i$'s (unless $\widehat{p}_i = p_i$, which happens if a maximizer of $\varrho_i(\cdot)$ is sampled). This implies the proof of Theorem 3.1, part (i) fails, at least when attempting to establish Eq. (4.5) for all sets $S \subseteq E$. However, if we only impose it for an $\left(1 - \frac{1}{n}\right)$-fraction of the sets, we can exploit that $\widehat{p}_i \geq \varrho_i(S)$ for *almost all* $S \subseteq E$. This is implemented in Algorithm 6.

Algorithm 6 Relaxed Sparsification Algorithm

Input: Set function $F = f_1 + \cdots + f_N : 2^E \to \mathbb{R}$ with f_i's given by evaluation oracles; parameters $\varepsilon, \delta \in (0,1)$

Output: Vector $w \in \mathbb{R}^N$ such that

- $\mathbb{P}\left[w \text{ is a relaxed } \varepsilon\text{-sparsifier}\right] \geq 1 - 1/n^c$ for any constant $c > 0$,
- $\mathbb{E}\left[\mathsf{size}(w)\right] \leq \mathcal{O}\left(\frac{n+\log\frac{1}{\delta}}{\varepsilon^2}\sum_{i=1}^N p_i\right)$ where $p_i = \max\limits_{\substack{A\subseteq E\\ F(A)\neq 0}} \frac{f_i(A)}{F(A)}$.

$w \leftarrow 0$
$m \leftarrow \lceil 2nN\left((c+2)\log n + \log N\right)\rceil$
$\kappa \leftarrow 3\log\left(\frac{2n^c}{\delta}\right)/\varepsilon^2$
for $i = 1, \ldots, N$ **do**
    Sample $A_1, \ldots, A_m$ independently and uniformly at random
    $\widehat{p}_i \leftarrow \max_{1\leq j\leq m} \varrho_i(A_j)$
    $\kappa_i \leftarrow \min\{1, \kappa\widehat{p}_i\}$
    $w_i \leftarrow \begin{cases} 1/\kappa_i & \text{with probability } \kappa_i \\ 0 & \text{with probability } 1-\kappa_i \end{cases}$
end for
return w

The main goal for the rest of this section is showing that the output w of Algorithm 6 is a relaxed ε-sparsifier with high probability, and of small expected size.

Theorem 4.17. *Let $c > 0$. For any $\varepsilon, \delta \in (0,1)$, Algorithm 6 outputs a vector $w \in \mathbb{R}^N$ such that*

(i) $\mathbb{P}\left[w \text{ is a relaxed } \varepsilon\text{-sparsifier}\right] \geq 1 - \dfrac{1}{n^c}$,

(ii) $\mathbb{E}\left[\mathsf{size}(w)\right] = \mathcal{O}\left(\frac{n+\log 1/\delta}{\varepsilon^2}\sum_{i=1}^N p_i\right)$,

where – as before – $p_i = \max_{S\subseteq E} \varrho_i(S)$ for $1 \leq i \leq N$.

The second claim

$$\mathbb{E}\left[\mathsf{size}(w)\right] \leq \mathcal{O}\left(\frac{n+\log\frac{1}{\delta}}{\varepsilon^2}\sum_{i=1}^N p_i\right)$$

follows from the analysis of Algorithm 1 and the fact that $\widehat{p}_i \leq p_i$ for all $1 \leq i \leq N$. As for the first claim, we start with a statement about the ratios $\varrho_i(\cdot)$, ranks $r_i(\cdot)$ and how randomly chosen sets perform.

Lemma 4.18. *Let $m \geq 0$ and $A_1, \ldots, A_m$ be sampled independently uniformly at random. Further suppose $\widehat{p}_i = \max_{1 \leq j \leq m} \varrho_i(A_j)$ for $1 \leq i \leq N$. Then:*

(i) For any $X, Y \subseteq E$, we have $r_i(X) < r_i(Y)$ iff $\varrho_i(X) < \varrho_i(Y)$.

(ii) It holds $\mathbb{P}\left[\varrho_i(S) > \widehat{p}_i\right] \leq r_i(S)^m$ for each $S \subseteq E$.

Proof. We prove (i) first. Suppose $X, Y \subseteq E$ fulfill $r_i(X) < r_i(Y)$. This is equivalent to $|\mathcal{R}_X| < |\mathcal{R}_Y|$ if we let

$$\mathcal{R}_X := \{A \subseteq E \mid \varrho_i(A) < \varrho_i(X)\} \quad \text{and} \quad \mathcal{R}_Y := \{A \subseteq E \mid \varrho_i(A) < \varrho_i(Y)\}.$$

Hence, there exists an element $Z \in \mathcal{R}_Y \setminus \mathcal{R}_X$. Note that this already follows from $|\mathcal{R}_X| < |\mathcal{R}_Y|$, we do not need the stronger statement $\mathcal{R}_X \subset \mathcal{R}_Y$. By definition of $\mathcal{R}_X$ and $\mathcal{R}_Y$, this implies $\varrho_i(Z) \geq \varrho_i(X)$ and $\varrho_i(Z) < \varrho_i(Y)$, hence $\varrho_i(X) < \varrho_i(Y)$.

For the other direction, observe that $\varrho_i(X) < \varrho_i(Y)$ implies $\mathcal{R}_X \subseteq \mathcal{R}_Y$, and moreover, that $X \in \mathcal{R}_Y \setminus \mathcal{R}_X$, hence the inclusion is strict and the size inequality follows.

For (ii), the event $\{\varrho_i(S) > \widehat{p}_i\}$ occurs exactly when all the events $\{\varrho_i(S) > \varrho_i(A_j)\}$ for $1 \leq j \leq m$ occur. Since all A_j are sampled independently, we get

$$\mathbb{P}\left[\varrho_i(S) > \widehat{p}_i\right] = \mathbb{P}\left[\bigcap_{j=1}^{m} \{\varrho_i(S) > \varrho_i(A_j)\}\right] = \prod_{j=1}^{m} \mathbb{P}\left[\varrho_i(S) > \varrho_i(A_j)\right],$$

so it suffices to show $\mathbb{P}\left[\varrho_i(S) > \varrho_i(A_j)\right] \leq r_i(S)$ for each $1 \leq j \leq m$. The number of sets A with $\varrho_i(A) < \varrho_i(S)$ equals $2^n \cdot r_i(S)$ by definition of $r_i(S)$. Thus,

$$\mathbb{P}\left[\varrho_i(S) > \varrho_i(A_j)\right] \leq \frac{2^n \cdot r_i(S)}{2^n} = r_i(S)$$

by the fact that A_j is sampled uniformly at random. □

Let's call a set $S \subseteq E$ *high-ranked* if $r_i(S) \geq 1 - \frac{1}{2nN}$ for at least one $1 \leq i \leq N$. Otherwise, we call S *low-ranked*.

Lemma 4.19. *There are most $2^{n-1}/n$ high-ranked sets $S \subseteq E$.*

Proof. For each $1 \leq i \leq N$, let

$$\mathcal{H}_i := \left\{ S \subseteq E \mid r_i(S) \geq 1 - \frac{1}{2nN} \right\}$$

be the family of high-ranked sets with "witness" i. It follows that $\mathcal{H} := \bigcup_{i=1}^{N} \mathcal{H}_i$ is precisely the family of high-ranked sets. Fix an arbitrary $1 \leq i \leq N$. We aim to bound the size of $\mathcal{H}_i$. Let $S^- \in \mathcal{H}_i$ be an element of $\mathcal{H}_i$ of minimal rank. Thus, $r_i(S) < r_i(S^-)$ iff $S \notin \mathcal{H}_i$. By part (i) of Lemma 4.18, this condition is equivalent to $\varrho_i(S) < \varrho_i(S^-)$. By definition of $r_i(\cdot)$, there are precisely $2^n \cdot r_i(S^-)$ such sets S. Thus,

$$\left|2^E \setminus \mathcal{H}_i\right| = 2^n \cdot r_i(S^-) \geq 2^n \left(1 - \frac{1}{2nN}\right)$$

which implies $|\mathcal{H}_i| \leq \frac{2^n}{2nN} = \frac{2^{n-1}}{nN}$. Thus, we conclude

$$|\mathcal{H}| = \left|\bigcup_{i=1}^{N} \mathcal{H}_i\right| \leq \sum_{i=1}^{N} \frac{2^{n-1}}{nN} = N \cdot \frac{2^{n-1}}{nN} = \frac{2^{n-1}}{n},$$

which proves the claim. □

Finally, we prove a statement very close to the correctness of Algorithm 6. It asserts that for any low-ranked set S, the rankings w. r. t. all $\varrho_i(\cdot)$'s are in the desired range with high probability.

Lemma 4.20. *Let $c > 0$. If $S \subseteq E$ is low-ranked and $m \geq 2nN\,(c \log n + \log N)$ samples are taken to estimate $\widehat{p}_i$, then $\mathbb{P}\left[\forall i : \varrho_i(S) \leq \widehat{p}_i\right] \geq 1 - \frac{1}{n^c}$.*

Proof. Fix $c > 0$ and a low-ranked set $S \subseteq E$. Consider the failure probability $\mathbb{P}\left[\exists i : \varrho_i(S) > \widehat{p}_i\right]$, which by the union bound and Lemma 4.18 part (ii) is at most

$$\mathbb{P}\left[\exists i : \varrho_i(S) > \widehat{p}_i\right] \leq \sum_{i=1}^{N} \mathbb{P}\left[\varrho_i(S) > \widehat{p}_i\right] \leq \sum_{i=1}^{N} r_i(S)^m.$$

Since S is low-ranked, we know that $r_i(S) < 1 - \frac{1}{2nN}$ for each $1 \leq i \leq N$. Hence,

$$\begin{aligned} r_i(S)^m \leq \left(1 - \frac{1}{2nN}\right)^m &\leq \left(1 - \frac{1}{2nN}\right)^{2nN(c \log n + \log N)} \\ &\leq \left[\left(1 - \frac{1}{2nN}\right)^{2nN}\right]^{c \log n + \log N} \leq (1/e)^{c \log n + \log N} = n^{-c} N^{-1}. \end{aligned}$$

Summing over $1 \leq i \leq N$, we obtain $\sum_{i=1}^{N} r_i(S)^m \leq n^{-c}$. Thus, for the success event,

$$\mathbb{P}\left[\forall i : \varrho_i(S) \leq \widehat{p}_i\right] \geq 1 - \sum_{i=1}^{N} r_i(S)^m \geq 1 - n^{-c} = 1 - \frac{1}{n^c},$$

as desired. □

Lemma 4.21. *Let $n \geq 2, c > 0$. Choosing $m \geq 2nN\left((c+2)\log n + \log N\right)$, the family*

$$\mathcal{S} := \{S \subseteq E \mid \forall i : \varrho_i(S) \leq \widehat{p}_i\}$$

consists of $|\mathcal{S}| \geq \left(1 - \frac{1}{n}\right) 2^n$ sets with probability $\geq 1 - \frac{1}{n^c}$.

Proof. For each $S \subseteq E$, define an indicator random variable

$$\mathbb{1}_S := \begin{cases} 1 & \text{if } S \in \mathcal{S}, \\ 0 & \text{if } S \notin \mathcal{S} \end{cases}$$

encoding membership of $\mathcal{S}$. Then the indicator random variables $I_S := 1 - \mathbb{1}_S$ encode non-membership of $\mathcal{S}$. For any low-ranked $S \subseteq E$, we have

$$\begin{aligned} \mathbb{E}\left[I_S\right] &= \mathbb{E}\left[1 - \mathbb{1}_S\right] = \mathbb{P}\left[S \notin \mathcal{S}\right] = 1 - \mathbb{P}\left[S \in \mathcal{S}\right] \\ &= 1 - \mathbb{P}\left[\forall i : \varrho_i(S) \leq \widehat{p}_i\right] \leq 1 - \left(1 - \frac{1}{n^{c+2}}\right) = \frac{1}{n^{c+2}} \end{aligned}$$

by Lemma 4.20. Letting $\mathcal{L} \subseteq 2^E$ denote the family of low-ranked sets, we have

$$\mathbb{E}\left[\sum_{S \in \mathcal{L}} I_S\right] = \sum_{S \in \mathcal{L}} \mathbb{E}\left[I_S\right] \leq \frac{|\mathcal{L}|}{n^{c+2}} \leq \frac{2^n}{n^{c+2}}.$$

Thus, by Markov's inequality,

$$\mathbb{P}\left[\sum_{S \in \mathcal{L}} I_S \geq \frac{2^{n-1}}{n}\right] \leq \frac{\mathbb{E}\left[\sum_{S \in \mathcal{L}} I_S\right]}{2^{n-1}/n} \leq \frac{2^n}{n^{c+2}} \frac{n}{2^{n-1}} = \frac{2}{n^{c+1}} \leq \frac{1}{n^c}.$$

Overall, if the event $\left\{\sum_{S \in \mathcal{L}} I_S \geq \frac{2^{n-1}}{n}\right\}$ does *not* occur, so $\sum_{S \in \mathcal{L}} I_S < \frac{2^{n-1}}{n}$, we conclude that

$$\begin{aligned} |\mathcal{S}| &= \sum_{S \subseteq E} \mathbb{1}_S \geq \sum_{S \in \mathcal{L}} \mathbb{1}_S = \sum_{S \in \mathcal{L}} (1 - I_S) \\ &= |\mathcal{L}| - \sum_{S \in \mathcal{L}} I_S > \left(1 - \frac{1}{2n}\right) 2^n - \frac{2^{n-1}}{n} = \left(1 - \frac{1}{n}\right) 2^n \end{aligned}$$

by Lemma 4.19, $\mathcal{L} \subseteq 2^E$ and $\sum_{S \in \mathcal{L}} I_S < \frac{2^{n-1}}{n}$. Since $\left\{\sum_{S \in \mathcal{L}} I_S \geq \frac{2^{n-1}}{n}\right\}$ does not occur with high probability, we get the bound $|\mathcal{S}| \geq \left(1 - \frac{1}{n}\right) 2^n$ with high probability, too. □

Proof of Theorem 4.17. Part (ii) we have shown earlier. Part (i) essentially follows from Lemma 4.21. Since Algorithm 6 uses

$$m = \lceil 2nN\left((c+2)\log n + \log N\right)\rceil \geq 2nN\left((c+2)\log n + \log N\right),$$

the set $\mathcal{S} = \{S \subseteq E \mid \forall i : \varrho_i(S) \leq \widehat{p}_i\}$ satisfies $\mathbb{P}\left[|\mathcal{S}| \geq \left(1 - \frac{1}{n}\right) 2^n\right] \geq 1 - n^{-c}$. Following the proof of Theorem 3.1, we get that

$$(1-\varepsilon)F(S) \leq F'(S) \leq (1+\varepsilon)F(S)$$

for all $S \in \mathcal{S}$. Thus, with probability at least $1 - n^{-c}$, there are $\left(1 - \frac{1}{n}\right) 2^n$ many sets satisfying the above inequality. Hence, w is a relaxed ε-sparsifier. □

Chapter 5

Generalised Submodular Sparsification

This short chapter transfers results for submodular functions of bounded arity from Section 4.2 to more general classes of functions, such as the well-studied k-submodular functions [WŽ16; KTŽ15; FI05; MO21]. Section 5.1 is a technical introduction to k-submodular functions, featuring definitions and basic properties. In Section 5.2, we generalise the bounded arity special case to k-submodular functions and beyond.

Contributions:

- Introduction *arity reducible classes*, a property that captures the closure of a class of set functions under the formation of auxiliary functions.
- $\vec{\alpha}$-bisubmodular functions of bounded arity admit the construction of a small sparsifier in polynomial time. This includes bisubmodular functions.
- Sparsifier construction for k-submodular functions of bounded arity that scales in its running time with the fastest known algorithms for (approximate) k-submodular minimisation.

5.1 Introducing K-Submodular Functions

The idea of k-submodular functions is having a function $f(S_1, \ldots, S_k)$ of k variables defined on the domain of disjoint subsets of the ground set E that admits a pair $(\sqcup, \sqcap)$ of appropriate operations such that

- $k = 1$ corresponds to submodular functions, and
- useful properties such as the diminishing returns property are preserved.

In this context, we say that a function $f : \mathcal{D} \to \mathbb{R}$ *admits* a pair $(\sqcup, \sqcap)$ of binary operations on $\mathcal{D}$ if

$$f(A) + f(B) \geq f(A \sqcup B) + f(A \sqcap B)$$

for all $A, B \in \mathcal{D}$. It turns out that this kind of generalisation respecting the above properties does actually exist. We introduce it in this section and establish basic properties that will come in handy later.

To ease notation, we use bold letters for entire k-tuples $\mathbf{A} = (A_1, \ldots, A_k) \in (k+1)^E$ of disjoint sets $A_1, \ldots, A_k$. The notation $(k+1)^E$ coinincides with the common interpretation as the set of functions $E \to \{0, 1, \ldots, k\}$ on purpose: Take a function $f : E \to \{0, 1, \ldots, k\}$. Each element $e \in E$ is mapped to the index of the argument it belongs to, i. e., $e \in A_{f(e)}$. In other words, $f(e)$ encodes the set it belongs to. In this context, $f(e) = 0$ means e does not occur in any of the sets $A_1, \ldots, A_k$. The one-to-one correspondence with the family of all k-tuples $(A_1, \ldots, A_k)$ of disjoint sets becomes obvious here. Moreover, we define component-wise set operations

$$\mathbf{A} \cup \mathbf{B} := (A_1 \cup B_1, \ldots, A_k \cup B_k) \qquad \mathbf{A} \cap \mathbf{B} := (A_1 \cap B_1, \ldots, A_k \cap B_k) \tag{5.1}$$

and the partial order $\mathbf{A} \subseteq \mathbf{B} := \forall i \in \{1, \ldots, k\} : A_i \subseteq B_i$, which is defined component-wise, too, for all $\mathbf{A}, \mathbf{B} \in (k+1)^E$.

Now we can define k-submodularity. Following [WŽ16], a function $f : (k+1)^E \to \mathbb{R}$ is *k-submodular* if it admits the operations $(\sqcup, \sqcap)$ defined by

$$\mathbf{A} \sqcup \mathbf{B} := \left((A_1 \cup B_1) \setminus \bigcup_{i \neq 1} (A_i \cup B_i), \ldots, (A_k \cup B_k) \setminus \bigcup_{i \neq k} (A_i \cup B_i) \right)$$
$$\mathbf{A} \sqcap \mathbf{B} := (A_1 \cap B_1, \ldots, A_k \cap B_k)$$

for all $\mathbf{A}, \mathbf{B} \in (k+1)^E$. There are equivalent characterisations, for instance via the old notion of submodularity and a monotonicity property; see [WŽ16] for a detailed discussion.

5.2 K-Set Functions of Bounded Arity

In Section 4.2, we have seen how to construct an ε-sparsifier of small size (matching the existence result from the core algorithm and Theorem 3.1) in polynomial time for any decomposable submodular function F with constituents of constant arity. The perhaps most natural question asks for the generalisation to k-submodular functions. However, a closer examination of Algorithm 5 reveals that it only exploits the submodularity of the input functions to conclude that the auxiliary function $\widehat{F}$ is minimisable in polynomial time. Apart from that, it works for any decomposable set function $F : 2^E \to \mathbb{R}$ of bounded arity. In this section, we build upon this observation in order to

- introduce the *arity reducible classes* which capture the property of a class of k-set functions (such as k-submodular functions) of being closed under forming auxiliary functions $\widehat{F}$,
- establish that k-submodular functions form an arity reducible class,
- establish that $\vec{\alpha}$-bisubmodular functions [FTY14] (a generalisation of skew bisubmodular functions [HKP14]) form an arity reducible class,

hence it is possible for any decomposable k-submodular or $\vec{\alpha}$-bisubmodular function F with constituents of constant arity to construct an ε-sparsifier of small size with a polynomial number of minimisation oracle calls. Since $\vec{\alpha}$-bisubmodular functions admit a polynomial-time minimisation algorithm [FTY14], the aforementioned sparsifier construction can be implemented in polynomial time for them, including the well-known bisubmodular functions.

The main work in this section revolves around computing the peak contributions of the constituents of F, the decomposable k-set function under consideration. As before, an ε-sparsifier with the guarantees stated in Corollary 3.4 is easily obtained by Algorithm 1 once the peak contributions are known.

5.2.1 Notions of Arity and Reducibility

The definition of arity can be generalised to k-set functions. We say that a k-set function $f : (k+1)^E \to \mathbb{R}$ has *arity* a if there is a set $C \subseteq E$ of size $|C| = a$ such that

$$f(\mathbf{S}) = f(S_1 \cap C, \ldots, S_k \cap C) \tag{5.2}$$

for all $\mathbf{S} \in (k+1)^E$. As before, we call C the *effective support* of f.

Remark 5.1. If one is interested in the more general situation of a k-set function $f : (k+1)^E \to \mathbb{R}$ with argument-specific supports $C_1, \ldots, C_k \subseteq E$, i. e.

$$f(\mathbf{S}) = f(S_1 \cap C_1, \ldots, S_k \cap C_k),$$

of sizes $|C_i| = a_i$, the above definition is still useful. Letting $C := \bigcup_{i=1}^{k} C_i$, we find that C is an effective support of f of size $|C| \leq a_1 + \cdots + a_k$, i. e., f has arity at most $a_1 + \cdots + a_k$. In particular, if $a_1, \ldots, a_k = \mathcal{O}(1)$, we know that f has constant arity. Thus, we will work with the notion of arity as it is captured by Eq. (5.2) above, since the more general notion of argument-specific supports can be reduced to it.

Having decided on a notion of arity, we can define the family of arity reducible classes. A class

$$\mathcal{C} \subseteq \bigcup_{E \text{ finite set}} \mathbb{R}^{(k+1)^E} \tag{5.3}$$

of k-set functions[1] is *arity reducible* if, for any finite set E, function $F : (k+1)^E \to \mathbb{R}$ with $F \in \mathcal{C}$, set $C \subseteq E$ and disjoint sets $H_1, \ldots, H_k \subseteq C$, the "reduced" function

$$\widehat{F} : (k+1)^{E \setminus C} \to \mathbb{R}, \quad (A_1, \ldots, A_k) \mapsto (A_1 \cup H_1, \ldots, A_k \cup H_k)$$

belongs to $\mathcal{C}$ as well, i. e., $\widehat{F} \in \mathcal{C}$.

Since our entire work is oblivious to the concrete elements of E, we may assume that $E = \{1, \ldots, n\}$ for some integer $n \geq 1$. This is accomplished by labelling the finitely many elements of E in some arbitrary order. Eq. (5.3) then becomes

$$\mathcal{C} \subseteq \bigcup_{n \geq 1} \mathbb{R}^{(k+1)^{\{1,\ldots,n\}}},$$

which might seem a bit more comprehensible.

5.2.2 Peak Contributions For Arity Reducible Classes

Fix $k \geq 1$ and an arity reducible class $\mathcal{C}$. Given the ability to minimise functions from $\mathcal{C}$, we can compute peak contributions for all constituents of a decomposable function F if they have bounded arity. This is exactly what Algorithm 5 accomplishes for the special case of submodular functions. Notation-wise, we let $\mathrm{MIN}(\cdot)$ denote a minimisation oracle for functions from the class $\mathcal{C}$, i. e., for any $f : (k+1)^E \to \mathbb{R}$ with $f \in \mathcal{C}$, $\mathrm{MIN}(f) := \arg\min_{\mathbf{A} \in (k+1)^E} f(\mathbf{A})$ denotes a minimiser of f. The peak contributions are then computed by Algorithm 7, which invokes $\mathrm{MIN}(\cdot)$.

The correctness and efficiency of Algorithm 7 is established by the following theorem.

[1] For instance, the class of all k-submodular functions.

Algorithm 7 Computing Peak Contributions For Arity Reducible Classes

Input: Decomposable $F = f_1 + \cdots + f_N : (k+1)^E \to \mathbb{R}$ with $F \in \mathcal{C}$ and f_i's of bounded arity with effective supports C_i; minimisation oracle $\mathrm{MIN}(\cdot)$ for k-set functions over E belonging to $\mathcal{C}$

Output: Peak contributions $p_i = \max_{\mathbf{A} \in (k+1)^E} \frac{f_i(\mathbf{A})}{F(\mathbf{A})}$ for $1 \leq i \leq N$

1: **for** $i = 1$ **to** N **do**
2: $\quad p_i \leftarrow 0$
3: $\quad$ **for all** $\mathbf{H} \in (k+1)^C$ **do**
4: $\quad\quad$ Define $\widehat{F} : (k+1)^{E \setminus C} \to \mathbb{R}, \quad \mathbf{A} \mapsto F(\mathbf{A} \cup \mathbf{H})$.
5: $\quad\quad \mathbf{A}^* \leftarrow \mathrm{MIN}(\widehat{F})$
6: $\quad\quad p_i \leftarrow \max\left\{p_i, \frac{f_i(\mathbf{H})}{\widehat{F}(\mathbf{A}^*)}\right\}$
7: $\quad$ **end for**
8: **end for**
9: **return** $(p_1, \ldots, p_N)$

Theorem 5.2. *Let $F : (k+1)^E \to \mathbb{R}$ denote a decomposable k-set function with $F \in \mathcal{C}$ that can be decomposed as $F = f_1 + \cdots + f_N$ such that each f_i has effective support C_i of size $|C_i| \leq a \in \mathcal{O}(1)$. Then Algorithm 7*

(i) correctly computes all peak contributions,

(ii) runs in time $\mathcal{O}\left(N \cdot \mathsf{MIN}(n) \cdot \mathsf{EO}(F)\right)$, where $\mathsf{MIN}(n)$ denotes the maximum number of evaluation oracle calls the minimisation oracle $\mathrm{MIN}(\cdot)$ takes to minimise a function $f \in \mathcal{C}$ on any ground set of size $\leq n$. Moreover, $\mathsf{EO}(F)$ is the (maximum) time it takes to invoke the evaluation oracle of F once.

Proof. We prove correctness first. Note that

$$\begin{aligned}
p_i &= \max_{\mathbf{A} \in (k+1)^E} \frac{f_i(\mathbf{A})}{F(\mathbf{A})} \\
&= \max_{\mathbf{A} \in (k+1)^E} \frac{f_i(A_1 \cap C, \ldots, A_k \cap C)}{F(\mathbf{A})} \\
&= \max_{\mathbf{H} \in (k+1)^C} \max_{\substack{\mathbf{A} \in (k+1)^E \\ \forall j : A_j \cap C = H_j}} \frac{f_i(H_1, \ldots, H_k)}{F(\mathbf{A})} \\
&= \max_{\mathbf{H} \in (k+1)^C} f_i(\mathbf{H}) \max_{\mathbf{A} \in (k+1)^{E \setminus C}} \frac{1}{F(\mathbf{A} \cup \mathbf{H})} \\
&= \max_{\mathbf{H} \in (k+1)^C} \frac{f_i(\mathbf{H})}{\min_{\mathbf{A} \in (k+1)^{E \setminus C}} F(\mathbf{A} \cup \mathbf{H})} \\
&= \max_{\mathbf{H} \in (k+1)^C} \frac{f_i(\mathbf{H})}{\min_{\mathbf{A} \in (k+1)^{E \setminus C}} \widehat{F}(\mathbf{A})},
\end{aligned}$$

where the last expression is exactly what Algorithm 7 evaluates. The denominator is computed by the invocation $\mathrm{MIN}(\widehat{F})$ of the minimisation oracle, which is possible because $\widehat{F} \in \mathcal{C}$ by the arity reducibility of $\mathcal{C}$. Notice that the **for all**-loop implements the maximisation over all $\mathbf{H} \in (k+1)^C$. This shows part (i).

For part (ii), consider the two nested **for**-loops. The outer **for**-loop runs for N iterations, the inner one always for $|(k+1)^C| \leq (k+1)^a = \mathcal{O}(1)$ iterations, as $a = \mathcal{O}(1)$ and k is constant, too. In each iteration of the inner **for**-loop, there is one call to the minimisation oracle, which takes at most $\mathsf{MIN}(n)$ invocations of the evaluation oracle for $\widehat{F}$. Each call to the evaluation oracle for $\widehat{F}$ can be implemented by one call to the evaluation oracle for F by invoking it on $\mathbf{A} \cup \mathbf{H}$, if the call was requesting $\widehat{F}(\mathbf{A})$. Since the evaluation oracle of F takes time at most $\mathsf{EO}(F)$ per call, the desired time bound follows. □

Remark 5.3. We should stress here that the proof of Theorem 5.2 crucially relies on $\mathcal{C}$ being an arity reducible class. Algorithm 7 invokes the minimisation oracle for $\mathcal{C}$ on the auxiliary function $\widehat{F}$. If $\widehat{F}$ was not a member of $\mathcal{C}$, this step would not be possible. As we will see, the setting of a class $\mathcal{C}$ with a minimisation oracle given for all members of $\mathcal{C}$ is a very natural one. Examples include bisubmodular functions, $\vec{\alpha}$-bisubmodular functions and – potentially – k-submodular functions for $k \geq 3$, which is an open question.

Given Theorem 5.2, it only remains to execute the core algorithm in order to obtain an ε-sparsifier for the kind of function Theorem 5.2 applies to.

Corollary 5.4. *Let $F : (k+1)^E \to \mathbb{R}$ denote a decomposable k-set function with $F \in \mathcal{C}$ that can be decomposed as $F = f_1 + \cdots + f_N$ such that each f_i has effective support C_i of size $|C_i| \leq a \in \mathcal{O}(1)$. Then, for any $\varepsilon, \delta > 0$, Algorithm 1 instantiated with peak contributions from Algorithm 7 outputs an ε-sparsifier with probability at least $1 - \delta$. Moreover, the expected size of the ε-sparsifier is in $\mathcal{O}\left(\frac{n+\log\frac{1}{\delta}}{\varepsilon^2}\sum_{i=1}^{n} p_i\right)$, where $n = |E|$. The running time is polynomial if $\mathsf{MIN}(n)$ is and all f_i have polynomial-time evaluation oracles.*

Proof. Execute Algorithm 7 and then Algorithm 1 with the modification that the p_i's are taken from the output of Algorithm 7. The correctness and claims about the running time follow from Theorem 5.2 and Corollary 3.4. □

We have now designed a framework that easily allows us to prove the existence of polynomial-time sparsifier constructions for functions decomposable into constituents

of bounded arity. For any class $\mathcal{C}$ of interest, we have to check whether (a) it is arity reducible, and (b) it admits a polynomial-time minimisation algorithm.

5.2.3 K-Submodular Functions

We apply our framework to k-submodular functions by proving that they form an arity reducible class. Recall that it is an open problem whether k-submodular functions can be minimised by a polynomial number of evaluation oracle calls. However, for the cases $k = 1$ (submodular functions) and $k = 2$ (bisubmodular functions), this question can be answered in the affirmative [GLS81; FI05]. It makes sense for us to look at k-submodular functions for general k, since Corollary 5.4 is formulated under the premise of a minimisation oracle. Thus, if it turns out that, for $k \geq 3$, it is indeed possible to minimise with polynomially many oracle calls, we immediately get a sparsifier construction in polynomial time. In particular, this works for k-submodular functions with additional properties, for which we might already know how to minimise them efficiently. Last but not least, even if the minimisation oracle at hand takes a superpolynomial number of oracle calls, the running time may still be good enough.

It remains to prove that k-submodular functions form an arity reducible class.

Theorem 5.5. *Let $k \geq 1$. The class of k-submodular functions is arity reducible.*

Proof. Consider any finite ground set E, let $F : (k+1)^E \to \mathbb{R}$ be k-submodular. Furthermore, let $C \subseteq E$ and $\mathbf{H} \in (k+1)^C$. We have to show that

$$\widehat{F} : (k+1)^{E \setminus C} \to \mathbb{R}, \quad \mathbf{A} \mapsto F(\mathbf{A} \cup \mathbf{H})$$

is k-submodular, too.

We verify that $\widehat{F}$ admits $(\sqcup, \sqcap)$. Consider arbitrary $\mathbf{S}, \mathbf{T} \in (k+1)^{E \setminus C}$. We simplify the expressions for $\widehat{F}(\mathbf{S} \sqcap \mathbf{T})$ and $\widehat{F}(\mathbf{S} \sqcup \mathbf{T})$ in order to show the k-submodularity inequality $\widehat{F}(\mathbf{S}) + \widehat{F}(\mathbf{T}) \geq \widehat{F}(\mathbf{S} \sqcup \mathbf{T}) + \widehat{F}(\mathbf{S} \sqcap \mathbf{T})$ in a second step.

The term $\widehat{F}(\mathbf{S} \sqcap \mathbf{T})$ can be transformed as

$$\begin{aligned} \widehat{F}(\mathbf{S} \sqcap \mathbf{T}) &= \widehat{F}(S_1 \cap T_1, \ldots, S_k \cap T_k) \\ &= F((S_1 \cap T_1) \cup H_1, \ldots, (S_k \cap T_k) \cup H_k) \\ &= F((S_1 \cup H_1) \cap (T_1 \cup H_1), \ldots, (S_k \cup H_k) \cap (T_k \cup H_k)) \\ &= F((\mathbf{S} \cup \mathbf{H}) \sqcap (\mathbf{T} \cup \mathbf{H})) \end{aligned} \tag{5.4}$$

by the definition of $\widehat{F}$ and the De Morgan law from set theory.

The term $\widehat{F}(\mathbf{S} \sqcup \mathbf{T})$ is a bit more complex, making it helpful to consider each argument separately. Fix any $1 \leq i \leq k$. Notice that

$$\begin{aligned}\left((S_i \cup T_i) \setminus \bigcup_{j \neq i}(S_j \cup T_j)\right) \cup H_i &\overset{(1)}{=} (S_i \cup T_i \cup H_i) \setminus \bigcup_{j \neq i}(S_j \cup T_j) \\ &\overset{(2)}{=} ((S_i \cup H_i) \cup (T_i \cup H_i)) \setminus \bigcup_{j \neq i}(S_j \cup T_j) \\ &\overset{(3)}{=} ((S_i \cup H_i) \cup (T_i \cup H_i)) \setminus \bigcup_{j \neq i}((S_j \cup H_j) \cup (T_j \cup H_j))\end{aligned}$$

where step (1) follows because H_i is disjoint from $\bigcup_{j \neq i}(S_i \cup T_i)$ (as $H_i \subseteq C$ but $S_i, T_i \subseteq E \setminus C$), step (2) from elementary set theoretic laws, and step (3) from the fact that the H_i's are disjoint, so no element added by H_i can be removed by adding the H_j's for $j \neq i$ to the $\setminus$ operation. Applying the above transformation to each argument, we conclude that $\widehat{F}(\mathbf{S} \sqcup \mathbf{T}) = F((\mathbf{S} \cup \mathbf{H}) \sqcup (\mathbf{T} \cup \mathbf{H}))$.

Combining Eq. (5.4) and $\widehat{F}(\mathbf{S} \sqcup \mathbf{T}) = F((\mathbf{S} \cup \mathbf{H}) \sqcup (\mathbf{T} \cup \mathbf{H}))$ with the k-submodularity of F, we obtain

$$\begin{aligned}\widehat{F}(\mathbf{S} \sqcup \mathbf{T}) + \widehat{F}(\mathbf{S} \sqcap \mathbf{T}) &= F((\mathbf{S} \cup \mathbf{H}) \sqcup (\mathbf{T} \cup \mathbf{H})) + F((\mathbf{S} \cup \mathbf{H}) \sqcap (\mathbf{T} \cup \mathbf{H})) \\ &\leq F(\mathbf{S} \cup \mathbf{H}) + F(\mathbf{T} \cup \mathbf{H}) \\ &= \widehat{F}(\mathbf{S}) + \widehat{F}(\mathbf{T})\end{aligned}$$

proving that $\widehat{F}$ is indeed k-submodular. □

5.2.4 Generalised Skew Bisubmodular Functions

It turns out that we can extend the sparsification result for bisubmodular functions to skew bisubmodular [HKP14] and even $\vec{\alpha}$-bisubmodular functions [FTY14], see below.

The motivation for generalised notions of bisubmodularity originates from the study of Valued Constraint Satisfaction Problems (VCSPs). It can be shown [HKP14; TŽ16] that submodularity and skew bisubmodilarity characterise (under $\mathsf{P} \neq \mathsf{NP}$) computationally tractable VCSPs where the variables are allowed take three values. If such a VCSP is not of this form, the NP-hard MAX-CUT problem can be expressed.

As introduced in [HKP14], a function $F : 3^E \to \mathbb{R}$ is *α-bisubmodular* if

$$F(\mathbf{X}) + F(\mathbf{Y}) \geq F(\mathbf{X} \sqcap \mathbf{Y}) + \alpha \cdot F(\mathbf{X} \cup_0 \mathbf{Y}) + (1 - \alpha) \cdot F(\mathbf{X} \cup_1 \mathbf{Y})$$

for all $\mathbf{X} = (X_+, X_-), \mathbf{Y} = (Y_+, Y_-) \in 3^E$, where

$$\mathbf{X} \cup_0 \mathbf{Y} := ((X_+ \cup Y_+) \setminus (X_- \cup Y_-), (X_- \cup Y_-) \setminus (X_+ \cup Y_+)),$$
$$\mathbf{X} \cup_1 \mathbf{Y} := (X_+ \cup Y_+, (X_- \cup Y_-) \setminus (X_+ \cup Y_+)).$$

If F is α-bisubmodular for some $0 \leq \alpha < 1$, we call F *skew bisubmodular*. Notice that 1-bisubmodular functions resemble exactly the bisubmodular functions. Huber and Krokhin [HK14] constructed a convex extension akin to the Lovász extension in the submodular case in order to show that skew bisubmodular functions can be minimised by a polynomial number of evaluation oracle calls. This motivates the hunt for more general settings that admit the same approach. Indeed, Fujishige, Tanigawa and Yoshida [FTY14] found a natural generalisation of skew bisubmodularity, the $\vec{\alpha}$-bisubmodularity for a pair $\vec{\alpha} = (\vec{\alpha_+}, \vec{\alpha_-})$ of vectors $\vec{\alpha_+}, \vec{\alpha_-} \in \mathbb{R}^E_{>0}$, that admits an analogue of the Lovász extension over n-dimensional rectangles. Hence, $\vec{\alpha}$-bisubmodular functions can be minimised by a polynomial number of evaluation oracle calls as well.

The formal definition presented in [FTY14] goes as follows. Fix a pair $\vec{\alpha} = (\vec{\alpha_+}, \vec{\alpha_-})$ of vectors $\vec{\alpha_+}, \vec{\alpha_-} \in \mathbb{R}^E_{>0}$ and, for each $0 < t < 1$, let $\mathbf{E_t} = (E_{t,+}, E_{t,-}) \in 3^E$ with

$$E_{t,+} := \left\{ e \in E \mid \frac{\vec{\alpha_-}(e)}{\vec{\alpha_+}(e)} \leq t \right\} \quad \text{and} \quad E_{t,-} := \left\{ e \in E \mid \frac{\vec{\alpha_+}(e)}{\vec{\alpha_-}(e)} \leq t \right\}.$$

Moreover, we define a binary operation $\cup_t$ on 3^E which is a variant of the union operation $\cup$ on 3^E as defined in Eq. (5.1). Let $\mathbf{X} \cup_t \mathbf{Y} := ((\mathbf{X} \cup_t \mathbf{Y})_+, (\mathbf{X} \cup_t \mathbf{Y})_-)$, where

$$(\mathbf{X} \cup_t \mathbf{Y})_+ := (\mathbf{X} \cup_0 \mathbf{Y})_+ \cup (E_{t,+} \cap (X_+ \cup Y_+) \cap (X_- \cup Y_-)),$$
$$(\mathbf{X} \cup_t \mathbf{Y})_- := (\mathbf{X} \cup_0 \mathbf{Y})_- \cup (E_{t,-} \cap (X_+ \cup Y_+) \cap (X_- \cup Y_-)).$$

Finally, let $0 = t_0 < t_1 < \cdots < t_{k+1} = 1$ denote the distinct elements of the set $\left\{ \min\left\{ \frac{\vec{\alpha_-}(e)}{\vec{\alpha_+}(e)}, \frac{\vec{\alpha_+}(e)}{\vec{\alpha_-}(e)} \right\} \mid e \in E \right\} \cup \{0, 1\}$.

Now, a function $F : 3^E \to \mathbb{R}$ is called *$\vec{\alpha}$-bisubmodular*, if

$$F(\mathbf{X}) + F(\mathbf{Y}) \geq F(\mathbf{X} \cap \mathbf{Y}) + \sum_{i=0}^{k} (t_{i+1} - t_i) F(\mathbf{X} \cup_{t_i} \mathbf{Y}) \tag{5.5}$$

for all $\mathbf{X}, \mathbf{Y} \in 3^E$.

We address the sparsification question for $\vec{\alpha}$-bisubmodular functions by showing that they form an arity reducible class. In combination with the minimisation result, we conclude that $\vec{\alpha}$-bisubmodular functions of bounded arity can be sparsified efficiently.

Theorem 5.6. *For any $\vec{\alpha} = (\vec{\alpha_+}, \vec{\alpha_-})$ with $\vec{\alpha_+}, \vec{\alpha_-} \in \mathbb{R}^E_{>0}$, the class of $\vec{\alpha}$-bisubmodular functions is arity reducible.*

Proof. Fix such a pair of vectors $\vec{\alpha} = (\vec{\alpha_+}, \vec{\alpha_-})$ and let F be α-bisubmodular. Moreover, fix an effective support $C \subseteq E$ and $\mathbf{H} \in 3^C$. To establish Eq. (5.5) for the auxiliary function

$$\widehat{F} : 3^{E \setminus C} \to \mathbb{R}, \quad \mathbf{A} \mapsto F(\mathbf{A} \cup \mathbf{H}),$$

we also fix arbitrary $\mathbf{X}, \mathbf{Y} \in 3^E$. Let's decompose the RHS of Eq. (5.5) into its two parts. The identity

$$\widehat{F}(\mathbf{X} \cap \mathbf{Y}) = F((\mathbf{X} \cup \mathbf{H}) \cap (\mathbf{Y} \cup \mathbf{H})) \tag{5.6}$$

is derived by the exact same steps as in Eq. (5.4) because $\sqcap$ and the component-wise $\cap$ coinincide. To establish the identities

$$\widehat{F}(\mathbf{X} \cup_{t_i} \mathbf{Y}) = F((\mathbf{X} \cup \mathbf{H}) \cup_{t_i} (\mathbf{Y} \cup \mathbf{H})) \tag{5.7}$$

for each $0 \leq i \leq k$, we first show note that $\cup_0$ is the same as $\sqcup$ in the $k = 2$ case, hence we get $(\mathbf{X} \cup_0 \mathbf{Y}) \cup \mathbf{H} = (\mathbf{X} \cup \mathbf{H}) \cup_0 (\mathbf{Y} \cup \mathbf{H})$ by the same steps as we carried out in the proof of Theorem 5.5. In addition, we have

$$\begin{aligned}
&E_{t_i,+} \cap (X_+ \cup Y_+) \cap (X_- \cap Y_-) \\
&= E_{t_i,+} \cap (X_+ \cup Y_+ \cup H_+) \cap (X_- \cup Y_- \cup H_-) \\
&= E_{t_i,+} \cap ((X_+ \cup H_+) \cup (Y_+ \cup H_+)) \cap ((X_- \cup H_-) \cup (Y_- \cup H_-))
\end{aligned}$$

where the first step is by the fact that $H_+ \cap H_- = \varnothing$ and the second step is by the idempotency law. Putting this and $(\mathbf{X} \cup_0 \mathbf{Y}) \cup \mathbf{H} = (\mathbf{X} \cup \mathbf{H}) \cup_0 (\mathbf{Y} \cup \mathbf{H})$ together, we obtain

$$\begin{aligned}
&((\mathbf{X} \cup_{t_i} \mathbf{Y}) \cup \mathbf{H})_+ \\
&= (\mathbf{X} \cup_{t_i} \mathbf{Y})_+ \cup H_+ \\
&= \big((\mathbf{X} \cup_0 \mathbf{Y})_+ \cup (E_{t_i,+} \cap (X_+ \cup Y_+) \cap (X_- \cap Y_-))\big) \cup H_+ \\
&= \big((\mathbf{X} \cup_0 \mathbf{Y})_+ \cup H_+\big) \\
&\qquad \cup (E_{t_i,+} \cap ((X_+ \cup H_+) \cup (Y_+ \cup H_+)) \cap ((X_- \cup H_-) \cup (Y_- \cup H_-))) \\
&= ((\mathbf{X} \cup \mathbf{H}) \cup_0 (\mathbf{Y} \cup \mathbf{H}))_+ \\
&\qquad \cup (E_{t_i,+} \cap ((\mathbf{X} \cup \mathbf{H})_+ \cup (\mathbf{Y} \cup \mathbf{H})_+) \cap ((\mathbf{X} \cup \mathbf{H})_- \cup (\mathbf{Y} \cup \mathbf{H})_-)) .
\end{aligned}$$

An analogous proof reveals the same identity for the $-$ instead of the $+$ component. Therefore, $(\mathbf{X} \cup_{t_i} \mathbf{Y}) \cup \mathbf{H} = (\mathbf{X} \cup \mathbf{H}) \cup_{t_i} (\mathbf{Y} \cup \mathbf{H})$, implying Eq. (5.7). We can now finish the proof by stitching Eq. (5.6) and Eq. (5.7) together:

$$\begin{aligned}
\widehat{F}(\mathbf{X}) + \widehat{F}(\mathbf{Y}) &= F(\mathbf{X} \cup \mathbf{H}) + F(\mathbf{Y} \cup \mathbf{H}) \\
&\geq F((\mathbf{X} \cup \mathbf{H}) \cap (\mathbf{Y} \cup \mathbf{H})) + \sum_{i=0}^{k} (t_{i+1} - t_i)\, F((\mathbf{X} \cup \mathbf{H}) \cup_{t_i} (\mathbf{Y} \cup \mathbf{H})) \\
&= \widehat{F}(\mathbf{X} \cap \mathbf{Y}) + \sum_{i=0}^{k} (t_{i+1} - t_i)\, \widehat{F}(\mathbf{X} \cup_{t_i} \mathbf{Y})
\end{aligned}$$

The first step is by definition of $\widehat{F}$, the second by the $\vec{\alpha}$-bisubmodularity of F, and the last step by Eq. (5.6) and Eq. (5.7). It follows that $\widehat{F}$ is indeed $\vec{\alpha}$-bisubmodular, proving that $\vec{\alpha}$-bisubmodular functions form an arity reducible class. $\square$

Chapter 6

Conclusion

We have seen several new algorithms making the efficient construction of small sparsifiers possible mostly for but not limited to broad classes of submodular functions. All algorithms build upon the core algorithm – a framework inspired by Rafiey and Yoshida [RY22] that we introduced for the sparsification of general decomposable functions, extending well beyond submodularity. This framework can be seen as a universal sparsification tool that we specialised to several specific classes of functions in order to gain more efficiency and better size bounds.

Moreover, by providing appropriate counterexamples, we revealed flaws in the papers by Rafiey and Yoshida [RY22] and Perrault et al. [Per+21], the flaw in the latter being admitted by the authors now. Regarding the flaws in both papers, we were able to provide corrections under reasonable assumptions.

There are a lot more possible directions for future research than any conclusion could reasonably mention. In fact, the research on sparsification links strongly to other algorithmic problems related to submodularity all the way down to the fundamental questions of complexity theory such as the P vs NP question [Bai+16]. For instance, if we could minimise k-submodular functions in polynomial time, we were able to efficiently construct small sparsifiers for k-submodular functions of bounded arity, remember Section 5.2.

For these reasons, we suggest a selection of open questions that are closely related to the results in this work and would form an interesting extension of it.

(1) k-submodular peak contributions: If $F = f_1 + \cdots + f_N : 2^E \to \mathbb{R}_{\geq 0}$ with all f_i's monotone submodular, we showed that $\sum_{i=1}^{N} p_i \leq Bn$ in Lemma 3.9, where $B = \max_{1 \leq i \leq N} |\mathsf{EX}(\mathcal{B}(f_i))|$. It is not clear whether and how this bound can be generalised to k-submodular functions for $k \geq 2$. Particularly interesting is the question whether $\sum_{i=1}^{N} p_i = \mathcal{O}(n)$ if all f_i's have bounded arity. This is the case for $k = 1$ because $B \leq 2^{a^2}$ by Lemma 4.15.

(2) Ψ-maximisation for k-submodular functions: Matsuoka and Ohsaka [MO21] generalise the notion of curvature from submodular to k-submodular functions. Under this definition, can Algorithm 2 be adapted to solve the Ψ-maximisation problem for k-submodular functions? An obvious adaption would be to keep the same algorithm but select a maximiser

$$\left(e^{(\ell)}, i^{(\ell)}\right) = \arg \max_{(e,i) \in U \times \{1,\dots,k\}} \frac{\Delta_{e,i}\left(f \mid \mathbf{S}^{(\ell-1)}\right)}{\Delta_{e,i}\left(g \mid \mathbf{S}^{(\ell-1)}\right)}$$

in each step. This modified algorithm always returns feasible solutions. But how close are they to optimality?

(3) Ψ-maximisation under knapsack constraints: Can Algorithm 2 in [Per+21] be fixed without any additional assumptions? If it exists, can such a fix be adapted to k-submodular functions? In other words, can a similar algorithm as suggested in (2) solve k-submodular Ψ-maximisation under knapsack constraints?

(4) In [RY22], Rafiey and Yoshida establish a tightness result suggesting that size $\mathcal{O}\left(\frac{n+\log\frac{1}{\delta}}{\varepsilon^2}\sum_{i=1}^{N} p_i\right)$ is essentially the best we can expect of an ε-sparsifier of a submodular function in general. However, is it possible to obtain a better bound under a relaxed notion of sparsification? The relaxed ε-sparsifiers introduced in Section 4.3 could serve as a starting point here.

(5) Bounded arity: Which classes of set functions are arity reducible and minimisable in polynomial time? We got this for k-submodular functions with $k \in \{1, 2\}$ as well as $\vec{\alpha}$-bisubmodular functions. What about set functions that are not k-submodular, e. g. functions satisfying a weaker notion of k-submodularity?

Solving any of these questions would not only lead to new sparsification results but also potentially reveal improvements in various applications, from machine learning to combinatorial optimisation.

Bibliography

[AMO93] Ravindra K. Ahuja, Thomas L Magnanti, and James B. Orlin. *Network Flows. Theory, Algorithms, and Applications*. eng. Englewood Cliffs, N.J: Prentice Hall, 1993. ISBN: 978-0-13-617549-0.

[Bai+16] Wenruo Bai et al. "Algorithms for Optimizing the Ratio of Submodular Functions". In: *Proceedings of The 33rd International Conference on Machine Learning*. Ed. by Maria Florina Balcan and Kilian Q. Weinberger. Vol. 48. Proceedings of Machine Learning Research. New York, New York, USA: PMLR, 20–22 Jun 2016, pp. 2751–2759. URL: https://proceedings.mlr.press/v48/baib16.html.

[BK96] András A. Benczúr and David R. Karger. "Approximating s-t Minimum Cuts in $\tilde{O}(n^2)$ Time". In: *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*. Ed. by Gary L. Miller. ACM, 1996, pp. 47–55. DOI: 10.1145/237814.237827.

[CKN20] Yu Chen, Sanjeev Khanna, and Ansh Nagda. "Near-linear Size Hypergraph Cut Sparsifiers". In: *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*. 2020, pp. 61–72. DOI: 10.1109/FOCS46700.2020.00015.

[Coh+17] Michael B. Cohen et al. "Almost-Linear-Time Algorithms for Markov Chains and New Spectral Primitives for Directed Graphs". In: *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*. STOC 2017. Montreal, Canada: Association for Computing Machinery, 2017, pp. 410–419. ISBN: 9781450345286. DOI: 10.1145/3055399.3055463.

[Cor+09] Thomas H. Cormen et al. *Introduction to Algorithms, 3rd Edition*. MIT Press, 2009. ISBN: 978-0-262-03384-8.

[FI05] Satoru Fujishige and Satoru Iwata. "Bisubmodular Function Minimization". In: *SIAM Journal on Discrete Mathematics* 19.4 (2005), pp. 1065–1073. DOI: 10.1137/S0895480103426339.

[FTY14] Satoru Fujishige, Shin-ichi Tanigawa, and Yuichi Yoshida. "Generalized skew bisubmodularity: A characterization and a min–max theorem". In: *Discrete Optimization* 12 (2014), pp. 1–9. ISSN: 1572-5286. DOI: https://doi.org/10.1016/j.disopt.2013.12.001.

[Fuj91] "Submodular Systems and Base Polyhedra". In: *Submodular Functions and Optimization*. Ed. by Satoru Fujishige. Vol. 47. Annals of Discrete Mathematics. Elsevier, 1991, pp. 17–108. DOI: https://doi.org/10.1016/S0167-5060(08)70127-4.

[GLS81] Martin Grötschel, Lovász László, and Alexander Schrijver. "The Ellipsoid Method and its Consequences in Combinatorial Optimization". In: *Combinatorica* 1 (June 1981), pp. 169–197. DOI: 10.1007/BF02579273.

[HK14] Anna Huber and Andrei Krokhin. "Oracle Tractability of Skew Bisubmodular Functions". In: *SIAM Journal on Discrete Mathematics* 28.4 (2014), pp. 1828–1837. DOI: 10.1137/130936038.

[HK73] John E. Hopcroft and Richard M. Karp. "An $n^{5/2}$ Algorithm for Maximum Matchings in Bipartite Graphs". In: *SIAM J. Comput.* 2 (1973), pp. 225–231. DOI: https://doi.org/10.1137/0202019.

[HKP14] Anna Huber, Andrei Krokhin, and Robert Powell. "Skew Bisubmodularity and Valued CSPs". In: *SIAM Journal on Computing* 43.3 (May 2014), pp. 1064–1084. DOI: 10.1137/120893549.

[IFF01] Satoru Iwata, Lisa Fleischer, and Satoru Fujishige. "A Combinatorial Strongly Polynomial Algorithm for Minimizing Submodular Functions". In: *J. ACM* 48.4 (July 2001), pp. 761–777. ISSN: 0004-5411. DOI: 10.1145/502090.502096.

[IP01] Russell Impagliazzo and Ramamohan Paturi. "On the Complexity of k-SAT". In: *Journal of Computer and System Sciences* 62.2 (2001), pp. 367–375. ISSN: 0022-0000. DOI: https://doi.org/10.1006/jcss.2000.1727.

[KK15] Dmitry Kogan and Robert Krauthgamer. "Sketching Cuts in Graphs and Hypergraphs". In: *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science*. ITCS '15. Rehovot, Israel: Association for Computing Machinery, 2015, pp. 367–376. ISBN: 9781450333337. DOI: 10.1145/2688073.2688093.

[KTŽ15] Vladimir Kolmogorov, Johan Thapper, and Stanislav Živný. "The Power of Linear Programming for General-Valued CSPs". In: *SIAM Journal on Computing* 44.1 (2015), pp. 1–36. DOI: 10.1137/130945648.

[LS18] Yin Tat Lee and He Sun. "Constructing Linear-Sized Spectral Sparsification in Almost-Linear Time". In: *SIAM Journal on Computing* 47.6 (2018), pp. 2315–2336. DOI: 10.1137/16M1061850.

[MO21] Tatsuya Matsuoka and Naoto Ohsaka. "Maximization of Monotone k-Submodular Functions with Bounded Curvature and Non-k-Submodular Functions". In: *Proceedings of The 13th Asian Conference on Machine Learning*. Ed. by Vineeth N. Balasubramanian and Ivor Tsang. Vol. 157. Proceedings of Machine Learning Research. PMLR, 17–19 Nov 2021, pp. 1707–1722. URL: https://proceedings.mlr.press/v157/matsuoka21b.html.

[MR95] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995. DOI: 10.1017/CBO9780511814075.

[NWF78] George Nemhauser, Laurence Wolsey, and M. Fisher. "An Analysis of Approximations for Maximizing Submodular Set Functions—I". In: *Mathematical Programming* 14 (Dec. 1978), pp. 265–294. DOI: 10.1007/BF01588971.

[Orl07] James Orlin. “A Faster Strongly Polynomial Time Algorithm for Submodular Function Minimization”. In: *Mathematical Programming* 118 (May 2007), pp. 240–251. DOI: 10.1007/s10107-007-0189-2.

[Per+21] Pierre Perrault et al. *On the Approximation Relationship between Optimizing Ratio of Submodular (RS) and Difference of Submodular (DS) Functions*. Jan. 2021. DOI: 10.48550/ARXIV.2101.01631.

[RY22] Akbar Rafiey and Yuichi Yoshida. “Sparsification of Decomposable Submodular Functions”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 36.9 (June 2022), pp. 10336–10344. DOI: 10.1609/aaai.v36i9.21275.

[Sch00] Alexander Schrijver. “A Combinatorial Algorithm Minimizing Submodular Functions in Strongly Polynomial Time”. In: *Journal of Combinatorial Theory, Series B* 80.2 (2000), pp. 346–355. ISSN: 0095-8956. DOI: https://doi.org/10.1006/jctb.2000.1989.

[SY19] Tasuku Soma and Yuichi Yoshida. “Spectral Sparsification of Hypergraphs”. In: *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*. Ed. by Timothy M. Chan. SIAM, 2019, pp. 2570–2581. DOI: 10.1137/1.9781611975482.159.

[TŽ16] Johan Thapper and Stanislav Živný. “The Complexity of Finite-Valued CSPs”. In: *Journal of the ACM* 63.4 (Nov. 2016), pp. 1–33. DOI: 10.1145/2974019.

[Von10] Jan Vondrak. “Submodularity and curvature: the optimal algorithm”. In: *RIMS Kôkyûroku Bessatsu* (Jan. 2010).

[WŽ16] Justin Ward and Stanislav Živný. “Maximizing K-Submodular Functions and Beyond”. In: *ACM Trans. Algorithms* 12.4 (Aug. 2016). ISSN: 1549-6325. DOI: 10.1145/2850419.