Disjunctive Temporal Problems under Structural Restrictions

Konrad K. Dabrowski,[1] **Peter Jonsson,**[2] **Sebastian Ordyniak,**[3] **George Osipov**[2]
[1]Durham University
[2]Linköping University
[3]University of Leeds
konrad.dabrowski@durham.ac.uk, {peter.jonsson,george.osipov}@liu.se, sordyniak@gmail.com

Abstract

The *disjunctive temporal problem* (DTP) is an expressive temporal formalism that extends Dechter et al.'s *simple temporal problem*. The DTP is well studied in the literature and has many important applications. It is known that deciding satisfiability of DTPs is NP-hard and that, in many cases, single-exponential algorithms (running in $O(c^n)$ time) do not exist under the Exponential-Time Hypothesis. The computational hardness makes it worthwhile to identify restricted problems that are efficiently solvable. One way of doing this is to restrict the interactions of variables and constraints. We show that instances of DTP of any arity with integers bounded by $poly(n)$ can be solved in $n^{f(w)}$ time, where n denotes the problem size, w is the treewidth of the incidence graph and f is a computable function; in other words, this problem is in the complexity class XP and it can be solved in polynomial time whenever w is fixed. We complement this result by showing that binary DTPs that only involve the integers 0 and 1 are not fixed-parameter tractable with respect to treewidth, i.e. they do not admit a $f(w) \cdot poly(n)$ time algorithm for any computable function f, under standard complexity assumptions. For instances with unbounded integers, we show that even binary DTPs parameterized by treewidth cannot be in XP, unless P = NP.

Introduction

Temporal reasoning is a fundamental task in AI and one of the most influential temporal formalisms is the *simple temporal problem* (STP) (Dechter, Meiri, and Pearl 1991). It is a constraint satisfaction problem (CSP) over a language with relations $\{(x, y) \in \mathbb{R}^2 : \ell \leq x - y \leq u\}$ where $\ell, u \in \mathbb{R} \cup \{-\infty, +\infty\}$. Even though the STP is immensely useful in a wide range of applications, its expressive power is limited. Increased expressibility can be achieved by using disjunctions (Barber 2000; Dechter, Meiri, and Pearl 1991; Oddi and Cesta 2000; Stergiou and Koubarakis 2000). Such disjunctive STPs are highly relevant in an AI context: examples can be found in automated planning (Gerevini, Saetti, and Serina 2006; Venable and Yorke-Smith 2005) and multi-agent systems (Bhargava and Williams 2019; Boerkoel and Durfee 2013), while Stergiou & Koubarakis (2000, Sec. 7), Tsamardinos & Pollack (2003) and Peintner et al. (2007) discuss various other applications. DTPs also have many applications in areas outside AI such as bioinformatics and graph theory (Pe'er and Shamir 1997), and in telecommunications via the channel assignment problem (see, for instance, Audhya et al. (2011) and Král' (2005)).

Augmenting STPs with disjunctions (Oddi and Cesta 2000; Stergiou and Koubarakis 2000) yields a language $\mathbf{D}$ as follows. We consider intervals over $\mathbb{Q}$ with endpoints in $\mathbb{Z} \cup \{-\infty, +\infty\}$. The intervals may be open, closed, half-closed, and even 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 $1 \leq \ell \leq m$. The CSP for $\mathbf{D}$ is known as the *disjunctive temporal problem* (DTP). We make, without loss of generality, the sensible assumption that the bounding values are integers (see e.g. Tsamardinos & Pollack (2003)): (most) real values cannot 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 the STP is a polynomial-time solvable problem, the DTP is NP-hard (even though a number of polynomial-time solvable fragments are known, see e.g. (Comin and Rizzi 2018; Kumar 2005)). Dabrowski et al. (2020) have additionally shown that the DTP (and many severely restricted variants) cannot be solved in $O(c^n)$ time (for any constant c) under the Exponential-Time Hypothesis (ETH), i.e. the hypothesis that 3SAT cannot be solved in $2^{o(n)}$ time, where n is the number of variables (Impagliazzo and Paturi 2001). This motivates the search for efficiently solvable subproblems.

To this end, we use the framework of *parameterized complexity* (Flum and Grohe 2006; Niedermeier 2006; Downey and Fellows 2013), 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 (*fixed-parameter tractable*), which contains all problems that 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. (Flum and Grohe 2006, Corr. 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, a problem is pNP-hard if it is NP-hard for some constant value of the parameter. 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 & Cooper (2016, 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 (Gottlob, Scarcello, and Sideri 2002), while this is not true (under standard complexity assumptions) if w is the incidence treewidth (Samer and Szeider 2010). We observe that treewidth has been successfully employed for many (related) application areas in AI (Gottlob, Pichler, and Wei 2006) with various relevant practical applications (Bliem et al. 2020).

To describe our results, let $\mathrm{num}(R)$ be the largest absolute value appearing in the definition of a relation $R \in \mathbf{D}$, and let $\mathbf{D}_{a,k}$ (where $a, k \in \mathbb{N}\cup\{\infty\}$) denote the class of relations of arity at most a with $\mathrm{num}(R) \leq k$. The primal treewidth is bounded from below by the incidence treewidth (Kolaitis and Vardi 2000) for arbitrary CSP instances. Thus, we present algorithms for DTPs parameterized by incidence treewidth and lower bounds with respect to primal treewidth. Certain families of infinite-domain CSPs are known to be in XP when parameterized by primal treewidth (Bodirsky and Dalmau 2013; Huang, Li, and Renz 2013). We start by showing that these results are not applicable even to CSP($\mathbf{D}_{2,1}$). In contrast to this, we present an XP algorithm for CSP($\mathbf{D}_{\infty,k}$) when $k \in \mathbb{N}$. Additionally, we prove that CSP($\mathbf{D}_{2,k}$) for $1 \leq k < \infty$, is not in FPT (under standard complexity-theoretic assumptions), thus showing that significantly faster algorithms for DTP are unlikely. Problems such as Allen's Algebra and RCC8 are in FPT (Dabrowski et al. 2021) so DTP is in fact a substantially harder problem. We complement all of these results by showing that CSP($\mathbf{D}_{2,\infty}$) is pNP-hard, i.e. the problem becomes much harder when the numeric values are unbounded.

We summarize our results in Table 1. All results for $k \geq 1$ can be found in this paper. 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 (Vilain and Kautz 1986). It is well known that CSP($\mathbf{D}_{k,0}$) for $k \geq 3$ is NP-hard (this follows, for instance, from an easy reduction from the BETWEENNESS problem (Garey and Johnson 1979)) and results by Dabrowski et al. (2021) show that these problems are in FPT.

	$k = 0$	$1 \leq k < \infty$	k unbounded
$a = 2$	$\in$ P	$\notin$ FPT, $\in$ XP	pNP-h.
$a > 2$	NP-h., $\in$ FPT	$\notin$ FPT, $\in$ XP	pNP-h.

Table 1: Summary of complexity landscape for CSP($\mathbf{D}_{a,k}$)

Preliminaries

Constraint Satisfaction

Let Γ (the *constraint language*) denote a set of finitary relations defined on a set D of values. Observe that we do not require Γ or D to be finite. The *constraint satisfaction problem* over Γ (CSP(Γ)) is defined as follows:

Instance: A tuple (V, C), where V is a set of variables and C is a set of constraints of the form $R(v_1, \ldots, v_t)$, where t is the arity of R, $v_1, \ldots, v_t \in V$, and $R \in \Gamma$.
Question: Is there a function $f : V \to D$ such that $(f(v_1), \ldots, f(v_t)) \in R$ for every $R(v_1, \ldots, v_t) \in C$?

Such a function f is a *satisfying assignment* or simply a *solution*. We denote the set of variables appearing in the scope of a constraint c by $\mathrm{S}(c)$.

Treewidth

Treewidth is based on *tree decompositions* (Bertelè and Brioschi 1972; Robertson and Seymour 1984): 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:

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.

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. 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 $\chi(t) = \chi(t_1) = \chi(t_2)$.

Proposition 1 (Bodlaender & Kloks (1996); Kloks (1994))**.** *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 $\mathcal{O}(|V|)$ nodes can be computed in linear time.*

Parameterized Complexity

The parameterized complexity classes we need were introduced in the introduction. We will prove that certain problems are not in FPT and this requires some extra 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 $\mathsf{W}[1]$ contains all problems that are fpt-reducible to Independent Set when parameterized by the size of the solution, i.e. the number of vertices in the independent set. Showing $\mathsf{W}[1]$-hardness (by an fpt-reduction) for a problem rules out the existence of a fixed-parameter algorithm under the standard assumption $\mathsf{FPT} \neq \mathsf{W}[1]$.

Upper bounds

To the best of our knowledge, there are two XP algorithms described in the literature (one by Bodirsky & Dalmau (2013) and another by Huang et al. (2013)) that could potentially be used for solving CSP($\mathbf{D}_{\infty,k}$). We start by showing that they are not applicable, and continue by presenting a novel XP algorithm for CSP($\mathbf{D}_{\infty,k}$), $k < \infty$.

Earlier Algorithms

Bodirsky & Dalmau (2013) and Huang et al. (2013) proved that CSP(Γ) is in XP for ω-categorical Γ and binary constraint languages Γ 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 $O(w^3 n \cdot \mathrm{e}^{w^2 \log n}) = n^{O(w^2)}$ time. These two general results apply to certain temporal problems, e.g. first-order reducts of $(\mathbb{Q}; <)$ are ω-categorical (Bodirsky and Kára 2010), while Allen's Interval Algebra has the aNAP (Lutz and Miličić 2007). However, these properties do not hold for the constraint language $\mathbf{D}$ or even its fragment $\mathbf{D}_{2,1}$.

By the theorem of Engeler, Ryll-Nardzewski, and Svenonius (see e.g. (Hodges 1997)), if Γ is an ω-categorical constraint language, then for all $n > 1$, there are finitely many inequivalent formulas over Γ 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 binary disjunctive Γ has aNAP, then for any pair of complete atomic instances (V_1, C_1) and (V_2, C_2) of CSP(Γ) 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 Γ 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 $y - x \in (1,2)$.

New Algorithm

We will now present an XP algorithm for CSP($\mathbf{D}_{\infty,k}$), $k \in \mathbb{N}$. To simplify the presentation, we define the set $CD(n,k) = \{z + \frac{q}{n} \mid z, q \in \mathbb{N}, 0 \leq z \leq (n-1)(k+1), 0 \leq q < n\}$ for $n \in \mathbb{N}$.

We will first show that any solvable instance of CSP($\mathbf{D}_{\infty,k}$) with n variables has a solution with domain $CD(n,k)$. We omit the proof.

Lemma 2. *Every satisfiable instance $\mathcal{I} = (V, C)$ of* CSP($\mathbf{D}_{\infty,k}$) *has a solution $f : V \to CD(|V|, k)$.*

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

Theorem 3. CSP($\mathbf{D}_{\infty,k}$) *can be solved in $(nk)^{\mathcal{O}(w)}$ time, where w is the treewidth of the incidence graph and n is the number of variables.*

Note that the bound implies that CSP($\mathbf{D}$) is in XP whenever the numeric values are bounded by a polynomial in the input size. Because of Proposition 1, 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,

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 Lemma 2, we may assume that every solution for $\mathcal{I}$ maps the variables into the set $CDC = CD(n,k)$.

Intuitively, the algorithm behind Theorem 3 works as follows. It uses a bottom-up dynamic programming approach on the nodes of T (starting from the leaves and finishing at the root) to compute a compact representation, in the following 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:

- $\alpha : \mathrm{var}_\chi(t) \to CDC$ is an assignment of values in CDC to the variables in $\mathrm{var}_\chi(t)$, and
- $\beta : \mathrm{con}_\chi(t) \to D_B$, where $D_B = \{S, U\} \cup \{ (v,d) \mid v \in V \text{ and } d \in CDC \}$ 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 $v \in \mathrm{S}(c) \cap (\mathrm{var}_\chi(T_t) \setminus \mathrm{var}_\chi(t))$ and $d \in CDC$, 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 $|CDC|$ possible choices for every variable in $\mathrm{var}_\chi(t)$ and at most $|V||CDC| + 2$ possible choices for every constraint in $\mathrm{con}_\chi(t)$. Therefore, the total number of valid records for t is at most $(|V||CDC|+2)^{w+1}$.

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 $d \in CDC$ 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 $\tau : \mathrm{var}_\chi(T_t) \to CDC$ 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 $v \in \mathrm{var}_\chi(t)$, and

(R3) $\tau(v) = d$ holds for every constraint $c \in \mathrm{con}_\chi(t)$ 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. A concrete solution can be computed by using standard techniques (Downey and Fellows 2013).

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 4 (leaf node). *Let $t \in V(T)$ be a leaf node with $\chi(t) = \{x\}$ for some variable or constraint x. Then, $\mathcal{R}(t)$ can be computed in $\mathcal{O}(|CDC|)$ time.*

Sketch of Proof. We first show the result when x is a variable. In this case, $\mathcal{R}(t)$ consists of all records $(\alpha, \emptyset)$ for every assignment $\alpha : \{x\} \to CDC$, so $\mathcal{R}(t)$ can be computed by enumerating all assignments α for x in time $\mathcal{O}(|CDC|)$. Correctness follows immediately from the definition of valid records.

We now show the result for the case that x is a constraint. Then, $\mathcal{R}(t)$ consists of the record $(\emptyset, \beta)$, where $\beta : \{x\} \to D_B$ is defined by setting $\beta(x) = U$. Hence, $\mathcal{R}(t)$ can be computed in constant time and correctness follows immediately from the definition of valid records. □

Lemma 5 (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 $\mathcal{O}(|\mathcal{R}(t_0)||CDC||\mathcal{I}|)$ 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 $\alpha_v : \{v\} \to CDC$ 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 $\alpha_v : \{v\} \to CDC$, 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 $c \in \mathrm{con}_\chi(t) \setminus (\beta_0^{-1}(S) \cup U' \cup F')$.

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 $\tau : \mathrm{var}_\chi(T_t) \to CDC$ satisfying (R1)–(R3). Let α_0 be the restriction of α to $\mathrm{var}_\chi(t_0)$ and let τ_0 be the restriction of τ to $\mathrm{var}_\chi(T_{t_0})$. Let Z be the set of all constraints in $\mathrm{con}_\chi(t) = \mathrm{con}_\chi(t_0)$ 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 $\mathrm{var}_\chi(T_t) \setminus \mathrm{var}_\chi(t)$ 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 $c \in \mathrm{con}_\chi(t_0) \setminus (X \cup Y)$, $\beta_0(c) = U$ 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 $R_0 = (\alpha_0, \beta_0) \in \mathcal{R}(t_0)$. Then, because R_0 is valid for t_0, there is an assignment $\tau_0 : \mathrm{var}_\chi(T_{t_0}) \to CDC$ satisfying (R1)–(R3). 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 $|\mathcal{R}(t_0)| \cdot |CDC|$ 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 $\mathrm{con}_\chi(t)$ for partial assignments and thus takes $\mathcal{O}(|\mathcal{I}|)$ time. □

Lemma 6 (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 $\mathcal{O}(|\mathcal{R}(t_0)||\mathcal{I}|)$ time.*

Sketch of 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 being to 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.

The correctness of the definition for $\mathcal{R}(t)$ follows from the semantics of records and the fact that (T, χ) is a tree decomposition: the scope of c does not contain any variable from $\mathrm{var}_\chi(T_t) \setminus \mathrm{var}_\chi(t)$; otherwise the edge between c and the variable in $\mathrm{var}_\chi(T_t) \setminus \mathrm{var}_\chi(t)$ in the incidence graph is not contained in any bag of T. Hence, it suffices to consider the assignment of the variables in $\mathrm{var}_\chi(t)$ to determine whether c is already satisfied.

Finally, the run-time follows because we have to consider every record (α_0, β_0) in $\mathcal{R}(t_0)$ and check in $\mathcal{O}(|\mathcal{I}|)$ time whether α satisfies c. □

Lemma 7 (**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 $\mathcal{O}(|\mathcal{R}(t_0)|2^w w)$ 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 $\mathrm{var}_\chi(t)$, 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 $\beta_0^{-1}(U) \cap \{\, c \in C \mid v \in \mathrm{S}(c) \,\}$, the set $\mathcal{R}(t)$ contains the record (α, β), where α is the restriction of α_0 to $\mathrm{var}_\chi(t)$ and β is defined by setting $\beta(c) = \beta_0(c)$ for every $c \in \mathrm{con}_\chi(t) \setminus U'$ 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 $\tau : \mathrm{var}_\chi(T_t) \to \mathit{CDC}$ satisfying (R1)–(R3). Let X be the set of all constraints c in $\mathrm{con}_\chi(t)$ such that $\beta(c) = (v, d)$. Because τ satisfies (R3), we actually have that $d = \tau(v)$ 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 $c \in X$. But then 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 $\tau_0 : \mathrm{var}_\chi(T_{t_0}) \to \mathit{CDC}$ 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 $\mathcal{O}(w)$. □

Lemma 8 (**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 $\mathcal{O}(|\mathcal{R}(t_0)||\chi(t)|)$ time.*

Sketch of 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 $\mathrm{con}_\chi(t)$. 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 $\mathrm{con}_\chi(t)$. The correctness of the definition for $\mathcal{R}(t)$ follows immediately from the definition of valid records.

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

Lemma 9 (**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 $\mathcal{O}(|\mathcal{R}(t_1)||\mathcal{R}(t_2)||\mathcal{I}|)$ time.*

Sketch of 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 $c \in \mathrm{con}_\chi(t)$ 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 $\beta_2(c) = (v, d)$

The correctness of the definition for $\mathcal{R}(t)$ follows immediately from the definition of valid records together with the following observations: (1) every constraint c in $\mathrm{con}_\chi(T_t) \setminus \mathrm{con}_\chi(t)$ is either satisfied by the variables in $\mathrm{var}_\chi(T_{t_1})$ or by the variables in $\mathrm{var}_\chi(T_{t_2})$, because (T, χ) is a tree decomposition and (2) every constraint c in $\mathrm{con}_\chi(t)$ is either satisfied by the variables in $\mathrm{var}_\chi(T_{t_1})$, by the variables in $\mathrm{var}_\chi(T_{t_2})$, or it is satisfied by a constraint containing one variable from $\mathrm{var}_\chi(T_{t_1})$ and one variable from $\mathrm{var}_\chi(T_{t_2})$.

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 $\mathcal{O}(|\mathcal{I}|)$ to compute the combined record for $\mathcal{R}(t)$. □

We can now conclude the results in this section.

Proof of Theorem 3. 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 4 to 9. 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 $\mathcal{O}(|\mathcal{I}|)$ (Proposition 1), 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 $\mathcal{O}(|\mathcal{R}(t_1)||\mathcal{R}(t_2)||\mathcal{I}|)$. Because $|\mathcal{R}(t)| \leq (|V||CDC|+2)^{w+1}$, we obtain $\mathcal{O}((|V||CDC|+2)^{2(w+1)}(|\mathcal{I}|)^2) \in (nk)^{\mathcal{O}(w)}$ as the total run-time. □

Lower bounds

We continue with lower bounds for CSP($\mathbf{D}$). We first show that if there are no restrictions 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. We then provide the complementary lower bound result for Theorem 3, showing that CSP($\mathbf{D}_{2,1}$) is already W[1]-hard parameterized by primal treewidth. Our first hardness result is based on the NP-hard problem SUBSET SUM (Garey and Johnson 1979).

SUBSET SUM
Instance: 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 10. 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 $S = \{s_1, \dots, s_n\}$. We construct an equivalent instance $\mathcal{I}$ of CSP($\mathbf{D}_{2,\infty}$) as follows. Introduce $n+1$ variables $x_0, \dots, x_n$. 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. The equivalence of the instances is obvious, since choosing an integer s_i corresponds to setting $x_i - x_{i-1}$ to s_i. □

Our second hardness result is based on a variant of SUBSET SUM. 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)
Instance: Integers k and N, and sets $V_1, \dots, V_k$ and $E_1, \dots, E_K$ of uniform K-dimensional vectors over the natural numbers such that:

- Every vector $\bar{v} \in V_i$ is non-zero only at the coordinates (a, b) such that $a = i$ or $b = i$.
- Every vector $\bar{v} \in E_r$ is non-zero only at the coordinate r.

Parameter: k
Question: Are there $\bar{v}^1, \dots, \bar{v}^k$ and $\bar{e}^1, \dots, \bar{e}^K$ with $\bar{v}^i \in V_i$ and $\bar{e}^i \in E_i$ such that $(\sum_{i=1}^{k} \bar{v}^i) + (\sum_{i=1}^{K} \bar{e}^i) = \bar{N}$?

Proposition 11. MPSS *is strongly* W*[1]-hard (i.e. it is* W*[1]-hard even if all numbers are encoded in unary).*

Sketch of Proof. This follows from the construction presented in the proof of Lemma 6 in (Ganian, Klute, and Ordyniak 2018). □

Theorem 12. CSP($\mathbf{D}_{2,1}$) *is strongly* W*[1]-hard parameterized by primal treewidth.*

Sketch of Proof. We provide a parameterized reduction from MPSS, which together with Proposition 11 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 $\mathcal{I}''$.

Let $\mathcal{I} = (k, N, (V_i)_{1 \leq i \leq k}, (E_r)_{1 \leq r \leq K})$ be the given instance of MPSS. Informally, the main ideas behind the reduction are as follows. First, for every vector $\bar{v}$ in 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 represented by two variables x and y at distance exactly $s(\bar{v})$ from each other. We then create a board consisting of 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. its place inside the bucket part or its place inside the garbage part. To illustrate the idea behind this gadget, assume 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 $2M$ for some number M. Then we can ensure that x is either equal to a or equal to b by using the constraints $x = a \vee x - a > M$ and $x = b \vee b - x > M$.

With that 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 $b_1^r, \dots, b_4^r$ for every coordinate r with the idea that, the place between b_1^r and b_2^r is a placeholder for the r-th coordinate of $\bar{v}^i$, the place between b_2^r and b_3^r is a placeholder the r-th coordinate of $\bar{v}_j$, and the place between b_3^r and b_4^r is a placeholder for the r-th coordinate of $\bar{e}_r$. Finally, to verify that

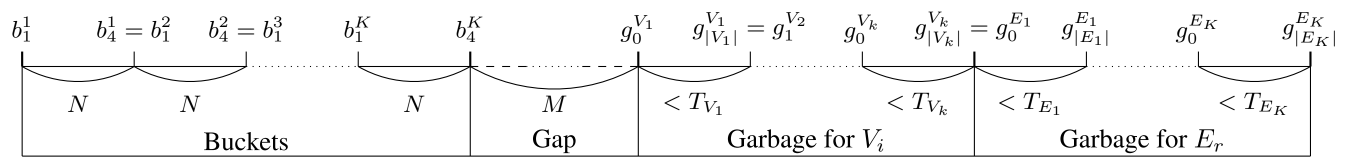

Figure 1: The board consisting of the bucket part and the garbage part defined in the proof of Theorem 12. Here, $T_A = \sum_{\bar{a}\in A} s(\bar{a})$ for $A \in \{V_1, \ldots, V_k, E_1, \ldots, E_K\}$.

the sum of all vectors in the solution is equal to N at each coordinate r, we introduce the constraint $b_4^r - b_1^r = N$.

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 $A \in \{V_1, \ldots, V_k, E_1, \ldots, E_K\}$, 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 $g_0^A, \ldots, g_{|A|}^A$ such that the place between g_{i-1}^A and g_i^A 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 $g_{|A|}^A - g_0^A < \sum_{\bar{a}\in A} s(\bar{a}) = T_A$, which ensures that not all vectors of A can fit into the garbage part.

Figure 1 illustrates the bucket part and the garbage part and also shows how these parts are aligned to each other. Finally, Figure 2 shows the two possibilities (being either in the bucket part or in the garbage part) for the segment representing the non-zero coordinate c of the ℓ-th vector $\bar{v}_\ell$ in V_i. The segment is represented by the two variables $x_{\ell,c}^{V_i}$ and $y_{\ell,c}^{V_i}$ that are at distance exactly $s(\bar{v})$ from each other.

This provides the main ideas behind the first step of the construction, i.e. the construction of the instance $\mathcal{I}''$ of CSP($\mathbf{D}_2$). The primal treewidth of $\mathcal{I}''$ can be shown to be at most $4K + 3$ by observing that the graph obtained

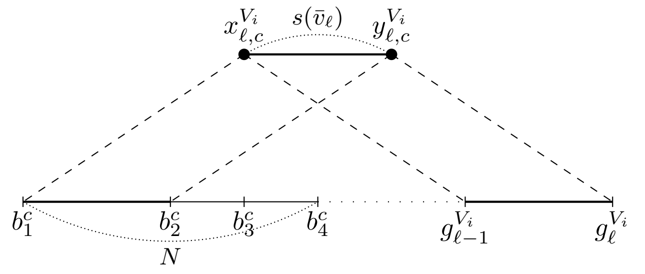

Figure 2: Possible ways to place the variables $x_{\ell,c}^{V_i}$ and $y_{\ell,c}^{V_i}$ corresponding to the non-zero coordinate $c = (i, j)$ of the ℓ-th vector $\bar{v}_\ell$ in V_i.

from primal graph after removing all of the $4K$ bucket variables has treewidth at most 3. Finally, to show the result for CSP($\mathbf{D}_{2,1}$), we need to replace all constraints in $\mathcal{I}''$ that use an integer say z larger than 1. Though this is not possible in general (without increasing the primal treewidth of the instance too much), we show that for our constraints this is possible by replacing those constraints with $\mathcal{O}(z)$ auxiliary constraints and variables that are arranged in a path-like manner. Since z can be assumed to be of size polynomial in the input size (because MPSS is *strongly* W[1]-hard), replacing those constraints with $\mathcal{O}(z)$ auxiliary constraints and variables is achievable in polynomial-time. $\square$

Discussion

We have studied the parameterized complexity (with parameters primal and incidence treewidth) of CSP($\mathbf{D}_{a,k}$), where a is relation arity and k bounds the numerical values. 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 (Barber 2000)). This allows us to 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 zero variable can only increase the treewidth of the incidence graph by 1, so Theorem 3 is still valid for the extended formalism.

This work may be extended in different directions. One way is to more closely analyze the structure of the relations in a given constraint language. This approach has been successful for identifying tractable cases: for instance, the STP disallows disjunctions, while the tractable class by Kumar (2005) is defined via highly restricted disjunctions. It seems likely that a parameterized analysis can also gain from this approach. Another 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 analyzed in a series of articles (Courcelle, Engelfriet, and Rozenberg 1993; Wanke 1994; Courcelle and Olariu 2000). 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 temporal reasoning.

Acknowledgements

The second and the fourth authors 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 2017-04112. Sebastian Ordyniak acknowledges support from the Engineering and Physical Sciences Research Council (EPSRC, project EP/V00252X/1).

References

Audhya, G. K.; Sinha, K.; and Ghosh, S. C. 2011. A Survey on the Channel Assignment Problem in Wireless Networks. *Wireless Communications & Mobile Computing* 11(5): 583–609.

Barber, F. 2000. Reasoning on Interval and Point-Based Disjunctive Metric Constraints in Temporal Contexts. *Journal of Artificial Intelligence Research* 12: 35–86.

Bertelè, U.; and Brioschi, F. 1972. *Nonserial Dynamic Programming*. Academic Press.

Bhargava, N.; and Williams, B. C. 2019. Multiagent Disjunctive Temporal Networks. In *Proc. 18th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS-2019)*, 458–466.

Bliem, B.; Morak, M.; Moldovan, M.; and Woltran, S. 2020. The Impact of Treewidth on Grounding and Solving of Answer Set Programs. *Journal of Artificial Intelligence Research* 67: 35–80.

Bodirsky, M.; and Dalmau, V. 2013. Datalog and Constraint Satisfaction with Infinite Templates. *Journal of Computer and System Sciences* 79(1): 79–100.

Bodirsky, M.; and Kára, J. 2010. The Complexity of Temporal Constraint Satisfaction Problems. *Journal of the ACM* 57(2): 9:1–9:41.

Bodlaender, H. L.; and Kloks, T. 1996. Efficient and Constructive Algorithms for the Pathwidth and Treewidth of Graphs. *Journal of Algorithms* 21(2): 358–402.

Boerkoel, J. C.; and Durfee, E. H. 2013. Decoupling the Multiagent Disjunctive Temporal Problem. In *Proc. 27th AAAI Conference on Artificial Intelligence (AAAI-2013)*.

Carbonnel, C.; and Cooper, M. C. 2016. Tractability in Constraint Satisfaction Problems: A Survey. *Constraints* 21(2): 115–144.

Comin, C.; and Rizzi, R. 2018. On Restricted Disjunctive Temporal Problems: Faster Algorithms and Tractability Frontier. In *Proc. 25th International Symposium on Temporal Representation and Reasoning, (TIME-2018)*, 10:1–10:20.

Courcelle, B.; Engelfriet, J.; and Rozenberg, G. 1993. Handle-Rewriting Hypergraph Grammars. *J. Comput. Syst. Sci.* 46(2): 218–270.

Courcelle, B.; and Olariu, S. 2000. Upper Bounds to the Clique Width of Graphs. *Discrete Applied Mathematics* 101(1–3): 77–114.

Dabrowski, K.; Jonsson, P.; Ordyniak, S.; and Osipov, G. 2020. Fine-Grained Complexity of Temporal Problems. In *Proc. 17th International Conference on Principles of Knowledge Representation and Reasoning (KR-2020)*, 284–293.

Dabrowski, K.; Jonsson, P.; Ordyniak, S.; and Osipov, G. 2021. Solving Infinite-Domain CSPs Using the Patchwork Property. In *Proc. 35th AAAI Conference on Artificial Intelligence (AAAI-2021)*, to appear.

Dechter, R.; Meiri, I.; and Pearl, J. 1991. Temporal Constraint Networks. *Artificial Intelligence* 49(1-3): 61–95.

Downey, R. G.; and Fellows, M. R. 2013. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer.

Flum, J.; and Grohe, M. 2006. *Parameterized Complexity Theory*. Springer.

Ganian, R.; Klute, F.; and Ordyniak, S. 2018. On Structural Parameterizations of the Bounded-Degree Vertex Deletion Problem. In *Proc. 35th Symposium on Theoretical Aspects of Computer Science (STACS-2018)*, 33:1–33:14.

Garey, M.; and Johnson, D. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company.

Gerevini, A.; Saetti, A.; and Serina, I. 2006. An Approach to Temporal Planning and Scheduling in Domains with Predictable Exogenous Events. *Journal of Artificial Intelligence Research* 25: 187–231.

Gottlob, G.; Pichler, R.; and Wei, F. 2006. Bounded Treewidth as a Key to Tractability of Knowledge Representation and Reasoning. In *Proc. 21st National Conference on Artificial Intelligence (AAAI-06)*, 250–256.

Gottlob, G.; Scarcello, F.; and Sideri, M. 2002. Fixed-Parameter Complexity in AI and Nonmonotonic Reasoning. *Artif. Intell.* 138(1-2): 55–86.

Hodges, W. 1997. *A Shorter Model Theory*. New York, NY, USA: Cambridge University Press.

Huang, J.; Li, J. J.; and Renz, J. 2013. Decomposition and Tractability in Qualitative Spatial and Temporal Reasoning. *Artificial Intelligence* 195: 140–164.

Impagliazzo, R.; and Paturi, R. 2001. On the Complexity of k-SAT. *Journal of Computer and System Sciences* 62(2): 367 – 375.

Kloks, T. 1994. *Treewidth: Computations and Approximations*, volume 842 of *LNCS*. Springer.

Kolaitis, P. G.; and Vardi, M. Y. 2000. Conjunctive-Query Containment and Constraint Satisfaction. *Journal of Computer and System Sciences* 61(2): 302–332.

Král', D. 2005. An Exact Algorithm for the Channel Assignment Problem. *Discrete Applied Mathematics* 145(2): 326–331.

Kumar, T. K. S. 2005. On the Tractability of Restricted Disjunctive Temporal Problems. In *Proc. 15th International Conference on Automated Planning and Scheduling (ICAPS-2005)*, 110–119.

Lutz, C.; and Miličić, M. 2007. A Tableau Algorithm for Description Logics with Concrete Domains and General TBoxes. *Journal of Automated Reasoning* 38(1-3): 227–259.

Niedermeier, R. 2006. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press.

Oddi, A.; and Cesta, A. 2000. Incremental Forward Checking for the Disjunctive Temporal Problem. In *Proc. 14th European Conference on Artificial Intelligence (ECAI-2000)*, 108–112.

Pe'er, I.; and Shamir, R. 1997. Satisfiability Problems on Intervals and Unit Intervals. *Theoretical Computer Science* 175(2): 349–372.

Peintner, B.; Venable, K. B.; and Yorke-Smith, N. 2007. Strong Controllability of Disjunctive Temporal Problems with Uncertainty. In *Proc. 13th International Conference on Principles and Practice of Constraint Programming (CP-2007)*, 856–863.

Robertson, N.; and Seymour, P. D. 1984. Graph Minors III. Planar Tree-Width. *Journal of Combinatorial Theory, Series B* 36(1): 49–64.

Samer, M.; and Szeider, S. 2010. Constraint Satisfaction with Bounded Treewidth Revisited. *Journal of Computer and System Sciences* 76(2): 103–114.

Stergiou, K.; and Koubarakis, M. 2000. Backtracking Algorithms for Disjunctions of Temporal Constraints. *Artificial Intelligence* 120(1): 81–117.

Tsamardinos, I.; and Pollack, M. E. 2003. Efficient Solution Techniques for Disjunctive Temporal Reasoning Problems. *Artificial Intelligence* 151(1-2): 43–89.

Venable, K. B.; and Yorke-Smith, N. 2005. Disjunctive Temporal Planning with Uncertainty. In *Proc. 19th International Joint Conference on Artificial Intelligence (IJCAI-2005)*, 1721–1722.

Vilain, M. B.; and Kautz, H. A. 1986. Constraint Propagation Algorithms for Temporal Reasoning. In *Proc. 5th National Conference on Artificial Intelligence (AAAI-1986)*, 377–382.

Wanke, E. 1994. k-NLC Graphs and Polynomial Algorithms. *Discrete Applied Mathematics* 54(2-3): 251–266.