Posimodular Function Optimization

Magnús M. Halldórsson1 · Toshimasa Ishii2 · Kazuhisa Makino3 · Kenjiro Takazawa4

Received: 17 April 2019 / Accepted: 29 November 2021 / Published online: 29 January 2022
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2022

Abstract
A function \( f : 2^V \to \mathbb{R} \) on a finite set \( V \) is posimodular if \( f(X) + f(Y) \geq f(X \setminus Y) + f(Y \setminus X) \), for all \( X, Y \subseteq V \). Posimodular functions often arise in combinatorial optimization such as undirected cut functions. We consider the problem of finding a nonempty subset \( X \) minimizing \( f(X) \), when the posimodular function \( f \) is given by oracle access. We show that posimodular function minimization requires exponential time, contrasting with the polynomial solvability of submodular function minimization that forms another generalization of cut functions. On the other hand, the problem is fixed-parameter tractable in terms of the size \( D \) of the image (or range) of \( f \). In more detail, we show that \( \Omega(2^{0.32n}T_f) \) time is necessary and \( O(2^{0.92n}T_f) \) sufficient, where \( T_f \) denotes the time for one function evaluation and \( n = |V| \). When the image of \( f \) is \( D = \{0, 1, \ldots, d\} \) for integer \( d \), \( O(2^{1.218d}nT_f) \) time is sufficient. We can also generate all sets minimizing \( f \) in time \( 2^{O(d)}n^2T_f \). Finally, we also

An extended abstract of this article was presented in Proceedings of the 15th International Symposium on Algorithms and Data Structures (WADS 2017) [7].

This research was partially supported by Icelandic Research Fund Grants 152679-05 and 174484-05, MEXT KAKENHI Grant Numbers JP24106002, JSPS KAKENHI Grant Numbers JP25280004, JP26280001, JP16K00001 and 20K11699, and JST CREST Grant Number JPMJCR1402, Japan.

Magnús M. Halldórsson
mmh@ru.is

Toshimasa Ishii
ishii@econ.hokudai.ac.jp

Kazuhisa Makino
makino@kurims.kyoto-u.ac.jp

Kenjiro Takazawa
takazawa@hosei.ac.jp

1 ICE-TCS, School of Computer Science, Reykjavik University, Reykjavik, Iceland
2 Graduate School of Economics, Hokkaido University, Sapporo, Japan
3 Research Institute for Mathematical Sciences, Kyoto University, Kyoto, Japan
4 Faculty of Science and Engineering, Hosei University, Tokyo, Japan
consider the problem of maximizing a given posimodular function, showing that it requires at least $2^{n-1}T_f$ time in general, while it has time complexity $\Theta((n^d-1)T_f)$ when $D = \{0, 1, \ldots, d\}$ is the image of $f$, for integer $d = O(n^{1/4})$.

**Keywords** Posimodular function · Algorithm · Lower bounds

**Mathematics Subject Classification** 68W40 · 68Q25

1 Introduction

Let $V$ denote a finite set with $n = |V|$. A set function $f : 2^V \rightarrow \mathbb{R}$ is called **posimodular** if

$$f(X) + f(Y) \geq f(X \setminus Y) + f(Y \setminus X) \quad (1.1)$$

for all $X, Y \subseteq V$, where $\mathbb{R}$ denotes the set of all reals. Posimodularity is a fundamental property in combinatorial optimization [6,8,14,16–18] and is typically the key for efficiently solving undirected network optimization and related problems, since cut functions for undirected networks are posimodular. In comparison, cut functions for directed networks are not posimodular.

There are numerous network optimization problems that are better solvable in undirected variants than directed ones, where in fact only the undirected versions satisfy posimodularity. One example is the local edge-connectivity augmentation problem, which is polynomially solvable in undirected networks but NP-hard in directed networks [5]. Similarly, undirected versions of the source location problem with uniform demands or with uniform costs can be solved in polynomial time, [1,20], while the directed versions are NP-hard [9].

More generally, posimodularity often helps in designing optimization algorithms with better time complexity or approximation ratio. For example, the current fastest algorithm for minimizing a submodular function requires $O(n^3 \log^2 n \cdot T_f + n^4 \log O(1) n)$ time [11], where a set function $f : 2^V \rightarrow \mathbb{R}$ is called **submodular** if

$$f(X) + f(Y) \geq f(X \cap Y) + f(X \cup Y) \quad (1.2)$$

for all $X, Y \subseteq V$, and $T_f$ denotes the time needed to evaluate the function value $f(X)$ for a given $X \subseteq V$. In comparison, there is an algorithm for minimizing a submodular and posimodular function that runs in $O(n^3 T_f)$ time [15]. Another example is the submodular multiway partition problem, which is a generalization of the graph multiway cut problems. It is 2-approximable in polynomial time, while the symmetric submodular multiway partition problem is 1.5-approximable [2], where a set function $f : 2^V \rightarrow \mathbb{R}$ is called **symmetric** if $f(X) = f(V \setminus X)$ holds for any $X \subseteq V$. We remark that a function is symmetric posimodular if and only if it is symmetric submodular, since the symmetricity of $f$ implies that $f(X) + f(Y) = f(V \setminus X) + f(Y)$ and $f(X \setminus Y) + f(Y \setminus X) = f((V \setminus X) \cup Y) + f((V \setminus X) \cap Y)$.
These phenomena can be partially explained by the following three structural properties on posimodular functions. The first structural property is used under the name of uncrossing techniques. There are many variants of partition problems that ask for a partition \(\{V_1, V_2, \ldots, V_k\}\) of \(V\) minimizing \(\sum_{i=1}^{k} f(V_i)\), for a given set function \(f\). This includes the graph multiway cut problem, the graph \(k\)-way cut problem, and the submodular multiway partition problem. If \(f\) is posimodular, then after obtaining a family \(\{V'_1, V'_2, \ldots, V'_l\}\) of subsets that covers \(V\) but may not be disjoint, we can apply uncrossing techniques to obtain a partition \(\{V_1, V_2, \ldots, V_k\}\) of \(V\) without increasing the cost (i.e., \(\sum_{i=1}^{k} f(V_i) \leq \sum_{i=1}^{k} f(V'_i)\)). This is because the posimodularity of \(f\) implies that \(f(X) + f(Y) \geq \min\{f(X) + f(Y\setminus X), f(Y) + f(X\setminus Y)\}\) for any two sets \(X\) and \(Y\). Indeed, this uncrossing technique results in a better approximation ratio for the symmetric submodular multiway partition problem than the (non-symmetric) submodular multiway partition problem [2] (recall that a symmetric submodular function is posimodular). Similar uncrossing techniques have been utilized in other partition problems [12,19].

The second structural property concerns extreme sets. A subset \(X\) of \(V\) is called **extreme** if every nonempty proper subset \(Y\) of \(X\) satisfies \(f(Y) > f(X)\). It is known that when \(f\) is posimodular, the family \(\mathcal{X}(f)\) of extreme sets is **laminar** (i.e., every two members \(X\) and \(Y\) of \(\mathcal{X}(f)\) satisfy \(X \cap Y = \emptyset, X \subseteq Y, \text{ or } X \supseteq Y\)). This is because, if \(X, Y \in \mathcal{X}(f)\) would satisfy \(X \cap Y, X\setminus Y, Y\setminus X \neq \emptyset\), then we have \(f(X) + f(Y) \geq f(X\setminus Y) + f(Y\setminus X) > f(X) + f(Y)\), a contradiction. The family \(\mathcal{X}(f)\) of extreme sets for an undirected cut function \(f\) represents the connectivity structure of a given network and helps to design many efficient network algorithms [10,22]. For example, the undirected source location problem with uniform demands can be solved in \(O(n)\) time, if the family \(\mathcal{X}(f)\) is known in advance, where \(n\) corresponds to the number of vertices in the network [13]. In fact, \(\mathcal{X}(f)\) can be computed in \(O(n(m + n \log n))\) time for any undirected cut function [13], where \(m\) denotes the number of edges in the network. We remark that \(\mathcal{X}(f)\) can be found in \(O(n^3T_f)\) time if \(f\) is posimodular and submodular [14].

The third structural property concerns solid sets, where a subset \(X\) of \(V\) is said to be **\(v\)-solid** for an element \(v \in V\), if \(v \in X\) and every nonempty proper subset \(Y\) of \(X\) that contains \(v\) satisfies \(f(Y) > f(X)\). Let \(\mathcal{S}(f)\) denote the family of all solid sets, i.e., \(\mathcal{S}(f) = \bigcup_{v \in V}\{X : X\text{ is }v\text{-solid}\}\). It is known [18] that the family \(\mathcal{S}(f)\) forms a tree hypergraph if \(f\) is posimodular. Similarly to the previous case for \(\mathcal{X}(f)\), if a host tree \(T\) of \(\mathcal{S}(f)\) is known in advance, this structure enables us to construct a polynomial time algorithm for the minimum transversal problem for posimodular functions \(f\) (see definition in Sec. 3.2.2), which is an extension of the undirected source location problem with uniform costs [20] and the undirected external network problem [21]. If \(f\) is in addition submodular, a host tree \(T\) can be computed in polynomial time.

We remark that the above structural properties of \(\mathcal{X}(f)\) and \(\mathcal{S}(f)\) follow from the posimodularity of \(f\), while the submodularity provides efficiency in computing such structures.

To the best of our knowledge, all previous results for posimodular optimization make use of submodularity or symmetry, since undirected cut functions, the most representative posimodular functions, are also submodular and symmetric. Nevertheless, posimodularity is a fundamental property that warrants study of its own, independent
of known applications. In this paper, therefore, we deal with general posimodular functions which are not necessarily submodular or symmetric.

In this paper, we focus on the posimodular function minimization defined as follows.

**Posimodular Function Minimization**

Input: A posimodular function \( f : 2^V \rightarrow \mathbb{R} \),

Output: A nonempty subset \( X^* \) of \( V \) such that \( f(X^*) = \min_{X \subseteq V: X \neq \emptyset} f(X) \). \( (1.3) \)

Here an input function \( f \) is given by an oracle that answers \( f(X) \) for a given subset \( X \) of \( V \). We refer to the set \( X^* = \arg \min_{X \subseteq V: X \neq \emptyset} f(X) \) as a minimizer of \( f \) and denote \( \min_f = f(X^*) \). The problem was posed as an open problem on the Egres open problem list [3] in 2010, as negamodular function maximization, where a set function \( f \) is negamodular, if \( -f \) is posimodular. We also consider the posimodular function maximization, as submodular function maximization has been intensively studied in recent years.

### 1.1 Our Contributions

The main results obtained in this paper can be summarized as follows.

1. **Intractability** We show that any algorithm for posimodular function minimization requires \( \Omega(2^{0.32n} T_f) \) time, while there exists an \( O(2^{0.92n} T_f) \)-time algorithm.
2. **Tractability on small images** We consider functions \( f : 2^V \rightarrow D \) with a restricted image. Our main positive result is that the problem is fixed-parameter tractable in terms of the image size \( |D| \). Specifically, we give an algorithm with complexity \( O(2^{|D|} n T_f) \). For the case of the image \( D = \{0, 1, \ldots, d\} \), we obtain an improved bound of \( O(2^{1.218d} n T_f) \). We then extend the parameterized algorithm to generate all minimizers with linear delay, after initial \( 2^{O(|D|)} n^2 T_f \)-time. This is the most technical part of the paper.
3. **Hardness of Maximization** We show that posimodular function maximization requires at least \( 2^{n-1} T_f \) time, and thus only trivial solutions are possible. For image restricted to (a subset of) \( D = \{0, 1, \ldots, d\} \), for \( d = O(n^{1/4}) \), we obtain a tight bound of \( \Theta\left(\binom{n}{d-1} T_f\right) \) on the time complexity.

We also obtain implications for related problems. For instance, we can compute all extreme sets in \( O(|D|2^{|D|} n T_f) \) time, which implies that the source location problem for posimodular functions can be solved in \( O(|D|2^{|D|} n T_f) \) time.

We remark that no complexity-theoretic assumptions are needed for the lower bounds. For related results, Feige et al. [4] showed that at least \( e^{e^2 n/8} \) oracle calls are necessary for obtaining a solution of at least \((1/2 + \epsilon)\) times optimal for symmetric submodular function maximization, which is equivalent to symmetric posimodular function maximization.

The rest of this paper is organized as follows. In Sect. 2, we give the hardness results and a \( o(2^n T_f) \)-time algorithm for posimodular function minimization. In Sect. 3, we consider the case where the image of \( f \) is bounded or given by \( D = \{0, 1, \ldots, d\} \).
and show hardness results and a fixed parameter algorithm in terms of the image size. Sect. 4 treats the posimodular function maximization.

2 General Posimodular Function Minimization

2.1 Hardness Results

Let $V$ be a finite set with $n = |V|$ and $f : 2^V \to \mathbb{R}$ be a posimodular function. Notice that $f$ satisfies

$$f(X) \geq f(\emptyset) \text{ for all } X \subseteq V, \tag{2.1}$$

since $f(X) + f(X) \geq f(\emptyset) + f(\emptyset)$. Throughout the paper, we assume that $f(\emptyset) = 0$, since otherwise, we can replace $f(X)$ by $f(X) - f(\emptyset)$ for all $X \subseteq V$.

In this section, we analyze the number of oracle calls necessary for posimodular function minimization.

Let $g : 2^V \to \mathbb{R}_+$ be the cardinality function defined by $g(X) = |X|$. Clearly, $g$ is posimodular since $g$ is monotone, i.e., $g(X) \geq g(Y)$ holds for all two subsets $X$ and $Y$ of $V$ with $X \supseteq Y$.

For a given positive integer $k$, we construct the family $G_k = \{g\} \cup \{g_S \mid S \subseteq V, |S| = 2k\}$ of functions, where $g_S : 2^V \to \mathbb{R}_+$ is defined by

$$g_S(X) = \begin{cases} 2k - |X| & \text{if } X \subseteq S \text{ and } |X| \geq k + 1, \\ g(X) = |X| & \text{otherwise.} \end{cases}$$

We can see that each $g_S$ is a posimodular function with $S$ as its unique minimizer.

**Claim 1** $g_S$ is posimodular.

**Proof** Note first that $g_S(X) \leq |X|$ for all $X \subseteq V$, since $|X| \geq 2k - |X|$ if $|X| \geq k + 1$. Let $X$ and $Y$ be non-disjoint subsets of $V$. We separately consider the following two cases.

If at least one of $X$ and $Y$ has identical function values for $g_S$ and $g$, say $g_S(X) = g(X)$, then we have $g_S(X) - g_S(X \setminus Y) \geq |X| - |X \setminus Y| = |X \cap Y|$. If $g_S(Y) = g(Y)$ is also satisfied, then we obtain $g_S(Y) - g_S(Y \setminus X) \geq |Y| - |Y \setminus X| = |X \cap Y|$, and hence

$$g_S(X) + g_X(Y) - (g_S(X \setminus Y) + g_S(Y \setminus X)) \geq 2|X \cap Y| \geq 2,$$

which implies that the posimodular inequality (1.1) holds for such $X$ and $Y$. On the other hand, if $g_S(Y) \neq g(Y)$, i.e., $Y \subseteq S$, $|Y| \geq k + 1$, and $g_S(Y) = 2k - |Y|$, then $g_S(Y \setminus X) = 2k - |Y \setminus X|$ or $g_S(Y \setminus X) = |Y \setminus X|$. In the former case, $g_S(Y) - g_S(Y \setminus X) = -|X \cap Y|$. In the latter case, $|Y \setminus X| \leq k$ holds by $Y \setminus X \subseteq S$ and hence $g_S(Y) - g_S(Y \setminus X) = 2k - |Y| - |Y \setminus X| \geq 2k - |Y| - (2k - |Y \setminus X|) = -|X \cap Y|$, which again implies the posimodular inequality (1.1).
If \( g_S(X) \neq g(X) \) and \( g_S(Y) \neq g(Y) \) are satisfied, then we have \( g_S(X) = 2k - |X| \) and \( g_S(Y) = 2k - |Y| \). Since \( |X \setminus Y|, |Y \setminus X| \leq k \), we also have \( g_S(X \setminus Y) = |X \setminus Y| \) and \( g_S(Y \setminus X) = |Y \setminus X| \). Hence, it holds that

\[
g_S(X) + g_S(Y) - (g_S(X \setminus Y) + g_S(Y \setminus X)) = 4k - 2|X \cup Y| \geq 0,
\]

where the last inequality follows from \( X \cup Y \subseteq S \) and \( |S| = 2k \). Therefore the posimodular inequality \((1.1)\) holds. \(\square\)

We show below that exponential number of oracle queries are necessary to distinguish between the posimodular functions in \( G_k \).

**Lemma 1** At least \( \binom{n}{2k} / \binom{n-k-1}{k-1} \) queries are necessary to distinguish among posimodular functions in \( G_k \).

**Proof** We say that a query on a set \( X \) differentiates \( Y \) if \( g_Y(X) \neq g(X) \). For \( X \) to differentiate \( Y \) it must be a subset of \( Y \) \((X \subseteq Y)\) and contain more than \( k \) elements \(|X| \geq k + 1\). Observe that a single query \( X \) differentiates \( \binom{n-|X|}{2k-|X|} \) sets of size \( 2k \), which is maximized when \( |X| = k + 1 \). In order to distinguish among all functions in \( G_k \), the sequence of queries must necessarily differentiate all the \( \binom{n}{2k} \) subsets of size \( 2k \). Hence, the number of queries is at least \( \binom{n}{2k} / \binom{n-k-1}{k-1} \). \(\square\)

**Theorem 1** Every algorithm for posimodular function minimization makes \( \Omega(2^{0.32n}) \) oracle calls in the worst case.

**Proof** Applying Stirling’s formula, we find that

\[
\binom{n}{2k} / \binom{n-k-1}{k-1} \geq c_1 \frac{n^n}{(4k)^k(n-k)^{n-k}},
\]

for some constant \( c_1 > 0 \). When \( n = 5k \), this equals \( c_1 \cdot 1.25^n = \Omega(2^{0.32n}) \). \(\square\)

The above arguments also imply that there is no fixed-parameter algorithm when parameterized by the solution size \(|S|\): for the above instance, any algorithm requires at least \( \frac{c_1}{(2|S|)^{|S|^2/2}} \cdot n^{|S|^2/2} \) oracle calls (note that \(|S| = 2k\)).

### 2.2 \( o(2^n T_f) \)-Time Algorithm

Posimodular function minimization can trivially be solved in \( 2^n T_f \) time, since the number of subsets of \( V \) is \( 2^n \). In this subsection, we give a \( c^n T_f \) time algorithm for the problem, for \( c < 2 \).

**Theorem 2** Posimodular function minimization can be solved in \( O\left( \binom{n}{n/3} n \log n \cdot T_f \right) = O\left( 2^{0.92n} T_f \right) \) time.

For a set \( S \subseteq V \), a set \( X \subseteq S \) is called a splitter with respect to \( S \) if \( f(X \cup \{v\}) > f(X \cup \{u\}) \) for all \( v \in V \setminus S \) and all \( u \in S \setminus X \). Let \( v_1, v_2, \ldots, v_{|V \setminus X|} \) be an ordering of
MinPosimodular Algorithm minimization.

Step 1. Compute a subset $f$ and hence $X$. Assume that no singleton is a minimizer of $f$. Let $S$ be a minimizer of $f$. Then, there exists a splitter with respect to $S$ that is of cardinality at most $\lceil(\lvert S \rvert - 1)/2 \rceil$.

**Proof** Let $Z \subset S$ be a set with maximum $f$-value, among subsets of $S$ of cardinality $\lfloor(\lvert S \rvert + 1)/2 \rfloor$ or $\lceil(\lvert S \rvert + 1)/2 \rceil$. Namely, $\lvert Z \rvert \in \{(\lvert S \rvert + 1)/2, (\lvert S \rvert + 1)/2\}$ and $f(Z) \geq f(Y)$ for any $Y \subseteq S$ with $\lvert Y \rvert \in \{(\lvert S \rvert + 1)/2, (\lvert S \rvert + 1)/2\}$. Let $Z' = S \setminus Z$. We claim that $Z'$ is a splitter with respect to $S$.

Consider an element $v \in V \setminus S$. It holds by posimodularity that $f(Z' \cup \{v\}) + f(S) \geq f(v) + f(S') = f(v) + f(Z)$. Since $\{v\}$ is not a minimizer, we have $f(S) < f(v)$, and hence $f(Z' \cup \{v\}) > f(Z)$.

Now, consider an element $u \in S \setminus Z'$. Observe that $Z' \cup \{u\}$ is in $S$ and is of cardinality $|S| + 1 - |Z|$, which is either $(|S| + 1)/2$ or $(|S| + 1)/2$. Thus, by the definition of $Z$, $f(Z' \cup \{u\}) \leq f(Z)$.

Hence, $Z'$ satisfies the properties of being a splitter with respect to $S$. \qed

Based on Lemma 2, we have the following algorithm for posimodular function minimization.

**Algorithm** MINPOSIMODULAR($f$)

**Step 1.** Compute a subset $S$ with minimum $f(S)$ among the sets $X$ with $|X| \leq \max\{2, n/3\}$ or $|X| \geq 2n/3$. (i.e., $f(S) = \min\{f(X) : |X| \leq \max\{2, n/3\} \text{ or } |X| \geq 2n/3\}$).

**Step 2.** For each set $X \subseteq V$ with $|X| \leq n/3$, execute the following steps (1) and (2).

(1) Let $v_1, v_2, \ldots, v_{|V \setminus X|}$ be an ordering of $V \setminus X$ such that $f(X \cup \{v_1\}) \leq f(X \cup \{v_2\}) \leq \ldots \leq f(X \cup \{v_{|V \setminus X|}\})$.

(2) Let $S'$ be a set with the minimum $f$-value among the sets $\{X \cup \{v_1, v_2, \ldots, v_i\} : 1 \leq i \leq |V \setminus X|\}$. If $f(S') < f(S)$, then let $S := S'$.

**Step 3.** Output $S$ and halt. \qed

The following lemma proves Theorem 2.

**Lemma 3** MINPOSIMODULAR($f$) finds a minimizer of $f$ in $O((\lceil n/3 \rceil)n \log n \cdot T_f)$ time.

**Proof** Clearly, if $n \leq 2$ or there exists a minimizer $S$ of $f$ with $|S| \leq n/3$ or $|S| \geq 2n/3$, then it is found in Step 1.

Consider the case where $n \geq 3$ and any minimizer $S$ satisfies $1 \leq n/3 < |S| < 2n/3$. Then, no singleton is a minimizer of $f$, and hence it follows from Lemma 2 that there exists a splitter $X$ with respect to $S$ with $|X| \leq |S|/2 < n/3$. Recall that as observed after the definition of a splitter, such an $S$ can be found by computing $f(X \cup \{v\})$ for each $v \in V \setminus X$. Hence, we find a minimizer of $f$ in Step 2.
The time complexity of Step 1 of the algorithm $O\left(\binom{n}{n/3}T_f\right)$. For Step 2, each iteration is achieved in $O(n)$ oracle calls but the sorting requires $O(n \log n)$ time. We conservatively bound the combined complexity by $O\left(\binom{n}{n/3}n \log n \cdot T_f\right)$. □

3 Minimization on Small Images

In light of the hardness of minimizing general posimodular functions, we turn our attention to parameterized algorithms. The intractability results still hold in terms of the solution size (cardinality of the minimizer) or the value of the minimum solution. Instead, we treat in this section the parameter $|D|$, the cardinality of the image $D$ of the function $f$. We may then assume that the range of $D$ is the set of integers. We obtain stronger results for the special case $D = \{0, 1, \ldots, d\}$, for an integer $d$.

3.1 Fixed-Parameter Algorithm

We propose a bounded-depth tree search algorithm. Each tree node corresponds to an invocation of a recursive procedure given three disjoint sets: $A$, $B$, and $C$. In each invocation, the algorithm either produces a solution and terminates, or it selects an element $v$ with which it makes two recursive calls: adding $v$ to $A$, and adding $v$ to one of $B$ or $C$. The crucial property maintained is that whenever an element is added to a set, the value of the set increases. It follows immediately that the recursion depth is at most $3|D|$. Since the recursion tree is binary, the number of recursive calls is at most $2^{3|D|}$. (Fig. 1).

The challenge is in showing that a solution can be obtained once no options for recursing remain. The parameter tuple $(A, B, C)$ is valid with respect to a minimizer $S$, if $A$ is disjoint from $S$ while $B$ and $C$ are contained in $S$ ($A \subseteq (V \setminus S) \land B \cup C \subseteq S$). An invocation SOLVE$(A, B, C)$ is valid (for $S$) if $(A, B, C)$ is valid (for $S$). The crucial characterization that we obtain is that once we reach a leaf with a valid tuple, we can easily identify the rest of $S$: namely, $S \setminus (B \cup C)$ is given by those elements in $V \setminus (A \cup B \cup C)$ whose addition to neither $B$ nor $C$ increases the values of those sets.

The algorithm MINPOSIMODULAR- $D(f)$ operates in more detail as follows. It uses a global variable $SS$, initially set as the singleton set of smallest function value, which progressively improves to a minimizer. The algorithm calls a recursive subroutine
Solve with parameters $A$, $B$, and $C$, initially empty sets. Note that the algorithm need not know anything about the image $D$.

**Algorithm 1**  
**MINPOSIMODULAR- $d(f)$**

1: $SS \leftarrow \{v^*\}$, where $v^* = \arg \min_{v \in V} f(v)$ is a singleton of smallest value
2: Call Solve$(\emptyset, \emptyset, \emptyset)$
3: Output $SS$

We say that an element $v$ increases a set $X$ if $f(X \cup \{v\}) > f(X)$. Similarly, a set $X$ increases set $Y$ if $f(Y \cup X) > f(Y)$. Let inc$A(v)$ denote the predicate that element $v$ increases $A$, i.e., $f(A \cup \{v\}) > f(A)$.

Similarly, let inc$BC(v)$ denote the predicate that $v$ increases $B$ or $v$ increases $C$ (or both).

Let TRY$(v)$ denote the following inline macro: Recursively call Solve$(A \cup \{v\}, B, C)$; then, call Solve$(A, B \cup \{v\}, C)$ if $v$ increases $B$, and otherwise call Solve$(A, B, C \cup \{v\})$.

**Algorithm 2**  
Procedure Solve$(A, B, C)$

**Require:** Disjoint subsets $A, B, C$ of $V$; $SS$ is a global variable representing the current champion
1: if $\exists v \in V \setminus (A \cup B \cup C)$, inc$A(v) \land$ inc$BC(v)$ then
2: TRY$(v)$
3: else
4: $\hat{S} \leftarrow B \cup C \cup \{v \in V \setminus A : \neg$inc$BC(v)\}$
5: if $f(\hat{S}) < f(SS)$ then
6: $SS \leftarrow \hat{S}$

The core tools for our results are the following two lemmas. They show that if given a set that is either properly inside or outside a minimizer, then adding elements from the other side must increase the value of the set. This helps us find sets with successively larger values; once we obtain sets of maximum value, the remaining elements can be quickly assigned, based on the characterization of the lemmas.

A subset $X$ is called *locally minimal* if $f(X) < f(X \setminus \{v\})$ holds for every $v \in X$.

**Lemma 4** Let $S$ be a locally minimal set (for $f$) with $|S| \geq 2$. If $A \subseteq V \setminus S$ and $v$ is an element in $S$, then $v$ increases $A$.

**Proof** By posimodularity, $f(S) + f(A \cup \{v\}) \geq f(S \setminus \{v\}) + f(A)$. By the local minimality of $S$, $f(S) < f(S \setminus \{v\})$, which implies that $f(A \cup \{v\}) > f(A)$, as claimed. $\square$

In the context of a function with a limited range on the integers, we say that $v$ increases $X$ by two if $f(X \cup \{v\}) \geq f(X) + 2$. When dealing with a general range, the meaning reverts to just increases $X$.

**Lemma 5** Let $S$ be a set and $v \in V \setminus S$ be an element such that $f(S) < f(v)$. If $B$ and $C$ are disjoint sets within $S$, then $v$ increases $B$ or $C$. If, additionally, $f$ has range $\{0, 1, \ldots, d\}$, then either $v$ increases $B$, or $v$ increases $C$ by two.
Proof By posimodularity, \( f(B \cup \{v\}) + f(S) \geq f(v) + f(S \setminus B) \). From the assumption that \( f(v) \geq f(S) + 1 \), we have that \( f(B \cup \{v\}) \geq f(S \setminus B) + 1 \). By the same argument, \( f(C \cup \{v\}) \geq f(S \setminus C) + 1 \). Combining these two inequalities,

\[
f(B \cup \{v\}) + f(C \cup \{v\}) \geq f(S \setminus B) + f(S \setminus C) + 2. \tag{3.1}
\]

Furthermore, it also follows from posimodularity that

\[
f(S \setminus B) + f(S \setminus C) \geq f((S \setminus B) \setminus (S \setminus C)) + f((S \setminus C) \setminus (S \setminus B)) = f(C) + f(B).
\]

Combining this with (3.1), we get that \( f(B \cup \{v\}) + f(C \cup \{v\}) \geq f(B) + f(C) + 2 \). Hence, the lemma. \( \square \)

We first argue validity.

Lemma 6 For every minimizer \( S \), there is an invocation \( \text{SOLVE}(A, B, C) \) valid for \( S \) that is a leaf in the recursion tree.

Proof Let \( (A, B, C) \) be an invocation of \( \text{SOLVE} \) valid for \( S \) that is maximally deep, i.e., none of its children are valid. It exists since \( (\emptyset, \emptyset, \emptyset) \) is valid for \( S \). Suppose that it is not a leaf. Then, by the algorithm there is an element \( v \in V \setminus (A, B, C) \) for which two recursion calls are made: (a) where \( v \) is added to \( A \), and (b) where \( v \) is added to either \( B \) or \( C \). Since \( v \) must be either in \( S \) (in which (b) is valid) or in \( V \setminus S \) (in which (a) is valid), one of them is valid, which is a contradiction. \( \square \)

We can now derive the correctness and time complexity of the algorithm using the above lemmas. A minimizer is minimal if it does not properly contain a minimizer.

Theorem 3 The algorithm \( \text{MINPOSIMODULAR-}(f) \) finds a (minimal) minimizer of \( f \) in time \( 2^{3|D|} n T_f \).

Proof Let \( S \) be a minimal minimizer. By Lemma 6, there is a leaf in the recursion tree with a valid tuple \( (A, B, C) \) for \( S \). Since the recurrence ended at this leaf, each element \( v \) outside \( A \cup B \cup C \) either fails \( \text{inc}(v) \) or fails \( \text{incBC}(v) \) (or both). Each element \( v \) failing \( \text{inc}(v) \) must be in \( V \setminus S \), by Lemma 4, while an element \( w \) failing \( \text{incBC}(w) \) must be in \( S \), by Lemma 5. Thus, \( S \) is necessarily given by \( S = B \cup C \cup \{v \in V \setminus A : \neg \text{incBC}(v)\} \), as computed in line 4 of \( \text{SOLVE} \).

The depth of the recursion is at most \( |A| + |B| + |C| \leq 3|D| \), and hence the number of recursive calls is at most \( 2^{3|D|} \). Each call performs linear number of queries and linear computation, and thus the time complexity follows. \( \square \)

When the image is a small range, a more nuanced analysis is possible.

Observation 1 Assume that \( D = \{0, 1, \ldots, d\} \) and there is no singleton minimizer. For a valid tuple \( (A, B, C) \), it holds that

\[
|A| + \max(|B|, 2|C|) \leq d.
\]
Proof For each \( v \in V \), it holds that \( f(v) \geq \min_f + 1 \), since \( v \) is not a minimizer. Each element added to \( A \) or \( B \) increases the set, and thus \( f(A) \geq |A| + \min_f \) and \( f(B) \geq |B| + \min_f \). Elements added to \( C \) increase it by two, and thus \( f(C) \geq 2|C| \). Also, by posimodularity, for \( X \in \{B, C\} \),

\[
f(X) + f(A) \leq f(S) + f(A \cup (S\setminus X)) \leq \min_f + d. 
\]

Combined, we obtain that \(|A| + |B| \leq d - \min_f \leq d \) and \(|A| + 2|C| \leq d \).

This yields better time complexity.

**Theorem 4** MinPosimodular- \( d(f) \) finds a minimizer of a posimodular function \( f : 2^V \to D \) in time \( O(2^{1.218d} nT_f) \), when \( D = \{0, 1, \ldots, d\} \).

Proof In light of Observation 1, we modify the algorithm Solve so that it truncates the search whenever \(|A| + \max(|B|, 2|C|) > d\). We assume for simplicity that the algorithm is given the value \( d \) (although it could be adapted to work with increasing estimates of \( d \), resulting in the same asymptotic time complexity).

Adding the two bounds of Obs. 1 yields that \( 3|A| + 2|B| + 2|C| \leq 3d \). A straightforward bound on the recursion depth is then \(|A| + |B| + |C| \leq 3d/2 \), for a time complexity of \( O(2^{3d/2} nT_f) \). A more refined bound follows from observing that each leaf \((A, B, C)\) of the recursion tree is at depth at most \((3d - |A|)/2\) and is reached using \( t = |A| \leq d \) left edges. The number of leaves is then bounded by

\[
\sum_{t=0}^{d} \left( \left\lfloor \frac{3d-t}{2} \right\rfloor \right) \sim 2^{1.218d}, 
\]

where we find numerically that the largest term occurs for \( t \sim 0.531d \).

3.1.1 Finding All Minimal Minimizers

We can extend the algorithm to find all minimal minimizers, i.e., all minimizers \( S \) such that no proper subset of \( S \) is also a minimizer.

**Corollary 1** All minimal minimizers can be found in time \( 2^{3|D|} nT_f \), and in time \( O(2^{1.218d} nT_f) \) when \( D = \{0, 1, \ldots, d\} \).

Proof First run MinPosimodular- \( d(f) \) to determine the value of \( \min_f \). Then compute \( Z = \{v : f(v) = \min_f\} \), the set of singleton minimizers. Every minimal minimizer either consists of an element in \( Z \) or is disjoint from \( Z \). Thus, to find the latters ones, we run MinPosimodular- \( d(f) \) on the universe \( V' = V \setminus Z \) (which by definition contains no singleton minimizers).

The correctness and time complexity arguments of Theorem 3, Observation 1, and Theorem 4 hold for every minimal minimizer (of \( V' \)).

This shows that the number of non-singleton minimal minimizers is at most \( 2^{3|D|} \). Note that the algorithm might output the same minimizer more than once.
3.2 Related Problems: Extreme Sets, Transversals, Approximation

3.2.1 Extreme Sets

We first show that the family $\mathcal{X}(f)$ of all extreme sets can be obtained as an application of Theorem 4. Recall that a subset $X$ of $V$ is called extreme if every nonempty proper subset $Y$ of $X$ satisfies $f(Y) > f(X)$. By definition, $\mathcal{X}(f)$ contains all singletons $\{v\}$, $v \in V$, and any extreme set $X$ with $|X| \geq 2$ is locally minimal.

Consider the subfamily $\mathcal{X}_p(f) = \{X: X $is extreme with $f(X) = p\}$ of extreme sets with value $p$, for $p \in D$. The singleton sets in $\mathcal{X}_p(f)$ are given precisely by $V_p = \{v \in V: f(v) = p\}$, while the non-singleton sets in $\mathcal{X}_p(f)$ can only contain elements $v$ with $f(v) > p$. Thus, to find non-singleton sets in $\mathcal{X}_p(f)$, we can restrict attention to the universe $V_{\geq p} = \{v \in V: f(v) \geq p\}$.

We observe that $\text{MinPosimodular-}d(f)$ restricted to $V_{\geq p}$ identifies all non-singleton sets in $\mathcal{X}_p(f)$, since Lemmas 4 and 5 hold for all non-singleton extreme sets with value at most $p$. Thus, by iterating over the $|D|$ possible values of $p$, we can produce all extreme sets in $O(|D|^{23}|D|^nT_f)$ time.

In summary, we have the following result.

**Corollary 2** For a posimodular function $f: 2^V \rightarrow D$, we can compute the family $\mathcal{X}(f)$ of all extreme sets of $f$ in $O(|D|^{23}|D|^nT_f)$ time.

3.2.2 Minimum Traversal

Consider the following problem:

**Minimum Transversal** ($f, c, r$)

Input: A posimodular function $f: 2^V \rightarrow \mathbb{R}$, a cost function $c: V \rightarrow \mathbb{R}$, and a demand function $r: 2^V \rightarrow \mathbb{R}$.

Output: A nonempty subset $S$ of $V$ minimizing $\sum_{v \in S} c(v)$ such that $f(X) \geq r(X)$ for every nonempty subset $X \subseteq V \setminus S$.

We remark that undirected source location problem defined as (3.3) is a special case of this problem where $f$ is a cut function in an undirected graph and $r(X) = \max\{r'(v) \mid v \in X\}$ for all nonempty subsets $X$ of $V$.

**Source Location** ($G, c, r'$)

Input: An undirected graph $G = (V, E)$ with a cut function $f$, a cost function $c: V \rightarrow \mathbb{R}$, and a demand function $r': V \rightarrow \mathbb{R}$.

Output: A nonempty subset $S$ of $V$ minimizing $\sum_{v \in S} c(v)$ such that $f(X) \geq r'(v)$ for every $v \in V \setminus S$ and every subset $X \subseteq V \setminus S$ with $v \in X$.

For the source location problem with uniform demands, we can find an optimal solution in linear time if the family $\mathcal{X}(f)$ of extreme sets is known in advance [13]. Similarly, we have the following result for the minimum transversal problem (3.2) with uniform $r$, when the image of $f$ is $D$. 
Corollary 3 The minimum transversal problem (3.2) for a posimodular function \( f : 2^V \to D \) can be solved in \( O(|D|2^{|D|}nT_f) \) time if \( r \) is uniform (i.e., \( r(X) = k \in \mathbb{R} \) for all \( X \subseteq V \)).

Proof Let \( \mathcal{X}(f) \) be the family of all minimal extreme sets \( X \) with \( f(X) < k \). As observed in Sect. 1, \( \mathcal{X}(f) \) is laminar, and hence every two sets in \( \mathcal{X}(f) \) are pairwise disjoint. Let \( S \) be the set of elements obtained by choosing one element with the minimum cost from each set in \( \mathcal{X}(f) \). Below, we show that \( S \) is an optimal solution.

Since every feasible solution contains at least one element from each set in \( \mathcal{X}(f) \), it follows that the cost of any feasible solution is at least \( \sum_{v \in S} c(v) \). Furthermore, \( S \) is feasible. Indeed, if some set \( X \subseteq V \setminus S \) satisfied \( f(X) < k \), then there would exist an extreme set \( Y \subseteq X \) with \( f(Y) \leq f(X) < k \) by definition of extreme sets, contradicting the construction of \( S \). Thus, \( S \) is optimal.

Clearly, \( S \) can be found in linear time if \( \mathcal{X}(f) \) is given. Hence, combined with Corollary 2, \( S \) can be obtained in the time claimed. \( \square \)

A set function \( r : 2^V \to \mathbb{R} \) is called modulotone if, for every nonempty subset \( X \) of \( V \), there exists an element \( v \in X \) such that all proper subsets \( Y \) of \( X \) with \( v \in Y \) satisfy \( r(Y) \geq r(X) \). We remark that the minimum transversal problem (3.2) for a uniform cost function \( c \) and a modulotone demand function \( r \) is studied in [18] as a generalization of the source location problem with uniform costs [20] or the external network problem [21]. As observed in [18], this problem can be solved if solid sets can be computed efficiently. More precisely, we need to compute solid sets including \( u \) and \( v \) but not \( w \) for all three distinct elements \( u, v, \) and \( w \) in \( V \). Therefore, Corollary 2 does not imply the tractability of this problem.

3.2.3 Approximation

We now give a faster algorithm that obtains an additive approximation. Given a parameter \( \rho > 0 \), the algorithm finds a solution \( T \) with \( f(T) \leq \min f + \rho \).

Theorem 5 For any given \( \rho \geq 1 \), there is an algorithm for finding an additive \( \rho \)-approximate solution of a posimodular function \( f \) over a range \( \{0, 1, \ldots, d\} \) which uses at most \( 2(e(\rho + 1)/2)^{2d/(\rho+1)} \) oracle queries.

Proof We first query all singleton sets, and let \( \tilde{v} \) be the singleton that attains the smallest value. We then run MINPOSIMODULAR- \( D(f) \) using a modified version of the inc\( BC(v) \) predicate stating that \( v \) increases either \( B \) or \( C \) by at least \( \rho + 1 \). We output the better of the two results: \( \tilde{v} \) and the set found by the modified MINPOSIMODULAR- \( D(f) \).

If there is a singleton set that is an additive \( \rho \)-approximation, we clearly output a solution that is no worse. If such a set does not exist, i.e., if \( f(v) > \min f + \rho \) for all \( v \in V \), then Lemma 5 can be strengthened to imply that every element \( v \) in \( V \setminus S \) increases either \( B \) or \( C \) by at least \( \rho + 1 \). Thus, we may assume that \( f(X) \geq (\rho+1)|X| \), for \( X \in \{B, C\} \), and \( f(A) \geq |A| + \min f \), as before. The recurrences induce a binary tree where a left child refers to a call that increases \( A \) and a right child a call that increases \( B \) or \( C \) by at least \( \rho + 1 \). By posimodularity,

\[
f(X) + f(A) \leq f(S) + f(A \cup (S \setminus X)) \leq \min f + d,
\]
for $X \in \{B, C\}$. Thus, $|A| + (\rho + 1)|B| \leq d$ and $|A| + (\rho + 1)|C| \leq d$. Adding these inequalities and using that $\rho \geq 1$ yields that $d \geq |A| + |B| + |C|$. Hence, the depth of the recurrence is at most $d$. Furthermore, the number of right edges on any root-to-leaf path is at most $t = |B| + |C| \leq 2d/(\rho + 1)$. Hence, the number of recursive calls is bounded by

$$2^{d/(\rho+1)} \sum_{t=0}^{2d/(\rho+1)} \left(\begin{array}{c} d \\ t \end{array} \right) \leq 2(e(\rho + 1)/2)^{2d/(\rho+1)},$$

each involving linear computation. \hfill \Box

### 3.3 Generating All Minimizers

In this subsection, we consider generating all minimizers of a posimodular function $f : 2^V \rightarrow D$. Note that $f$ might have exponentially many minimizers. In fact, constant functions have $2^n - 1$ minimizers. We therefore consider output sensitive algorithms.

**Theorem 6** For a posimodular function $f : 2^V \rightarrow D$, we can generate all minimizers of $f$ with $O(nT_f)$ delay, after $O(3^{|D|n^2T_f})$ time to compute the first minimizer.

This part is the most technical; we outline the approach. We focus on finding locally minimal minimizers, as other minimizers can be quickly generated from those. The main challenge is dealing with the set $Z$ of singleton minimizers. We can treat the elements of $V \setminus Z$ as before, and also those elements of $Z$ that do not increase $A$. We partition the remaining elements of $Z$ into a collection $Z$ of maximal minimizers, and observe that these sets do not cross (or overlap with) other minimizers. Those sets in $Z$ that increase both $A$ and at least one of $B, C$ can be treated the same way as before (with TRY). The rest is split into two parts: $Z_0$, those that don’t increase $A$ and $Z_1$, those that do (but don’t increase $B$).

The key idea is to examine pairs of sets from $Z_1$: if there are two sets in $Z_1$ that together increase $B$ or $C$, either they are contained in $S$, or one of them is outside $S$. In either case we make progress. Otherwise, we show that $Z_1$ contains at most one set outside $S$, and we try all such possibilities. We finally observe that a minimizer contains at most a single set from $Z_0$.

We remark that we consider the singleton minimizers to also be locally minimal, since the empty set is not a minimizer.

#### 3.3.1 Algorithm for Generating Locally Minimal Minimizers

The main challenge in finding all locally minimal solutions is dealing with singleton minimizers, as they violate the assumptions of Lemma 5. In fact, MinPosimodular-$D(f)$ finds all locally minimal minimizers when there are no singleton minimizers.

Our top-level algorithm FindAllMinimizers (Algorithm 3) for computing all minimizers first computes $\text{min}_f$, the value of minimizers, by calling MinPosimodular-$D(f)$. It then computes $Z = \{v \in V : f(\{v\}) = \text{min}_f\}$, the set of all singleton
minimizers, and outputs them. Finally, it initiates the recursive algorithm \textsc{Solve} (Algorithm 5) that searches through all possible minimizers. \textsc{Solve} uses a subroutine, \textsc{Condense} (Algorithm 4), and two macros, \textsc{Try} and \textsc{Try BC}.

\textbf{Algorithm 3} \textsc{FindAllMinimizers()}

\begin{algorithmic}
  \STATE $\min f \leftarrow f(\text{MINPOSIMODULAR-D}(f))$
  \STATE $Z \leftarrow \emptyset$
  \FOR{$v \in V$}
    \IF{$f(v) = \min f$}
      \STATE $Z \leftarrow Z + v$
      \STATE Output $v$ (as a minimizer)
    \ENDIF
  \ENDFOR
  \RETURN \textsc{Solve}($\emptyset$, $\emptyset$, $\emptyset$)
\end{algorithmic}

\textsc{Condense} partitions a given subset $Z_a$ of $Z$ into minimizers $Z_1$, $Z_2$, \ldots, $Z_s$, such that no two together minimize $f$; namely, $f(Z_i) = \min f$, for all $i$, while $f(Z_i \cup Z_j) > \min f$, for all $i \neq j$. To this end, it uses a subroutine \textsc{Maximal} that greedily identifies a maximal minimizer consisting of singleton minimizers. As we shall see, the sets $Z_i$ satisfy a non-crossing property with each locally minimal minimizer, and can therefore be treated as indivisible.

\textbf{Algorithm 4} \textsc{Condense($Z_a$)}

\begin{algorithmic}
  \STATE $i \leftarrow 0$; $Z' \leftarrow Z_a$
  \WHILE{$Z' \neq \emptyset$}
    \STATE $i \leftarrow i + 1$
    \STATE $Z_i \leftarrow \text{Maximal}(Z')$
    \STATE $Z' \leftarrow Z' \setminus Z_i$
  \ENDWHILE
  \RETURN $Z = \{Z_1, Z_2, \ldots, Z_i\}$
\end{algorithmic}

\begin{algorithmic}
  \STATE Subroutine \textsc{Maximal}(Z')
  \STATE $M \leftarrow \{v\}$, for some $v \in Z'$
  \WHILE{$\exists v \in Z'$ that does not increase $M$}
    \STATE $M \leftarrow M \cup \{v\}$
  \ENDWHILE
  \RETURN $M$
\end{algorithmic}

Recall the predicates \textsc{incA} and \textsc{incBC}, which we generalize to accept a set $X$ as parameter. Thus, e.g., \textsc{incA}($X$) denotes the predicate that $X$ increases $A$, i.e., $f(A \cup X) > f(A)$. Similarly, \textsc{incBC}($X$) denotes the predicate $(f(B \cup X) > f(B)) \lor (f(C \cup X) > f(C))$.

Define the macros

\textsc{TryBC($X$)} : \textbf{if} ($X$ increases $B$) \textsc{Solve}($A$, $B \cup X$, $C$) \textbf{else} \textsc{Solve}($A$, $B$, $C \cup X$),

and

\textsc{Try($X$)} : \textsc{Solve}($A \cup X$, $B$, $C$); \textsc{TryBC($X$)}. 
Algorithm 5 Procedure SOLVE(A, B, C)

Require: Disjoint subsets A, B, C of V
1: if \( \exists v \in V \setminus (A \cup B \cup C), \text{inc}A(v) \land \text{inc}BC(v) \) then
2: TRY(v)
3: return
4: \( Z_a \leftarrow Z \setminus (A \cup B \cup C \cup \{v \in Z : \neg \text{inc}A(v)\}) = \{v \in Z \setminus (A \cup B \cup C) : \text{inc}A(v)\}\)
5: \( Z \leftarrow \text{CONDENSE}(Z_a) \)
6: if \( \exists Z_i \in Z, \text{inc}A(Z_i) \land \text{inc}BC(Z_i) \) then
7: TRY(Z_i)
8: return
9: \( Z_0 \leftarrow \{Z_i \in Z : \neg \text{inc}A(Z_i)\}; Z_1 \leftarrow Z \setminus Z_0 = \{Z_i \in Z : \text{inc}A(Z_i)\}\).
10: if \( \exists Z_x, Z_y \in Z_1, \text{inc}BC(Z_x \cup Z_y) \) then
11: SOLVE(A \cup Z_x, B, C); SOLVE(A \cup Z_y, B, C)
12: TRYBC(Z_x \cup Z_y);
13: return
14: \( \hat{S} \leftarrow B \cup C \cup \{v \in V \setminus (A \cup Z) : \text{inc}A(v)\}\)
15: for each subcollection \( Z' \subseteq Z_1 \) with \( |Z_1 \setminus Z'| \leq 1 \) do
16: \( \hat{S} = \hat{S} \cup Z' \)
17: if \( f(\hat{S}) = \min f \) then
18: Output \( \hat{S} \)
19: \( \hat{Z} \leftarrow \{Z_i \in Z_0 : f(\hat{S} \cup Z_i) = \min f\} \)
20: for each \( Z_i \in \hat{Z} \) do
21: Output \( \hat{S} \cup Z_i \)
22: if \( |\hat{Z}| = 2 \) then
23: Output \( \hat{S} \cup \hat{Z} \), if it is a minimizer

3.3.2 Analysis

We first argue a series of lemmas, deriving properties of specific groups of units, before combining into a proof of correctness and time complexity.

Recall that a tuple \((A, B, C)\) is valid for a minimizer \(S\) if \(A \subseteq V \setminus S\) and \(B \cup C \subseteq S\).

The following variation of Lemma 5 can easily be shown with the same argument.

Lemma 7 Let \( S \) be a minimizer and \((A, B, C)\) be a valid tuple for \( S\). If \( Y \) is a non-empty subset of \( V \setminus S\) that is not a minimizer, then \( \text{inc}BC(Y)\).

A general monotonicity property holds for any set whose elements don’t increase the value of another set.

Lemma 8 Let \( Y \subseteq V \) be a set and \( W = \{v \in V \setminus Y : v \text{does not increase } Y\} \) be the set of elements that don’t increase \( Y\). Then, \( f \) is monotone non-decreasing on \( W\).

Proof Let \( X \subseteq W \) and \( v \in X\). By posimodularity, \( f(X) + f(Y \cup \{v\}) \geq f(X \setminus \{v\}) + f(Y)\). Since \( v \) does not increase \( Y\), \( f(Y \cup \{v\}) \leq f(Y)\), and thus \( f(X) \geq f(X \setminus \{v\})\). Since this holds for every \( X \subseteq W\), it follows that \( f \) is monotone non-decreasing on \( W\). \(\square\)

Lemma 9 The union of a pair of sets in \( Z\) is not a minimizer. Namely, for every distinct \( Z_i, Z_j \in Z\), it holds that \( f(Z_i \cup Z_j) > \min f\).
Proof Suppose without loss of generality that $Z_i$ was formed by CONDENSE before $Z_j$. Since $Z_i$ was a maximal minimizer, $f(Z_i \cup \{u\}) > \min f$, for any $u \in Z_j$. By the monotonicity of $f$ on $Z_a$ (Lemma 8), it follows that $f(Z_i \cup Z_j) \geq f(Z_i \cup \{u\}) > \min f$.

The elements of $Z_a$ can be treated in groups given by $Z$, as indicated by the following non-crossing property of $Z$. We say that two sets $X, Y$ cross if $X \setminus Y$, $Y \setminus X$ and $X \cap Y$ are all nonempty.

Lemma 10 Let $S$ be a locally minimal minimizer, $(A, B, C)$ a valid tuple for $S$, and $Z$ as computed by SOLVE($A, B, C$). Then, every set in $Z$ is non-crossing with $S$.

Proof Let $Z_i$ be a set in $Z$. Suppose, for the sake of obtaining a contradiction, that $Z_i$ does cross $S$. Then $S$ is not a singleton set. By posimodularity, it holds for each $v \in Z_i \cap S$ that

$$f((Z_i \setminus S) \cup \{v\}) + f(S) \geq f(Z_i \setminus S) + f(S\setminus\{v\}). \tag{3.4}$$

Observe that the elements of $Z_a$ ($\supseteq Z_i$) fail incBC, since they satisfy incA and it was established in line 1 that either incA or incBC fails. In particular, they don’t increase $B$, so by Lemma 8, $f$ is monotone on $Z_a$. Thus, $f(Z_i) \geq f((Z_i \setminus S) \cup \{v\}) \geq f(Z_i \setminus S)$. Moreover, since $Z_i$ is a minimizer, it follows that $f(Z_i) = f((Z_i \setminus S) \cup \{v\}) = f(Z_i \setminus S)$. This implies from (3.4) that $f(S) \geq f(S\setminus\{v\})$, contradicting the local minimality of $S$.

The following lemma argues that sets in $Z_0$ have a limited representation within minimizers, implying the correctness of lines 19–23 of SOLVE. Denote $\bigcup X = \bigcup_{X \in \mathcal{X}} X$ for the support of a collection $\mathcal{X}$ of sets.

Lemma 11 Let $S$ be a minimizer and $Z_0$ be as computed in line 9 during an invocation of SOLVE that is valid for $S$. Let $Z_0 = \bigcup Z_0$ and $S_0 = S\setminus Z_0$. Then: a) $S_0$ is a minimizer, b) Every minimizer containing $S_0$ contains at most two sets from $Z_0$, and c) If a minimizer contains $S_0$ and two sets of $Z_0$, then those are the only sets in $Z$ whose addition to $S_0$ gives a minimizer (i.e., $|\{Z_i \in Z_0 : f(S_0 \cup Z_i) = \min f\}| = 2$).

Proof We say that a set $Y$ is $\mathcal{X}$-monotone if, for every subcollection $\mathcal{X}' \subseteq \mathcal{X}$ and every set $X \in \mathcal{X}'$, $f(Y \cup \bigcup \mathcal{X}') \geq f(Y \cup (\bigcup \mathcal{X}' \setminus X))$. We first argue that $S_0 = S \setminus Z_0$ is $Z_0$-monotone.

Let $Z'_0 \subseteq Z_0$ be a subcollection of $Z_0$ and let $Z'_0 = \bigcup Z'_0$. By posimodularity, for each $Z_i \in Z'_0$,

$$f(S_0 \cup Z'_0) + f(A \cup Z_i) \geq f(S_0 \cup Z'_0 \setminus Z_i) + f(A).$$

Since $Z_i$ does not increase $A$ (as it is contained in $Z_0$), $f(A \cup Z_i) \leq f(A)$. Thus, we have that $f(S_0 \cup Z'_0) \geq f(S_0 \cup Z'_0 \setminus Z_i)$, which establishes that $S_0$ is $Z_0$-monotone. This establishes part a), that $S_0$ is a minimizer, since $S$ is a minimizer.
Let $Z_i \in \mathcal{Z}_0$ be such that $S_0 \cup Z_i$ is also a minimizer, and let $Z_j, Z_k$ also be different sets in $\mathcal{Z}_0$. Then, by posimodularity,

$$f(S_0 \cup (Z_j \cup Z_k)) + f(S_0 \cup Z_i) \geq f(Z_j \cup Z_k) + f(Z_i).$$

By assumption, $f(S_0 \cup Z_i) = f(Z_i) = \min f$, while $f(Z_j \cup Z_k) > \min f$, by Lemma 9. Thus, $S_0 \cup (Z_j \cup Z_k)$ is not a minimizer.

It follows that if $\mathcal{Z}_0$ contains three or more sets, the addition of more than a single set to $S_0$ results in a non-minimizer. That leaves only the possibility that $\mathcal{Z}_0$ contains exactly two sets, whose addition may still result in a minimizer.

We are now ready to argue the correctness of the algorithm. We first argue the base case separately.

**Lemma 12** Let $S$ be a locally minimal minimizer, and let $\text{Solve}(A, B, C)$ be a valid call (for $S$) that is a leaf in the recursion tree. Then, the call outputs $S$.

**Proof** Since the call to $\text{Solve}$ is a leaf, we reach line 15. We show separately how $\text{Solve}$ identifies the parts of $S$ contained in $V \setminus Z_a$, $Z_1$, and $\mathcal{Z}_0 = Z_a \cup Z_1$. We can quickly characterize the elements of $V \setminus Z_a$ that are in $S$: By validity, $S$ contains $B \cup C$ but none of $A$. By Lemma 4, $S$ cannot contain elements $v \in V \setminus A$ that fail $\text{inc}A(v)$. The remaining elements $v \in V \setminus (A \cup B \cup C \cup \{v \in V \setminus Z : \neg \text{inc}A(v)\})$ satisfy $\text{inc}A(v)$, and since $\text{inc}A(v) \land \text{inc}BC(v)$ did not hold, they also satisfy $\neg \text{inc}BC(v)$. By Lemma 5, they must be contained in $S$. Hence, $S \setminus Z_a = \tilde{S}$, as computed in line 14 of $\text{Solve}$.

We now characterize the elements of $\cup Z_1$ that are in $S$. Since the third if-block is not entered, there does not exist $Z_x, Z_y \in Z_1$ such that $\text{inc}BC(Z_x \cup Z_y)$ holds. We claim that at most one set in $Z_1$ is within $V \setminus S$ while the other must be contained in $S$: For the sake of contradiction, suppose we have two sets $Z_x, Z_y$ of $Z_1$ that are both contained in $V \setminus S$. Then, by Lemma 9, $Z_x \cup Z_y$ is not a minimizer. By Lemma 7, we have that $\text{inc}BC(Z_x \cup Z_y)$ should hold, a contradiction.

We finally characterize the elements of $\cup \mathcal{Z}_0$ that are in $S$. Lemma 11 shows that $S_0 = S \setminus \mathcal{Z}_0$ is also a minimizer. From the argument above, it is given by $S_0 = \tilde{S}$, as computed in line 16 of $\text{Solve}$. By Lemma 11, $S \setminus S_0$ consists of one of three alternatives: a) the emptyset (line 18), b) a single set from $\mathcal{Z}_0$ (lines 19–21), or c) both of the two sets in $\mathcal{Z}_0$ whose addition to $S_0$ gives a minimizer (lines 22–23). In all cases, the algorithm finds $S$. \hfill \Box

**Theorem 7** The algorithm $\text{FindAllMinimizers}$ finds all locally minimal minimizers of $f$, in $O(3^{3|D|}n^2T_f)$ time.

**Proof** Let $S$ be a locally minimal minimizer. We prove by induction that if $\text{Solve}(A, B, C)$ is valid for $S$, then it outputs $S$. Since $\text{FindAllMinimizers}$ calls $\text{Solve}(\emptyset, \emptyset, \emptyset)$ which is trivially valid, this implies the lemma. The base case, when the call is a leaf, follows from Lemma 12.

In the inductive case, $\text{Solve}$ makes a recursive call. Consider the three different blocks where recursive calls are made.
If the first if-block is reached, then for the same reason as in Lemma 6, TRY examines both possibilities for the element \( v \), and one of the calls made is valid.

If the second if-block is reached, \( Z_i \) is either contained in \( S \) or in \( V \setminus S \), since it is non-crossing with \( S \) (by Lemma 10) and it does not contain \( S \) (by the hypothesis). TRY examines both possibilities.

Finally, suppose the third if-block is reached. Neither \( Z_x \) nor \( Z_y \) cross \( S \) (and they do not contain \( S \)), so either both of them are in \( S \), or one of them is in \( V \setminus S \). The call to \( TRYBC \) tries the former possibility, while the calls to \( SOLVE \) explore the latter. Thus, at least one of the calls made is valid.

In all cases, one of the recursive calls made is valid for \( S \), and by the inductive hypothesis, it will output \( S \).

The time complexity follows from the consideration of the recursion depth, the branching factor, and the complexity of each call to \( SOLVE \). The recursion depth is at most \( 3|D| \), since each recursive call increases one of \( A \), \( B \), or \( C \). The branching factor is (at most) 3, as each invocation makes at most 3 recursive calls. The complexity of each invocation of \( SOLVE \) (ignoring its recursive calls) is dominated by the call to \( CONDENSE \) and line 10, both of which have complexity \( O(n^2T_f) \).

For functions with range \( D = \{0, 1, \ldots, d\} \), the time complexity can be somewhat improved, along the lines of Theorem 4.

### 3.3.3 Finding All Minimizers

We observe that after (or during) the generation of the locally minimal minimizers of \( f \), all the remaining (not locally minimal) minimizers of \( f \) can be generated with \( O(nT_f) \) delay. Indeed, by the definition of local minimality, for each minimizer \( S \) of \( f \) that is not locally minimal, there is a chain \( T_0 ( = T ) \subset T_1 \subset \cdots \subset T_k ( = S ) \) starting from a locally minimal minimizer \( T \) and ending with \( S \) such that, for each \( i = 1, \ldots, k \), \( T_i \) is a minimizer of \( f \) and \( |T_i \setminus T_{i-1}| = 1 \). Therefore, after generating all locally minimal minimizers of \( f \), we recursively check, for each minimizer \( T \) of \( f \) and each \( v \notin T \), whether \( T \cup \{v\} \) is also minimizer of \( f \). For each successful recursive call, we then make \( O(n) \) queries.

To avoid repeating solutions (and possibly increasing the time complexity), we store all solutions visited in a tree (a binary tree). This results in space complexity \( O(3^{|D|}n^3) \), but does not increase the time complexity.

### 3.4 Hardness Results

The hard instance formed in Theorem 1 has \( d = n \). Thus, it also yields an \( \Omega(2^{0.32d}) \) lower bound on the oracle calls needed. We additionally give the following lower bound that shows that the time complexity must involve a factor linear in \( n \).

**Theorem 8** Posimodular function minimization requires \( \Omega(n) \) oracle calls, even when restricted to functions with image \( D = \{0, 1\} \).
Let $g : 2^V \to \{0, 1\}$ be a function defined by $g(X) = 1$ if $X \neq \emptyset$, and $g(\emptyset) = 0$. For each element $v \in V$, define $g_v : 2^V \to \{0, 1\}$ by

$$g_v(X) = \begin{cases} 
0 & \text{if } X = \emptyset \text{ or } X = \{v\} \\
1 & \text{otherwise.}
\end{cases}$$

Note that both $g$ and $g_v$ are monotone and thus posimodular. Also note that the minimum $g$-value is 1, and each function $g_v$ has exactly one minimizer $\{v\}$ with $g_v(v) = 0$ for $v \in V$. Observe that $n$ oracle calls are necessary to distinguish functions in $\{g\} \cup \{g_v \mid v \in V\}$.

**Remark 1** Modifying the construction of Theorem 1 by multiplying the function values by $\rho + 1$ shows that obtaining an additive $\rho$-approximation also requires $\exp(d/(\rho + 1))$ oracle calls, matching Theorem 5.

### 4 Posimodular Function Maximization

In this section, we consider posimodular function maximization defined as follows.

*Posimodular Function Maximization*

**Input:** A posimodular function $f : 2^V \to \mathbb{R}_+$, 

**Output:** A subset $X$ of $V$ maximizing $f$. 

(4.1)

Similar to posimodular function minimization, the problem (4.1) is in general intractable.

**Theorem 9** Every algorithm for posimodular function maximization requires at least $2^{n-1}$ oracle calls in the worst case.

**Proof** Let us first consider the case in which $n$ is even, i.e., $n = 2k$ for some positive integer $k$. Let $g : 2^V \to \mathbb{R}_+$ be a function defined by $g(X) = |X|$ if $|X| \leq k - 1$, and $g(X) = k$ otherwise, and for a subset $S \subseteq V$ with $|S| \geq k$, define a function $g_S : 2^V \to \mathbb{R}_+$ by $g_S(X) = g(X)$ if $X \neq S$, and $g_S(X) = k + 1$ if $X = S$. Since $g$ is monotone, it is posimodular. We claim that $g_S$ is also posimodular.

Note that $g_S(Z) \geq g_S(Z')$ holds for any pair of subsets $Z$ and $Z'$ with $Z \supseteq Z'$ except for $Z' = S$. Let $X$ and $Y$ be two subsets of $V$ with $X \cap Y \neq \emptyset$. In order to check the posimodular inequality (1.1), we can assume that $S = X \setminus Y$ or $Y \setminus X$, since all the other cases can be proven easily. By symmetry, let $S = X \setminus Y$. Then we have $g_S(X) = k$, $g_S(X \setminus Y) = k + 1$, and since $|Y \setminus X| \leq n - k - 1 = k - 1$, $g_S(Y) \geq g_S(Y \setminus X) + 1$ holds. These imply the posimodular inequality 1.1.

Let $q = \sum_{i=k}^{n} \binom{n}{i} \geq 2^{n-1}$ (note that $q$ is the number of subsets with cardinality at least $k$). Assume that there exists an algorithm $A$ for the posimodular function maximization which requires oracle calls fewer than $q$. Let $\mathcal{X}$ denote the family of subsets of $V$ which are called by $A$ if the posimodular function $g$ is an input of $A$. Since $|\mathcal{X}| \leq q - 1$, we have a subset $S$ such that $S \notin \mathcal{X}$ and $|S| \geq k$. This implies...
that \( g_S(X) = g(X) \) for all \( X \in \mathcal{X} \), which contradicts that Algorithm \( A \) distinguishes between \( g \) and \( g_S \) (i.e., \( A \) cannot know if the optimal value is either \( k \) or \( k+1 \)).

Next let us consider the case in which \( n \) is odd, i.e., \( n = 2k+1 \) for some nonnegative integer \( k \). Let \( g : 2^V \to \mathbb{R}_+ \) be a function defined by \( g(X) = |X| \) if \( |X| \leq k \), and \( g(X) = k+1 \) otherwise, and for a subset \( S \subseteq V \) with \( |S| \geq k+1 \), define a function \( g_S : 2^V \to \mathbb{R}_+ \) by \( g_S(X) = g(X) \) if \( X \neq S \), and \( g(X) = k+2 \) if \( X = S \). In a similar way to the previous case, we can observe that \( g_S \) is posimodular and that at least \( \sum_{i=k+1}^n \binom{n}{i} \geq 2^{n-1} \) oracle calls are required to solve posimodular function maximization.

\( \square \)

For the case \( f : 2^V \to \{0, 1, \ldots, d\} \) we have the following tight bound.

**Theorem 10** Posimodular function maximization for \( f : 2^V \to \{0, 1, \ldots, d\} \) can be solved in \( \Theta\left(\binom{n}{d-1} T_f\right) \) time if \( d = O(n^{1/4}) \).

The following lemma shows the lower bound for posimodular function maximization, while the upper bound will be shown in the next subsection.

**Lemma 13** Posimodular function maximization for \( f : 2^V \to \{0, 1, \ldots, d\} \) requires \( \Omega\left(\binom{n}{d-1}\right) \) oracle calls, if \( n \geq 2d - 2 \).

**Proof** Let \( g : 2^V \to \{0, 1, \ldots, d\} \) be a function defined by \( g(X) = |X| \) if \( |X| \leq d-2 \), and \( g(X) = d-1 \) otherwise. For a subset \( S \subseteq V \) with \( |S| \geq n-d+1 (\geq d-1) \), define a function \( g_S : 2^V \to \{0, 1, \ldots, d\} \) by \( g_S(X) = g(X) \) if \( X \neq S \), and \( g_S(X) = d \) if \( X = S \). Since \( g \) is monotone, it is posimodular. We claim that \( g_S \) is also posimodular.

Note that \( g_S(Z) \geq g_S(Z') \) holds for any pair of subsets \( Z \) and \( Z' \) with \( Z \supseteq Z' \) except for \( Z' = S \). Let \( X \) and \( Y \) be two subsets of \( V \) with \( X \cap Y \neq \emptyset \). In order to check the posimodular inequality (1.1), we can assume that \( S = X \setminus Y \) or \( Y \setminus X \), since all the other cases can be proven easily. By symmetry, let \( S = X \setminus Y \). Then we have \( g_S(X) = d-1 \), \( g_S(X \cup Y) = d \), and since \( |Y \setminus X| \leq n - |S| - 1 \leq d - 2 \) and \( |Y| > |Y \setminus X| \), \( g_S(Y) \geq g_S(Y \setminus X) + 1 \) holds. These imply the posimodular inequality (1.1). In a similar way to the proof of Theorem 9, we can observe that posimodular function maximization requires at least \( \sum_{i=n-d+1}^n \binom{n}{i} = \Omega\left(\binom{n}{d-1}\right) \) oracle calls, to distinguish among \( g \) and all \( g_S \) with \( |S| \geq n-d+1 \). \( \square \)

4.1 Polynomial Time Algorithm for Small \( d \)

In this section, we consider the case where \( d = o(n) \), and present an \( O\left(\binom{n}{d-1} T_f\right) \)-time algorithm for posimodular function maximization when the image size is \( d = O(n^{1/4}) \).

The following simple lemma implies that the problem can be solved in \( O\left(\binom{n}{d} T_f\right) \) time.

**Lemma 14** Let \( f \) be a posimodular function, and \( S \) be a maximal maximizer of \( f \) (i.e., a maximizer such that no proper superset is a maximizer of \( f \)). Then, \( f(X \cup \{v\}) > f(X) \) holds for any pair of a set \( X \subseteq V \) and an element \( v \in V \) such that \( X, \{v\} \) and \( S \) are pairwise disjoint.

\( \square \) Springer
Proof By (1.1), we have \( f(X \cup \{v\}) + f(S \cup \{v\}) \geq f(X) + f(S) \). By the maximality of \( S \), we have \( f(S \cup \{v\}) < f(S) \). Hence, \( f(X \cup \{v\}) > f(X) \).

\[ \Box \]

Corollary 4 Let \( f : 2^V \rightarrow \{0, 1, \ldots, d\} \) be a posimodular function. Then we have \( |S| \geq n - d \) for any maximal maximizer \( S \) of \( f \).

Proof Let \( k = |S| \), and let \( X_0 ( = \emptyset) \subseteq X_1 \subseteq \cdots \subseteq X_{n-k} ( = V \setminus S) \) be a chain with \( |X_i| = i \) for all \( i \). Then it follows from Lemma 14 that

\[ f(X_0) = 0 < f(X_1) < \cdots < f(X_{n-k}) (\leq d), \tag{4.2} \]

which implies that \( n - k \leq d \).

\[ \Box \]

By the corollary, posimodular function maximization can be solved in \( O\left(\binom{n}{d} T_f\right) \) time by checking all subsets \( X \) with \( |X| \geq n - d \).

In the remaining part of this section, we reduce the complexity to \( O\left(\binom{n}{d-1} T_f\right) \). In the case where there exists a maximizer of \( f \) with size at least \( n - d + 1 \), we can find it in \( O\left(\binom{n}{d-1} T_f\right) \) time by checking all subsets of size at least \( n - d + 1 \). Hence, we consider the case where no maximizer of \( f \) has size at least \( n - d + 1 \). Then we have \( \max\{ f(X) \mid X \subseteq V \} = d \), since otherwise (i.e., \( \max\{ f(X) \mid X \subseteq V \} \leq d - 1 \)) by the proof of Corollary 4, any maximal maximizer \( X^* \) would satisfy \( |X^*| \geq n - d + 1 \). Again by Corollary 4 and (4.2),

Every maximal maximizer \( X^* \) of \( f \) satisfies \( |X^*| = n - d \) and \( f(X^*) = d \). \( (4.3) \)

Moreover, (4.2) implies that \( f(V \setminus X^*) = d \) and \( V \setminus X^* \) is also a maximizer, which gives \( n \geq 2d \) (by \( |X^*| \geq |V \setminus X^*| \)). Also note that there exists a maximizer of \( f \) with size \( d \) (by \( |V \setminus X^*| = d \)). Below, we show that under the assumption (4.3), we can choose \( O(d^2 \binom{n}{d-3}) \) members \( X' \) from the family of subsets with size \( d - 1 \) to find a maximizer \( X \cup \{v\} \) of \( f ' \) (with size \( d \)) such that \( X \in X' \) and \( v \notin X \), by showing the following series of lemmas.

Lemma 15 Under the assumption (4.3), we have the following two statements.

(i) For any maximizer \( S \) of \( f \), there exists a maximizer \( S' \) of \( f \) with \( S' \cap S = \emptyset \) and \( |S'| = d \).

(ii) Let \( S_1, S_2 \) be two maximizers of \( f \) with \( S_1 \cap S_2 = \emptyset \). Then, there exist two maximizers \( X_1, X_2 \) of \( f \) with \( |X_1| = |X_2| = d \) and \( X_i \subseteq S_i, i = 1, 2 \). Moreover, any subset \( Y \subseteq V \) with \( X_1 \subseteq Y \subseteq V \setminus X_2 \) or \( X_2 \subseteq Y \subseteq V \setminus X_1 \) is a maximizer of \( f \).

Proof (i) Let \( S \) be an arbitrary maximizer of \( f \), and \( S_1 \) be a maximal maximizer of \( f \) with \( S_1 \supseteq S \). By (4.3), we have \( |S_1| = n - d \) and hence \( |V \setminus S_1| = d \). It follows from (4.2) that \( f(V \setminus S_1) = d \), which means that \( V \setminus S_1 \) is a maximizer of \( f \) with size \( d \) which is disjoint from \( S \).

(ii) Since we have \( f(V \setminus S_1) + f(V \setminus S_2) \geq f(S_1) + f(S_2) \) by (1.1), both \( V \setminus S_1 \) and \( V \setminus S_2 \) are also maximizers of \( f \). By (4.3), we have \( |V \setminus S_j| \leq n - d \) and thus \( |S_j| \geq d \) for \( j = 1, 2 \). By applying (i) to \( V \setminus S_j \) (\( j = 1, 2 \)), we obtain a maximizer \( X_j \subseteq S_j \) with \( |X_j| = d \). Here note that \( X_1 \cap X_2 = \emptyset \). Moreover, for any set
By Lemma 15, we have two maximizers \( X \) with \( f(X_1 \cup Z) + f(X_2 \cup Z) \geq f(X_1) + f(X_2) \) by (1.1). This completes the proof. \( \square \)

**Lemma 16** Assume that (4.3) holds. Let \( S \) be a maximizer of \( f \) with size \( d \), and let \( X \) be a subset of \( V \) such that \( |X| = f(X) = d - 1 \) and \( X \cap S = \emptyset \). Then, there exists an element \( v \in V \setminus (S \cup X) \) with \( f(X \cup \{v\}) = d \).

**Proof** Let \( S' \) be a maximizer of \( f \) with \( S' \cap S = \emptyset \) and \( |S'| = d \) such that \( |S'| < |X| \). We remark that such an \( S' \) always exists by Lemma 15 (i), and \( S' \cap S = \emptyset \) is satisfied by \( |S'| < |X| \). Moreover, it follows from Lemma 15 (ii) that \( V \setminus (X \cup S') \) is also a maximizer of \( f \). For \( v \in S' \setminus X \), we have \( f(X \cup \{v\}) + f(V \setminus (X \cup (S' \setminus \{v\}))) \geq f(X) + f(V \setminus (X \cup S')) = 2d - 1 \) by (1.1). Therefore, it suffices to show that \( f(V \setminus (X \cup (S' \setminus \{v\}))) \leq d - 1 \) to prove \( f(X \cup \{v\}) = d \).

Assume to the contrary that \( f(V \setminus (X \cup (S' \setminus \{v\}))) = d \). By Lemma 15 (i), there exists a maximizer \( S'' \) of \( f \) with \( |S''| = d \) and \( S'' \cap (V \setminus (X \cup (S' \setminus \{v\}))) = \emptyset \), i.e., \( S'' \subseteq X \cup (S' \setminus \{v\}) \), which contradicts the minimality of \( |S'| \). \( \square \)

We remark that \( S \) and \( X \) in Lemma 16 always exist if (4.3) is satisfied. In fact, by Lemma 15, we have two maximizers \( X_1 \) and \( X_2 \) of \( f \) such that \( |X_1| = |X_2| = d \), \( X_1 \cap X_2 = \emptyset \), and \( V \setminus X_2 \) is also a maximizer of \( f \). Let \( S = X_1 \) and \( X = X_2 \setminus \{v\} \) for any \( v \in X_2 \). Then \( S \) satisfies the condition in Lemma 16, and since \( V \setminus X_2 \) is a maximal maximizer of \( f \), (4.2) implies that \( X \) also satisfies the condition in Lemma 16.

**Lemma 17** Let \( \mathcal{X} \) be the family of all subsets \( X \) of \( V \) such that \( |X| = d - 1 \) and \( X \cap S \neq \emptyset \) for all maximizers \( S \) of \( f \) with \( |S| = d \). Then, under the assumption (4.3), we have \( |\mathcal{X}| = O(d^2 \binom{n}{d-3}) \).

**Proof** By Lemma 15, there exist two maximizers \( S_1 \) and \( S_2 \) of \( f \) with \( |S_1| = |S_2| = d \) and \( S_1 \cap S_2 = \emptyset \). Clearly, \( |\mathcal{X}| \) is bounded by the number of sets \( X \) with size \( d - 1 \) with \( X \cap S_1 \), \( X \cap S_2 \neq \emptyset \), which is

\[
\sum_{i,j \geq 0, i+j \leq d-1} \binom{d}{i} \binom{d}{j} \left( \frac{n-2d}{d-1-i-j} \right) \leq \sum_{k=2}^{d-1} \binom{2d}{k} \left( \frac{n-2d}{d-1-k} \right) = O(d^2 \binom{n}{d-3}).
\]

\( \square \)

Let \( c \) be a constant such that \( |\mathcal{X}| \leq cd^2 \binom{n}{d-3} \) for \( \mathcal{X} \) in Lemma 17. Based on these lemmas, we can find a maximizer of \( f \) in the following manner:

**Algorithm MAXPOSIMODULAR**

**Step 1.** Find a subset \( X_1 \) of \( V \) such that \( |X_1| \geq n - d + 1 \) and \( f(X_1) = \max \{ f(X) \mid X \subseteq V, |X| \geq n - d + 1 \} \). If \( f(X_1) = d \), then output \( X_1 \) and halt.

**Step 2.** Find a subset \( X_2 \) of \( V \) such that \( |X_2| = d - 1 \) and \( f(X_2) = \max \{ f(X) \mid X \subseteq V, |X| = d - 1 \} \). If \( f(X_2) = d \), then output \( X_2 \) and halt. If \( f(X_2) \leq d - 2 \), then output \( X_1 \) and halt.
Step 3. Choose \( \min \{ cd^2 (d_{n-3}) + 1, |X_1| \} \) members \( X \) from \( X_1 = \{ X \subseteq V \mid |X| = d - 1, f(X) = d - 1 \} \). For each such \( X \), if \( f(X \cup \{v\}) = d \) for some \( v \notin X \), then output \( X \cup \{v\} \) and halt.

Step 4. Output \( X_1 \) and halt.

**Lemma 18** Algorithm MAXIMODULAR(\( f \)) solves posimodular function minimization for \( f : 2^V \rightarrow \{ 0, 1, \ldots, d \} \) in \( O(\max(\binom{n}{d-1}, nd^2 (d_{n-3}))T_f) \) time.

**Proof** Let us first prove the correctness of the algorithm. Let \( S \) be a maximal maximizer of \( f \). Assume that \( f(S) = d \) holds. Then Corollary 4 implies that \( |S| \geq n - d \). If \( |S| \geq n - d + 1 \), then \( S \) is found in Step 1.

On the other hand, we consider the case where there is no maximal maximizer with size at least \( n - d + 1 \). Thus, \( |S| = n - d \) holds (again by Corollary 4), and then we have (4.3). By the discussion after Lemma 16, \( f(X_2) \geq d - 1 \) must hold. If \( f(X_2) = d \), then \( X_2 \) is clearly a maximizer of \( f \), which is output in Step 2. Otherwise (i.e., \( f(X_2) = d - 1 \)), again by the discussion after Lemma 16, there exists a subset \( X \) with \( |X| = f(X) = d - 1 \) such that \( X \cap S = \emptyset \) for some maximizer \( S \) of \( f \) with size \( d \). By Lemma 16, for such an \( X \), we can obtain a maximizer of \( f \) by checking if \( f(X \cup \{v\}) = d \) for some \( v \notin X \). Now it follows from Lemma 17 that there exist at most \( cd^2 (d_{n-3}) \) sets \( X' \) with size \( d - 1 \) satisfying \( X' \cap S \neq \emptyset \) for all maximizers \( S \) of \( f \) with size \( d \). Hence, we can find a suitable set \( X \) (i.e., \( X \) disjoint with some maximizer of \( f \)) in at most \( cd^2 (d_{n-3}) + 1 \) attempts. Therefore, in this case, Step 3 correctly outputs a maximizer of \( f \).

Assume next that \( f(S) \leq d - 1 \). Then Algorithm MAXIMODULAR(\( f \)) output \( X_1 \) in Step 2 or 4, which is correct, since there exists a maximal maximizer of size at least \( n - d + 1 \) by Corollary 4.

As for the time complexity of Algorithm MAXIMODULAR(\( f \)), we see that Steps 1 and 2 can be executed in \( O(\binom{n}{d-1}T_f) \) time. Since Steps 3 and 4 respectively require \( O(nd^2 (d_{n-3})T_f) \) and \( O(n) \) time, in total, algorithm requires \( O(\max(\binom{n}{d-1}, nd^2 (d_{n-3}))T_f) \) time.

By Stirling’s formula, we have \( \binom{n}{d-1} = \Theta\left(\frac{n^{d-1}}{d^{d-1}e^{d-1}}\right) \) and \( nd^2 (d_{n-3}) = \Theta\left(\frac{n^{d-2}}{d^{d-3}e^{d-3}}\right) \). It follows that if \( d = O(n^{1/4}) \), then the time complexity of Algorithm MAXIMODULAR(\( f \)) is \( O(\binom{n}{d-1}T_f) \), which proves Theorem 10. We also remark that if \( d > n^{2/5} \), then the \( O(\binom{n}{d}T_f) \)-time algorithm which checks all subsets of \( X \) with \( |X| \geq n - d \) (based on Corollary 4) is faster than Algorithm MAXIMODULAR(\( f \)), since then \( \binom{n}{d} = O(nd^2 (d_{n-3})) \).

**Acknowledgements** We are most grateful for suggestions from anonymous reviewers that allowed us to improve the time complexity of Algorithm 3 and simplify several arguments. We thank S. Fujishige, M. Grötschel, and S. Tanigawa for their helpful comments. This research was partially supported by the Scientific Grant-in-Aid from Ministry of Education, Culture, Sports, Science and Technology of Japan.

**References**

1. Arata, K., Iwata, S., Makino, K., Fujishige, S.: Locating sources to meet flow demands in undirected networks. J. Algorithms 42, 54–68 (2002)
2. Chekuri, C., Ene, A.: Approximation algorithms for submodular multiway partition. In: IEEE 52nd Annual Symposium on Foundations of Computer Science, pp. 807–816 (2011)
3. Egres open problem list: http://lemon.cs.elte.hu/egres/open/Maximizing_a_skew-supersubmodular_function
4. Feige, U., Mirrokni, V.S., Vondrák, J.: Maximizing non-monotone submodular functions. SIAM J. Comput. 40, 1133–1153 (2011)
5. Frank, A.: Augmenting graphs to meet edge-connectivity requirements. SIAM J. Discrete Math. 5(1), 25–53 (1992)
6. Fujishige, S.: A laminarity property of the polyhedron described by a weakly posi-modular set function. Discrete Appl. Math. 100(1–2), 123–126 (2000)
7. Halldórsson, M.M., Ishii, T., Makino, K., Takazawa, K.: Posimodular function optimization. In: Proceedings of 15th International Symposium on Algorithms and Data Structures (WADS), pp. 437–448. Springer (2017)
8. Ishii, T., Makino, K.: Posi-modular systems with modulotone requirements under permutation constraints. Discrete Math. Algorithms Appl. 2(1), 61–76 (2010)
9. Ito, H., Makino, K., Arata, K., Honami, S., Itsutsu, Y., Fujishige, S.: Source location problem with flow requirements in directed networks. Optim. Methods Softw. 18, 427–435 (2003)
10. Lawler, E.L.: Cutsets and partitions of hypergraphs. Networks 3(3), 275–285 (1973)
11. Lee, Y.T., Sidford, A., Wong, S.C.: A faster cutting plane method and its implications for combinatorial and convex optimization. In: IEEE 56th Annual Symposium on Foundations of Computer Science (FOCS), pp. 1049–1065. IEEE Computer Society (2015)
12. Lokshtanov, D., Marx, D.: Clustering with local restrictions. Inf. Comput. 222, 278–292 (2013)
13. Nagamochi, H.: Graph algorithms for network connectivity problems. J. Oper. Res. Soc. Japan 47(4), 199–223 (2004)
14. Nagamochi, H.: Minimum degree orderings. Algorithmica 56(1), 17–34 (2010)
15. Nagamochi, H., Ibaraki, T.: A note on minimizing submodular functions. Inf. Process. Lett. 67(5), 239–244 (1998)
16. Nagamochi, H., Ibaraki, T.: Polyhedral structure of submodular and posi-modular systems. Discrete Appl. Math. 107(1–3), 165–189 (2000)
17. Nagamochi, H., Shiraki, T., Ibaraki, T.: Augmenting a submodular and posi-modular set function by a multigraph. J. Combin. Optim. 5(2), 175–212 (2001)
18. Sakashita, M., Makino, K., Nagamochi, H., Fujishige, S.: Minimum transversals in posi-modular systems. SIAM J. Discrete Math. 23, 858–871 (2009)
19. Svitkina, Z., Tardos, É.: Min–max multiway cut. In: Approximation, Randomization, and Combinatorial Optimization, pp. 207–218 (2004)
20. Tamura, H., Sugawara, H., Sengoku, M., Shinoda, S.: Plural cover problem on undirected flow networks. IEICE Trans. J81–A, 863–869 (1998). (in Japanese)
21. van den Heuvel, J., Johnson, M.: The external network problem with edge- or arc-connectivity requirements. In: Combinatorial and Algorithmic Aspects of Networking, Lecture Notes in Computer Science, vol. 3405, pp. 114–126. Springer (2004)
22. Watanabe, T., Nakamura, A.: Edge-connectivity augmentation problems. J. Comput. Syst. Sci. 35, 96–144 (1987)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.