Parameterized Complexity of Submodular Minimization Under Uncertainty

Naonori Kakimura
Department of Mathematics, Keio University, Yokohama, Japan

Ildikó Schlotter
HUN-REN Centre for Economic and Regional Studies, Budapest, Hungary
Budapest University of Technology and Economics, Hungary

Abstract

This paper studies the computational complexity of a robust variant of a two-stage submodular minimization problem that we call Robust Submodular Minimizer. In this problem, we are given k submodular functions $f_1, \dots, f_k$ over a set family 2^V, which represent k possible scenarios in the future when we will need to find an optimal solution for one of these scenarios, i.e., a minimizer for one of the functions. The present task is to find a set $X \subseteq V$ that is close to *some* optimal solution for each f_i in the sense that some minimizer of f_i can be obtained from X by adding/removing at most d elements for a given integer $d \in \mathbb{N}$. The main contribution of this paper is to provide a complete computational map of this problem with respect to parameters k and d, which reveals a tight complexity threshold for both parameters:

- Robust Submodular Minimizer can be solved in polynomial time when $k \leq 2$, but is NP-hard if k is a constant with $k \geq 3$.
- Robust Submodular Minimizer can be solved in polynomial time when $d = 0$, but is NP-hard if d is a constant with $d \geq 1$.
- Robust Submodular Minimizer is fixed-parameter tractable when parameterized by (k, d).

We also show that if some submodular function f_i has a polynomial number of minimizers, then the problem becomes fixed-parameter tractable when parameterized by d. We remark that all our hardness results hold even if each submodular function is given by a cut function of a directed graph.

2012 ACM Subject Classification Theory of computation → Submodular optimization and polymatroids; Theory of computation → Fixed parameter tractability; Mathematics of computing → Combinatorial optimization

Keywords and phrases Submodular minimization, optimization under uncertainty, parameterized complexity, cut function

Digital Object Identifier 10.4230/LIPIcs.SWAT.2024.30

Related Version *Full Version*: `https://arxiv.org/abs/2404.07516` [17]

Funding *Naonori Kakimura*: Supported by JSPS KAKENHI Grant Numbers JP22H05001, JP20H05795, and JP21H03397, Japan and JST ERATO Grant Number JPMJER2301, Japan.
Ildikó Schlotter: Supported by the Hungarian Academy of Sciences under its János Bolyai Research Scholarship and its Momentum Programme (LP2021-2).

Acknowledgements The authors are grateful to the organizers of the 14th Emléktábla Workshop which took place in Vác, Hungary in 2023, and provided a great opportunity for our initial discussions.

1 Introduction

This paper proposes a two-stage robust optimization problem under uncertainty. Suppose that we want to find a minimum cut on a directed graph under uncertainty. The uncertainty here is represented by k directed graphs $G_1, \dots, G_k$ on the same vertex set $V \cup \{s, t\}$. That is, we have k possible scenarios of graph realizations in the future. At the moment, we want to choose an (s,t)-cut in advance, so that after the graph is revealed, we will be able to obtain a minimum (s,t)-cut in the graph with small modification. Therefore, our aim is to find an (s,t)-cut that is close to some minimum (s,t)-cut in each graph G_i for $i = 1, \dots, k$.

Let us formalize this problem. For a vertex set X in a directed graph $G = (V \cup \{s,t\}, E)$, the *cut function* $f : 2^V \to \mathbb{Z}$ is the number of out-going edges from X. Let us denote the family of minimum (s,t)-cuts in G by $\mathcal{C}_{s,t}(G)$, that is, $\mathcal{C}_{s,t}(G) = \{Y \subseteq V : f(Y) \leq f(Y') \ \forall Y' \subseteq V\}$. Given directed graphs $G_1, \dots, G_k$ over a common vertex set $V \cup \{s,t\}$, we want to find a subset $X \subseteq V$ and sets $Y_i \in \mathcal{C}_{s,t}(G_i)$ for each $i \in [k]$ that minimizes $\max_{i \in [k]} |X \triangle Y_i|$ where $\triangle$ stands for symmetric difference and $[k]$ denotes $\{1, \dots, k\}$ for any positive integer k.

We study a natural generalization of this problem where, instead of the cut functions of directed graphs which are known to be submodular [28], we consider arbitrary submodular set functions over some non-empty finite set V. Let $f_1, \dots, f_k \colon 2^V \to \mathbb{R}$ be k submodular functions. Let $\arg\min f_i = \{Y \subseteq V : f_i(Y) \leq f_i(Y') \ \forall Y' \subseteq V\}$ refer to the set of minimizers of f_i. We want to find a subset $X \subseteq V$ and sets $Y_i \in \arg\min f_i$ for all $i \in [k]$ that

$$\text{minimize} \qquad \max_{i \in [k]} |X \triangle Y_i|.$$

We call the decision version of this problem Robust Submodular Minimizer.

Robust Submodular Minimizer:	
Input:	A finite set V, submodular functions $f_1, \dots, f_k : 2^V \to \mathbb{R}$, and an integer $d \in \mathbb{N}$.
Task:	Find a set $X \subseteq V$ such that for each $i \in [k]$ there exists $Y_i \in \arg\min f_i$ with $|X \triangle Y_i| \leq d$, or detect if no such set exists.

We remark that the min-sum variant of the problem, that is, the problem obtained by replacing the condition $\max_{i \in [k]} |X \triangle Y_i| \leq d$ with $\sum_{i \in [k]} |X \triangle Y_i| \leq d$, was introduced by Kakimura et al. [16], who showed that it can be solved in polynomial time.

1.1 Our Contributions and Techniques

Our contribution is to reveal the complete computational complexity of Robust Submodular Minimizer with respect to the parameters k and d. We also provide an algorithm for the case when one of the submodular functions has only polynomially many minimizers. Our results are as follows:

1. Robust Submodular Minimizer can be solved in polynomial time when $k \leq 2$ (Theorem 6), but is NP-hard if k is a constant with $k \geq 3$ (Corollary 24).
2. Robust Submodular Minimizer can be solved in polynomial time when $d = 0$ (Observation 4), but is NP-hard if d is a constant with $d \geq 1$ (Theorem 20).
3. Robust Submodular Minimizer is fixed-parameter tractable when parameterized by (k, d).
4. Robust Submodular Minimizer is fixed-parameter tractable when parameterized by d, if the size of $\arg\min f_i$ for some $i \in [k]$ is polynomially bounded.

When $k = 1$, Robust Submodular Minimizer is equivalent to the efficiently solvable submodular function minimization problem [20], in which we are given a single submodular function $f \colon 2^V \to \mathbb{R}$ and want to find a set $X \subseteq V$ in $\arg\min f$. It is not difficult to observe that Robust Submodular Minimizer for $d = 0$ can also be solved in polynomial time by computing a minimizer of the submodular function $\sum_{i=1}^{k} f_i$; see Section 3.1.

The rest of our positive results are based on Birkhoff's representation theorem on distributive lattices [1] that allows us to maintain the family of minimizers of a submodular function in a compact way. Specifically, even though the number of minimizers may be exponential in the input size, we can represent all minimizers as a family of cuts in a directed acyclic graph with polynomial size. As we show in Section 3.1, we can use this representation to solve an instance I of Robust Submodular Minimizer with $k = 2$ by constructing a directed graph with two distinct vertices, s and t, in which a minimum (s,t)-cut yields a solution for I. More generally, Birkhoff's compact representation allows us to reduce Robust Submodular Minimizer for arbitrary k to the so-called Multi-Budgeted Directed Cut problem, solvable by an algorithm due to Kratsch et al. [18], leading to a fixed-parameter tractable algorithm for the parameter (k,d). We note that a similar construction was used to show that the min-sum variant of the problem is polynomial-time solvable [16].

In Section 3.3, we consider the case when one of the k submodular functions has only polynomially many minimizers. As mentioned in [16], Robust Submodular Minimizer is NP-hard even when each submodular function f_i has a unique minimizer. In fact, the problem is equivalent to the Closest String problem over a binary alphabet, shown to be NP-hard under the name Minimum Radius by Frances and Litman [11]. For the case when $|\arg\min f_i|$ is polynomially bounded for some $i \in [k]$, we present a fixed-parameter tractable algorithm parameterized only by d. Our algorithm guesses a set in $\arg\min f_i$ and uses it as an "anchor," then solves the problem recursively by the bounded search-tree technique.

Section 4 contains our NP-hardness results for the cases when either d is a constant at least 1, or k is a constant at least 3. We present reductions from an intermediate problem that may be of independent interest: in this problem, we are given k set families $\mathcal{F}_1, \dots, \mathcal{F}_k$ over a universe V containing two distinguished elements, s and t, with each $\mathcal{F}_i$ containing pairwise disjoint subsets of V; the task is to find a set $X \subseteq V$ containing s but not t that has a bounded distance from each family $\mathcal{F}_i$ for a specific distance measure.

The symbol $\star$ marks statements whose proofs we defer to the full version of our paper [17].

1.2 Related Work

Robust Submodular Minimizer is related to the *robust recoverable* combinatorial optimization problem, introduced by Liebchen et al. [22]. It is a framework of mathematical optimization that allows recourse in decision-making to deal with uncertainty. In this framework, we are given a problem instance with some scenarios and a recovery bound d, and the task is to find a feasible solution X (the first-stage solution) to the instance that can be transformed to a feasible solution Y_i (the second-stage solutions) in each scenario i respecting the recovery bound (e.g., $|X \triangle Y_i| \leq d$ for each i). The cost of the solution is usually evaluated by the sum of the cost of X and the sum of the costs of Y_i's. Robust recoverable versions have been studied for a variety of standard combinatorial optimization problems. Examples include the shortest path problem [5], the assignment problem [10], the travelling salesman problem [12], and others [14, 19, 21]. The setting was originally motivated from the situation where the source of uncertainty was the cost function which changes in the second stage. We consider another situation dealing with *structural uncertainty*, where some unknown set of input elements can be interdicted [8, 15]. Recently, a variant of robust recoverable problems has been studied where certain operations are allowed in the second stage [13].

Reoptimization is another concept related to Robust Submodular Minimizer. In general reoptimization, we are given an instance I of a combinatorial optimization problem and an optimal solution X for I. Then, for a slightly modified instance I' of the problem, we need to make a small change to X so that the resulting solution X' is an optimal (or a good approximate) solution to the modified instance I'. Reoptimization has been studied for several combinatorial optimization problems such as the minimum spanning tree problem [4], the traveling salesman problem [23], and the Steiner tree problem [2].

2 Preliminaries

Graphs and Cuts

Given a directed graph $G = (V, E)$, we write uv for an edge pointing from u to v. For a subset $X \subseteq V$ of vertices in G, let $\delta_G(X)$ denote the set of edges leaving X. If G is an undirected graph, then $\delta_G(X)$ for some set X of vertices denotes the set of edges with exactly one endpoint in X. We may simply write $\delta(X)$ if the graph is clear from the context.

For two vertices s and t in a directed or undirected graph $G = (V, E)$, an *(s,t)-cut* is a set X of vertices such that $s \in X$ but $t \notin X$. A *minimum (s,t)-cut* in G is an (s,t)-cut X that minimizes $|\delta(X)|$. Given a cost function $c \colon E \to \mathbb{R}_+ \cup \{+\infty\}$ on the edges of G where $\mathbb{R}_+$ is the set of all non-negative real numbers, the *(weighted) cut function* $\kappa_G : 2^V \to \mathbb{R}_+ \cup \{+\infty\}$ is defined by

$$\kappa_G(X) = \sum_{e \in \delta(X)} c(e). \tag{1}$$

A *minimum-cost (s,t)-cut* is an (s,t)-cut X that minimizes $\kappa_G(X)$.

Distributive Lattices

In this paper, we will make use of properties of finite distributive lattices on a ground set V.

A *distributive lattice* is a set family $\mathcal{L} \subseteq 2^V$ that is closed under union and intersection, that is, $X, Y \in \mathcal{L}$ implies $X \cup Y \in \mathcal{L}$ and $X \cap Y \in \mathcal{L}$. Then $\mathcal{L}$ is a partially ordered set with respect to set inclusion $\subseteq$, and has a unique minimal element and a unique maximal element.

Birkhoff's representation theorem is a useful tool for studying distributive lattices.

▶ **Theorem 1** (Birkhoff's representation theorem [1]). *Let $\mathcal{L} \subseteq 2^V$ be a distributive lattice. Then there exists a partition of V into $U_0, U_1, \ldots, U_b, U_\infty$, where U_0 and U_∞ can possibly be empty, such that the following hold:*

(1) *Every set in $\mathcal{L}$ contains U_0.*

(2) *Every set in $\mathcal{L}$ is disjoint from U_∞.*

(3) *For every set $X \in \mathcal{L}$, there exists a set $J \subseteq [b]$ of indices such that $X = U_0 \cup \bigcup_{j \in J} U_j$.*

(4) *There exists a directed acyclic graph $G(\mathcal{L})$ that has the following properties.*

  (a) *The vertex set is $\{U_0, U_1, \ldots, U_b\}$.*

  (b) *U_0 is a unique sink[1] of $G(\mathcal{L})$.*

  (c) *For a non-empty set Z of vertices in $G(\mathcal{L})$, Z has no out-going edge if and only if $\bigcup_{U_j \in Z} U_j \in \mathcal{L}$.*

For a distributive lattice $\mathcal{L} \subseteq 2^V$, we call the directed acyclic graph $G(\mathcal{L})$ above a *compact representation* of $\mathcal{L}$. Note that the size of $G(\mathcal{L})$ is $O(|V|^2)$ while $|\mathcal{L}|$ can be as large as $2^{|V|}$.

Submodular Function Minimization

Let V be a non-empty finite set. A function $f \colon 2^V \to \mathbb{R}$ is *submodular* if $f(X) + f(Y) \geq f(X \cup Y) + f(X \cap Y)$ for all $X, Y \subseteq V$. A typical example of submodular functions is the cut function κ_G of a directed (or undirected) edge-weighted graph G as defined in (1). If the graph $G = (V \cup \{s,t\}, E)$ contains two distinct vertices, s and t, then we can restrict the cut function to the domain of (s,t)-cuts in the following sense: each $X \subseteq V$ corresponds to an (s,t)-cut $X \cup \{s\}$ in G; then the function $\lambda_G : 2^V \to \mathbb{R}_+ \cup \{+\infty\}$ defined by $\lambda_G(X) = \kappa_G(X \cup \{s\})$ is submodular.

[1] A *sink* is a vertex of out-degree zero.

When we discuss computations on a submodular function $f\colon 2^V \to \mathbb{R}$, we assume that we are given a *value oracle* of f. A value oracle takes $X \subseteq V$ as an input, and returns the value $f(X)$. Assuming that we are given a value oracle, we can minimize a submodular function in polynomial time. The currently fastest algorithm for submodular function minimization was given by Lee et al. [20] and runs in $\tilde{O}(n^3\mathrm{EO} + n^4)$ time, where $n = |V|$ and EO is the query time of a value oracle.

Let $f\colon 2^V \to \mathbb{R}$ be a submodular function. A subset $Y \subseteq V$ is a *minimizer* of the function f if $f(Y) \le f(Y')$ for all $Y' \subseteq V$. The set of minimizers of f is denoted by $\arg\min f$. The following is a well-known fact on submodular functions.

▶ **Lemma 2** (See e.g., [28]). *Let $f\colon 2^V \to \mathbb{R}$ be a submodular function. Then* $\arg\min f$ *forms a distributive lattice.*

A compact representation of the distributive lattice $\arg\min f$ can be constructed in $\tilde{O}(n^5\mathrm{EO} + n^6)$ time via Orlin's submodular function minimization algorithm [25]. See [24, Notes 10.11–10.12]. We will assume that the submodular functions given in our problem instances are given via their compact representation.

As a special case, consider minimum (s,t)-cuts in a directed graph $G = (V \cup \{s,t\}, E)$ with a positive cost function c on its edges. By Lemma 2, the family of minimum (s,t)-cuts forms a distributive lattice. A compact representation for this lattice can be constructed from a maximum flow in the (s,t)-network in linear time [26]. Thus the running time is dominated by the maximum flow computation, and this can be done in $|E|^{1+o(1)}$ time [6].

Parameterized Complexity

In parameterized complexity, each input instance I of a *parameterized problem* Q is associated with a *parameter* k, usually an integer or a tuple of integers, and we consider the running time of any algorithm solving Q as not only a function of the input length $|I|$, but also as a function of the parameter k. An algorithm for Q is *fixed-parameter tractable* or FPT, if it runs in time $g(k)|I|^{O(1)}$ for some computable function g. Such an algorithm can be efficient in practice if the parameter is small. See the books [7, 9] for more background.

3 Algorithms for Robust Submodular Minimizer

In this section, we present algorithms for Robust Submodular Minimizer. We start with a construction that we will use in most of our algorithms. Let $I_{\mathrm{RSM}} = (V, f_1, \dots, f_k, d)$ be our input instance.

For each $i \in [k]$, let $\mathcal{L}_i = \arg\min f_i$ denote the set of minimizers. By Lemma 2, using Birkhoff's representation theorem we may assume that f_i is given through a compact representation $G(\mathcal{L}_i)$ of $\mathcal{L}_i$, whose vertex set is $\{U_0^i, U_1^i, \dots, U_{b_i}^i\}$ with $U_\infty^i = V \setminus \bigcup_{j=0}^{b_i} U_j^i$.

We then construct a directed graph G^i from $G(\mathcal{L}_i)$ by expanding each vertex in $G(\mathcal{L}_i)$ to a complete graph. More precisely, G^i has vertex set $V^i \cup \{s,t\}$ where $V^i = \{v^i : v \in V\}$ is a copy of V, and its edge set E^i is defined as follows.

- $u^i v^i \in E^i$ if $u, v \in U_j^i$ for some $j \in \{0, 1, \dots, b_i, \infty\}$.
- $u^i v^i \in E^i$ for any $u \in U_j^i$ and $v \in U_{j'}^i$ if $G(\mathcal{L}_i)$ has an edge from U_j^i to $U_{j'}^i$.
- $u^i s \in E^i$ and $s u^i \in E^i$ if $u \in U_0^i$.
- $u^i t \in E^i$ and $t u^i \in E^i$ if $u \in U_\infty^i$.

We define the function $\lambda_i : 2^{V^i} \to \mathbb{Z}_+$ so that $\lambda_i(X) = |\delta_{G^i}(X \cup \{s\})|$ for a subset $X \subseteq V^i$. Then it is observed below that each subset $X \subseteq V^i$ with $\lambda_i(X) = 0$ corresponds to a minimizer of f_i.

▶ **Lemma 3** ([16, Lemma 3.2]). *Let X be a subset in V, and $X^i = \{v^i \in V^i : v \in X\}$ its copy in G^i. Then $\lambda_i(X^i) = 0$ if and only if $X \in \mathcal{L}_i$.*

The rest of the section is organized as follows. In Section 3.1 we present polynomial-time algorithms for the cases $d = 0$ and $k = 2$. In Section 3.2 we give an FPT algorithm for the combined parameter (k, d). Section 3.3 deals with the case when some function f_i has only polynomially many minimizers, allowing for an FPT algorithm with parameter d.

3.1 Polynomial-time algorithms

We start by observing that the case $d = 0$ is efficiently solvable by computing a minimizer for the function $\sum_{i\in[k]} f_i$ which is also submodular.

▶ **Observation 4** ($\star$). *Robust Submodular Minimizer can be solved in polynomial time if $d = 0$.*

Next, we show that the problem is polynomial-time solvable when $k = 2$. We will need the following intuitive fact.

▶ **Proposition 5** ($\star$). *Let Y_1, Y_2 be two subsets of a set V. Then $|Y_1 \triangle Y_2| \le 2d$ if and only if there exists a set $X \subseteq V$ such that $|X \triangle Y_i| \le d$ for each $i \in \{1, 2\}$.*

▶ **Theorem 6.** *Robust Submodular Minimizer for $k = 2$ can be solved in polynomial time via a maximum flow computation.*

Proof. Let our instance be $I_{\text{RSM}} = (V, f_1, f_2, d)$. Using the method explained at the beginning of Section 3, we construct the directed graphs $G^1 = (V^1 \cup \{s, t\}, E^2)$ and $G^2 = (V^2 \cup \{s, t\}, E^2)$ for f_1 and f_2. We then construct a directed graph $\tilde{G} = (\tilde{V}, \tilde{E})$ by identifying s, as well as t, in G^1 and G^2, and then connecting the corresponding copies of each vertex with a bidirected edge. That is, $\tilde{V} = V^1 \cup V^2 \cup \{s, t\}$ and $\tilde{E} = E^1 \cup E^2 \cup E'$ where $E' = \big\{v^1v^2, v^2v^1 : v \in V\big\}$. We set $c(e) = +\infty$ for all edges $e \in E^1 \cup E^2$, and we set $c(e) = 1$ for all edges $e \in E'$.

We next compute a minimum-cost (s, t)-cut Z in the graph $\tilde{G}$ with cost function c using standard flow techniques. Let $\kappa_{\tilde{G}}$ denote the cut function in this graph. We will show below that $\kappa_{\tilde{G}}(Z) \le 2d$ if and only if the answer is "yes".

First suppose that $\kappa_{\tilde{G}}(Z) \le 2d$. Let $Y_1 = \{v \in V : v^1 \in Z\}$ and $Y_2 = \{v \in V : v^2 \in Z\}$. Since $\delta_{\tilde{G}}(Z)$ has no edges in $E^1 \cup E^2$, we see that $\lambda_i(\{v^i \in V^i : v \in Y_i\}) = 0$ for both $i = 1, 2$, and therefore the set Y_i is in $\mathcal{L}_i$ by Lemma 3. Since $|Y_1 \triangle Y_2| = \kappa_{\tilde{G}}(Z) \le 2d$, we can compute a set X such that $|X \triangle Y_i| \le d$ for both $i = 1, 2$ by Proposition 5.

Conversely, let $X \subseteq V$ and $Y_i \in \mathcal{L}_i$ for each $i = 1, 2$ such that $|X \triangle Y_i| \le d$. Define $Z = \{s\} \cup \{v^1 \in V^1 : v \in Y_1\} \cup \{v^2 \in V^2 : v \in Y_2\}$. Due to Lemma 3 we know that $\lambda_i(\{v^i \in V^i : v \in Y_i\}) = 0$ for both $i = 1, 2$. This implies $\kappa_{\tilde{G}}(Z) = |Y_1 \triangle Y_2| \le 2d$ where the inequality follows from Proposition 5. ◀

3.2 FPT algorithm for parameter (k, d)

We propose a fixed-parameter tractable algorithm for Robust Submodular Minimizer parameterized by k and d; let $I_{\text{RSM}} = (V, f_1, \ldots, f_k, d)$ denote our instance.

▶ **Theorem 7.** *Robust Submodular Minimizer can be solved in FPT time when parameterized by (k, d).*

To this end, we reduce our problem to the Multi-Budgeted Directed Cut problem [18], defined as follows. We are given a directed graph $D = (V, E)$ with distinct vertices s and t, together with pairwise disjoint edge sets $A_1, \ldots, A_k$, and positive integers $d_1, \ldots, d_k$. The task is to decide whether D has an (s,t)-cut X such that $|\delta(X) \cap A_i| \le d_i$ for each $i \in [k]$.

▶ **Proposition 8** (Kratsch et al. [18]). *The Multi-Budgeted Directed Cut problem can be solved in FPT time when the parameter is $\sum_{i=1}^k d_i$.*

In fact, we will need to use *forbidden* edges, so let us define the Multi-Budgeted Directed Cut with Forbidden Edges problem as follows. Given an instance I_{MBC} of Multi-Budgeted Directed Cut and a set F of forbidden edges, find a solution X for I_{MBC} such that $\delta(X)$ is disjoint from F. It is straightforward to solve this problem using the results by Kratsch et al. [18], after replacing each forbidden edge with an appropriate number of parallel edges. Hence, we get the following.

▶ **Proposition 9** ($\star$). *The Multi-Budgeted Directed Cut with Forbidden Edges problem can be solved in FPT time when the parameter is $\sum_{i=1}^k d_i$.*

Reduction to Multi-Budgeted Directed Cut with Forbidden Edges

Compute the graph G^i for each $i \in [k]$, as described at the beginning of Section 3. We construct a large directed graph $\tilde{G} = (\tilde{V}, \tilde{E})$ as follows. We identify all vertices s (and t, respectively) in the graphs G^i into a single vertex s (and t, respectively). We further prepare another copy of V, which is denoted by $V^* = \{v^* : v \in V\}$. Thus the vertex set of $\tilde{G}$ is defined by $\tilde{V} = \bigcup_{i=1}^k V^i \cup V^* \cup \{s,t\}$. The edge set of $\tilde{G}$ consists of E^i and bidirected edges connecting v^* and the copy v^i of v in G^i, for each $i \in [k]$. That is,

$$\tilde{E} = \bigcup_{i=1}^{k} \left(E^i \cup A^i\right) \quad \text{where} \quad A^i = \left\{v^*v^i, v^iv^* : v \in V\right\}.$$

We also set $d_i = d$ for each $i \in [k]$. Consider the instance $I_{\text{MBC}} = (\tilde{G}, s, t, \{A^i\}_{i=1}^k, \{d_i\}_{i=1}^k)$ of multi-budgeted directed cut with $F = \bigcup_{i=1}^k E^i$ as forbidden edges; note that its parameter is $k \cdot d$. Theorem 7 immediately follows from Proposition 9 and Lemma 10 below.

▶ **Lemma 10.** *There exists a solution for I_{RSM} if and only if there exists a solution for the instance (I_{MBC}, F) of Multi-Budgeted Directed Cut with Forbidden Edges.*

Proof. Suppose that (I_{MBC}, F) admits a solution. That is, there exists a subset X of $\tilde{V}$ containing s but not t such that $\delta_{\tilde{G}}(X)$ is disjoint from F and satisfies $|\delta_{\tilde{G}}(X) \cap A^i| \le d_i$ for each $i \in [k]$. Define $Y^i = X \cap V^i$ for $i = 1, \ldots, k$. Observe that all edges within G^i leaving $Y^i \cup \{s\}$ also leave X in $\tilde{G}$, since $s \in X$ but $t \notin X$. Since all edges in E^i are forbidden edges, we see that $\lambda_i(Y^i) = 0$. Let $Y_i = \{v \in V : v^i \in Y^i\}$, so that Y^i contains the copy of each vertex of Y_i in G^i. Then Y_i is in $\mathcal{L}_i$ by Lemma 3.

Define the subset $X^* = \{v : v^* \in X\}$ of V. Observe that

$$\delta_{\tilde{G}}(X) \cap A^i = \{v^*v^i : v \in X^*, v \notin Y_i\} \cup \{v^iv^* : v \notin X^*, v \in Y_i\}.$$

Therefore, we get that $|X^* \triangle Y_i| = |\delta_{\tilde{G}}(X) \cap A^i| \le d_i = d$ for each $i \in [k]$ as required, so X^* is a solution to our instance I_{RSM} of Robust Submodular Minimizer.

Conversely, let $X \subseteq V$ and $Y_i \in \mathcal{L}_i$ for each $i \in [k]$ such that $|X \triangle Y_i| \leq d$. Define $X^* = \{v^* \in V^* : v \in X\}$ and $Y^i = \{v^i \in V^i : v \in Y_i\}$. Then the set $\tilde{X} = \{s\} \cup X^* \cup \bigcup_{i=1}^k Y^i$ is an (s,t)-cut of $\tilde{G}$ such that

$$\begin{aligned}\delta_{\tilde{G}}(\tilde{X}) \cap A^i &= \{v^*v^i : v^* \in X^*, v \notin Y^i\} \cup \{v^iv^* : v^* \notin X^*, v^i \in Y^i\} \\ &= \{v^*v^i : v \in X, v \notin Y_i\} \cup \{v^iv^* : v \notin X, v \in Y_i\} = X \triangle Y_i.\end{aligned}$$

Hence we obtain $|\delta_{\tilde{G}}(\tilde{X}) \cap A^i| = |X \triangle Y_i| \leq d = d_i$ for each $i \in [k]$. Since Y_i is in $\mathcal{L}_i$, by Lemma 3 we know $\lambda_i(Y^i) = 0$ for each $i \in [k]$. Thus $\delta_{\tilde{G}}(\tilde{X})$ is disjoint from the set F of forbidden edges, and therefore $\tilde{X}$ is indeed a solution to our instance (I_{MBC}, F) of MULTI-BUDGETED DIRECTED CUT WITH FORBIDDEN EDGES. ◀

3.3 Polynomially many minimizers: FPT algorithm parameterized by d

In this section, we present a fixed-parameter tractable algorithm for the case when our threshold d is small, assuming that $|\mathcal{L}_1|$ can bounded by a polynomial of the input size. Note that even with a much stronger assumption, ROBUST SUBMODULAR MINIMIZER remains intractable (see also [16]):

▶ **Observation 11.** *ROBUST SUBMODULAR MINIMIZER is* NP*-hard even if* $|\mathcal{L}_i| = 1$ *for each* $i \in [k]$.

Proof. If $|\mathcal{L}_i| = 1$ for each $i \in [k]$, then there is a unique minimizer $Y_i \subseteq V$ for each f_i, and the problem is equivalent with finding a set $X \subseteq V$ whose symmetric difference is at most d from each of the sets Y_i, $i \in [k]$. This is the CLOSEST STRING problem over a binary alphabet, shown to be NP-hard under the name MINIMUM RADIUS by Frances and Litman [11]. ◀

▶ **Theorem 12.** *ROBUST SUBMODULAR MINIMIZER can be solved in* $|\mathcal{L}_1|g(d)n^c$ *time where* c *is a constant and* g *is a computable function.*

Let us consider a slightly more general version of ROBUST SUBMODULAR MINIMIZER which we call ANCHORED SUBMODULAR MINIMIZER. In this problem, in addition to an instance $I_{\text{RSM}} = (V, f_1, \ldots, f_k, d)$ of ROBUST SUBMODULAR MINIMIZER, we are given a set $Y_0 \subseteq V$ and integer $d_0 \leq d$, and we aim to find a subset X such that

$$|X \triangle Y_0| \leq d_0 \quad \text{and} \tag{2}$$

$$|X \triangle Y_i| \leq d \quad \text{for some } Y_i \in \mathcal{L}_i, \text{ for each } i \in [k]. \tag{3}$$

Observe that we can solve our instance $I_{\text{RSM}} = (V, f_1, \ldots, f_k, d)$ by solving the instance $(V, f_2, \ldots, f_k, d, Y_0, d_0)$ of ANCHORED SUBMODULAR MINIMIZER for each $Y_0 \in \mathcal{L}_1$ and $d_0 = d$. Hence, Theorem 12 follows from Theorem 13 below.

▶ **Theorem 13.** *ANCHORED SUBMODULAR MINIMIZER can be solved in FPT time when parameterized by* d.

To prove Theorem 13, we will use the technique of bounded search-trees. Given an instance $I = (V, f_1, \ldots, f_k, d, Y_0, d_0)$, after checking whether Y_0 itself is a solution, we search for a minimizer $Y_i \in \mathcal{L}_i$ for which $d < |Y_0 \triangle Y_i| \leq d + d_0$. It is not hard to see the following.

▶ **Observation 14.** *If* X *is a solution for an instance* $I = (V, f_1, \ldots, f_k, d, Y_0, d_0)$ *of ANCHORED SUBMODULAR MINIMIZER, and* $Y_i \in \mathcal{L}_i$ *fulfills* $|X \triangle Y_i| \leq d$*, then for all* $T \subseteq Y_0 \triangle Y_i$ *with* $|T| > d$ *it holds that there exists some* $v \in T$ *with* $v \in X \triangle Y_0$.

Proof. Indeed, assuming that the claim does not hold, we have that $T \cap (Y_0 \setminus Y_i) \subseteq X$ and that $(T \cap (Y_i \setminus Y_0)) \cap X = \emptyset$. From the former, $T \cap (Y_0 \setminus Y_i) \subseteq X \setminus Y_i$ follows, while the latter implies $T \cap (Y_i \setminus Y_0) \subseteq Y_i \setminus X$. Thus,

$$X \triangle Y_i = (X \setminus Y_i) \cup (Y_i \setminus X) \supseteq (T \cap (Y_0 \setminus Y_i)) \cup (T \cap (Y_i \setminus Y_0)) = T \cap (Y_0 \triangle Y_i) = T.$$

Hence, $|X \triangle Y_i| \geq |T| > d$, contradicting our assumption that X is a solution for I. ◀

Our algorithm will compute in $O^*(2^d)$ time[2] a set $T \subseteq Y_0 \setminus Y_i$ of size $d < |T| \leq d + d_0$ that contains some element v fulfilling the above conditions. Then, by setting $Y_0 \leftarrow Y_0 \triangle \{v\}$ and reducing the value of d_0 by one, we obtain an equivalent instance I' of Anchored Submodular Minimizer which we solve by applying recursion.

Description of our algorithm

Our algorithm will make "guesses"; nevertheless, it is a deterministic one, where guessing a value from a given set U is interpreted as branching into $|U|$ branches. We continue the computations in each branch, and whenever a branch returns a solution for the given instance, we return it; if all branches reject the instance (by outputting "No"), we also reject it. See Algorithm ASM for a pseudo-code description.

We start by checking whether Y_0 is a solution for our instance $I = (V, f_1, \ldots, f_k, d, Y_0, d_0)$, that is, whether it satisfies (3). This can be done in polynomial time, since the set function $\gamma_i(Z) = \min\{|Z \triangle Y_i| : Y_i \in \mathcal{L}_i\}$ is known to be submodular and can be computed via a maximum flow computation [16]. If Y_0 satisfies (3), i.e., $\gamma_i(Y_0) \leq d$ for each $i \in [k]$, then we output Y_0; note that (2) is obviously satisfied by Y_0, so Y_0 is a solution for I.

Otherwise, if $d_0 = 0$, then we output "No" as in this case the only possible solution could be Y_0. We proceed by fixing an index $i \in [k]$ such that $\gamma_i(Y_0) > d$, that is, $|Y_0 \triangle Y| > d$ for all minimizers $Y \in \mathcal{L}_i$.

▶ **Observation 15.** *If X is a solution for I that satisfies $|X \triangle Y_i| \leq d$ for some $Y_i \in \mathcal{L}_i$, then $|Y_i \triangle Y_0| \leq d + d_0$.*

Proof. Since X is a solution for I, we have $|X \triangle Y_0| \leq d_0$, and thus the triangle inequality implies $|Y_i \triangle Y_0| \leq |X \triangle Y_i| + |X \triangle Y_0| \leq d + d_0$. ◀

By our choice of i and Observation 15, we know that $d < |Y_0 \triangle Y_i| \leq d + d_0$. We are going to compute a set $T \subseteq Y_0 \triangle Y_i$ with the same bounds on its cardinality, i.e., $d < |T| \leq d + d_0$.

To this end, we compute a compact representation $G(\mathcal{L}_i)$ of the distributive lattice $\mathcal{L}_i$; let $\mathcal{P} = \{U_0, U_1, \ldots, U_b, U_\infty\}$ be the partition of V in this representation.

Next, we proceed with an iterative procedure which also involves a set of guesses. We start by setting $Y = Y_0$ and $T = \emptyset$. We will maintain a family of *fixed sets* from $\mathcal{P}$ for which we already know whether they are in Y_i or not (according to our guesses); initially, no set from $\mathcal{P}$ is fixed.

After this initialization, we start an iteration where at each step we check whether $Y \in \mathcal{L}_i$ or $|T| > d$. If yes, then we stop the iteration. If not, then it can be shown that one of the following conditions holds:

Condition 1: there exists a set $S \in \mathcal{P}$ such that $S \cap Y \neq \emptyset$ and $S \setminus Y \neq \emptyset$;

Condition 2: there exists an edge (S, S') in $G(\mathcal{L}_i)$ for which $S \subseteq Y$ but $S' \cap Y = \emptyset$.

[2] The $O^*()$ notation hides polynomial factors.

If Condition 1 holds for some set $S \in \mathcal{P}$, then we guess whether S is contained in Y_i. If $S \subseteq Y_i$ according to our guesses, then we add $S \setminus Y$ to T; otherwise, we add $S \cap Y$ to T. In either case, we declare S as fixed, and proceed with the next iteration.

By contrast, if Condition 1 fails, but Condition 2 holds for some edge (S, S') in $G(\mathcal{L}_i)$ with endpoints $S, S' \in \mathcal{P}$, then we proceed as follows. If both S and S' are fixed, then we stop and reject the current set of guesses. If S is fixed but S' is not, then we add all elements of S' to T. If S' is fixed but S is not, then we add S to T. If neither S nor S' is fixed, then we guess whether S is contained in Y_i or not, and in the former case we add S' to T, while in the latter case we add S to T. In all cases except for the last one, we declare both S and S' as fixed; in the last case declare only S as fixed.

Next, we modify Y to reflect the current value of T by updating Y to $Y_0 \triangle T$. If $|T| > d+d_0$, then we reject the current branch. If $d < |T| \leq d + d_0$, then we finish the iteration; otherwise, we proceed with the next iteration.

Finally, when the iteration stops, we guess a vertex $v \in T$, define $Y'_{0,v} = Y_0 \triangle \{v\}$ and call the algorithm recursively on the instance $I'_v := (V, f_1, \dots, f_k, d, Y'_{0,v}, d_0 - 1)$.

Algorithm ASM Solving ANCHORED SUBMODULAR MINIMIZER on $I = (V, f_1, \dots, f_k, d, Y_0, d_0)$.

```
1: for all $j \in [k]$ do compute the value $\gamma_j = \min\{|Y_0 \triangle Y| : Y \in \arg\min f_j\}$.
2: if $\gamma_j \leq d$ for each $j \in [k]$ then return $Y_0$.
3: if $d_0 = 0$ then return "No".
4: Fix an index $i \in [k]$ such that $\gamma_i > d$.
5: Compute the graph $G(\mathcal{L}_i)$, and let $\mathcal{P}$ be its vertex set.
6: Set $T := \emptyset$ and $Y := Y_0$, and fixed$(S)$ := false for each $S \in \mathcal{P}$.
7: while $Y \notin \mathcal{L}_i$ and $|T| \leq d$ do
8:     if $\exists S \in \mathcal{P} : S \cap Y_0 \neq \emptyset, S \setminus Y_0 \neq \emptyset$ then
9:         Guess contained$(S)$ from {false, true}.
10:        if contained$(S)$ = true then set $T := T \cup (S \setminus Y)$.
11:        else set $T := T \cup (S \cap Y)$.
12:        Set fixed$(S)$ := true.
13:    else Find an edge $(S, S') \in G(\mathcal{L}_i)$ such that $S \subseteq Y$ and $S' \cap Y = \emptyset$.
14:        if fixed$(S)$ = true then
15:            if fixed$(S')$ = true then return "No".
16:            else set $T := T \cup S'$ and fixed$(S')$ := true.
17:        else                                                  ▷ fixed$(S)$ = false.
18:            if fixed$(S')$ = true then set $T := T \cup S$ and fixed$(S)$ := true.
19:            else guess contained$(S)$ from {false, true}.
20:                if contained$(S)$ = true then set $T := T \cup S'$, fixed$(S)$ := fixed$(S')$ := true.
21:                else set $T := T \cup S$ and fixed$(S)$ := true.
22:    Set $Y := Y_0 \triangle T$.
23:    if $|T| > d + d_0$ then return "No".
24: Guess a vertex $v$ from $T$.
25: Set $Y'_{0,v} = Y_0 \triangle \{v\}$ and $I'_v = (V, f_1, \dots, f_k, d, Y'_{0,v}, d_0 - 1)$.
26: return ASM$(I'_v)$.
```

Proof of Theorem 13. We first prove the correctness of the algorithm. Clearly, for $d_0 = 0$, the algorithm either correctly outputs the solution Y_0, or rejects the instance. Hence, we can apply induction on d_0, and assume that the algorithm works correctly when called for an instance with a smaller value for d_0.

We show that any set X returned by the algorithm is a solution for I. First, this is clear if $X = Y_0$, as the algorithm explicitly checks whether $\gamma_i(Y_0) \leq d$ holds for each $i \in [k]$; second, if X was returned by a recursive call on some instance I'_v, then by our induction hypothesis we know that X is a solution for $I'_v = (V, f_1, \dots, f_k, d, Y'_{0,v}, d_0 - 1)$. Hence, X satisfies (3); moreover, by $|X \triangle Y'_{0,v}| \leq d_0 - 1$, it also satisfies $|X \triangle Y_0| \leq d_0$, because $|Y_0 \triangle Y'_{0,v}| = 1$.

Let us now prove that if I admits a solution X, then the algorithm correctly returns a solution for I. Let $Y_i \in \mathcal{L}_i$ be a minimizer such that $|X \triangle Y_i| \leq d$ where i is the index fixed for which $\gamma_i(Y_0) > d$.

▷ Claim 16 (⋆). Assuming that all guesses made by the algorithm are correct, in the iterative process of modifying T and Y it will always hold that

(i) $T \subseteq Y_i \triangle Y_0$, and

(ii) for each $S \in \mathcal{P}$:

 (a) if S is fixed, then $S \subseteq Y \iff S \subseteq Y_i$, and $S \cap Y = \emptyset \iff S \cap Y_i = \emptyset$, and

 (b) if $v \in S$ and S is not fixed, then $v \in Y \iff v \in Y_0$.

Next, we show that in each run of the iteration, Condition 1 or Condition 2 holds. Indeed, if neither holds, then (1) $Y = \bigcup_{U \in \mathcal{P}'} U$ for some $\mathcal{P}' \subseteq \mathcal{P}$, and (2) no edge leaves $\mathcal{P}'$ in $G(\mathcal{L}_i)$. Hence, $Y \in \mathcal{L}_i$ by Birkhoff's representation theorem. However, since $|T| \leq d$ holds at the beginning of each iteration, $|Y \triangle Y_0| = |T| \leq d$ follows, contradicting our choice of i.

Therefore, in each run of the iteration, at least one element of V is put into T. Thus, the iteration stops after at most $d+1$ runs, at which point the obtained set T has size greater than d. Using now statement (i) of Claim 16, Observation 14 yields that T contains at least one vertex from $X \triangle Y_0$. Assuming that the algorithm guesses such a vertex v correctly, it is clear that our solution X for I will also be a solution for the instance I'_v. Using our inductive hypothesis, we obtain that the recursive call returns a correct solution for I'_v which, as discussed already, will be a solution for I as well. Hence, our algorithm is correct.

Finally, let us bound the running time. Consider the search tree $\mathcal{T}$ where each node corresponds to a call of Algorithm ASM. Note that the value of d_0 decreases by one in each recursive call, and the algorithm stops when $d_0 = 0$. Hence $\mathcal{T}$ has depth at most d_0. Consider the guesses made during the execution of a single call of the algorithm (without taking into account the guesses in the recursive calls): we make at most one guess in each iteration on line 9 or on line 19, leading to at most 2^{d+1} possibilities. Then the algorithm further guesses a vertex from T, leading to a total of at most $2^{d+1}|T| \leq 2^{d+1}(d + d_0) = 2^{O(d)}$ possibilities; recall that $d_0 \leq d$. We get that the number of nodes in our search tree is $2^{d_0 O(d)}$. Since all computations for a fixed series of guesses take polynomial time, we obtain that the running time is indeed fixed-parameter tractable with parameter d. ◀

4 Hardness Results

We first introduce a separation problem that we will use as an intermediary problem in our hardness proofs. Given a subset $X \subseteq V$ of some universe V that contains two distinguished elements, s and t, and a family Π of pairwise disjoint subsets of V, we define the *distance* of the set X from Π as $\sum_{S \in \Pi} \mathsf{dist}_{s,t}(X, S)$ where

$$\mathsf{dist}_{s,t}(X,S) = \begin{cases} \min\{|S \setminus X|, |S \cap X|\} & \text{if } s \notin S, t \notin S; \\ |S \setminus X| & \text{if } s \in S, t \notin S; \\ |S \cap X| & \text{if } s \notin S, t \in S; \\ +\infty & \text{if } s \in S, t \in S. \end{cases}$$

Given a collection of set families $\Pi_1, \dots, \Pi_k$, the goal is to find a set $X \subseteq V$ that separates s from t in the sense that $s \in X$ but $t \notin X$, and subject to this constraint, minimizes the maximum distance of X from the given set families. Formally, the problem is:

ROBUST SEPARATION:

Input: A finite set V with two elements $s, t \in V$, set families $\Pi_1, \dots, \Pi_k$ where each Π_i is a collection of pairwise disjoint subsets of V, and an integer $d \in \mathbb{N}$.

Task: Find a set $X \subseteq V$ containing s but not t such that for each $i \in [k]$

$$\sum_{S \in \Pi_i} \mathsf{dist}_{s,t}(X, S) \le d, \tag{4}$$

or output "No" if no such set X exists.

Given an instance $(V, s, t, \Pi_1 \dots, \Pi_k, d)$ of ROBUST SEPARATION, the reduction proving Lemma 17 below constructs a graph G_i over V for each $i \in [k]$ in which each set in Π_i forms a clique, and defines a submodular function f_i based on the cut function of G_i.

▶ **Lemma 17** ($\star$). *ROBUST SEPARATION can be reduced to ROBUST SUBMODULAR MINIMIZER in polynomial time via a reduction that preserves the values of both k and d.*

4.1 NP-hardness for a constant $d \ge 1$

In this section, we prove that ROBUST SUBMODULAR MINIMIZER is NP-hard for each constant $d \ge 1$. To this end, we first prove the NP-hardness of ROBUST SEPARATION in the case $d = 1$, and then extend this result to hold for any constant $d \ge 1$.

For the case $d = 1$, we present a reduction from the 1-IN-3 SAT problem. In this problem, we are given a set V of variables and a set $\mathcal{C}$ of clauses, with each clause $C \in \mathcal{C}$ containing exactly three distinct literals; here, a *literal* is either a variable $v \in V$ or its negation $\overline{v}$. Given a truth assignment $\phi : V \to \{\texttt{true}, \texttt{false}\}$, we automatically extend it to the set $\overline{V} = \{\overline{v} : v \in V\}$ of negative literals by setting $\phi(\overline{v}) = \texttt{true}$ if and only if $\phi(v) = \texttt{false}$. We say that a truth assignment is *valid*, if it maps *exactly* one literal in each clause to $\texttt{true}$. The task in the 1-IN-3 SAT problem is to decide whether a valid truth assignment exists. This problem is NP-complete [27].

▶ **Theorem 18** ($\star$). *ROBUST SEPARATION is NP-hard even when $d = 1$.*

Proof. Suppose that we are given an instance of the 1-IN-3 SAT problem with variable set V and clause set $\mathcal{C} = \{C_1 \dots, C_m\}$. We construct an instance I_{RS} of ROBUST SEPARATION as follows. In addition to the set V of variables and the set $\overline{V} = \{\overline{v} : v \in V\}$ of negative literals, we introduce our two distinguished elements, s and t. We further introduce a set $R_j = \{r_{j,1}, r_{j,2}, r_{j,3}\}$ together with an extra element z_j for each clause $C_j \in \mathcal{C}$ to form our universe U. We let $R = R_1 \cup \dots \cup R_m$ and $Z = \{z_1, \dots, z_m\}$, so that

$$U = V \cup \overline{V} \cup \{s, t\} \cup \bigcup_{j \in [m]} (R_j \cup \{z_j\}) = V \cup \overline{V} \cup \{s, t\} \cup R \cup Z.$$

Next, for each variable, we introduce two set families, Π_v and $\Pi_{\overline{v}}$, where

$$\Pi_v = \{\{s, v, \overline{v}\} \cup R\} \qquad \text{and} \qquad \Pi_{\overline{v}} = \{\{v, \overline{v}, t\}\}.$$

For simplicity, we write $\Pi(V) = \langle \Pi_v, \Pi_{\overline{v}} : v \in V \rangle$ to denote the $2|V|$-tuple formed by these set families. For each clause $C_j \in \mathcal{C}$, we fix an arbitrary ordering of its literals, and we denote the first, second, and third literals in C_j as $\ell_{j,1}, \ell_{j,2}$ and $\ell_{j,3}$. We define three set families:

$$\begin{array}{lll}
\Pi_{C_j} = \{S_j\} & \text{where} & S_j \;\; = C_j \cup \{t\} = \{\ell_{j,1}, \ell_{j,2}, \ell_{j,3}, t\}, \\
\Pi^{\alpha}_{C_j} = \{S^{\alpha,1}_j, S^{\alpha,2}_j\} & \text{where} & S^{\alpha,1}_j = \{\ell_{j,1}, z_j\}, \\
 & & S^{\alpha,2}_j = \{\ell_{j,2}, r_{j,2}\}; \\
\Pi^{\beta}_{C_j} = \{S^{\beta,1}_j, S^{\beta,2}_j\} & \text{where} & S^{\beta,1}_j = \{r_{j,1}, z_j\}, \\
 & & S^{\beta,2}_j = \{\ell_{j,3}, r_{j,3}\}.
\end{array}$$

We also write $\Pi(\mathcal{C}) = \langle \Pi_C, \Pi^{\alpha}_C, \Pi^{\beta}_C : C \in \mathcal{C}\rangle$ to denote the $3|\mathcal{C}|$-tuple formed by these set families in an arbitrarily fixed ordering. We set our threshold as $d = 1$. Thus, our instance of Robust Separation is $I_{\mathrm{RS}} = (U, s, t, \Pi(V), \Pi(\mathcal{C}), 1)$.

We will show that the constructed instance I_{RS} has a solution if and only if our instance $(V, \mathcal{C})$ of the 1-in-3 SAT problem is solvable.

First suppose that there is a valid truth assignment ϕ for $(V, \mathcal{C})$. Consider the set

$$X = \{s\} \cup R \cup \{\ell : \ell \in V \cup \overline{V}, \phi(\ell) = \texttt{true}\} \cup \{z_j : z_j \in Z, \phi(\ell_{j,3}) = \texttt{false}\}.$$

Note that X contains s, but not t; we are going to show that it is a solution for I_{RS}. Since ϕ maps exactly one literal in $\{v, \overline{v}\}$ to true for each $v \in V$, by $R \cup \{s\} \subseteq X$ we get that

$$\sum_{S \in \Pi_v} \mathsf{dist}_{s,t}(X, S) = |(\{s, v, \overline{v}\} \cup R) \setminus X| = |\{v, \overline{v}\} \setminus X| = 1 \qquad \text{and}$$
$$\sum_{S \in \Pi_{\overline{v}}} \mathsf{dist}_{s,t}(X, S) = |(\{v, \overline{v}, t\}) \cap X| = |\{v, \overline{v}\} \cap X| = 1.$$

For the distance of X from the set families associated with some clause $C_j \in \mathcal{C}$, by the validity of ϕ we obtain

$$\sum_{S \in \Pi_{C_j}} \mathsf{dist}_{s,t}(X, S) = |(C_j \cup \{t\}) \cap X| = 1;$$

$$\begin{aligned}
\sum_{S \in \Pi^{\alpha}_{C_j}} \mathsf{dist}_{s,t}(X, S) &= \min\{|S^{\alpha,1}_j \setminus X|, |S^{\alpha,1}_j \cap X|\} + \min\{|S^{\alpha,2}_j \setminus X|, |S^{\alpha,2}_j \cap X|\} \\
&= \min\{|\{\ell_{j,1}, z_j\} \setminus X|, |\{\ell_{j,1}, z_j\} \cap X|\} \\
&\qquad\qquad + \min\{|\{\ell_{j,2}, r_{j,2}\} \setminus X|, |\{\ell_{j,2}, r_{j,2}\} \cap X|\} \\
&= \left\{\begin{array}{ll} \min\{0,2\} + \min\{1,1\} = 1 & \text{if } \phi(\ell_{j,1}) = \texttt{true} \\ \min\{1,1\} + \min\{0,2\} = 1 & \text{if } \phi(\ell_{j,2}) = \texttt{true} \\ \min\{2,0\} + \min\{1,1\} = 1 & \text{if } \phi(\ell_{j,3}) = \texttt{true} \end{array}\right\} = 1;
\end{aligned}$$

$$\begin{aligned}
\sum_{S \in \Pi^{\beta}_{C_j}} \mathsf{dist}_{s,t}(X, S) &= \min\{|S^{\beta,1}_j \setminus X|, |S^{\beta,1}_j \cap X|\} + \min\{|S^{\beta,2}_j \setminus X|, |S^{\beta,2}_j \cap X|\} \\
&= \min\{|\{r_{j,1}, z_j\} \setminus X|, |\{r_{j,1}, z_j\} \cap X|\} \\
&\qquad\qquad + \min\{|\{\ell_{j,3}, r_{j,3}\} \setminus X|, |\{\ell_{j,3}, r_{j,3}\} \cap X|\} \\
&= \left\{\begin{array}{ll} \min\{0,2\} + \min\{1,1\} = 1 & \text{if } \phi(\ell_{j,1}) = \texttt{true} \\ \min\{0,2\} + \min\{1,1\} = 1 & \text{if } \phi(\ell_{j,2}) = \texttt{true} \\ \min\{1,1\} + \min\{0,2\} = 1 & \text{if } \phi(\ell_{j,3}) = \texttt{true} \end{array}\right\} = 1.
\end{aligned}$$

Hence, X satisfies constraint (4) for each set family, and thus is a solution for I_{RS}.
We prove the other direction of the claim in the full version of our paper [17]. ◀

Using Theorem 18, it is not hard to show that Robust Separation remains NP-hard for any constant $d \geq 1$.

▶ **Lemma 19** ($\star$). *Robust Separation is* NP*-hard for each constant $d \geq 1$.*

▶ **Corollary 20.** *Robust Submodular Minimizer is* NP*-hard for each constant $d \geq 1$.*

4.2 NP-hardness for a constant $k \geq 3$

In this section we prove that Robust Separation, and hence, Robust Submodular Minimizer is NP-hard even for $k = 3$. To this end, we are going to define another intermediary problem. First consider the Most Balanced Minimum Cut problem, proved to be NP-complete by Bonsma [3]. The input of this problem is an undirected graph $G = (V, E)$ with two distinguished vertices, s and t, and a parameter ℓ. The task is to decide whether there exists a minimum (s,t)-cut $X \subseteq V$ in G such that $\min\{|X|, |V \setminus X|\} \geq \ell$; recall that a set of vertices $X \subseteq V$ is a minimum (s,t)-cut in the *undirected* graph G if $s \in X, t \notin X$ and subject to this, the value $|\delta(X)|$, i.e., the number of edges between X and $V \setminus X$, is minimized.

Instead of the Most Balanced Minimum Cut problem, it will be more convenient to use a variant that we call Perfectly Balanced Minimum Cut where we seek a minimum (s,t)-cut that contains exactly half of the vertices. Formally, its input is an undirected graph $G = (V, E)$ with two distinguished vertices, s and t, and its task is to find a minimum (s,t)-cut X with $|X| = |V|/2$. Since Most Balanced Minimum Cut can be reduced to Perfectly Balanced Minimum Cut by simply adding a sufficient number of isolated vertices, we obtain the following.

▶ **Lemma 21** ($\star$). *Perfectly Balanced Minimum Cut is* NP*-complete.*

▶ **Theorem 22** ($\star$). *Robust Separation is NP-hard even when $k = 3$.*

Proof. We present a reduction from the Perfectly Balanced Minimum Cut problem. Let $I = (G, s, t)$ be our input instance where $G = (V, E)$. Clearly, we may assume that $|V|$ is even, as otherwise I is trivially a "no"-instance. First we compute the number of edges in a minimum (s,t)-cut using standard flow techniques; let δ^* denote this value, that is, $\delta^* = \min_{Y: s \in Y \subseteq V \setminus \{t\}} |\delta(Y)|$.

Second, we modify G in order to ensure that there are at least $2\delta^* + 2$ vertices in the graph; if this holds already for G, then we set $G' = G$. Otherwise, we construct a new graph $G' = (V', E')$ by adding two sets of vertices, A_s and A_t, to the graph with $|A_s| = |A_t| = \lceil (2\delta^* + 2 - |V|)/2 \rceil$, and connecting each vertex in A_s to s, as well as each vertex in A_t to t, with an edge. Observe that all minimum (s,t)-cuts in G' contain A_s and are disjoint from A_t. Moreover, any minimum (s,t)-cut X in G corresponds to a minimum (s,t)-cut $X \cup A_S$ in G' and vice versa. Thus, $I' = (G', s, t)$ is an instance of Perfectly Balanced Minimum Cut equivalent with I. Let $2n + 2$ denote the number of vertices in G', so that $\tilde{V} := V' \setminus \{s, t\}$ has $2n$ vertices. By our choice of $|A_s| = |A_t|$, we know that the number of vertices in G' is $|V'| = 2n + 2 \geq |V| + (2\delta^* + 2 - |V|) = 2\delta^* + 2$, as promised.

Let us construct an instance J of Robust Separation. We define our universe U as follows. For each $v \in V'$ we introduce a set $P(v) = \{\hat{v}\} \cup \{v^u : uv \in E\}$, and we additionally define a copy $V^* = \{v^* : v \in V\}$ of V, a set R of size $|R| = n - \delta^*$, and a copy $R' = \{r' : r \in R\}$ of R. Thus, we have

$$U = \bigcup_{v \in V'} P(v) \cup V^* \cup R \cup R'.$$

We set s^* and t^*, both in V^*, as our two distinguished vertices.

We define our three families for J as follows:

$$\begin{aligned}
&\Pi_1 = \{S_1\} && \text{where } S_1 = V^* \setminus \{t^*\} \cup R \cup P(s);\\
&\Pi_2 = \{S_2\} \cup \{S_2^v : v \in \tilde{V}\} && \text{where } S_2 = V^* \setminus \{s^*\} \cup R' \cup P(t),\\
& && \qquad S_2^v = P(v) \quad \forall v \in \tilde{V};\\
&\Pi_3 = \{S_3^v : v \in \tilde{V}\} \cup \{S_3^e : e \in E'\} \cup \{S_3^r : r \in R\} && \text{where } S_3^v = \{\hat{v}, v^*\} \quad \forall v \in \tilde{V},\\
& && \qquad S_3^e = \{u^v, v^u\} \ \forall e = uv \in E',\\
& && \qquad S_3^r = \{r, r'\} \quad \forall r \in R.
\end{aligned}$$

Thus, Π_1 contains only a single set, Π_2 contains $|\tilde{V}|+1$ pairwise disjoint sets, and Π_3 contains $|\tilde{V}| + |E'| + |R|$ pairwise disjoint sets. We finish the definition of our instance J by setting $d = n$ as our threshold, so that $J = (U, s^*, t^*, \Pi_1, \Pi_2, \Pi_3, n)$.

We claim that G' admits a minimum (s,t)-cut containing exactly $n+1$ vertices if and only if J is a "yes"-instance of Robust Separation. The proof of this claim can be found in the full version of our paper [17]. ◀

Clearly, we can increase the value of parameter k without changing the solution set of our instance of Robust Separation by repeatedly adding a copy of, say, the first set family Π_1. Using also Lemma 17, we have the following easy consequences of Theorem 22:

▶ **Corollary 23.** *Robust Separation is NP-hard for each constant $k \geq 3$.*

▶ **Corollary 24.** *Robust Submodular Minimizer is NP-hard for each constant $k \geq 3$.*

5 Conclusion

In this paper, we studied the computational complexity of Robust Submodular Minimizer, and provided a complete computational map of the problem with respect to the parameters k and d, offering dichotomies for the case when one of these parameters is a constant, and giving an FPT algorithm for the combined parameter (k,d). Regarding the case when one of the functions f_i has only polynomially bounded minimizers, there are a few questions left open: First, what is the computational complexity of this variant when parameterized by k? Second, is there an algorithm for this case with running time $2^{O(d)}|I|^{O(1)}$ on some instance I instead of the running time $2^{O(d^2)}|I|^{O(1)}$ we obtained based on the algorithm for Theorem 13?

We remark that our algorithmic results can be adapted in a straightforward way to a slightly generalized problem: given k submodular functions $f_1, \ldots, f_k$ with non-negative integers $d_1, \ldots, d_k$, we aim to find a set X such that, for each $i \in [k]$, there exists some set $Y_i \in \arg\min f_i$ with $|X \triangle Y_i| \leq d_i$ for each $i \in [k]$. As mentioned in Section 1.2, Robust Submodular Minimizer is related to recoverable robustness. We can consider the robust recoverable variant of submodular minimization: given submodular functions $f_0, f_1, \ldots, f_k$, we aim to find a set X that minimizes

$$f_0(X) + \max_{i \in [k]} \min_{Y_i : |Y_i \triangle X| \leq d} f_i(X_i).$$

The optimal value is lower-bounded by $f_0(Y_0) + \max_{i \in [k]} f_i(Y_i)$ where $Y_i \in \arg\min f_i$ for each $i \in \{0, 1, \ldots, k\}$. Our results imply that we can decide efficiently whether the optimal value attains this lower bound or not, when d and k are parameters, or when f_0 has polynomially many minimizers.

References

1 Garrett Birkhoff. Rings of sets. *Duke Math. J.*, 3(3):443–454, 1937. doi:10.1215/S0012-7094-37-00334-X.

2 Hans-Joachim Böckenhauer, Karin Freiermuth, Juraj Hromkovic, Tobias Mömke, Andreas Sprock, and Björn Steffen. Steiner tree reoptimization in graphs with sharpened triangle inequality. *J. Discrete Algorithms*, 11:73–86, 2012. doi:10.1016/J.JDA.2011.03.014.

3 Paul Bonsma. Most balanced minimum cuts. *Discrete Applied Mathematics*, 158:261–276, 2010. doi:10.1016/j.dam.2009.09.010.

4 Nicolas Boria and Vangelis Th. Paschos. Fast reoptimization for the minimum spanning tree problem. *J. Discrete Algorithms*, 8(3):296–310, 2010. doi:10.1016/J.JDA.2009.07.002.

5 Christina Büsing. Recoverable robust shortest path problems. *Networks*, 59(1):181–189, 2012. doi:10.1002/NET.20487.

6 Li Chen, Rasmus Kyng, Yang P. Liu, Richard Peng, Maximilian Probst Gutenberg, and Sushant Sachdeva. Almost-linear-time algorithms for maximum flow and minimum-cost flow. *Commun. ACM*, 66(12):85–92, 2023. doi:10.1145/3610940.

7 Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized algorithms*. Springer, Cham, 2015. doi:10.1007/978-3-319-21275-3.

8 Mitre Costa Dourado, Dirk Meierling, Lucia Draque Penso, Dieter Rautenbach, Fábio Protti, and Aline Ribeiro de Almeida. Robust recoverable perfect matchings. *Networks*, 66(3):210–213, 2015. doi:10.1002/NET.21624.

9 Rod G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, London, 2013. doi:10.1007/978-1-4471-5559-1.

10 Dennis Fischer, Tim A. Hartmann, Stefan Lendl, and Gerhard J. Woeginger. An investigation of the recoverable robust assignment problem. In Petr A. Golovach and Meirav Zehavi, editors, *16th International Symposium on Parameterized and Exact Computation, IPEC 2021, September 8-10, 2021, Lisbon, Portugal*, volume 214 of *LIPIcs*, pages 19:1–19:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPICS.IPEC.2021.19.

11 Moti Frances and Ami Litman. On covering problems of codes. *Theory Comput. Syst.*, 30(2):113–119, 1997. doi:10.1007/s002240000044.

12 Marc Goerigk, Stefan Lendl, and Lasse Wulf. On the recoverable traveling salesman problem. *CoRR*, abs/2111.09691, 2021. arXiv:2111.09691.

13 Felix Hommelsheim, Nicole Megow, Komal Muluk, and Britta Peis. Recoverable robust optimization with commitment. *CoRR*, abs/2306.08546, 2023. arXiv:2306.08546.

14 Mikita Hradovich, Adam Kasperski, and Pawel Zielinski. Recoverable robust spanning tree problem under interval uncertainty representations. *J. Comb. Optim.*, 34(2):554–573, 2017. doi:10.1007/S10878-016-0089-6.

15 Takehiro Ito, Naonori Kakimura, Naoyuki Kamiyama, Yusuke Kobayashi, and Yoshio Okamoto. A parameterized view to the robust recoverable base problem of matroids under structural uncertainty. *Oper. Res. Lett.*, 50(3):370–375, 2022. doi:10.1016/J.ORL.2022.05.001.

16 Naonori Kakimura, Naoyuki Kamiyama, Yusuke Kobayashi, and Yoshio Okamoto. Submodular reassignment problem for reallocating agents to tasks with synergy effects. *Discret. Optim.*, 44(Part):100631, 2022. doi:10.1016/j.disopt.2021.100631.

17 Naonori Kakimura and Ildikó Schlotter. Parameterized complexity of submodular minimization under uncertainty. *CoRR*, abs/2404.07516, 2024. arXiv:2404.07516.

18 Stefan Kratsch, Shaohua Li, Dániel Marx, Marcin Pilipczuk, and Magnus Wahlström. Multi-budgeted directed cuts. *Algorithmica*, 82(8):2135–2155, 2020. doi:10.1007/S00453-019-00609-1.

19 Thomas Lachmann, Stefan Lendl, and Gerhard J. Woeginger. A linear time algorithm for the robust recoverable selection problem. *Discret. Appl. Math.*, 303:94–107, 2021. doi:10.1016/J.DAM.2020.08.012.

20 Yin Tat Lee, Aaron Sidford, and Sam Chiu-wai Wong. A faster cutting plane method and its implications for combinatorial and convex optimization. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17–20 October, 2015*, pages 1049–1065, 2015. doi:10.1109/FOCS.2015.68.

21 Stefan Lendl, Britta Peis, and Veerle Timmermans. Matroid bases with cardinality constraints on the intersection. *Math. Program.*, 194(1):661–684, 2022. doi:10.1007/S10107-021-01642-1.

22 Christian Liebchen, Marco E. Lübbecke, Rolf H. Möhring, and Sebastian Stiller. The concept of recoverable robustness, linear programming recovery, and railway applications. In Ravindra K. Ahuja, Rolf H. Möhring, and Christos D. Zaroliagis, editors, *Robust and Online Large-Scale Optimization: Models and Techniques for Transportation Systems*, volume 5868 of *Lecture Notes in Computer Science*, pages 1–27. Springer, 2009. doi:10.1007/978-3-642-05465-5_1.

23 Jérôme Monnot. A note on the traveling salesman reoptimization problem under vertex insertion. *Inf. Process. Lett.*, 115(3):435–438, 2015. doi:10.1016/J.IPL.2014.11.003.

24 Kazuo Murota. *Discrete Convex Analysis*. SIAM, 2003. doi:10.1137/1.9780898718508.

25 James B. Orlin. A faster strongly polynomial time algorithm for submodular function minimization. *Math. Program.*, 118(2):237–251, 2009. doi:10.1007/s10107-007-0189-2.

26 Jean-Claude Picard and Maurice Queyranne. On the structure of all minimum cuts in a network and applications. *Mathematical Programming Studies*, 13:8–16, 1980. doi:10.1007/BFb0120902.

27 Thomas J Schaefer. The complexity of satisfiability problems. In *Proceedings of the Tenth ACM Symposium on Theory of Computing (STOC '78)*, pages 216–226. ACM, 1978. doi:10.1145/800133.804350.

28 Alexander Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*. Springer, Berlin, 2003.