Improved Algorithms for Allen's Interval Algebra: a Dynamic Programming Approach

Leif Eriksson*[1] and Victor Lagerkvist†[1]

[1]Department of Computer and Information Science, Linköping University, Linköping, Sweden

Abstract

The constraint satisfaction problem (CSP) is an important framework in artificial intelligence used to model e.g. qualitative reasoning problems such as *Allen's interval algebra* ($\mathcal{A}$). There is strong practical incitement to solve CSPs as efficiently as possible, and the classical complexity of temporal CSPs, including $\mathcal{A}$, is well understood. However, the situation is more dire with respect to running time bounds of the form $O(f(n))$ (where n is the number of variables) where existing results gives a best theoretical upper bound $2^{O(n \cdot logn)}$ which leaves a significant gap to the best (conditional) lower bound $2^{o(n)}$. In this paper we narrow this gap by presenting two novel algorithms for temporal CSPs based on dynamic programming. The first algorithm solves temporal CSPs limited to constraints of arity three in $O^*(3^n)$ time, and we use this algorithm to solve $\mathcal{A}$ in $O^*((1.5922n)^n)$ time. The second algorithm tackles $\mathcal{A}$ directly and solves it in $O^*((1.0615n)^n)$, implying a remarkable improvement over existing methods since no previously published algorithm belongs to $O^*((cn)^n)$ for *any* c. We also extend the latter algorithm to higher dimensions block algebras where we obtain the first explicit upper bound.

1 Introduction

Allen's interval algebra is an influential formalism within qualitative reasoning where the basic domain elements are intervals on the real line. For example, we can say that x *precedes* y, or that x *overlaps* y. These relationships are typically represented by 13 pairwise disjoint and jointly exhaustive *basic relations* and one can then express more complicated relationships between intervals by taking unions of the 13 basic relations. Allen's interval algebra has seen numerous applications within artificial intelligence and related areas, e.g., in planning [2, 8, 17, 20], natural language processing [7, 23], and molecular biology [10]. While deciding consistency of a graph of interval constraints, *the network consistency problem*, over the 13 basic relations is tractable, the problem is well-known to be NP-complete once unions of the basic relations are allowed [1].

Despite this, significant effort has been devoted towards solving real-world instances of the problem, using a wide array of different techniques [18, 11, 24, 21, 22]. Unfortunately, these methods are not sufficient to guarantee improved worst-case complexity, and little is

*leif.eriksson@liu.se
†victor.lagerkvist@liu.se

thus known concerning *fine-grained complexity*. As remarked, the consistency problem is NP-complete, but recent progress in parameterised and fine-grained complexity suggests that NP-hardness is a starting point, rather than the end goal. Hence, we are interested in bounding the running time required to solve the problem from above, by constructing improved algorithms (*upper bounds*), and below, by considering size-preserving reductions from problems where breaking a particular bound is deemed unlikely (*lower bounds*). Thus, taking an expected superpolynomial running time into account, what is the smallest $c > 1$ such that the network consistency problem for Allen's interval algebra is solvable in $O(c^n)$ time, where n denotes the number of intervals? Or is it even feasible to expect a single-exponential running time, or would that contradict any existing results or conjectures? Crucially, questions asking for sharp upper and lower bounds cannot be answered by analysing existing methods, which to the best of our knowledge do not give any improved worst-case figures, but require novel and previously unexplored algorithms.

Before attacking these questions let us first remark that the network consistency problem for Allen's interval algebra can be seen as a special case of an infinite-domain *constraint satisfaction problem* (CSP). We view a CSP instance as a set of atomic *constraints* over a predetermined domain, and wants to know whether all variables can be assigned values without contradicting any constraint. Furthermore, it is common to restrict the constraints that may occur by parameterising the problem by a *constraint language*, and the resulting problem is then written as CSP(Γ). Then the network consistency problem for Allen's interval algebra is nothing else than CSP($\mathcal{A}$) where $\mathcal{A}$ is the set of relations obtained by closing the 13 basic relations in Allen's algebra under union. A related problem is the *temporal* CSP problem where each relation is first-order definable over $(\mathbb{Q}; <)$, and we write $\Gamma_<^{(n)}$ for the set of all temporal relations of arity $n \geq 1$.

It is then known that CSP($\mathcal{A}$) is not solvable in *subexponential time*, i.e., in $2^{o(n)}$ time, unless 3-SAT is solvable in subexponential time [14]. The latter statement is in complexity theory known as the *exponential-time hypothesis* (ETH) [12] and is a popular conjecture for proving superpolynomial lower bounds. Let us also remark that CSP($\mathcal{A}$) can be brute forced by enumerating all possible assignments from pairs of variables to the set of basic relations. This results in a $2^{O(n^2)}$ algorithm, suggesting that there is plenty of room for improvement. Indeed, the problem can be solved in $O^*((2n)^{2n})$ (i.e., in $2^{O(n \log n)}$) time by translating each interval constraint to a temporal constraint in $\Gamma_<^{(4)}$ (over the start- and endpoints of the intervals) and the resulting problem can be solved by enumerating all possible orderings among the variables [13]. However, there is still a large gap between this bound and $2^{o(n)}$, suggesting that at least one of the bounds can be improved further. Interestingly, CSP($\Gamma_<^{(4)}$) *cannot* in general be solved in $2^{o(n \log n)}$ time unless the (randomised) ETH is false [14], showing that this approach has its limits. Another optimistic development is the recent algorithm for CSP($\mathcal{A}$) restricted to *unit intervals*, which solves the problem in $2^{O(n \log \log n)}$ time [6]. Unfortunately, generalising the unit case to the full problem in a manner that yields an improved algorithm appears difficult.

After properly defining the relevant fundamental notions (Section 2) we turn to the problem of constructing improved algorithms for CSP($\mathcal{A}$). We consider two different approaches. First, in Section 3 we consider a novel approach of translating interval constraints to temporal constraints of arity at most 3, based on the idea of first fixing an order on the start points of the intervals, and then carefully translating each interval constraint to a constraint over a relation in $\Gamma_<^{(3)}$. Naturally, this only results in an improved algorithm if CSP($\Gamma_<^{(3)}$) admits an improved algorithm, and given the aforementioned $2^{o(n \log n)}$ lower bound for CSP($\Gamma_<^{(4)}$), it may appear doubtful that this is achievable. However, contrary to intuition, we manage to construct an $O^*(3^n)$ time algorithm for CSP($\Gamma_<^{(3)}$) based on *dynamic programming* (here, and in the sequel, we use the notation $O^*(\cdot)$ to suppress polynomial factors). Dynamic

programming is an influential design principle and has e.g. been used to obtain improved exponential-time algorithms for the Knapsack Problem [3], the k-coloring problem [4], and for the channel assignment problem with limited k channel distance [16], but has to the best of our knowledge, not been applied to CSPs of this type before. We also give a conditional lower bound for $\mathrm{CSP}(\Gamma_<^{(3)})$ and show that it cannot be solved in $O(c^n)$ time for *any* $c < 2$ unless k-coloring, for each $k \geq 1$, is solvable in $O(c^n)$ time. This matches the best concrete lower bound for $\mathrm{CSP}(\mathcal{A})$ [13], making it likely that even faster algorithms should target $\mathrm{CSP}(\mathcal{A})$ directly, rather than being based on reductions to a temporal CSP. In Section 4 we use this insight and attack the full $\mathrm{CSP}(\mathcal{A})$ problem directly, and obtain an $O^*((1.0615n)^n)$ time algorithm utilising dynamic programming. This algorithm uses the fact that if we once know the exact relations of one interval compared to all others and know that these are all allowed relations, we from that point onward only need to remember that said interval has passed. Compared to previously mentioned $O^*((cn)^{2n})$ algorithms, this is a significant improvement, as our algorithm is the first yielding a linear expression in the base of the exponent, compared to a quadratic one. Interestingly, this algorithm can be adapted to give an $O^*(2^{dn}(0.5307n)^{(2d-1)n})$ time algorithm for the *d-dimensional box algebra* by reducing the dimension down to one, and then using the improved algorithm for $\mathrm{CSP}(\mathcal{A})$. To the best of our knowledge, this is the first explicit upper bound for the d-dimensional box algebra, which naturally raises the question of whether even faster algorithms can be obtained by targeting the box algebra directly, rather than going the route via $\mathrm{CSP}(\mathcal{A})$. We discuss this and other open questions in Section 6.

In summary, we not only succeed in strengthening upper bounds for $\mathrm{CSP}(\mathcal{A})$ and related problems, but also show that dynamic programming appears to be a generally usable method for constructing algorithms with theoretical worst-case guarantees for infinite-domain CSPs.

2 Preliminaries

We begin by defining the *constraint satisfaction problem* over a set of relations Γ ($\mathrm{CSP}(\Gamma)$).

Definition 1. *Let Γ be a set of finitary relations defined on a set D of values. 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 a set of constraints of the form $R(v_1, \ldots, v_t)$, where t is the arity of $R \in \Gamma$ and $v_1, \ldots, v_t \in V$.*

QUESTION: *is there a function $f\colon V \to D$ such that $(f(v_1), \ldots, f(v_t)) \in R$ for every $R(v_1, \ldots, v_t) \in C$?*

The set Γ is referred to as a *constraint language*, while the function f is a function *satisfying* an instance I, or simply a *model* for I. Observe that Γ or D may be infinite. Given an instance I of $\mathrm{CSP}(\Gamma)$, we let $||I||$ denote the number of bits required to represent I. Our predominant way of defining useful constraint languages is to first fix a relational structure $\mathcal{R} = (D; R_1, \ldots, R_m)$ of *basic relations* and then consider *first-order reducts* of $\mathcal{R}$, i.e., sets of relations where each relation is defined as the set of models of a first-order formula over $\mathcal{R}$ (with equality).

Definition 2. *A* temporal *constraint language is a first-order reduct of* $(\mathbb{Q}; <)$.

In addition, we write $\Gamma_<^{(k)}$ for the set of all temporal relations of arity $k \geq 1$, use $=$ in infix notation for equality, and write $>$ for the converse of $<$. It is then well-known that $\mathrm{CSP}(\Gamma_<^{(2)})$ is tractable but that $\mathrm{CSP}(\Gamma_<^{(k)})$ is NP-hard for every $k \geq 3$ [5]. Furthermore, note that an instance (V, C) of any temporal problem $\mathrm{CSP}(\Gamma)$ can always be solved in $O^*(|V|^{|V|})$ time by enumerating functions from V to $\{1, \ldots, |V|\}$. This observation can be generalised as follows.

Definition 3. *A finite sequence of non-empty finite sets* $(S_1, \ldots, S_\ell)$ *is an* ordered partition *of a set* S *if* $S_1, \ldots, S_\ell$ *is a partitioning of* S*. The* ranking function r *is the function* $r : S \to \{1, \ldots, \ell\}$ *such that* $r(x) = i$ *for every* $x \in S_i$ *and every* $i \in \{1, \ldots, \ell\}$*.*

For simplicity we will denote $r(x) \odot r(y)$ as $x \odot_r y$ when $\odot \in \{<, =, >\}$. Given any function $f\colon S \to X$, where X is totally ordered, f defines an ordered partition on S as well as a ranking function r such that $u \odot_r v$ if and only if $f(u) \odot f(v)$ for all $u, v \in S$ and $\odot \in \{<, =, >\}$.

The total number of different ordered partition of a set containing n elements is counted by the n'th Ordered Bell Number ($OBN(n)$) (also known as the n'th Fubini number). This number $OBN(n)$ is closely approximated by $(n!/2)(\log_2 e)^{n+1}$, or roughly $(0.5307n)^n \cdot \mathrm{poly}(n)$ after applying Sterling's approximation of $n!$.

Lemma 4. *A model* f *to an arbitrary temporal CSP*(Γ) *instance* $I = (V, C)$ *implies the existence of an ordered partition* $S = (S_1, \ldots, S_\ell)$ *over* V *with ranking function* r*, such that* $x \odot_r y$ *if and only if* $x \odot_f y$ *for all* $x, y \in V$*.*

Proof. Given $f\colon V \to X$, take the ordered partition $(\alpha_1, \ldots, \alpha_\ell)$, where $\alpha_i = \{x_j\}$ for every $x_j \in X$. Define r as $r : V \to \{1, \ldots, \ell\}$ such that $r(v) = i$ if and only if $f(v) \in \alpha_i$, and define $S = (S_1, \ldots, S_\ell)$ as the ordered partition having r as ranking function. If $x \odot_f y$ then $x \odot_r y$ must now be true, and vice versa. □

We close this section by defining *Allen's interval algebra*. Here, the underlying domain consists of intervals on the real line, and we consider the 13 binary relations as described by Table 1. We let $\mathcal{A}$ be the constraint language obtained by closing these 13 basic relations under union, and observe that CSP($\mathcal{A}$) is a reformulation of the network consistency problem for Allen's interval algebra. Strictly speaking, $\mathcal{A}$ is not a temporal language, but recall

Table 1: The 13 basic relations between two intervals on the same line. (i denotes the $i$nverse/converse of a relation.)

Relation	***Illustration***	***Interpretation***
$\boldsymbol{X < Y}$ $\boldsymbol{Y > X}$	XXX YYY	X precedes Y
$\boldsymbol{X = Y}$	XXXXXXX YYYYYYY	X is equal to Y
$\boldsymbol{X\ m\ Y}$ $\boldsymbol{Y\ mi\ X}$	XXX YYY	X meets Y
$\boldsymbol{X\ o\ Y}$ $\boldsymbol{Y\ oi\ X}$	XXXXX YYYY	X overlaps with Y
$\boldsymbol{X\ s\ Y}$ $\boldsymbol{Y\ si\ X}$	XXX YYYYYYY	X starts Y
$\boldsymbol{X\ d\ Y}$ $\boldsymbol{Y\ di\ X}$	XXX YYYYYYY	X during Y
$\boldsymbol{X\ f\ Y}$ $\boldsymbol{Y\ fi\ X}$	XXX YYYYYYY	X finishes Y

from Section 1 that if we treat every interval x as a pair of start- and end-points (x^-, x^+), any CSP($\mathcal{A}$) instance can trivially be transformed to an equisatisfiable CSP($\Gamma_<^{(4)}$)-instance. To simplify statements about start- and end-points we introduce the following two sets:

$V^- = \{x^- \mid (x^-, x^+) \in V\}$, and similarly $V^+ = \{x^+ \mid (x^-, x^+) \in V\}$. Note that $x^- < x^+$ trivially holds for every interval x and that Lemma 4 is also valid with respect to ordered partitions over $V^- \cup V^+$.

3 An $O^*(3^n)$ Algorithm for CSP($\Gamma_<^{(3)}$)

We begin by constructing an improved algorithm for CSP($\Gamma_<^{(3)}$), i.e., the CSP problem with temporal constraints of arity 3. As we will see, this algorithm can be used to obtain an improved algorithm for CSP($\mathcal{A}$), although its running time turns out to be worse than the specialised CSP($\mathcal{A}$) algorithm presented in Section 4.

Note that $\Gamma_<^{(3)}$ is a *symmetric* constraint language in the following sense: for every relation $R \in \Gamma_<^{(3)}$ and permutation $\pi\colon \{1,2,3\} \to \{1,2,3\}$ there exists $R' \in \Gamma_<^{(3)}$ such that a constraint $R(x_1, x_2, x_3)$ holds if and only if $R'(x_{\pi(1)}, x_{\pi(2)}, x_{\pi(3)})$ holds. We will later make use of this symmetry to simplify our notation when proving correctness. Furthermore, recall that Lemma 4 tells us that any solution to a CSP($\Gamma_<^{(3)}$) instance (V, C) directly defines an ordered partition over V.

We proceed by describing our dynamic programming inspired algorithm, using the following notion of *records*. To avoid some tedious situations, we will from now on step away from the formal definition of an ordered partition by allowing the sequences to contain zero to three empty sets. This will have a polynomial increase in the number of possible ordered partitions, which we can safely ignore.

Definition 5. *A* record *for a CSP($\Gamma_<^{(3)}$) instance (V, C) is an ordered partition $(V_<, V_=, V_>)$ of V, with some ranking function r, such that for every $u \in V_<$, every $v \in V_>$, and for all $x, y, z \in V_=$, every constraint $c(i, j, k) \in C$ with $i, j, k \in \{x, y, z, u, v\}$ evaluates to true for r.*

Algorithm 1 DP algorithm for CSP($\Gamma_<^{(3)}$).

```
function 3-Arity-solver(I = (V, C))
    S ← {(∅, ∅, V)}
    for each record (V<, ∅, V>) ∈ S do
        if V> = ∅ then
            return True
        end if
        for each V= ⊆ V> do
            if (V<, V=, V> \ V=) is a record then
                S = S ∪ {(V< ∪ V=, ∅, V> \ V=)}
            end if
        end for
    end for
    return False
end function
```

Our algorithm for CSP($\Gamma_<^{(3)}$) is presented in Algorithm 1. The basic idea is to inductively build records by moving subsets from $V_>$ to $V_<$ via the set $V_=$, while not contradicting any constraints in the instance.

Theorem 6. *Algorithm 1 is correct and has an upper bound on runtime complexity of $O^*(3^n)$.*

Proof. Given a yes-instance $I = (V, C)$, we know from Lemma 4 that there exists some ordered partition $s = (S_1, \ldots, S_\ell)$ with ranking function r, satisfying I. In this case, the algorithm can in every iteration $1 \leq i \leq \ell$ choose $V_=$ to equal S_i since this always results in a new record, and since $s = (S_1, \ldots, S_\ell)$ is an (ordered) partition, we in the last iteration reach the case when $V_> = \emptyset$ and return 'True'. Hence, Algorithm 1 answers 'True' if I is a yes-instance.

In the other direction, if Algorithm 1 answers 'True', there exists a sequence of records (constructed in line 8) $(S_1, \ldots, S_\ell)$ with $S_i = (V_{<,i}, V_{=,i}, V_{>,i})$ and such that $V_{<,i+1} = V_{<,i} \cup V_{=,i}$, $\emptyset \neq V_{=,i+1} \subseteq V_{>,i}$, $V_{>,i+1} = V_{>,i} \setminus V_{=,i+1}$ for all $1 \leq i < \ell$, $V_{>,1} = V$ and $V_{>,\ell} = \emptyset$. Let f be the ranking function for $(V_{=,1}, \ldots, V_{=,\ell})$. Now, $u <_f v$ only if $u <_r v$ for r for some S_i, and $u =_f v$ only if $u =_r v$ for r for all S_i. For every triple $x, y, z \in V$ there will be a record S_i such that one of the following, or a symmetrical case, holds:

- $x \in V_{<,i}$, $y \in V_{=,i}$ and $z \in V_{>,i}$,
- $x, y \in V_{=,i}$ and $z \in V_{>,i}$,
- $x \in V_{<,i}$ and $y, z \in V_{=,i}$, or
- $x, y, z \in V_{=,i}$.

Since S_i is a record, any constraint $c(x, y, z)$ (recall that $\Gamma^{(3)}_<$ is symmetric) must evaluate to true for r for S_i, and hence for f, and since this is true for all $x, y, z \in V$, every constraint in C must evaluate to true for f. Hence, f is a model of I, meaning that I is a 'yes'-instance whenever Algorithm 1 returns 'True'.

For the complexity, every variable in V can be placed in at most one of the three sets $V_<$, $V_=$ and $V_>$, at a time, which creates 3^n possible combinations. Evaluating related constraints in C can be done in poly($||I||$) time, and hence the final upper bound is $O^*(3^n)$. □

Let us now see how Algorithm 1 can be used to obtain an improved algorithm for CSP($\mathcal{A}$). Here, the basic idea is to enumerate partial orders over the start-points V^-, and given such an order we can then simplify the instance and solve it as a CSP($\Gamma^{(3)}_<$) instance. However, since we in Section 4 investigate a better, more direct method, we only provide a proof sketch of the main ideas.

Theorem 7. *Any arbitrary CSP($\mathcal{A}$) instance $I = (V, C)$ is solvable in $O^*((1.5922n)^n)$ time and space.*

Proof. (Sketch) Enumerate all ordered partitions for V^-. For each ordered partition, all constraints in C can be transformed to $\Gamma^{(3)}_<$ constraints and this give a new instance I_3 that also enforces the ordered partition given. The exact input-output mapping of this transform can be calculated by combining the 13 basic relations for $\mathcal{A}$ with the three possible relations between related start-points. These combinations then show if a relation is directly false and if not, what the relations are still needed between x^+, y^+ and either x^- or y^-, to make the original interval relation true. Take the constraint $((x < y) \vee (x\, o\, y) \vee (x\, f\, y))$ together with an ordered partition such that $x^- < y^-$ as an example. This would yield the constraint $(x^+ < y^- \vee (y^- < x^+ \wedge x^+ < y^+) \vee false)$ after the transform. Our new instance I_3 can then be solved with Algorithm 1. Due how the transform works, any ordered partition solving I_3 would now also solve I.

The complexity is given by the product of the number ordered partitions $OBN(n)$ and the cost for solving I_3 using Algorithm 1. As the variables in V^- must follow the same ordering as in the ordered partition used for the transform, the number of possible records are reduced from $3^{|V^-|+|V^+|}$ to less than $3^{|V^+|}(2|V^-| + 1)$, with some very minor modifications to the algorithm. □

Naturally, an even faster algorithm for $\mathrm{CSP}(\Gamma_<^{(3)})$ could improve this even further, making it interesting to investigate how far we could push this bound. To this aim, consider the following reduction from the *k-coloring problem* (kCP): given a graph (V, E) we begin by introducing k fresh variables $\{c_1, \ldots, c_k\}$ and the constraints $(c_1 < c_2)$, ..., $(c_{k-1} < c_k)$. For every vertex $x_i \in V$ we then introduce the constraints $(c_1 < x_i \vee c_1 = x_i) \cup (c_k > x_i \vee c_k = x_i)$ and $\bigcup_{j=1}^{k-1}(c_j > x_i \vee c_j = x_i \vee c_{j+1} = x_i \vee c_{k+1} < x_i)$, ensuring that every x_i is equal to one c_j. Last, for each edge $(x, y) \in E$ we introduce a constraint $(x < y \vee x > y)$. It is easy to see that this reduction is correct, and put together, an $O(c^n)$ time algorithm for $\mathrm{CSP}(\Gamma_<^{(3)})$ for $c < 2$ would then imply an $O(c^n)$ time algorithm for kCP for *every* $k > 0$, and would thus beat the currently leading $O^*(2^n)$ time algorithm by Björklund and Husfeldt [4].

4 Allen's Interval Algebra

We now present an $O^*((1.0615n)^n)$ algorithm for $\mathrm{CSP}(\mathcal{A})$, and thus beat the $O^*((1.5922n)^n)$ algorithm from Section 3 by a substantial margin. Similar to $\Gamma_<^{(3)}$, $\mathcal{A}$ is *symmetric* in the following sense: for every $R \in \mathcal{A}$ there exists $R' \in \mathcal{A}$ such that a constraint $R(x_1, x_2)$ holds if and only if $R'(x_2, x_1)$ holds. This simplifies our notations when talking about individual constraints, as there is no need to have different cases for $R(x_1, x_2)$ and $R'(x_2, x_1)$. Recall from Lemma 4, and the subsequent remark, that any solution to a $\mathrm{CSP}(\mathcal{A})$ instance (V, C) directly defines an ordered partition over $V^- \cup V^+$. We begin by defining the analogous notion of a record.

Definition 8. *A* record *for a CSP($\mathcal{A}$) instance (V, C) is a tuple $(V_{ok}, V_1, \ldots, V_\ell, V_>)$ such that*

1. *$(V_1, \ldots, V_\ell, V_>)$ is an ordered partitioning of $U \subseteq V^- \cup V^+$ with some ranking function r,*
2. *$x^- <_r x^+$ for all $x^-, x^+ \in U$,*
3. *$x^- \in V_{ok}$ if and only if $x^+ \in V_{ok}$, and*
4. *all constraints $c(x, y) \in C$ such that $x^+ \in V_\ell$ and $y^-, y^+ \in V_1 \cup \ldots \cup V_\ell \cup V_>$ evaluate to true for r.*

We will assume that if we create a tuple $(V_{ok}, V_1, \ldots, V_\ell, V_>)$ where some set V_i, $1 \leq i \leq \ell$ is empty, that we instead use the tuple $(V_{ok}, V_1, \ldots, V_{i-1}, V_{i+1}, \ldots, V_\ell, V_>)$, i.e. any empty sets out of V_i, $1 \leq i \leq \ell$ are simply ignored. This assumption simplifies our discussion of tuples and records by hiding unnecessary complications that have no impact on any results. Ignoring empty sets in this manner is often needed for $(V_1, \ldots, V_\ell, V_>)$ to be an ordered partition.

The point of records is to give us full knowledge of any start- or end-point in V_i, $1 \leq 1 \leq \ell$, while simultaneously allowing us to maintain minimal knowledge about variables we have already made sure are positioned such that any related constraints are satisfied. Similarly, records also provides information about the start- and end-points we have yet to place. Our algorithmic approach is now to build tuples inductively from previous records, and to check if these tuples are records. See Algorithm 2 for a full description. The main idea is that once an interval has had both start- and end-point placed, and we have made sure that all constraint that this interval was part of are satisfied, it does no longer matter exactly where the interval was placed, we can safely remove the interval from the ordered partition, and thus reduce the amount of information that needs to be stored. The algorithm thus keeps track of the relations between all partially placed intervals, while ignoring the positions of all already placed, and compared, ones. For each end-point we are currently placing at line

8, we now know the relation to all other unplaced, or partially placed, intervals. For every interval placed before the ones we are currently placing, we have already passed through the same check, and can hence guarantee the any constraints containing at least one of those intervals have already been satisfied.

Algorithm 2 DP algorithm for CSP($\mathcal{A}$).

```
1: function DP-IA-SOLVER(I = (V, C))
2:     Let S ← {(∅, V⁻ ∪ V⁺)}
3:     for each record (V_ok, V_1, ..., V_ℓ, V_>) ∈ S do
4:         if V_> = ∅ then
5:             return 'True'
6:         end if
7:         for each set v ⊆ V_> such that either
               x⁻ ∈ v and x⁺ ∉ v, or
               x⁺ ∈ v and x⁻ ∉ (V_> ∪ v)
           do
8:             if (V_ok, V_1, ..., V_ℓ, v, V_> \ v) is a record then
9:                 Let t ← {x⁻, x⁺ | x⁺ ∈ v ∩ V⁺ and
                             x⁻ ∈ (V_1 ∪ ... ∪ V_ℓ) ∩ V⁻}
10:                S ← S ∪ (V_ok ∪ t, V_1 \ t, ..., V_ℓ \ t,
                             v \ t, V_> \ v)
11:            end if
12:        end for
13:    end for
14:    return 'False'
15: end function
```

Lemma 9. *Algorithm 2 returns 'True' if and only if a given CSP($\mathcal{A}$) instance $I = (V, C)$ is a yes-instance.*

Proof. If I is a yes-instance, we know from Lemma 4 that there exists some ordered partition $(S_1, \ldots, S_\gamma)$ with ranking function r satisfying I. Choosing v, at Line 7, such that $v = S_i$ when we have a record $(V_{ok}, V_1, \ldots, V_{i-1}, V_>)$ with $V_j \subseteq S_j$ for all $1 \leq j \leq i-1$ and $V_> = S_i \cup \ldots \cup S_\ell$, will construct the new record $(V_{ok}, V_1, \ldots, V_{i-1}, v, V_> \setminus v)$, at Line 8. Specifically the ordered partition $(V_{ok}, V_1, \ldots, V_{i-1}, v, V_> \setminus v)$ with ranking function f, will always be such that $x^\pm \odot_f y^\pm$ if $x^\pm \odot_r y^\pm$, for all $x^\pm \in v$ and $y^\pm \in (V^+ \cup V^-)$, and also for all $x^\pm \in V_1 \cup \ldots \cup V_{i-1} \cup v$, and $y^\pm \in (V^+ \cup V^-) \setminus V_{ok}$. Hence, the ranking functions agree for all cases where at least one variable is in $V_1 \cup \ldots \cup V_{i-1} \cup v$. Since the algorithm does not move pairs (x^-, x^+) to V_{ok} until $x^+ \in v$, i.e. $x^- \in V_j$ for some j until $x^+ \in v$ occurs, $x^+ \odot_f z$ if and only if $x^+ \odot_r z$ for all $z \in (V^- \cup V^+)$, and similarly for x^-. Given the definition of r, all constraints in C evaluates to true for r, so all constraints $c(x, y)$ with $y^+, y^- \in (V^- \cup V^+) \setminus V_{ok}$, will evaluate to true for f, too. Since all intervals in $x \in V$ will have $x^+ \in v$ at one point when constructing records in this manner, every constraint in C evaluate to true for some f. Hence, a record where $V_> = \emptyset$ will be reached, and the algorithm will return 'True'.

If Algorithm 2 returns 'True', there exists a sequence of records $S_{\text{seq}} = (S_1, \ldots, S_\ell)$, constructed at Line 8, going from $(\emptyset, V^- \cup V^+)$ to some $(V_{end}, \ldots, \emptyset)$. Let $v_{\text{seq}} = (v_1, \ldots, v_\ell)$ with ranking function r, be the sequence of the v sets chosen at Line 7 corresponding

to S_{seq}. For any pair of intervals $x, y \in V$ with $x^+ \leq_r y^+$, we will have a record $s = (V_{ok}, V_1, \ldots, v, V_>) \in S_{\text{seq}}$ with $x^+ \in v$ and $y^+ \in (v \cup V_>)$. Let f be the ranking function for $(V_1, \ldots, v, V_>)$, any constraint $c(x, y) \in C$ will evaluate to true for f, since s is a record. For all $z, w \in V_1 \cup \ldots \cup v \cup \{y^+\}$ then $z \odot_f w$ implies $z \odot_r w$ given the ordering we choose our v sets. Since this is true for any pair $x, y \in V$, r is a model for I. Hence, Algorithm 2 returns 'True' if and only if I is a yes-instance.

□

Lemma 10. *Algorithm 2 has an upper bound on run time complexity of $O^*(2^n OBN(n))$ or $O^*((1.0615n)^n)$ for an n-variable CSP($\mathcal{A}$) instance $I = (V, C)$.*

Proof. Checking whether a tuple is a record in line 8 is obviously polynomial. The size of $V_1 \cup \ldots \cup V_\ell$ in a record stored in S, can never grow bigger than n, as it can only contain subsets of V^-. There are $\binom{n}{m}$ ways to select m variables out of V^- for $V_1 \cup \ldots \cup V_\ell$, and then $OBN(m)$ different ordered partitions of those m variables. The remaining $n - m$ variables from V^- can either be placed into V_{ok} or into $V_>$, in 2^{n-m} different ways. Similarly, the m variables in V^+ bound to the m chosen variables from V^-, can be placed in either v (always at the very end of the ordered partition, so no extra ordering cost for these), or in $V_>$, giving another 2^m. This gives the equation $\sum_{m=0}^{n} \binom{n}{m} OBN(m) 2^m$. Simplifying this, and ignoring polynomial factors, gives

$$\sum_{m=0}^{n} \frac{\frac{n}{e}^{n}}{\frac{m}{\cancel{e}}^{\cancel{m}} \frac{n-m}{e}^{n-m}} \frac{\cancel{m}}{\cancel{e} \ln 2}^{m} 2^n < n 2^n OBN(n)$$

which concludes the proof. □

5 Higher dimensions Algebra

The *d-dimensional block algebra* is a well-known extension of the interval algebra where the basic objects are d-dimensional boxes, related by interval constraints over the respective dimensions [9].

Definition 11. *The d-dimensional block algebra ($\mathcal{A}^d$) is a constraint language of arity two, where each variable represents a set of d independent intervals $X = (X_1, ..., X_d)$. Each basic relation is as a set of d relations from $\mathcal{A}$, one over each dimension, such that two different dimensions are never compared.*

For example, the $\mathcal{A}^2$ relation $X <, = Y$ means that X precedes Y in the first dimension, and equals Y in the second. While literature specifically about $\mathcal{A}^d$ becomes more sparse for larger values of d, using $\mathcal{A}^2$ to describe relations in the plane is not unheard of, e.g. [19, 25, 15]. Intuitively, we can also see how $\mathcal{A}^3$ and $\mathcal{A}^4$ can be used for relations in a three-dimensional space, or in a three-dimensional space together with a time dimension.

Inspired by the improved algorithm for CSP($\mathcal{A}$) from Section 4 we now turn to the problem of constructing an improved algorithm for CSP($\mathcal{A}^d$) for every $d \geq 2$. This problem can be brute forced by enumerating all possible orderings over the start- and end-points in the induced intervals, but to the best of our knowledge no faster algorithm has been detailed in the literature.

For CSP($\mathcal{A}^d$) we will treat the variables in V as tuples containing one pair of start- and end-points for each dimension, i.e. $x = (x_1^-, x_1^+, ..., x_d^-, x_d^+) \in V$. In addition, we extend the sets V^+ and V^- to also contain the numbering of the dimension, i.e., let $V_i^- = \{x_i^- \mid (x_1^-, x_1^+, ..., x_d^-, x_d^+) \in V\}$, $V_i^+ = \{x_i^+ \mid (x_1^-, x_1^+, ..., x_d^-, x_d^+) \in V\}$, $0 < i \leq d$. Then, similar to how a solution to a CSP($\mathcal{A}$) instances defines an ordered partition over $V^- \cup$

V^+, a solution to a CSP($\mathcal{A}^d$) instance defines d ordered partitions, one for each pair $V_i^- \cup V_i^+$, $0 < i \leq d$. Our idea is then to enumerate all ordered partitions for all dimensions but one, and then solve the remaining instance as a CSP($\mathcal{A}$) instance. If the dimensions is $d \leq 1$, this leads to a complexity of $O^*(2^{dn} OBN(n)^{2d-1})$ using Algorithm 2 to solve the constructed CSP($\mathcal{A}$) instances.

Theorem 12. *CSP($\mathcal{A}^d$) is solvable in $O^*(2^{dn} OBN(n)^{2d-1})$ time.*

Proof. While $d > 1$, we enumerate over each ordered partitions over $V_d^- \cup V_d^+$ such that $x^- < x^+$ is true for all $x \in V$. For each such ordered partition we can, as all relations in that dimension are known, transform I to a new CSP($\mathcal{A}^{d-1}$) instance I_{d-1}. We then repeat this recursively until $d = 1$ and then solve I_1 using Algorithm 2. Hence, if Algorithm 2 returns 'True' for some I_1 we have a set of d ordered partitions that satisfies I. Similarly, since we enumerate all possible combinations of ordered partitions until we reach I_1, and since we know from Lemma 9 that Algorithm 2 is correct, it is possible to find a combination of ordered partitions satisfying I if I is a yes-instance.

For the complexity, there are $OBN(2n)^{d-1}$ ordered partitions over $V_i^- \cup V_i^+$ for $1 < i \leq d$. However, only one in $2^{(d-1)n}$ of these ordered partitions will obey $x_i^- < x_i^+$ for all $x_i \in V$ and $1 < i \leq d$. Since the complexity of Algorithm 2 is in $O^*(2^n OBN(n))$ (Lemma 10) we obtain a run time of $O^*(2^{dn} OBN(n)^{2d-1})$ or $O^*(2^{dn}(0.5307n)^{(2d-1)n})$ after a few simplifications. □

6 Discussion and Conclusion

We studied the fine-grained complexity of temporal CSPs, with a particular focus on Allen's interval algebra, where we managed to obtain an $O^*((1.0615n)^n$ time algorithm. As remarked, this is a *vast* improvement over the $2^{O(n \log n)}$ time algorithm by Jonsson and Lagerkvist [13]. We also extended the algorithm to yield an improved upper bound for the d-dimensional block algebra which is substantially faster than the naive enumeration algorithm. Importantly, we demonstrated that dynamic programming appears to be a generally usable method for improved algorithms of qualitative reasoning problems. Let us now briefly discuss a few potential directions for future research.

CSP($\mathcal{A}$) in single-exponential time? The holy grail in exponential-time algorithms is to obtain single-exponential time algorithms of the form $O^*(c^n)$ for some constant $c > 1$. Thus, can CSP($\mathcal{A}$) be solved in single-exponential time? If this appears insurmountable, might it then be possible to prove stronger lower bounds than an expected non-subexponential running time [14], assuming either the *exponential-time hypothesis* or its stronger variant?

CSPs over partial orders. CSP($\mathcal{A}$) is a special case of a broader class of CSPs, namely constraint languages consisting of binary, pairwise disjoint and jointly exhaustive relations containing a strict partial order satisfying certain additional properties [14]. For example, the *region connection calculus* is also included in this class. Can our dynamic programming algorithms be adapted to problems of this form, and might it even be possible to find a uniform algorithm which solves all problems in this class faster than the naive enumeration algorithm? Even more ambitiously, does there exist an NP-hard problem in this class which is solvable in single-exponential time?

The case for higher dimensions. Similar to how Algorithm 2 solves CSP($\mathcal{A}$) directly, rather than via temporal constraints, there is an extension of Algorithm 2 for solving higher dimensions directly, without having to fixate all but one dimensions. This can be done by

extending Definition 8 to contain d-ordered partitions and by not moving variables from the partitions to the V_{ok} set until every part of a d-dimensional variable have been fixed in the partitions. However, this gives a $O^*((d2^{d+\frac{1}{d}}(\frac{2d-1}{2d})^{2d-1})^n OBN(n)^{2d-1})$ run time, and hence, unexpectedly, results in a *worse* bound than Theorem 12. Thus, does there exist a faster, direct algorithm for CSP($\mathcal{A}^d$)?

Acknowledgements

We thank Peter Jonsson for helpful discussions on the topic of the paper, and the anonymous reviewers for several insightful comments. The first author is partially supported by the *National Graduate School in Computer Science* (CUGS), Sweden. The second author is partially supported by the Swedish Research Council (VR) under grant 2019-03690.

References

[1] J. F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26:510–521, 01 1983.

[2] J. F. Allen and J. A. G. M. Koomen. Planning using a temporal world model. In Alan Bundy, editor, *Proceedings of the 8th International Joint Conference on Artificial Intelligence (IJCAI-1983). Karlsruhe, FRG, August 1983*, pages 741–747. William Kaufmann, 1983.

[3] R. Andonov, V. Poirriez, and S. Rajopadhye. Unbounded knapsack problem: Dynamic programming revisited. *European Journal of Operational Research*, 123(2):394 – 407, 2000.

[4] A. Björklund, T. Husfeldt, and M. Koivisto. Set partitioning via inclusion-exclusion. *SIAM Journal on Computing*, 39(2):546–563, 2009.

[5] M. Bodirsky and J. Kára. The complexity of temporal constraint satisfaction problems. *Journal of the ACM*, 57(2):9:1–9:41, 2010.

[6] K. K. Dabrowski, P. Jonsson, S. Ordyniak, and G. Osipov. Fine-grained complexity of temporal problems. In Diego Calvanese, Esra Erdem, and Michael Thielscher, editors, *Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning (KR-2020)*, pages 284–293, 2020.

[7] P. Denis and P. Muller. Predicting globally-coherent temporal structures from texts via endpoint inference and graph decomposition. In Toby Walsh, editor, *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI-2011), Barcelona, Catalonia, Spain, July 16-22, 2011*, pages 1788–1793. IJCAI/AAAI, 2011.

[8] J. Dorn. Dependable reactive event-oriented planning. *Data & Knowledge Engineering*, 16(1):27–49, 1995.

[9] F. Dylla, J. H. Lee, T. Mossakowski, T. Schneider, A. Van Delden, J. Van De Ven, and D. Wolter. A survey of qualitative spatial and temporal calculi: Algebraic and computational properties. *ACM Computing Surveys (CSUR)*, 50(1):7:1–7:39, April 2017.

[10] M. C. Golumbic and R. Shamir. Complexity and algorithms for reasoning about time: A graph-theoretic approach. *Journal of the ACM*, 40(5):1108–1133, 1993.

[11] J. Huang, J. Jingshi Li, and J. Renz. Decomposition and tractability in qualitative spatial and temporal reasoning. *Artificial Intelligence*, 195:140–164, 2013.

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

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

[14] P. Jonsson and V. Lagerkvist. Why are CSPs based on partition schemes computationally hard? In *43rd International Symposium on Mathematical Foundations of Computer Science (MFCS-2018)*, pages 43:1–43:15, 2018.

[15] W. Liu, S. Li, and J. Renz. Combining rcc-8 with qualitative direction calculi: Algorithms and complexity. pages 854–859, 01 2009.

[16] C. McDiarmid. On the span in channel assignment problems: bounds, computing and counting. *Discrete Mathematics*, 266(1):387 – 397, 2003. The 18th British Combinatorial Conference.

[17] L. Mudrová and N. Hawes. Task scheduling for mobile robots using interval algebra. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA-2015)*, pages 383–388. IEEE, 2015.

[18] B. Nebel. Solving hard qualitative temporal reasoning problems: Evaluating the efficiency of using the ord-horn class. *Constraints - An International Journal*, 1(3):175–190, 1997.

[19] D. Papadias and Y. Theodoridis. Spatial relations, minimum bounding rectangles, and spatial data structures. *International Journal of Geographical Information Science*, 11(2):111–138, 1997.

[20] R. N. Pelavin and J. F. Allen. A model for concurrent actions having temporal extent. In Kenneth D. Forbus and Howard E. Shrobe, editors, *Proceedings of the 6th National Conference on Artificial Intelligence (AAAI-1987). Seattle, WA, USA, July 1987*, pages 246–250. Morgan Kaufmann, 1987.

[21] D. Nghia Pham, J. Thornton, and A. Sattar. Modelling and solving temporal reasoning as propositional satisfiability. *Artificial intelligence*, 172(15):1752–1782, 2008.

[22] M. Sioutis and T. Janhunen. Towards leveraging backdoors in qualitative constraint networks. In *Proceedings of the 42nd German Conference on AI (KI-2019)*, pages 308–315, 2019.

[23] F. Song and R. Cohen. The interpretation of temporal relations in narrative. In Howard E. Shrobe, Tom M. Mitchell, and Reid G. Smith, editors, *Proceedings of the 7th National Conference on Artificial Intelligence (AAAI-1988), St. Paul, MN, USA, August 21-26, 1988*, pages 745–750. AAAI Press / The MIT Press, 1988.

[24] J. Thornton, M. Beaumont, A. Sattar, and M. J. Maher. A local search approach to modelling and solving interval algebra problems. *Journal of logic and computation*, 14(1):93–112, 2004.

[25] P. Zhang and J. Renz. Qualitative spatial representation and reasoning in angry birds: The extended rectangle algebra. In *Proceedings of the Fourteenth International Conference on Principles of Knowledge Representation and Reasoning*, page 378–387. AAAI Press, 2014.