# Parameterized Algorithms and Kernels for 3-Hitting Set with Parity Constraints

Vikram Kamat[1]⋆ and Neeldhara Misra[2]⋆⋆

[1] University of Warsaw
*vkamat@mimuw.edu.pl*
[2] Indian Institute of Science, Bangalore
*neeldhara@csa.iisc.ernet.in*

**Abstract.** The 3-Hitting Set problem involves a family of subsets $\mathcal{F}$ of size at most three over an universe $\mathcal{U}$. The goal is to find a subset of $\mathcal{U}$ of the smallest possible size that intersects every set in $\mathcal{F}$. The version of the problem with parity constraints asks for a subset $S$ of size at most $k$ that, in addition to being a hitting set, also satisfies certain parity constraints on the sizes of the intersections of $S$ with each set in the family $\mathcal{F}$. In particular, an odd (even) set is a hitting set that hits every set at either one or three (two) elements, and a perfect code is a hitting set that intersects every set at exactly one element. These questions are of fundamental interest in many contexts for general set systems. Just as for Hitting Set, we find these questions to be interesting for the case of families consisting of sets of size at most three. In this work, we initiate an algorithmic study of these problems in this special case, focusing on a parameterized analysis. We show, for each problem, efficient fixed-parameter tractable algorithms using search trees that are tailor-made to the constraints in question, and also polynomial kernels using sunflower-like arguments in a manner that accounts for equivalence under the additional parity constraints.

## 1 Introduction

The 3-Hitting Set problem involves a family of subsets $\mathcal{F}$ of size at most three over an universe $\mathcal{U}$. The goal is to find a subset of $\mathcal{U}$ of the smallest possible size that intersects every set in $\mathcal{F}$. This is a fundamental NP-complete problem, and has been extensively studied from an algorithmic perspective. In particular, consider the decision version of the problem: here we are given, in addition to $\mathcal{U}$ and $\mathcal{F}$, a positive integer $k$, and the question is if $\mathcal{F}$ admits a hitting set of size at most $k$. Using a standard exhaustive search, analyzed using a depth-bounded search tree, this question can be answered in time $3^k n^{\mathcal{O}(1)}$.

⋆ Supported by a postdoctoral fellowship from the Warsaw Center of Mathematics and Computer Science

⋆⋆ Supported by the INSPIRE Faculty Scheme, DST India (project DSTO-1209).

Such algorithms are said to be *fixed-parameter tractable* with respect to $k$, since the exponential complexity is contained in $k$ alone, and the algorithm is efficient for small values of $k$. Such an analysis belongs naturally to the framework of parameterized complexity: here, in addition to the overall input size $n$, one studies how a secondary measurement (called the *parameter*), that captures additional relevant information, affects the computational complexity of the problem in question. Parameterized decision problems are defined by specifying the input, the parameter, and the question to be answered. The two-dimensional analogue of the class P is decidability within a time bound of $f(k)n^c$, where $n$ is the total input size, $k$ is the parameter, $f$ is some computable function and $c$ is a constant that does not depend on $k$ or $n$. A parameterized problem that can be decided in such a time-bound is termed *fixed-parameter tractable* (FPT). For general background on the theory of fixed-parameter tractability, see [3], [4], and [10].

A parameterized problem is said to admit a *polynomial kernel* if every instance $(I, k)$ can be reduced in polynomial time to an equivalent instance with both size and parameter value bounded by a polynomial in $k$. The study of kernelization is a major research frontier of parameterized complexity and many important recent advances in the area are on kernelization. For overviews of kernelization we refer to surveys [1, 5] and to the corresponding chapters in books on parameterized complexity [4, 10].

As it turns out, 3-Hitting Set is known to admit an instance kernel of size $\mathcal{O}(k^3)$, with an universe of size $\mathcal{O}(k^2)$ [11]. The earlier arguments for kernels relied on sunflowers, while the argument in [11] relies on a linear vertex-kernel for Vertex Cover. Further, unless the polynomial hierarchy collapses, it is also known that there are no kernels with $\mathcal{O}(k^{d-\varepsilon})$ sets [2]. In this work, we analyze the 3-Hitting Set problem with parity constraints. The constraints we study correspond to the well known problems Odd Set, Even Hitting Set and Perfect Code when restricted to families with sets of size at most three:

**3-Odd Set** Given a family $\mathcal{F}$ of sets of size at most three over an universe $\mathcal{U}$, and an integer $k$, is there a subset $S$ of $\mathcal{U}$ of size at most $k$ such that $|S \cap X|$ is odd for every $X \in \mathcal{F}$?

**3-Perfect Code** Given a family $\mathcal{F}$ of sets of size at most three over an universe $\mathcal{U}$, and an integer $k$, is there a non-empty subset $S$ of $\mathcal{U}$ of size at most $k$ such that $|S \cap X| = 1$ for every $X \in \mathcal{F}$?

**3-Even Hitting Set** Given a family $\mathcal{F}$ of sets of size at most three over an universe $\mathcal{U}$, and an integer $k$, is there a hitting set $S$ of $\mathcal{U}$ of size at most $k$ such that $|S \cap X|$ is even for every $X \in \mathcal{F}$?

All of these problems have been studied closely in the setting of families that involve sets of unbounded size. They are of fundamental interest in several contexts, including matroids and coding theory. Surprisingly, the natural question of understanding these problems on families where the set sizes are bounded appears to be relatively unexplored, although Hitting Set is very well-understood in the specialized context (for example, see the algorithm in [9]). In this work, we study the parameterized complexity of each of these problems.

We note that each of these problems are NP-complete even when restricted to families that have sets of size at most three, due to the characterization in [7]. For each of these problems, we provide fixed-parameter tractable algorithms and polynomial kernels. We remark that all the problems above can be reduced to the Min-Ones 3-SAT problem by an appropriate formulation of the constraints (see Proposition **??**). Therefore, by the results in [12], all of these problems are FPT with running time $2.85^k n^{\mathcal{O}(1)}$. We improve upon this running time significantly in each case by exploiting branching rules that are tailor-made to the constraints in question. In particular, for 3-Odd Set and 3-Perfect Code, we obtain a running time of $2.56^k n^{\mathcal{O}(1)}$; while for 3-Even Hitting Set, we obtain a running time of $1.73^k n^{\mathcal{O}(1)}$.

The approaches that have been successful for 3-Hitting Set are not easy to employ directly for the versions of the problem with parity constraints. The fundamental operations of deleting "irrelevant" elements from the instance or incorporating "forced" elements in our solution are now non-trivial to implement. For instance, notice that when an element $x$ is *forced*, it is typically safe to delete all sets containing $x$ from the family — and this is essential to proving kernel bounds. For, say, Odd Set on the other hand, it may not be safe to delete these sets without a mechanism for remembering the additional constraints imposed on them. In particular, let $S := \{x, y, z\}$ be a set containing $x$. If $x$ belongs to a solution, then we are obliged to "remember" that the elements $y$ and $z$ are, in some sense, coupled to each other — they must now either belong together or completely avoid any odd set that contains $S$.

We develop a natural auxiliary constraint graph to keep track of these additional implications. With a more refined notion of forcing vertices, a standard branching argument works. For the kernels, similarly, using these carefully defined operations, we are able to apply sunflower-style reduction rules to reduce the size of the instance. To the best of our knowledge, this is the first explicit attempt to obtain FPT algorithms that are faster than that suggested by the exhaustive branching strategy, or from the known algorithms for Min-Ones 3-SAT.

The work of [8] has a very intricate framework for applying reduction rules involving sunflowers to solve a much more general problem, namely, Min-Ones CSPs. As a corollary of this work, we already know that all three problems that we are discussing admit kernels where the instance size is bounded by $\mathcal{O}(k^4)$. However, with the help of a slightly specialized analysis, we obtain kernels that have instance sizes bounded $\mathcal{O}(k^3)$ for all the problems considered. It would be very interesting to improve these to quadratic or even linear bounds on the number of vertices in the universe. We also note that we are technically describing *bikernels*, which are instances of "annotated" versions of the original problems. However, in each case, there are simple reductions that will generate an instance of the original problem that has the same size bounds asymptotically.

The remainder of this paper is organized as follows. After introducing the terminology and notation in Section 2, we address the 3-Odd Set problem. We the describe both the branching algorithm and reduction rules towards kernelization. The details for 3-Perfect Code are quite similar to 3-Odd Set. On the other hand, 3-Even Hitting Set requires significantly different definitions of the two basic operations of deletion and incorporation, the details of which are deferred to a full version due to space constraints.

## 2 Preliminaries

A parameterized problem is denoted by a pair $(Q, k) \subseteq \Sigma^* \times \mathbb{N}$. The first component $Q$ is a classical language, and the number $k$ is called the parameter. Such a problem is *fixed–parameter tractable* (FPT) if there exists an algorithm that decides it in time $\mathcal{O}(f(k)n^{\mathcal{O}(1)})$ on instances of size $n$. A *kernelization algorithm* takes an instance $(x, k)$ of the parameterized problem as input, and in time polynomial in $|x|$ and $k$, produces an equivalent instance $(x', k')$ such that both $|x'|$ and $k'$ are functions purely of $k$. The output $x'$ is called the kernel of the problem. A kernel is said to be a *polynomial kernel* if its size $|x'|$ is polynomial in the parameter $k$. We refer the reader to standard text books on the subject [3, 10] for more details on the notion of fixed-parameter tractability.

The Min-Ones 3-SAT problem is the following: Given a 3-CNF SAT formula $\phi$, is there a satisfying assignment for $\phi$ that sets at most $k$ variables to True? A standard branching algorithm for this problem would branch on all clauses with no negated literals, attempting to set each variable in turn to True in turn, and the instances at the leaf nodes of the search tree can be satisfied by setting all variables to False. This algorithm has running time $\mathcal{O}(3^k)$. This has since been improved in [12] to $\mathcal{O}^*(2.85^k)$. We note that all the hitting set problems with parity constraints can be reduced to Min-Ones 3-SAT in a manner that preserves the size of the solution. We refer the reader to the appendix for a sketch of the proof. Therefore, an immediate corollary is that each of these problems admit algorithms with running time $\mathcal{O}^*(2.85^k)$.

We will use the notion of a sunflower frequently during our description of kernelization algorithms. A *sunflower* with $k$ petals and a core $Y$ is a collection of sets $\{S_1, \ldots, S_k\}$ such that $S_i \cap S_j = Y$ for all $i \neq j$; the sets $S_i \setminus Y$ are *petals*, and we require that none of them is empty. Note that a family of pairwise disjoint sets is a sunflower (with an empty core).

**Lemma 1 (The Sunflower Lemma[6]).** *Let $\mathcal{F}$ be family of sets each of cardinality $s$. If $|\mathcal{F}| > s!(k-1)^s$ then $\mathcal{F}$ contains a sunflower with $k$ petals.*

We use standard notation for branching vectors, focusing only on the drop on the measures in question and ignoring polynomial factors in $n$. We refer the reader to [10] for an overview of this notation. We also use the $\mathcal{O}^*()$ notation to suppress running times that are polynomial in $n$, focusing on the function of the parameter.

## 3 A Polynomial Case: Two-Sized Sets

A subroutine we will need in the subsequent sections is the case of Odd Set restricted to families where all sets have size two. We show that not only is this polynomially solvable, but that any solution has a special structure. Recall that the problem would require us to find a subset of the universe of size at most $k$ that chooses *exactly* one element from any set of size two. First, it is easy to see that the graphs corresponding to Yes-instances must be bipartite.

**Observation 1** ($\star$). *Let $(\mathcal{U}, \mathcal{F}, k)$ be an instance of* Odd Set*, where every set in $\mathcal{F}$ has size at most two. If this is a* Yes*-instance, then the graph given by $\mathcal{Z} := (\mathcal{U}, \mathcal{F})$ is bipartite.*

Now, we show that any solution must pick exactly one of the partitions of every bipartite component of $\mathcal{Z}$.

**Lemma 2** ($\star$). *Let $(\mathcal{U}, \mathcal{F}, k)$ be a* Yes*-instance of* Odd Set*, where every set in $\mathcal{F}$ has size at most two, and there are no isolated vertices. Let $\mathcal{Z} := (\mathcal{U}, \mathcal{F})$. Further, let $(A_1, B_1), \ldots, (A_t, B_t)$ be the connected components of $\mathcal{Z}$. Then, for any odd set $O \subseteq \mathcal{U}$, we have that for all $1 \leq i \leq t$, either $A_i \subseteq O$ and $B_i \cap O = \emptyset$, or $B_i \subseteq O$ and $A_i \cap O = \emptyset$.*

Due to space constraints, a proof of the lemmas above is deferred to a full version of this work.

## 4 3-Odd Set

An exhaustive branching algorithm for Odd Set would require us to branch on four possibilities for a given set $S = \{x, y, z\}$ in the input family — either the set $S$ is hit at exactly one element (leading to three different cases), or all of $S$ belongs to the odd set. However, in each of these branches, we are required to execute the operation of "incorporating" an element in our solution and solving an equivalent, recursive instance. incorporating an element $x \in U$ in the solution is straightforward for Hitting Set — we may simply delete all sets that contain $x$ and decrease the parameter by one. For Odd Set, on the other hand, we have to additionally remember for every three-element set $S$ that $x$ appears in, the elements $S \setminus \{x\}$ are no longer independent — they must either both be chosen in the future or not be chosen at all. To remember these constraints, we adopt the use of an auxiliary graph on the universe, adding an edge between elements that must be chosen.

Formally, we solve the following auxiliary problem:

**Constrained Odd Set**

Input: A family $\mathcal{F}$ of sets of size at most three over an universe $\mathcal{U}$, a graph $G = (\mathcal{U}, E)$, and a positive integer $k$.

Parameter: $k$

Question: Is there a subset $X \subseteq \mathcal{U}$ of size at most $k$ such that for every $S \in \mathcal{F}$, $|S \cap X|$ is odd, and further, for every connected component $C \in G$, either $C \subseteq X$ or $C \cap X = \emptyset$?

Notice that an instance of Odd Set is a special case of Constrained Odd Set, where the constraint graph is the edgeless graph on $|U|$ vertices. We define special operations that mimic the process of "incorporating" an element into a solution, and "deleting" an element from the instance. Let $\mathcal{Z} := (\mathcal{U}, \mathcal{F}, G, k)$ be an instance of Constrained Odd Set, let $x$ be an element from $\mathcal{U}$. Let $C(x)$ be the set of elements in the connected component containing $x$, in $G$. We now have the following.

**Delete $x$ from the instance.** If there is any set $S \in \mathcal{F}$ such that $S \subseteq C(x)$, then the operation of deleting $x$ from $\mathcal{Z}$ results in a trivial No instance. Otherwise, let $\mathcal{F}' := \{S \setminus C(x) \mid S \in \mathcal{F}\}$. The instance that results from *deleting $x$ from $\mathcal{Z}$* is given by $(\mathcal{U} \setminus C(x), \mathcal{F}', G \setminus C(x), k)$.

**Incorporate $x$ in the solution.** Consider the set $D(x)$ obtained as follows: to begin with, $D(x) = C(x)$. As long as there exists a set $S \in \mathcal{F}$ of size three such that $|S \cap D(x)| = 2$, we include the element $S \setminus D(x)$ in $D(x)$. At any stage, if the number of elements in $D(x)$ is more than $k$, or if there is a two-sized set $S \in \mathcal{F}$ that is contained in $D(x)$, then the operation of incorporating $x$ from $\mathcal{Z}$ results in a trivial No instance. Otherwise, we have the following. Let $\mathcal{F}'_1$ be the collection of sets of size two in $\mathcal{F}$ that intersect $D(x)$ at exactly one element, and let $\mathcal{F}'_2$ be the collection of sets of size three in $\mathcal{F}$ that intersect $D(x)$ at exactly one element. That is,

$$\mathcal{F}'_1 := \{S \mid S \in \mathcal{F}, |S| = 2, |S \cap D(x)| = 1\},$$

$$\mathcal{F}'_2 := \{S \mid S \in \mathcal{F}, |S| = 3, |S \cap D(x)| = 1\},$$

Now, let $G'$ be the graph obtained from $G$ by adding all possible edges among the vertices of $D(x)$. Then, the process of incorporating $x$ entails the following:

1. Let $G^*$ be obtained from $G'$ by adding the edges corresponding to the pairs $(S \setminus D(x))$ for all $S \in \mathcal{F}'_2$.
2. For all $S \in \mathcal{F}'_1$, delete $S \setminus D(x)$ from the instance $(\mathcal{U}, \mathcal{F}, G^*, k)$. If any of these operations result in a trivial No instance, then abort and return a No instance. Otherwise, let the resulting instance be $(\mathcal{U}', \mathcal{F}', H, k)$.
3. Return the instance $(\mathcal{U} \setminus D(x), \mathcal{F}'', H \setminus D(x), k - |D(x)|)$, where $\mathcal{F}''$ is the family obtained from $\mathcal{F}'$ after deleting all sets that intersect $D(x)$.

We prove the correctness of these operations in the following lemmas.

**Lemma 3 ($\star$).** *Let $(\mathcal{U}, \mathcal{F}, \mathcal{G}, k)$ be an instance of* Constrained Odd Set *and let $x \in \mathcal{U}$ be such that any odd set of size at most $k$ for $(\mathcal{U}, \mathcal{F})$ respecting the constraints in $G$ must contain $x$. Let $(\mathcal{U}', \mathcal{F}', \mathcal{G}', k')$ be the instance obtained by applying the operations associated with incorporating $x$ in the solution. Then, $(\mathcal{U}, \mathcal{F}, \mathcal{G}, k)$ is a* Yes *instance if, and only if, $(\mathcal{U}', \mathcal{F}', \mathcal{G}', k')$ is a* Yes *instance.*

**Lemma 4 ($\star$).** *Let $(\mathcal{U}, \mathcal{F}, \mathcal{G}, k)$ be an instance of* Constrained Odd Set *and let $x \in \mathcal{U}$ be such that any odd set of size at most $k$ for $(\mathcal{U}, \mathcal{F})$ respecting the constraints in $G$ cannot contain $x$. Let $(\mathcal{U}', \mathcal{F}', \mathcal{G}', k')$ be the instance obtained by applying the operations associated with deleting $x$ in the solution. Then, $(\mathcal{U}, \mathcal{F}, \mathcal{G}, k)$ is a* Yes *instance if, and only if, $(\mathcal{U}', \mathcal{F}', \mathcal{G}', k')$ is a* Yes *instance.*

We are now ready to present our branching and kernelization algorithms.

## 4.1 A Branching Algorithm for 3-Odd Set

We are now ready to describe the branching algorithm. Before branching, we always incorporate the elements of any singleton sets. For hitting set, a further simplification that is often exploited is the *domination strategy*: if there are elements $x$ and $y$ such that $x$ appears in all sets that $y$ appears in, then it is safe to delete $x$ from the instance, by a standard pushing argument — a solution containing $x$ can be replaced with a solution that contains $y$ instead of $x$. However, consider the following instance of Constrained Odd Set:

$$(\{x, y, z\}, \{y, w\}, \{w, x_1\}, \cdots, \{w, x_{k+1}\}, k)$$

Note that although $z$ is dominated by $y$; a solution containing $z$ cannot be morphed into a solution containing $y$ — because any constrained odd set is forced to contain $w$, which in turn forbids us from picking $y$. This rule, when valid, is useful in generating sets of size two, which can be branched on more efficiently. However, since we are unable to apply the domination rule in the stated form, several arguments used across different cases of the 3-Hitting Set algorithm in [9] cannot be adapted directly to our scenario. This motivates the need for a somewhat different branching strategy. Fortunately, it turns out that an elegant branching strategy leads us to an algorithm with running time $\mathcal{O}^*(2.56^k)$. We first consider the case when we have at least two sets of size three that overlap; and then consider the case when all the three-sized sets are disjoint and some sets have size two. We use the standard measure, which is the size of the odd set sought.

**Case A.** There exist a pair of three-sized sets that have at least one element in common. Let these sets be:

$$\{x, a_1, b_1\}, \{x, a_2, b_2\}$$

*Case A1.* Suppose $a_1 = a_2 = y$. Then we have the following branches:

- Incorporate $x$ and $y$. This forces us to incorporate $b_1$ and $b_2$, and the measure drops by four.
- Incorporate neither $x$ nor $y$. This forces us to incorporate $b_1$, and $b_2$, and the measure drops by two.
- Incorporate $x$ but not $y$. This leads to a drop of one in the measure.
- Incorporate $y$ but not $x$. This leads to a drop of one in the measure.

The overall branch vector, therefore, is $(\mathbf{4}, \mathbf{2}, \mathbf{1}, \mathbf{1})$.

*Case A2.* Without loss of generality, we may assume here that $a_1 \neq a_2$ and $b_1 \neq b_2$. Here, we branch exhaustively on these sets as follows:

- Incorporate $x$. This leads to a drop of one in the measure.
- Delete $x$. Incorporate $a_1$ and $b_2$, deleting $b_1$ and $a_2$. This is leads to a drop of two in the measure.
- Delete $x$. Incorporate $a_1$ and $a_2$, deleting $b_1$ and $b_2$. This is leads to a drop of two in the measure.
- Delete $x$. Incorporate $b_1$ and $a_2$, deleting $a_1$ and $b_2$. This is leads to a drop of two in the measure.
- Delete $x$. Incorporate $b_1$ and $b_2$, deleting $b_1$ and $a_2$. This is leads to a drop of two in the measure.

The overall branch vector, therefore, is $(\mathbf{2}, \mathbf{2}, \mathbf{2}, \mathbf{2}, \mathbf{1})$.

**Case B.** All the three-sized sets are disjoint. Let $\mathcal{H} \subseteq \mathcal{F}$ be all the two-sized sets in the family $\mathcal{F}$, and let $V \subseteq U$ be the set of all elements $\mathcal{U}$ that appear in some set of $\mathcal{H}$. Let the graph $\mathcal{Z}$ be defined as $(V, \mathcal{H})$. If the graph $\mathcal{Z}$ is not bipartite, then return No. Otherwise, let the connected components of $\mathcal{Z}$ be $Z_1 := (A_1, B_1), \cdots, Z_i := (A_i, B_i), \cdots, Z_\ell := (A_\ell, B_\ell)$, where $(A_i, B_i)$ are the bipartitions of the component $Z_i$. Recall from Lemma 2, any odd set $S$ for $(\mathcal{U}, \mathcal{F})$ must contain, for all $1 \leq i \leq \ell$, exactly one of $A_i$ or $B_i$. In other words, for all $1 \leq i \leq \ell$, either $S \cap Z_i = A_i$ or $S \cap C_i = B_i$.

On the other hand, let $C_1, \ldots, C_t$ denote the connected components of the constraint graph $G$. Now we have a few rules of simplification, applied in the order stated below, based on this observation. The correctness of these rules are easily established.

- If there exists $1 \leq i \leq \ell$ for which both $A_i$ and $B_i$ contain the vertices of the same component $C_i$ in the constraint graph, then return No.
- Let $S$ be a three-sized set in $\mathcal{F}$. If there exists $1 \leq i \leq \ell$ such that $S \subseteq A_i$, then incorporate $A_i$ and delete $B_i$. If there exists $1 \leq i \leq \ell$ such that $S \subseteq B_i$, then incorporate $B_i$ and delete $A_i$.
- Let $S$ be a three-sized set in $\mathcal{F}$. If there exists $1 \leq i \leq \ell$ such that $|S \cap B_i| = 2$ and $|S \cap A_i| = 1$, then incorporate $A_i$ and delete $B_i$. Similarly, there exists $1 \leq i \leq \ell$ such that $|S \cap A_i| = 2$ and $|S \cap B_i| = 1$, then incorporate $B_i$ and delete $A_i$.

We are now ready to branch as follows: for all $1 \leq i \leq \ell$, we incorporate all of $A_i$ in one branch (while deleting $B_i$) and all of $B_i$ in the other (while deleting $A_i$), with the measure dropping by at least one in both branches. In the base case, we are left with a collection of three-sized sets $\{S_1, S_2, \ldots, S_t\}$ that are mutually disjoint. Although tempting at this stage, it may not be correct to pick one element from each set, because different elements have different "costs" of incorporation, corresponding to the sizes of their connected components in the conflict graph. Further, not all elements of a set are equal in the context of the conflict graph; different elements may lead us to different paths of further incorporations; some which may be of greater benefit than others. To resolve such instances, we use the following strategy:

1. If there is some $S_i$ that contains an element $a$ with $|C(a)| > 1$, then branch exhaustively on $S_i$. Formally, if $S_i := \{a, b, c\}$, then we branch as follows:
   - Incorporate $a, b$ and $c$. This forces us to incorporate $C(a)$, and since $|C(a)| > 1$, the measure drops by at least four.
   - Incorporate $a$, delete $b$ and $c$;. This forces us to incorporate $C(a)$, and since $|C(a)| > 1$, the measure drops by at least two.
   - Incorporate $b$, delete $a$ and $c$; this leads to a drop of one in the measure.
   - Incorporate $c$, delete $a$ and $b$; this leads to a drop of one in the measure.

   The overall branch vector, therefore, is $(\mathbf{4}, \mathbf{2}, \mathbf{1}, \mathbf{1})$.
2. Now, every $S_i$ contains elements that belong to singleton components in the conflict graph. Here, we may arbitrarily choose an element from each set to be incorporated in the final solution.

The running time of the algorithm that we have just described is bounded by the following recurrence:

$$
\begin{aligned}
T(k) \leq \max\{&T(k-4) + T(k-2) + T(k-1) + T(k-1), \\
&4T(k-2) + T(k-1), \\
&T(k-1) + T(k-1)\}.
\end{aligned}
$$

Notice that correctness follows from Lemmas 3 and 4; along with the exhaustive nature of the branching strategy. Using standard techniques to bound the recurrence above, we have the following theorem.

**Theorem 1.** *The* Constrained Odd Set *problem admits a* FPT *algorithm with running time* $\mathcal{O}^*(2.56^k)$.

## 4.2 A Polynomial Kernel for 3-Odd Set

We now turn to the kernelization algorithm for Constrained Odd Set. For this discussion, we work with a weighted version of Constrained Odd Set, where all elements of the universe $\mathcal{U}$ have positive integer weights, and we are looking for a solution of total weight at most $k$. The delete and incorporate operations are adapted in the natural way, to use the weights of the elements when decreasing the budget, rather than the number of elements. To begin with, we have some simple pre-processing rules for an instance $(\mathcal{U}, \mathcal{F}, G, k)$:

**Reduction Rule 1.** If $\{x\} \in \mathcal{F}$, then we incorporate $x$ into the solution.

**Reduction Rule 2.** If $\{x\} \in \mathcal{U}$, and the weight of $x$ is greater than $k$, then delete $x$.

**Reduction Rule 3.** If $\{x, y\} \in \mathcal{F}$, and $x$ and $y$ are in the same connected component of $G$, then we return a trivial No-instance.

**Reduction Rule 4.** Let $C$ be a connected component in $G$ such that the sum of weights of all vertices in $C$ is greater than k, and let $x \in C$. We delete $x$ from the instance.

**Reduction Rule 5.** If $k \leq 0$ but $\mathcal{F}$ is non-empty then return a trivial No-instance.

We call Reduction Rules 1—5 the *basic reduction rules*. The correctness of these rules are easy to verify, and the arguments are omitted here due to space constraints. An instance is always reduced with respect to the basic reduction rules before the application of any further reduction rules. As with the algorithm for 3-Hitting Set, we will also appeal to the Sunflower Lemma (Lemma 1).

We now consider all the three-sized sets in $\mathcal{F}$, let us use $\mathcal{H}$ to denote this sub-family of $\mathcal{F}$. If $|\mathcal{H}| \geq 6k^3$, then $\mathcal{H}$ contains a sunflower with at least $(k+1)$ petals. It is easy to construct the sub-family corresponding to the sunflower in polynomial time. If the sunflower has an empty core, then we abort and return a trivial No-instance. Otherwise, let the petals of the sunflower be $P_1, \ldots, P_t$, where $t \geq k+1$. We have the following cases.

**Reduction Rule 6.** (The core has one element.) Let $\bigcap_{i=1}^{t} P_i := \{x\}$. We then incorporate $x$ in the solution.

**Reduction Rule 7.** (The core has two elements.) Let $\bigcap_{i=1}^{t} P_i := \{x, y\}$.

Let $P_i := \{x, y, z_i\}$. For all $1 \leq i \leq t$, we delete $z_i$ from the instance. Let $(\mathcal{U}', \mathcal{F}', G', k)$ be the resulting instance. We now have the following cases:

1. If $\mathcal{U}'$ does not contain either $x$ or $y$, then we return a trivial No-instance.
2. If exactly one of $x$ or $y$ is present in $\mathcal{U}'$, then we incorporate the element $\mathcal{U}' \cap \{x, y\}$ in our solution, and return the resulting instance.
3. If both $x$ and $y$ are present in $\mathcal{U}'$, and $x$ and $y$ are in the same connected component of $G'$, then we return a trivial No instance.
4. Otherwise, we return the instance $(\mathcal{U}', \mathcal{F}' \cup \{x, y\}, G', k)$.

We now claim the correctness of the two rules above.

**Lemma 5 ($\star$).** *Reduction rules 6 and 7 are safe.*

We now consider the collection of two-sized sets in $\mathcal{F}$, let us use $\mathcal{J}$ to denote this sub-family of $\mathcal{F}$. If $|\mathcal{J}| \geq 2k^2$, then $\mathcal{J}$ contains a sunflower with at least $(k+1)$ petals. Again, if the sunflower has an empty core, then we abort and return a trivial No-instance. Otherwise, let the petals of the sunflower be $P_1, \ldots, P_t$, where $t \geq k+1$. This sunflower necessarily has a core of size one; and this leads us to the following reduction rule:

**Reduction Rule 8.** Let $P_i := \{x, y_i\}$. For all $1 \leq i \leq t$, delete $y_i$ from the instance. If the resulting instance is a trivial No-instance, then return a trivial No-instance. Otherwise, return the instance obtained by incorporate $x$ in the solution.

**Lemma 6 ($\star$).** *Reduction rule 8 is safe.*

Notice that once the reduction rules are exhaustively applied, we are left with either a trivial No-instance, or an instance with at most $6k^3$ sets of size three, at most $2k^2$ sets of size two, and no singletons. Let us denote a reduced instance by $(\mathcal{U}^R, \mathcal{F}^R, G^R, k^R)$. We call an element of $\mathcal{U}^R$ *useful* if it appears in some set of $\mathcal{F}^R$. We now have two final reduction rules, whose correctness is self-evident, and which we apply only once on a reduced instance:

**Reduction Rule 9.** Let $C$ be a connected component in $G$ with no useful vertices, and let $x \in C$. We delete $x$ from the instance.

**Reduction Rule 10.** Let $C$ be a connected component in $G$ with at least one useful vertex, and $\ell$ useless vertices of total weight $w$. We replace all the useless vertices of $C$ by a dummy vertex of weight $w$ and make it adjacent to all vertices in the component $C$.

Since it is evident that elements of the universe that do not appear in any sets do not belong to any minimal solutions, the correctness of Reduction Rule 9 follows from Lemma 4.

Let us denote an instance reduced with respect to the reduction rule 9 by $(\mathcal{U}^\dagger, \mathcal{F}^\dagger, G^\dagger, k^\dagger)$. Since $|\mathcal{U}^\dagger| = |V(G^\dagger)|$, note that a bound on the universe of the kernel will follow from a bound on the number of vertices in $G^\dagger$. Observe that, by Reduction Rules 4 and 9, every connected component of $G^\dagger$ has size at most $k$, and there are at most as many components as there are useful vertices. Further, every component contains at most one vertex that is not useful, and this leads to a bound on the number of elements in the universe, as we argue in the following lemma.

**Lemma 7** **($\star$).** *Let $(\mathcal{U}, \mathcal{F}, G, k)$ be an instance that is recursively subjected to the basic reduction rules and the sunflower reduction rules (in that order), and finally to Reduction Rule 9. Let the resulting instance be denoted by $(\mathcal{U}^\dagger, \mathcal{F}^\dagger, G^\dagger, k^\dagger)$. Then, $|\mathcal{U}^\dagger| = \mathcal{O}(k^3)$.*

Lemmata 4—7 lead to the following theorem, which also concludes the objective of this section.

**Theorem 2.** *The* Constrained Odd Set *problem has an instance kernel of size $\mathcal{O}(k^3)$.*

## References

[1] Hans L. Bodlaender. "Kernelization: New Upper and Lower Bound Techniques". In: *Proceedings of the 4th Workshop on Parameterized and Exact Computation (IWPEC 2009).* Vol. 5917. 2009, pp. 17–37.

[2] Holger Dell and Dieter van Melkebeek. "Satisfiability Allows No Nontrivial Sparsification unless the Polynomial-Time Hierarchy Collapses". In: *J. ACM* 61.4 (2014), p. 23.

[3] Rod G. Downey and M. R. Fellows. *Parameterized Complexity.* Springer, 1999.

[4] J. Flum and M. Grohe. *Parameterized Complexity Theory.* Springer-Verlag New York Inc, 2006.

[5] Jiong Guo and Rolf Niedermeier. "Invitation to data reduction and problem kernelization". In: *SIGACT news* 38.1 (2007), pp. 31–45.

[6] Stasys Jukna. *Extremal combinatorics - with applications in computer science.* Springer, 2001, pp. I–XVII, 1–375.

[7] Khanna et al. "The Approximability of Constraint Satisfaction Problems". In: *SICOMP: SIAM Journal on Computing* 30 (2001).

[8] S. Kratsch and M. Wahlström. "Preprocessing of Min Ones Problems: A Dichotomy". In: *Automata, Languages and Programming, 37th International Colloquium, ICALP 2010, Part I.* Vol. 6198. 2010, pp. 653–665.

[9] Niedermeier and Rossmanith. "An Efficient Fixed-Parameter Algorithm for 3-Hitting Set". In: *Journal of Discrete Algorithms, Elsevier (unseen by me).* Vol. 1. 2003.

[10] Rolf Niedermeier. *Invitation to Fixed Parameter Algorithms (Oxford Lecture Series in Mathematics and Its Applications).* Oxford University Press, USA, 2006.

[11] Nishimura, Ragde, and Thilikos. "Smaller Kernels for Hitting Set Problems of Constant Arity". In: *International Workshop on Parameterized and Exact Computation (IWPEC), LNCS.* Vol. 1. 2004.

[12] Venkatesh Raman and Bal Sri Shankar. "Improved Fixed-Parameter Algorithm for the Minimum Weight 3-SAT Problem". In: *WALCOM.* Vol. 7748. 2013, pp. 265–273.