Search-Space Reduction for Boolean MinCSPs via Essential Constraints

Bart M. P. Jansen
Eindhoven University of Technology, The Netherlands

Ruben F. A. Verhaegh
Eindhoven University of Technology, The Netherlands

Abstract

For a fixed set $\mathcal{F}$ of Boolean constraint types, a MinCSP($\mathcal{F}$)-instance consists of a formula F that applies m constraints from $\mathcal{F}$ to a set of n Boolean variables. The goal is to remove a minimum subset of constraint applications from F to make the remaining formula satisfiable. Previous work characterized how the choice of $\mathcal{F}$ affects its polynomial-time solvability and approximability. We extend a recently introduced preprocessing framework for graph problems to the problem above. Rephrased in the context of CSPs, this framework defines a constraint application from a given formula F as *c-essential* if it is contained in all c-approximate solutions to F. Being able to efficiently detect these essential parts of a solution reduces the search space of any follow-up FPT algorithms parameterized by the solution size and yields an immediate asymptotic improvement to the runtime of such algorithms. In this work, we present a dichotomy theorem that distinguishes constraint sets $\mathcal{F}$ for which $c_{\mathcal{F}}$-essential constraint applications can be detected efficiently for some $c_{\mathcal{F}} \in \mathcal{O}(1)$, from those for which this task is intractable under established complexity-theoretic conjectures. Our results show that for any set $\mathcal{F}$ of *bijunctive* constraints, there is a polynomial-time algorithm that detects $\mathcal{O}(1)$-essential constraint applications. This contrasts the fact that constant-factor approximating a bijunctive MinCSP($\mathcal{F}$)-problem is intractable under the Unique Games Conjecture.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis; Theory of computation → Approximation algorithms analysis; Theory of computation → Parameterized complexity and exact algorithms; Theory of computation → Constraint and logic programming

Keywords and phrases fixed-parameter tractability, constraint satisfaction problems

Digital Object Identifier 10.4230/LIPIcs.SWAT.2026.22

Related Version *Full Version*: https://arxiv.org/abs/2606.00770

1 Introduction

Constraint Satisfaction Problems (CSPs) form a broad class of problems that describe many well-known computational tasks and allow us to study them in a unified language [19]. A *constraint* of arity r over a domain D is a function $f\colon D^r \to \{0,1\}$, and a *constraint set* $\mathcal{F}$ is a set of such constraints over the same domain and possibly of different arity. In this work, we focus on Boolean CSPs, and assume the domain D to be $\{0,1\}$ from now on. For a fixed constraint set $\mathcal{F}$, a CSP instance contains a formula consisting of multiple applications of constraints from $\mathcal{F}$ over a set of variables $\mathbf{X}$. Such a *constraint application* is a pair consisting of a constraint $f \in \mathcal{F}$ and a sequence $\mathbf{X}'$, whose length equals the arity of f, of variables from $\mathbf{X}$ that f is applied to. A formula consisting of several constraint applications is called an *$\mathcal{F}$-formula* if all constraint applications come from a constraint in $\mathcal{F}$.

Though many variants of CSPs exist, the goal in such problems is typically related to the satisfiability of a given formula F. An *assignment* $s : \mathbf{X} \to \{0,1\}$ over variable set $\mathbf{X}$ is a function that maps every variable in this set to a value in the domain (in our setting the Boolean domain $\{0,1\}$). Then, an assignment s over variable set $\mathbf{X}$ is said to *satisfy* a constraint application $(f, \mathbf{X}')$ over variable set $\mathbf{X}' \subseteq \mathbf{X}$ if $f(s(\mathbf{X}')) = 1$. We say that an assignment s satisfies a formula F if the assignment satisfies all constraint applications in F. Finally, a formula is *satisfiable* if there exists at least one assignment that satisfies it.

Perhaps the most well-known class of CSPs is the one where, for a given $\mathcal{F}$-formula F, the task is to determine whether or not F is satisfiable. Varying the pool of allowed constraints $\mathcal{F}$ impacts the difficulty of the problem and, in 1978, Schaefer gave a dichotomy theorem that characterizes for which constraint sets $\mathcal{F}$ the resulting problem is polynomial-time solvable and for which this is NP-hard [21]. Since then, many other variants of CSPs have been studied. We continue this line of research for the following class of optimization problems.

Weighted MinCSP($\mathcal{F}$)	
Input:	An $\mathcal{F}$-formula F in which each constraint application has a positive integer weight encoded in unary.
Task:	Find a minimum-weight set X of constraint applications in F s.t. $F - X$ is satisfiable.

For an instance F, we denote the weight of an optimal solution by $\mathrm{opt}(F)$. Additionally, we define the unweighted variant MinCSP($\mathcal{F}$) of the above problem as the restriction in which the weights in the input are all 1. Then, the resulting task is equivalent to finding a minimum-*size* set X for which $F - X$ is satisfiable.

This problem is NP-hard for many constraint classes $\mathcal{F}$. As such, e.g. approximation algorithms and fixed-parameter tractable (FPT) algorithms parameterized by the solution size t (i.e. algorithms that decide whether $\mathrm{opt}(F) \leq t$ in $g(t) \cdot \mathrm{poly}(n)$ time on formulas of size n, for some computable function g) have been studied. This has resulted in dichotomy theorems for the problem's constant-factor approximability in polynomial time [13], its solvability in FPT time [15], and its constant-factor approximability in FPT time [5].

We study a class of preprocessing algorithms for this problem. Various preprocessing algorithms are known to be very useful in practice as observed in SAT solvers for example [1], motivating a study of the extent to which polynomial-time preprocessing can aid in solving Weighted MinCSP($\mathcal{F}$). Preprocessing with the goal of reducing to an instance whose size is polynomially bounded in $\mathrm{opt}(F)$ has been studied for several problems through the notion of kernelization [7]. These studies include, for example, the Min 2CNF-Deletion problem (a.k.a. Almost 2-SAT [18]). Here, we take a different perspective on preprocessing with the goal of investigating how preprocessing can help for inputs whose solutions are large, that is, of size comparable to the total input size.

Essential constraint applications. We extend a recent framework by Bumpus et al. that introduces the notion of c-essential objects [6]. Although originally defined within the context of vertex selection problems on graphs, we can rephrase their original definition to apply to MinCSP($\mathcal{F}$) problems. Formally, for a given constant $c \in \mathbb{R}_{\geq 1}$ and an $\mathcal{F}$-formula F, we call a constraint application in this formula *c-essential* if it is contained in all c-approximate solutions to the (Weighted) MinCSP($\mathcal{F}$) instance F.

One could expect such c-essential objects to somehow stand out in the solution space, since we cannot even form a c-approximation without them. This might make it feasible to identify such objects during preprocessing, which would be useful for problems like MinCSP($\mathcal{F}$) in which the goal is to arrive at a certain structure by deleting a minimum number of objects. All essential objects are in particular part of all optimal solutions, so that, after finding and reducing them from the input, it suffices to solve an instance that looks for a strictly smaller solution. Since many FPT algorithms have a running time that scales exponentially with the solution size, each element of the solution we find during preprocessing decreases the running time by a multiplicative factor. As such, we are interested in preprocessing that finds essential objects, if there are any. This algorithmic task is formalized in the following problem statement, defined for a fixed constraint set $\mathcal{F}$ and constant $c \in \mathbb{R}_{\geq 1}$.

c-essential Detection for Weighted MinCSP($\mathcal{F}$)

Input: An $\mathcal{F}$-formula F with associated weights, and an integer k

Task: Find a subset S of constraint applications in F such that:

(G1) if $\text{opt}(F) \leq k$, then there is an optimal solution to F containing all of S, and

(G2) if $\text{opt}(F) = k$, then S contains all c-essential constraint applications.

Again we define the unweighted variant of the problem by requiring all weights in the input to be 1. Note that this problem is, in two ways, easier than the task of simply returning a set containing exactly the c-essential constraint applications in an input. First, the output set is allowed to contain more constraint applications, as long as they belong to an optimal solution. Secondly, the algorithm only needs to give a meaningful output if the integer k in its input provides a correct guess or upper bound on the optimal solution of the input. Nonetheless, being able to solve this detection task suffices to speed up FPT algorithms parameterized by solution size, in a black-box fashion. In particular, if for a constraint set $\mathcal{F}$ and a constant $c \in \mathbb{R}_{\geq 1}$, (1) there is a polynomial-time algorithm for c-essential Detection for MinCSP($\mathcal{F}$), and (2) there is an algorithm that outputs a solution to MinCSP($\mathcal{F}$) of size at most p, if one exists, in $g(p) \cdot \text{poly}(n)$ time for some computable function g, then there is an algorithm that computes an optimal solution of MinCSP($\mathcal{F}$) in $g(p-q) \cdot \text{poly}(n)$ time, where q is the number of c-essential constraint applications in the input and $p = \text{opt}(F)$. Bumpus et al. have proven this guarantee in the setting of graph problems, but their proof is easily seen to be valid for MinCSP($\mathcal{F}$) as well [6, Theorem 5.1].

Using this framework, several positive results and hardness results have been achieved for graph problems such as Vertex Multicut and Directed Feedback Vertex Set [6, 11]. Here, we extend the framework to achieve new results for CSPs.

Our results and other dichotomies. We present a dichotomy theorem that characterizes for which constraint sets $\mathcal{F}$ there is a constant c such that c-essential Detection for MinCSP($\mathcal{F}$) is polynomial-time solvable. Specifically, we show that this is true for constraint sets $\mathcal{F}$ that are 0-valid, 1-valid, IHS-B, or bijunctive (these terms are defined in Section 2). The dichotomy reveals that all other constraint sets $\mathcal{F}$ yield a MinCSP($\mathcal{F}$) problem for which c-essential detection is hard for every $c \in \mathbb{R}_{\geq 1}$, under established hardness assumptions.

▶ **Theorem 1.1.** *Let $\mathcal{F}$ be a finite Boolean constraint set.*

1. *If $\mathcal{F}$ is 0-valid, 1-valid, or 2-monotone, then there is a polynomial-time algorithm for c-essential Detection for MinCSP($\mathcal{F}$) for any $c \in \mathbb{R}_{\geq 1}$, since even Weighted MinCSP($\mathcal{F}$) is polynomial-time solvable in this case [13].*
2. *Otherwise, if $\mathcal{F}$ is IHS-B or bijunctive, then there is a constant $c_{\mathcal{F}}$ such that $c_{\mathcal{F}}$-essential Detection for Weighted MinCSP($\mathcal{F}$) is solvable in polynomial time.*
3. *Otherwise, if $\mathcal{F}$ is affine, then c-essential Detection for MinCSP($\mathcal{F}$) is NP-hard for every constant $c \in \mathbb{R}_{\geq 1}$ under the Unique Games Conjecture.*
4. *Otherwise, if $\mathcal{F}$ is weakly positive or weakly negative, then c-essential Detection for MinCSP($\mathcal{F}$) is NP-hard for every constant $c \in \mathbb{R}_{\geq 1}$.*
5. *Otherwise, c-essential Detection for MinCSP($\mathcal{F}$) is NP-hard for every constant $c \in \mathbb{R}_{\geq 1}$, even on formulas F with $\text{opt}(F) \leq 1$.*

As mentioned, other dichotomies exist that characterize the efficient solvability and approximability of MinCSP($\mathcal{F}$). Table 1 shows a comparison between some of these that closely relate to our work. It focuses on the boundary between positive and hardness results. The table includes, in left-to-right order, results on constant-factor approximating the problem

Table 1 A comparison of dichotomies for results on solving or approximating MinCSP($\mathcal{F}$). *Solving the problem on IHS-B or bijunctive constraint sets is FPT if an underlying graph (respectively the arrow graph or Gaifman graph) of the constraint set is $2K_2$-free. It is W[1]-hard otherwise [15].

Restriction on $\mathcal{F}$	Poly-time approx. [13]	FPT exact [15]	FPT approx. [5]	Essential detection
0-valid, 1-valid, 2-monotone	*easy*	*easy*	*easy*	*easy*
IHS-B	*easy*	*sometimes easy**	*easy*	*easy*
Bijunctive	hard	*sometimes easy**	*easy*	*easy*
None of the above	hard	hard	hard	hard

in polynomial time [13], exactly solving the problem in FPT time, parameterized by solution size [15], and constant-factor approximating the problem in FPT time [5]. A summary of our results from Theorem 1.1 is included in the final column. In a given row, the table indicates whether the specified tasks for constraint sets from that category – and not from any category from a previous row – are tractable under a suitable hardness assumption.

We point out that c-essential detection appears to be a strictly easier task than approximating the problem in polynomial-time or exactly solving the problem in FPT time. It is in particular interesting to see that all cases that admit an exact FPT algorithm also admit a polynomial-time c-essential detection algorithm. Based on previous work on c-essential objects mentioned earlier, this implies that all these FPT algorithms for MinCSP($\mathcal{F}$) can be updated to yield an improved runtime dependence on the complexity parameter. We also note that our results show the same boundary of hardness as the existing dichotomy on constant-factor approximating the problem in FPT time.

Finally, we mention some of the existing work on preprocessing, specifically kernelization and compression, for MinCSP and related problems. Unlike kernelization, compression allows problem instances to be transformed to an instance of a possibly different problem. Jansen and Włodarczyk have characterized the best-possible compression size achievable for MinCSP($\mathcal{F}$) in terms of the number of variables [12]. Related CSPs include determining whether a given $\mathcal{F}$-formula can be satisfied by an assignment that sets at least, or at most, a given number k of variables to 1. Kratsch and Wahlström characterized for which constraint sets the minimization problem admits a kernel of size polynomial in k [17]. Later, such a characterization was also given for the maximization problem [16].

Techniques. We proceed by giving a brief overview of the techniques we use to obtain our results. Our algorithmically most interesting positive result is the polynomial-time c-essential detection algorithm on bijunctive constraint sets. To achieve it, we use the insight that bijunctive formulas admit a representation in graph form. Then, for each constraint application in the input formula, we solve a sequence of separation problems on this graph to determine whether to put it in the output set of our c-essential detection algorithm or not.

Next, for each of our hardness results, we provide a two-step approach. To show, for a given category of constraint sets, that c-essential detection is hard for MinCSP($\mathcal{F}$), we start by formulating a canonical constraint set from that category and showing that the problem is hard for that set. Then, we show that the corresponding problem for all other constraint sets in that category is as hard as the problem for the canonical constraint set.

To achieve this second step, we show that an *essential implementation* of the canonical set $\mathcal{F}_{\text{can}}$ for a category can be made via any constraint set $\mathcal{F}$ in that category. That is, for every constraint application over $\mathcal{F}_{\text{can}}$, there is a conjunction of constraint applications over $\mathcal{F}$

that is equivalent to it, while satisfying an additional property as defined in Definition 4.1. The latter ensures that for every constant c, a reduction can be made from MinCSP($\mathcal{F}_{\text{can}}$) to Weighted MinCSP($\mathcal{F}$) that preserves the entire space of c-approximate solutions.

We remark that our definition of essential implementations is strictly stronger than the very similar notion of strong and perfect implementations by Khanna et al. [13]. An implementation from their framework also yields a reduction between CSP instances over different constraint sets. They showed that this reduction preserves the constant-factor approximability of the involved problems, but, in contrast to our result, it need not preserve the exact approximation ratio. We were surprised to see that most of the implementations we encountered in related work (also beyond strong and perfect implementations) happen to fit our definition of an essential implementation as well. This reveals even stronger links between problems that have already been shown to be reducible to one another.

Organization. The rest of the paper is organized as follows. In Section 2, we define some fundamental concepts and notation. Section 3 contains our positive results, proving Case 2 of Theorem 1.1. Section 4 contains our hardness results, starting with an introduction to essential implementations. Then, in Sections 4.1–4.3, we prove Cases 3, 4, and 5 of Theorem 1.1 respectively. We conclude with open questions in Section 5. The proofs of statements marked (★) are deferred to the full version.

2 Preliminaries

A constraint is a Boolean function $f\colon \{0,1\}^p \to \{0,1\}$ for some integer p, which is its *arity*. We assume all individual Boolean constraints to be satisfiable: unsatisfiable constraint applications in a formula F must be part of every solution to F and could thus be filtered in a simple preprocessing step. We call two constraints *equivalent* if they have the same arity and are satisfied by the same set of assignments.

Next, we define some standard Boolean constraints that are used throughout the paper. We write True or False for the arity-1 constraint that is satisfied if and only if its input variable is respectively 1 or 0. For non-negative integers p and $q \leq p$, we denote the arity-p constraint that takes the logical OR over p literals, of which q are negated, by $\text{OR}_{p,q}$. Hence, $\text{OR}_{p,q}$ denotes the constraint $(\neg x_1 \vee \ldots \vee \neg x_q \vee x_{q+1} \vee \ldots \vee x_p)$. We use shorthands OR_p for $\text{OR}_{p,0}$, and NOR_p for $\text{OR}_{p,p}$. We denote the constraint $(x_1 \oplus \ldots \oplus x_p = 1)$ by XOR_p and the constraint $(x_1 \oplus \ldots \oplus x_p = 0)$ by XNOR_p. We use shorthands $[\neq]$ for XOR_2 and $[=]$ for XNOR_2. We denote the constraint of arity p that is satisfied if and only if its p input variables are not all equal by NAE_p.

Now, we can define different classes of constraints. First, we call a constraint 0-*valid* or 1-*valid* if it is satisfied by the all-0 or the all-1 assignment respectively. We call a constraint *IHS-B*$^+$ if it can be expressed as a conjunction of OR_p constraints for integers p, the constraint $\text{OR}_{2,1}$, and the arity-1 constraint False. Likewise, we call constraints *IHS-B*$^-$ if they can be expressed as conjunction of NOR_p constraints for integers p, the constraint $\text{OR}_{2,1}$, and the arity-1 constraint True. We call a constraint *bijunctive* if it can be expressed in 2CNF (i.e.: as conjunction of OR_2, $\text{OR}_{2,1}$, and NOR_2 constraints). As subset of the bijunctive constraints, 2-monotone constraints are defined as those that can be expressed in the form $(a_1 \wedge \ldots \wedge a_p) \vee (\neg b_1 \wedge \ldots \wedge \neg b_q)$ for some $p, q \geq 0$. We call a constraint *affine* if it can be expressed as conjunction of linear constraints mod 2 (i.e.: as conjunction of XOR_p and XNOR_q constraints for integers $p \geq 1$ and integers $q \geq 1$). We call a constraint *weakly positive* (resp. *weakly negative*) if it can be expressed in CNF with all clauses containing at most one negated (resp. positive) variable.

We call a constraint set $\mathcal{F}$ 0-valid / 1-valid / bijunctive / 2-monotone / affine / weakly positive / weakly negative if every constraint $f \in \mathcal{F}$ satisfies the corresponding property. We call $\mathcal{F}$ IHS-B if all its constraints are IHS-B$^+$ or all its constraints are IHS-B$^-$ and we call a single constraint f IHS-B if it is either IHS-B$^+$ or IHS-B$^-$.

Finally, we note that we can use a reduction that transforms integer-weighted formulas into unweighted ones by duplicating constraint applications as many times as their weight, to show the following.

▶ **Lemma 2.1 (★).** *Let $\mathcal{F}$ be a constraint set and $c \geq 1$. If MinCSP($\mathcal{F}$) admits a polynomial-time c-essential detection algorithm, then so does Weighted MinCSP($\mathcal{F}$) if all weights are integers that are polynomial in the number of constraint applications.*

3 Positive results

To achieve our positive results, we establish conditions on a constraint set $\mathcal{F}$ that imply the existence of a c-essential detection algorithm for MinCSP($\mathcal{F}$). The statement below admits a similar proof to Theorem 4.1 in the earlier work by Bumpus et al. on essential vertices [6].

▶ **Lemma 3.1 (★).** *Let $\mathcal{F}$ be a constraint set and let $c \in \mathbb{R}_{\geq 1}$ be a constant. Suppose there is a polynomial-time algorithm that takes as input an $\mathcal{F}$-formula F and a constraint application C in it, and outputs a set of constraint applications P_C such that:*

1. *$C \notin P_C$; and*
2. *if X_C is a smallest set of constraints from $F - C$ such that $F - X_C$ is satisfiable, then $|P_C| \leq c \cdot |X_C|$; and*
3. *for every set X such that $F - X$ is satisfiable, $F - ((X \setminus \{C\}) \cup P_C)$ is also satisfiable,*

then, there is a polynomial-time $(c+1)$-essential detection algorithm for MinCSP($\mathcal{F}$).

In the proof of this statement, we show that such a set P_C is strictly larger than $c \cdot \mathrm{opt}(F)$ if C is a c-essential constraint application, and that C does not belong to every optimal solution if $|P_C| \leq c \cdot \mathrm{opt}(F)$. As such, for a given integer k, we can construct a c-essential detection algorithm by computing the set P_C for every constraint application C in F and returning the constraint applications for which $|P_C| > c \cdot k$.

In the full version of this paper, we show that a polynomial-time c-approximation algorithm for Weighted MinCSP($\mathcal{F}$) can be used to construct an algorithm that satisfies all three properties from Lemma 3.1. Khanna et al. have shown that such an algorithm exists when $\mathcal{F}$ is IHS-B [13, Theorem 2.13], so we conclude the following.

▶ **Corollary 3.2.** *Let $\mathcal{F}$ be an IHS-B constraint set. Then, there is a constant $c_{\mathcal{F}}$ for which MinCSP($\mathcal{F}$) admits a polynomial-time $c_{\mathcal{F}}$-essential detection algorithm.*

For bijunctive constraint sets $\mathcal{F}$, we can also use Lemma 3.1 to prove that a c-essential detection algorithm exists for MinCSP($\mathcal{F}$). This does require some more work and to do so, we make use of the well-known fact [3, 20] that formulas in 2CNF can be represented in graph-form as follows.

▶ **Definition 3.3.** *Let F be a 2CNF formula over variable set $\mathbf{X}$, or more generally, an $\mathcal{F}$-formula for some bijunctive constraint set $\mathcal{F}$. We construct a directed (multi)graph G_F as follows. For every variable $x \in \mathbf{X}$, we add x and $\neg x$ to the vertex set of G_F. For every clause $(\ell_1 \vee \ell_2)$ in the 2CNF representation of F with literals ℓ_1 and ℓ_2, we add the arcs $(\neg\ell_1, \ell_2)$ and $(\neg\ell_2, \ell_1)$ to G_F. We call the resulting graph the* implication graph *of F.*

We highlight that clauses may appear in multiple constraint applications of F and that this is reflected in the implication graph of F by allowing duplicate arcs in it. Furthermore, in our remaining arguments, we assume that we keep track of which clause in a formula corresponds to which arc in the implication graph. We exploit some of the structural properties of implication graphs to prove the following result.

▶ **Lemma 3.4 (★).** *Suppose that $\mathcal{F}$ is bijunctive so that every constraint $f \in \mathcal{F}$ can be written as conjunction of at most d disjunctions of two literals, for some constant d. Then, there is a polynomial-time $(2d^2+1)$-essential detection algorithm for MinCSP($\mathcal{F}$).*

Proof sketch. First note that we may assume w.l.o.g. that input formulas F are already given to us in suitable 2CNF representation when dealing with MinCSP($\mathcal{F}$) instances. Otherwise, we could, for every fixed bijunctive $\mathcal{F}$, provide a polynomial-time transformation using a hard-coded bijunctive representation of each constraint in $\mathcal{F}$. Thus, let F be an $\mathcal{F}$-formula represented in 2CNF in which every constraint is written as conjunction of at most d clauses. Let G_F be its implication graph, and let C be a constraint application in F that is expressed as $(a_1 \vee b_1) \wedge \ldots \wedge (a_\ell \vee b_\ell)$, where $a_1, \ldots a_\ell, b_1, \ldots, b_\ell$ are literals for some $\ell \leq d$.

To prove the lemma, we give a polynomial-time algorithm that computes a set P_C from F and C, after which we show that this set meets the three preconditions described in Lemma 3.1. To ensure that our algorithm is well-defined, we prove the claim below.

▷ Claim 3.5 (★). Let F' be any 2CNF formula that contains C and let $G_{F'}$ be the implication graph of F'. If there is an $i \in [\ell]$ such that $G_{F'}$ contains both an $(a_i, \neg a_i)$-path and a $(b_i, \neg b_i)$-path, then F' is unsatisfiable.

Proof sketch. We observe that an assignment s that satisfies F' and sets an arbitrary literal r to 1 must set all other literals reachable from r in $G_{F'}$ to 1 as well. Thus, if literals $\neg a_i$ and $\neg b_i$ are reachable from a_i and b_i respectively, any satisfying assignment to F' must set a_i and b_i to 0. Such an assignment cannot satisfy the clause C in F' that requires $(a_i \vee b_i)$. ◁

Now, P_C can be computed as follows.

- Construct the implication graph G_F of F.
- For all $i \in [\ell]$, compute a smallest subset of arcs in G_{F-f} that breaks all $(a_i, \neg a_i)$-paths in G_F and compute a smallest subset of arcs in G_{F-f} that breaks all $(b_i, \neg b_i)$-paths in G_F. If only one of these two sets exists, let T_i be this set, or, if both exist, let T_i be a minimum-size set among the two. (Note that at least one of these cuts must exist. If not, the implication graph G_C of the singular constraint application C would have to contain both an $(a_i, \neg a_i)$-path and a $(b_i, \neg b_i)$-path, which, by Claim 3.5, would imply C to be unsatisfiable. This would contradict our assumption that all individual constraints we consider are satisfiable.)
- Let T be $\bigcup_{i \in [\ell]} T_i$ and let P_C be the set of constraint applications in F with at least one of their corresponding implication arcs in T.

First, we observe that the construction above can be executed in polynomial time. The bottleneck is the second step in which a linear number of cut problems need to be solved. These are variants of the standard min-cut problem on directed graphs, and are easily seen to be solvable efficiently by a simple modification of a polynomial-time mincut algorithm, such as the Ford-Fulkerson algorithm [8].

It remains to prove that P_C satisfies the three properties specified in Lemma 3.1. By construction, P_C does not contain C, so it satisfies the first property.

For the other two properties, we start by noting that T is a set of arcs that, for every $i \in [\ell]$, breaks either all $(a_i, \neg a_i)$-paths or all $(b_i, \neg b_i)$-paths in G_F without including any arcs that correspond to a clause of C. We call such a set a *C-respecting arc set*. (Note that such sets

may include duplicates of arcs that correspond to a clause of C, but not those that specifically correspond to C.) Likewise, we call a set of constraint applications from F a *C-respecting constraint application set* if the union of their respective arcs in G_F is a C-respecting arc set. As such, P_C is a C-respecting constraint application set. We use this notation to show that P_C satisfies the second property specified in Lemma 3.1.

▷ Claim 3.6 (★). Let X_C be a smallest set of constraints from $F - C$ such that $F - X_C$ is satisfiable. Then, $|P_C| \leq 2d^2 \cdot |X_C|$.

Proof sketch. Since $F - X_C$ contains C, it follows from Claim 3.5 that X_C is a C-respecting constraint application set. By definition, the union T^* of all arcs in G_F that correspond to a clause in X_C is a C-respecting arc set in G_F. Since C contains at most d clauses that each introduce 2 arcs in G_F, we find that $|T^*| \leq 2d \cdot |X_C|$. Since every T_i breaks its respective paths optimally, we find that $\max_{i \in [\ell]} |T_i| \leq |T^*|$, so that the union T of all $\ell \leq d$ sets T_i has $|T| \leq d \cdot |T^*|$. With P_C being the set of constraint applications corresponding to the arcs in T, we find that $|P_C| \leq |T|$. Combining all inequalities we obtain $|P_C| \leq 2d^2 \cdot |X_C|$. ◁

Now, we show that P_C also satisfies the third property listed in Lemma 3.1.

▷ Claim 3.7 (★). For every set X such that $F - X$ is satisfiable, $F^* := F - ((X \setminus \{C\}) \cup P_C)$ is also satisfiable.

Proof sketch. Given an assignment s that satisfies $F - X$, we explain how to transform it into a satisfying assignment for F^*. Pick an arbitrary clause $(a_i \vee b_i)$ from C that is not yet satisfied by s and determine whether G_{F^*} contains no $(a_i, \neg a_i)$-path or that it contains no $(b_i, \neg b_i)$-paths. Since P_C is C-respecting, at least one of these statements is true and we assume w.l.o.g. that the former is true. Then, we modify s by setting a_i and all literals reachable from a_i in G_{F^*} to 1. We repeat this for as long as C contains unsatisfied clauses.

To see that s is a valid assignment after a single iteration, suppose for contradiction that s was modified to set both x and $\neg x$ to 1. Then, x and $\neg x$ are both reachable from a_i. However, the structure of implication graphs ensures that, for every arc (u, v), the arc $(\neg v, \neg u)$ also exists. Thus, $\neg a_i$ is reachable from x, which in turn is reachable from a_i, contradicting that G_{F^*} contains no $(a, \neg a_i)$-path.

Thus, every iteration ends with a valid assignment. Moreover, by setting a_i to 1, s satisfies C. Moreover, no previously satisfied clause $(x \vee y)$ becomes unsatisfied, by setting both $\neg x$ and $\neg y$ to 1: if, e.g., $\neg x$ is reachable from a_i, then so is y via the arc $(\neg x, y)$. Thus, setting $\neg x$ to 1 means that y will also be set to 1. Therefore, the described procedure strictly increases the number of satisfied clauses in each iteration and it stops when all clauses from C are satisfied, resulting in a satisfying assignment for G_{F^*}. ◁

This proves that the algorithm described above computes a set P_C in polynomial time that satisfies all three preconditions for Lemma 3.1. In particular, our algorithm satisfies these constraints with $c = 2d^2$. By Lemma 3.1, there is a polynomial-time $(2d^2 + 1)$-detection algorithm for MinCSP($\mathcal{F}$). ◀

We recall that FPT algorithms, parameterized by solution size, are known for MinCSP($\mathcal{F}$) on bijunctive constraint sets $\mathcal{F}$ if and only if the so-called Gaifman graph of $\mathcal{F}$ is $2K_2$-free [15]. So, by a result analogous to one from the original work [6, Theorem 5.1] on c-essential detection, Lemma 3.4 effectively shows that we can reduce the search-space of these FPT algorithms. In particular, it shows that their exponential runtime dependence on the total size of an optimal solution can be improved to depend only on the number of constraint applications in an optimal solution that are *not* c-essential.

4 Hardness results

Our hardness results all follow the same overarching structure. For a given category of constraint sets, we first give a canonical set of constraints $\mathcal{F}$ from that category and prove for the corresponding MinCSP($\mathcal{F}$) problem that c-essential detection is hard for all constants $c \in \mathbb{R}_{\geq 1}$. Then, we show that there is a reduction from this problem to every other MinCSP problem for constraint sets in the same category. This reduction is constructive. To achieve this, we show that every constraint set from a given category can be used to express the constraints from the canonical set. Under the right restrictions, this immediately shows the reducibility from the canonical set to the other sets in the category. These restrictions are given in the notion of *essential implementations* as defined below.

▶ **Definition 4.1.** *Let $f : \{0,1\}^p \to \{0,1\}$ be a constraint over p variables. A collection $\mathcal{C}$ of constraint applications $C_1, \ldots, C_m$ over variables $\mathbf{X} = \{x_1, \ldots, x_p\}$ and dummy variables $\mathbf{Y} = \{y_1, \ldots, y_q\}$ is an* essential implementation *of the constraint application $(f, \mathbf{X})$ if:*

(I1) *an assignment to $\mathbf{X}$ satisfies $(f, \mathbf{X})$ if and only if there is an extension of this assignment to $\mathbf{X} \cup \mathbf{Y}$ that satisfies all constraint applications $C_1, \ldots, C_m$; and*

(I2) *for every assignment to $\mathbf{X}$, there is an extension to $\mathbf{X} \cup \mathbf{Y}$ that satisfies $C_2, \ldots, C_m$. (Note the exclusion of C_1, which we call the* head *of the implementation.)*

We say that a constraint set $\mathcal{F}$ *essentially implements* a constraint f if f admits an essential implementation via constraint applications that are each from $\mathcal{F}$. We denote this as $\mathcal{F} \xrightarrow{\text{ess.}} f$. We say that a constraint set $\mathcal{F}_1$ essentially implements a constraint set $\mathcal{F}_2$ if $\mathcal{F}_1$ essentially implements each of the constraints from $\mathcal{F}_2$. We denote this as $\mathcal{F}_1 \xrightarrow{\text{ess.}} \mathcal{F}_2$. In both arrow notations, if $\mathcal{F}_1$ is a singleton set, we may replace it by just its singleton element.

This notion of essential implementations is stricter than, e.g., the notion of pp-definitions, which are similarly defined but without requiring Property **(I2)**. Such definitions facilitate reductions between formulas of different constraint sets that preserve the satisfiability of the formulas, as for example used by Schaefer [21]. Even more similar is the notion of α-implementations as defined by Khanna et al. [13], especially those that are both "strict" and "perfect". These implementations only differ from our Definition 4.1 by allowing different assignments to $\mathbf{X}$ that do not satisfy f to extend to an assignment on $\mathbf{X} \cup \mathbf{Y}$ that leaves a different (but still singular) constraint application C_i unsatisfied. This uncertainty in which constraint application remains unsatisfied leads to changes in the structure of the solution space that makes these implementations unsuitable for our purposes. However, the authors use their definition of α-implementations to construct reductions that preserve the constant-factor approximability of MinCSP($\mathcal{F}$).

Our definition of essential implementations allows for a reduction that also preserves the exact approximation-ratio for constant factor-approximable instances. In fact, it even preserves the entire space of c-approximate solutions in a predictable and useful manner. This is needed to prove the following result.

▶ **Lemma 4.2 (★).** *Let $\mathcal{F}_1$ and $\mathcal{F}_2$ be constraint sets such that $\mathcal{F}_1 \xrightarrow{\text{ess.}} \mathcal{F}_2$, and let $c \in \mathbb{R}_{\geq 1}$ be a constant. If MinCSP($\mathcal{F}_1$) admits a polynomial-time c-essential detection algorithm, then MinCSP($\mathcal{F}_2$) also admits a polynomial-time c-essential detection algorithm.*

Proof sketch. We prove the statement by showing how to reduce an $\mathcal{F}_2$-formula to a weighted $\mathcal{F}_1$-formula in such a way that a polynomial-time c-essential detection algorithm on the $\mathcal{F}_1$-formula can be used to detect c-essential constraint applications in the $\mathcal{F}_2$-formula. By Lemma 2.1, this suffices to prove the lemma.

Let F_2 be an $\mathcal{F}_2$-formula with constraint applications $C_1, \ldots, C_\ell$ over variables $\mathbf{X} = \{x_1, \ldots, x_n\}$. To transform F_2 into a weighted $\mathcal{F}_1$-formula F_1, we replace every C_i in F_2 by an essential implementation $\mathcal{C}_i$ of constraint applications from $\mathcal{F}_1$ over variables $\mathbf{X} \cup \mathbf{Y}_i$. We denote the head of $\mathcal{C}_i$ by C_i^*. We give all constraint applications a weight of $\lceil c \cdot \ell \rceil + 1$, except for the heads $C_1^*, \ldots, C_\ell^*$ that each receive a weight of 1. We define $\mathbf{Y} := \bigcup_{i \in [\ell]} \mathbf{Y}_i$.

The first step towards proving the correctness of this reduction is to observe that non-head constraint applications in F_1 do not appear in any c-approximate solution. After all, it follows from Property **(I2)** that the set of heads $H := \{C_1^*, \ldots, C_\ell^*\}$, with total weight ℓ, makes $F - H$ satisfiable, whereas every other constraint has weight $\lceil c \cdot \ell \rceil + 1$. Now, the following claim effectively shows that the set H spans the exact same space of c-approximate solutions to F_1 as the set of original constraint applications $C_1, \ldots, C_\ell$ does for F_2.

▷ Claim 4.3. Let S be a set of constraint applications in F_2 and let $S_H \subseteq H$ be the corresponding set of heads in F_1. Then, $F_2 - S$ is satisfiable if and only if $F_1 - S_H$ is satisfiable.

Proof. For one direction, let s be an assignment on $\mathbf{X}$ that satisfies $F_2 - S$. For every $i \in [\ell]$ such that $C_i \in S$, assignment s can be extended to an assignment on $\mathbf{X} \cup \mathbf{Y}_i$ that satisfies all constraint applications in $\mathcal{C} \setminus \{C_i^*\}$, due to Property **(I2)**. For every $i \in [\ell]$ such that $C_i \notin S$, since s satisfies C_i, assignment s can be extended to an assignment on $\mathbf{X} \cup \mathbf{Y}_i$ that satisfies all constraint applications in $\mathcal{C}_i$, due to Property **(I1)**. This covers all constraint applications in $F_1 - S_H$, showing that an extension of s to $\mathbf{X} \cup \mathbf{Y}$ exists that satisfies $F_1 - S_H$.

For the other direction, observe that any assignment s' on $\mathbf{X} \cup \mathbf{Y}$ that satisfies $F_1 - S_H$, satisfies the constraint applications in $\mathcal{C}_i$ for every i such that $C_i \notin S$. Thus, by Property **(I1)**, the restriction of s' to $\mathbf{X}$ satisfies $F_2 - S$. ◁

Since every head has weight 1, it follows that $\mathrm{opt}(F_1) = \mathrm{opt}(F_2)$. Now we show how to use a polynomial-time c-essential detection algorithm $\mathcal{A}$ for Weighted MinCSP($\mathcal{F}_1$) to construct a polynomial-time c-essential detection algorithm for MinCSP($\mathcal{F}_2$).

Given an $\mathcal{F}_2$-formula F_2 and an integer k, we start by transforming it into a weighted $\mathcal{F}_1$-formula F_1 using the reduction above; the weights are polynomial. Then, we apply $\mathcal{A}$ to F_1 and let D_1 denote its output. If D_1 contains non-head constraint applications of F_1, we return the empty set. Otherwise, we return the set of constraint applications C_i for which the corresponding head C_i^* is in D_1. In either case, let D_2 be the output of our algorithm.

It remains to show that the above is a c-essential detection algorithm for F_2. First, suppose that $\mathrm{opt}(F_1) = \mathrm{opt}(F_2) \leq k$. Since D_1 satisfies Property **(G1)** it is a subset of some optimal solution S_1 to F_1. S_1, and by extension D_1 must consist entirely of heads. Then, the set S_2 of constraint applications in F_2 for which the corresponding head is in S_1 must be a superset of D_2. By Claim 4.3, S_2 is an optimal solution to F_2, showing that D_2 satisfies Property **(G1)**.

Next, suppose that $\mathrm{opt}(F_1) = \mathrm{opt}(F_2) = k$. Since D_1 satisfies Property **(G2)**, it contains all c-essential constraint applications from F_1. Since the set of heads was shown to preserve the entire space of c-approximate solutions, a constraint application C_i in F_2 is c-essential if and only if its corresponding head C_i^* is c-essential in F_1. Since D_2 contains the set of heads corresponding to D_1, it contains all c-essential constraint applications in F_2. ◀

Next, we prove that the notion of essential implementations is a transitive relation.

▶ **Lemma 4.4.** *Let $\mathcal{F}_1$, $\mathcal{F}_2$, and $\mathcal{F}_3$ be constraint sets. If $\mathcal{F}_1 \xrightarrow{\text{ess.}} \mathcal{F}_2$ and $\mathcal{F}_2 \xrightarrow{\text{ess.}} \mathcal{F}_3$, then $\mathcal{F}_1 \xrightarrow{\text{ess.}} \mathcal{F}_3$.*

Proof sketch. Let $(f_3, \mathbf{X})$ be an arbitrary constraint application from $\mathcal{F}_3$ over variables $\mathbf{X} = \{x_1, \ldots, x_p\}$. To prove the lemma, we show that $\mathcal{F}_1 \xrightarrow{\text{ess.}} f_3$.

Since $\mathcal{F}_2 \xrightarrow{\text{ess.}} \mathcal{F}_3$, there is an essential implementation of $(f_3, \mathbf{X})$ consisting of constraint applications $C_1, \ldots, C_m$ from $\mathcal{F}_2$ over variables $\mathbf{X}$ and shared dummy variables $\mathbf{Y}$. Let C_1 be the head of this implementation. Since $\mathcal{F}_1 \xrightarrow{\text{ess.}} \mathcal{F}_2$, there is, for each $i \in [m]$, an essential implementation $\mathcal{C}_i$ of C_i consisting of constraint applications from $\mathcal{F}_1$ over variables $\mathbf{X} \cup \mathbf{Y}$ and dummy variables $\mathbf{Z}_i$. Let $C_i^* \in \mathcal{C}_i$ denote the head of $\mathcal{C}_i$. Note that the constraint applications in one $\mathcal{C}_i$ share one set of dummy variables $\mathbf{Z}_i$, but that this set differs between different implementations $\mathcal{C}_i$.

We show that the concatenation of all constraint applications in $\mathcal{C}_i, \ldots, \mathcal{C}_m$ is an essential implementation of f_3 with head C_1^*. It is easily verified that this satisfies Property **(I1)**, so we continue by showing the more interesting fact that it satisfies Property **(I2)**.

Consider an arbitrary assignment s on $\mathbf{X}$. By Property **(I2)** of implementation $C_1, \ldots, C_m$, it extends to an assignment s' on $\mathbf{X} \cup \mathbf{Y}$ that satisfies $C_2, \ldots, C_m$. By Property **(I1)** of the implementations $\mathcal{C}_2, \ldots, \mathcal{C}_m$, s' extends to an assignment on $\mathbf{X} \cup \mathbf{Y} \cup (\mathbf{Z} \setminus \mathbf{Z}_1)$ that satisfies all constraint applications in $\mathcal{C}_2, \ldots, \mathcal{C}_m$. By Property **(I2)** of the implementation $\mathcal{C}_1$, every assignment on $\mathbf{X} \cup \mathbf{Y}$ – including s' – extends to an assignment on $\mathbf{X} \cup \mathbf{Y} \cup \mathbf{Z}_1$ that satisfies $\mathcal{C}_1 \setminus \{C_1^*\}$. Thus, s extends to an assignment s' that in turn extends to an assignment on $\mathbf{X} \cup \mathbf{Y} \cup \mathbf{Z}$ that satisfies $\mathcal{C} \setminus \{C_1^*\}$, proving that $\mathcal{C}$ satisfies Property **(I2)**. ◀

Next, in Sections 4.1–4.3, we prove Cases 3, 4, and 5 from Theorem 1.1. Due to space limitations, the proofs in Sections 4.1 and 4.3 are deferred to the full version. The proofs in Section 4.2 also showcase the general proving strategies used in the other two sections, but we remark that each case of the dichotomy represents a significant effort in tailoring these strategies to their specific setting. Although we consider the introduction and use of essential implementations to be one of our main contributions, this also means that our proofs show only two examples of specific essential implementations.

4.1 Affine constraint sets

We pick $\{\text{XOR}_4, \text{XNOR}_4\}$ as canonical constraint set for the category of affine constraint sets that are neither 0-valid, 1-valid nor bijunctive. First, we prove that c-essential Detection for MinCSP$(\{\text{XOR}_4, \text{XNOR}_4\})$ is NP-hard for every $c \in \mathbb{R}_{\geq 1}$ under the Unique Games Conjecture (UGC). As starting point for our hardness reduction, we use the fact that distinguishing between two regimes of solution sizes for gap instances of MinCSP$(\{[=], [\neq]\})$ is NP-hard under the UGC [14]. We give a reduction that transforms an $\{[=], [\neq]\}$-formula F_1 into an $\{\text{XOR}_4, \text{XNOR}_4\}$-formula F_2 in such a way that we can use the output of a c-essential detection algorithm on F_2 to determine whether F_1 admits a small or large optimal solution. For example, each arity-2 constraint over $[\neq]$ turns into an arity-4 constraint over XOR_4 by inserting two global variables z_1, z_2 to which the constraint is additionally applied. Their contributions cancel whenever $z_1 = z_2$ and we add a medium-weight constraint enforcing this behavior. This proves c-essential detection to be hard for MinCSP$(\{\text{XOR}_4, \text{XNOR}_4\})$ and the details of this reduction are given in the full version.

Then, if $\mathcal{F}$ is any affine constraint set that is neither 0-valid, 1-valid nor bijunctive, we show that $\mathcal{F} \xrightarrow{\text{ess.}} \{\text{XOR}_4, \text{XNOR}_4\}$. To start, we use an existing result that shows that $\mathcal{F}$ can be written as conjunction of XOR and XNOR constraints that are each implemented by $\mathcal{F}$ [13, Lemma 4.18]. It is easily seen that the provided implementations in this work are even essential. Then, since $\mathcal{F}$ is not bijunctive, it implements at least one XOR or XNOR constraint over 3 or more variables. We also note that XNOR_q is 0-valid for all integers q and that XOR_p

and XNOR_q are 1-valid for odd values of p and even values of q. Combining these insights with several sequences of new implementations, we can show that $\mathcal{F} \xrightarrow{\text{ess.}} \{\mathrm{XOR}_4, \mathrm{XNOR}_4\}$, the details of which are given in the full version.

The two results above combine to prove Case 3 of Theorem 1.1.

4.2 Weakly positive and weakly negative constraint sets

In this section, we prove Case 4 of Theorem 1.1. Throughout this section, we assume to be working with weakly positive constraint sets, rather than weakly negative ones. Proofs for weakly negative constraint sets follow symmetrically. We pick $\mathcal{F}_{\text{can}} := \{\mathrm{OR}_{2,1}, \mathrm{OR}_{3,1}, \textsc{True}, \textsc{False}\}$ as canonical constraint set and we start by showing that c-essential Detection for $\mathrm{MinCSP}(\mathcal{F}_{\text{can}})$ is NP-hard in Section 4.2.1. Then, in Section 4.2.2, we show for any weakly positive constraint set $\mathcal{F}$ that is neither 0-valid, 1-valid, bijunctive, nor IHS-B, that $\mathcal{F} \xrightarrow{\text{ess.}} \mathcal{F}_{\text{can}}$. (Note that these conditions imply $\mathcal{F}$ is not affine either.)

4.2.1 Hardness of c-essential detection for the canonical problem

We give a reduction from the Set Cover optimization problem. In this problem, a universe U is given together with a collection $\mathcal{S}$ of subsets of U, and the goal is to determine the size of a smallest subset $Y \subseteq \mathcal{S}$ such that every $u \in U$ is contained in at least one set in Y. We call such a set Y (of any size) a *set cover* of $(U, \mathcal{S})$, and we denote the size of a smallest set cover of $(U, \mathcal{S})$ by $\mathrm{opt}(U, \mathcal{S})$. The following hardness result is known.

▶ **Lemma 4.5** ([2, Theorem 22.31]). *For every sufficiently small constant $\varepsilon > 0$, the following holds. Unless P=NP, there is no polynomial-time algorithm that takes an integer $t \geq 1$ and a Set Cover instance $(U, \mathcal{S})$ as input and distinguishes between the following two cases:*

(i) $\mathrm{opt}(U, \mathcal{S}) \leq t$;

(ii) $\mathrm{opt}(U, \mathcal{S}) > \frac{t}{4\sqrt{\varepsilon}}$.

We use this result to prove the statement below.

▶ **Lemma 4.6.** *Unless P=NP, there is no constant $c \in \mathbb{R}_{\geq 1}$ for which a polynomial-time c-essential detection algorithm exists for MinCSP($\{\mathrm{OR}_{2,1}, \mathrm{OR}_{3,1}, \textsc{True}, \textsc{False}\}$).*

Proof. Let $\mathrm{HORN}_p := \left(\bigcup_{i \in \{2 \ldots p\}} \mathrm{OR}_{i,1}\right) \cup \{\textsc{True}, \textsc{False}\}$, so that the lemma states hardness of c-essential detection for $\mathrm{MinCSP}(\mathrm{HORN}_3)$. We start by showing that the existence of a polynomial-time c-essential detection algorithm for Weighted $\mathrm{MinCSP}(\mathrm{HORN}_\infty)$ implies that P=NP. We assume that such an algorithm exists for an arbitrary value of $c \in \mathbb{R}_{\geq 1}$ and argue that it can be used to solve the distinguishing task from Lemma 4.5 in polynomial time. Afterwards, we explain how to extend this result to HORN_3.

Now, let $(U, \mathcal{S})$ be a Set Cover instance, and let $t \geq 1$ be an arbitrary integer. We let m denote the maximum size of a set in $\mathcal{S}$, and show how to transform $(U, \mathcal{S})$ into a weighted HORN_{m+1}-formula F.

We start by introducing two variables z and w and adding constraint applications $\textsc{True}(w)$ with weight $c(t+1)+1$ and $\textsc{False}(z)$ with weight $t+1$ to F. Then, for every set $S_j \in \mathcal{S}$, we introduce a variable y_j and we add a constraint application $(\neg y_j \vee z)$ to F with weight 1.

Next, for every $u_i \in U$, we introduce a variable x_i and we add $(x_i \vee \neg w)$ to F with weight $c(t+1)+1$. Additionally, if we let $S_{j_1}, \ldots, S_{j_\ell}$ be the collection of sets that u_i appears in, we add $(\neg x_i \vee y_{j_1} \vee \ldots \vee y_{j_\ell})$ to F with weight $c(t+1)+1$.

This concludes the reduction. Observe that the resulting formula F is a HORN$_{m+1}$-formula whose size is polynomial in the total size of the original Set Cover instance. We continue by proving the correctness of our reduction, starting with the following claim.

▷ Claim 4.7. A collection $\mathcal{C} := S_{j_1}, \ldots, S_{j_r}$ is a set cover of $(U, \mathcal{S})$ if and only if the collection of constraint applications $X_\mathcal{C} := \{(\neg y_{j_1} \vee z), \ldots, (\neg y_{j_r} \vee z)\}$ is such that $F - X_\mathcal{C}$ is satisfiable.

Proof. For one direction, suppose that $\mathcal{C}$ is a set cover of $(U, \mathcal{S})$. Next, consider the assignment s to the variables of F that sets w to 1, z to 0, all variables x_i to 1, the variables y_j for which $S_j \in \mathcal{C}$ to 1, and the variables y_j for which $S_j \notin \mathcal{C}$ to 0.

Clearly, True(w) and False(z) are satisfied by this assignment. All constraint applications of the form $(\neg y_j \vee z)$ in $F - X_\mathcal{C}$ are also satisfied because all variables y_j for which $S_j \notin \mathcal{C}$ are set to 0. Finally, note that, because $\mathcal{C}$ is a set cover of U, every constraint application of the form $(\neg x_i \vee y_{j_1} \vee \ldots \vee y_{j_\ell})$ contains at least one y_j for which $S_j \in \mathcal{C}$. All such variables are set to 1, so all constraint applications of this form in $F - X_\mathcal{C}$ are also satisfied by s.

For the other direction, suppose that $F - X_\mathcal{C}$ is satisfiable and let s be a satisfying assignment. To satisfy True(w) and False(z), we must have $s(w) = 1$ and $s(z) = 0$. Then, to satisfy all constraint applications of the form $(x_i \vee \neg w)$, we must have $s(x_i) = 1$ for all variables x_i. Likewise, to satisfy all constraint applications in $F - X_\mathcal{C}$ of the form $(\neg y_j \vee z)$, we must have $s(y_j) = 0$ for all variables y_j with $S_j \notin \mathcal{C}$.

Next, consider an arbitrary constraint application of the form $(\neg x_i \vee y_{j_1} \vee \ldots \vee y_{j_\ell})$. Since s satisfies this constraint application but also has $s(x_i) = 1$ it must set at least one of the variables $y_{j_1}, \ldots, y_{j_\ell}$ to 1. However, since s sets all variables y_j for which $S_j \notin \mathcal{C}$ to 0, at least one of the variables $y_{j_1}, \ldots, y_{j_\ell}$ must correspond to a set from $\mathcal{S}$ that is in $\mathcal{C}$. Thus, we obtain that for every variable x_i, the constraint application $(\neg x_i \vee y_{j_1} \vee \ldots \vee y_{j_\ell})$ contains at least one variable y_j for which $S_j \in \mathcal{C}$. This means that $\mathcal{C}$ is a set cover of $(U, \mathcal{S})$. ◁

Now, we define $\varepsilon := \frac{1}{65c^2}$ and use the claim above to show that a c-essential detection algorithm for MinCSP(HORN$_p$) on input F can be used to determine whether the original Set Cover input $(U, \mathcal{S})$ satisfies $\text{opt}(U, \mathcal{S}) \leq t$ or $\text{opt}(U, \mathcal{S}) > \frac{t}{4\sqrt{\varepsilon}}$. To this end, let S be the result of a c-essential detection algorithm for MinCSP(HORN$_p$) on input F with $k = t + 1$. To recognize the first case, consider the following claim.

▷ Claim 4.8. If $\text{opt}(U, \mathcal{S}) \leq t$, then S does not contain the constraint False(z).

Proof. Let $\mathcal{C} \subseteq \mathcal{S}$ be a set cover of $(U, \mathcal{S})$ of size at most t. Then, by Claim 4.7, the corresponding set $X_\mathcal{C}$ is such that $F - X_\mathcal{C}$ is satisfiable. Since all constraint applications in $X_\mathcal{C}$ have weight 1, this constraint set is a solution to F of weight t. Thus, $\text{opt}(F) \leq t < k$. As such, S must be a subset of an optimal solution to F to satisfy Property **(G1)**. Since an optimal solution of F has weight $\leq t$ and the constraint application False(z) has weight $t + 1$, False(z) is not in any optimal solution and by extension also not in S. ◁

We continue by showing how the set S differs for the other case.

▷ Claim 4.9. If $\text{opt}(U, \mathcal{S}) > \frac{t}{4\sqrt{\varepsilon}}$, then S does contain the constraint False(z).

Proof. First, we argue that the singleton set $\{\text{False}(z)\}$ of weight $t + 1$ is a solution to F. To see this, note that every constraint application in $F - \{\text{False}(z)\}$ contains at least one positive literal so that the all-one assignment satisfies it. We continue by showing that solutions to F that do not contain False(z) have a weight of more than $c \cdot (t + 1)$.

Let X be an arbitrary such solution and suppose for contradiction that it has a weight of $\leq c \cdot (t+1)$. Recall that the only constraint applications that do not individually exceed this weight are $\textsc{False}(z)$ and the weight-1 constraint applications of the type $(\neg y_i \vee z)$. As such, X consists solely of such weight-1 constraints.

However, by Claim 4.7, a solution that only consists of these weight-1 constraint applications corresponds to a set cover of $(U, \mathcal{S})$ of equal size. Therefore, X must be at least of size $\text{opt}(U, \mathcal{S})$, which, by assumption, is larger than $\frac{t}{4\sqrt{\varepsilon}}$. We defined $\varepsilon = \frac{1}{65c^2}$, so by rewriting the inequality $\varepsilon < \frac{1}{64c^2}$, we obtain that $\frac{1}{4\sqrt{\varepsilon}} > 2c$, which in turn yields that $\frac{t}{4\sqrt{\varepsilon}} > 2ct$. Since $2t \geq t+1$ for all $t \geq 1$, we get that $|X| \geq \text{opt}(U, \mathcal{S}) > \frac{t}{4\sqrt{\varepsilon}} > c \cdot (t+1)$. This contradicts our assumption that X has weight at most $c \cdot (t+1)$.

We conclude that all solutions to F that do not contain $\textsc{False}(z)$ are more than c times as heavy as $\textsc{False}(z)$ itself. This means that $\{\textsc{False}(z)\}$ is an optimal solution of weight $(t+1)$ and even that $\textsc{False}(z)$ is c-essential. As $k = t+1 = \text{opt}(F)$, S must contain all c-essential constraint applications to satisfy Property **(G1)**. Hence, S contains $\textsc{False}(z)$. ◁

Now, the two claims above indicate how a polynomial-time c-essential detection algorithm for F can be used to distinguish between $\text{opt}(U, \mathcal{S}) \leq t$ and $\text{opt}(U, \mathcal{S}) > \frac{t}{4\sqrt{\varepsilon}}$ in polynomial time as well: after running such an algorithm, it suffices to check whether the output contains $\textsc{False}(z)$. By Lemma 4.5, this distinction is NP-hard to make. As $c \in \mathbb{R}_{\geq 1}$ was chosen arbitrarily, this shows that c-essential detection for Weighted MinCSP(HORN_∞) is NP-hard for every $c \in \mathbb{R}_{\geq 1}$. By Lemma 2.1, the same holds for the unweighted variant of the problem.

It remains to show that the same is true for MinCSP(HORN_3). To this end, we show how to transform the HORN_{m+1}-formula F into a HORN_3-formula whose size is polynomial in the size of F. We note that, for any integer $p \geq 2$, $\{\text{OR}_{2,1}, \text{OR}_{3,1}\} \xrightarrow{\text{ess.}} \text{OR}_{p,1}$. This can be seen by inductively applying the following claim.

▷ Claim 4.10. Let $p \geq 4$. Then $\{\text{OR}_{p-1,1}, \text{OR}_{3,1}\} \xrightarrow{\text{ess.}} \text{OR}_{p,1}$.

Proof. We show that $(\neg x_1 \vee x_2 \vee \ldots \vee x_{p-2} \vee v) \wedge (\neg v \vee x_{p-1} \vee x_p)$, with dummy variable v and the first clause as its head, is an essential implementation of $(\neg x_1 \vee x_2 \vee \ldots \vee x_p)$. It is easily seen that this implementation satisfies Property **(I1)** by verifying that $(\neg x_1 \vee x_2 \vee \ldots \vee x_{p-2} \vee v) \wedge (\neg v \vee x_{p-1} \vee x_p)$ is satisfied if and only if $(\neg x_1 \vee x_2 \vee \ldots \vee x_p)$ is satisfied.

Next, note that every assignment to the variables $x_1, \ldots, x_p$ extends to a satisfying assignment for $(\neg v \vee x_{p-1} \vee x_p)$ by setting v to 0. Hence, Property **(I2)** is satisfied. ◁

Now, a single essential implementation of $\text{OR}_{p,1}$ can be made using $\mathcal{O}(p)$ applications of $\text{OR}_{2,1}$ and $\text{OR}_{3,1}$, since the construction in Claim 4.10 shows that we can build an $\text{OR}_{p,1}$ application from one (constant-size) $\text{OR}_{3,1}$ application and only one $\text{OR}_{p-1,1}$ application.

To transform F into a HORN_3-formula, we replace every HORN_p application in it with $p > 3$ by an essential implementation of $\text{OR}_{2,1}$ and $\text{OR}_{3,1}$ applications in which the head receives weight 1 and the other applications receive weight $c(t+1)+1$. Then, the same logic as in Lemma 4.2 shows that a polynomial-time c-essential detection algorithm for the resulting HORN_3-formula can be used to perform c-essential detection for the original formula F in polynomial time. This, in turn, was shown to imply P=NP. ◀

4.2.2 Essential implementation of the canonical constraint set

This section is dedicated to proving the following result.

▶ **Lemma 4.11.** *Let $\mathcal{F}$ be a constraint set that is neither 0-valid, nor 1-valid nor IHS-B. If $\mathcal{F}$ is weakly positive, then $\mathcal{F} \xrightarrow{\text{ess.}} \{\text{OR}_{2,1}, \text{OR}_{3,1}, \textsc{True}, \textsc{False}\}$.*

Proof sketch. Using a characterization of weakly positive constraint sets [13, Lemma 4.20], we can prove that $\mathcal{F} \xrightarrow{\text{ess.}} \{\textsc{True}, \textsc{False}\}$. Then, we can follow a sequence of implementations by Khanna et al. almost directly to show that $\mathcal{F} \cup \{\textsc{True}, \textsc{False}\} \xrightarrow{\text{ess.}} \{\text{OR}_{2,1}, \text{OR}_{3,1}\}$. Almost all implementations in the proof of Lemma 7.18 in their work happen to be essential as well [13]. The only exception is that their provided implementation $(\neg x \vee y) \wedge (\neg y \vee x)$ of $(x = y)$ is not essential. Note that it does not satisfy Property **(I2)**. Using a more complicated implementation, we can however still show that $\text{OR}_{2,1} \xrightarrow{\text{ess.}} [=]$. In particular, we note that $(\neg v_1 \vee v_2) \wedge (\neg x \vee v_1) \wedge (\neg y \vee v_1) \wedge (\neg v_2 \vee x) \wedge (\neg v_2 \vee y)$ is an essential implementation of $(x = y)$ with dummy variables v_1 and v_2 and the first clause as head. Using a simple case distinction, one can easily check that it satisfies Property **(I1)**. Moreover, any assignment to x and y can be extended to satisfy the last four clauses by setting v_1 to 1 and v_2 to 0, showing that the implementation satisfies Property **(I2)**. ◀

Now, Lemmas 4.6 and 4.11 combine to prove Case 4 of Theorem 1.1.

4.3 All other constraint sets

For constraint sets $\mathcal{F}$ that do not belong to any of the categories specified in the first four cases of Theorem 1.1, we pick $\{\text{NAE}_5, [\neq], [=]\}$ as the canonical constraint set. First, we prove that c-essential Detection for MinCSP$(\{\text{NAE}_5, [\neq], [=]\})$ is NP-hard for every $c \in \mathbb{R}_{\geq 1}$, even on formulas F with $\text{opt}(F) = 1$. To achieve this, we give a reduction that transforms a 3-SAT instance F_1 into a weighted $\{\text{NAE}_5, [\neq], [=]\}$-formula F_2.

We replace every 3CNF clause in F_1 by a large-weight NAE_5-constraint application in F_2 by supplementing it with two global dummy variables w_1 and w_2. We use large-weight $[\neq]$ constraints to model negated variables in the original 3CNF clauses. Then, including a weight-1 constraint application that requires that $(w_1 = w_2)$ ensures that F_1 is satisfiable if and only if F_2 is satisfiable. Moreover, if F_1 is not satisfiable, we can make F_2 satisfiable by removing the constraint application $(w_1 = w_2)$. Then, the remaining NAE_5-constraint applications in F_2 can indeed be satisfied by setting w_1 and w_2 unequal. Since all other constraint applications have a large weight, this makes $(w_1 = w_2)$ essential and it means that we can determine whether F_1 is satisfiable from the output of a c-essential detection algorithm on F_2. The details of this are given in the full version.

Secondly, if $\mathcal{F}$ is any constraint set that is neither 0-valid, 1-valid, IHS-B, bijunctive, affine, weakly positive, nor weakly negative, we show that $\mathcal{F} \xrightarrow{\text{ess.}} \{\text{NAE}_5, [\neq], [=]\}$. To this end, we extend a reduction by Schaefer, which is split into two cases, depending on whether or not all constraints in $\mathcal{F}$ are complementive [21]. (A constraint is complementive if the complement of a satisfying assignment is also satisfying.) If not, we can – with a few minor modifications and extensions – follow Schaefer's sequence of implementations rather directly as it consists mostly of implementations that are already essential. For complementive constraint sets however, we provide a new sequence of implementations to ensure they are essential. The details of this are given in the full version and, together with the hardness of c-essential Detection for MinCSP$(\{\text{NAE}_5, [\neq], [=]\})$, this proves Case 5 of Theorem 1.1.

5 Conclusion and Discussion

We extended the framework of search-space reduction via essential objects beyond the setting of graph problems for which it was originally defined. Investigating the task of detecting c-essential constraint applications for MinCSP$(\mathcal{F})$ has resulted in a dichotomy

that characterizes for which constraint sets $\mathcal{F}$ this is polynomial-time solvable. We note the similarity between our dichotomy theorem and the dichotomy theorem from Bonnet et al. [5] regarding the constant-factor approximability of MinCSP($\mathcal{F}$) in FPT time. Both dichotomies yield positive results for exactly the same classes of constraint sets. We leave it to future work to investigate whether this is a coincidence or a consequence of a deeper connection between the two tasks.

Other directions for follow-up research include extending the results of this dichotomy to CSPs beyond the Boolean domain. In fact, successfully applying the idea of c-essentiality to a problem domain different from graph theory has solidified the framework as a robust method by which meaningful and non-trivial new results can be attained across problem domains. As such, one could also attempt to find results using it in even different areas such as scheduling or computational geometry.

While our dichotomy theorem characterizes when c-essential detection is possible for *some* constant c, it does not shed any light on what the smallest value of c is for which this task is tractable, nor on the relation between this value and the choice of constraint set $\mathcal{F}$. We leave this to future work. A concrete question in this direction is to determine the smallest constant $c_{\mathcal{F}}$ for which $c_{\mathcal{F}}$-essential Detection for MinCSP($\mathcal{F}$) is possible for bijunctive constraint languages $\mathcal{F}$. Our approach in Lemma 3.4 of solving several cut problems in the implication graph gives a guarantee of $c_{\mathcal{F}} = (2d^2 + 1)$ whenever each constraint in $\mathcal{F}$ can be expressed as a 2CNF-formula with d clauses, but we do not expect this to be best-possible. Perhaps a discrete relaxation of the cut problem, like those studied in earlier work on Almost 2-SAT [9, 10], can provide a better bound?

A final open question concerns the theoretical basis for the impossibility of $c_{\mathcal{F}}$-essential Detection for affine constraint languages $\mathcal{F}$. We currently rely on the Unique Games Conjecture to rule out $c_{\mathcal{F}}$-detection algorithms for this family, via known lower bounds for MinCSP($\{[=], [\neq]\}$) [14]. Can the same lower bound be established relative to the standard assumption P $\neq$ NP, for example via known hardness of approximation results for Nearest Codeword [4]?

References

1 Tobias Achterberg, Robert E. Bixby, Zonghao Gu, Edward Rothberg, and Dieter Weninger. Presolve reductions in mixed integer programming. Technical Report 16-44, ZIB, Takustr.7, 14195 Berlin, 2016. URL: http://nbn-resolving.de/urn:nbn:de:0297-zib-60370.

2 Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009. doi:10.1017/CBO9780511804090.

3 Bengt Aspvall, Michael F. Plass, and Robert Endre Tarjan. A linear-time algorithm for testing the truth of certain quantified boolean formulas. *Inf. Process. Lett.*, 8(3):121–123, 1979. doi:10.1016/0020-0190(79)90002-4.

4 Vijay Bhattiprolu, Venkatesan Guruswami, and Xuandi Ren. PCP-free APX-hardness of nearest codeword and minimum distance. *Electron. Colloquium Comput. Complex.*, TR25, 2025. URL: https://eccc.weizmann.ac.il/report/2025/029.

5 Édouard Bonnet, László Egri, and Dániel Marx. Fixed-parameter approximability of Boolean MinCSPs. In Piotr Sankowski and Christos D. Zaroliagis, editors, *24th Annual European Symposium on Algorithms, ESA 2016, Aarhus, Denmark, August 22-24, 2016*, volume 57 of *LIPIcs*, pages 18:1–18:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.ESA.2016.18.

6 Benjamin Merlin Bumpus, Bart M. P. Jansen, and Jari J. H. de Kroon. Search-space reduction via essential vertices. *SIAM J. Discret. Math.*, 38(3):2392–2415, 2024. doi:10.1137/23M1589347.

7 Fedor Fomin, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. *Kernelization: theory of parameterized preprocessing*. Cambridge University Press, 2019.

8 L. R. Ford and D. R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8:399–404, 1956. doi:10.4153/CJM-1956-045-5.

9 Yoichi Iwata, Magnus Wahlström, and Yuichi Yoshida. Half-integrality, lp-branching, and FPT algorithms. *SIAM J. Comput.*, 45(4):1377–1411, 2016. doi:10.1137/140962838.

10 Yoichi Iwata, Yutaro Yamaguchi, and Yuichi Yoshida. 0/1/All CSPs, half-integral A-path packing, and linear-time FPT algorithms. In Mikkel Thorup, editor, *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 462–473. IEEE Computer Society, 2018. doi:10.1109/FOCS.2018.00051.

11 Bart M. P. Jansen and Ruben F. A. Verhaegh. Search-space reduction via essential vertices revisited: Vertex multicut and cograph deletion. *J. Comput. Syst. Sci.*, 156:103730, 2026. doi:10.1016/J.JCSS.2025.103730.

12 Bart M. P. Jansen and Michal Wlodarczyk. Optimal polynomial-time compression for Boolean max CSP. *ACM Trans. Comput. Theory*, 16(1):4:1–4:20, 2024. doi:10.1145/3624704.

13 Sanjeev Khanna, Madhu Sudan, Luca Trevisan, and David P. Williamson. The approximability of constraint satisfaction problems. *SIAM J. Comput.*, 30(6):1863–1920, 2000. doi:10.1137/S0097539799349948.

14 Subhash Khot. On the power of unique 2-prover 1-round games. In John H. Reif, editor, *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada*, pages 767–775. ACM, 2002. doi:10.1145/509907.510017.

15 Eun Jung Kim, Stefan Kratsch, Marcin Pilipczuk, and Magnus Wahlström. Flow-augmentation III: Complexity dichotomy for Boolean CSPs parameterized by the number of unsatisfied constraints. *SIAM J. Comput.*, 54(4):1065–1137, 2025. doi:10.1137/23M1553698.

16 Stefan Kratsch, Dániel Marx, and Magnus Wahlström. Parameterized complexity and kernelizability of max ones and exact ones problems. *ACM Trans. Comput. Theory*, 8(1):1:1–1:28, 2016. doi:10.1145/2858787.

17 Stefan Kratsch and Magnus Wahlström. Preprocessing of min ones problems: A dichotomy. In Samson Abramsky, Cyril Gavoille, Claude Kirchner, Friedhelm Meyer auf der Heide, and Paul G. Spirakis, editors, *Automata, Languages and Programming, 37th International Colloquium, ICALP 2010, Bordeaux, France, July 6-10, 2010, Proceedings, Part I*, volume 6198 of *Lecture Notes in Computer Science*, pages 653–665. Springer, 2010. doi:10.1007/978-3-642-14165-2_55.

18 Stefan Kratsch and Magnus Wahlström. Representative sets and irrelevant vertices: New tools for kernelization. *J. ACM*, 67(3):16:1–16:50, 2020. doi:10.1145/3390887.

19 Andrei Krokhin and Stanislav Zivny. *The Constraint Satisfaction Problem: Complexity and Approximability*, volume 7 of *Dagstuhl Follow-Ups*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 2017. doi:10.4230/DFU.Vol7.15301.

20 M. S. Ramanujan and Saket Saurabh. Linear-time parameterized algorithms via skew-symmetric multicuts. *ACM Trans. Algorithms*, 13(4):46:1–46:25, 2017. doi:10.1145/3128600.

21 Thomas J. Schaefer. The complexity of satisfiability problems. In Richard J. Lipton, Walter A. Burkhard, Walter J. Savitch, Emily P. Friedman, and Alfred V. Aho, editors, *Proceedings of the 10th Annual ACM Symposium on Theory of Computing, May 1-3, 1978, San Diego, California, USA*, pages 216–226. ACM, 1978. doi:10.1145/800133.804350.