Fine-Grained Complexity of Temporal Problems

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

Abstract

Expressive temporal reasoning formalisms are essential for AI. One family of such formalisms consists of disjunctive extensions of the simple temporal problem (STP). Such extensions are well studied in the literature and they have many important applications. It is known that deciding satisfiability of disjunctive STPs is NP-hard, while the fine-grained complexity of such problems is virtually unexplored. We present novel algorithms that exploit structural properties of the solution space and prove, assuming the Exponential-Time Hypothesis, that their worst-case time complexity is close to optimal. Among other things, we make progress towards resolving a long-open question concerning whether Allen's interval algebra can be solved in single-exponential time, by giving a $2^{O(n \log \log n)}$ algorithm for the special case of unit-length intervals.

1 Introduction

Temporal reasoning is a fundamental task in AI. One of the most influential temporal formalisms is the *simple temporal problem* (STP). The STP was first proposed by Dechter et al. (1991) and it has become the basis for a major part of research on temporal reasoning. Even though STPs are immensely useful in a wide range of applications, their expressive power is limited. A common way of increasing their: remove this expressibility is to introduce disjunctions in various ways (Barber 2000; Dechter, Meiri, and Pearl 1991; Oddi and Cesta 2000; Stergiou and Koubarakis 2000). Such disjunctive STPs have proven to be highly relevant in an AI context (two examples that come to mind are automated planning (Gerevini, Saetti, and Serina 2006; Venable and Yorke-Smith 2005) and multi-agent systems (Bhargava and Williams 2019; Boerkoel and Durfee 2013); a thorough discussion of applications can be found in Stergiou & Koubarakis (2000, Sec. 7)). STPs also have important applications in other areas: we merely point out that unit interval problems (with applications in bioinformatics and graph theory, cf. (Pe'er and Shamir 1997)) and the channel assignment problem (which is a central problem in telecommunications (Audhya, Sinha, and Ghosh 2011)) can be viewed as disjunctive STPs.

While the STP is a polynomial-time solvable problem, the disjunctive versions are typically NP-hard (even though a number of polynomial-time fragments are known, cf. (Kumar 2005)). 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. The framework of *fine-grained complexity* has enabled proving close-to-optimal bounds on time complexity for a multitude of problems assuming suitable conjectures, cf. the textbook by Gaspers (2010). Recent AI examples can be found in planning (Bäckström and Jonsson 2017), constraint satisfaction (Jonsson and Lagerkvist 2017) and in general modelling languages such as integer linear programming (Knop, Pilipczuk, and Wrochna 2019).

The aim of this paper is to present a fine-grained complexity analysis of disjunctive extensions of STPs. Some technical machinery is required to describe the results in detail, so we postpone this description to a later section. To give the reader a flavour of the results, we consider the disjunctive extension of STP proposed by Dechter et al. (1991). Constraints in this problem are of the form $x - y \in I_1 \cup \cdots \cup I_k$, where variables x and y represent points in time and $I_1, \ldots, I_k$ are closed intervals on the real line. Given a set of such constraints, the basic computational problem is to decide whether the variables can be assigned real values that satisfy all the constraints. We show that this problem can be solved in $2^{O(n \log n)}$ time, where n is the number of variables. Furthermore, we obtain a tightly matching lower bound by proving that no algorithm can solve it $2^{o(n \log n)}$ time assuming the *Exponential-Time Hypothesis* (ETH). The ETH was introduced by Impagliazzo et al. (2001) and it is probably the best-known conjecture in fine-grained complexity. It states that the BOOLEAN 3-SATISFIABILITY problem cannot be solved in $2^{o(n)}$ time, where n is the number of variables.

Our results are also relevant outside AI. Allen's Algebra over unit-length intervals is an important formalism with applications in, for example, bioinformatics. Our algorithm for a restricted version of a disjunctive STP allows us to solve the decision problem for this formalism in $2^{O(n \log \log n)}$ time. The best previously known bound, $2^{O(n \log n)}$, can be obtained by viewing the problem as an instance of the existential theory of the reals and solving it using an algorithm due to Renegar (1992).

To a large extent, our results are based on exploiting structural properties of the solution space. Such properties include combinatorial properties of solutions (Lemma 3), compact representations of the solution space (Theorem 5), and amalgamation-like properties (Lemma 16). We stress that we need to introduce novel methods for analysing the fine-grained complexity of CSP problems. The standard method for analysing the complexity of CSP problems—the polymorphism-based algebraic approach—is very well suited for analysing the borderline between tractable and NP-hard problems, but it is not very useful for studying fine-grained complexity. We note that there are extended algebraic approaches (typically based on *partial polymorphisms* (Couceiro, Haddad, and Lagerkvist 2019)) that may be useful in this context, but such approaches are difficult to apply since the theory is underdeveloped for infinite-domain CSPs.

The paper has the following structure. We begin by introducing CSPs and the temporal problems we will study, followed by a summary of our results. Next, we present some general tools and continue by proving our technical results. The paper concludes with a short discussion section.

2 CSPs and Temporal Problems

We begin by defining the CSP.

Definition 1. Let Γ denote a set of finitary relations defined on a set D of values. The *constraint satisfaction problem* over Γ ($\mathrm{CSP}(\Gamma)$) is defined as follows:

INSTANCE. A tuple $(V, \mathcal{C})$, where V is a set of variables and $\mathcal{C}$ 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 $R(v_1, \ldots, v_t) \in \mathcal{C}$?

The set Γ is referred to as a *constraint language*, while the function f is a *satisfying assignment* or simply a *solution*. Observe that we do not require Γ or D to be finite. Given an instance $\mathcal{I}$ of $\mathrm{CSP}(\Gamma)$, 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 — a SAT instance may contain up to 2^{2n} distinct clauses if repeated literals are disallowed — and 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 instead of knowing that it is solvable in $O(2^{\|\mathcal{I}\|})$ time, where $O^*(\cdot)$ hides polynomial factors in n.

The simple temporal problem (Dechter, Meiri, and Pearl 1991) can be viewed as $\mathrm{CSP}(\mathrm{S}^{\leqslant})$, where the constraint language $\mathrm{S}^{\leqslant}$ consists of relations $\{(x, y) \in \mathbb{R}^2 : a \odot_1 x - y \odot_2 b\}$ for any $a \in \mathbb{Z} \cup \{-\infty\}$, $b \in \mathbb{Z} \cup \{\infty\}$ and $\odot_1 = \odot_2 = \leq$. A more general temporal problem, denoted by $\mathrm{CSP}(\mathrm{S})$, allows for strict inequalities in the relations: $\odot_1, \odot_2 \in \{<, \leq\}$. $\mathrm{CSP}(\mathrm{S}^{\leqslant})$ and $\mathrm{CSP}(\mathrm{S})$ are solvable in polynomial time.

One way of augmenting STPs with disjunctions (Oddi and Cesta 2000; Stergiou and Koubarakis 2000) yields a general language D_ω with relations

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

for arbitrary $t, m \geq 1$, $i_\ell, j_\ell \in \{1, \ldots, t\}$ and intervals I_ℓ with endpoints $I_\ell^- \in \mathbb{Z} \cup \{-\infty\}$, $I_\ell^+ \in \mathbb{Z} \cup \{\infty\}$ for all $1 \leq \ell \leq m$. Another way of augmenting STPs with disjunctions (Dechter, Meiri, and Pearl 1991) results in a binary constraint language D_2 containing all binary relations in D_ω. In the literature, this problem is known as the *Disjunctive Temporal Problem*.

To simplify the presentation, we sometimes use an alternative notation for such constraints. A disjunctive constraint $\bigvee_{\ell=1}^{m} x_{i_\ell} - x_{j_\ell} \in I_\ell$ can be represented as a set of simple constraints $\{x_{i_\ell} - x_{j_\ell} \in I_\ell : \ell \in \{1, \ldots, m\}\}$. An assignment satisfies the disjunctive constraint whenever it satisfies at least one simple constraint in the corresponding set.

We conclude by observing that solutions to the temporal problems defined above are invariant under translation, i.e. if $\varphi : V \to \mathbb{R}$ satisfies an instance $(V, \mathcal{C})$, then so does $\varphi'(v) = v + c$ for all $v \in V$ and an arbitrary c. Thus, we can pick any variable in V and assume that its value is zero without loss of generality. We call such a variable a *zero variable*. Augmenting an instance of $\mathrm{CSP}(\mathrm{D}_\omega)$ with a zero variable z allows us to express unary constraints, e.g. the constraint $x - z \in (0, 2]$ is equivalent to $x \in (0, 2]$.

3 Summary of Results

Given a relation $R \in \mathrm{D}_\omega$, let $K(R)$ denote the set of numerical bounds appearing in R, e.g. for $R = \{(x, y, z) \in \mathbb{R}^3 : (-\infty < x - y \leq 3) \vee (0 \leq x - z < 6)\}$ 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 $k(\Gamma) = \max\{|a| : a \in K(\Gamma)\}$, i.e. $k(\Gamma)$ is the least upper bound on absolute values of all numerical bounds appearing in the relations of Γ. For an instance $\mathcal{I} = (V, \mathcal{C})$ of a disjunctive temporal problem, let $n = |V|$ denote the number of variables and let the numerical upper bound be $k = k(\mathcal{C})$.

We let $\mathrm{D}_{\omega,k}$ denote the maximal subset Γ of D_ω such that $k(\Gamma) = k$ and define $\mathrm{D}_{2,k}$ analogously. Note that $\mathrm{D}_{2,k}$ is always a finite language, while $\mathrm{D}_{\omega,k}$ is always infinite. Similarly to the notation used for $\mathrm{S}^{\leqslant}$ and S, we use $\mathrm{D}_2^{\leqslant}$ and $\mathrm{D}_\omega^{\leqslant}$ for the constraint languages that never use strict inequalities, and D_2 and D_ω for the full languages.

We prove the following results:

1. $\mathrm{CSP}(\mathrm{D}_\omega)$ is solvable in $2^{O(nk(\log n + \log k))}$ time (Corollary 6).
2. $\mathrm{CSP}(\mathrm{D}_{\omega,k})$ is solvable in $2^{O(n \log n)}$ time for arbitrary k, and there is a finite constraint language $\Gamma \subseteq \mathrm{D}_{\omega,0}^{\leqslant} \subseteq \mathrm{D}_{\omega,k}^{\leqslant}$ such that $\mathrm{CSP}(\Gamma)$ cannot be solved in $2^{o(n \log n)}$ time assuming the ETH (Theorem 9).
3. $\mathrm{CSP}(\mathrm{D}_2)$ is solvable in $2^{O(n(\log n + \log k))}$ time (Theorem 11), while $\mathrm{CSP}(\mathrm{D}_2^{\leqslant})$ cannot be solved in $2^{o(n \log n)}$ time assuming the ETH (Theorem 12).

4. $\mathrm{CSP}(\mathrm{D}_{2,k})$ is solvable in $2^{O(n \log\log n)}$ time for arbitrary k (Lemmas 19 and 20), while for every $c > 1$, there is a finite constraint language $\Gamma \subseteq \mathrm{D}_2^{\leqslant}$ such that $\mathrm{CSP}(\Gamma)$ cannot be solved in $O(c^n)$ time assuming the ETH (Theorem 15).

Results 2 and 3 prove that the algorithms for $\mathrm{CSP}(\mathrm{D}_{\omega,k})$ and $\mathrm{CSP}(\mathrm{D}_2)$ are close to optimal with respect to worst-case time complexity. Result 4 indicates that there is no uniform single-exponential algorithm for $\mathrm{CSP}(\mathrm{D}_{2,k})$. The result does not, however, rule out the possibility that $\mathrm{CSP}(\mathrm{D}_{2,k})$ can be solved in $2^{c_k \cdot n}$ time, where $c_1, c_2, \dots$ is an increasing sequence.

The problems we consider can be expressed in the existential theory of the reals and solved by Renegar's (1992) algorithm. This leads to a $2^{O(n^3 k)}$-time algorithm for $\mathrm{CSP}(\mathrm{D}_\omega)$ and, consequently, $2^{O(n^3)}$ time for $\mathrm{CSP}(\mathrm{D}_{\omega,k})$. For binary constraint languages this method yields much better results with $2^{O(n(\log n + \log k))}$ time for $\mathrm{CSP}(\mathrm{D}_2)$ and $2^{O(n \log n)}$ time for $\mathrm{CSP}(\mathrm{D}_{2,k})$. We note that the running time for $\mathrm{CSP}(\mathrm{D}_2)$ obtained this way matches our result. However, we claim that our algorithm represents a very simple and natural approach to solving this problem.

4 Normal Forms

We introduce two normal forms on $\mathrm{CSP}(\mathrm{D}_\omega)$ instances: the *standard* and the *reduced* form. The standard form is based on constraint languages $\mathrm{T}_k \subseteq \mathrm{D}_2$, $k \geq 0$, that contain the following relations:

$\{(x,y) \in \mathbb{R}^2 : x - y \in \{i\}\}$ for all $i \in \{0, \dots, k\}$,
$\{(x,y) \in \mathbb{R}^2 : x - y \in (i, i+1)\}$ for all $i \in \{0, \dots, k-1\}$,
$\{(x,y) \in \mathbb{R}^2 : x - y \in (k, \infty)\}$.

Let $\mathcal{I} = (V, \mathcal{C})$ be an instance of $\mathrm{CSP}(\mathrm{D}_\omega)$ and let $k = k(\mathcal{C})$. Any relation used in $\mathcal{C}$ can be expressed as a disjunction of relations in T_k. We illustrate this by an example. Given a simple relation $x - y \in (-\infty, b]$ with $b > 0$, first split it around zero: $y - x \in [0, \infty) \vee x - y \in (0, b]$. Then express $y - x \in [0, \infty)$ as $\bigvee_{R \in \mathrm{T}_k} R(y, x)$. Finally, express $x - y \in (0, b]$ as $\bigvee_{i=1}^{b} x - y \in (i-1, i) \vee \bigvee_{i=1}^{b} x - y \in \{i\}$.

Any instance $\mathcal{I} = (V, \mathcal{C})$ of $\mathrm{CSP}(\mathrm{D}_\omega)$ with $k = k(\mathcal{C})$ can be converted into an equivalent one where the disjuncts are members of T_k. We say that the latter instance is in *standard form*. The standard form of an arbitrary instance of $\mathrm{CSP}(\mathrm{D}_\omega)$ can be computed in polynomial time.

We continue with the *reduced form*. Our starting point is a polynomial-time reduction from $\mathrm{CSP}(\mathrm{D}_\omega)$ to $\mathrm{CSP}(\mathrm{D}_{\omega,1})$. An important observation here is that the reduction introduces relatively few additional variables.

Lemma 1. *For every instance $\mathcal{I} = (V, \mathcal{C})$ of $\mathrm{CSP}(\mathrm{D}_\omega)$ with $k = k(\mathcal{C})$ there is an equivalent instance $\mathcal{I}' = (V', \mathcal{C}')$ with $k(\mathcal{C}') = 1$ and $|V'| = (k+1)|V|$.*

Proof. Assume that $\mathcal{I}$ is in standard form. Define the set of variables $V' = \{v^{(\delta)} : v \in V \text{ and } 0 \leq \delta \leq k\}$. We start by defining the set $\mathcal{C}'_s$ of *structural constraints*. For all $v \in V$ and $\delta \in \{1, \dots, k\}$, add the constraint $v^{(\delta)} - v^{(\delta-1)} = 1$ to $\mathcal{C}'_s$. Observe that these constraints ensure that $v^{(\delta)} - v^{(0)} = \delta$ for every δ.

Next, we define a set $\mathcal{C}'_\iota$ of *instance-specific constraints* Consider an arbitrary $C \in \mathcal{C}$. We will construct an equivalent constraint C' over the variables V' using the relations from T_1. For each simple constraint $\tau(v, w)$ in C over a pair of variables $v, w \in V$, use the following rules to convert it into $\tau'(v, w)$:

$$v - w \in \{i\} \quad \longleftrightarrow \quad v^{(0)} - w^{(i)} = 0 \qquad \text{(R1)}$$
$$v - w \in (i, i+1) \quad \longleftrightarrow \quad v^{(0)} - w^{(i)} \in (0, 1) \qquad \text{(R2)}$$
$$v - w \in (k, \infty) \quad \longleftrightarrow \quad v^{(0)} - w^{(k)} \in (0, \infty) \qquad \text{(R3)}$$

We add $\tau'(v, w)$ to C'. After converting all simple constraints in C we add C' to $\mathcal{C}'_\iota$. Finally, we set $\mathcal{C}' = \mathcal{C}'_s \cup \mathcal{C}'_\iota$.

Observe that rules R1, R2, R3 ensure that φ satisfies a simple constraint $\tau(v, w)$ if and only if φ' satisfies $\tau'(v, w)$. Hence, the instances $\mathcal{I}$ and $\mathcal{I}'$ are equivalent. □

We say that the instance $\mathcal{I}'$ as defined in Lemma 1 is in *reduced form*. To convert from standard form to reduced form, one can apply rules R1, R2, R3 to each simple constraint, following the instruction from left to right (we call this procedure ReducedForm). The reverse conversion is done by applying these rules in the opposite direction (we call this procedure InvReducedForm). Note that both conversion procedures require polynomial time.

An assignment φ' to an instance $\mathcal{I}'$ in reduced form is said to be *valid* if it satisfies all the structural constraints of the instance. Note that a valid assignment does not necessarily satisfy the instance.

5 Enumeration of Certificates

The goal of this section is to present a method for enumerating compact representations of solutions to $\mathrm{CSP}(\mathrm{D}_\omega)$ instances. Let $\mathcal{I} = (V, \mathcal{C})$ be an instance of $\mathrm{CSP}(\mathrm{D}_\omega)$. Define $\mathcal{U}(\mathcal{C}) = \bigcup_{C \in \mathcal{C}} C$ to be the set of all simple constraints in $\mathcal{C}$. Let $\varphi : V \to \mathbb{R}$ be an assignment to $\mathcal{I}$. We identify φ with the subset of constraints $F \subseteq \mathcal{U}(\mathcal{C})$ satisfied by φ. This allows us to define an equivalence relation on the assignments: $\varphi_1 \sim \varphi_2$ 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 $C \in \mathcal{C}$. 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: there are at most as many certificates as there are subsets of $\mathcal{U}(\mathcal{C})$.

We begin this section by introducing ordered partitions. Next, we establish certain connections between certificates and ordered partitions, which form the basis for our enumeration algorithm presented at the end of the section.

5.1 Ordered Partitions

An *ordered partition* of a finite set S is a sequence of nonempty disjoint subsets $(S_1, \dots, S_\ell)$ such that $\bigcup_{i=1}^{\ell} S_i = S$. An ordered partition implicitly defines a ranking function

$r : S \to \{1, \dots, \ell\}$, where $r(x) = i$ for every $x \in S_i$. We write $x \odot_r y$ to denote that $r(x) \odot r(y)$ when $\odot \in \{<, =, >\}$.

Any function $f : S \to X$, where X is a totally ordered set, induces an ordered partition on S with the ranking function r such that $a \odot_r b$ if and only if $f(a) \odot f(b)$ for all $a, b \in S$ and $\odot \in \{<, =, >\}$.

The total number of ordered partitions of an n-element set is counted by the n'th Fubini number $F(n)$ (also known as the n'th ordered Bell number). The value of $F(n)$ is closely approximated by $(n!/2)(\log_2 e)^{n+1}$ (Gross 1962). For our purposes the rough upper bound $F(n) \leq n^n$ is sufficient.

Generating all (unordered) partitions of a set $\{1, \dots, n\}$ takes $O(1)$ amortized time per partition (Ichiro 1984) and generating all permutations takes $O(1)$ time per permutation (Sedgewick 1977). Hence, we have the following:

Proposition 2. *All ordered partitions of an n-element set can be enumerated in $O(n^n)$ time.*

5.2 Valid Assignments

Let $\mathcal{I} = (V, \mathcal{C})$ be an instance of $\mathrm{CSP}(\mathrm{D}_\omega)$ and let $\mathcal{I}' = (V', \mathcal{C}')$ be $\mathcal{I}$ in reduced form. Consider an ordered partition Φ on V'. We say that Φ is *valid* if there is a valid assignment $\varphi' : V' \to \mathbb{R}$ that induces this ordered partition. The following lemma characterizes valid ordered partitions:

Lemma 3. *An ordered partition $\Phi = (\Phi_1, \dots, \Phi_\ell)$ on V' with a ranking function r is valid if the following hold:*

1. $v^{(\delta-1)} <_r v^{(\delta)}$ for $v \in V$ and $\delta \in \{1, \dots, k\}$;
2. $v^{(0)} \odot_r w^{(0)} \iff v^{(\delta)} \odot_r w^{(\delta)}$ for $v, w \in V$, $\delta \in \{1, \dots, k\}$ and $\odot \in \{<, =, >\}$.

Proof. An assignment φ' inducing Φ on V' should be constant on each subset in Φ, so we write $\varphi'(\Phi_i)$ to denote the value of $\varphi'(x)$ for any $x \in \Phi_i$. Consider the sequence $\boldsymbol{d} = (d_1, \dots, d_{\ell-1})$, where $d_i = \varphi'(\Phi_{i+1}) - \varphi'(\Phi_i)$. Note that $\boldsymbol{d}$ defines φ' uniquely up to translation. Thus, to prove that φ' exists it suffices to construct $\boldsymbol{d}$ with positive entries respecting the following property: if $v^{(\delta-1)} \in \Phi_i$ and $v^{(\delta)} \in \Phi_j$ for any v and δ, then $\sum_{s=i}^{j-1} d_s = 1$. This ensures validity of φ':

$$\begin{aligned}\textstyle\sum_{s=i}^{j-1} d_s &= \textstyle\sum_{s=i}^{j-1} \varphi'(\Phi_{s+1}) - \varphi'(\Phi_s) = \\ &= \varphi'(\Phi_j) - \varphi'(\Phi_i) = \\ &= \varphi'(v^{(\delta)}) - \varphi'(v^{(\delta-1)}) = 1\end{aligned}$$

Note that $i < j$ by Condition 1 on Φ. Essentially, we have a system of linear equations over the entries of $\boldsymbol{d}$ and need to show that it has a positive solution. In matrix form, the problem is to solve $\boldsymbol{Md} = \mathbb{1}$ with $\boldsymbol{d} > \mathbb{0}$.

The matrix $\boldsymbol{M}$ has a very specific structure. First, note that the entries in $\boldsymbol{M}$ are either 0 or 1. Each row encodes a linear constraint equivalent to $\varphi'(v^{(\delta)}) - \varphi'(v^{(\delta-1)}) = 1$ for some v and δ. Remove duplicate rows and permute the rest in the increasing order of $r(v^{(\delta)})$. Let s_i and e_i be the maximal and minimal indices of nonzero entries in row i, respectively. Note that all entries between s_i and e_i are also nonzero. For a successive pair of rows i and $i+1$ we have $s_i < s_{i+1}$ and $e_i < e_{i+1}$ by Condition 2. We say that $\boldsymbol{M}$ is a *staircase matrix* (see Figure 1).

$$\begin{bmatrix} \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 & 0 \\ 0 & 0 & \mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 \\ 0 & 0 & 0 & \mathbf{1} & \mathbf{1} & \mathbf{1} & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathbf{1} & \mathbf{1} \end{bmatrix}$$

Figure 1: An example of a staircase matrix

We show inductively that $\boldsymbol{Md} = \mathbb{1}$ has a positive solution when $\boldsymbol{M}$ is a staircase matrix. Let $R_i = \{s_i, \dots, e_i\}$ be the set of indices of nonzero entries in row i. To satisfy the linear equation for the first row, set $d_j = \frac{1}{|R_1|}$ for all $j \in R_1$. For row $i > 1$, consider $X = R_{i-1} \setminus R_i$, $Y = R_{i-1} \cap R_i$ and $Z = R_i \setminus R_{i-1}$. By the staircase property, neither X nor Z is empty. Compute $S_X = \sum_{x \in X} d_x$ and set $d_z = \frac{S_X}{|Z|}$ for all $z \in Z$. By the inductive hypothesis, $S_X > 0$, so the assigned values are positive. Furthermore, $\sum_{j \in R_i} d_j = \sum_{y \in Y} d_y + \sum_{z \in Z} d_z = \sum_{y \in Y} d_y + S_X = 1$, so the linear equation for row i is satisfied. ☐

Now we show that it is possible to recover a certificate of an assignment given the ordered partition it induces.

Lemma 4. *Let φ' be a valid assignment to $\mathcal{I}' = (V', \mathcal{C}')$. Let $F' \subseteq \mathcal{U}(\mathcal{C}')$ be the certificate of φ' and suppose φ' induces an ordered partition Φ on V'. There is a polynomial-time algorithm that computes F' provided $\mathcal{I}'$ and Φ as input.*

Proof. Let r be the ranking function defined by Φ and consider an arbitrary constraint in $\mathcal{U}(\mathcal{C}')$. If it is a structural constraint, then it is included in F' since φ' is valid. Otherwise, there are three possible cases. If the constraint is $v^{(0)} - w^{(i)} \in \{0\}$, then φ' satisfies it if and only if $v^{(0)} =_r w^{(i)}$. If the constraint is $v^{(0)} - w^{(i)} \in (0, 1)$, then φ' satisfies it if and only if $w^{(i)} <_r v^{(0)} <_r w^{(i+1)}$. If the constraint is $v^{(0)} - w^{(k)} \in (0, \infty)$, then φ' satisfies it if and only if $w^{(k)} <_r v^{(0)}$.

These conditions are exhaustive since Φ is valid. Thus, the ranking function r allows us to decide for each constraint in $\mathcal{U}(\mathcal{C}')$ whether it is in F' or not. Clearly, the overall procedure can be carried out in polynomial time. ☐

5.3 Enumeration Algorithm

We present an algorithm that enumerates the certificates to an instance of $\mathrm{CSP}(\mathrm{D}_\omega)$. The output contains a certificate for every equivalence class of satisfying assignments. The algorithm is summarized in Algorithm 1. We refer to the procedure from Lemma 4 computing the certificate given an ordered partition as ORDERTOCONSTR.

Theorem 5. *Algorithm 1 outputs the list of certificates to an instance $\mathcal{I} = (V, \mathcal{C})$ of $\mathrm{CSP}(\mathrm{D}_\omega)$ in $2^{O(nk(\log n + \log k))}$ time, where $n = |V|$ and $k = k(\mathcal{C})$.*

Proof. To see that the algorithm is correct, observe that every satisfying assignment φ to $\mathcal{I}$ defines an ordered partition on V' via φ'. Algorithm 1 enumerates every valid ordered partition, including those defined by the satisfying assignments to $\mathcal{I}$. The rest follows by Lemma 4.

Algorithm 1

```
1: procedure ListCert($\mathcal{I} = (V, \mathcal{C})$)
2:     $\mathcal{I}' = (V', \mathcal{C}') \leftarrow$ ReducedForm($\mathcal{I}$)
3:     $L \leftarrow \varnothing$
4:     for each valid ordered partition $\Phi$ of $V'$ do
5:         $F' \leftarrow$ OrderToConstr($\mathcal{I}', \Phi$)
6:         if $F' \cap C' \neq \varnothing$ for $C' \in \mathcal{C}'$ then
7:             $(V, F) \leftarrow$ InvReducedForm($V', F'$)
8:             $L \leftarrow L \cup \{F\}$
9:     return $L$
```

The running time of the algorithm is dominated by the enumeration of ordered partitions on V'. Checking whether the assignment derived from each ordered partition satisfies $\mathcal{I}$ requires polynomial time. By Proposition 2, the ordered partitions of V' can be enumerated in $O(|V'|^{|V'|})$ time. Substituting $n(k+1)$ for $|V'|$ yields the running time $2^{O(n(k+1)(\log n + \log(k+1)))} \in 2^{O(nk(\log n + \log k))}$. □

Corollary 6. *$\text{CSP}(\text{D}_\omega)$ is solvable in $2^{O(nk(\log n + \log k))}$ time.*

6 Complexity of CSP($\text{D}_{\omega,k}$)

We will present a lower bound on the time complexity of $\text{CSP}(\text{D}_{\omega,k})$ that matches the upper bound implied by the ListCert algorithm. Let $\text{E}_6 \subseteq \text{D}_\omega$ be the maximal constraint language containing relations of arity at most 6 such that $k(\text{E}_6) = 1$. Observe that both the arity of the relations and the parameter $k(\text{E}_6)$ are bounded, so the language is finite. We prove a lower bound on $\text{CSP}(\text{E}_6)$ by a reduction from the $n \times n$ Independent Set problem. An instance of this problem is a graph G with an $n \times n$ grid as the vertex set: $V = \{1, \ldots, n\} \times \{1, \ldots, n\}$. The question is whether G has an independent set with one vertex from each row.

Theorem 7 (Lokshtanov, Marx, and Saurabh)**.** *$n \times n$ Independent Set cannot be solved in $2^{o(n \log n)}$ time unless the ETH fails.*

Lemma 8. *$\text{CSP}(E_6)$ cannot be solved in $2^{o(n \log n)}$ time unless the ETH fails.*

Proof. We reduce from $n \times n$ Independent Set. For brevity, we write $x \geq y$ or $y \leq x$ in place of $x - y \in [0, \infty)$. Given an instance G of this problem, introduce n column variables $c_1, \ldots, c_n$ and the constraints $c_i - c_{i-1} = 1$ for all $i \in \{2, \ldots, n\}$. Next, 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 $j \in \{2, \ldots, n\}$.

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:

$$(r_i \leq c_{j-1}) \vee (r_i \geq c_{j+1}) \vee (r_k \leq c_{\ell-1}) \vee (r_k \geq c_{\ell+1})$$

If G has an independent set I, than 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 $\text{CSP}(\text{E}_6)$ admits a $2^{o(n \log n)}$ algorithm, then so does $n \times n$ Independent Set and this contradicts the ETH by Theorem 7. □

Theorem 9. *$\text{CSP}(\text{D}_{\omega,k})$ can be solved in $2^{O(n \log n)}$ time while $\text{CSP}(\text{D}^{\leqslant}_{\omega,k})$ cannot be solved in $2^{o(n \log n)}$ time unless the ETH fails.*

Proof. Algorithm ListCert solves $\text{CSP}(\text{D}_{\omega,k})$ in $2^{O(nk(\log n + \log k))} = 2^{O(n \log n)}$ time, since k is constant. The language E_6 is a subset of $\text{D}^{\leqslant}_{\omega,k}$ and this gives us the lower bound. □

7 Complexity of CSP(D_2)

We begin by examining a restricted version of $\text{CSP}(\text{D}_2)$ (denoted by w-$\text{CSP}(\text{D}_2)$) where solutions can only take values in the interval $[0, w)$. We show that this problem can be solved in $O^*(w^n)$ time.

Lemma 10. *w-$\text{CSP}(\text{D}_2)$ can be solved in $O^*(w^n)$ time.*

Proof. Let $\mathcal{I} = (V, \mathcal{C})$ be an instance of w-$\text{CSP}(\text{D}_2)$ in standard form and assume $\varphi : V \to [0, w)$ is a satisfying assignment. 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 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 (Jonsson and Bäckström 1998; Koubarakis 2001).

There are w^n possible functions $\varphi_i : V \to \{0, \ldots, w-1\}$ and checking whether a 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 are now ready to prove the upper bound on $\text{CSP}(\text{D}_2)$.

Theorem 11. *$\text{CSP}(\text{D}_2)$ is solvable in $2^{O(n(\log n + \log k))}$ time.*

Proof. First, we prove that a satisfiable instance $\mathcal{I} = (V, \mathcal{C})$ of $\text{CSP}(\text{D}_2)$ has an assignment with width $w < n(k+1)$, where $n = |V|$ and $k = k(\mathcal{C})$.

Consider instance $\mathcal{I}' = (V', \mathcal{C}')$ in reduced form equivalent to $\mathcal{I}$. A satisfying assignment to $\mathcal{I}$ induces an ordered partition $\Phi = (\Phi_1, \ldots, \Phi_\ell)$ on V'. By Lemma 3, we can construct a satisfying assignment φ' from Φ such that $\varphi'(\Phi_\ell)-\varphi'(\Phi_1) \leq \ell-1$, since each $d_i = \varphi'(\Phi_{i+1})-\varphi'(\Phi_i)$ in the proof of the lemma is assigned a value no greater than 1. There are $n(k+1)$ variables in $\mathcal{I}'$, so $\ell \leq n(k+1)$. Now consider the assignment φ to $\mathcal{I}$ obtained from φ' by setting $\varphi(v) = \varphi'(v^{(0)})$ for all $v \in V$. Note that $w = \varphi_{\max} - \varphi_{\min}$, i.e. w is equal to the difference between the maximal and minimal value assigned by φ. Observe also that $\varphi'_{\min} = \varphi_{\min}$ and $\varphi'_{\max} = \varphi_{\max} + k$, thus $w \leq \varphi'_{\max} - \varphi'_{\min} < n(k+1)$.

Finally, observe that the algorithm from Lemma 10 can decide satisfiability of $\mathcal{I}$ in width $n(k+1)$ in $O^*((n(k+1))^n) = 2^{O(n(\log n + \log k))}$ time. □

Next, we prove a lower bound on $\mathrm{CSP}(\mathrm{D}_2^{\leqslant})$.

Theorem 12. $\mathrm{CSP}(\mathrm{D}_2^{\leqslant})$ *is not solvable in* $2^{o(n \log n)}$ *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. The domain of each x_r is set to $\{1,\dots,n\}$ by adding the constraint $\bigvee_{i=1}^{n} x_r - z = i$. We want to constrain x'_r in such a way that $x'_r = nx_r$ for all $r \in \{1,\dots,n\}$. This is achieved by adding the constraints $\bigvee_{i=1}^{n} x'_r - x_r = ni - i$ and $\bigvee_{i=1}^{n} x'_r - z = ni$. Next, for each pair of vertices (a,i) and (b,j) that are adjacent in G, we need to ensure that a solution does not contain both of them. To this end, we add the constraint $x'_a - x_b \in (-\infty, ni-j-1] \vee x'_a - x_b \in [ni-j+1, \infty)$ which is equivalent to setting $x'_a - x_b \neq ni - j$. Observe that $x'_a - x_b = ni - j$ if and only if $x_a = i$ and $x_b = j$, because the mapping $f : \{1,\dots,n\} \times \{1,\dots,n\} \to \{0,\dots,n^2-1\}$ defined as $f(i,j) = ni - j$ is bijective. Clearly, the resulting instance of $\mathrm{CSP}(\mathrm{D}_2^{\leqslant})$ 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 $\mathrm{CSP}(\mathrm{D}_2^{\leqslant})$ 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 7. □

8 Complexity of CSP($\mathrm{D}_{2,k}$)

In this section we consider the time complexity of $\mathrm{CSP}(\mathrm{D}_{2,k})$. We prove the lower bound in the this section and present an algorithm solving $\mathrm{CSP}(\mathrm{D}_{2,k})$ in the next.

Our hardness proof relies on a modification of a result by Traxler (2008). Let d-CSP be the constraint satisfaction problem with domain $D = \{1,\dots,d\}$ and binary relations $R_{a,b} = (x \neq a \vee y \neq b)$ for all $a, b \in D$. Consider a modified version of this problem denoted by d-CSP$^\chi$ with the 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 13. *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 $V_1,\dots,V_\ell$ 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$^\chi$.

Define an instance $\mathcal{I}' = (V', \mathcal{C}')$ 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 $\mathcal{C}$. Assume $x \in V_i$ and i_x is the index of x in V_i. Add constraints $z_i \neq t$ to $\mathcal{C}'$ 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 $\mathcal{C}$. 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$^\chi$. For the proof of equivalence of $\mathcal{I}$ and $\mathcal{I}'$ refer to Lemma 1 in (Traxler 2008). □

Next, we introduce Golomb rulers. A *Golomb ruler* (Golomb 1972) (also known as a *Sidon set* (Sidon 1932)) is a set of integers such that the difference between any pair of its elements is unique. The order of a Golomb ruler is the number of elements in it and the length of a ruler is the difference between its maximal and minimal elements. For example, $\{0,1,4,6\}$ is a Golomb ruler of order 4 with length 6. There is a way to construct Golomb rulers with length quadratic in their order.

Proposition 14 (Erdős and Turán)**.** *Let* $p \geq n$ *be an odd prime. Then* $G_n = \{pa + (a^2 \bmod p) : a \in \{0,\dots,n-1\}\}$ *is a Golomb ruler.*

We are now ready to prove the lower bound.

Theorem 15. *If we assume that the ETH holds, then for arbitrary* $m > 0$ *there is an integer* k *such that any algorithm solving* $\mathrm{CSP}(\mathrm{D}_{2,k})$ *requires at least* $O^*(m^n)$ *time.*

Proof. We start by proving that the time complexity of d-CSP$^\chi$ increases with d assuming the ETH. Let $c_d = \inf\{c : \text{there is a } 2^{cn}\text{-time algorithm solving } d\text{-CSP}^\chi\}$. We show that $\lim_{d\to\infty} c_d = \infty$. The 3-COLOURABILITY problem cannot be solved in subexponential time assuming the ETH (Impagliazzo, Paturi, and Zane 2001). 3-CSP$^\chi$ is a generalization of 3-COLOURABILITY, hence $c_3 > 0$. By Lemma 13, 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$^\chi$ with n variables there is an equivalent instance of CSP$(\mathrm{D}_{2,k})$ with $n+1$ variables and $k \in O(d^2)$. Let $\mathcal{I}$ be an instance of d-CSP$^\chi$ with n variables. Construct an instance $\mathcal{I}'$ of CSP$(\mathrm{D}_{2,k})$ as follows. Choose k so that $\{-k,\dots,k\}$ contains a Golomb ruler of order d as a subset. By Proposition 14, one can choose such k to be in $O(d^2)$. Denote the Golomb ruler by G_d. Associate each integer in $\{1,\dots,d\}$ with a unique element of G_d via the bijection $\rho : \{1,\dots,d\} \to G_d$. Introduce zero variable z to express unary relations. For each variable x in $\mathcal{I}$ introduce a new variable v_x. Define $E_x = \{\rho(c) \mid (x \neq c) \in \mathcal{C}\}$. 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 disallowing $v_x - v_y$ to equal $\delta = \rho(a) - \rho(b)$: set $v_x - v_y \in (-\infty, \delta-1] \vee v_x - v_y \in [\delta+1,\infty)$. By the properties of Golomb rulers, this constraint is satisfied if and only if $x \neq \rho(a)$ or $y \neq \rho(b)$.

The reduction introduces only one extra variable. Thus, the lower bound on d-CSP$^\chi$ carries over to CSP$(\mathrm{D}_{2,k})$. $\square$

9 Algorithm for CSP$(\mathrm{D}_{2,k})$

In this section we prove that CSP$(\mathrm{D}_{2,k})$ can be solved in $2^{O(n\log\log n)}$ time. Before we begin, we introduce a refined standard form for instances of CSP(D_2): for any pair of variables, the instance contains at most one disjunctive constraint involving them. This property can be ensured by intersecting all constraints over the same pair of variables. We assume that the instances are given in this standard form throughout this section and write $\sigma_\mathcal{C}(v,w)$ to denote the constraint in $\mathcal{C}$ over variables v and w.

The rest of this section is divided into two parts. First, we study how to decompose the instances of CSP(D_2) in order to solve the smaller parts recursively. The full algorithm is presented in the second part.

9.1 Instance Decomposition

Our algorithm for CSP$(\mathrm{D}_{2,k})$ is based on the divide-and-conquer approach: we split the instances into smaller parts and solve them recursively. A *subinstance* of an instance $\mathcal{I} = (V,\mathcal{C})$ of CSP(D_2) induced by a subset of variables $U \subseteq V$ is an instance $\mathcal{I}_U = (U,\mathcal{C}_U)$, where $\mathcal{C}_U \subseteq \mathcal{C}$ contains the constraints that only involve the variables in U.

Lemma 16. *Let A, B, C be three disjoint sets of variables. Consider an instance $\mathcal{I} = (V,\mathcal{C})$ of CSP$(\mathrm{D}_{2,k})$ with $V = A\cup B\cup C$. Assume φ_1 and φ_2 are satisfying assignments to subinstances $\mathcal{I}_{A\cup B}$ and $\mathcal{I}_{B\cup C}$. If the following conditions hold, then $\mathcal{I}$ is satisfiable:*

1. *For every pair of variables $a \in A$ and $c \in C$, the constraint $\sigma_\mathcal{C}(a,c)$ contains the simple constraint $c - a > k$.*
2. *Assignments φ_1 and φ_2 satisfy the same set of simple constraints over the variables in B.*
3. *There is $T_1 \in \mathbb{R}$ such that $\varphi_1(a) < T_1 \leq \varphi_1(b) < T_1 + k$ for all $a \in A$ and $b \in B$.*
4. *There is $T_2 \in \mathbb{R}$ such that $T_2 \leq \varphi_2(b) < T_2 + k \leq \varphi_2(c)$ for all $b \in B$ and $c \in C$.*

Proof. Let $\mathcal{I}' = (V',\mathcal{C}')$ be the instance in reduced form equivalent to $\mathcal{I}$. Define A', B' and C' similarly to V'. Extended assignments φ_1' and φ_2' induce ordered partitions on $A'\cup B'$ and $B'\cup C'$. We merge them into an ordered partition on $V' = A'\cup B'\cup C'$, and prove that it is valid by Lemma 4. Next, we prove that the assignment inducing this ordered partition satisfies $\mathcal{I}'$.

Divide V' into two subsets: $V_1' = A' \cup \{b^{(\delta)} : b \in B, \delta \in \{0,\dots,k-1\}\}$ and $V_2' = \{b^{(k)} : b \in B\} \cup C'$. Let $r_1 : V_1' \to \{1,\dots,w_1\}$ be the ranking function induced by φ_1', and let $r_2 : V_2' \to \{1,\dots,w_2\}$ be the ranking function induced by φ_2'. We define the ranking function $r : V' \to \{1,\dots,w_1+w_2\}$ as follows:

$$r(x) = \begin{cases} r_1(x) & \text{if } x \in V_1', \\ r_2(x) + w_1 & \text{if } x \in V_2'. \end{cases}$$

Claim 1. The ordered partition induced by r on V' is valid.

By Lemma 3, we only need to check two conditions. First, we need to confirm that $x^{(\delta-1)} <_r x^{(\delta)}$ for all $x \in V$ and $\delta \in \{1,\dots,k\}$. Since the relative ordering of elements inside V_1' and V_2' is unchanged, the only case that needs to be considered is that $x \in B$ and $\delta = k$. By construction, $r(x^{(k-1)}) \leq w_1 < w_1 + r_2(x^{(k)})$, so $r(x^{(k-1)}) < r(x^{(k)})$.

Secondly, we need to confirm that $x^{(0)} \odot_r y^{(0)} \iff x^{(\delta)} \odot_r y^{(\delta)}$ for all $x,y \in V$, $\delta \in \{1,\dots,k\}$ and $\odot \in \{<,=,>\}$. As previously noted, the relative ordering inside V_1' and V_2' is unchanged, so we only need to consider the cases when one of the variables is in V_1' and the other one is in V_2':

- for all $a \in A, b \in B$: by construction, we have $a^{(k)} <_r b^{(k)}$; by Condition 3, we also have $a^{(0)} <_r b^{(0)}$;
- for all $a \in A$, $c \in C$: by construction, $a^{(\delta)} <_r c^{(\delta)}$ for all $\delta \in \{0,\dots,k\}$;
- for all $b \in B$, $b' \in B$: by Condition 2, ranking functions r_1 and r_2 agree on the relative ordering of elements in B', so the desired property is ensured both by r_1 and r_2;
- for all $b \in B$, $c \in C$: by construction, $b^{(0)} <_r c^{(0)}$; by Condition 4, $b^{(k)} <_r c^{(k)}$.

Let F' be the certificate to the assignment inducing r on V'.

Claim 2. Certificate F' satisfies $\mathcal{I}'$.

As shown above, the relative ordering of variables in $A'\cup B'$ and $B'\cup C'$ is the same under r as it is under r_1 and r_2, respectively. Hence, F' satisfies $\mathcal{I}'_{A'\cup B'}$ and $\mathcal{I}'_{B'\cup C'}$. For all $a \in A$ and $c \in C$ we have the constraint $c^{(0)} - a^{(k)} \in (0,\infty)$ in F'. Together with Condition 1 this implies that F' satisfies $\mathcal{I}'_{A'\cup C'}$. Therefore, F' satisfies $\mathcal{I}'$.

Combining Claims 1 and 2 completes the proof. $\square$

Let $\varphi : V \to [0,kw)$ be a satisfying assignment of $\mathcal{I}$. We can assume that the minimal value assigned by φ is zero by translational invariance. For notational convenience, we let $w \in \mathbb{N}$, so that the upper bound is a multiple of k. Divide the domain of φ into intervals $[ki-k, ki)$ of width k for every $i \in \{1,\dots,w\}$. Partition the variables in V into disjoint, possible empty subsets $\{V_i : i \in \{0, \dots, w+1\}\}$ defined as follows:

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

for all $i \in \{1, \dots, w\}$, 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.

If V_i has at most $n/\log n$ variables, then we say it is *sparse*, otherwise we say that it is *dense*. The choice of $n/\log n$ as the threshold value is justified by the following proposition:

Proposition 17. *For any $N \leq n/\log n$, the number of ordered partitions of an N-element set is less than 2^n.*

Proof. $F(N) \leq F(n/\log n) < n^{n/\log n} = 2^n$. □

Combined with Proposition 2, this proposition implies that Algorithm 1 can generate a list of satisfying assignments to an instance of $\mathrm{CSP}(\mathrm{D}_\omega)$ with at most $n/\log n$ variables in $O^*(2^n)$ time. In particular, this applies to any subinstance of $\mathcal{I}$ induced by a sparse subset of variables.

We proceed with an observation about the sparse subsets.

Lemma 18. *If $|V| \geq 8$, then one of the following holds:*

1. *There is an index $0 \leq i \leq w+1$ such that V_i is sparse, $|L_i| \geq n/3$ and $|R_i| \geq n/3$.*
2. *There are indices $0 \leq i < j \leq w+1$ 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 $|L_i| + |R_i| = |V| - |V_i| \geq n - n/\log n$. Since $\log n \geq 3$, we have $|L_i| + |R_i| \geq 2n/3$. Thus, for every sparse V_i either $|L_i| \geq n/3$ or $|R_i| \geq n/3$.

Let i be the maximal index such that V_i is sparse and $|R_i| \geq n/3$. Similarly, let j be the minimal index such that V_j is sparse and $|L_j| \geq n/3$. Such indices always exist since V_0 and V_{w+1} are sparse. If $i \geq j$, 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. □

9.2 Algorithm for CSP$(\mathrm{D}_{2,k})$

We refer to the algorithm in Lemma 10 (that solves w-CSP(D_2)) as SOLVEBOUNDED. When enumerating ordered partitions we allow subsets to be empty. New instances are defined by only specifying the constraints, as the corresponding sets of variables can be derived from them.

Lemma 19. *Algorithm 2 solves* $\mathrm{CSP}(\mathrm{D}_{2,k})$.

Proof. Consider an arbitrary instance $\mathcal{I} = (V, \mathcal{C})$ of $\mathrm{CSP}(\mathrm{D}_{2,k})$. We prove the claim by induction based on $|V|$. If the instance has fewer than 8 variables, the claim follows by Theorem 5. Otherwise, assume 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 16 are fulfilled. First, note that subinstances $\mathcal{I}_1$ and $\mathcal{I}_2$ in lines 14 and 15 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 by appending F'_B to both subinstances $\mathcal{I}_1$ and $\mathcal{I}_2$. Conditions 3 and 4 are ensured by the introduction of $b_{\min}$ and the constraints involving it.

Algorithm 2

```
1:  procedure SOLVE($\mathcal{I} = (V, \mathcal{C})$)
2:      if $\varnothing \in \mathcal{C}$ then reject
3:      if $|V| < 8$ then
4:          accept if LISTCERT($\mathcal{I}$) $\neq \varnothing$, else reject
5:      if THREESPLIT($\mathcal{I}$) then accept
6:      if FIVESPLIT($\mathcal{I}$) then accept
7:      reject

8:  procedure THREESPLIT($\mathcal{I} = (V, \mathcal{C})$)
9:      for each 3-partition $(A, B, C)$ of $V$ do
10:         if $|A|, |C| \geq \frac{|V|}{3}$ and $|B| \leq \frac{|V|}{\log|V|}$ and
11:             $c - a > k \in \sigma_{\mathcal{C}}(a, c)$ for $a \in A, c \in C$ then
                // introduce a new variable $b_{\min}$
12:             $\mathcal{I}_{B'} \leftarrow \mathcal{C}_B \cup \{0 \leq b - b_{\min} < k\}_{b \in B}$
13:             for $F_{B'} \in$ LISTCERT($\mathcal{I}_{B'}$) do
14:                 $\mathcal{I}_1 \leftarrow \mathcal{C}_{A \cup B} \cup F_{B'} \cup \{b_{\min} - a > 0\}_{a \in A}$
15:                 $\mathcal{I}_2 \leftarrow \mathcal{C}_{B \cup C} \cup F_{B'} \cup \{c - b_{\min} > k\}_{c \in C}$
16:                 if SOLVE($\mathcal{I}_1$) and SOLVE($\mathcal{I}_2$) then
17:                     accept
18:     reject

19: procedure FIVESPLIT($\mathcal{I} = (V, \mathcal{C})$)
20:     for each 5-partition $(A, B, C, D, E)$ of $V$ do
21:         $X \leftarrow C \cup D \cup E$
22:         if $|A|, |E| < \frac{|V|}{3}$ and $|B|, |D| \leq \frac{|V|}{\log|V|}$ and
23:             $x - a > k \in \sigma_{\mathcal{C}}(a, x)$ for $a \in A$, $x \in X$ and
24:             $e - c > k \in \sigma_{\mathcal{C}}(c, e)$ for $c \in C$, $e \in E$ then
                // introduce new variables $b_{\min}$ and $d_{\min}$
25:             $\mathcal{I}_{B'} \leftarrow \mathcal{C}_B \cup \{0 \leq b - b_{\min} < k\}_{b \in B}$
26:             $\mathcal{I}_{D'} \leftarrow \mathcal{C}_D \cup \{0 \leq d - d_{\min} < k\}_{d \in D}$
27:             for $F_{B'} \in$ LISTCERT($\mathcal{I}_{B'}$) and
28:                 $F_{D'} \in$ LISTCERT($\mathcal{I}_{D'}$) do
29:                 $\mathcal{I}_1 \leftarrow \mathcal{C}_{A \cup B} \cup F_{B'} \cup \{b_{\min} - a > 0\}_{a \in A}$
30:                 $\mathcal{I}_2 \leftarrow \mathcal{C}_{B \cup C \cup D} \cup F_{B'} \cup \{c - b_{\min} > k\}_{c \in C} \cup$
                        $F_{D'} \cup \{d_{\min} - c > 0\}_{c \in C}$
31:                 $\mathcal{I}_3 \leftarrow \mathcal{C}_{D \cup E} \cup F_{D'} \cup \{e - d_{\min} > k\}_{e \in E}$
32:                 $w \leftarrow k(\log|V| + 2)$
33:                 if SOLVE($\mathcal{I}_1$) and SOLVE($\mathcal{I}_3$) and
34:                     SOLVEBOUNDED($w, \mathcal{I}_2$) then
35:                     accept
36:     reject
```

Suppose instead that the procedure FIVESPLIT accepts $\mathcal{I}$. First, note that subinstances $\mathcal{I}_1$ and $\mathcal{I}_3$ in lines 29 and 31 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 10. Observe that Lemma 16 applies to the subinstance induced by $C \cup D \cup E$. We see that Condition 1 is ensured by the check on line 24, Condition 2 is ensured by appending F'_D to both subinstances $\mathcal{I}_2$ and $\mathcal{I}_3$, and Conditions 3 and 4 are ensured by the introduction of

$d_{\min}$ and the constraints involving it. Let $X = C \cup D \cup E$ and consider the subinstance induced by $A \cup B \cup X$. Lemma 16 is applicable and we see that Condition 1 is ensured by the check on line 23, Condition 2 is ensured by appending F'_B to both subinstances $\mathcal{I}_1$ and $\mathcal{I}_2$, and Conditions 3 and 4 are ensured by the introduction of $b_{\min}$ 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, then the algorithm accepts it. By Lemma 18, the instance admits an assignment that splits 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 $A = L_i$, $B = V_i$ and $C = R_i$, where (L_i, V_i, R_i) is a split under the assignment φ. We set $\varphi(b_{\min}) = ki - k$ and observe that φ satisfies instances in lines 12, 14 and 15. 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 $A = L_i$, $B = V_i$, $C = R_i \cap L_j$, $D = V_j$ and $E = R_j$, where $(L_i, V_i, R_i \cap L_j, V_j, R_j)$ is a split under the assignment φ. We set $\varphi(b_{\min}) = ki - k$ and $\varphi(d_{\min}) = kj - k$ and observe that φ satisfies instances in lines 25, 26, 29, 30 and 31. Note that the procedure SOLVE accepts $\mathcal{I}_1$ and $\mathcal{I}_3$ by the inductive hypothesis. By Lemma 18, 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 $\mathcal{I}_2$ in width at most $k(\log n + 2)$. □

Lemma 20. *Algorithm 2 solves instances of* CSP$(\mathrm{D}_{2,k})$ *in* $2^{O(n \log \log n)}$ *time.*

Proof Sketch. Let $T(n)$ be the running time of Algorithm 2 on an instance of CSP$(\mathrm{D}_{2,k})$ 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, $T(n) = T_1(n) + T_2(n) + \text{poly}(n)$, 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 $5^n \cdot 2^{2n} \cdot (2T(\frac{n}{3} + 1) + (k(\log n + 2))^n) \cdot \text{poly}(n)$, 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 27 and 28, $2T(\frac{n}{3} + 1)$ is an upper bound on the running time of the recursive calls in line 33, and $(k(\log n + 2))^n$ comes from the running time of the bounded-width algorithm in line 34. Observe that for sufficiently large values of n we have $(\log n + 2)^n > C \log{(\varepsilon n)}^{\varepsilon n}$ for any constant C and $\varepsilon < 1$. Hence, $(k(\log n + 2))^n$ asymptotically dominates $2T(\frac{n}{3} + 1)$ by our initial hypothesis. Finally, observe that

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

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

10 Allen's Algebra over Unit Intervals

Allen's (1983) interval algebra is a well-known formalism for reasoning about time intervals. We consider a restricted version of it where the intervals are only allowed to have unit length. Recall that I^- and I^+ denote the left and the right endpoints of interval I. *Unit Allen* is a binary constraint language with the following basic 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 $J^- < I^+$
$I\{e\}J$	I equals J	$I^- = J^-$ and $I^+ = J^+$

Note that relations p, m, o admit converses p^{-1}, m^{-1}, o^{-1}, while the relation e is symmetric. *Unit Allen* contains every disjunction of the basic relations. Formally, let $\mathbb{U}$ denote the set of all unit intervals on the real line. *Unit Allen* contains $\{(I, J) \in \mathbb{U}^2 : \bigvee_{r \in S} I\{r\}J\}$ for every $S \subseteq \{p, m, o, e, o^{-1}, m^{-1}, p^{-1}\}$. CSP(*Unit Allen*) is NP-complete (Pe'er and Shamir 1997).

Theorem 21. CSP(*Unit Allen*) *is solvable in* $2^{O(n \log \log n)}$ *time.*

Proof. Observe that every basic relation in *Unit Allen* can be expressed as a relation in $\mathrm{D}_{2,1}$ over the left endpoints of the intervals, i.e.

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

and similarly for the converse relations. Thus, an instance of CSP(*Unit Allen*) can be reduced in polynomial time to an instance of CSP$(\mathrm{D}_{2,1})$ with the same number of variables. Applying Algorithm 2 completes the proof. □

11 Conclusion and Future Work

We initiated the fine-grained complexity analysis of disjunctive extensions of STP and obtained an almost comprehensive picture with respect to the number of variables n and the maximum number k occurring in any constraint. For future work, we would like to close the remaining gaps between our algorithmic lower bounds and upper bounds and consider other natural parameters that are likely to have an influence on the fine-grained complexity of temporal problems, such as the maximum arity of relations and the maximum number of disjuncts in constraints. On a slightly different note, we would like to determine whether Allen's interval algebra can be solved in single-exponential time. We believe that our $2^{O(n \log \log n)}$ algorithm for the case of unit intervals provides a good starting point for this investigation, as it indicates that this should indeed be the case, given the 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. In this regard, we remark that the running time of the bounded-width algorithm (Lemma 10) is the dominant term in the time complexity of our algorithm for *Unit Allen*. Improving this part would reduce the overall time complexity.

Acknowledgements

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 2017-04112.

References

Allen, J. F. 1983. Maintaining knowledge about temporal intervals. *Communications of the ACM* 26(11):832–843.

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.

Bäckström, C., and Jonsson, P. 2017. Time and space bounds for planning. *Journal of Artificial Intelligence Research* 60:595–638.

Barber, F. 2000. Reasoning on interval and point-based disjunctive metric constraints in temporal contexts. *Journal of Artificial Intelligence Research* 12:35–86.

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.

Boerkoel, J. C., and Durfee, E. H. 2013. Decoupling the multiagent disjunctive temporal problem. In *Proc. 27th AAAI Conference on Artificial Intelligence (AAAI 2013)*.

Couceiro, M.; Haddad, L.; and Lagerkvist, V. 2019. Fine-grained complexity of constraint satisfaction problems through partial polymorphisms: A survey. In *Proc. 49th International Symposium on Multiple-Valued Logic (ISMVL 2019)*, 170–175. IEEE.

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

Erdős, P., and Turán, P. 1941. On a problem of Sidon in additive number theory, and on some related problems. *Journal of the London Mathematical Society* 1(4):212–215.

Gaspers, S. 2010. *Exponential Time Algorithms - Structures, Measures, and Bounds*. VDM.

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.

Golomb, S. W. 1972. How to number a graph. In *Graph Theory and Computing*. Elsevier. 23–37.

Gross, O. A. 1962. Preferential arrangements. *The American Mathematical Monthly* 69(1):4–8.

Ichiro, S. 1984. An efficient algorithm for generating all partitions of the set $\{1, 2, \ldots, n\}$. *Journal of Information Processing* 7(1):41–42.

Impagliazzo, R.; Paturi, R.; and Zane, F. 2001. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences* 63(4):512–530.

Jonsson, P., and Bäckström, C. 1998. A unifying approach to temporal constraint reasoning. *Artificial Intelligence* 102(1):143–155.

Jonsson, P., and Lagerkvist, V. 2017. An initial study of time complexity in infinite-domain constraint satisfaction. *Artificial Intelligence* 245:115–133.

Knop, D.; Pilipczuk, M.; and Wrochna, M. 2019. Tight complexity lower bounds for integer linear programming with few constraints. In *Proc. 36th International Symposium on Theoretical Aspects of Computer Science (STACS 2019)*, 44:1–44:15.

Koubarakis, M. 2001. Tractable disjunctions of linear constraints: basic results and applications to temporal reasoning. *Theoretical Computer Science* 266(1-2):311–339.

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.

Lokshtanov, D.; Marx, D.; and Saurabh, S. 2018. Slightly superexponential parameterized problems. *SIAM Journal on Computing* 47(3):675–702.

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.

Renegar, J. 1992. 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. *Journal of Symbolic Computation* 13(3):255–299.

Sedgewick, R. 1977. Permutation generation methods. *ACM Computing Surveys* 9(2):137–164.

Sidon, S. 1932. Ein Satz über trigonometrische Polynome und seine Anwendung in der Theorie der Fourier-Reihen. *Mathematische Annalen* 106(1):536–539.

Stergiou, K., and Koubarakis, M. 2000. Backtracking algorithms for disjunctions of temporal constraints. *Artificial Intelligence* 120(1):81–117.

Traxler, P. 2008. The time complexity of constraint satisfaction. In *Proc. 3rd International Workshop on Parameterized and Exact Computation (IWPEC 2008)*, 190–201. Springer.

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.