Journal of Computer and System Sciences 159 (2026) 103780

Contents lists available at ScienceDirect

Journal of Computer and System Sciences

journal homepage: www.elsevier.com/locate/jcss

Algorithms and complexity of difference logic ☆

Konrad K. Dabrowski [a], Peter Jonsson [b], Sebastian Ordyniak [c,*], George Osipov [b]

[a] School of Computing, Newcastle University, UK
[b] Department of Computer and Information Science, Linköping University, Sweden
[c] School of Computing, University of Leeds, UK

ARTICLE INFO

Article history:
Received 7 December 2023
Received in revised form 16 October 2025
Accepted 19 January 2026
Available online 5 February 2026

Keywords:
Difference logic
Algorithms and complexity
Fine-grained complexity
Parameterized complexity
Treewidth

ABSTRACT

Difference Logic (DL) is a fragment of linear arithmetic where atoms are constraints $x+k \leq y$ for variables x, y (ranging over $\mathbb{Q}$ or $\mathbb{Z}$) and integer k. We study the complexity of deciding the truth of existential DL sentences. This problem appears in many contexts: examples include verification, bioinformatics, telecommunications, and spatio-temporal reasoning in AI. We begin by considering sentences in CNF with rational-valued variables. We restrict the allowed clauses via two natural parameters: *arity* and *coefficient bounds*. The problem is NP-hard for most choices of these parameters. As a response to this, we refine our understanding by analysing the time complexity and the parameterized complexity (with respect to well-studied parameters such as primal and incidence treewidth). We obtain a comprehensive picture of the complexity landscape in both cases. Finally, we generalise our results to integer domains and sentences that are not in CNF.

Contents

☆ This article improves and extends results from two conference papers [25,26].

* Corresponding author.
E-mail addresses: konrad.dabrowski@newcastle.ac.uk (K.K. Dabrowski), peter.jonsson@liu.se (P. Jonsson), sordyniak@gmail.com (S. Ordyniak), george.osipov@pm.me (G. Osipov).

https://doi.org/10.1016/j.jcss.2026.103780

1. Introduction

We have divided this introductory section into four parts. In the first one (Section 1.1), we present difference logic (DL) and some of its applications, and we describe our approach for studying the complexity of DL. In short, the satisfiability problem for DL is almost always NP-hard and a more fine-grained analysis becomes necessary; we will thus study the time complexity of DL together with its parameterized complexity under natural structural parameters. Our time complexity results are discussed in Section 1.2 while our parameterized results are discussed in Section 1.3. Finally, an outline of the article is given in Section 1.4.

1.1. Background

Difference Logic is a fragment of linear arithmetic where atoms are constraints of the form $x + k \leq y$ for variables x, y (with some numeric domain such as $\mathbb{Q}$ or $\mathbb{Z}$) and some integer k. The *satisfiability* problem for DL is the computational problem of deciding the truth of sentences

$$\exists x_1, \ldots, x_n . \phi$$

where ϕ is a quantifier-free formula over variable set $\{x_1, \ldots, x_n\}$. The satisfiability problem for conjunctions of difference atoms is solvable in polynomial time (by, for instance, the Floyd-Warshall algorithm), while adding various logical features often leads to computational hardness. We note, for instance, that the satisfiability problem is NP-hard (since the satisfiability problem for propositional logic is NP-hard [22]) and that the problem of deciding the truth of an arbitrary formula is PSPACE-hard (since deciding the truth of quantified propositional formulas is PSPACE-hard [76]). These complexity results hold both for rational and integer variable domains. DL is a well-studied formalism due to its many applications: the archetypal example is from verification where timed automata have natural connections with DL [3,58]. Other important applications include the channel assignment problem (which is a central problem in telecommunications [5,54]), unit interval problems (with applications in bioinformatics and graph theory [39,65]), and problems in connection with answer set programming [55,60]—all of these can be viewed as restricted variants of DL. Applications like these and the relative simplicity of DL have made it into one of the most ubiquitous theories in the context of *satisfiability modulo theories* (SMT) [7,20,61]. DL is also interesting from a complexity-theoretic point of view. One example is the *max-atom* problem (see the paper by Bezem et al. [9] or Section 6 in [15]) that can be viewed as a severely restricted version of DL. This problem is polynomial-time equivalent to problems such as mean pay-off games, scheduling under and-or precedence constraints, and finding solutions to certain classes of equations. The max-atom problem is intriguing since it is known to be in NP ∩ coNP but no polynomial-time algorithm has yet been identified.

DL is of major importance in AI but the connections are in general not clearly spelled out in the literature. Spatial-temporal reasoning is a fundamental task in AI and one of the most influential formalisms is the *simple temporal problem* (STP) that was first proposed in an AI context by Dechter et al. [28]. It is a constraint satisfaction problem (CSP) over a constraint language with relations

$$\{(x, y) \in \mathbb{Q}^2 : x - y \in [l, u]\}$$

where $\ell, u \in \mathbb{Q} \cup \{-\infty, +\infty\}$ and $[\ell, u]$ denotes a closed interval. We refer to constraints using such relations as *simple constraints*. The close relationship to DL is obvious. The STP formalism is often generalised so that the intervals may be half-closed, open, or a single point. Dechter et al. [28, Sec. 7] point out that this generalisation apparently do not have

Table 1
Summary of the most important notation concerning the disjunctive temporal problem (DTP) used in this article.

Notation	Meaning
$\mathbf{D}$	The constraint language for the disjunctive temporal problem (DTP).
CSP($\mathbf{A}$) for $\mathbf{A} \subseteq \mathbf{D}$	The constraint satisfaction problem using only relations in $\mathbf{A}$.
num($\mathbf{A}$) for $\mathbf{A} \subseteq \mathbf{D}$	The maximum absolute numerical value used by any numerical bound in $\mathbf{A}$.
$\mathbf{D}_{a,k}$ for $a, k \in \mathbb{N} \cup \{\infty\}$	$\mathbf{D}$ restricted to relations of arity a and numerical bounds up to k.

any adverse effects and, in particular, the resulting CSP is still solvable in polynomial time. Even though STPs have proven to be immensely useful in AI, their expressive power is limited. Thus, a common way of obtaining increased expressibility is to introduce disjunctions in various ways [6,28,63,75]. From the DL perspective, this is equivalent to considering DL formulas on conjunctive normal form and restricting the set of allowed clauses in various ways. The resulting formalisms are highly relevant in an AI context. Well-known examples can be found in automated planning [38,79] and multi-agent systems [10,18]. Stergiou & Koubarakis [75, Sec. 7], Tsamardinos & Pollack [78] and Peintner et al. [66] discuss various other applications, and Zavatteri et al. [82] have recently presented a large-scale evaluation of software for solving DTPs.

We traditionally view a computational problem as intractable if it is NP-hard. NP-hardness rules out polynomial-time algorithms (assuming P $\neq$ NP), but it does not say anything about the time complexity of the best possible algorithm. Recent advances in complexity theory allow us to prove conditional lower bounds via restricted reductions from complexity-theoretic conjectures that are stronger than the P $\neq$ NP conjecture. This methodology has enabled proving close-to-optimal bounds on time complexity for a multitude of problems assuming suitable conjectures, see also the textbook by Gaspers [37]. The goal of this article is to analyse the satisfiability problem for DL following this methodology. Our time complexity results reveal that many severely restricted variants of DL cannot be solved in a reasonable amount of time under the Exponential-Time Hypothesis (ETH). This computational hardness makes it worthwhile to use parameterized complexity for analysing DL with restricted interactions between variables and constraints.

We need some definitions and notation to facilitate the discussion of the problems that we will study. In the sequel, we restrict ourselves to the satisfiability problem for DL over rational numbers where the input is in CNF, and we study various ways of restricting the allowed clauses. We will return to DL without these restrictions in Section 7. The restriction to CNF formulas allows us to view the satisfiability problem for DL as a constraint satisfaction problem where the constraint language corresponds to the allowed clauses. Our clause restrictions will be based on two parameters: *arity* and *coefficient bounds*. The arity bounds the number of distinct variables that may appear in a clause. It is closely connected to the *length* of a clause, i.e. the maximum number of literals, since if a clause has length k, then its arity is at most $2k$. The coefficient bound simply equals the maximum over the absolute values of constants appearing in clauses.

We continue by introducing the maximally expressive constraint language $\mathbf{D}$. We consider intervals over $\mathbb{Q}$ with end-points in $\mathbb{Z} \cup \{-\infty, +\infty\}$. The intervals may be open, closed, half-closed, or a single point. Let $\mathbb{I}$ denote the set of these intervals and let $\mathbf{D}$ contain all relations

$$\{(x_1, \ldots, x_t) \in \mathbb{Q}^t : \bigvee_{\ell=1}^{m} x_{i_\ell} - x_{j_\ell} \in I_\ell\}$$

for arbitrary $t, m \geq 1$ where $i_\ell, j_\ell \in \{1, \ldots, t\}$ and $I_\ell \in \mathbb{I}$ for all $1 \leq \ell \leq m$. We remark that one may equally well use the reals instead of the rationals as the underlying domain. The CSP for $\mathbf{D}$ is known as the *disjunctive temporal problem* (DTP) in the AI literature. It is easy to verify that CSP($\mathbf{D}$) is in NP since the STP is solvable in polynomial time. Given a relation $R \in \mathbf{D}$, let $K(R)$ denote the set of numerical bounds appearing in R, e.g. for

$$R = \{(x, y, z) \in \mathbb{Q}^3 : (-\infty < x - y \leq 3) \vee (0 \leq x - z < 6)\}$$

we have $K(R) = \{3, 0, 6\}$. If X is a set of relations, then the definition of K extends naturally: $K(X) = \bigcup_{R \in X} K(R)$. Let $\mathbf{A} \subseteq \mathbf{D}$ and define $\text{num}(\mathbf{A}) = \max\{|a| : a \in K(\mathbf{A})\}$, i.e. num($\mathbf{A}$) is the least upper bound on absolute values of all numerical bounds appearing in the relations of $\mathbf{A}$. We let $\mathbf{D}_{a,k}$ (where $a, k \in \mathbb{N} \cup \{\infty\}$) denote the class of relations of arity at most a and with $\text{num}(\mathbf{D}_{a,k}) \leq k$. We refer to Table 1 for an overview of this notation.

We illustrate the basic definitions with an example: consider *Allen's interval algebra* [2] restricted so that the intervals are only allowed to have unit length. This formalism (which is referred to as the *unit Allen algebra)* has, for example, applications in bioinformatics and graph theory [39,65]. Given a closed interval I, we let I^- and I^+ denote the left and the right endpoint, respectively. We let $\mathbf{A}_{\text{ua}}$ denote a binary structure based on the following relations:

$I\{p\}J$	I precedes J	$I^+ < J^-$
$I\{m\}J$	I meets J	$I^+ = J^-$
$I\{o\}J$	I overlaps J	$I^- < J^-$ and $J^- < I^+$ and $I^+ < J^+$
$I\{e\}J$	I equals J	$I^- = J^-$ and $I^+ = J^+$

Table 2
Summary of computational complexity landscape for CSP($\mathbf{D}_{a,k}$).

	$k=0$	$1 \le k < \infty$	k unbounded
$a=2$	$\in$ P	NP-complete	NP-complete
$a \ge 3$	NP-complete	NP-complete	NP-complete

Note that relations p, m, o admit converses p^{-1}, m^{-1}, o^{-1} while the relation e is symmetric. We let the structure $\mathbf{A}_{\text{ua}}$ contain every disjunction of the basic relations. Formally, let $\mathbb{U}$ denote the set of all unit intervals on the real line. $\mathbf{A}_{\text{ua}}$ contains $\{(I,J) \in \mathbb{U}^2 : \bigvee_{r \in S} I\{r\}J\}$ for every $S \subseteq \{p, m, o, e, o^{-1}, m^{-1}, p^{-1}\}$. Observe that every basic relation in the unit Allen algebra can be expressed as a simple relation in $\mathbf{D}_{2,1}$ over the left endpoints of the intervals, i.e.

$$\begin{aligned} I\{p\}J &\iff I^- - J^- \in (-\infty, -1), \\ I\{m\}J &\iff I^- - J^- \in \{-1\}, \\ I\{o\}J &\iff I^- - J^- \in (-1, 0), \\ I\{e\}J &\iff I^- - J^- \in \{0\}, \end{aligned}$$

and similarly for the converse relations. Moreover, every simple relation in $\mathbf{D}_{2,1}$ can be expressed as a basic relation of the unit Allen algebra since the correspondence is one to one. This reasoning naturally extends to taking disjunctions of simple/basic relations. Thus, CSP($\mathbf{A}_{\text{ua}}$) and CSP($\mathbf{D}_{2,1}$) are the same computational problem, and any upper/lower bound that applies to one of the problems also applied to the other.

Let us now summarise the computational complexity of CSP($\mathbf{D}_{a,k}$). The polynomial-time solvability of CSP($\mathbf{D}_{2,0}$) follows from the fact that the relations in $\mathbf{D}_{2,0}$ equal the point algebra [80]. It is well known that CSP($\mathbf{D}_{k,0}$) for $k \ge 3$ is NP-hard (this follows, for instance, from an easy reduction from the Betweenness problem [36]). Finally, CSP($\mathbf{D}_{2,1}$) (and thus CSP($\mathbf{A}_{\text{ua}}$)) are NP-hard via a straightforward reduction from 3-Colourability; NP-hardness for CSP($\mathbf{D}_{2,k}$), $k > 1$, is a direct consequence. These results are presented in Table 2—we immediately see that there is a conspicuous lack of polynomial-time solvable cases. In the rest of this article, we will refine our understanding of the complexity of CSP($\mathbf{D}_{a,k}$) by first analysing its time complexity and continue with its parameterized complexity. We discuss these results in Sections 1.2 and 1.3, respectively.

1.2. Time complexity

We prove the following results concerning the time complexity of DTPs. Our lower bounds are based on the *Exponential-Time Hypothesis* (ETH) by Impagliazzo et al. [46], i.e. the 3-Satisfiability problem cannot be solved in $2^{o(n)}$ time, where n is the number of variables. We let $\mathbf{D}^{\le}_{a,k}$ denote the subset of $\mathbf{D}_{a,k}$ where the relations are defined by only using closed intervals.

1. CSP($\mathbf{D}$) is solvable in $2^{O(n(\log n + \log k))}$ time (Corollary 7).
2. CSP($\mathbf{D}_{2,k}$) is solvable in $2^{O(n \log\log n)}$ time (Theorem 13).
3. CSP($\mathbf{D}_{4,0}$) and CSP($\mathbf{D}^{\le}_{3,1}$) are not solvable in $2^{o(n \log n)}$ time (Theorems 15 and 16).
4. CSP($\mathbf{D}^{\le}_{2,\infty}$) is not solvable in $2^{o(n(\log n + \log k))}$ time (Theorem 17).
5. For every $c > 1$, there exist $k \ge 0$ and $\mathbf{A} \subseteq \mathbf{D}^{\le}_{2,k}$ such that CSP($\mathbf{A}$) cannot be solved in $O(c^n)$ time (Theorem 19).

We additionally use a result by Eriksson and Lagerkvist [31, Section 3].

Theorem 1 *([31]).* CSP($\mathbf{D}_{3,0}$) *is solvable in* $2^{O(n)}$ *but not in* $2^{o(n)}$ *time (if the ETH is true).*

The results are summarised in Table 3 and we see that the upper and lower bounds are reasonably close. The lower bounds hold for constraint languages that do not use strict inequalities except for CSP($\mathbf{D}_{a,0}$), $a \ge 2$; an instance of CSP($\mathbf{D}^{\le}_{a,0}$) is always satisfiable by assigning each variable value 0. The results concerning CSP($\mathbf{D}_{2,k}$) indicate that there is no uniform single-exponential algorithm for CSP($\mathbf{D}_{2,k}$). The result does not, however, rule out the possibility that CSP($\mathbf{D}_{2,k}$) can be solved in $2^{c_k \cdot n}$ time, where $c_1, c_2, \ldots$ is an increasing sequence. All results in Table 3 remain intact if we restrict the variables to take integer values only (see Section 7.2). We remark that our main goal is in delineating single-exponential vs super-exponential running times, for which the ETH is a reasonable starting point. To obtain more fine-grained lower bounds, e.g. rule out concrete constants in the bases of exponential functions, one typically needs to rely on stronger hypotheses like the *strong* ETH [19].

Our algorithm for CSP($\mathbf{D}$) is based on proving a *small-solution property*: every satisfiable instance of CSP($\mathbf{D}$) has a solution that assigns sufficiently small values to the variables. The small-solution property is not very common in infinite-domain CSPs but it is, for instance, known to hold for the max-atom problem [9] and the CSP problem for *unit two variables per inequality* relations [72]. Our proof utilises certain ordering properties inherent in $\mathbf{D}$ together with a method for handling

Table 3
Summary of time complexity landscape for CSP($\mathbf{D}_{a,k}$) under the ETH. Each case for a is split into two rows, with the upper row giving the upper bound and the lower row giving the lower bound. (∗) means that for every $c > 1$, there exists a $k \geq 0$ and $\mathbf{A} \subseteq \mathbf{D}_{2,k}$ such that CSP($\mathbf{A}$) cannot be solved in $O(c^n)$ time.

	$k = 0$	$1 \leq k < \infty$	k unbounded
$a = 2$	P	$2^{O(n \log\log n)}$ (Theorem 13)	$2^{O(n(\log n + \log k))}$ (Corollary 7)
	–	(∗) (Theorem 19)	$2^{o(n(\log n + \log k))}$ (Theorem 17)
$a = 3$	$2^{O(n)}$ (Theorem 1)	$2^{O(n \log n)}$ (Corollary 7)	$2^{O(n(\log n + \log k))}$ (Corollary 7)
	$2^{o(n)}$ (Theorem 1)	$2^{o(n \log n)}$ (Theorem 16)	$2^{o(n(\log n + \log k))}$ (Theorem 17)
$4 \leq a < \infty$	$2^{O(n \log n)}$ (Corollary 7)	$2^{O(n \log n)}$ (Corollary 7)	$2^{O(n(\log n + \log k))}$ (Corollary 7)
	$2^{o(n \log n)}$ (Theorem 15)	$2^{o(n \log n)}$ (Theorem 15)	$2^{o(n(\log n + \log k))}$ (Theorem 17)
a unbounded	$2^{O(n \log n)}$ (Corollary 7)	$2^{O(n \log n)}$ (Corollary 7)	$2^{O(n(\log n + \log k))}$ (Corollary 7)
	$2^{o(n \log n)}$ (Theorem 15)	$2^{o(n \log n)}$ (Theorem 15)	$2^{o(n(\log n + \log k))}$ (Theorem 17)

the integer and fractional part of the variables independently; this approach is distinctly different compared to the proof techniques used in [9] and [72]. With the aid of this result, we can enumerate a suitable collection of assignments and check whether at least one of them satisfies all constraints in the instance. The small-solution property will be important once again when we consider the parameterized setting (see Section 1.3). Our algorithm for CSP($\mathbf{D}_{2,k}$) is based on a non-trivial divide-and-conquer approach. The relations in $\mathbf{D}_{2,k}$ exhibit even stronger ordering properties than the relations in $\mathbf{D}$ and this allows us to show that any solution for an instance $\mathcal{I}$ of CSP($\mathbf{D}_{2,k}$) suggests a natural split of the whole instance into either two or three subinstances sharing only a small number of variables. Hence, our algorithm enumerates all possible decompositions into two or three subinstances with small variable overlap and recurses on these for every possible assignment of the shared variables. An immediate consequence of this algorithm is the following result (since CSP($\mathbf{A}_{\text{ua}}$) and CSP($\mathbf{D}_{2,1}$) are the same computational problem).

Proposition 2. CSP($\mathbf{A}_{\text{ua}}$) *is solvable in* $2^{O(n \log\log n)}$ *time.*

Our lower bounds are based on a mixture of related ideas. We exploit the lower bound on the $(k \times k)$-Independent Set problem by Lokshtanov, Marx and Saurabh [56], and the lower bound by Traxler [77]. The latter result concerns binary CSPs over finite domains, where the complexity is measured with respect to the number of variables. Intuitively, Traxler shows that, under the ETH, the complexity of binary CSPs grows together with the domain size. We illustrate the main technical idea by an example. Suppose an instance of a CSP over the domain $\{1, 2, 3\}$ has two variables v_1, v_2 and two unary constraints: $v_1 \in \{1, 2\}$ and $v_2 \in \{2, 3\}$. One can reduce it to a CSP over the domain $\{1, 2, 3\}^2$ with a single constraint $v \in \{1, 2\} \times \{2, 3\}$, where $\{1, 2\} \times \{2, 3\} = \{(1, 2), (1, 3), (2, 2), (2, 3)\}$. Here variable v encodes the pair of variables (v_1, v_2). Applying the same idea, one can reduce any instance of binary CSP over domain d with n variables to a binary CSP over domain d^r with roughly n/r variables, for any constant r. Thus, with increased domain size, the number of variables required to encode the same set of constraints decreases. Lokshtanov et al. push this idea to the limit, where the domain size and the number of variables are roughly equal. A helpful technical tool that we use in adapting these results to CSP($\mathbf{D}_{a,k}$) are *Sidon sets*. A set S of natural numbers is called a Sidon set if all pairwise sums of its elements are distinct, i.e. the equation $a + b = c + d$ with $a, b, c, d \in S$ is only solvable when $\{a, b\} = \{c, d\}$. Sidon sets are also used in our lower bound proofs in the parameterized case.

We conclude this section with a few words about related problems from the literature. The CSP($\mathbf{D}$) problem can be expressed in the *existential theory of the reals* ($\exists\mathbb{R}$). An $\exists\mathbb{R}$-formula is a Boolean combination of atomic predicates of the form $p(x_1, \ldots, x_n) \odot 0$, where p is a real polynomial and $\odot \in \{<, \leq, =, \geq, >, \neq\}$. Renegar's algorithm [68] decides the satisfiability problem for $\exists\mathbb{R}$-formulas in $L \log L \log\log L \cdot (md)^{O(n)}$ time where L is the number of bits needed to represent the coefficients in the polynomials, m is the number of polynomials in the sentence, d is maximum among total degrees of the polynomials, and n is the number of variables. Observe that instances of CSP($\mathbf{D}$) can be written as $\exists\mathbb{R}$-formulas by replacing atomic formulas of the form $x - y \leq a$ with $p(x, y) \leq 0$ where $p(x, y) = x - y - a$. An instance $\mathcal{I}$ of CSP($\mathbf{D}$) with n variables and $k = \text{num}(\mathcal{I})$ can have $O(n^2 k)$ atomic formulas: there are $\binom{n}{2}$ pairs of variables and $O(k)$ possible bounds can be expressed on their difference. We are allowed to use disjunctions, which can be applied to an arbitrary subset of the $O(n^2 k)$ atomic formulas. Thus, cast as a $\exists\mathbb{R}$-formula, I has $m \leq 2^{O(n^2 k)}$ polynomials of degree $d = 1$. This leads to a $2^{O(n^3 k)}$-time algorithm for CSP($\mathbf{D}$) and, consequently, $2^{O(n^3)}$ time for CSP($\mathbf{D}_{\infty,k}$). For binary constraint languages, Renegar's algorithm yields better results with $2^{O(n(\log n + \log k))}$ time for CSP($\mathbf{D}_{2,\infty}$) and $2^{O(n \log n)}$ time for CSP($\mathbf{D}_{2,k}$) because only $O(n^2 k)$ disjunctive formulas are available. In fact, the running time for CSP($\mathbf{D}_{2,\infty}$) obtained this way matches our result. However, we claim that our algorithm represents a very simple and natural approach to solving this problem. While asymptotically the result is the same, Renegar's algorithm solves a much more general problem, and the hidden constants in its running time are astronomical (see e.g. the practical evaluation in [44]).

Let us turn our attention to lower bounds. Socała [74] shows that the *channel assignment* problem cannot be solved in $2^{o(n \log n)}$ time under the ETH. This problem can be viewed as CSP($\mathbf{A}_{\text{ca}}$) where $\mathbf{A}_{\text{ca}}$ contains the relation $\{(x, y) \in \mathbb{N}^2 : |x - y| \geq$

$a\}$ for every $a \in \mathbb{N}$. This result implies that CSP($\mathbf{D}^{\leq}_{2,\infty}$) is not solvable in $2^{o(n \log n)}$ time but it does not directly imply our stronger $2^{o(n(\log n+\log k))}$ lower bound for integer solutions (that is derived by combining Theorem 17 and Lemma 37). *Unit two variables per inequality* (UTVPI) relations are defined as $\{(x, y) \in \mathbb{Z}^2 : ax + by \geq c\}$ where $a, b \in \{-1, 0, 1\}$ and $c \in \mathbb{Z}$. This is a well-studied and interesting generalisation of CSP($\mathbf{D}^{\leq}_{2,\infty}$) over the integers; Schutt and Stuckey write the following [71, p. 514].

> Unit two-variable-per-inequality (UTVPI) constraints form one of the largest classes of integer constraints which are polynomial time solvable (unless P = NP). There is considerable interest in their use for constraint solving, abstract interpretation, spatial databases, and theorem proving.

Seshia et al. [72] have presented an algorithm for checking the satisfiability of first-order formulas without universal quantification over UTVPI constraints. This algorithm runs in $2^{O(n(\log n+\log k))}$ time (where, as usual, n is the number of variables and k is the coefficient bound). Our lower bound result for CSP($\mathbf{D}^{\leq}_{2,\infty}$) over the integers shows that this algorithm is essentially optimal with respect to running time.

1.3. Parameterized complexity

We have seen that CSP($\mathbf{D}$) (and many severely restricted variants) cannot be solved in single-exponential time under the Exponential-Time Hypothesis (ETH). This motivates the search for efficiently solvable subproblems. To this end, we use the framework of *parameterized complexity* [29,32,59], where the run-time of an algorithm is studied with respect to a parameter $p \in \mathbb{N}$ and the input size n. The idea is that the parameter describes the structure of the instance in a computationally meaningful way. Here, the most favourable complexity class is FPT, which contains all problems that are *fixed-parameter tractable (fpt)*, i.e. can be decided in $f(p) \cdot n^{O(1)}$ time, where f is a computable function. The next best option is the complexity class XP, which contains all problems decidable in $n^{f(p)}$ time, i.e. the problems solvable in polynomial time when the parameter p is bounded. Clearly, FPT $\subseteq$ XP and this inclusion is strict (see e.g. [32, Cor. 2.26]). It is significantly better if a problem is in FPT than in XP since the order of the polynomial factor in the former case does not depend on the parameter p. Finally, the class pNP contains all problems that can be decided in $f(p) \cdot n^{O(1)}$ time by a non-deterministic algorithm for some computable function f. It is known that a problem is pNP-hard (under *fpt-reductions*; see Section 6) if it is NP-hard for some constant value of the parameter. Problems that are pNP-hard are considered to be significantly harder than those in XP since a problem that is pNP-hard cannot be in XP unless P = NP.

A prominent method for identifying tractable fragments of CSPs is to restrict variable-constraint interactions (see, for instance, the survey by Carbonnel and Cooper [21, Sec. 5]); these are referred to as *structural restrictions* and are commonly studied via the primal and incidence graphs associated with instances of the CSP. The *primal graph* has the variables as its vertices with any two joined by an edge if they occur together in a constraint. The *incidence graph* is the bipartite graph with two disjoint sets of vertices corresponding to the variables and the constraints, respectively. A constraint vertex and a variable vertex are joined by an edge if the variable occurs in the scope of the constraint. The *treewidth* of such graphs has been used extensively. It is, for example, known that the finite-domain CSP is in FPT with the parameter $w + d$ if w is the primal treewidth and d is the domain size [41], while this is not true (under standard complexity assumptions) if w is the incidence treewidth [70].

We now describe our parameterized results. It is known that the primal treewidth is bounded from below by the incidence treewidth [52] for arbitrary CSP instances. This means, in particular, that any parameterized tractability result obtained when parameterizing by incidence treewidth implies an equivalent tractability result for primal treewidth. Moreover, any parameterized hardness result for primal treewidth implies an equivalent parameterized hardness result for incidence treewidth. Thus, we present algorithms for CSP($\mathbf{D}_{a,k}$) parameterized by incidence treewidth and lower bounds with respect to primal treewidth. We exhibit an XP-algorithm for CSP($\mathbf{D}_{\infty,k}$) when $k \in \mathbb{N}$. This is a bottom-up dynamic programming algorithm along a nice tree decomposition of the incidence graph that exploits the fact that CSP($\mathbf{D}$) has the small-solution property. The algorithm runs in time $(nk)^{O(w)}$ where w is the treewidth of the incidence graph. One may note that CSP($\mathbf{D}$) is in XP whenever the numeric values occurring in the instance are bounded by a polynomial in the number of variables.

We complement this algorithmic result by proving that CSP($\mathbf{D}_{2,k}$) for $1 \leq k < \infty$ is W[1]-hard when parameterized by primal treewidth and thus not in FPT under standard complexity-theoretic assumptions. This shows that significantly faster algorithms for CSP($\mathbf{D}_{a,k}$) with $k < \infty$ are unlikely. This W[1]-hardness result carries over to CSP($\mathbf{D}^{\leq}_{a,k}$) when $k \geq 1$ almost without extra effort; note that the condition on k is required since CSP($\mathbf{D}^{\leq}_{a,0}$) is trivially in P. The reduction is from a novel multi-dimensional variant of the well-known Subset Sum problem, which we show to be W[1]-hard. Many important problems from the AI literature such as Allen's Algebra and RCC8 are in FPT with parameter primal treewidth [27] so even CSP($\mathbf{D}_{2,1}$) is a substantially harder problem. We finally show that CSP($\mathbf{D}_{2,\infty}$) is pNP-hard, i.e. the problem becomes much harder when the numeric values are unbounded. If a language L is in NP, then all parameterized languages $L' \subseteq L \times \mathbb{N}$ are members of pNP so CSP($\mathbf{D}_{2,\infty}$) is a pNP-complete problem. We summarise our results in Table 4. All results for $k \geq 1$ can be found in this article, while the result for $k = 0$ was proven by Dabrowski et al. [27]. We note that the results still hold if we restrict ourselves to integer variable domains (see Corollary 34 and Section 7.2). The results outlined above immediately imply the following since CSP($\mathbf{A}_{\text{ua}}$) and CSP($\mathbf{D}_{2,1}$) are the same computational problem.

Table 4
Summary of the parameterized complexity landscape for CSP($\mathbf{D}_{a,k}$) parameterized by primal treewidth and incidence treewidth. The parameter for upper bounds is incidence treewidth and for lower bounds is primal treewidth. Each case for a is split into two: the top row is for the upper bound and the bottom row for the lower bound. The only open case is for CSP($\mathbf{D}_{\infty,0}$), which is known to be in FPT parameterized by primal treewidth under additional restrictions [27, Section 3.4], but it is open what holds for incidence treewidth.

	$k=0$	$1 \le k < \infty$	k unbounded
$a=2$	P	XP (Theorem 22)	
		W[1]-h. (Theorem 33)	pNP-h. (Theorem 31)
$3 \le a < \infty$	FPT ([27, Theorem 6])	XP (Theorem 22)	
		W[1]-h. (Theorem 33)	pNP-h. (Theorem 31)
a unbounded	XP (Theorem 22)	XP (Th. 22)	
	?	W[1]-h. (Theorem 33)	pNP-h. (Theorem 31)

Proposition 3. CSP($\mathbf{A}_{\text{ua}}$) *with parameter treewidth of incidence graph is in XP and it is W[1]-hard with parameter treewidth of primal graph.*

We conclude this section by discussing some related algorithms from the literature. Bodirsky & Dalmau [13] and Huang et al. [45] proved that CSP($\mathbf{A}$) is in XP (with treewidth of the primal graph as parameter) for ω-categorical $\mathbf{A}$ and binary constraint languages $\mathbf{A}$ that have the *atomic network amalgamation property* (aNAP), respectively. Huang et al. write that their algorithm is fixed-parameter tractable, but this is due to non-standard terminology; according to their Theorem 6, the algorithm runs in $O(w^3 n \cdot e^{w^2 \log n}) = n^{O(w^2)}$ time. These two general results apply to many interesting problems: ω-categoricity is a fundamental property in the study of infinite-domain CSPs and many AI-relevant CSPs have this property (see also the book by Bodirsky [12]). Similarly, the aNAP and other amalgamation properties are highly important in this context, too. However, these properties do not hold for the constraint language $\mathbf{D}$ or even the fragment $\mathbf{D}_{2,1}$, as we will show next.

The theorem by Engeler, Ryll-Nardzewski, and Svenonius (see e.g. [43, Theorem 6.3.1]) implies that if $\mathbf{A}$ is an ω-categorical constraint language, then for all $n > 1$, there are finitely many nonequivalent formulas over $\mathbf{A}$ with n free variables. This is not true for $\mathbf{D}_{2,1}$: consider the infinite sequence of formulas $\phi_2(x,y), \phi_3(x,y), \ldots$ defined as follows:

$$\phi_k(x,y) \equiv \exists z_1, \ldots, z_k.\, x = z_1 \wedge y = z_k \wedge \bigwedge_{i=1}^{k-1} z_{i+1} - z_i = 1$$

and note that $\phi_k(x,y)$ holds if and only if $y = x + k - 1$. If a structure $\mathbf{A}$ containing binary relations has aNAP, then for any pair of complete atomic instances (V_1, C_1) and (V_2, C_2) of CSP($\mathbf{A}$) that have the same constraints over the variables in $V_1 \cap V_2$, their union $(V_1 \cup V_2, C_1 \cup C_2)$ is satisfiable. An instance of CSP($\mathbf{A}$) is *complete* if there is one constraint for every pair of variables, and it is *atomic* if no constraints involve disjunctions. Consider the instances

$$\mathcal{I}_1 = (\{x, a, y\}, \{a - x = 1, y - a = 1, y - x \in (1, \infty)\}),$$
$$\mathcal{I}_2 = (\{x, b, y\}, \{b - x = 1, y - b \in (0, 1), y - x \in (1, \infty)\}).$$

$\mathcal{I}_1$ and $\mathcal{I}_2$ are complete, satisfiable, atomic instances of CSP($\mathbf{D}_{2,1}$), and they agree on their intersection. However, their union is not satisfiable, since $\mathcal{I}_1$ implies that $y - x = 2$, while $\mathcal{I}_2$ implies that $y - x \in (1, 2)$.

Dabrowski et al. [27] have presented a fixed-parameter tractable algorithm for constraint languages having the *patchwork* property [57]; this is yet another amalgamation property. The applicability of this algorithm can naturally be ruled out with the aid of the hardness results presented in Section 6. It is also straightforward to verify directly that the problems we study do not have the patchwork property: in fact, the example above for ruling out that $\mathbf{D}_{2,1}$ has aNAP also shows that $\mathbf{D}_{2,1}$ does not have the patchwork property.

1.4. Outline

This article is based on two conference papers [25,26]. The major differences are that (1) this article generalises our earlier results on various temporal formalisms to difference logic, (2) it gives a comprehensive picture of the time complexity landscape, (3) the proofs are both unified and significantly simplified by the addition of multi-purpose results such as Theorem 6, and (4) the results are extended to both general formulas and variables with integer domains. A succinct summary of our main contributions can be found in Table 5. The article has the following structure. We present the necessary preliminaries in Section 2. The upper and lower bounds on time complexity are collected in Sections 3 and 4, respectively, while the parameterized upper and lower bounds are collected in Sections 5 and 6, respectively. We look at two generalisations of our results in Section 7: formulas that are not in conjunctive normal form are considered in Section 7.1 and problems where variables have integer domains in Section 7.2. We conclude the article in Section 8 with a discussion of our results.

Table 5
Simplified guide to our main contributions, where n is the number of variables, k is the coefficient bound and w is the treewidth of the incidence graph.

Result	Stated in
CSP($\mathbf{D}$) is solvable in $2^{O(n(\log n+\log k))}$ time.	Corollary 7
CSP($\mathbf{D}_{2,k}$) is solvable in $2^{O(n\log\log n)}$ time.	Theorem 13
CSP($\mathbf{D}_{4,0}$) is ETH-hard to solve in $2^{o(n\log n)}$ time.	Theorem 15
CSP($\mathbf{D}_{3,1}$) is ETH-hard to solve in $2^{o(n\log n)}$ time.	Theorem 16
CSP($\mathbf{D}_{2,\infty}$) is ETH-hard to solve in $2^{o(n(\log n+\log k))}$ time.	Theorem 17
CSP($\mathbf{D}_{\infty,k}$) is solvable in $(nk)^{O(w)}$ time.	Theorem 22
CSP($\mathbf{D}_{2,\infty}$) is pNP-hard parameterized by primal graph treewidth.	Theorem 31
CSP($\mathbf{D}_{2,1}$) is W[1]-hard parameterized by primal graph treewidth.	Theorem 33

2. Preliminaries

In this section we provide some prerequisites. We present the basic language of difference logic in Section 2.1 and give a compact overview of the constraint satisfaction problem in Section 2.2. Finally, Section 2.3 contains a primer on Sidon sets that we use as a tool for proving our lower bound results.

2.1. Difference logic

We begin with some basic logical terminology. A *(relational) signature* τ is a set of symbols, each with an associated natural number called their *arity*. A *(relational)* τ*-structure* $\mathbf{A}$ consists of a set D (the domain), together with relations $R^{\mathbf{A}} \subseteq D^k$ for each k-ary symbol $R \in \tau$. To avoid overly complex notation, we sometimes do not distinguish between the symbol R for a relation and the relation $R^{\mathbf{A}}$ itself. We also allow ourselves to view relational structures as sets and, for instance, write expressions like $R \in \mathbf{A}$. Let $\mathbf{A}$ be a τ-structure over a domain D. We say that $\mathbf{A}$ has arity a if every relation in $\mathbf{A}$ has arity at most a.

Let $\mathbf{A}$ be a τ-structure. First-order formulas ϕ over $\mathbf{A}$ (or, for short, $\mathbf{A}$-formulas) are defined using the logical symbols of universal and existential quantification, disjunction, conjunction, negation, equality, bracketing, variable symbols, the relation symbols from τ, and the symbol $\perp$ for the truth-value false. First-order formulas over $\mathbf{A}$ can be used to define relations: for a formula $\phi(x_1,\dots,x_k)$ with free variables $x_1,\dots,x_k$, the corresponding relation R is the set of all k-tuples $(t_1,\dots,t_k) \in D^k$ such that $\phi(t_1,\dots,t_k)$ is true in $\mathbf{A}$. In this case we say that R is *first-order definable* in $\mathbf{A}$. Our definitions of relations are always parameter-free, i.e. we do not allow the use of domain elements within them.

Certain types of first-order formulas are particularly interesting for our purposes. Let ϕ denote a first-order formula.

- ϕ is a *sentence* if it has no free variables.
- ϕ is in *conjunctive normal form* (CNF), if it is a conjunction of disjunctions of *literals*, i.e. atomic formulas or their negations. A disjunction of literals is called a *clause*.
- ϕ is *quantifier-free* if it does not contain existential and/or universal quantifiers.
- ϕ is *existential* if $\phi = \exists x_1,\dots,x_n.\psi$ where ψ is quantifier-free.

We let $\mathbf{S}$ denote the relational structure representing the atomic DL formulas, i.e. the infinite set of relations

$$\{(x,y) \in \mathbb{Q}^2 : \ell \odot_1 x - y \odot_2 u\}$$

for any $\ell \in \mathbb{Z} \cup \{-\infty\}$, $u \in \mathbb{Z} \cup \{\infty\}$ and $\odot_1, \odot_2 \in \{<, \leq\}$. We will sometimes consider a restricted set $\mathbf{S}^{\leq}$ where $\odot_1 = \odot_2 = \leq$. The satisfiability problem for DL is the following problem.

DL-Sat
Input: An existential first-order sentence ϕ over $\mathbf{S}$.
Question: Is ϕ true?

Note that we make (without loss of generality) the sensible assumption that the bounding values are integers (see e.g. the article by Tsamardinos & Pollack [78]): real values cannot in general be written down with a finite number of bits, and rational numbers can be scaled in a suitable way. We use the rationals as the value domain (also without loss of generality): if there is a solution to an instance of CSP($\mathbf{D}$) over the reals, then there is also a solution over the rationals. While this is not of major importance in this article, the differences between $\mathbb{R}$ and $\mathbb{Q}$ sometimes cause confusion and/or technical problems. We refer the reader to the literature for a more thorough discussion of representational issues [14,50].

2.2. Constraint satisfaction

We continue by defining the *constraint satisfaction problem* (CSP). Let **A** denote a relational τ-structure defined on a set D of values. The *constraint satisfaction problem* over **A** (CSP(**A**)) is defined as follows:

CSP(**A**)	
Input:	A tuple (V, C), where V is a set of variables and C is a set of constraints of the form $R(v_1, \ldots, v_a)$, where a is the arity of R, $v_1, \ldots, v_a \in V$, and $R \in \mathbf{A}$.
Question:	Is there a function $f : V \to D$ such that $(f(v_1), \ldots, f(v_a)) \in R$ for every $R(v_1, \ldots, v_a) \in C$?

Observe that we do not require **A** to have finite signature or D to be a finite set. The structure **A** is sometimes referred to as a *constraint language*, while the function f is a *satisfying assignment* or simply a *solution*. If $c = R(x_1, \ldots, x_a)$ is a constraint, then the set $\{x_1, \ldots, x_a\}$ is the *scope* of c. We denote this set by scope(c). A basic example of a CSP is the STP problem: it is easy to verify that it equals CSP(**S**). Another example is the max-atoms problem: it is conveniently defined as a CSP with the infinite constraint language $\mathbf{A}_{\max}$ containing the relations $R_a = \{(x, y, z) \in \mathbb{Q}^3 : \max(x, y) + d \geq z\}$ for every $d \geq 0$. We note that R_d is quantifier-free definable in **S** since

$$\max(x, y) + d \geq z \Leftrightarrow (x + d \geq z) \vee (y + d \geq z).$$

One may view CSP(**A**) with $\mathbf{A} \subseteq \mathbf{D}$ as a restricted DL-Sat problem. An **A**-sentence is *primitive positive* if it is of the form

$$\exists x_1, \ldots, x_n . \psi_1 \wedge \cdots \wedge \psi_l$$

where $\psi_1, \ldots, \psi_l$ are atomic formulas over **A**, i.e. formulas (1) $R(y_1, \ldots, y_a)$ with $R \in \mathbf{A}$, (2) $y_i = y_j$, or (3) $\perp$. Thus, CSP(**A**) can be viewed as DL-Sat restricted to primitive positive **A**-formulas whenever the equality relation is in **A**. This assumption is harmless for the CSP problem: adding equality to the constraint language does not affect the complexity of the CSP up to log-space reductions (see Lemma 1.2.6 in [12]). This connection between CSP and DL-Sat will be exploited in Section 7.

To simplify the presentation, we sometimes use an alternative notation for a disjunctive constraint $\bigvee_{\ell=1}^{m} x_{i_\ell} - x_{j_\ell} \in I_\ell$ and write it as a set of simple constraints $\{x_{i_\ell} - x_{j_\ell} \in I_\ell : \ell \in \{1, \ldots, m\}\}$. Then, an assignment satisfies the disjunctive constraint whenever it satisfies at least one simple constraint in the corresponding set. This way of viewing disjunctions simplifies, for instance, the treatment of certificates in Section 3.2.1.

When considering CSPs with infinite constraint languages, it is important to specify how the relation symbols are represented in the input instances. In our case, it would (for instance) be sufficient to represent the relation symbol for a relation R by a quantifier-free CNF definition of R using atomic formulas of the form $x - y \odot c$ with $\odot \in \{<, \leq\}$, and coefficients $c \in \mathbb{Z}$ represented in binary. Such a representation has certain pleasant features: one may, for instance, check in polynomial time whether a given rational tuple (where the numerator and denominator are viewed as integers represented in binary) is a member of R or not. Note that there are no representational issues like these when considering finite constraint languages.

For an instance $\mathcal{I}$ of CSP(**A**), we write $\|\mathcal{I}\|$ for the number of bits required to represent $\mathcal{I}$. We primarily measure time complexity in terms of n (the number of variables). Historically, this has been the most common way of measuring time complexity: for instance, the vast majority of work concerning finite-domain CSPs concentrates on the number of variables. One reason for this is that an instance may be much larger than the number of variables. Consider an instance of the propositional SAT problem, i.e. a propositional logical formula in CNF. Such a formula may contain up to 2^{2n} distinct clauses if repeated literals are disallowed, so measuring in terms of the instance size may give far too optimistic figures. It is thus more informative to know that SAT can be solved in $O^*(2^n)$ time[1] instead of knowing that it is solvable in $O^*(2^{\|\mathcal{I}\|})$ time.

The various constraint languages that we will consider were defined in Section 1. We note that disjunctive temporal relations are sometimes defined in a more general way which allows for unary atomic relations $x \in I$ (as opposed to binary atomic relations $x - y \in I$). The standard trick for handling unary relations is to introduce a *zero variable* (see [6]). Solutions to CSP(**D**) have the following property: if $\varphi : V \to \mathbb{Q}$ satisfies an instance (V, C), then so does $\varphi'(v) = \varphi(v) + c$ where $c \in \mathbb{Q}$ is an arbitrary constant. Thus, we can pick an arbitrary variable in V and assume that its value is zero: such a variable is called a *zero variable*. We can now easily express unary constraints, e.g. the constraint $x - z \in (0, 2]$ is equivalent to $x \in (0, 2]$ if z is the zero variable. Adding a single zero variable does not affect the time complexity with more than a multiplicative factor.

2.3. Sidon sets

Our lower bound results presented in Sections 4 and 6 use *Sidon sets* [73]. The study of Sidon sets is an important topic in additive number theory and elsewhere; see e.g. the survey by O'Bryant [62] or the book by Halberstam and Roth [42].

[1] The $O^*(\cdot)$ notation hides polynomial factors.

The terminology used in the literature may appear confusing: they are known under several names such as *Golomb rulers*, *Sidon sequences*, and B_2-sets, and the term Sidon set has different meanings in number theory and functional analysis. A Sidon set S is a set of integers such that the sum of any pair of its elements is unique, i.e. if $a+b=c+d$ for $a,b,c,d \in S$, then $\{a,b\}=\{c,d\}$. It is easier to work with differences in our proofs so we use the following equivalent condition: for all $a,b,c,d \in S$ such that $a \neq b$ and $c \neq d$, $a-b=c-d$ holds if and only if $a=c$ and $b=d$. This indicates one way of using Sidon sets: they allow us (under certain conditions) to rewrite a disjunction $x \neq a \vee y \neq b$ (where x, y are variables and a, b integers) as a difference $x - y \neq c$ for some integer c.

The *order* of a Sidon set is the number of elements in it and the *length* is the difference between its maximal and minimal elements. For example, $\{0,1,4,6\}$ is a Sidon set of order 4 with length 6. We will use a particular way of constructing Sidon sets with length quadratic in their order.

Proposition 4 *([30]). Let $p \geq n$ be an odd prime. Then*

$$S_n = \left\{pa + (a^2 \bmod p) : a \in \{0,\dots,n-1\}\right\}$$

is a Sidon set.

We sometimes need to ensure that the length of a Sidon set is bounded by a polynomial in its order k. Indeed, Proposition 4 shows that there is a Sidon set containing k positive integers and whose largest element is at most $2p^2$, where p is the smallest prime number larger than or equal to k. This set can clearly be constructed in polynomial time. Together with Bertrand's postulate (see e.g. Chapter 2 in the book by Aigner and Ziegler [1]) which states that for every natural number n there is a prime number between n and $2n$, we see that a Sidon set of order k and length $8k^2$ can be generated in polynomial time.

3. Upper bounds on time complexity

This section contains two main results: a $2^{O(n(\log n+\log k))}$ time algorithm for CSP($\mathbf{D}$) (Section 3.1) and a $2^{O(n \log\log n)}$ time algorithm for CSP($\mathbf{D}_{2,k}$) when $k < \infty$ is fixed (Section 3.2). These results together with the lower bound results that are proved in Section 4 are summarised in Table 3.

To simplify some of the proofs in this section, we note that it is sufficient to concentrate on *unit* constraints, which are defined as follows. Let $\mathbf{T} \subseteq \mathbf{D}_{2,k}$ be the constraint language with relations

$$\{(x,y) \in \mathbb{Q}^2 : x-y \in \{i\}\},$$
$$\{(x,y) \in \mathbb{Q}^2 : x-y \in (i,i+1)\} \text{ and}$$
$$\{(x,y) \in \mathbb{Q}^2 : x-y \in (i,\infty)\}$$

for all $i \in \mathbb{Z}$. We refer to the relations in $\mathbf{T}$ as *unit* relations. Consider constraint $x-y \in (-1,0] \cup [1,\infty)$. An equivalent constraint can be enforced by a disjunction of unit constraints $x-y \in (-1,0) \vee x-y \in \{0\} \vee x-y \in \{1\} \vee x-y \in (1,\infty)$. In a similar manner, we can rewrite every disjunctive temporal constraint as a disjunction of unit constraints.

3.1. *Upper bound for* CSP($\mathbf{D}$)

We will prove a *small-solution property* for CSP($\mathbf{D}$). Small-solution properties are results that state that a solvable instance of CSP($\mathbf{A}$) has a solution that assign 'small' values to the variables. Exactly what is meant by 'small' varies in different contexts. A concrete example is provided by Bezem et al. [9] for the max-atoms problem that we encountered in Section 1.1: every satisfiable instance (V,C) of the max-atoms problem has a solution $f: V \to \{0,\dots,p\}$ where

$$p = \sum_{\max(x,y)+d \geq z \in C} |d|.$$

Another example (from Section 1.2) is UTVPI relations. These are defined as $\{(x,y) \in \mathbb{Z}^2 : ax+by \geq c\}$ where $a,b \in \{-1,0,1\}$ and $c \in \mathbb{Z}$. Seshia et al. [72] prove that every satisfiable instance (V,C) of the CSP over UTVPI relations has a solution in the interval $\{-|V|\cdot(k+1),\dots,|V|\cdot(k+1)\}$, where $k=\text{num}(C)$. This result implies that every satisfiable instance (V,C) of CSP($\mathbf{D}^{\leq}$) has a solution $\{-|V|\cdot(k+1),\dots,|V|\cdot(k+1)\}$ but it does not give a bound for CSP($\mathbf{D}$)—note, for instance, that CSP($\mathbf{D}$) is not guaranteed to have integer solutions, e.g. $\{x-y \in (0,1)\}$. Bezem et al.'s proof has a graph-theoretical flavour while Sesha et al.'s proof is based on a polyhedral approach. None of these methods appear to be directly applicable to CSP($\mathbf{D}$): Bezem et al.'s proof uses intrinsic properties of the max-atom problem while Sesha et al.'s approach is built around the fact that solutions must assign integers to the variables. Our proof strategy has more of an order-theoretic flavour. Define the set

$$CD(n,k) = \left\{ z + \frac{q}{n} : z, q \in \mathbb{N},\ 0 \le z \le (n-1)(k+1),\ \text{and } 0 \le q < n \right\}$$

for $n, k \in \mathbb{N}$. This set will serve as a ruler, and we will show that any satisfying assignment to an instance of CSP($\mathbf{D}$) with n variables and numerical bound k can be transformed into one that only chooses values from the ruler. To achieve this we will split the assignment of each variable into the integral part and the fractional part and show how to independently transform these parts. Our starting point is the following lemma which provides sufficient conditions for two assignments to satisfy the same set of simple constraints. This lemma will also be useful when proving the forthcoming Lemma 10.

We let $\lfloor x \rfloor$ denote the *floor* function (i.e. $\lfloor x \rfloor$ is the largest integer less than or equal to the real number x) and we let $\mathrm{frac}(x)$ denote the fractional part of the non-negative real number x (i.e. $\mathrm{frac}(x) = x - \lfloor x \rfloor$). We are now ready to prove the main technical lemma.

Lemma 5. *Let k be an integer and let $\phi_1 : V \to \mathbb{Q}$ and $\phi_2 : V \to \mathbb{Q}$ be two assignments of the variables in V that satisfy the following two conditions:*

1. *For every $x, y \in V$, it holds that $\phi_1(x) - \phi_1(y)$ and $\phi_2(x) - \phi_2(y)$ have the same integer part up to $k+1$, i.e.* $\min\{\lfloor \phi_1(x) - \phi_1(y) \rfloor, k+1\} = \min\{\lfloor \phi_2(x) - \phi_2(y) \rfloor, k+1\}$.
2. *For every $x, y \in V$, it holds that* $\mathrm{frac}(\phi_1(x)) \odot \mathrm{frac}(\phi_1(y))$ *if and only if* $\mathrm{frac}(\phi_2(x)) \odot \mathrm{frac}(\phi_2(y))$ *for every* $\odot \in \{<, =, >\}$.

Then, ϕ_1 and ϕ_2 satisfy the same simple constraints over V with relations in $\mathbf{D}_{2,k}$.

Proof. To show the lemma, it is sufficient to show that ϕ_1 and ϕ_2 satisfy the same unit constraints. Suppose that ϕ_1 and ϕ_2 satisfy conditions 1 and 2. We need to show that ϕ_1 satisfies any of the unit constraints on two variables x and y if and only if so does ϕ_2. We distinguish the following cases according to the three types of unit constraints given above.

- If $\phi_1(x) - \phi_1(y) \in \{i\}$ for some $i \le k$, then $\lfloor \phi_1(x) - \phi_1(y) \rfloor = i$ and $\mathrm{frac}(\phi_1(x)) = \mathrm{frac}(\phi_1(y))$. Therefore, $\lfloor \phi_2(x) - \phi_2(y) \rfloor = i$ and $\mathrm{frac}(\phi_2(x)) = \mathrm{frac}(\phi_2(y))$, which implies that $\phi_2(x) - \phi_2(y) \in \{i\}$.
- If $\phi_1(x) - \phi_1(y) \in (i, i+1)$ for some $i < k$, then $\lfloor \phi_1(x) - \phi_1(y) \rfloor = i$ and $\mathrm{frac}(\phi_1(x)) > \mathrm{frac}(\phi_1(y))$. Therefore, $\lfloor \phi_2(x) - \phi_2(y) \rfloor = i$ and $\mathrm{frac}(\phi_2(x)) > \mathrm{frac}(\phi_2(y))$, which implies that $\phi_2(x) - \phi_2(y) \in (i, i+1)$.
- If $\phi_1(x) - \phi_1(y) \in (i, \infty)$ for some $i \le k$, then either $\lfloor \phi_1(x) - \phi_1(y) \rfloor = i$ and $\mathrm{frac}(\phi_1(x)) > \mathrm{frac}(\phi_1(y))$ or $\lfloor \phi_1(x) - \phi_1(y) \rfloor > i$. In the former case, we have that $\lfloor \phi_2(x) - \phi_2(y) \rfloor = i$ and $\mathrm{frac}(\phi_2(x)) > \mathrm{frac}(\phi_2(y))$ and therefore $\phi_2(x) - \phi_2(y) \in (i, \infty)$. In the latter case, we have that $\lfloor \phi_2(x) - \phi_2(y) \rfloor > i$ and therefore $\phi_2(x) - \phi_2(y) \in (i, \infty)$.

This completes the proof. □

Lemma 5 enables us to give a clear-cut proof of the small-solution property.

Theorem 6. *Every satisfiable instance $\mathcal{I} = (V, C)$ of CSP($\mathbf{D}$) has a solution $f : V \to CD(|V|, \mathrm{num}(C))$.*

Proof. Let $\mathcal{I} = (V, C)$ be a satisfiable instance of CSP($\mathbf{D}$) with solution $g : V \to \mathbb{Q}$. Let $n = |V|$, $k = \mathrm{num}(C)$, our strategy is to take the assignment g and construct a new assignment $f : V \to CD(n, k)$ that satisfies the same simple constraints as g over V with relations in $\mathbf{D}_{2,k}$.

Index the variables $\{v_1, \ldots, v_n\}$ so that $g(v_i) \le g(v_{i+1})$ for all $1 \le i < n$. Then, split the values $g(v_i)$ into integral and fractional parts, i.e. define $z_i = \lfloor g(v_i) \rfloor$ and $q_i = \mathrm{frac}(g(v_i))$ for all i. Note that $0 \le q_i < 1$ and the integers $z_1, \ldots, z_n$ are in non-decreasing order.

We recursively define the assignment $f(v_i) = c_i + d_i$ for all i, where c_i is the integral part and d_i is the fractional part of $f(v_i)$. Set $c_1 = 0$ and let $c_{i+1} = c_i + \min\{z_{i+1} - z_i, k+1\}$ for all $1 \le i < n$. Note that $c_1, \ldots, c_n$ are sorted in non-decreasing order. Furthermore, let $\sigma : \{q_1, \ldots, q_n\} \to \{0, \ldots, n-1\}$ be an injective function such that $\sigma(q_i) \odot \sigma(q_j) \iff q_i \odot q_j$ for all i, j and $\odot \in \{<, =, >\}$. One may view σ as an order-preserving 'scaling' of the fractional parts into the integers. Let $d_i = \frac{\sigma(q_i)}{n}$ for all i. Note that $c_n \le (n-1)(k+1)$ and $0 \le \sigma(q_i) \le n-1$, so f maps the variables in V into the set $CD(n, k)$, as desired. Moreover, since f and g satisfy the conditions on ϕ_1 and ϕ_2 given in the statement of Lemma 5, we obtain that f and g satisfy the same simple constraints over V with relations in $\mathbf{D}_{2,k}$. Therefore, f also satisfies $\mathcal{I}$, as required. □

Theorem 6 gives us straightforward upper bounds on the time complexity of CSP($\mathbf{D}$) and many of its subclasses.

Corollary 7. *Every instance $\mathcal{I} = (V, C)$ of CSP($\mathbf{D}$) can be solved in $2^{O(n(\log n + \log k))}$ time where $n = |V|$ and $k = \mathrm{num}(C)$. In particular, every instance of CSP($\mathbf{D}_{\infty,k}$) can be solved in $2^{O(n \log n)}$ time.*

Proof. Enumerate all assignments $f : V \to CD(n, k)$ and check if they satisfy $\mathcal{I}$. Theorem 6 implies that $\mathcal{I}$ is satisfiable if and only if at least one such assignment is satisfying. This takes $O^*(|CD(n, k)|^n)$ time in total. The set $CD(n, k)$ contains $O(n^2 k)$ elements so

$$O^*(|CD(n,k)|^n) = O^*((n^2k)^n) = 2^{O(n\log n)} \cdot 2^{O(n\log k)} = 2^{O(n(\log n+\log k))}. \quad \square$$

3.2. *Upper bound for* CSP($\mathbf{D}_{2,k}$)

In this section we prove that CSP($\mathbf{D}_{2,k}$) can be solved in $2^{O(n\log\log n)}$ time. Our algorithm for CSP($\mathbf{D}_{2,k}$) is based on a divide-and-conquer approach, i.e. we split the instance into smaller parts and solve them recursively. To achieve the splitting, we first show that any solution for an instance $\mathcal{I}$ of CSP($\mathbf{D}_{2,k}$) suggests a natural split of the whole instance into either two or three almost independent subinstances; here, almost independent refers to the instances sharing only a small set of variables. This will be exploited by the algorithm to enumerate all possible decompositions into two or three subinstances and recurse on these for every possible assignment of the shared variables. We also need a subroutine that allows us to find all solutions with small domain values. This is used to solve the middle instance in the case that the instance decomposes into three subinstances.

Before we begin, we describe some polynomial time preprocessing steps for CSP($\mathbf{D}_2$). Suppose there are two constraints for a pair of variables $x - y$:

$$x - y \in I_1 \vee \cdots \vee x - y \in I_p,$$
$$x - y \in J_1 \vee \cdots \vee x - y \in J_q,$$

where $I_1, \ldots, I_p$ and $J_1, \ldots, J_q$ are intervals. For both constraints to hold, there must exist $1 \le i \le p$ and $1 \le j \le q$ such that $x - y \in I_i \cap J_j$. Thus, we can replace these two constraints with

$$\bigvee_{i=1}^{p}\bigvee_{j=1}^{q} x - y \in I_i \cap J_j.$$

Applying this procedure exhaustively, we obtain an instance with at most one constraint for every pair of variables. In the rest of the section we assume that all instances $\mathcal{I} = (V, C)$ are preprocessed, and we write $\sigma_C(x, y)$ to denote *the* constraint in C over variables x and y; if there are no constraints over x and y, we let $\sigma_C(x, y)$ be the set of all possible simple constraints over x and y.

The rest of this section is divided into three parts (Sections 3.2.1–3.2.3): the first two sections introduce certain subroutines that are needed in the algorithm, and the algorithm itself is presented and proven correct in the third section.

3.2.1. *Certificates*

We begin by presenting an alternative method for enumerating compact representations of solutions to CSP($\mathbf{D}$) instances in terms of certificates. The main advantage of certificates compared to representing solutions by assignments is that certificates allow us to express the partial solution in terms of constraints. In particular, it allows us to fix the behaviour of certain variables by simply adding additional constraints to the instance.

Let $\mathcal{I} = (V, C)$ be an instance of CSP($\mathbf{D}$). Define $\mathcal{U}(C)$ to be the set of simple temporal constraints appearing as disjuncts in C. For example, if $C = \{(x - y \le 0) \vee (x - y \ge 1), x - z \ge 1\}$, then $\mathcal{U}(C) = \{x - y \le 0, x - y \ge 1, x - z \ge 1\}$. Let $\varphi : V \to \mathbb{R}$ be an assignment to $\mathcal{I}$. We identify φ with the subset of constraints $F \subseteq \mathcal{U}(C)$ satisfied by φ. This allows us to define an equivalence relation $\sim$ on the assignments where $\varphi_1 \sim \varphi_2$ holds if and only if $F_1 = F_2$. This way, F represents the entire class of assignments equivalent to φ. We say that F is a *certificate* of the satisfiability of $\mathcal{I}$. An assignment φ is satisfying if the certificate F contains at least one simple constraint from every $c \in C$. Note that if $\varphi_1 \sim \varphi_2$ and φ_1 is a satisfying assignment, then so is φ_2. While there may be infinitely many satisfying assignments to $\mathcal{I}$, the number of certificates is finite since there are at most as many certificates as there are subsets of $\mathcal{U}(C)$.

Theorem 8. *The list of certificates to an instance $\mathcal{I} = (V, C)$ of* CSP($\mathbf{D}$) *can be computed in $2^{O(n(\log n+\log k))}$ time, where $n = |V|$ and $k =$* num(C).

Proof. By definition, every certificate $\mathcal{I}' = (V', C')$ for $\mathcal{I}$ is a satisfiable instance of CSP($\mathbf{S}$) with n variables and num(C') $\le k$. Theorem 6 implies that each $\mathcal{I}'$ admits a satisfying assignment in $CD(n,k)$. Thus, we can enumerate assignments $f : V(I) \to CD(n,k)$, check whether it satisfies I, and if so, collect the simple constraints in $\mathcal{U}(C)$ satisfied by f and output them as a certificate. This requires $|CD(n,k)|^n \le (n^2k)^n = 2^{O(n(\log n+\log k))}$ time. $\square$

The algorithm underlying the previous theorem will be used as a subroutine in our algorithm for CSP($\mathbf{D}_{2,k}$). We will refer to it as ListCert in what follows.

3.2.2. *Instances with bounded span*

We continue by examining a restricted version of CSP($\mathbf{D}_2$) (denoted w-CSP($\mathbf{D}_2$)) where solutions can only take values in the interval $[0, w)$; we say that such a solution has *span* w. We show that this problem can be solved in $O^*(w^n)$ time.

Lemma 9. *w-CSP($\mathbf{D}_2$) can be solved in $O^*(w^n)$ time.*

Proof. Let $\mathcal{I} = (V, C)$ be an instance of w-CSP($\mathbf{D}_2$) and assume $\varphi : V \to [0, w)$ is a satisfying assignment. Without loss of generality, assume that all constraints in C are represented as disjunctions of unit constraints. We split φ into integral and fractional parts: $\varphi(x) = \varphi_i(x) + \varphi_f(x)$, where $\varphi_i(x) \in \{0, \ldots, w-1\}$ and $0 \leq \varphi_f(x) < 1$. Suppose we fix φ_i and want to check whether any φ_f extends φ_i to a satisfying assignment. For every pair of distinct variables x and y we have $\varphi_i(x) - \varphi_i(y) = c$ for some integer c. There are only six nontrivial unit constraints that agree with this assignment, each of them expressible as a linear inequality or disequality:

$$
\begin{aligned}
x - y &\in (c-1, c) &&\longrightarrow && \varphi_f(x) < \varphi_f(y)\\
x - y &\in \{c\} &&\longrightarrow && \varphi_f(x) = \varphi_f(y)\\
x - y &\in (c, c+1) &&\longrightarrow && \varphi_f(x) > \varphi_f(y)\\
x - y &\in (c-1, c] &&\longrightarrow && \varphi_f(x) \leq \varphi_f(y),\\
x - y &\in [c, c+1) &&\longrightarrow && \varphi_f(x) \geq \varphi_f(y),\\
x - y &\in (c-1, c) \cup (c, c+1) &&\longrightarrow && \varphi_f(x) \neq \varphi_f(y).
\end{aligned}
$$

These constraints together with the domain restriction $0 \leq \varphi_f(v) < 1$ for each v yield a system of linear inequalities and disequalities that has a solution if and only if there is a fractional assignment φ_f that extends φ_i to a satisfying assignment. Feasibility of a system of linear inequalities and disequalities can be decided in polynomial time [47,53]. There are w^n possible functions $\varphi_i : V \to \{0, \ldots, w-1\}$ and checking whether an integer assignment φ_i can be extended to a satisfying assignment requires polynomial time. Hence, the total running time of this algorithm is $O^*(w^n)$. □

We refer to the algorithm underlying the previous lemma as SolveBounded.

3.2.3. Divide-and-conquer strategy

Our algorithm for CSP($\mathbf{D}_{2,k}$) is based on a divide-and-conquer approach: we split the instances into smaller parts and solve them recursively. To achieve the splitting, we first show that any satisfying assignment for an instance of CSP($\mathbf{D}_{2,k}$) provides a natural split of the instance into either two or three subinstances. These subinstances are almost independent in the sense that they only share a small number of variables. By enumerating suitable values for the shared variables, we can thus solve the original instance recursively. To show that every satisfying assignment allows one of the two splits, consider a satisfying assignment φ of an instance $\mathcal{I} = (V, C)$ of CSP($\mathbf{D}_{2,k}$). We can assume that the minimal value assigned by φ is zero by translational invariance and we additionally assume that $\varphi : V \to [0, kw)$ for some $w \in \mathbb{N}$. Next, we divide the domain of φ into intervals $[ki - k, ki)$ of span k for every $i \in \{1, \ldots, w\}$. This implies a partition of the variables in V into disjoint (possible empty) subsets $\{V_i : i \in \{0, \ldots, w+1\}\}$ defined as follows:

$$V_i = \{v \in V : \varphi(v) \in [ki - k, ki)\}$$

for all $i \in \{1, \ldots, w\}$, and $V_0 = V_{w+1} = \varnothing$. Also, define the sets $L_i = \bigcup_{j=0}^{i-1} V_j$ and $R_i = \bigcup_{j=i+1}^{w+1} V_j$ for all i. We start by showing that we can split the instance at any V_i to obtain two subinstances that are independent up to their overlap at V_i. That is, let $\mathcal{I}[U] = (U, C[U])$ be the subinstance of $\mathcal{I}$ induced by the variables in U, where $C[U]$ is the subset of C containing only the constraints with all variables in U. Then, the following lemma provides necessary and sufficient conditions that allows us to transform solutions for the subinstances $\mathcal{I}[L_i \cup V_i]$ and $\mathcal{I}[V_i \cup R_i]$ into a solution for the whole instance; informally the lemma allows us to split the instance at any V_i.

Lemma 10. *Let $\mathcal{I} = (V, C)$ be an instance of CSP($\mathbf{D}_{2,k}$), where V is equal to the disjoint union of the sets X, Y, and Z. Assume φ_1 and φ_2 are satisfying assignments to the subinstances $\mathcal{I}[X \cup Y]$ and $\mathcal{I}[Y \cup Z]$, respectively. If the following conditions hold, then $\mathcal{I}$ is satisfiable:*

1. *For every pair of variables $x \in X$ and $z \in Z$, the constraint $\sigma_C(x, z)$ is empty or it implies $z - x > k$.*
2. *Assignments φ_1 and φ_2 satisfy the same unit constraints over every pair of variables in Y.*
3. *There is $T_1 \in \mathbb{Q}$ such that $\varphi_1(x) < T_1 \leq \varphi_1(y) < T_1 + k$ for all $x \in X$ and $y \in Y$.*
4. *There is $T_2 \in \mathbb{Q}$ such that $T_2 \leq \varphi_2(y) < T_2 + k \leq \varphi_2(z)$ for all $y \in Y$ and $z \in Z$.*

Proof. First note that the ordering of the variables in Y with respect to φ_1 and φ_2 is the same since, otherwise, there are two variables y and y' that do not satisfy the same simple constraints under φ_1 and φ_2 and this contradicts condition 2. Rename the variables in Y so that $y_1, \ldots, y_{|Y|}$ is in non-decreasing order with respect to φ_1 and φ_2. It is convenient to assume, without loss of generality, that $\varphi_1(y_1) = \varphi_2(y_1) = 0$—we can arrive at such assignments by subtracting $\varphi_i(y_1)$ from $\varphi_i(y)$ for every $y \in Y$ and $i \in \{1, 2\}$. We show next that $\lfloor \varphi_1(y) \rfloor = \lfloor \varphi_2(y) \rfloor$ for every $y \in Y$. This clearly holds for y_1 so we arbitrarily pick another variable $y \in Y$. Now, it holds that $\varphi_i(y) - \varphi_i(y_1) = \varphi_i(y)$ for every $i \in \{1, 2\}$. We distinguish the following cases:

- $\varphi_1(y) - \varphi_1(y_1) = i$ for some $i \in \{0, \dots, k\}$,
- $\varphi_1(y) - \varphi_1(y_1) \in (i, i+1)$ for some $i \in \{0, \dots, k-1\}$, or
- $\varphi_1(y) - \varphi_1(y_1) \in (k, \infty)$.

Note that the third case cannot occur because of conditions 3 or 4. Moreover, in the first case we obtain from condition 2 that $\varphi_2(y) - \varphi_2(y_1) = i$, which implies that $\varphi_1(y) = \varphi_2(y) = i$. Finally, in the second case, we obtain from condition 2 that $\varphi_2(y) - \varphi_2(y_1) \in (i, i+1)$, which implies that $\lfloor \varphi_1(y) \rfloor = \lfloor \varphi_2(y) \rfloor$.

Now we show that $\text{frac}(\varphi_1(y)) \odot \text{frac}(\varphi_1(y'))$ if and only if $\text{frac}(\varphi_2(y)) \odot \text{frac}(\varphi_2(y'))$ for every $y, y' \in Y$ and $\odot \in \{<, =, >\}$. We distinguish the following cases. If $\text{frac}(\varphi_i(y)) = \text{frac}(\varphi_i(y'))$ for some $i \in \{1, 2\}$, then $\varphi_i(y) - \varphi_i(y')$ is an integer and conditions 3 and 4 imply that $\varphi_i(y) - \varphi_i(y') \in \{-k, \dots, k\}$. Therefore, we obtain from condition 2 that $\varphi_1(y) - \varphi_1(y') = \varphi_2(y) - \varphi_2(y')$ so $\varphi_2(y) - \varphi_2(y')$ is an integer and $\text{frac}(\varphi_2(y)) = \text{frac}(\varphi_2(y'))$, as required. Otherwise, suppose without loss of generality that $\text{frac}(\varphi_1(y)) < \text{frac}(\varphi_1(y'))$ and $\lfloor \varphi_1(y) \rfloor \le \lfloor \varphi_1(y') \rfloor$. Then, using condition 3, we obtain that $\varphi_1(y') - \varphi_1(y) \in (i, i+1)$ for some $i \in \{0, \dots, k-1\}$. By condition 2, it follows that $\varphi_2(y') - \varphi_2(y) \in (i, i+1)$, too. Since the integer parts of y and y' are the same for both φ_1 and φ_2, we see that $\text{frac}(\varphi_2(y)) < \text{frac}(\varphi_2(y'))$, as required.

We are now ready to define a common assignment φ for $\mathcal{I}$ as follows. We keep the integer part of every variable in $X \cup Y \cup Z$ the same as before, i.e.:

- for every $x \in X$, we set the integer part of $\varphi(x)$ equal to the integer part of $\varphi_1(x)$,
- for every $z \in Z$, we set the integer part of $\varphi(z)$ equal to the integer part of $\varphi_2(z)$, and
- for every $y \in Y$, we set the integer part of $\varphi(y)$ equal to the integer part of $\varphi_1(y)$ and $\varphi_2(y)$; using the property that the integer parts of $\varphi_1(y)$ and $\varphi_2(y)$ are equal.

To define the fractional part of $\varphi(v)$ for every $v \in X \cup Y \cup Z$, let ρ_1 (ρ_2) be the ordered partition[2] of the variables in $X \cup Y$ ($Y \cup Z$) ordered by non-decreasing fractional part with respect to φ_1 (φ_2). Then, ρ_1 and ρ_2 are equal if restricted to the variables in Y and therefore we can combine them into an ordered partition ρ of the variables in $X \cup Y \cup Z$. Moreover, from ρ we can obtain a function $\sigma : X \cup Y \cup Z \to \{0, \dots, |X \cup Y \cup Z| - 1\}$ such that for every $\odot \in \{<, =, >\}$ it holds that:

- $\sigma(u) \odot \sigma(v)$ if and only if $\text{frac}(\varphi_1(u)) \odot \text{frac}(\varphi_1(v))$ for every $u, v \in X \cup Y$ and
- $\sigma(u) \odot \sigma(v)$ if and only if $\text{frac}(\varphi_2(u)) \odot \text{frac}(\varphi_2(v))$ for every $u, v \in Y \cup Z$.

We now obtain the fractional part of $\varphi(v)$ by setting $\varphi(v) = \sigma(v)/n$, which implies that φ satisfies:

- $\varphi(u) \odot \varphi(v)$ if and only if $\text{frac}(\varphi_1(u)) \odot \text{frac}(\varphi_1(v))$ for every $u, v \in X \cup Y$ and
- $\varphi(u) \odot \varphi(v)$ if and only if $\text{frac}(\varphi_2(u)) \odot \text{frac}(\varphi_2(v))$ for every $u, v \in Y \cup Z$.

It now follows from Lemma 5 (applied to φ and φ_1 as well as φ and φ_2) that φ satisfies $\mathcal{I}[X \cup Y]$ and $\mathcal{I}[Y \cup Z]$. Moreover, because of condition 3 and 4, it holds that $\varphi(z) - \varphi(x) > k$ for every $x \in X$ and $z \in Z$, which together with condition 1 implies that φ also satisfies $\mathcal{I}[X \cup Z]$. Therefore, φ satisfies $\mathcal{I}$, as required. □

The above lemma shows that the instance can be split at any V_i into two subinstances $\mathcal{I}[L_i \cup V_i]$ and $\mathcal{I}[V_i \cup R_i]$ that are independent once we fix a certificate for the instance $\mathcal{I}[V_i]$. This also means that a split at V_i will only be useful if the number of possible certificates of $\mathcal{I}[V_i]$, or equivalently the number of variables in V_i, is not too large. That is, if V_i contains at most $n/\log n$ variables, then we say it is *sparse* and, otherwise, we say that it is *dense*. The choice of $n/\log n$ as the threshold value is justified by the following proposition:

Proposition 11. *The number of certificates for any instance $\mathcal{I} = (V, C)$ of CSP(**D**) with at most $N \le n/\log(n)$ variables can be computed in time $2^{O(n)}$.*

Proof. This follows immediately from Theorem 8 after observing that $2^{O(N(\log N + \log k))} = 2^{O(n)}$, where $k = \text{num}(C)$ is constant. □

The next lemma is crucial for our algorithm since it allows us to show that any solution naturally splits the instance into either two or three almost independent subinstances; we will later see how the subinstances are obtained from the partitions identified by the lemma. Informally, case 1 in the lemma corresponds to a three-split and gives rise to the two

[2] An *ordered partition* of a finite set S is a sequence of non-empty disjoint subsets $(S_1, \dots, S_\ell)$ such that $\bigcup_{i=1}^{\ell} S_i = S$.

instances $\mathcal{I}[L_i \cup V_i]$ and $\mathcal{I}[V_i \cup R_i]$ and case 2 in the lemma corresponds to a five-split and gives rise to the three instances $\mathcal{I}[L_i \cup V_i]$, $\mathcal{I}[V_i \cup V_{i+1} \cup \cdots \cup V_{j-1} \cup V_j]$ (where $i \le j$), and $\mathcal{I}[V_j \cup R_j]$.

Lemma 12. *Let $\mathcal{I} = (V, C)$ be an instance of* CSP($\mathbf{D}_{2,k}$)*, let $\varphi : V \to [0, kw)$ be an assignment of $\mathcal{I} = (V, C)$, and let V_i, L_i, and R_i be defined as above with respect to φ. If $|V| \ge 8$, then one of the following holds:*

1. *There is an index $0 \le i \le w+1$ such that V_i is sparse, $|L_i| \ge n/3$ and $|R_i| \ge n/3$.*
2. *There are indices $0 \le i < j \le w+1$ such that V_i and V_j are sparse, V_s is dense for all $i < s < j$, $|L_i| < n/3$ and $|R_j| < n/3$.*

Proof. If V_i is sparse, then $|L_i| + |R_i| = |V| - |V_i| \ge n - n/\log n$. Since $\log n \ge 3$, we have $|L_i| + |R_i| \ge 2n/3$. Thus, for every sparse V_i either $|L_i| \ge n/3$ or $|R_i| \ge n/3$.

Let i be the maximal index such that V_i is sparse and $|R_i| \ge n/3$. Similarly, let j be the minimal index such that V_j is sparse and $|L_j| \ge n/3$. Such indices always exist since V_0 and V_{w+1} are sparse. If $i \ge j$, then both V_i and V_j meet the conditions of Case 1. Otherwise, all V_s for $i < s < j$ are dense. If neither V_i nor V_j fulfils the conditions of Case 1, then $|L_i| < n/3$ and $|R_j| < n/3$, and we are in Case 2. □

Algorithm 1

```
procedure Solve(I = (V, C))
  if ∅ ∈ C then reject
  if |V| < 8 then
    accept if ListCert(I) ≠ ∅ else reject
  if ThreeSplit(I) then accept
  if FiveSplit(I) then accept
  reject

procedure ThreeSplit(I = (V, C))
  for each 3-partition (X, Y, Z) of V do
    if |X|, |Z| ≥ |V|/3 and |Y| ≤ |V|/log|V| and
      z − x > k ∈ σ_C(x, z) for x ∈ X, z ∈ Z then
      // introduce a fresh variable y_min
      Y' ← Y ∪ {y_min}
      I_{Y'} ← (Y', C_Y ∪ {0 ≤ y − y_min < k}_{y∈Y})
      for F_{Y'} ∈ ListCert(I_{Y'}) do
        I_1 ← (X ∪ Y', C_{X∪Y} ∪ F_{Y'} ∪ {y_min − x > 0}_{x∈X})
        I_2 ← (Y' ∪ Z, C_{Y∪Z} ∪ F_{Y'} ∪ {z − y_min > k}_{z∈Z})
        if Solve(I_1) and Solve(I_2) then
          accept
  reject

procedure FiveSplit(I = (V, C))
  for each 5-partition (S_1, S_2, S_3, S_4, S_5) of V do
    X ← S_3 ∪ S_4 ∪ S_5
    if |S_1|, |S_5| < |V|/3 and |S_2|, |S_4| ≤ |V|/log|V| and
      (x − s_1 > k) ∈ σ_C(s_1, x) for s_1 ∈ S_1, x ∈ X and
      (s_5 − s_3 > k) ∈ σ_C(s_3, s_5) for s_3 ∈ S_3, s_5 ∈ S_5 then
      // introduce fresh variables s_2^min and s_4^min
      S'_2 = S_2 ∪ {s_2^min}
      S'_4 = S_4 ∪ {s_4^min}
      I_{S'_2} ← (S'_2, C_{S_2} ∪ {0 ≤ s_2 − s_2^min < k}_{s_2∈S_2})
      I_{S'_4} ← (S'_4, C_{S_4} ∪ {0 ≤ s_4 − s_4^min < k}_{s_4∈S_4})
      for F_{S'_2} ∈ ListCert(I_{S'_2}) and
          F_{S'_4} ∈ ListCert(I_{S'_4}) do
        I_1 ← (S_1 ∪ S'_2, C_{S_1∪S_2} ∪ F_{S'_2} ∪ {s_2^min − s_1 > 0}_{s_1∈S_1})
        I_2 ← (S'_2 ∪ S_3 ∪ S'_4, C_{S_2∪S_3∪S_4} ∪ F_{S'_2} ∪ {s_3 − s_2^min > k}_{s_3∈S_3} ∪ F_{S'_4} ∪ {s_4^min − s_3 > 0}_{s_3∈S_3})
        I_3 ← (S'_4 ∪ S_5, C_{S_4∪S_5} ∪ F_{S'_4} ∪ {s_5 − s_4^min > k}_{s_5∈S_5})
        w ← k(log|V| + 2)
        if Solve(I_1) and Solve(I_3) and
          SolveBounded(w, I_2) then
          accept
  reject
```

We are now ready to prove that the algorithm is correct and analyse its running time.

Theorem 13. *Algorithm 1 solves instances of* CSP($\mathbf{D}_{2,k}$) *in $2^{O(n \log\log n)}$ time.*

Proof. Consider an arbitrary instance $\mathcal{I} = (V, C)$ of CSP($\mathbf{D}_{2,k}$). We prove the claim by induction based on $|V|$. If the instance has fewer than 8 variables, the claim follows by Corollary 7. Assume henceforth that $|V| \geq 8$.

First, we prove that if the algorithm accepts an instance, then it is satisfiable. Suppose the procedure ThreeSplit accepts $\mathcal{I}$. We show that instance $\mathcal{I}$ is satisfiable since all four conditions of Lemma 10 are fulfilled. First, note that subinstances $\mathcal{I}_1$ and $\mathcal{I}_2$ in lines 15 and 16 have at most $2n/3 + 1 < n$ variables. Hence, they admit satisfying assignments by the inductive hypothesis. Condition 1 is ensured by the check on line 11. Condition 2 is ensured since both subinstances $\mathcal{I}_1$ and $\mathcal{I}_2$ satisfy all constraints in $F_{Y'}$. Conditions 3 and 4 are ensured by the introduction of $y_{\min}$ and the constraints involving it.

Suppose instead that the procedure FiveSplit accepts $\mathcal{I}$. First, note that subinstances $\mathcal{I}_1$ and $\mathcal{I}_3$ in lines 32 and 34 have at most $n/3 + 1 < n$ variables. Hence, they admit satisfying assignments by the inductive hypothesis. Subinstance $\mathcal{I}_2$ is satisfiable by Lemma 9. Observe that Lemma 10 applies to the subinstance induced by $S_3 \cup S_4 \cup S_5$. We see that Condition 1 is ensured by the check on line 25, Condition 2 is ensured since both subinstances $\mathcal{I}_2$ and $\mathcal{I}_3$ satisfy all constraints in $F_{S'_4}$, and Conditions 3 and 4 are ensured by the introduction of $s_4^{\min}$ and the constraints involving it. Let $X = S_3 \cup S_4 \cup S_5$ and consider the subinstance induced by $S_1 \cup S_2 \cup X$. Lemma 10 is applicable and we see that Condition 1 is ensured by the check on line 24, Condition 2 is ensured since both subinstances $\mathcal{I}_1$ and $\mathcal{I}_2$ satisfy all constraints in $F_{S'_2}$, and Conditions 3 and 4 are ensured by the introduction of $s_2^{\min}$ and the constraints involving it. Hence, we have showed that $\mathcal{I}$ is satisfiable.

We proceed by proving the other direction: if $\mathcal{I}$ is satisfiable with satisfying assignment φ, then the algorithm accepts it. Lemma 12 implies that φ splits the instance as in Case 1 or 2.

Case 1. We show that ThreeSplit accepts $\mathcal{I}$. The procedure enumerates every 3-partition of the variables, so at some step of the algorithm $X = L_i$, $Y = V_i$ and $Z = R_i$, where (L_i, V_i, R_i) is the split under the assignment φ according to Lemma 12 Case 1. We set $\varphi(y_{\min}) = ki - k$ and observe that φ satisfies instances in Lines 13, 15 and 16. Note that the procedure Solve accepts $\mathcal{I}_1$ and $\mathcal{I}_2$ by the inductive hypothesis.

Case 2. We show that FiveSplit accepts $\mathcal{I}$. The procedure enumerates every 5-partition of the variables, so at some step of the algorithm $S_1 = L_i$, $S_2 = V_i$, $S_3 = R_i \cap L_j$, $S_4 = V_j$ and $S_5 = R_j$, where $(L_i, V_i, R_i \cap L_j, V_j, R_j)$ is the split under the assignment φ according to Lemma 12 Case 2. We set $\varphi(s_2^{\min}) = ki - k$ and $\varphi(s_4^{\min}) = kj - k$ and observe that φ satisfies the instances in Lines 28, 29, 32, 33 and 34. Note that the procedure Solve accepts $\mathcal{I}_1$ and $\mathcal{I}_3$ by the inductive hypothesis. By Lemma 12, all subsets in $R_i \cap L_j$ are dense. Since each of them contains at least $n/\log n$ variables, there may be at most $\log n$ such subsets. Taking V_i and V_j into account, we conclude that φ satisfies instance $\mathcal{I}_2$ with span at most $k(\log n + 2)$. This completes the correctness proof.

Time complexity. Let $T(n)$ be the running time of Algorithm 1 on an instance of CSP($\mathbf{D}_{2,k}$) with n variables. We claim that $T(n) \leq c^n (\log n + 2)^n = 2^{O(n \log\log n)}$ for some constant c. If $n < 8$, then $T(n)$ is constant. Otherwise, $T(n) = T_1(n) + T_2(n) + \mathrm{poly}(n)$, where T_1 and T_2 are the running times of the procedures ThreeSplit and FiveSplit, respectively. Note that $T_1(n) < T_2(n)$ for all n, so we can focus our attention on the running time of FiveSplit.

The running time $T_2(n)$ is bounded from above by $5^n \cdot 2^{2n} \cdot (2T(\frac{n}{3} + 1) + (k(\log n + 2))^n) \cdot \mathrm{poly}(n)$, where 5^n is an upper bound on the number of 5-partitions of V, 2^{2n} comes from the upper bound on the running time of the calls to ListCert in lines 30 and 31, $2T(\frac{n}{3} + 1)$ is an upper bound on the running time of the recursive calls in line 36, and $(k(\log n + 2))^n$ comes from the running time of the bounded-span algorithm SolveBounded in line 37 (see Lemma 9). Observe that for sufficiently large values of n we have $(\log n + 2)^n > \alpha \log(\varepsilon n)^{\varepsilon n}$ for arbitrary $\alpha \geq 0$ and $0 \leq \varepsilon < 1$. Hence, $(k(\log n + 2))^n$ asymptotically dominates $2T(\frac{n}{3} + 1)$ by our initial hypothesis. Finally, observe that

$$T(n) < 2 \cdot (40k)^n \cdot (\log n + 2)^n \cdot \mathrm{poly}(n).$$

Setting $c = 81k$ completes the proof. □

4. Lower bounds on time complexity

This section contains our time complexity lower bound results for various subclasses of CSP($\mathbf{D}$). The full picture of upper and lower bounds on time complexity is summarised in Table 3. The majority of our lower-bound results are based on reductions from the $n \times n$ Independent Set problem.

> $n \times n$ Independent Set
> Input: A graph $G = (V, E)$ with vertex set $V = \{(i, j) : 1 \leq i \leq n, 1 \leq j \leq n\}$.
> Question: Is there an independent set in G with one vertex from each row, i.e. subset of vertices $\{(1, j_1), \ldots, (n, j_n)\}$ where no pair of vertices is connected by an edge?

Lokshtanov et al. [56] proved the following result.

Theorem 14. *$n \times n$ Independent Set cannot be solved in $2^{o(n \log n)}$ time unless the ETH fails.*

We begin by proving lower bounds for CSP($\mathbf{D}_{a,0}$) when $a \geq 4$. Note that CSP($\mathbf{D}^{\leq}_{a,0}$) is trivially in P for every $a \geq 0$ so this result cannot be extended to CSP($\mathbf{D}^{\leq}_{a,0}$).

Theorem 15. CSP($\mathbf{D}_{4,0}$) *cannot be solved in* $2^{o(n \log n)}$ *time unless the ETH fails.*

Proof. We reduce from $n \times n$ Independent Set. Given an instance G of this problem, do the following.

1. Introduce n column variables $c_1, \ldots, c_n$ and the constraints $c_i < c_{i-1}$ for all $i \in \{2, \ldots, n\}$.
2. Introduce n row variables $r_1, \ldots, r_n$. To ensure that each r_i is equal to one of the column variables, add the following constraints: $c_1 \leq r_i$, $r_i \leq c_n$ and $(r_i \leq c_{j-1}) \vee (r_i \geq c_j)$ for all $j \in \{2, \ldots, n\}$.
3. No pair of vertices (i, j) and (k, ℓ) adjacent in G can be simultaneously included in the independent set. To ensure this property, we add the following constraint:

$$(r_i < c_j) \vee (r_i > c_j) \vee (r_k < c_\ell) \vee (r_k > c_\ell).$$

The resulting set of constraints only use relations in CSP($\mathbf{D}_{4,0}$). The correctness of the reduction is easy to verify: If G has an independent set I, then setting $r_i = c_j$ for all $(i, j) \in I$ satisfies all constraints of the instance above, and vice versa. The reduction requires polynomial time and introduces $2n$ variables. Thus, if CSP($\mathbf{D}_{4,0}$) admits a $2^{o(n \log n)}$ algorithm, then so does $n \times n$ Independent Set and this contradicts the ETH by Theorem 14. □

A slightly weaker bound can be inferred from Theorem 11 in [48]: if the randomised ETH holds, then there is no randomised algorithm for CSP($\mathbf{D}_{4,0}$) that runs in $O(c^n)$ time for any $c \geq 0$. We continue by studying CSP($\mathbf{D}^{\leq}_{3,1}$). We use Sidon sets in the proof so the reader may want to skip back to Section 2.3 for a reminder.

Theorem 16. CSP($\mathbf{D}^{\leq}_{3,1}$) *is not solvable in* $2^{o(n \log n)}$ *time if the ETH holds.*

Proof. The proof is by a reduction from $n \times n$ Independent Set. Assume $G = (V, E)$ to be an arbitrary instance of this problem with r rows. We use the results from Section 2.3 and construct (in polynomial time) a Sidon set $S = \{a_0, \ldots, a_r\}$ where $0 = a_0 < a_2 < \cdots < a_r$ and $a_r \leq 8r^2$. We present the rest of the reduction in three steps.

1. Introduce a_r fresh variables $y_0, \ldots, y_{a_r}$ and use the relation $y = x + 1$ for enforcing $y_i = i$, $0 \leq i \leq a_r$.
2. Introduce one variable x_r for each row r. We first ensure that the range of each variable is in $\{0, \ldots, y_{a_r}\}$ by adding the constraint

$$x_r \leq a_i - 1 \vee x_r = a_i \vee x_r \geq a_i + 1$$

for each $1 \leq i \leq a_r$ together with the constraints $x \geq 0$ and $x \leq y_{a_r}$. Next, we restrict the range to be in S by adding the constraint

$$x \leq y_s - 1 \vee x \geq y_s + 1$$

for each $s \in \{0, \ldots, a_r\} \setminus S$.
3. Let $R(x, y, z)$ denote the relation $x - y \neq z$, i.e. $R(x, y, z) \equiv (x - y < z) \vee (y - x < z)$. For every edge $((c, c'), (d, d')) \in E$, compute $e = a_{c'} - a_{d'}$ and add the constraint $R(x_c, x_d, y_e)$.

The resulting set of constraints only use relations in CSP($\mathbf{D}^{\leq}_{3,1}$) This reduction can be performed in polynomial time. Steps 1. and 2. take polynomial time since $a_r \leq 8r^2 \leq 8\|G\|^2$ and step 3. can obviously be performed in polynomial time in the size of E. Let (V', C') denote the resulting instance of CSP($\mathbf{D}_{3,1}$). We see that $|V'| \leq 8r^2 + r \leq 8|V| + \lceil\sqrt{|V|}\rceil \leq 9|V|$. Thus, if (V', C') is satisfiable if and only if G is a yes-instance, then CSP($\mathbf{D}_{3,1}$) is not solvable in $2^{o(n \log n)}$ time by Theorem 14. We conclude the proof by proving this equivalence.

Forward direction. Assume f is a solution to (V', C'). We claim that for every $1 \leq c \neq d \leq r$, it holds that $((c, f(c)), (d, f(d)))$ is not in E and G is a yes-instance. Assume to the contrary that c, d can be chosen such that $((c, f(c)), (d, f(d))) \in E$. This edge implies that there is a constraint $R(x_c, x_d, y_e) \in C'$ where $e = a_{f(c)} - a_{f(d)}$. Hence,

$$\begin{aligned}
&f(x_c) - f(x_d) \neq f(y_e) \Rightarrow \\
&a_{c'} - a_{d'} \neq a_{f(c)} - a_{f(d)} \Rightarrow \\
&a_{c'} - a_{d'} \neq a_{c'} - a_{d'}
\end{aligned}$$

and we have reached a contradiction.

Backward direction. Assume $X = \{v_1, \ldots, v_r\} \subseteq V$ is an independent set and v_i occurs in position (i, i'), $1 \leq i, i' \leq r$. Define the assignment f such that $f(x_i) = a_{i'}$, $1 \leq i \leq r$, and $f(y_i) = i$, $0 \leq i \leq a_r$. This assignment satisfies all constraints introduced in step 2.

Arbitrarily choose an edge $((c, c^*), (d, d^*)) \in E$. It gives rise to the constraint $R(x_c, x_d, y_e)$ where $e = (a_{c^*} - a_{d^*})$. We know that $((c, c'), (d, d'))$ is a non-edge in E since $(c, c') = v_c$, $(d, d') = v_d$, and v_c, v_d are in the independent set X. The constraint $R(x_c, x_d, y_e)$ is equivalent to $x_c - x_d \neq y_e$. We apply the assignment f to it:

$$f(x_c) - f(x_d) \neq f(y_{a_{c^*} - a_{d^*}}) \Rightarrow$$
$$a_{c'} - a_{d'} \neq a_{c^*} - a_{d^*}$$

We know that $((c, c'), (d, d'))$ is a non-edge in E while $((c, c^*), (d, d^*))$ is an edge in E. Hence, $a_{c'} \neq a_{c^*}$ or $a_{d'} \neq a_{d^*}$. Recall that for all $w_1, \ldots, w_4 \in S$ such that $w_1 \neq w_2$ and $w_3 \neq w_4$, $w_1 - w_2 = w_3 - w_4$ if and only if $w_1 = w_3$ and $w_2 = w_4$. Thus, if $a_{c'} \neq a_{d'}$ and $a_{c^*} \neq a_{d^*}$, then the disequality holds. If $a_{c'} = a_{d'}$, then $a_{c^*} \neq a_{d^*}$ (since $((c, c'), (d, d')) \notin E$ and $((c, c^*), (d, d^*)) \in E$) and we conclude that the disequality holds. The case when $a_{c^*} = a_{d^*}$ is symmetric. $\square$

Our final lower bound based on a reduction from $n \times n$ Independent Set concerns the problem CSP($\mathbf{D}_2^{\leqslant}$). Since there is no upper bound on the coefficients in this case, the lower bound is parameterized both by the number of variables and the maximum over the coefficients.

Theorem 17. CSP($\mathbf{D}_2^{\leqslant}$) *is not solvable in* $2^{o(n(\log n + \log k))}$ *time if the ETH holds.*

Proof. The proof is by reduction from $n \times n$ Independent Set. Given an instance G of this problem, we introduce a zero variable z and two variables x_r, x'_r for each row r. We add the following constraints:

1. $\bigvee_{i=1}^{n} x_r - z = i$ for every row r,
2. $\bigvee_{i=1}^{n} x'_r - x_r = ni - i$ and $\bigvee_{i=1}^{n} x'_r - z = ni$ for every row r,
3. $x'_a - x_b \in (-\infty, ni - j - 1] \vee x'_a - x_b \in [ni - j + 1, \infty)$ for every edge $\{(a, i), (b, j)\}$ in G.

Constraints of the first type restrict the domain of x_r to $\{1, \ldots, n\}$. Constraints of the second type ensure that $x'_r = nx_r$. Constraints of the third type ensure are equivalent to $x'_a - x_b \neq ni - j$, which forbids setting $x_a = i$ and $x_b = j$. Furthermore, function $f : \{1, \ldots, n\} \times \{1, \ldots, n\} \to \{0, \ldots, n^2 - 1\}$ defined as $f(i, j) = ni - j$ is bijective, so $x'_a - x_b \neq ni - j$ is satisfied by every other choice of values, i.e. whenever $x_a \neq i$ or $x_b \neq j$. Thus, the resulting instance of CSP($\mathbf{D}_2^{\leqslant}$) has a solution if and only if G has an independent set with one variable per row. The total number of variables in the resulting instance is $2n + 1$ and the absolute values of the integers appearing in the constraints do not exceed n^2. Thus, an algorithm solving CSP($\mathbf{D}_2^{\leqslant}$) in $2^{o(n(\log n + \log k))}$ time can be used for solving $n \times n$ Independent Set in

$$2^{o(n(\log(2n+1) + \log n^2))} = 2^{o(n \log n)}$$

time and this contradicts the ETH by Theorem 14. $\square$

Our final lower bound for CSP($D_{2,k}$) relies on a modification of a result by Traxler [77]. Let d-CSP be the constraint satisfaction problem with domain $D = \{1, \ldots, d\}$ and binary relations $R_{a,b} = (x \neq a \vee y \neq b)$ for all $a, b \in D$. Note that this relation is akin to the binary relation in our reductions from $(n \times n)$ Independent Set that is used to forbid choosing an edge. We cannot use the same reduction here because the numerical bound k is fixed. Indeed, a $2^{o(n \log n)}$ lower bound for this problem stands in contradiction with Theorem 13 or the ETH. For our reductions, we consider a modified version of d-CSP denoted by d-CSP$^{\chi}$. It allows binary relations

$$R^{\chi}_{a,b} = (x \neq a \vee y \neq b) \wedge \bigwedge_{c \in D} (x \neq c \vee y \neq c)$$

for all $a, b \in D$. For convenience, we consider the constraints that rule out (c, c) tuples (i.e. the right-hand side constraints in the definition above) separately, and assume the following rule: if an instance of d-CSP$^{\chi}$ includes a constraint $(x \neq a \vee y \neq b)$, then it implicitly includes the required constraints $(x \neq c \vee y \neq c)$ for all $c \in D$. Additionally, we allow unary relations $x \neq a$ for all $a \in D$.

Lemma 18. *For any* $r \in \mathbb{N}$ *and any instance of* d-CSP$^{\chi}$ *with* n *variables, there exists an equivalent instance of* d^r-CSP$^{\chi}$ *with* $\lceil n/r \rceil$ *variables.*

Proof. Let $\mathcal{I} = (V, \mathcal{C})$ be an instance of d-CSP$^{\chi}$ with $|V| = n$. Augment V with at most $r - 1$ extra variables so that its new size n' becomes a multiple of r. Partition V into $\ell = n'/r = \lceil n/r \rceil$ disjoint subsets $V_1, \ldots, V_\ell$ of equal size and index the elements of each subset arbitrarily.

The set of tuples D^r represents all assignments to the variables in a subset V_i. For convenience, we use the tuples directly as the domain of d^r-CSP$^{\mathcal{X}}$. Define an instance $\mathcal{I}' = (V', C')$ of d^r-CSP as follows. For each subset V_i in the partition introduce a variable z_i to V'. Note that $|V'| = \ell$.

First, consider a unary constraint $x \neq a$ in C. Assume $x \in V_i$ and i_x is the index of x in V_i. Add constraints $z_i \neq t$ to C' for all $t \in D^r$ such that $t_{i_x} = a$. Now consider a binary constraint $(x \neq a) \vee (y \neq b)$ in C. Assume $x \in V_i$, $y \in V_j$ and i_x, j_y are the indices of x, y in the respective subsets. If $i \neq j$, then add constraints $(z_i \neq s \vee z_j \neq t)$ for all tuples $s, t \in D^r$ such that $s_{i_x} = a$ and $t_{j_y} = b$. If $i = j$ (index of y is i_y), add unary constraints $z_i \neq t$ for all t such that $t_{i_x} = a$ and $t_{i_y} = b$.

Observe that the required constraints $(x \neq c \vee y \neq c)$ for all $c \in D$ are converted into $(z_i \neq s \vee z_j \neq t)$ for every pair of tuples $s, t \in D^r$ that coincide in at least one position. Clearly, this includes the case when s and t are equal. Hence, $\mathcal{I}'$ is an instance of d^r-CSP$^{\mathcal{X}}$. Proving the equivalence of $\mathcal{I}$ and $\mathcal{I}'$ is analogous to the proof of Lemma 1 in [77]. □

We are now ready to prove the lower bound.

Theorem 19. *If we assume that the ETH holds, then for arbitrary $m > 0$ there is an integer k such that any algorithm solving* CSP($\mathbf{D}^{\leq}_{2,k}$) *requires at least $O^*(m^n)$ time.*

Proof. We start by proving that the time complexity of d-CSP$^{\mathcal{X}}$ increases with d assuming the ETH. Let

$$c_d = \inf\{c \in \mathbb{R} \ : \text{there is a } 2^{cn}\text{-time algorithm solving } d\text{-CSP}^{\mathcal{X}}\}.$$

We show that $\lim_{d \to \infty} c_d = \infty$. The 3-Colourability problem cannot be solved in subexponential time assuming the ETH [46]. 3-CSP$^{\mathcal{X}}$ is a generalisation of 3-Colourability so $c_3 > 0$. By Lemma 18, for any $r \in \mathbb{N}$ we have $c_{3^r} \geq c_3 \cdot r$. Observe that $\lim_{r \to \infty} c_3 \cdot r = \infty$, so $\lim_{d \to \infty} c_d = \infty$.

Next, we show that for any instance of d-CSP$^{\mathcal{X}}$ with n variables there is an equivalent instance of CSP($\mathbf{D}^{\leq}_{2,k}$) with $n+1$ variables and $k \in O(d^2)$. Let $\mathcal{I}$ be an instance of d-CSP$^{\mathcal{X}}$ with n variables. Construct an instance $\mathcal{I}'$ of CSP($\mathbf{D}_{2,k}$) as follows. Choose k so that $\{-k, \ldots, k\}$ contains a Sidon set of order d as a subset. One can choose such a k to be in $O(d^2)$ by Section 2.3. Denote the corresponding Sidon set by G_d. Associate each integer in $\{1, \ldots, d\}$ with a unique element of G_d via the bijection $\rho : \{1, \ldots, d\} \to G_d$. Introduce zero variable z to express unary relations. For each variable x in $\mathcal{I}$ introduce a new variable v_x. Define $E_x = \{\rho(c) : (x \neq c) \in C\}$. Restrict the domain of v_x to $D_x = G_d \setminus E_x$ by the constraint $\bigvee_{i \in D_x} v_x - z \in \{i\}$. Any constraint $(x \neq a \vee y \neq b)$ can be expressed by enforcing $v_x - v_y \neq \delta$ where $\delta = \rho(a) - \rho(b)$. This is done by adding the constraint

$$v_x - v_y \in (-\infty, \delta - 1] \vee v_x - v_y \in [\delta + 1, \infty).$$

By the properties of Sidon sets, this constraint is satisfied if and only if $x \neq \rho(a)$ or $y \neq \rho(b)$.

The reduction introduces only one extra variable and the lower bound on d-CSP$^{\mathcal{X}}$ carries over to CSP($\mathbf{D}^{\leq}_{2,k}$). □

5. Parameterized upper bounds

The results in Section 4 imply that most subproblems of CSP($\mathbf{D}$) cannot be solved in subexponential time under the ETH. This is a strong motivation for analysing the problems from a parameterized perspective. A highly successful approach for identifying tractable fragments of CSPs is to restrict variable-constraint interactions via the underlying primal and incidence graphs. The *primal graph* has the variables as its vertices with any two joined by an edge if they occur together in a constraint. The *incidence graph* is the bipartite graph with two disjoint sets of vertices corresponding to the variables and the constraints, respectively, and a constraint vertex and a variable vertex are joined by an edge if the variable occurs in the scope of the constraint. The *treewidth* of such graphs has been used extensively and it has been successfully employed for many application areas. In particular, it has been used for problems such as SAT, CSP, and ILP [11,13,34,40,41,45,70]. Formal definitions and some auxiliary results are collected in Section 5.1.

The results in this section (together with the lower bound results that will be proved in Section 6) are summarised in Table 4. The main result is an XP-algorithm for CSP($\mathbf{D}_{\infty,k}$), $k < \infty$, where the parameter is the treewidth of the incidence graph. This algorithm is presented in Section 5.2. The treewidth of the incidence graph cannot be larger than the treewidth of the primal graph plus one so tractability results are more general if they hold for incidence treewidth and hardness results are more general if they hold for primal treewidth.

5.1. Treewidth

Treewidth is based on *tree decompositions* [8,69]: a tree decomposition (T, χ) of an undirected graph $G = (V, E)$ consists of a rooted tree T and a mapping χ from nodes $V(T)$ of the tree to subsets of V. The subsets $\chi(t)$ are called *bags*. T_t denotes the sub-tree rooted at t, while $\chi(T_t)$ denotes the set of all vertices occurring in the bags of T_t, i.e. $\chi(T_t) = \bigcup_{s \in V(T_t)} \chi(s)$. A tree decomposition has the following properties:

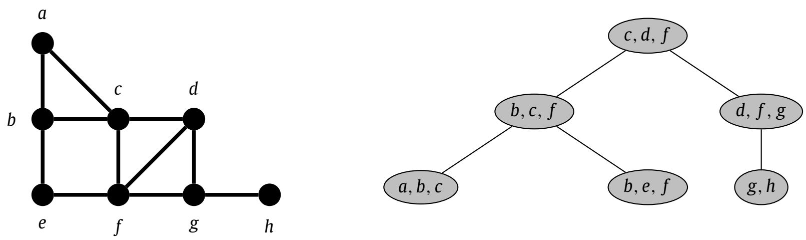

Fig. 1. A graph (left) and an optimal tree decomposition of the graph (right).

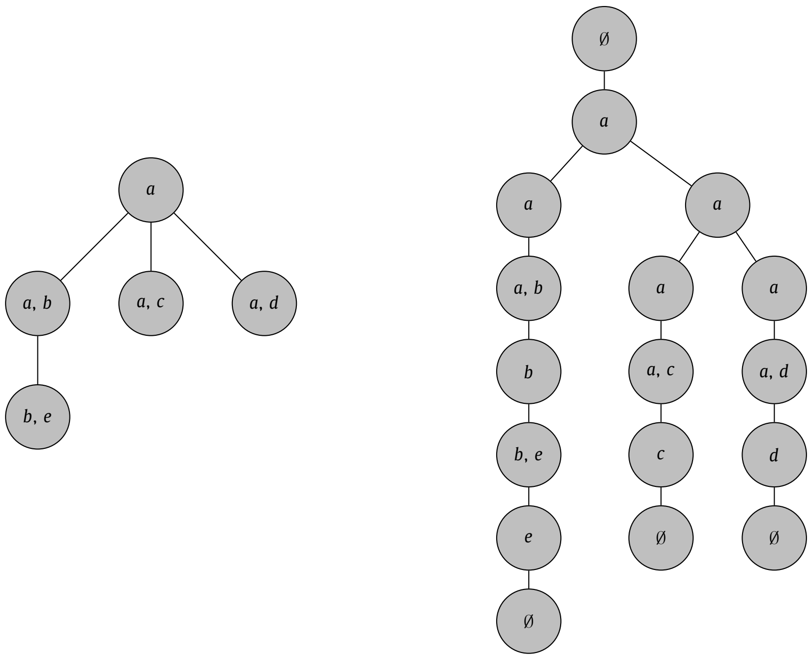

Fig. 2. A tree decomposition (left) and a corresponding nice tree decomposition (right).

1. for every $\{u, v\} \in E$, there is a node $t \in V(T)$ such that $u, v \in \chi(t)$, and
2. for every $v \in V$, the set of bags of T containing v forms a non-empty sub-tree of T.

An example is given in Fig. 1. The width of a tree decomposition T is $\max\{|\chi(t)| - 1 : t \in T\}$. The *treewidth* of a graph G, denoted by $\mathrm{tw}(G)$, is the minimum width of a tree decomposition of G. It is NP-complete to determine if a graph has treewidth at most w [4] but when w is fixed, the graphs with treewidth w can be recognised and corresponding tree decompositions can be constructed in linear time [16].

We simplify the presentation by using restricted tree decompositions. A tree decomposition is *nice* if (1) $\chi(r) = \varnothing$ for the root r and $|\chi(l)| = 1$ and for all leaf nodes l in T, and (2) every non-leaf node in T is of one of the following types:

- An *introduce node*: a node t with exactly one child t_0 such that $\chi(t) = \chi(t_0) \cup \{v\}$ for some $v \in V$.
- A *forget node*: a node t with exactly one child t_0 such that $\chi(t) = \chi(t_0) \setminus \{v\}$ for some $v \in V$.
- A *join node*: a node t with exactly two children t_1 and t_2 such that $\chi(t) = \chi(t_1) = \chi(t_2)$.

Nice tree decompositions are illustrated in Fig. 2. Note that nice tree decompositions are merely a structured type of tree decomposition and their only purpose is to simplify the presentation of dynamic programming algorithms. It is NP-complete to determine if a graph has treewidth at most w [4], but when w is fixed, then graphs with treewidth w can be recognised and corresponding tree decompositions can be constructed in linear time.

Proposition 20 *(Bodlaender & Kloks [17]; Kloks [51]). Let $G = (V, E)$ be a graph. For fixed w, if G has treewidth at most w, then a nice tree decomposition of width at most w with $O(|V|)$ nodes can be computed in linear time.*

Let $\mathcal{I} = (V, C)$ be an instance of CSP(**A**). The *primal graph* of $\mathcal{I}$, denote by $P(\mathcal{I})$, is the graph with vertices V having an edge between two variables if both appear together in the scope of one constraint. The incidence graph of $\mathcal{I}$, denote by $I(\mathcal{I})$, is the bipartite graph with V on one side and C on the other side having an edge between a variable and a constraint if the variable appears in the scope of the constraint. It is well known that the treewidth of the incidence graph is at most equal to the treewidth of the primal graph plus one [52] and that the incidence treewidth can be arbitrary smaller than the primal treewidth. This means that tractability results are more general if they hold for incidence treewidth and hardness results are more general if they hold for primal treewidth.

We continue with a few observations concerning treewidth. We discussed in Section 2.2 that zero variables can be used for simulating unary constraints, and that they do not affect the time complexity with more than a multiplicative factor. We have a similar situation in the parameterized setting: adding a zero variable can increase the treewidth of the incidence graph by at most 1 so algorithmic results (such as the forthcoming Theorem 22) are still valid for this extended formalism. We will use a related observation when we argue about the treewidth of the graphs obtained in our hardness result (Theorem 33).

Proposition 21 *(Bodlaender [16]). Let G be a graph and $B \subseteq V(G)$. Then the treewidth of G is at most $|B| + \mathrm{tw}(G - B)$.*

5.2. XP-algorithm for CSP($\mathbf{D}_{\infty,k}$)

We are now ready to present our dynamic programming algorithm for CSP($\mathbf{D}_{\infty,k}$).

Theorem 22. CSP($\mathbf{D}_{\infty,k}$) *can be solved in $(nk)^{O(w)}$ time, where w is the treewidth of the incidence graph and n is the number of variables.*

Note that the bound implies that CSP(**D**) is in XP whenever the numeric values are bounded by a polynomial in the input size. Proposition 20 implies that the computation of a nice tree decomposition of the incidence graph does not incur an additional run-time overhead. We may thus assume that a nice tree decomposition is provided in the input, and it is hence sufficient to show the following.

Theorem 23. CSP($\mathbf{D}_{\infty,k}$) *can be solved in time $(nk)^{O(w)}$ provided that a nice tree decomposition of the incidence graph of width at most w is given as part of the input.*

Let $\mathcal{I} = (V, C)$ be an instance of CSP($\mathbf{D}_{\infty,k}$) with n variables and assume (T, χ) is a nice tree decomposition of the incidence graph of $\mathcal{I}$ of width w. Bags of this decomposition contain vertices corresponding to both variables and constraints. To distinguish between them, we use $\mathrm{var}_\chi(t)$ to denote all variables in the bag $\chi(t)$ and $\mathrm{con}_\chi(t)$ to denote all constraints in $\chi(t)$. These definitions naturally extend to the subsets of $V(T)$. Note that by Theorem 6, we may assume that every solution for $\mathcal{I}$ maps the variables into the set $CD = CD(n, k)$.

Intuitively, the algorithm behind Theorem 22 works as follows. It uses bottom-up dynamic programming on the nodes of T starting from the leaves and finishing at the root. It computes a compact representation, represented by a set of valid records, of all solutions to $\mathcal{I}$ restricted to the variables and constraints in $\chi(T_t)$ for every node $t \in V(T)$.

A record for $t \in V(T)$ is a pair (α, β) where

- $\alpha : \mathrm{var}_\chi(t) \to CD$ is an assignment of values in CD to the variables in $\mathrm{var}_\chi(t)$, and
- $\beta : \mathrm{con}_\chi(t) \to D_B$, where $D_B = \{S, U\} \cup \{ (v, d) \mid v \in V \text{ and } d \in CD \}$ such that for every constraint $c \in \mathrm{con}_\chi(t)$ either:
  - $\beta(c) = S$ signalling that the constraint c is already satisfied,
  - $\beta(c) = U$ signalling that the constraint c is not yet satisfied,
  - $\beta(c) = (v, d)$, where $v \in \mathrm{scope}(c) \cap (\mathrm{var}_\chi(T_t) \setminus \mathrm{var}_\chi(t))$ and $d \in CD$, signalling that c is not yet satisfied, but satisfying c can use the assumption that v is set to d. This also means that c will be satisfied by satisfying a simple constraint on v and some variable in $V \setminus \mathrm{var}_\chi(T_t)$.

Note that there are at most $|CD|$ possible choices for every variable in $\mathrm{var}_\chi(t)$ and at most $|V||CD| + 2$ possible choices for every constraint in $\mathrm{con}_\chi(t)$. Therefore, the total number of valid records for t is at most $(|V||CD| + 2)^{w+1}$.

For $X \in \{S, U\}$, define the inverse $\beta^{-1}(X)$ as $\{c \in \mathrm{con}_\chi(t) \mid \beta(c) = X\}$ and let

$$\beta^{-1}(F) = \mathrm{con}_\chi(t) \setminus (\beta^{-1}(S) \cup \beta^{-1}(U)),$$

i.e. $\beta(c) = (v, d)$ for some $v \in V$ and $d \in CD$ for all $c \in \beta^{-1}(F)$.

The semantic of a record is defined as follows. We say that a record (α, β) is *valid* for t if there is an assignment $\tau : \mathrm{var}_\chi(T_t) \to CD$ such that:

(R1) τ does not satisfy any constraint in $Y = \mathrm{con}_\chi(t) \setminus \beta^{-1}(S)$ and satisfies all constraints in $\mathrm{con}_\chi(T_t) \setminus Y$,

(R2) $\tau(v) = \alpha(v)$ for every $v \in \text{var}_\chi(t)$, and
(R3) $\tau(v) = d$ holds for every constraint $c \in \text{con}_\chi(t)$ with $\beta(c) = (v, d)$.

Let $\mathcal{R}(t)$ be the set of all valid records for t. Note that $\mathcal{I}$ has a solution if and only if $\mathcal{R}(r) \neq \emptyset$ for the root r of T since the records in $\mathcal{R}(r)$ represent solutions for the whole instance. Moreover, once we have computed the set of records for all nodes, a straightforward application of standard techniques [29] can be used to obtain a solution for $\mathcal{I}$ using a second top-to-bottom run through the tree decomposition.

Next, we will show that $\mathcal{R}(t)$ can be computed via a dynamic programming algorithm on (T, χ) in a bottom-up manner. The algorithm starts by computing the set of all valid records for the leaves of T and then proceeds by computing the set of all valid records for the other three types of nodes of a nice tree decomposition (always selecting nodes all of whose children have already been processed). The following lemmas show how this is achieved for the different types of nodes of (T, χ).

Lemma 24 *(variable leaf node). Let $t \in V(T)$ be a leaf node with $\chi(t) = \{v\}$ for some variable $v \in V$. Then, $\mathcal{R}(t)$ can be computed in $O(|CD|)$ time.*

Proof. $\mathcal{R}(t)$ consists of all records $(\alpha, \emptyset)$ for every assignment $\alpha : \{v\} \to CD$, so $\mathcal{R}(t)$ can be computed by enumerating all assignments $\alpha : \{v\} \to CD$ for v in $O(|CD|)$ time. Correctness follows immediately from the definition of valid records. $\square$

Lemma 25 *(constraint leaf node). Let $t \in V(T)$ be a leaf node with $\chi(t) = \{c\}$ for some constraint $c \in C$. Then, $\mathcal{R}(t)$ can be computed in $O(1)$ time.*

Proof. $\mathcal{R}(t)$ consists of the record $(\emptyset, \beta)$, where $\beta : \{c\} \to D_B$ is defined by setting $\beta(c) = U$. Thus, $\mathcal{R}(t)$ can be computed in constant time and correctness follows immediately from the definition of valid records. $\square$

Lemma 26 *(variable introduce node). Let $t \in V(T)$ be an introduce node with child t_0 such that $\chi(t) \setminus \chi(t_0) = \{v\}$ for some variable $v \in V$. Then, $\mathcal{R}(t)$ can be computed in $O(|\mathcal{R}(t_0)| \cdot |CD| \cdot |\mathcal{I}|)$ time.*

Proof. Informally, the set $\mathcal{R}(t)$ is obtained from $\mathcal{R}(t_0)$ by extending every record $R_0 = (\alpha_0, \beta_0)$ in $\mathcal{R}(t_0)$ with an assignment $\alpha_v : \{v\} \to CD$ for the variable v and then updating the record (i.e. updating β_0) if α_v causes additional constraints to be satisfied. More formally, for every $(\alpha_0, \beta_0) \in \mathcal{R}(t_0)$ and every assignment $\alpha_v : \{v\} \to CD$, the set $\mathcal{R}(t)$ contains the record (α, β), where:

- $\alpha(u) = \alpha_0(u)$ for all $u \in \chi(t_0)$ and $\alpha(v) = \alpha_v(v)$,
- $\beta(c) = S$ for every constraint $c \in \beta_0^{-1}(S) \cup U' \cup F'$, where:
  - U' is the set of all constraints $c \in \beta_0^{-1}(U)$ that are satisfied by the (partial) assignment α and
  - F' is the set of all constraints $c \in \beta_0^{-1}(F)$ that are satisfied by setting v to $\alpha_v(v)$ and u to d, where $(u, d) = \beta_0(c)$.
- $\beta(c) = \beta_0(c)$ for every other constraint c, i.e. every constraint $c \in \text{con}_\chi(t) \setminus (\beta_0^{-1}(S) \cup U' \cup F')$.

Towards showing correctness of the definition for $\mathcal{R}(t)$, we first show that every valid record $R = (\alpha, \beta)$ for t is added to $\mathcal{R}(t)$. Because R is valid, there is an assignment $\tau : \text{var}_\chi(T_t) \to CD$ satisfying (R1)–(R3). Let α_0 be the restriction of α to $\text{var}_\chi(t_0)$ and let τ_0 be the restriction of τ to $\text{var}_\chi(T_{t_0})$. Let Z be the set of all constraints in $\text{con}_\chi(t) = \text{con}_\chi(t_0)$ that are satisfied by τ but not satisfied by τ_0. Moreover, let $X \subseteq Z$ contain the constraints that are satisfied by α and set $Y = Z \setminus X$. Then, for every constraint $c \in Y$, there is (at least one) variable, denoted by $y(c)$, in $\text{var}_\chi(T_t) \setminus \text{var}_\chi(t)$ such that the partial assignment setting $y(c)$ to $\tau(y(c))$ and setting v to $\alpha(v)$ satisfies c. This implies that the record $R_0 = (\alpha_0, \beta_0)$ defined by setting $\beta_0(c) = \beta(c)$ for every $c \in \text{con}_\chi(t_0) \setminus (X \cup Y)$, $\beta_0(c) = U$ for every $c \in X$, and $\beta_0(c) = (y(c), \tau(y(c)))$ for every $c \in Y$ is contained in $\mathcal{R}(t_0)$. Finally, $U' = X$ and $F' = Y$ holds for the record R_0, so R is added to $\mathcal{R}(t)$.

It remains to show that if a record $R = (\alpha, \beta)$ is added to $\mathcal{R}(t)$, then R is valid for t. Suppose that R is obtained from the record $R_0 = (\alpha_0, \beta_0) \in \mathcal{R}(t_0)$. There is an assignment $\tau_0 : \text{var}_\chi(T_{t_0}) \to CD$ satisfying (R1)–(R3) since R_0 is valid for t_0. Now it is straightforward to verify that the extension τ of τ_0 obtained by setting $\tau(v) = \alpha(v)$ witnesses that R is a valid record.

Finally, the run-time of the procedure follows because there are $|\mathcal{R}(t_0)| \cdot |CD|$ pairs of records in $\mathcal{R}(t_0)$ and assignments α_v for v. Computing the record for one such combination requires evaluating the constraints in $\text{con}_\chi(t)$ for partial assignments and thus takes $O(|\mathcal{I}|)$ time. $\square$

Lemma 27 *(constraint introduce node). Let $t \in V(T)$ be an introduce node with child t_0 such that $\chi(t) \setminus \chi(t_0) = \{c\}$ for some constraint $c \in C$. Then, $\mathcal{R}(t)$ can be computed in $O(|\mathcal{R}(t_0)||\mathcal{I}|)$ time.*

Proof. Informally, the set $\mathcal{R}(t)$ is obtained from $\mathcal{R}(t_0)$ by checking, for every record $(\alpha_0, \beta_0) \in \mathcal{R}(t_0)$, whether α_0 satisfies the constraint c and if so, extending β_0 by setting c to being satisfied, and if not, extending β_0 by setting c to being unsatisfied. More formally, for every record $(\alpha_0, \beta_0) \in \mathcal{R}(t_0)$:

- if the constraint c is satisfied by the partial assignment α_0, then $\mathcal{R}(t)$ contains the record (α_0, β), where β is the extension of β_0 that sets c to S.
- otherwise, i.e. if α_0 does not satisfy c, then $\mathcal{R}(t)$ contains the record (α_0, β), where β is the extension of β_0 that sets c to U.

Towards showing correctness of the definition for $\mathcal{R}(t)$, we first show that every valid record $R = (\alpha, \beta)$ for t is added to $\mathcal{R}(t)$. Because R is valid, there is an assignment $\tau : \text{var}_\chi(T_t) \to CD$ satisfying (R1)–(R3). Because (T, χ) is a tree decomposition, it follows that the scope of c does not contain any variable from $\text{var}_\chi(T_t) \setminus \text{var}_\chi(t)$; otherwise the edge between c and the variable in $\text{var}_\chi(T_t) \setminus \text{var}_\chi(t)$ in the incidence graph is not contained in any bag of T. Therefore, $\beta(c) \in \{S, U\}$ so if $\beta(c) = S$, then c is already satisfied by the partial assignment α. It follows that τ witnesses that the record (α, β_0), where β_0 is the restriction of β to $\text{con}_\chi(t')$, is in $\mathcal{R}(t_0)$ and R is added to $\mathcal{R}(t)$.

It remains to show that if a record $R = (\alpha, \beta)$ is added to $\mathcal{R}(t)$, then R is valid for t. Assume that R is obtained from the record $R_0 = (\alpha, \beta_0) \in \mathcal{R}(t_0)$. We know that R_0 is valid for t_0 so there is an assignment $\tau_0 : \text{var}_\chi(T_{t_0}) \to CD$ satisfying (R1)–(R3). Because (T, χ) is a tree decomposition, it follows that the scope of c does not contain any variable from $\text{var}_\chi(T_t) \setminus \text{var}_\chi(t)$. Hence, $\beta(c) \in \{S, U\}$ so if $\beta(c) = S$, then c is already satisfied by the partial assignment α. Therefore, τ witnesses that R is valid.

Finally, the run-time follows because we have to consider every record (α_0, β_0) in $\mathcal{R}(t_0)$ and we can check in $O(|\mathcal{I}|)$ time whether α satisfies c or not. □

Lemma 28 *(variable forget node). Let $t \in V(T)$ be a forget node with child t_0 such that $\chi(t_0) \setminus \chi(t) = \{v\}$ for some variable $v \in V$. Then, $\mathcal{R}(t)$ can be computed in $O(|\mathcal{R}(t_0)|2^w w)$ time.*

Proof. Informally, $\mathcal{R}(t)$ is obtained from $\mathcal{R}(t_0)$ by restricting α_0 of every record $(\alpha_0, \beta_0) \in \mathcal{R}(t_0)$ to $\text{var}_\chi(t)$, but allowing the assignment that sets v to $\alpha_0(v)$ to satisfy any set of yet unsatisfied constraints in $\beta_0^{-1}(U)$ that have v in their scope. More formally, for every record $(\alpha_0, \beta_0) \in \mathcal{R}(t_0)$ and every subset U' of $\beta_0^{-1}(U) \cap \{\, c \in C \mid v \in \text{scope}(c) \,\}$, the set $\mathcal{R}(t)$ contains the record (α, β), where α is the restriction of α_0 to $\text{var}_\chi(t)$ and β is defined by setting $\beta(c) = \beta_0(c)$ for every $c \in \text{con}_\chi(t) \setminus U'$ and $\beta(c) = (v, \alpha_0(v))$ for every $c \in U'$.

Towards showing the correctness of the definition for $\mathcal{R}(t)$, we first show that every valid record $R = (\alpha, \beta)$ for t is added to $\mathcal{R}(t)$. Because R is valid, there is an assignment $\tau : \text{var}_\chi(T_t) \to CD$ satisfying (R1)–(R3). Let X be the set of all constraints c in $\text{con}_\chi(t)$ such that $\beta(c) = (v, d)$. We know that τ satisfies (R3) so $d = \tau(v)$ for all constraints in X. Then, τ witnesses validity of the record $R_0 = (\alpha_0, \beta_0)$, where α_0 is the extension of α setting v to $\tau(v)$ and β_0 is obtained from β by setting $\beta_0(c) = U$ for every $c \in X$. Now, the record R_0 together with the set $U' = X$ shows that R is added to $\mathcal{R}(t)$.

It remains to show that if a record $R = (\alpha, \beta)$ is added to $\mathcal{R}(t)$, then R is valid for t. Assume that R is obtained from the record $R_0 = (\alpha_0, \beta_0) \in \mathcal{R}(t_0)$. Then, because R_0 is valid for t_0, there is an assignment $\tau_0 : \text{var}_\chi(T_{t_0}) \to CD$ satisfying (R1)–(R3). Moreover, the assignment τ_0 witnesses the validity of R. Finally, the run-time follows because there are at most $|\mathcal{R}(t_0)|2^w$ pairs of a record in $\mathcal{R}(t_0)$ and a subset U' and the time required to compute a record for such a pair is at most $O(w)$. □

Lemma 29 *(constraint forget node). Let $t \in V(T)$ be a forget node with child t_0 such that $\chi(t_0) \setminus \chi(t) = \{c\}$ for some constraint $c \in C$. Then, $\mathcal{R}(t)$ can be computed in $O(|\mathcal{R}(t_0)||\chi(t)|)$ time.*

Proof. Informally, $\mathcal{R}(t)$ is obtained from $\mathcal{R}(t_0)$ by taking all records (α_0, β_0) in $\mathcal{R}(t_0)$ that satisfy c and restricting β_0 to $\text{con}_\chi(t)$. More formally, for every record $(\alpha_0, \beta_0) \in \mathcal{R}(t_0)$ such that $\beta_0(c) = S$, $\mathcal{R}(t)$ contains the record (α_0, β), where β is the restriction of β_0 to $\text{con}_\chi(t)$.

Towards showing the correctness of the definition for $\mathcal{R}(t)$, we first show that every valid record $R = (\alpha, \beta)$ for t is added to $\mathcal{R}(t)$. Because R is valid, there is an assignment $\tau : \text{var}_\chi(T_t) \to CD$ satisfying (R1)–(R3). Because τ satisfies (R1), it also satisfies the constraint c. Therefore the record $R_0 = (\alpha, \beta_0)$, where β_0 is the extension of β to c by setting $\beta_0(c) = S$, is valid and hence $R_0 \in \mathcal{R}(t_0)$. Therefore, R is added to $\mathcal{R}(t)$.

It remains to show that if a record $R = (\alpha, \beta)$ is added to $\mathcal{R}(t)$, then R is valid for t. Assume that R is obtained from the record $R_0 = (\alpha_0, \beta_0) \in \mathcal{R}(t_0)$. Then, because R_0 is valid for t_0, there is an assignment $\tau_0 : \text{var}_\chi(T_{t_0}) \to CD$ satisfying (R1)–(R3). Moreover, from the definition of $\mathcal{R}(t)$, we obtain that τ_0 satisfies c so τ_0 witnesses that R is valid.

Finally, the run-time estimate is correct because it takes $O(|\chi(t)|)$ time to check whether $\beta_0(c) = S$ and to compute the restriction of β to $\text{con}_\chi(t)$ for a record (α_0, β_0) in $\mathcal{R}(t_0)$. □

Lemma 30 *(join node). Let $t \in V(T)$ be a join node with children t_1 and t_2, where $\chi(t) = \chi(t_1) = \chi(t_2)$. Then, $\mathcal{R}(t)$ can be computed in $O(|\mathcal{R}(t_1)||\mathcal{R}(t_2)||\mathcal{I}|)$ time.*

Proof. Informally, $\mathcal{R}(t)$ is obtained from $\mathcal{R}(t_1)$ and $\mathcal{R}(t_2)$ by combining all pairs of records (α_i, β_i) in $\mathcal{R}(t_i)$ that agree on the assignments α_i to a new record and updating the set of satisfied constraints. More formally, we say that two records $(\alpha_1, \beta_1) \in \mathcal{R}(t_1)$ and $(\alpha_2, \beta_2) \in \mathcal{R}(t_2)$ are *compatible* if $\alpha_1 = \alpha_2$ and for every constraint $c \in \text{con}_\chi(t)$ such that for $i \in \{1, 2\}$, $\beta_i(c) = (v_i, d_i)$, and the partial assignment setting v_i to d_i satisfies c. Then, for every pair of compatible records $(\alpha_1, \beta_1) \in \mathcal{R}(t_1)$ and $(\alpha_2, \beta_2) \in \mathcal{R}(t_2)$, the set $\mathcal{R}(t)$ contains the record (α, β), where:

- $\alpha = \alpha_1 = \alpha_2$ and
- $\beta(c) = S$ if either:
  - $\beta_1(c) = S$ or $\beta_2(c) = S$ or
  - $\beta_1(c) = (v_1, d_1)$ and $\beta_2(c) = (v_2, d_2)$ and the (partial) assignment setting v_1 to d_1 and v_2 to d_2 satisfies c.
- $\beta(c) = U$ if $\beta_1(c) = U$ and $\beta_2(c) = U$,
- $\beta(c) = (v, d)$ if either:
  - $\beta_1(c) = (v, d)$ and $\beta_2(c) = U$ or
  - $\beta_1(c) = U$ and $\beta_2(c) = (v, d)$

We will now show the correctness of the definition of $\mathcal{R}(t)$. We first show that every valid record $R = (\alpha, \beta)$ for t is added to $\mathcal{R}(t)$. Because R is valid, there is an assignment $\tau : \text{var}_\chi(T_t) \to CD$ satisfying (R1)–(R3). Let τ_i be the restriction of τ to $\text{var}_\chi(T_{t_i})$. Note first that every constraint $c \in \text{con}_\chi(T_t) \setminus \text{con}_\chi(t)$, is either satisfied by τ_1 or by τ_2. This is because (T, χ) is a tree decomposition so $\text{scope}(c) \subseteq \text{var}_\chi(T_{t_i})$ for some $i \in \{1, 2\}$. Moreover, every constraint $c \in \text{con}_\chi(t)$ that is satisfied by τ is either (1) already satisfied by τ_i (for some $i \in \{1, 2\}$) or (2) it is satisfied by a simple constraint involving two variables $v_i \in \text{var}_\chi(T_i) \setminus \text{var}_\chi(t)$ assigned according to τ. We set $\beta_i(c) = S$ if c is satisfied by τ_i. Otherwise, we set $\beta_i(c) = U$ if $\beta(c) = U$ or $\beta(c) = S$, and τ_{3-i} satisfies c. Finally, we set $\beta_i(c) = (v_i, \tau(v_i))$ if either $\beta(c) = (v_i, \tau(v_i))$ and $v_i \in \text{var}_\chi(T_{t_i}) \setminus \text{var}_\chi(t)$, or $\beta(c) = S$ but neither τ_1 nor τ_2 satisfy c and setting v_i to d_i satisfies c. Then, the records $R_i = (\alpha, \beta_i)$ are valid for t_i as witnessed by τ_i, so $(\alpha, \beta_i) \in \mathcal{R}(t_i)$. Moreover, R_1 and R_2 are compatible and R is the result of combining R_1 and R_2, showing that R is added to $\mathcal{R}(t)$.

It remains to show that if a record $R = (\alpha, \beta)$ is added to $\mathcal{R}(t)$, then R is valid for t. Assume that R is obtained from two compatible records $R_i = (\alpha, \beta_i) \in \mathcal{R}(t_i)$. Then, because R_i is valid for t_i, there is an assignment $\tau_i : \text{var}_\chi(T_{t_i}) \to CD$ satisfying (R1)–(R3). It is now straightforward to verify that the assignment τ obtained by combining τ_1 and τ_2 witnesses that R is valid for t.

Finally, the run-time follows because there are at most $|\mathcal{R}(t_1)||\mathcal{R}(t_2)|$ compatible pairs of records and for every such pair it takes time at most $O(|\mathcal{I}|)$ to compute the combined record for $\mathcal{R}(t)$. $\square$

We can now conclude the results in this section.

Proof of Theorem 22. The algorithm computes the set of all valid records $\mathcal{R}(t)$ for every node t of T using a bottom-up dynamic programming algorithm starting in the leaves of T. It then solves $\mathcal{I}$ by checking whether $\mathcal{R}(r) \neq \emptyset$. The correctness of the algorithm follows from Lemmas 24–30. The run-time of the algorithm is at most the number of nodes of T, which can be assumed to be bounded from above by $O(|\mathcal{I}|)$ (Proposition 20), times the maximum time required to compute $\mathcal{R}(t)$ for any of the node types of a nice tree decomposition, which is obtained for join nodes with a run-time of $O(|\mathcal{R}(t_1)||\mathcal{R}(t_2)||\mathcal{I}|)$. It follows that

$$O((|V||CD| + 2)^{2(w+1)}(|\mathcal{I}|)^2) \in (nk)^{O(w)}$$

is the total run-time because $|\mathcal{R}(t)| \leq (|V||CD| + 2)^{w+1}$. $\square$

6. Parameterized lower bounds

This section contains two main results: we show that CSP($\mathbf{D}_{2,\infty}$) is pNP-hard (Section 6.1) and that CSP($\mathbf{D}_{2,1}$) is W[1]-hard (Section 6.3) when parameterized by primal treewidth. These results indicate that there is no fpt algorithm for CSP($\mathbf{D}_{2,k}$), $k \in \{1, 2, \dots\} \cup \{\infty\}$, under standard complexity-theoretic assumptions. The pNP-hardness result is proved by a direct reduction from Subset Sum while the W[1]-hardness result is based on a reduction from a variant of the Subset Sum problem that we call Multi-Dimensional Partitioned Subset Sum. W[1]-hardness for the latter problem is proved in Section 6.2. The full picture of parameterized upper and lower bounds is summarised in Table 4.

To present our results, we need some additional technical machinery. A parameterized problem is, formally speaking, a subset of $\Sigma^* \times \mathbb{N}$ where Σ is the input alphabet. Reductions between parameterized problems need to take the parameter into account. To this end, we will use *parameterized reductions* (or fpt-reductions). Let L_1 and L_2 denote parameterized problems with $L_1 \subseteq \Sigma_1^* \times \mathbb{N}$ and $L_2 \subseteq \Sigma_2^* \times \mathbb{N}$. A parameterized reduction from L_1 to L_2 is a mapping $P : \Sigma_1^* \times \mathbb{N} \to \Sigma_2^* \times \mathbb{N}$ such that

(1) $(x, k) \in L_1$ if and only if $P((x, k)) \in L_2$,
(2) the mapping can be computed by an fpt-algorithm with respect to the parameter k, and

(3) there is a computable function $g : \mathbb{N} \to \mathbb{N}$ such that for all $(x, k) \in L_1$ if $(x', k') = P((x, k))$, then $k' \leq g(k)$.

The class W[1] contains all problems that are fpt-reducible to Independent Set parameterized by the size of the solution, i.e. the number of vertices in the maximum independent set. Showing W[1]-hardness (by an fpt-reduction) for a problem rules out the existence of a fixed-parameter algorithm under the standard assumption FPT $\neq$ W[1]. The class pNP contains all parameterized problems that can be solved by a nondeterministic algorithm in time $f(k) \cdot \|x\|^{O(1)}$ for some computable function f. It is known that FPT = pNP if and only if P = NP. A problem is pNP-hard if it is NP-hard for a constant value of the parameter.

6.1. pNP-hardness for CSP($\mathbf{D}_{2,\infty}$)

We show that if there is no upper bound on the size of the numbers used in the constraints, then CSP($\mathbf{D}_{2,\infty}$) is NP-hard, even for instances whose primal graph has constant treewidth. In other words, we prove that CSP($\mathbf{D}_{2,\infty}$) is pNP-hard. This result is based on the NP-hard problem Subset Sum [36].

Subset Sum	
Input:	A set of integers S and an integer N.
Question:	Is there a set $S' \subseteq S$ such that $N = \sum_{s \in S'} s$?

Theorem 31. CSP($\mathbf{D}_{2,\infty}$) *is NP-hard, even for instances whose primal graph has treewidth at most* 2.

Proof. We present a polynomial-time reduction from the Subset Sum problem to CSP($\mathbf{D}_{2,\infty}$). Let (S, N) be an instance of Subset Sum with $S = \{s_1, \ldots, s_n\}$. We construct an equivalent instance $\mathcal{I}$ of CSP($\mathbf{D}_{2,\infty}$) as follows. Introduce $n + 1$ variables $x_0, \ldots, x_n$. For every i with $1 \leq i \leq n$, introduce the constraint $x_i - x_{i-1} = 0 \vee x_i - x_{i-1} = s_i$. Finally, add the constraint $x_n - x_0 = N$. Note that the primal graph of $\mathcal{I}$ is a cycle, so its treewidth is at most 2. Given a solution to $\mathcal{I}$, selecting those s_i for which $x_i - x_{i-1} = s_i$ yields a subset of S that sums up to N. In the opposite direction, a solution to $\mathcal{I}$ can be constructed from the subset $S' \subseteq S$ that sums up to N by setting $x_i - x_{i-1} = s_i$ if $s_i \in S'$ and $x_i - x_{i-1} = 0$ otherwise. □

6.2. W[1]-hardness for multi-dimensional partitioned subset sum

Our parameterized hardness result for CSP($\mathbf{D}_{2,k}$) is based on a variant of Subset Sum; we note here that similar but slightly different variants of Subset Sum have been considered before [33,35]. Let k denote a natural number and let $\bar{v}$ denote a vector of dimension $K = \binom{k}{2}$. We sometimes refer to the coordinates of $\bar{v}$ by a pair (a, b) of natural numbers with $1 \leq a < b \leq k$; here, we implicitly use an arbitrary bijection between the K pairs (a, b) satisfying the inequality and the K coordinates of the vector $\bar{v}$. We say that $\bar{v}$ is *uniform* if every non-zero coordinate of $\bar{v}$ has the same value $s(\bar{v})$. Finally, for an integer N, we let $\bar{N}$ denote the K-dimensional vector that is equal to N at every coordinate.

Multi-dimensional Partitioned Subset Sum (MPSS)	
Input:	Integers k and N, and sets $V_1, \ldots, V_k$ and $E_1, \ldots, E_K$ of uniform K-dimensional vectors over the natural numbers such that: • Every vector $\bar{v} \in V_i$ is non-zero at all coordinates (a, b) such that $a = i$ or $b = i$ and zero elsewhere. • Every vector $\bar{v} \in E_r$ is non-zero only at the coordinate r.
Parameter:	k
Question:	Are there $\bar{v}^1, \ldots, \bar{v}^k$ and $\bar{e}^1, \ldots, \bar{e}^K$ with $\bar{v}^i \in V_i$ and $\bar{e}^r \in E_r$ such that $(\sum_{i=1}^{k} \bar{v}^i) + (\sum_{r=1}^{K} \bar{e}^r) = \bar{N}$?

Theorem 32. MPSS *is strongly W[1]-hard (i.e. it is W[1]-hard even if all numbers are encoded in unary).*

Proof. We prove the lemma by a parameterized reduction from Multicoloured Clique, which is well known to be W[1]-complete [67]. Given an integer k and a k-partite graph G with partition $U_1, \ldots, U_k$, the Multicoloured Clique problem asks whether G contains a k-clique (note that since the sets U_i are independent, any k-clique must contain exactly one vertex from each set U_i). We let $W_{(i,j)}$ denote the set of all edges in G with one endpoint in U_i and the other endpoint in U_j, for every i and j with $1 \leq i < j \leq k$. To show the lemma, we construct an instance

$$\mathcal{I} = (k, N, (V_i)_{1 \leq i \leq k}, (E_r)_{1 \leq r \leq K}),$$

of MPSS in polynomial time where all integers in $\mathcal{I}$ are bounded by a polynomial in $|V(G)|$ and $K = \binom{k}{2}$. Our construction yields an instance $\mathcal{I}$ such that G contains a k-clique if and only if $\mathcal{I}$ has a solution.

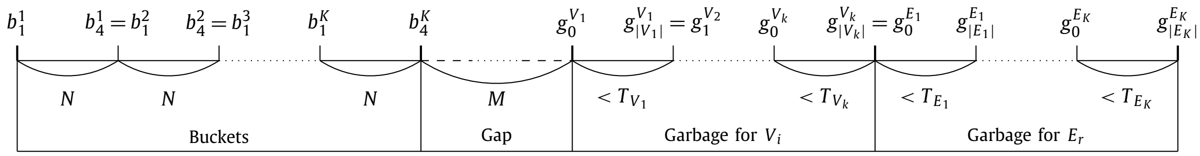

Fig. 3. The board consisting of the bucket part and the garbage part defined in the proof of Theorem 33. Here, $T_A = \sum_{\bar{a} \in A} s(\bar{a})$ for $A \in \{V_1, \ldots, V_k, E_1, \ldots, E_K\}$.

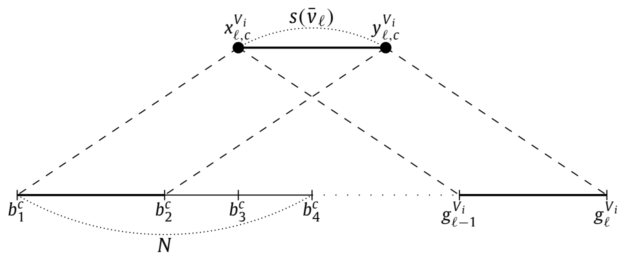

Fig. 4. Possible ways to place the variables $x_{\ell,c}^{V_i}$ and $y_{\ell,c}^{V_i}$ corresponding to the non-zero coordinate $c = (i, j)$ of the ℓ-th vector $\bar{v}_\ell$ in V_i.

We will employ Sidon sets from Section 2.3 in the reduction. Namely, we need a Sidon set containing $|V(G)|$ natural numbers, i.e. one number for each vertex of G. Since the numbers in the Sidon set will be used as numbers in $\mathcal{I}$, we need to ensure that the largest of these numbers is bounded by a polynomial in $|V(G)|$. We know from Section 2.3 that such a set (where the bound on the largest element is $8|V|^2$) can be computed in polynomial time. In the following, we will assume that we are given such a Sidon sequence $\mathcal{S}$ and we let $\mathcal{S}(i)$ denote the i-th element of $\mathcal{S}$ for $1 \le i \le |V(G)|$. Let $\max(\mathcal{S})$ and $\max_2(\mathcal{S})$ denote the largest element of $\mathcal{S}$ and the maximum sum of any two distinct elements in $\mathcal{S}$, respectively. We will furthermore assume that the vertices of G are identified with the numbers from 1 to $|V(G)|$ and therefore $\mathcal{S}(v)$ is properly defined for every $v \in V(G)$.

We are now ready to construct the instance $\mathcal{I}$. We set $N = \max_2(\mathcal{S}) + 1$ and proceed to the construction of the sets $V_1, \ldots, V_k$ and the sets $E_1, \ldots, E_K$. For every i with $1 \le i \le k$ and every $u \in U_i$, the set V_i contains the vector $\bar{u}$ with $s(\bar{u}) = \mathcal{S}(u)$ being non-zero at all coordinates (a, b) such that either $a = i$ or $b = i$. Moreover, for every $1 \le r \le K$ and every $e = (u, v) \in W_r$, the set E_r contains the vector $\bar{e}$ with $s(\bar{e}) = (\max_2(\mathcal{S}) + 1) - (\mathcal{S}(u) + \mathcal{S}(v))$, and the non-zero value appearing only at coordinate r.

This completes the construction of $\mathcal{I}$. It is clear that $\mathcal{I}$ can be constructed in polynomial time and that every integer in $\mathcal{I}$ is at most $\max_2(\mathcal{S}) + 1$ so $\mathcal{I}$ is polynomially bounded in $|V(G)|$. Intuitively, the construction relies on the fact that since the sum of each pair of vertices is unique, we can uniquely associate each pair with an edge between these vertices, whose value will then be the global upper bound of $\max_2(\mathcal{S}) + 1$ minus the unique sum.

It remains to show that G contains a k-clique if and only if $\mathcal{I}$ has a solution.

Forward direction. Let C be a k-clique in G with vertices $u_1, \ldots, u_k$ such that $u_i \in U_i$ for every i with $1 \le i \le k$. Choose the vector $\bar{u}_i$ from V_i, $1 \le i \le k$, and the vector $\bar{e}_{(i,j)}$ from $E_{(i,j)}$, where $\bar{e}_{(i,j)}$ is the edge with endpoints u_i and u_j for every i and j with $1 \le i < j \le k$. We claim that this choice is a solution for $\mathcal{I}$. Let $\bar{t}$ be the vector $(\sum_{i=1}^{k} \bar{u}_i) + (\sum_{i=1}^{K} \bar{e}_i)$. For every coordinate (i, j) with $1 \le i < j \le k$, the vectors $\bar{u}_i$, $\bar{u}_j$, and $\bar{e}_{(i,j)}$ are the only vectors that are non-zero at the coordinate (i, j). Therefore, $\bar{t}[(i, j)] = s(\bar{u}_i) + s(\bar{u}_j) + s(\bar{e}_{(i,j)})$. Moreover, using the identities $s(\bar{u}_i) = \mathcal{S}(u_i)$, $s(\bar{u}_j) = \mathcal{S}(u_j)$, and $s(\bar{e}_{(i,j)}) = (\max_2(\mathcal{S}) + 1) - (\mathcal{S}(u_i) + \mathcal{S}(u_j))$, we obtain that

$$\bar{t}[(i, j)] = \mathcal{S}(u_i) + \mathcal{S}(u_j) + (\max_2(\mathcal{S}) + 1) - (\mathcal{S}(u_i) + \mathcal{S}(u_j)) = \max_2(\mathcal{S}) + 1 = N,$$

as required.

Backward direction. Assume $\bar{u}_i \in V_i$, $1 \le i \le k$ together with $\bar{e}_j \in E_j$, $1 \le j \le K$ is a solution for $\mathcal{I}$. We claim that $\{u_1, \ldots, u_k\}$ forms a k-clique in G, i.e. $e_{(i,j)} = \{u_i, u_j\}$ is an edge of G for every i and j with $1 \le i < j \le k$. Note that the only vectors in the solution for $\mathcal{I}$ that have a non-zero contribution towards the (i, j)-th coordinate of the sum vector are the vectors $\bar{u}_i$, $\bar{u}_j$, and $\bar{e}_{i,j}$. Since $s(\bar{u}_i) = \mathcal{S}(u_i)$, $\bar{u}_j = \mathcal{S}(u_j)$, and $N = \max_2(\mathcal{S}) + 1$, we see that $s(\bar{e}_{(i,j)}) = (\max_2(\mathcal{S}) + 1) - (\mathcal{S}(u_i) + \mathcal{S}(u_j))$. Moreover, $\mathcal{S}$ is a Sidon sequence so the sum $(\mathcal{S}(u_i) + \mathcal{S}(u_j))$ is unique. It follows that $e_{(i,j)} = \{u_i, u_j\}$ as required. □

6.3. *W[1]-hardness for* CSP($\mathbf{D}_{2,1}$)

We provide a parameterized reduction from MPSS, which together with Theorem 32 establishes the result. To simplify the reduction, we provide it in two stages. First we show how to construct an equivalent instance $\mathcal{I}''$ of CSP($\mathbf{D}_2$) and then we show how to obtain the desired instance $\mathcal{I}'$ of CSP($\mathbf{D}_{2,1}$) from $\mathcal{I}''$.

Before giving the formal proof in Theorem 33, let us provide an informal overview of the main ideas behind the proof. Let $\mathcal{I} = (k, N, (V_i)_{1\le i\le k}, (E_r)_{1\le r\le K})$ be an instance of MPSS. We let our playing board be the real line; please refer to Fig. 3 for an illustration. First, for every vector $\bar{v}$ in $\text{VE} = (\bigcup_{i=1}^{k} V_i) \cup (\bigcup_{i=1}^{K} E_i)$ and every non-zero coordinate c of $\bar{v}$, we introduce a segment on the board represented by two variables x and y at distance exactly $s(\bar{v})$ from each other. We divide our board into two main parts: the *bucket* part and the *garbage* part. While the bucket part provides placeholders for the segments of the vectors chosen to be in a solution for $\mathcal{I}$, the garbage part provides placeholders for all other segments. Crucial for the idea is a gadget that ensures that a segment can only be in one of two places, i.e. either its place inside the bucket part or its place inside the garbage part. To illustrate the idea behind this gadget, suppose one wants to ensure that a variable x is either equal to a variable a or equal to a variable b. This can be achieved by the ternary constraint $x = a \vee x = b$. However, since we are only allowed to use binary constraints, it becomes more complicated. The idea is that we additionally ensure that the distance between a and b is between M and $2M - 1$ for some number M. Then we can ensure that x is either equal to a or equal to b by using the constraints $x = a \vee x - a \ge M$ and $x = b \vee b - x \ge M$.

With this in mind, let us provide some details on the bucket part and the garbage part. The main idea behind the bucket part is that it provides placeholders for the segments representing the non-zero coordinates of all vectors that are in the solution for $\mathcal{I}$. More specifically, consider a solution for $\mathcal{I}$ choosing exactly one vector $\bar{v}^i$ from each V_i and exactly one vector $\bar{e}^r$ from each E_r. Then for every coordinate $r = (i, j)$, the solution contains exactly three vectors that are non-zero at coordinate r, i.e. the vector $\bar{v}^i$, the vector $\bar{v}^j$, and the vector $\bar{e}^r$. Thus, the bucket part will provide three placeholders. This is achieved by introducing four variables $b^r_1, \dots, b^r_4$ for every coordinate r with the idea that, the place between b^r_1 and b^r_2 is a placeholder for the r-th coordinate of $\bar{v}^i$, the place between b^r_2 and b^r_3 is a placeholder the r-th coordinate of $\bar{v}_j$, and the place between b^r_3 and b^r_4 is a placeholder for the r-th coordinate of $\bar{e}_r$. Finally, to verify that the sum of all vectors in the solution is equal to N at each coordinate r, we introduce the constraint $b^r_4 - b^r_1 = N$.

The main function of the garbage part is to ensure two things: (1) if a segment representing a non-zero coordinate of some vector $\bar{v}$ in VE is chosen to be in the bucket part, then all segments representing non-zero coordinates of $\bar{v}$ are chosen to be in the bucket part and (2) the segments of at least one vector from every set V_i and every set E_r are chosen to be in the bucket part. To achieve this, the garbage part consists of $k + K$ parts, i.e. one part for every set V_i and one part for every set E_r. Moreover, the part for a set $A \in \{V_1, \dots, V_k, E_1, \dots, E_K\}$, has one placeholder for every vector $\bar{a} \in A$, which can hold all segments representing non-zero coordinates of the vector $\bar{a}$. This is achieved by introducing $|A| + 1$ variables $g^A_0, \dots, g^A_{|A|}$ such that the place between g^A_{i-1} and g^A_i is reserved to hold all segments of the i-th vector in A. Here, it is important to recall that every non-zero coordinate of every vector $\bar{v}$ in VE has the same value $s(\bar{v})$. Finally, we ensure (2) by adding the constraint $g^A_{|A|} - g^A_0 < \sum_{\bar{a}\in A} s(\bar{a}) = T_A$, which ensures that not all vectors of A can fit into the garbage part. We are now ready to provide the formal proof.

Theorem 33. CSP($\mathbf{D}_{2,1}$) *is strongly W[1]-hard parameterized by primal treewidth.*

Proof. Let $\mathcal{I} = (k, N, (V_i)_{1\le i\le k}, (E_r)_{1\le r\le K})$ be an arbitrary instance of MPSS. We start by introducing the board consisting of the bucket part and the garbage part; see Fig. 3 for an illustration.

The bucket part. For every coordinate r (of the vectors in $\mathcal{I}$), where $1 \le r \le K$, we introduce 4 *bucket variables* $b^r_1, \dots, b^r_4$ and ensure that these appear consecutively in any solution by using the constraints $b^r_{l+1} - b^r_l \ge 0$ for every l with $1 \le l < 4$. We introduce a constraint that ensures that the distance between b^r_1 and b^r_4 is exactly N, i.e. the constraint $b^r_4 - b^r_1 = N$. Finally, we arrange the bucket variables for different coordinates in the natural order by introducing the constraints $b^r_4 = b^{r+1}_1$ for every r with $1 \le r < K$.

The garbage part (for the sets V_i). For every set $V_i = \{\bar{v}_1, \dots, \bar{v}_{|V_i|}\}$, we introduce $|V_i| + 1$ *garbage variables* $g^{V_i}_0, \dots g^{V_i}_{|V_i|}$ and ensure that these appear consecutively in any solution by using the constraints $g^{V_i}_{l+1} - g^{V_i}_l \ge 0$ for every l with $0 \le l < |V_i|$. We introduce a constraint that ensures that the distance between $g^{V_i}_0$ and $g^{V_i}_{|V_i|}$ is smaller than $T = \sum_{\bar{v}\in V_i} s(\bar{v})$, i.e. the constraint $g^{V_i}_{|V_i|} - g^{V_i}_0 < T$. Additionally, we arrange the garbage variables for different sets V_i in the natural order by introducing the constraints $g^{V_{i+1}}_0 = g^{V_i}_{|V_i|}$ for every i with $1 \le i < k$.

The garbage part (for the sets E_r). For every set $E_i = \{\bar{v}_1, \dots, \bar{v}_{|E_i|}\}$, we introduce $|E_i| + 1$ *garbage variables* $g^{E_i}_0, \dots g^{E_i}_{|E_i|}$ and ensure that these appear consecutively in any solution by using the constraints $g^{E_i}_{l+1} - g^{E_i}_l \ge 0$ for every l with $0 \le l < |E_i|$. We introduce a constraint that ensures that the distance between $g^{E_i}_0$ and $g^{E_i}_{|E_i|}$ is smaller than $T = \sum_{\bar{v}\in E_i} s(\bar{v})$, i.e. the constraint $g^{E_i}_{|E_i|} - g^{E_i}_0 < T$. Additionally, we arrange the garbage variables for different sets E_i in the natural order by introducing the constraints $g^{E_{i+1}}_0 = g^{E_i}_{|E_i|}$ for every i with $1 \le i < K$.

We ensure that the garbage variables of the sets E_i are placed after the garbage variables of the sets V_i by adding the constraint $g^{V_k}_{|V_k|} = g^{E_1}_0$. Finally, to make the later arguments simpler, we make sure that the last bucket variable b^K_4 has sufficient distance to the first garbage variable $g^{|V_i|}_0$. We let $M = K \cdot N + \sum_{\bar{v} \in \mathrm{VE}} s(\bar{v})$ and add the constraint $g^{|V_1|}_0 - b^K_4 = M$.

The vector variables for the sets V_i**.** For every set $V_i = \{\bar{v}_1, \ldots, \bar{v}_{|V_i|}\}$, every ℓ with $1 \le \ell \le |V_i|$, and every non-zero coordinate c of $\bar{v}_\ell$, we introduce two variables $x^{V_i}_{\ell,c}$ and $y^{V_i}_{\ell,c}$ and the constraint $y^{V_i}_{\ell,c} - x^{V_i}_{\ell,c} = s(\bar{v}_\ell)$ ensuring that the distance between $y^{V_i}_{\ell,c}$ and $x^{V_i}_{\ell,c}$ is exactly $s(\bar{v}_\ell)$. We associate one bucket variable, denoted by $B(x^{V_i}_{\ell,c})$ and $B(y^{V_i}_{\ell,c})$, respectively, and one garbage variable, denoted by $G(x^{V_i}_{\ell,c})$ and $G(x^{V_i}_{\ell,c})$, respectively, with $x^{V_i}_{\ell,c}$ and $y^{V_i}_{\ell,c}$ as follows. If $c = (i, j)$ (for some $j > i$), we set $B(x^{V_i}_{\ell,c}) = b^c_1$ and $B(y^{V_i}_{\ell,c}) = b^c_2$. Otherwise, i.e. if $c = (j, i)$ (for some $j < i$), we set $B(x^{V_i}_{\ell,c}) = b^c_2$ and $B(y^{V_i}_{\ell,c}) = b^c_3$. Moreover, we set $G(x^{V_i}_{\ell,c}) = g^c_{\ell-1}$ and $G(y^{V_i}_{\ell,c}) = g^c_\ell$. We add constraints that ensure that $x^{V_i}_{\ell,c}$ is either equal to $B(x^{V_i}_{\ell,c})$ or $G(x^{V_i}_{\ell,c})$ (see Fig. 4). As we will show later this can be guaranteed by the constraints:

- $x^{V_i}_{\ell,c} = B(x^{V_i}_{\ell,c}) \vee x^{V_i}_{\ell,c} - B(x^{V_i}_{\ell,c}) \ge M$ and
- $x^{V_i}_{\ell,c} = G(x^{V_i}_{\ell,c}) \vee G(x^{V_i}_{\ell,c}) - x^{V_i}_{\ell,c} \ge M$.

Similarly, we add constraints so that $y^{V_i}_{\ell,c}$ is either equal to $B(y^{V_i}_{\ell,c})$ or $G(y^{V_i}_{\ell,c})$, i.e. we add the constraints:

- $y^{V_i}_{\ell,c} = B(y^{V_i}_{\ell,c}) \vee y^{V_i}_{\ell,c} - B(y^{V_i}_{\ell,c}) \ge M$ and
- $y^{V_i}_{\ell,c} = G(y^{V_i}_{\ell,c}) \vee G(y^{V_i}_{\ell,c}) - y^{V_i}_{\ell,c} \ge M$.

Let XY_V denote the set of all vector variables for the sets V_i, i.e. the set

$$\{x^{V_i}_{\ell,c}, y^{V_i}_{\ell,c} \mid 1 \le i \le k \wedge 1 \le \ell \le |V_i| \wedge c \text{ is a non-zero coordinate of } \bar{v}^i_\ell\}.$$

The Vector Variables for the sets E_i**.** For every set $E_i = \{\bar{e}_1, \ldots, \bar{e}_{|E_i|}\}$ and every ℓ with $1 \le \ell \le |E_i|$, we introduce two variables $x^{E_i}_\ell$ and $y^{E_i}_\ell$ and the constraint $y^{E_i}_\ell - x^{E_i}_\ell = s(\bar{e}_\ell)$ ensuring that the distance between $y^{E_i}_\ell$ and $x^{E_i}_\ell$ is exactly $s(\bar{e}_\ell)$. Similarly, to the vector variables for the sets V_i, we associate a bucket variable and a garbage variable with $x^{E_i}_\ell$ and $y^{E_i}_\ell$, defined by setting: $B(x^{E_i}_\ell) = b^i_3$, $G(x^{E_i}_\ell) = g^{E_i}_{\ell-1}$, $B(y^{E_i}_\ell) = b^i_4$, and $G(y^{E_i}_\ell) = g^{E_i}_\ell$. We add constraints that ensure that $x^{E_i}_\ell$ is either equal to $B(x^{E_i}_\ell)$ or $G(x^{E_i}_\ell)$. As we will show later, this is guaranteed by the constraints:

- $x^{E_i}_\ell = B(x^{E_i}_\ell) \vee x^{E_i}_\ell - B(x^{E_i}_\ell) \ge M$ and
- $x^{E_i}_\ell = G(x^{E_i}_\ell) \vee G(x^{E_i}_\ell) - x^{E_i}_\ell \ge M$.

Finally, we add constraints that imply that $y^{E_i}_\ell$ is either equal to $B(y^{E_i}_\ell)$ or $G(y^{E_i}_\ell)$:

- $y^{E_i}_\ell = B(y^{E_i}_\ell) \vee y^{E_i}_\ell - B(y^{E_i}_\ell) \ge M$ and
- $y^{E_i}_\ell = G(y^{E_i}_\ell) \vee G(y^{E_i}_\ell) - y^{E_i}_\ell \ge M$.

We will verify that these constraints have the required properties later on. Let XY_E denote the set of all vector variables for the sets E_i, i.e. the set

$$\{x^{E_i}_\ell, y^{E_i}_\ell \mid 1 \le i \le K \wedge 1 \le \ell \le |V_i|\}$$

and let $\mathrm{XY} = \mathrm{XY}_V \cup \mathrm{XY}_E$.

This completes the construction of the instance $\mathcal{I}''$ of CSP($\mathbf{D}_2$). We first show that the primal treewidth of $\mathcal{I}''$ is at most $4K + 3$ and consequently bounded by a function of the parameter k only. Let $B = \{b^i_l : 1 \le i \le K \wedge 1 \le l \le 4\}$ be the set of all $4K$ bucket variables and let G be the primal graph of $\mathcal{I}''$ after removing the variables in B. It is straightforward to verify that G has treewidth at most 3 and we obtain, from Proposition 21, that the primal graph of $\mathcal{I}''$ has treewidth at most $|B| + 3 = 4K + 3$.

We now show the equivalence of the instances $\mathcal{I}$ and $\mathcal{I}''$.

Forward direction. Let $\bar{v}^1_{i_1}, \ldots, \bar{v}^k_{i_k}$ and $\bar{e}^1_{j_1}, \ldots, \bar{e}^K_{j_K}$ with $\bar{v}^\ell_{i_\ell} \in V_\ell$ and $\bar{e}^\ell_{j_\ell} \in E_\ell$ be a solution for $\mathcal{I}$. Informally, the main idea to obtain a solution for $\mathcal{I}''$ is to set the variables $x^{V_\ell}_{i_\ell,c}$ and $y^{V_\ell}_{i_\ell,c}$ equal to their respective bucket variables, i.e. the variables $B(x^{V_\ell}_{i_\ell,c})$ and $B(y^{V_\ell}_{i_\ell,c})$, and similarly for the variables $x^{E_\ell}_{j_\ell}$ and $y^{E_\ell}_{j_\ell}$. All other variables in XY are then set to be equal to their respective garbage variables. Since $\bar{v}^1_{i_1}, \ldots, \bar{v}^k_{i_k}$ and $\bar{e}^1_{j_1}, \ldots, \bar{e}^K_{j_K}$ is a solution for $\mathcal{I}$ (and $\sum_{\ell=1}^{k} \bar{v}^\ell_{i_\ell} + \sum_{\ell=1}^{K} \bar{e}^\ell_{j_\ell} = \bar{N}$), this ensures that the distance between b_1^r and b_4^r is exactly N for every coordinate/bucket r and the distance between g_0^A and $g_{|A|}^A$ is less than the sum of all vectors in the set $A \in \{V_1, \ldots, V_k, E_1, \ldots, E_K\}$.

More formally, we set $x_{i_\ell,c}^{V_\ell}$ and $y_{i_\ell,c}^{V_\ell}$ equal to $B(x_{i_\ell,c}^{V_\ell})$ and $B(y_{i_\ell,c}^{V_\ell})$, respectively, for every ℓ with $1 \leq \ell \leq k$ and every non-zero coordinate c of $\bar{v}_{i_\ell}^\ell$. Similarly, we set $x_{j_\ell}^{E_\ell}$ and $y_{j_\ell}^{E_\ell}$ equal to $B(x_{j_\ell}^{E_\ell})$ and $B(y_{j_\ell}^{E_\ell})$, respectively, for every ℓ with $1 \leq \ell \leq K$. For every other variable v in XY, we set v equal to $G(v)$. Finally, we set $g_{i_\ell-1}^{V_\ell}$ equal to $g_{i_\ell}^{V_\ell}$ for every ℓ with $1 \leq \ell \leq k$ and $g_{j_\ell-1}^{E_\ell}$ equal to $g_{j_\ell}^{E_\ell}$ for every ℓ with $1 \leq \ell \leq k$. Note that because of the distances between the variables in XY, this already fixes the position (value) of each variable (up to an additive constant). Note also that all constraints are satisfied. In particular, the constraints $b_4^c - b_1^c = N$ are satisfied because $\sum_{\ell=1}^k \bar{v}_{i_\ell}^\ell + \sum_{\ell=1}^K \bar{e}_{j_\ell}^\ell = \bar{N}$. Similarly, for every set $A \in \{V_1, \ldots, V_k, E_1, \ldots, E_K\}$ the constraints $g_{|A|}^A - g_0^A < \sum_{\bar{v} \in A} s(\bar{v})$ are satisfied since $g_{|A|}^A - g_0^A = (\sum_{\bar{v} \in A \setminus C} s(\bar{v}))$ and $A \cap C \neq \emptyset$, where $C = \{\bar{v}_{i_1}^1, \ldots, \bar{v}_{i_k}^k, \bar{e}_{j_1}^1, \ldots, \bar{e}_{j_K}^K\}$.

Backward direction. Let α be an arbitrary solution to $\mathcal{I}''$. We start by showing the following claim.

Claim 33.1. *For every $v \in$ XY either $\alpha(v) = \alpha(B(v))$ or $\alpha(v) = \alpha(G(v))$.*

Proof of claim: Suppose to the contrary that this is not the case. Because $v \in$ XY, v appears in the two constraints:

- $v = B(v) \vee v - B(v) \geq M$ and
- $v = G(v) \vee G(v) - v \geq M$.

Thus, $\alpha(v) - \alpha(B(v)) \geq M$ and $\alpha(G(v)) - \alpha(v) \geq M$. However, this is only possible if $\alpha(G(v)) - \alpha(B(v)) \geq 2M$, which, as we will show now, is not the case. It follows from the relation between the bucket and garbage variables and the definition of $B(v)$ and $G(v)$ that $\alpha(B(v)) \geq b_1^1$ and $\alpha(G(v)) \leq g_{|E_K|}^{E_K}$. Hence, $\alpha(G(v)) - \alpha(B(v)) \leq \alpha(g_{|E_K|}^{E_K}) - \alpha(b_1^1)$. Because of the constraints on the bucket variables and the garbage variables, i.e. the constraints:

- $b_4^i - b_1^i = N$ for every i with $1 \leq i \leq K$,
- $b_1^{i+1} = b_4^i$ for every i with $1 \leq i < K$,
- $g_{|A|}^A - g_0^A < \sum_{\bar{v} \in A} s(\bar{v})$ for every $A \in \{V_1, \ldots, V_k, E_1, \ldots, E_K\}$,
- $g_0^{V_{i+1}} = g_{|V_i|}^{V_i}$ for every i with $1 \leq i \leq k$,
- $g_0^{E_{i+1}} = g_{|E_i|}^{E_i}$ for every i with $1 \leq i \leq K$,
- $g_{|V_k|}^{V_k} = g_0^{E_1}$,
- $g_0^{V_1} - b_4^K = M$.

we obtain that:

$$
\begin{aligned}
\alpha(G(v)) - \alpha(B(v)) &\leq \alpha(g_{|E_K|}^{E_K}) - \alpha(b_1^1) \\
&< NK + M + \sum_{\bar{v} \in S} s(\bar{v}) \\
&\leq NK + \sum_{\bar{v} \in S} s(\bar{v}) + M \\
&\leq 2M
\end{aligned}
$$

We see that $\alpha(G(v)) - \alpha(B(v)) < 2M$. This contradiction concludes the proof of Claim 33.1. ◇

We say that a vector $\bar{v}_\ell^i \in V_i$ is in the bucket if $\alpha(x_{\ell,c}^{V_i}) = \alpha(B(x_{\ell,c}^{V_i}))$ and $\alpha(y_{\ell,c}^{V_i}) = \alpha(B(y_{\ell,c}^{V_i}))$ for every non-zero coordinate c of $\bar{v}_\ell^i$. Moreover, we say that $\bar{v}_\ell^i \in V_i$ is in the garbage if $\alpha(x_{\ell,c}^{V_i}) = \alpha(G(x_{\ell,c}^{V_i}))$ and $\alpha(y_{\ell,c}^{V_i}) = \alpha(G(y_{\ell,c}^{V_i}))$ for every non-zero coordinate c of $\bar{v}_\ell^i$. Similarly, we say that a vector $\bar{e}_\ell^i \in E_i$ is in the bucket if $\alpha(x_\ell^{E_i}) = \alpha(B(x_\ell^{E_i}))$ and $\alpha(y_\ell^{E_i}) = \alpha(B(y_\ell^{E_i}))$ and we say that $\bar{e}_\ell^i \in E_i$ is in the garbage if $\alpha(x_\ell^{E_i}) = \alpha(G(x_\ell^{E_i}))$ and $\alpha(y_\ell^{E_i}) = \alpha(G(y_\ell^{E_i}))$.

Based on Claim 33.1, we will now show that for every set V_i and every set E_i exactly one vector is in the bucket and all other vectors (of the set) are in the garbage. We start by showing the claim for the sets V_i.

Claim 33.2. *For every V_i there is exactly one vector $\bar{v}_\ell^i \in V_i$ such that $\bar{v}_\ell^i$ is in the bucket, and all other vectors in V_i are in the garbage.*

Proof of claim: We first show that at least one vector $\bar{v}_\ell^i$ is in the bucket. Suppose to the contrary that this is not the case. Claim 33.1 implies that for every ℓ there is a non-zero coordinate c of $\bar{v}_\ell^i$ such that $\alpha(x_{\ell,c}^{V_i}) = \alpha(G(x_{\ell,c}^{V_i}))$ and, consequently,

$\alpha(y^{V_i}_{\ell,c}) = \alpha(G(y^{V_i}_{\ell,c}))$. Thus, the distance between $g^{V_i}_{\ell-1}$ and $g^{V_i}_{\ell}$ is exactly $s(\bar{v}^i_\ell)$ so the distance between $g^{V_i}_0$ and $g^{V_i}_{|V_i|}$ equals $\sum_{\bar{v}\in V_i} s(\bar{v})$. This violates the constraint $g^{V_i}_{|V_i|} - g^{V_i}_0 < \sum_{\bar{v}\in V_i} s(\bar{v})$ and consequently contradicts our assumption that α is a solution for $\mathcal{I}''$. We conclude that there is at least one vector $\bar{v}^i_\ell$ in the bucket.

It remains to show that all other vectors $\bar{v}^i_{\ell'} \in V_i$ for $\ell' \neq \ell$ are in the garbage, i.e. $\alpha(x^{V_i}_{\ell',c}) = \alpha(G(x^{V_i}_{\ell',c}))$ and $\alpha(y^{V_i}_{\ell',c}) = \alpha(G(y^{V_i}_{\ell',c}))$ for every $\ell' \neq \ell$ and every non-zero component c of $\bar{v}^i_{\ell'}$. Suppose this is not the case and assume that the claim is violated for ℓ' and c. Then, by Claim 33.1, it follows that $\alpha(x^{V_i}_{\ell',c}) = \alpha(B(x^{V_i}_{\ell',c}))$ (and therefore also $\alpha(y^{V_i}_{\ell',c}) = \alpha(B(y^{V_i}_{\ell',c}))$). This implies that the distance between $B(x^{V_i}_{\ell',c})$ and $B(y^{V_i}_{\ell',c})$ is equal to $s(\bar{v}^i_{\ell'})$. However, since $\alpha(x^{V_i}_{\ell,c}) = \alpha(B(x^{V_i}_{\ell,c}))$ and $\alpha(y^{V_i}_{\ell,c}) = \alpha(B(y^{V_i}_{\ell,c}))$, we obtain that the distance between $B(x^{V_i}_{\ell',c}) = B(x^{V_i}_{\ell,c})$ and $B(y^{V_i}_{\ell',c}) = B(y^{V_i}_{\ell,c})$ is equal to $s(\bar{v}^i_\ell)$. However, this is not possible because no two vectors in V_i agree on $s(\bar{v})$ and hence $s(\bar{v}^i_{\ell'}) \neq s(\bar{v}^i_\ell)$. $\diamond$

An analogous proof shows the statement of Claim 33.2 for the sets E_i (instead of the sets V_i).

Claim 33.3. *For every E_i there is exactly one vector $\bar{e}^i_\ell \in E_i$ such that $\bar{e}^i_\ell$ is in the bucket, and all other vectors in E_i are in the garbage.*

We continue by showing that the vectors that are in the bucket form a solution for $\mathcal{I}$. Let $\bar{v}^i \in V_i$ and $\bar{e}^j \in E_j$ with $1 \leq i \leq k$ and $1 \leq j \leq K$ be the vectors that are in the bucket; these vectors exist due to Claims 33.2 and 33.3. The constraints $b^l_4 - b^l_1 = N$ imply that

$$\left(\sum_{i=1}^{k} \bar{v}^i\right) + \left(\sum_{i=1}^{K} \bar{e}^i\right) = \bar{N}$$

for every $1 \leq l \leq K$. Hence, the vectors $\bar{v}^1, \ldots, \bar{v}^k$ and $\bar{e}^1, \ldots, \bar{e}^K$ indeed form a solution for $\mathcal{I}$.

The proof so far shows that $\mathcal{I}$ has a solution if and only if $\mathcal{I}''$ has a solution. In the final part of the proof, we show how to transform the instance $\mathcal{I}''$ into the equivalent instance $\mathcal{I}'$ of CSP($\mathbf{D}_{2,1}$). To achieve this we first replace every constraint of the form $a - b \odot n$ (for variables a and b, natural number n with $n > 1$, and $\odot \in \{\leq, <, =, \geq, >\}$) by a 'path' on n auxiliary variables. More formally, to replace the constraint $C = a - b \odot n$, we add n auxiliary variables $h^C_1, \ldots, h^C_n$ and the constraints $h_1 - b \odot 1$, $h^C_{i+1} - h^C_i \odot 1$ for every i with $1 \leq i \leq n$, and $h^C_n = b$. This allows us to replace the following constraints of $\mathcal{I}''$:

- the constraints $b^i_4 - b^i_1 = N$ for every i with $1 \leq i \leq K$,
- the constraints $g^A_{|A|} - g^A_0 < \sum_{\bar{v}\in A} s(\bar{v})$ for every $A \in \{V_1, \ldots, V_k, E_1, \ldots, E_K\}$,
- the constraint $g^{V_1}_0 - b^K_4 = M$,
- the constraints $y^{E_i}_\ell - x^{E_i}_\ell = s(\bar{e}^i_\ell)$, for every i and ℓ with $1 \leq i \leq K$ and $1 \leq \ell \leq |E_i|$, and
- the constraints $y^{V_i}_{\ell,c} - x^{V_i}_{\ell,c} = s(\bar{v}^i_\ell)$, for every i and ℓ with $1 \leq i \leq k$, $1 \leq \ell \leq |E_i|$, and every non-zero component c of $\bar{v}^i_\ell$.

Note that this reduction is polynomial since the numbers in the instance $\mathcal{I}$ can be assumed to be polynomially bounded in the input size because MPSS is strongly W[1]-hard. Also note that this replacement does not increase the treewidth of the primal graph by more than 1 since the new primal graph can be obtained by subdividing edges of the original primal graph and it is well known that subdividing edges can only increase the treewidth of a graph by at most 1. Let $\mathcal{I}'''$ be the instance obtained from $\mathcal{I}''$ after replacing all of the constraints as described above.

It now only remains to replace the remaining constraints for the variables in XY. Recall that these constraints are of the form $z = B(z) \vee z - B(z) \geq M$ and $z = G(z) \vee G(z) - z \geq M$ for some $z \in$ XY. As we saw in Claim 33.1, the effect of these two constraints is that for every variable $z \in$ XY and every solution α for $\mathcal{I}''$, either $\alpha(z) = \alpha(B(z))$ or $\alpha(z) = \alpha(G(z))$. To replace these constraints, we first introduce a gadget $U(a, b, Z)$, where a and b are variables and Z is a natural number, which ensures that either $a = b$ or $b - a \geq Z$. The gadget $U(a, b, Z)$ has $2Z + 1$ auxiliary variables $h_0, \ldots, h_{2Z}$ and the following constraints:

(C1) $h_0 = a$, $h_{i+1} - h_i \in [0, 1]$ for every i with $0 \leq i < 2Z$, and $h_{2Z} = b$,
(C2) $h_{i+2} - h_i = 0 \vee h_{i+2} - h_i > 1$ for every i with $0 \leq i < 2Z - 1$.

Claim 33.4. *Consider the instance $U(a, b, Z)$ for variables a and b and natural number Z. Then:*

- *every solution β for $U(a, b, Z)$ satisfies either $\beta(a) = \beta(b)$ or $\beta(b) - \beta(a) \in (Z, 2Z]$, and*
- *for every number $Z' \in \{0\} \cup (Z, 2Z]$, there is a solution β for $U(a, b, Z)$ such that $\beta(b) - \beta(a) = Z'$.*

Proof of claim: Let β be a solution for $U(a, b, Z)$. The constraints in (C1) imply that $\beta(b) - \beta(a) \in [0, 2Z]$. If $\beta(a) = \beta(b)$, then there is nothing to show. Hence, assume that $\beta(a) < \beta(b)$. Now, there is an i with $0 \leq i < 2Z - 1$ such that $h_{i+2} - h_i > 0$.

We first show that $h_{i+2} - h_i > 0$ for every i with $0 \le i < 2Z - 1$. Suppose that this is not the case and let i be an index such that $h_{i+2} - h_i > 0$ but either $h_{i+3} - h_{i+1} = 0$ or $h_{i+1} - h_{i-1} = 0$. In both cases it follows from the constraints in (C2) that $h_{i+2} - h_i > 1$. Consequently, the constraints in (C1) imply that $h_{i+1} - h_i > 0$ and $h_{i+2} - h_{i+1} > 0$. However, this implies that $h_{i+3} - h_{i+1} > 0$ (since $i < 2Z - 2$) and $h_{i+1} - h_{i-1} > 0$ (since $i > 0$) so we obtain a contradiction. Hence, $h_{i+2} - h_i > 0$ for every i with $0 \le i < 2Z - 1$, which together with the constraints in (C2) implies that $h_{i+2} - h_i > 1$ for every i with $0 \le i < 2Z - 1$. Since

$$\begin{aligned}\beta(b) - \beta(a) &\ge \sum_{i=1}^{Z} \beta(h_{2i}) - \beta(h_{2i-2}) \\ &> \sum_{i=1}^{Z} 1 \\ &= Z\end{aligned}$$

it follows that $\beta(b) - \beta(a) > Z$. This completes the proof of the first statement of the claim.

We continue with the second statement of the claim. Arbitrarily choose $Z' \in \{0\} \cup (Z, 2Z]$. If $Z' = 0$, then we set

$$\beta(a) = \beta(h_0) = \beta(h_1) = \cdots = \beta(h_{2Z-1}) = \beta(h_{2Z}) = \beta(b)$$

and this assignment clearly satisfies all constraints in (C1) and (C2). If $Z' \in (Z, 2Z)$, then we set $\beta(a) = \beta(h_0)$, $\beta(h_{i+1}) - \beta(h_i) = 0.5 + \epsilon$ for every i with $0 \le i < 2Z$, and $\beta(h_{2Z}) = \beta(b)$, where $\epsilon = (Z' - Z)/2Z$. It is straightforward to verify that this assignment satisfies the constraints in (C1) and (C2). This completes the proof of Claim 33.4. $\diamond$

We are now ready to show how to replace the constraints $z = B(z) \vee z - B(z) \ge M$ and $z = G(z) \vee G(z) - z \ge M$ for every variable $z \in \mathrm{XY}$. That is, for $z \in \mathrm{XY}$, we replace the constraint $z = B(z) \vee z - B(z) \ge M$ with the gadget $U(z, B(z), M)$ and we replace the constraint $z = G(z) \vee G(z) - z \ge M$ with the gadget $U(G(z), z, M)$.

Then, $\mathcal{I}'$ is obtained from $\mathcal{I}'''$ after replacing all the remaining constraints of the variables in XY as described above. Clearly, $\mathcal{I}'$ is an instance of CSP($\mathbf{D}_{2,1}$). Furthermore, the treewidth of the primal graph of $\mathcal{I}'$ is at most the treewidth of the primal graph of $\mathcal{I}''$ plus 2. This is because the treewidth of the primal graph of $\mathcal{I}'''$ is at most the treewidth of $\mathcal{I}''$ plus 1 (as we already argued above). Furthermore, the primal graph for $\mathcal{I}'$ is obtained from the primal graph of $\mathcal{I}'''$ by replacing the edges between x and $B(x)$ as well as between x and $G(x)$ with the primal graph of the gadget $U(a, b, Z)$ for every $x \in \mathrm{XY}$. The result now follows because the treewidth of the primal graph of $U(a, b, Z)$ is at most 2. Because the treewidth of the primal graph of $\mathcal{I}''$ is at most $4K + 3$, we obtain that the treewidth of the primal graph of $\mathcal{I}'$ is at most $4K + 5$. $\square$

Since all variables in the proof of Theorem 33 are only assigned integers, we can replace every constraint $L < R$ that uses $<$ in the construction, i.e. the constraints of the form $g^{V_i}_{|V_i|} - g^{V_i}_0 < T$ and $g^{E_i}_{|E_i|} - g^{E_i}_0 < T$, by $L \le R - 1$. Therefore, we obtain the following corollary from Theorem 33.

Corollary 34. CSP($\mathbf{D}^{\le}_{2,1}$) *is strongly W[1]-hard parameterized by primal treewidth.*

7. Generalisations

The results that we have proved in Sections 3–6 are restricted in two ways: (1) formulas are assumed to be in conjunctive normal form and (2) the variable domains are assumed to be the set of rationals. We consider the satisfiability problem for DL without these restrictions in the following two sections. Thus, we discuss DL with general formulas (i.e. the DL-Sat problem from Section 2.1) in Section 7.1 and we discuss DL with integer variable domains in Section 7.2.

Our generalisation to DL-Sat is based on replacing clauses with more complex subformulas. This can be done by viewing CSP instances as primitive positive sentences over suitable structures (as explained in Section 2.2), and restricting the subformulas so that the resulting formulas can be analysed with respect to computational and parameterized complexity using the results from Sections 3–6. Proposition 36 shows that many of our results are still valid under this generalisation. When switching to integer domains instead of rational domains, we use a well-known fact: a solvable set of constraints of the form $x - y \le c$ with $c \in \mathbb{Z}$ always has an integral solution. This has been observed in many different contexts, see [28, Section 3] and [64, Section 13.2]. The very definition of DL now allows us to transfer results for rational domains into integer domains in a uniform way, and this shows that, for instance, all results in Tables 3 and 4 hold in the integer case.

7.1. General formulas

Every DL formula can be converted into a logically equivalent formula that is in CNF by using well-known laws of logic. We use this fact for proving the following result.

Theorem 35. DL-Sat *is solvable in* $2^{O(n(\log n+\log k))}$ *time where n is the number of variables in the given formula* ϕ *and* $k = \mathrm{num}(\phi)$. DL-Sat *is not solvable in* $2^{o(n(\log n+\log k))}$ *if the ETH holds.*

Proof. The lower bound is an immediate consequence of Theorem 17 since every instance of CSP($\mathbf{D}_2^{\leqslant}$) can be viewed as a DL formula (as was discussed in Section 2.2). To show the upper bound, we let ϕ denote an arbitrary instance of DL-Sat. Assume ϕ contains n variables and that $k = \mathrm{num}(\mathcal{I})$. Every existential sentence ϕ admits a logically equivalent existential sentence ϕ' such that ϕ' is in CNF, ϕ and ϕ' contains the same number of variables, and $k = \mathrm{num}(\phi')$. The formula ϕ' may be viewed as an instance $\mathcal{I} = (V, C)$ of CSP($\mathbf{D}$) where $|V| = n$ and $\mathrm{num}(\mathcal{I}) = k$. Theorem 6 implies that ϕ' is satisfiable if and only if it has a solution $f : V \to CD(n,k)$. Since ϕ and ϕ' are logically equivalent formulas, the same holds for ϕ. The upper bound follows immediately since $CD(n,k)$ contains $2^{O(n(\log n+\log k))}$ elements (as was proved in Corollary 7). $\square$

The conversion of a DL formula into CNF can obviously lead to an exponentially larger formula and the conversion process may thus take exponential time. Note, however, that we do not need to compute the CNF formula explicitly in the proof of Theorem 35.

Theorem 35 is closely connected to *Satisfiability Modulo Theories* (SMT), i.e. the decision problem for logical sentences with respect to a given background theory, where logical formulas are expressed in classical first-order logic with equality. Let SMT($\mathcal{T}$) be the problem of determining whether a first-order sentence is true with respect to a background theory $\mathcal{T}$, and let $\mathrm{SMT}_\exists(\mathcal{T})$ be the subproblem where universal quantifiers are not allowed. If we let $\mathcal{T}_{\mathrm{diff}}$ denote the background theory for difference constraints, then DL-Sat and $\mathrm{SMT}_\exists(\mathcal{T}_{\mathrm{diff}})$ are the same computational problems. Jonsson and Lagerkvist [49, Theorem 9] prove bounds valid for any background theory: $\mathrm{SMT}_\exists(\emptyset)$ is solvable in $2^{O(|V|\log|V|)}$ time but it cannot be solved in $2^{o(|V|\log|V|)}$ time unless the ETH is false. Theorem 35 thus implies that $\mathrm{SMT}_\exists(\mathcal{T}_{\mathrm{diff}})$ is only marginally harder than $\mathrm{SMT}_\exists(\emptyset)$.

Applying the restricted time complexity results and the parameterized results to DL-Sat directly is, unfortunately, not possible. The clause arity parameter is obviously not well-defined for an arbitrary existential formula ϕ since it is not required to be in CNF. Similarly, the primal and incidence graphs are not well-defined in this case. Converting the formula into CNF is typically not a viable option since this process may take exponential time and it may produce a formula that is exponentially larger than the original formula. A simple (but sometimes sufficiently powerful) workaround is based on generalising the results to more complex subformulas than clauses. We present one possible way of doing this. Recall from Section 2.2 that we can always view a CSP instance as a primitive positive sentence over some structure. We have used this perspective throughout the article: we view an existential formula in CNF as a CSP instance where the structure contains the relations that describe the allowed clauses. Clearly, we can instead consider relations that describe other subformulas than clauses. This must be done with care, though. These relations cannot use auxiliary variables in their definitions since this introduces a time complexity dependency on the number of subformulas and not only on the number of variables and the magnitude of the coefficients. Furthermore, the definitions of the subformulas must (in a certain sense) be easy to compute. We circumvent this problem by restricting ourselves to a finite number of subformula types; this restriction can often be lifted but it needs a careful analysis based on the chosen relations and the representation of them. We arrive at the following result.

Proposition 36. *If* $\mathbf{A}$ *is a finite structure that is quantifier-free definable in* $\mathbf{S}$*, then the following hold.*

1. *CSP($\mathbf{A}$) is solvable in* $2^{O(n\log n)}$ *time.*
2. *If the relations in* $\mathbf{A}$ *have arity at most 2, then CSP($\mathbf{A}$) is solvable in* $2^{O(n\log\log n)}$ *time.*
3. *If the relations in* $\mathbf{A}$ *have arity at most 3 and* $\mathrm{num}(\mathbf{A}) = 0$*, then CSP($\mathbf{A}$) is solvable in* $2^{O(n)}$ *time.*
4. *CSP($\mathbf{A}$) is in XP when parameterized by the treewidth of the incidence graph.*

Proof. The structure $\mathbf{A}$ is finite and every relation in $\mathbf{A}$ has a quantifier-free definition in $\mathbf{S}$. We may without loss of generality assume that the defining formulas are in CNF. This implies that every relation can be viewed as a conjunction of relations in $\mathbf{D}_{a,k}$ where $a, k < \infty$. Since $\mathbf{A}$ is finite, we may assume that we have access to a table containing the defining CNF formulas for each relation in $\mathbf{A}$.

Let $\mathcal{I} = (V, C)$ denote an instance of CSP($\mathbf{A}$) and arbitrarily choose a constraint $R(x_1, \ldots, x_n)$ in C. The relation R has a definition

$$R(x_1, \ldots, x_n) \equiv \bigwedge_{i=1}^{p} R_i(x_{i,1}, \ldots, x_{i,\mathrm{ar}(R_i)})$$

where $R_1, \ldots, R_p \in \mathbf{D}_{a,k}$ and $\{x_{i,j} : 1 \le i \le p, 1 \le j \le \max\{\mathrm{ar}(R_m) : 1 \le m \le p\}\} \subseteq \{x_1, \ldots, x_n\}$. This constraint in $\mathcal{I}$ can be replaced by p constraints

$$R_1(x_{1,1}, \ldots, x_{1,\mathrm{ar}(R_1)}), \ldots, R_p(x_{p,1}, \ldots, x_{p,\mathrm{ar}(R_p)})$$

and this transformation does not affect the solvability of the instance since it preserves the set of solutions. Let $\mathcal{I}' = (V', C')$ denote the instance that results from applying this transformation to each constraint in C. We note that $\mathcal{I}'$ can be computed in polynomial time, $V' = V$, $\text{num}(C) = \text{num}(C')$, and $\mathcal{I}'$ has a solution if and only if $\mathcal{I}$ has a solution. Now, item 1 follows from Corollary 7, item 2 follows from Theorem 13, and item 3 follows from Theorem 1. Finally, we claim that item 4 follows from Theorem 22 together with the observation that $\text{tw}(I(\mathcal{I}')) \leq q \cdot \text{tw}(I(\mathcal{I}))$, where q is the smallest integer such that any constraint in $\mathcal{I}$ is replaced by at most q constraints in $\mathcal{I}'$; note that q can be considered constant because **A** is finite. The last observation follows because any tree decomposition of $I(\mathcal{I})$ can be transformed into a tree decomposition of $I(\mathcal{I}')$ by replacing any vertex corresponding to a constraint c in $\mathcal{I}$ by the at most q vertices corresponding to the constraints that replace c in $\mathcal{I}'$. □

7.2. Integer domains

We show that the complexity results presented in this article also hold if we restrict DL to integer variable domains. Henceforth, we let $\mathbf{D}^{\mathbb{Z}}_{a,k}$ denote the set of relations in $\mathbf{D}_{a,k}$ restricted to the integers. Let **A** and **B** be two structures. We write $\text{CSP}(\mathbf{A}) \leq_0 \text{CSP}(\mathbf{B})$ if there exists a polynomial-time reduction F from CSP(**A**) to CSP(**B**) that introduces no additional variables, i.e. if (V, C) is an instance of CSP(**A**), then $F((V, C)) = (V, C')$. The existence of such a reduction implies the following.

1. If CSP(**A**) is not solvable within a time bound $f(|V|)$, then CSP(**B**) is not solvable within $f(|V|)$, either.
2. If CSP(**B**) is solvable within a time bound $f(|V|)$, then CSP(**A**) is solvable within $f(|V|)$, too.

We continue by presenting a number of reductions.

Lemma 37. $\text{CSP}(\mathbf{D}^{\leq}_{a,k}) \leq_0 \text{CSP}(\mathbf{D}^{\mathbb{Z}}_{a,k}) \leq_0 \text{CSP}(\mathbf{D}^{\leq}_{a,k+1})$.

Proof. We first verify that if an instance $\mathcal{I}$ of $\text{CSP}(\mathbf{D}^{\leq})$ has a solution, then it has a solution over $\mathbb{Z}$, too. Let $\mathcal{I} = (V, C)$ with $C = \{c_1, \dots, c_p\}$ denote an arbitrary satisfiable instance of $\text{CSP}(\mathbf{D}^{\leq})$. Since $\mathcal{I}$ is satisfiable, we can pick one literal l_i out of the definition of every constraint $c_1, \dots, c_p$ such that $\{l_1, \dots, l_p\}$ is satisfiable. This set of constraints admits an integer solution since the literals are of the form $x - y \leq c$ and the bound c is an integer: this follows from the original algorithm for solving STPs by Dechter et al. [28, Section 3] but it is also a consequence of the theory of total unimodularity [64, Section 13.2]. Consequently, $\mathcal{I}$ admits an integer solution.

We begin with the reduction $\text{CSP}(\mathbf{D}^{\leq}_{a,k}) \leq_0 \text{CSP}(\mathbf{D}^{\mathbb{Z}}_{a,k})$. Let R denote a relation in $\mathbf{D}^{\leq}_{a,k}$ and let $R_{\mathbb{Z}}$ have the same definition as R but with domain $\mathbb{Z}$ instead of $\mathbb{Q}$. Let $\mathcal{I} = (V, C)$ denote an arbitrary instance of $\text{CSP}(\mathbf{D}^{\leq}_{a,k})$ and let $\mathcal{I}_{\mathbb{Z}} = (V, C_{\mathbb{Z}})$ where $C_{\mathbb{Z}} = \{R_{\mathbb{Z}}(x_1, \dots, x_k) : R(x_1, \dots, x_k) \in C\}$. If $\mathcal{I}$ is not satisfiable, then $\mathcal{I}_{\mathbb{Z}}$ is not satisfiable since $\mathbb{Z} \subseteq \mathbb{R}$. If $\mathcal{I}$ is satisfiable, then $\mathcal{I}_{\mathbb{Z}}$ is satisfiable as pointed out earlier.

We continue with the reduction $\text{CSP}(\mathbf{D}^{\mathbb{Z}}_{a,k}) \leq_0 \text{CSP}(\mathbf{D}^{\leq}_{a,k+1})$. Let $R_{\mathbb{Z}}$ denote a relation in $\mathbf{D}^{\mathbb{Z}}_{a,k}$. We define a relation R over $\mathbb{Q}$ as follows: R has the same definition as $R_{\mathbb{Z}}$ but every literal that is a strict inequality $x - y < c$ is replaced by $x - y \leq c - 1$. Observe that $R_{\mathbb{Z}} \subseteq R$ and R is a member of $\mathbf{D}^{\leq}_{a,k+1}$. Let $\mathcal{I}_{\mathbb{Z}} = (V, C_{\mathbb{Z}})$ denote an arbitrary instance of $\mathbf{D}^{\mathbb{Z}}_{a,k}$ and let $\mathcal{I} = (V, C)$ where $C = \{R(x_1, \dots, x_k) : R_{\mathbb{Z}}(x_1, \dots, x_k) \in C_{\mathbb{Z}}\}$. If $\mathcal{I}_{\mathbb{Z}}$ has a solution $f_{\mathbb{Z}}$, then this solution is a solution to $\mathcal{I}$, too, since $R_{\mathbb{Z}} \subseteq R$ for every $R_{\mathbb{Z}} \in \mathbf{D}^{\mathbb{Z}}_{a,k}$. If $\mathcal{I}$ is satisfiable, then it has an integer solution (as discussed earlier) and this solution witnesses the satisfiability of $\mathcal{I}_{\mathbb{Z}}$. □

Lemma 38. $\text{CSP}(\mathbf{D}_{a,0}) \leq_0 \text{CSP}(\mathbf{D}^{\mathbb{Z}}_{a,0}) \leq_0 \text{CSP}(\mathbf{D}_{a,0})$

Proof. We first verify that if an instance I of $\text{CSP}(\mathbf{D}_{a,0})$ has a solution, then it has a solution over $\mathbb{Z}$, too. Arbitrarily choose a satisfiable instance $\mathcal{I}$ of $\text{CSP}(\mathbf{D}_{a,0})$. Assume that $f : V \to \mathbb{Q}$ is a solution to $\mathcal{I}$. Observe that the function $f_c(x) = c \cdot f(x)$ is a solution to $\mathcal{I}$ whenever $c \neq 0$. We assume that $V = \{x_1, \dots, x_n\}$ and $f(x_i) = a_i/b_i$, $1 \leq i \leq n$, where a_i and $b_i \neq 0$ are integers. Let $c = b_1 \cdot \ldots \cdot b_n$ and note that f_c is a function from V to $\mathbb{Z}$. Thus, a satisfiable instance $\mathcal{I}$ of $\text{CSP}(\mathbf{D}_{a,0})$ always has an integer solution.

Let us now consider the reduction $\text{CSP}(\mathbf{D}_{a,0}) \leq_0 \text{CSP}(\mathbf{D}^{\mathbb{Z}}_{a,0})$. Given a relation $R \in \mathbf{D}_{a,0}$, we let $R_{\mathbb{Z}}$ denote R restricted to the integers. Let $\mathcal{I}_{\mathbb{Z}}$ denote an arbitrary instance of $\mathbf{D}^{\mathbb{Z}}_{a,0}$ and let $\mathcal{I} = (V, C)$ where $C = \{R(x_1, \dots, x_k) : R_{\mathbb{Z}}(x_1, \dots, x_k) \in C_{\mathbb{Z}}\}$. If $\mathcal{I}_{\mathbb{Z}}$ has a solution, then $\mathcal{I}$ has a solution, too. If $\mathcal{I}$ has a solution, then $\mathcal{I}_{\mathbb{Z}}$ has a solution as pointed out earlier. The other reduction is analogous. □

These reductions imply that all results in Table 3 hold for $\mathbf{D}^{\mathbb{Z}}_{a,k}$. Similarly, the results in Table 4 hold since the reductions do not change the primal and incidence graphs of a given instance. The results for DL-SAT (Theorem 35) also hold in the integer case. The reductions show that $\text{CSP}(\mathbf{D}^{\mathbb{Z}})$ is solvable in $2^{O(n(\log n + \log k))}$ time but not in $2^{o(n(\log n + \log k))}$ (under the ETH). The proof of Theorem 35 shows that these results immediately carry over to DL-SAT over the integers.

8. Conclusion and future work

We have initiated a fine-grained complexity analysis of the satisfiability problem for DL. We have studied the time complexity of CSP($\mathbf{D}_{a,k}$) and obtained closely matching bounds for almost all choices of $a, k \in \mathbb{N} \cup \{\infty\}$. We have studied the parameterized complexity of CSP($\mathbf{D}_{a,k}$) (with parameters primal and incidence treewidth) and obtained an almost comprehensive picture for all choices of a and k. We have considered generalisations where arbitrary formulas are allowed and where variable domains are the integers; many of our results survive such generalisations.

A future research direction is to close the gaps between lower bounds and upper bounds for time complexity. This boils down to a better understanding of the time complexity of CSP($\mathbf{D}_{2,k}$). There is a lack of natural problems that can be solved in $2^{O(n \log \log n)}$ time but do not admit a single-exponential time algorithm. This may point in the direction that CSP($\mathbf{D}_{2,k}$) is solvable in single-exponential time but it may equally well indicate a need for new lower bound techniques. We remark that the running time of the bounded-span algorithm (Lemma 9) is the dominant term in the time complexity of our algorithm for CSP($\mathbf{D}_{2,k}$) so improving this part would reduce the overall time complexity.

Our work on parameterized complexity has focused on the parameters primal and incidence treewidth; here we only leave open the question of the exact parameterized complexity of CSP($\mathbf{D}_{\infty,0}$) parameterized by incidence treewidth (which could be either in FPT or W[1]-hard). One possible way forward is to study other structural parameters. The notion of treewidth captures the fact that trees are structurally simple, but fails to do this for cliques since the treewidth of an n-clique is $n-1$. An alternative graph decomposition with a corresponding quality measure (known as *clique-width*) was introduced and analysed in a series of articles [23,24,81]. This decomposition captures the structure of both sparse graphs (such as trees) and dense graphs (such as cliques), and it is known to have algorithmic properties that are similar to those of bounded treewidth graphs. It may thus be highly relevant in connection with DL.

Algorithms for deciding the truth of DL formulas containing universal quantifiers are a natural step forward. Theorem 6 suggests a straightforward but incorrect approach. Consider a formula $Q_1 x_1 \dots Q_n x_n . \phi$ where $Q_i \in \{\forall, \exists\}$ and ϕ is quantifier-free. Let $D = CD(n,k)$ be the set of values needed for ϕ via Lemma 6. If $Q_1 = \forall$, then we assign the values from D to variable x_1 and recursively check that all assignments lead to satisfiability. If $Q_1 = \exists$, then we check that at least one assignment leads to satisfiability. However, such an algorithm does not work as intended: the formula $\forall x \exists y . y - x \geq 1$ is false when interpreted over any finite $D \subseteq \mathbb{Q}$ while it is true when interpreted over $\mathbb{Q}$. This implies that another algorithmic approach is needed for handling quantified DL formula.

CRediT authorship contribution statement

Konrad K. Dabrowski: Writing – review & editing, Writing – original draft, Validation, Methodology, Investigation, Formal analysis, Conceptualization. **Peter Jonsson:** Writing – review & editing, Writing – original draft, Validation, Methodology, Investigation, Funding acquisition, Formal analysis, Conceptualization. **Sebastian Ordyniak:** Writing – review & editing, Writing – original draft, Validation, Methodology, Investigation, Funding acquisition, Formal analysis, Conceptualization. **George Osipov:** Writing – review & editing, Writing – original draft, Validation, Methodology, Investigation, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The second and the fourth author were supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation. In addition, the second author was partially supported by the Swedish Research Council (VR) under grant 2021-0437. The third author was supported by the Engineering and Physical Sciences Research Council (EPSRC) (Project EP/V00252X/1).

Data availability

No data was used for the research described in the article.

References

[1] Martin Aigner, Günter M. Ziegler, Proofs from THE BOOK, 6th edition, Springer, 2018.
[2] James F. Allen, Maintaining knowledge about temporal intervals, Commun. ACM 26 (11) (1983) 832–843.
[3] Rajeev Alur, Timed automata, in: Proc. 11th International Conference on Computer Aided Verification (CAV-1999), 1999, pp. 8–22.
[4] Stefan Arnborg, Derek Corneil, Andrzej Proskurowski, Complexity of finding embeddings in a k-tree, SIAM J. Matrix Anal. Appl. 8 (2) (1987) 277–284.
[5] Goutam K. Audhya, Koushik Sinha, Sasthi C. Ghosh, A survey on the channel assignment problem in wireless networks, Wirel. Commun. Mob. Comput. 11 (5) (2011) 583–609.

[6] Federico Barber, Reasoning on interval and point-based disjunctive metric constraints in temporal contexts, J. Artif. Intell. Res. 12 (2000) 35–86.
[7] Clark W. Barrett, Roberto Sebastiani, Sanjit A. Seshia, Cesare Tinelli, Satisfiability modulo theories, in: Armin Biere, Marijn Heule, Hans van Maaren, Toby Walsh (Eds.), Frontiers in Artificial Intelligence and Applications, in: Handbook of Satisfiability, vol. 185, IOS Press, 2009, pp. 825–885.
[8] Umberto Bertelé, Franscesco Brioschi, Nonserial Dynamic Programming, Academic Press, 1972.
[9] Marc Bezem, Robert Nieuwenhuis, Enric Rodríguez-Carbonell, The max-atom problem and its relevance, in: Proc. 15th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR-2008), 2008, pp. 47–61.
[10] Nikhil Bhargava, Brian C. Williams, Multiagent disjunctive temporal networks, in: Proc. 18th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS-2019), 2019, pp. 458–466.
[11] Bernhard Bliem, Michael Morak, Marius Moldovan, Stefan Woltran, The impact of treewidth on grounding and solving of answer set programs, J. Artif. Intell. Res. 67 (2020) 35–80.
[12] Manuel Bodirsky, Complexity of Infinite-Domain Constraint Satisfaction, Cambridge University Press, 2021.
[13] Manuel Bodirsky, Víctor Dalmau, Datalog and constraint satisfaction with infinite templates, J. Comput. Syst. Sci. 79 (1) (2013) 79–100.
[14] Manuel Bodirsky, Peter Jonsson, A model-theoretic view on qualitative constraint reasoning, J. Artif. Intell. Res. 58 (2017) 339–385.
[15] Manuel Bodirsky, Marcello Mamino, Constraint satisfaction problems over numeric domains, in: Dagstuhl Follow-Ups, in: The Constraint Satisfaction Problem: Complexity and Approximability, vol. 7, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017, pp. 79–111.
[16] Hans L. Bodlaender, A linear-time algorithm for finding tree-decompositions of small treewidth, SIAM J. Comput. 25 (6) (1996) 1305–1317.
[17] Hans L. Bodlaender, Ton Kloks, Efficient and constructive algorithms for the pathwidth and treewidth of graphs, J. Algorithms 21 (2) (1996) 358–402.
[18] James C. Boerkoel, Edmund H. Durfee, Decoupling the multiagent disjunctive temporal problem, in: Proc. 27th AAAI Conference on Artificial Intelligence (AAAI-2013), 2013.
[19] Chris Calabro, Russell Impagliazzo, Ramamohan Paturi, The complexity of satisfiability of small depth circuits, in: Proc. 4th International Workshop on Parameterized and Exact Computation (IWPEC-2009), 2009, pp. 75–85.
[20] Lorenzo Candeago, Daniel Larraz, Albert Oliveras, Enric Rodríguez-Carbonell, Albert Rubio, Speeding up the constraint-based method in difference logic, in: Proc. 19th International Conference on the Theory and Applications of Satisfiability Testing (SAT-2016), 2016, pp. 284–301.
[21] Clément Carbonnel, Martin C. Cooper, Tractability in constraint satisfaction problems: a survey, Constraints 21 (2) (2016) 115–144.
[22] Stephen A. Cook, The complexity of theorem-proving procedures, in: Proc. 3rd Annual ACM Symposium on Theory of Computing (STOC-1971), 1971, pp. 151–158.
[23] Bruno Courcelle, Joost Engelfriet, Grzegorz Rozenberg, Handle-rewriting hypergraph grammars, J. Comput. Syst. Sci. 46 (2) (1993) 218–270.
[24] Bruno Courcelle, Stephan Olariu, Upper bounds to the clique width of graphs, Discrete Appl. Math. 101 (1–3) (2000) 77–114.
[25] Konrad K. Dabrowski, Peter Jonsson, Sebastian Ordyniak, George Osipov, Fine-grained complexity of temporal problems, in: Proc. 17th International Conference on Principles of Knowledge Representation and Reasoning (KR-2020), 2020, pp. 284–293.
[26] Konrad K. Dabrowski, Peter Jonsson, Sebastian Ordyniak, George Osipov, Disjunctive temporal problems under structural restrictions, in: Proc. 35th AAAI Conference on Artificial Intelligence (AAAI-2021), 2021, pp. 3724–3732.
[27] Konrad K. Dabrowski, Peter Jonsson, Sebastian Ordyniak, George Osipov, Solving infinite-domain CSPs using the patchwork property, Artif. Intell. 317 (2023) 103880.
[28] Rina Dechter, Itay Meiri, Judea Pearl, Temporal constraint networks, Artif. Intell. 49 (1–3) (1991) 61–95.
[29] Rodney G. Downey, Michael R. Fellows, Fundamentals of Parameterized Complexity, Springer, 2013.
[30] Paul Erdős, Pál Turán, On a problem of Sidon in additive number theory, and on some related problems, J. Lond. Math. Soc. 1 (4) (1941) 212–215.
[31] Leif Eriksson, Victor Lagerkvist, Improved algorithms for Allen's interval algebra: a dynamic programming approach, in: Proc. 30th International Joint Conference on Artificial Intelligence (IJCAI-2021), 2021, pp. 1873–1879.
[32] Jörg Flum, Martin Grohe, Parameterized Complexity Theory, Springer, 2006.
[33] Robert Ganian, Fabian Klute, Sebastian Ordyniak, On structural parameterizations of the bounded-degree vertex deletion problem, Algorithmica 83 (1) (2021) 297–336.
[34] Robert Ganian, Sebastian Ordyniak, The complexity landscape of decompositional parameters for ILP, Artif. Intell. 257 (2018) 61–71.
[35] Robert Ganian, Sebastian Ordyniak, C.S. Rahul, Group activity selection with few agent types, Algorithmica 85 (5) (2023) 1111–1155.
[36] Michael R. Garey, David S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman and Company, 1979.
[37] Serge Gaspers, Exponential Time Algorithms - Structures, Measures, and Bounds, VDM, 2010.
[38] Alfonso Gerevini, Alessandro Saetti, Ivan Serina, An approach to temporal planning and scheduling in domains with predictable exogenous events, J. Artif. Intell. Res. 25 (2006) 187–231.
[39] Martin Charles Golumbic, Haim Kaplan, Ron Shamir, On the complexity of DNA physical mapping, Adv. Appl. Math. 15 (1994) 251–261.
[40] Georg Gottlob, Reinhard Pichler, Fang Wei, Bounded treewidth as a key to tractability of knowledge representation and reasoning, in: Proc. 21st National Conference on Artificial Intelligence (AAAI-2006), 2006, pp. 250–256.
[41] Georg Gottlob, Francesco Scarcello, Martha Sideri, Fixed-parameter complexity in AI and nonmonotonic reasoning, Artif. Intell. 138 (1–2) (2002) 55–86.
[42] Heini Halberstam, Klaus Roth, Sequences, Springer, 1966.
[43] Wilfrid Hodges, A Shorter Model Theory, Cambridge University Press, New York, NY, USA, 1997.
[44] Hoon Hong, Comparison of several decision algorithms for the existential theory of the reals, Technical Report 91-41, RISC, Linz, 1991.
[45] Jinbo Huang, Jason Jingshi Li, Jochen Renz, Decomposition and tractability in qualitative spatial and temporal reasoning, Artif. Intell. 195 (2013) 140–164.
[46] Russell Impagliazzo, Ramamohan Paturi, Francis Zane, Which problems have strongly exponential complexity?, J. Comput. Syst. Sci. 63 (4) (2001) 512–530.
[47] Peter Jonsson, Christer Bäckström, A unifying approach to temporal constraint reasoning, Artif. Intell. 102 (1) (1998) 143–155.
[48] Peter Jonsson, Victor Lagerkvist, Why are CSPs based on partition schemes computationally hard?, in: Proc. 43rd International Symposium on Mathematical Foundations of Computer Science (MFCS-2018), 2018, pp. 43:1–43:15.
[49] Peter Jonsson, Victor Lagerkvist, General lower bounds and improved algorithms for infinite-domain CSPs, Algorithmica 85 (1) (2023) 188–215.
[50] Peter Jonsson, Tomas Lööw, Computational complexity of linear constraints over the integers, Artif. Intell. 195 (2013) 44–62.
[51] Kloks Ton, Treewidth: Computations and Approximations, LNCS., vol. 842, Springer, 1994.
[52] Phokion G. Kolaitis, Moshe Y. Vardi, Conjunctive-query containment and constraint satisfaction, J. Comput. Syst. Sci. 61 (2) (2000) 302–332.
[53] Koubarakis Manolis, Tractable disjunctions of linear constraints: basic results and applications to temporal reasoning, Theor. Comput. Sci. 266 (1–2) (2001) 311–339.
[54] Král' Daniel, An exact algorithm for the channel assignment problem, Discrete Appl. Math. 145 (2) (2005) 326–331.
[55] Yuliya Lierler, Benjamin Susman, On relation between constraint answer set programming and satisfiability modulo theories, Theory Pract. Log. Program. 17 (4) (2017) 559–590.
[56] Daniel Lokshtanov, Dániel Marx, Saket Saurabh, Slightly superexponential parameterized problems, SIAM J. Comput. 47 (3) (2018) 675–702.
[57] Carsten Lutz, Maja Miličić, A tableau algorithm for description logics with concrete domains and general tboxes, J. Autom. Reason. 38 (1–3) (2007) 227–259.

[58] Peter Niebert, Moez Mahfoudh, Eugene Asarin, Marius Bozga, Oded Maler, Navendu Jain, Verification of timed automata via satisfiability checking, in: Proc. 7th International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems (FTRTFT-2002), 2002, pp. 225–244.
[59] Rolf Niedermeier, Invitation to Fixed-Parameter Algorithms, Oxford University Press, 2006.
[60] Niemelä Ilkka, Stable models and difference logic, Ann. Math. Artif. Intell. 53 (1–4) (2008) 313–329.
[61] Robert Nieuwenhuis, Albert Oliveras, Cesare Tinelli, Solving SAT and SAT modulo theories: from an abstract Davis–Putnam–Logemann–Loveland procedure to DPLL(T), J. ACM 53 (6) (2006) 937–977.
[62] Kevin O'Bryant, A complete annotated bibliography of work related to Sidon sequences, Electron. J. Comb. Dynamic Survey 11 (2004).
[63] Angelo Oddi, Amedeo Cesta, Incremental forward checking for the disjunctive temporal problem, in: Proc. 14th European Conference on Artificial Intelligence (ECAI-2000), 2000, pp. 108–112.
[64] Christos H. Papadimitriou, Kenneth Steiglitz, Combinatorial Optimization: Algorithms and Complexity, Prentice-Hall, 1982.
[65] Itsik Pe'er, Ron Shamir, Satisfiability problems on intervals and unit intervals, Theor. Comput. Sci. 175 (2) (1997) 349–372.
[66] Bart Peintner, Kristen Brent Venable, Neil Yorke-Smith, Strong controllability of disjunctive temporal problems with uncertainty, in: Proc. 13th International Conference on Principles and Practice of Constraint Programming (CP-2007), 2007, pp. 856–863.
[67] Krzysztof Pietrzak, On the parameterized complexity of the fixed alphabet shortest common supersequence and longest common subsequence problems, J. Comput. Syst. Sci. 67 (4) (2003) 757–771.
[68] James Renegar, On the computational complexity and geometry of the first-order theory of the reals. Part I: introduction. Preliminaries. The geometry of semi-algebraic sets. The decision problem for the existential theory of the reals, J. Symb. Comput. 13 (3) (1992) 255–299.
[69] Neil Robertson, Paul D. Seymour, Graph minors. III. Planar tree-width, J. Comb. Theory, Ser. B 36 (1) (1984) 49–64.
[70] Marko Samer, Stefan Szeider, Constraint satisfaction with bounded treewidth revisited, J. Comput. Syst. Sci. 76 (2) (2010) 103–114.
[71] Andreas Schutt, Peter J. Stuckey, Incremental satisfiability and implication for UTVPI constraints, INFORMS J. Comput. 22 (4) (2010) 514–527.
[72] Sanjit A. Seshia, K. Subramani, Randal E. Bryant, On solving Boolean combinations of UTVPI constraints, J. Satisf. Boolean Model. Comput. 3 (1–2) (2007) 67–90.
[73] Simon Sidon, Ein Satz über trigonometrische Polynome und seine Anwendung in der Theorie der Fourier-Reihen, Math. Ann. 106 (1) (1932) 536–539.
[74] Arkadiusz Socała, Tight lower bound for the channel assignment problem, ACM Trans. Algorithms 12 (4) (2016) 48.
[75] Kostas Stergiou, Manolis Koubarakis, Backtracking algorithms for disjunctions of temporal constraints, Artif. Intell. 120 (1) (2000) 81–117.
[76] Larry J. Stockmeyer, Albert R. Meyer, Word problems requiring exponential time: preliminary report, in: Proc. 5th Annual ACM Symposium on Theory of Computing (STOC-1973), 1973, pp. 1–9.
[77] Patrick Traxler, The time complexity of constraint satisfaction, in: Proc. 3rd International Workshop on Parameterized and Exact Computation (IWPEC-2008), Springer, 2008, pp. 190–201.
[78] Ioannis Tsamardinos, Martha E. Pollack, Efficient solution techniques for disjunctive temporal reasoning problems, Artif. Intell. 151 (1–2) (2003) 43–89.
[79] Kristen Brent Venable, Neil Yorke-Smith, Disjunctive temporal planning with uncertainty, in: Proc. 19th International Joint Conference on Artificial Intelligence (IJCAI-2005), 2005, pp. 1721–1722.
[80] Marc B. Vilain, Henry A. Kautz, Constraint propagation algorithms for temporal reasoning, in: Proc. 5th National Conference on Artificial Intelligence (AAAI-1986), 1986, pp. 377–382.
[81] Egon Wanke, k-NLC graphs and polynomial algorithms, Discrete Appl. Math. 54 (2–3) (1994) 251–266.
[82] Matteo Zavatteri, Alice Raffaele, Dario Ostuni, Romeo Rizzi, An interdisciplinary experimental evaluation on the disjunctive temporal problem, Constraints 28 (1) (2023) 1–12.