Towards an Efficient SAT Encoding for Temporal Reasoning

Duc Nghia Pham, John Thornton, and Abdul Sattar

Safeguarding Australia Program
National ICT Australia Ltd., Australia

Institute for Integrated and Intelligent Systems
Griffith University, Australia
{duc-nghia.pham, abdul.sattar, john.thornton}@nicta.com.au

Abstract. In this paper, we investigate how an IA network can be effectively encoded into the SAT domain. We propose two basic approaches to modelling an IA network as a CSP: one represents the relations between intervals as variables and the other represents the relations between end-points of intervals as variables. By combining these two approaches with three different SAT encoding schemes, we produced six encoding schemes for converting IA to SAT. These encodings were empirically studied using randomly generated IA problems of sizes ranging from 20 to 100 nodes. A general conclusion we draw from these experimental results is that encoding IA into SAT produces better results than existing approaches. Further, we observe that the phase transition region maps directly from the IA encoding to each SAT encoding, but, surprisingly, the location of the hard region varies according to the encoding scheme. Our results also show a fixed performance ranking order over the various encoding schemes.

1 Introduction

Representing and reasoning about time dependent information (i.e., *temporal reasoning*), is a central research issue in computer science and artificial intelligence. The basic tasks include the design and development of efficient reasoning methods for finding scenarios that are consistent with the given information, and effectively answering queries. Often, such information is incomplete and uncertain. One of the most expressive formalisms used to represent such qualitative temporal information is the Interval Algebra (IA) proposed by Allen [1].

While IA is an expressively rich framework, the reasoning problem is computationally intractable [21]. Existing reasoning techniques are generally based on the backtracking approach (proposed by Ladkin and Reinefeld [10]), which uses path consistency as forward checking. Although this approach has been further improved [13, 20], it and its variants still rely on path consistency checking at each step to prune the search space. This *native* IA approach has the advantage of being fairly compact, but is disadvantaged by the overhead of continually ensuring path-consistency. Additionally, the native IA representation of variables and constraints means that state-of-the-art local

search and systematic search heuristics cannot be easily transferred to the temporal domain.

In practice, existing native IA backtracking approaches are only able to find consistent solutions for relatively small general IA instances [18, 17]. On the other hand, recent research has shown that modelling and solving hard combinatorial problems (including planning problems) as SAT instances can produce significant performance benefits over solving problems in their original form [9, 8, 15]. This motivated us to undertake this study.

In this paper we investigate whether the representation of IA problems using specialised models that require specialised algorithms is necessary in the general case. Given that the development of such approaches takes considerable effort, we would expect significant performance benefits to result. To answer this question, we look at expressing IA as a CNF formula using six different SAT encoding schemes. This enables us to apply a range of SAT solvers and to compare the performance of these with the existing native IA approaches. To the best of our knowledge, there is no explicit and thorough work on formulating temporal problems as SAT instances. Nebel and Bürckert [14] pointed out that qualitative temporal instances can be translated to SAT instances but that such a translation causes an exponential blowup in problem size. Hence, no further investigation was provided in their work.[1]

The remainder of the paper is structured as follows: next we review the basic definitions of IA. Then in Section 3 we introduce two models for transforming IA instances into CSP instances. Using these methods, combined with three CSP-to-SAT encodings, six IA-to-SAT encodings are presented in Section 4. Sections 5-7 present an empirical study to investigate the hardness distribution of these SAT encodings and evaluate their performance relative to each other, and in comparison to existing approaches. Finally, Section 8 presents the conclusion and discusses future research directions.

2 Interval Algebra

Interval Algebra [1] is the most commonly used formalism to represent temporal interval events. It consists of a set of 13 atomic relations between two time intervals: $\mathcal{I} = \{eq, b, bi, m, mi, o, oi, d, di, s, si, f, fi\}$ (see Table 1). Indefinite information between two time intervals can be expressed as a subset of $\mathcal{I}$ (e.g. a disjunction of atomic relations). For example, the statement *"Event A can happen either before or after event B"* can be expressed as $A\{b, bi\}B$. Hence there are a total of $2^{|\mathcal{I}|} = 8,192$ possible relations between pairs of temporal intervals.

Let R_1 and R_2 be two IA relations. Then the four operators of IA: *union* ($\cup$), *intersection* ($\cap$), *inversion* ($^{-1}$), and *composition* ($\circ$), can be defined as follows:

$$\forall\, A, B : A(R_1 \cup R_2)B \leftrightarrow (AR_1B \lor AR_2B)$$
$$\forall\, A, B : A(R_1 \cap R_2)B \leftrightarrow (AR_1B \land AR_2B)$$
$$\forall\, A, B : A(R_1^{-1})B \leftrightarrow BR_1A$$
$$\forall\, A, B : A(R_1 \circ R_2)B \leftrightarrow \exists\, C : (AR_1C \land CR_2B).$$

---

[1] Recent independent work [6] has proposed representing IA as SAT, but the authors do not specify the transformation in detail, and do not provide an adequate empirical evaluation.

Atomic relation	Symbol	Meaning	Endpoint relations
A before B	b		$A^- < B^-, A^- < B^+$
B after A	bi		$A^+ < B^-, A^+ < B^+$
A meets B	m		$A^- < B^-, A^- < B^+$
B met by A	mi		$A^+ = B^-, A^+ < B^+$
A overlaps B	o		$A^- < B^-, A^- < B^+$
B overlapped by A	oi		$A^+ > B^-, A^+ < B^+$
A during B	d		$A^- > B^-, A^- < B^+$
B includes A	di		$A^+ > B^-, A^+ < B^+$
A starts B	s		$A^- = B^-, A^- < B^+$
B started by A	si		$A^+ > B^-, A^+ < B^+$
A finishes B	f		$A^- > B^-, A^- < B^+$
B finished by A	fi		$A^+ > B^-, A^+ = B^+$
A equals B	eq		$A^- = B^-, A^- < B^+$
			$A^+ > B^-, A^+ = B^+$

Table 1. The 13 IA atomic relations. Note that the endpoint relations $A^- < A^+$ and $B^- < B^+$ have been omitted.

Hence, the *intersection* and *union* of any two temporal relations (R_1, R_2) are simply the standard set-theoretic intersection and union of the two sets of atomic relations describing R_1 and R_2, respectively. The *inversion* of a temporal relation R is the union of the inversion of each atomic relation $r_i \in R$. The *composition* of any pair of temporal relations (R_1, R_2) is the union of all results of the composition operation on each pair of atomic relations (r_{1i}, r_{2j}), where $r_{1i} \in R_1$ and $r_{2j} \in R_2$. The full composition results of these IA atomic relations can be found in [1].

An IA network can be represented as a *constraint graph* or a *constraint network* where the vertices represent interval events and the arcs are labelled with the possible interval relations between a pair of intervals [13]. Usually, such a constraint graph for n interval events is described by an $n \times n$ matrix M, where each entry M_{ij} is the label of the arc between the ith and j^{th} intervals. An IA *scenario* is a *singleton* IA network where each arc (constraint) is labelled with *exactly one* atomic relation.

An IA network with n intervals is *globally consistent* iff it is strongly n-consistent [11]. Hence, the ISAT problem of determining the satisfiability of a given IA network becomes the problem of determining whether that network is globally consistent [1, 13]. ISAT is the *fundamental* reasoning task in the temporal reasoning community because all other interesting reasoning problems can be reduced to it in polynomial time [7] and it is one of the most important tasks in practical applications [20].

It is worth noting that enforcing *path consistency* [11, 3] is enough to ensure global consistency for the maximal tractable subclasses of IA, including singleton networks [13]. Allen [1] proposed a path consistency method for an IA network M that repeatedly computes the following *triangle operation*: $M_{ij} \leftarrow M_{ij} \cap M_{ik} \circ M_{kj}$ for all triplets of vertices (i, j, k) until no further change occurs or until $M_{ij} = \emptyset$. These operations remove all the atomic relations that cause an inconsistency between any triple (i, j, k) of intervals. The resulting network is a path consistent IA network. If $M_{ij} = \emptyset$, then the original IA network is path inconsistent. More sophisticated path consistency algorithms have been applied to IA networks that run in $O(n^3)$ time [19, 13].

3 Reformulation of IA into CSP

A common approach to encode combinatorial problems into SAT is to divide the task into two steps: (i) modelling the original problem as a CSP; and (ii) mapping the new CSP into SAT. In the next two subsections, we propose two transformation methods to model IA networks as CSPs such that these CSPs can be feasibly translated into SAT. We then discuss three SAT encoding schemes to map the CSP formulations into SAT, producing six different approaches to encode IA networks into SAT. [2]

3.1 The Interval-Based CSP Formulation

A straightforward method to formulate IA networks as CSPs is to represent each arc as a CSP variable. We then limit the domain of each variable to the set of permissible IA atomic relations for that arc, rather than the set of all subsets of $\mathcal{I}$ used in existing IA approaches. This allows us to reduce the domain size of each variable from 2^{13} to a maximum of 13 values. Thus an instantiation of an *interval*-based CSP maps each variable (arc) to *exactly one* atomic relation in its domain. In other words, an instantiation of this new CSP model is actually a singleton network of the original IA network.

Lemma 1. *Let Θ be a singleton IA network with 3 intervals I_1, I_2, and I_3. Then Θ is consistent iff $r_{13} \in r_{12} \circ r_{23}$ where r_{ij} is an arc between any two I_i and I_j intervals.*

Proof. Trivial as there is exactly one mapping of a singleton network onto the time line.

Theorem 1. *Let Θ be a singleton IA network with n intervals and r_{ij} be the label of the arc between I_i and I_j. Then Θ is consistent iff for any triple $(i < k < j)$ of vertices, $r_{ij} \in r_{ik} \circ r_{kj}$.*

Proof. ($\Rightarrow$) This direction is trivial as Θ is also path consistent.

($\Leftarrow$) As $r_{ij} \in r_{ik} \circ r_{kj}$ holds for all triplets $(i < k < j)$ of vertices, Θ is path consistent by Lemma 1. In addition, Θ is singleton. Hence, Θ is globally consistent.

Based on the results of Theorem 1, an *interval*-based CSP representation of a given IA network is defined as follows:

Definition 1. *Given an IA network M with n intervals, $I_1, \ldots, I_n$; the corresponding interval-based CSP is $(\mathcal{X}, \mathcal{D}, \mathcal{C})$, where*
$\mathcal{X} = \{X_{ij} \mid i, j \in [1..n], i < j\}$; each variable X_{ij} represents a relation between two intervals I_i and I_j;
$\mathcal{D} = \{D_{ij}\}$, each D_{ij} is a set of domain values for X_{ij}, and $D_{ij} = M_{ij}$ the set of relations between interval I_i and I_j; and
$\mathcal{C}$ consists of the following constraints:

$$\bigwedge_{x \in D_{ik}, y \in D_{kj}} X_{ik} = x \wedge X_{kj} = y \Longrightarrow X_{ij} \in D'_{ij} \tag{1}$$

where $i < k < j$ and $D'_{ij} = D_{ij} \cap (x \circ y)$.

---

[2] In practice, IA networks can be directly encoded into SAT formulae without being reformulated as CSPs. However, for the sake of clarity we first transform IA into two different CSP formulations and then to SAT.

Theorem 2. *Let Θ be an IA network and Φ be the corresponding interval-based CSP defined by Definition 1. Then Θ is globally consistent or satisfiable iff Φ is satisfiable.*

Proof. We first rewrite the constraint (1) into two equivalent clauses

$$\bigwedge_{x \in D_{ik}, y \in D_{kj}} X_{ik} = x \wedge X_{kj} = y \implies X_{ij} \in D_{ij} \tag{2}$$

$$\bigwedge_{x \in D_{ik}, y \in D_{kj}} X_{ik} = x \wedge X_{kj} = y \implies X_{ij} \in x \circ y \tag{3}$$

($\Rightarrow$) Let Θ' be a consistent scenario of Θ. As Θ' is a singleton network, Θ' is also an instantiation of Φ by Definition 1. Hence clause (2) is satisfied. In addition, as Θ' is globally consistent, clause (3) is also satisfied by Theorem 1. Hence Θ' satisfies all constraints of Φ. As a result, Φ is satisfiable.

($\Leftarrow$) Let Φ' be an instantiation of Φ such that it satisfies all constraints of Φ (i.e. clauses (2) and (3) are satisfied). We construct a singleton network Θ' by labelling each arc (i, j) of Θ' with the atomic relation (value) $\Phi'(i, j)$. As Φ' satisfies clause (2), Θ' is a singleton network of Θ. In addition, as Φ' satisfies clause (3), we have $\Theta'(i, j) \in \Theta'(i, k) \circ \Theta'(k, j)$ for all triples $(i < k < j)$ of vertices. Applying Theorem 1, Θ' is globally consistent. As a result, Θ is satisfiable.

Example For the sake of clarity, we use the IA network in Figure 1(a) as a running example to illustrate the transformation of IA networks into CSPs and SAT encodings. The example represents the following scenario: *"Anne usually reads her paper (I_1) before or during her breakfast (I_3). In addition, she always drinks a cup of coffee (I_2) during her breakfast. This morning, she started reading her paper before her coffee was served and finished reading before drinking the last of her coffee"*. The corresponding interval-based CSP of this IA network is shown in Figure 1(b), having 3 variables, which represent the temporal relations between each pair of actions. These variables and their corresponding domains are described using the same order in $\mathcal{X}$ and $\mathcal{D}$. Note that as $\{o\} \circ \{d\} = \{o, d, s\}$, the constraint between I_1, I_2 and I_3 further restricts the domain of X_{13} to $\{d\}$ instead of its original $\{b, d\}$, i.e. Anne could not have read her paper before breakfast if she was still reading it while drinking coffee *during* breakfast.

$$\mathcal{X} = \{ X_{12}, X_{13}, X_{23} \}$$

$$\mathcal{D} = \{ \{o\}, \{b, d\}, \{d\} \}$$

$$\mathcal{C} = \{ (X_{12} = o \wedge X_{23} = d \implies X_{13} = d) \}$$

(a) IA network   (b) interval-based CSP

Fig. 1. An interval-based CSP representation of the running example.

3.2 The Point-Based CSP Formulation

Vilain and Kautz [21] proposed the Point Algebra (PA) to model qualitative information between time points. PA consists of a set of 3 atomic relations $\mathcal{P} = \{<, =, >\}$ and 4 operators defined in a similar manner to IA. In addition, the concepts of consistency discussed above for IA networks are also applicable to PA networks. Again, we use an $n \times n$ matrix P to represent a PA network with n points where P_{ij} is the relation between two points i and j.

As mentioned in Section 2, IA atomic relations can be uniquely expressed in terms of their endpoint relations. However, representing non-atomic IA relations is more complex, as not all IA relations can be translated into point relations. For example, the following combination of point relations

$$(A^- \neq B^-) \wedge (A^- < B^+) \wedge (A^+ \neq B^-) \wedge (A^+ < B^+)$$

represents not only $A\{b,d\}B$ but also $A\{b,d,o\}B$. This means that PA can only cover 2% of IA [13].

Using the CSP formalism, we can prevent the instantiation of such undesired IA relations by simply introducing new constraints into the CSP model. Let $\mu(r) = (v_{ss}, v_{se}, v_{es}, v_{ee})$ be the PA representation of an IA atomic relation r between two intervals A and B, where v_{se}, for example, is the corresponding PA relation between two endpoints A^- and B^+. We then define the *point*-based CSP model of an IA network as follows:

Definition 2. *Given an IA network M with n intervals and its corresponding PA network P (with $2n$ points, $P_1,...,P_{2n}$), the point-based CSP of M is $(\mathcal{X}, \mathcal{D}, \mathcal{C})$, where $\mathcal{X} = \{X_{ij} \mid i, j \in [1..2n], i < j\}$; each variable X_{ij} represents a relation between two points P_i and P_j of P;*
$\mathcal{D} = \{D_{ij}\}$, each D_{ij} is the set of domain values for X_{ij} and $D_{ij} = P_{ij}$ the set of point relations between P_i and P_j; and
$\mathcal{C}$ consists of the following constraints:

$$\bigwedge_{x \in D_{ik}, y \in D_{kj}} X_{ik} = x \wedge X_{kj} = y \implies X_{ij} \in D'_{ij} \tag{4}$$

$$\bigwedge_{r \notin M_{lm}} (X_{l^- m^-}, X_{l^- m^+}, X_{l^+ m^-}, X_{l^+ m^+}) \neq \mu(r) \tag{5}$$

where $i < k < j$, $D'_{ij} = D_{ij} \cap (x \circ y)$, and $X_{l^* m^*}$ is the CSP variable representing the relation between one endpoint of interval l and one endpoint of interval m.

Theorem 3. *Let Ω be a singleton PA network with n points and r_{ij} be the label of the arc between two points I_i and I_j. Then Ω is consistent iff for any triple $(i < k < j)$ of vertices, $r_{ij} \in r_{ik} \circ r_{kj}$.*

As Theorem 3 is similar to Theorem 1, we can construct its proof in a similar way to the proof of Theorem 1.

Theorem 4. *Let Θ be an IA network and Ψ be the corresponding point-based CSP defined by Definition 2. Then Θ is globally consistent or satisfiable iff Ψ is satisfiable.*

Proof. ($\Rightarrow$) Let Θ' be a consistent scenario of Θ. As Θ' is a singleton network, its corresponding point-based CSP Ψ', defined by Definition 2, is an instantiation of Ψ. Hence, Ψ' satisfies all constraints (5). In addition, as Θ' is globally consistent, Ψ' satisfies all constraints (4) due to Theorem 3. As a result, Ψ is satisfiable.

($\Leftarrow$) Let Ψ' be an instantiation of Ψ such that all constraints (4) and (5) are satisfied. Let $\mu^{-1}(X_{l^-m^-}, X_{l^-m^+}, X_{l^+m^-}, X_{l^+m^+}) = r$ be the inversion of $\mu(r)$, such that it maps the combination of the PA atomic relations of four endpoints $(X_{l^-m^-}, X_{l^-m^+}, X_{l^+m^-}, X_{l^+m^+})$ to the IA atomic relation r between two intervals l and m. As every variable $X_{l^*m^*}$ of Ψ' is instantiated with exactly one atomic relation, $\mu^{-1}(X_{l^-m^-}, X_{l^-m^+}, X_{l^+m^-}, X_{l^+m^+})$ maps to exactly one interval relation.

We construct a singleton IA network Θ' from Ψ' by labelling each arc (l, m) with the corresponding IA atomic relation $\mu^{-1}(X_{l^-m^-}, X_{l^-m^+}, X_{l^+m^-}, X_{l^+m^+})$. As Ψ' satisfies all constraints (5), Θ' is a scenario of Θ. In addition, Θ' is globally consistent by the application of Theorem 3 as Ψ' satisfies all constraints (4). As a result, Θ is satisfiable.

Example Figure 2 shows a point-based CSP corresponding to the original IA network from Figure 1(a), including a partial PA graph to assist in understanding the point-based CSP translation. In this graph (Figure 2(a)), each interval I_i has been replaced by its endpoints I_{i-} (the start point) and I_{i+} (the finish point) and all temporal relations between pairs of intervals have been replaced by corresponding relations between their endpoints. These endpoint relations are the CSP variables in the new model, which are in turn instantiated with PA atomic relations. For example, the expression $X_{1-1+} = <$ means that the arc between the endpoints I_{1-} and I_{1+} must be instantiated with the value $<$, thereby expressing the underlying PA constraint $I_{1-} < I_{1+}$. The power we obtain from this CSP model is that we can disallow unwanted interpretations that cannot be eliminated from a simple PA network. For example, in Figure 2(a) the PA graph is not a correct alternative representation of the original IA network as it allows interval I_1 to overlap (o) with interval I_3. In the CSP formalism we can disallow this overlapping relation using the third constraint in Figure 2(b): $(X_{1-3-} \neq <) \wedge (X_{1-3+} \neq <) \wedge (X_{1+3-} \neq >) \wedge (X_{1+3+} \neq <)$. It should further be noted that the order of domains in $\mathcal{D}$ is preserved exactly with respect to their corresponding variables in $\mathcal{X}$ and that all constraints of type (4) that do not further restrict the domain values of a variable have been omitted.

4 Reformulation of IA into SAT

In this section, we describe three different schemes to encode the interval-based or point-based non-binary CSP formulations (as described in the previous section) into SAT, resulting in six different ways of encoding IA into SAT. First, we describe the one-dimensional (1D) support scheme that naturally translates IA CSPs into CNF formulae. We then present extensions of the *direct* and *log* encoding schemes [8, 22].

(a) PA network (b) point-based CSP

Fig. 2. A point-based CSP representation of the running example.

4.1 The SAT 1-D Support Encoding

Using either interval-based or point-based CSP formulations, an IA network can be encoded as a SAT instance, in which each Boolean variable x_{ij}^r represents an assignment of a domain value r to a CSP variable X_{ij}. The Boolean variable x_{ij}^r is true iff the value r is assigned to the CSP variable X_{ij}. For each CSP variable X_{ij} having a domain of values D_{ij}, two sets of at-least-one (ALO) and at-most-one (AMO) clauses are used to ensure that there is exactly one domain value $v \in D_{ij}$ assigned to X_{ij} at any time:

$$ALO: \quad \bigvee_{v \in D_{ij}} x_{ij}^v \tag{6}$$

$$AMO: \quad \bigwedge_{u,v \in D_{ij}} \neg x_{ij}^u \vee \neg x_{ij}^v \tag{7}$$

It is common practice to encode a general CSP into a SAT formula without the AMO clauses, thereby allowing CSP variables to be instantiated with more than one value [22]. A CSP solution can then be extracted by taking any single SAT-assigned value for each CSP variable. However, our two CSP formulation methods strongly depend on the fact that each CSP variable can only be instantiated with exactly one value at any time. This maintains the completeness of our reformulation methods (see the proofs above). A counter-example is shown in Figure 3. I_1 is *before* I_4 because I_1 is *during* I_2 and I_2 is *before* I_4. In addition, as I_1 *overlaps* I_3 and I_3 *starts* I_4, I_1 *overlaps* I_4. As a result, I_1 is either *before* or *overlaps* I_4. However, neither of the scenarios obtained from this network is consistent. Hence, the AMO clauses cannot be removed from our translation.

A natural way to encode the consistency constraints, i.e. constraints (1) and (4) above, is to add the following support (SUP) clauses:

$$SUP: \quad \bigwedge_{u \in D_{ik}, v \in D_{kj}} \neg x_{ik}^u \vee \neg x_{kj}^v \vee x_{ij}^{w_1} \vee \ldots \vee x_{ij}^{w_m} \tag{8}$$

where $D'_{ij} = D_{ij} \cap (u \circ v) = \{w_1, \ldots, w_m\}$. Note that we use the IA composition table for the interval-based reduction method and the PA composition table for the point-based reduction method.

$$X_{12} = d \wedge X_{24} = b \Rightarrow X_{14} = b$$
$$X_{13} = o \wedge X_{34} = s \Rightarrow X_{14} = o$$

Fig. 3. A counter-example of removing AMO clauses.

The constraints (5) in a point-based CSP are translated into a SAT formula using the following *forbidden* (FOR) clauses:

$$FOR: \quad \bigwedge_{r \notin M_{lm}} \neg x^u_{l^- m^-} \vee \neg x^v_{l^- m^+} \vee \neg x^y_{l^+ m^-} \vee \neg x^z_{l^+ m^+} \tag{9}$$

where u, v, y, z are PA atomic relations and $\mu(r) = (u, v, y, z)$. For example, given that the PA representation of $X_l\{o\}X_m$ is $\mu(o) = (<, <, >, <)$, the corresponding forbidden clause is $\neg x^<_{l^- m^-} \vee \neg x^<_{l^- m^+} \vee \neg x^>_{l^+ m^-} \vee \neg x^<_{l^+ m^+}$.

We refer to this method as the 1-D *support* encoding scheme because it encodes the support values of the original problem. In Gent's support encoding scheme [5], the support clauses are necessary for both implication directions of the CSP constraints. However, in our scheme, only one SUP clause is needed for each triple of intervals $(i < k < j)$, and not for *all* permutation orders of this triple.

ALO:	$(x^o_{12})\ (x^b_{13} \vee x^d_{13})\ (x^d_{23})$
AMO:	$(\neg x^b_{13} \vee \neg x^d_{13})$
SUP:	$(\neg x^o_{12} \vee \neg x^d_{23} \vee x^d_{13})$

Fig. 4. An interval-based 1-D support encoding of the running example.

ALO: $\quad (x^<_{1^- 1^+})\qquad (x^<_{2^- 2^+})\qquad (x^<_{3^- 3^+})$
$\qquad (x^<_{1^- 2^-})\qquad (x^<_{1^- 2^+})\qquad (x^>_{1^+ 2^-})\qquad (x^<_{1^+ 2^+})$
$(x^<_{1^- 3^-} \vee x^>_{1^- 3^-})\ (x^<_{1^- 3^+})\ (x^<_{1^+ 3^-} \vee x^>_{1^+ 3^-})\ (x^<_{1^+ 3^+})$
$\qquad (x^>_{2^- 3^-})\qquad (x^<_{2^- 3^+})\qquad (x^>_{2^+ 3^-})\qquad (x^<_{2^+ 3^+})$
AMO: $\quad (\neg x^<_{1^- 3^-} \vee \neg x^>_{1^- 3^-})\qquad\qquad (\neg x^<_{1^+ 3^-} \vee \neg x^>_{1^+ 3^-})$
SUP: $(\neg x^<_{1^- 1^+} \vee \neg x^<_{1^+ 3^-} \vee x^<_{1^- 3^-})\ (\neg x^>_{1^+ 2^-} \vee \neg x^>_{2^- 3^-} \vee x^>_{1^+ 3^-})$
FOR: $\qquad (\neg x^<_{1^- 3^-} \vee \neg x^<_{1^- 3^+} \vee \neg x^>_{1^+ 3^-} \vee \neg x^<_{1^+ 3^+})$

Fig. 5. A point-based 1-D support encoding of the running example.

4.2 The SAT Direct Encoding

Another way of representing CSP constraints as SAT clauses is to encode the conflict values between any pair of CSP variables [8, 22]. This *direct* encoding scheme for IA networks can be derived from our 1-D support encoding scheme by replacing the SUP clauses with conflict (CON) clauses. If we represent SUP clauses between a triple of intervals $(i < k < j)$ as a 3D array of allowable values for the CSP variable X_{ij} given the values of X_{ik} and X_{kj}, then the corresponding CON clauses are defined as:

$$CON: \bigwedge_{u\in D_{ik}, v\in D_{kj}, w\in D''_{ij}} \neg x^u_{ik} \lor \neg x^v_{kj} \lor \neg x^w_{ij} \qquad (10)$$

where $D''_{ij} = D_{ij} - (u \circ v)$.

The *multivalued* encoding [15] is a variation of the direct encoding, where all AMO clauses are omitted. As discussed earlier, we did not consider such an encoding because in our IA transformations the AMO clauses play a necessary role.

$$
\begin{aligned}
&\text{ALO:} \quad (x^o_{12})\ (x^b_{13} \lor x^d_{13})\ (x^d_{23}) \\
&\text{AMO:} \quad\quad (\neg x^b_{13} \lor \neg x^d_{13}) \\
&\text{CON:} \quad (\neg x^o_{12} \lor \neg x^d_{23} \lor \neg x^b_{13})
\end{aligned}
$$

Fig. 6. An interval-based direct encoding of the running example.

$$
\begin{aligned}
&\text{ALO:} && (x^<_{1^-1^+}) && (x^<_{2^-2^+}) && (x^<_{3^+3^+}) \\
& && (x^<_{1^-2^-}) && (x^<_{1^-2^+}) && (x^>_{1^+2^-}) && (x^<_{1^+2^+}) \\
& && (x^<_{1^-3^-} \lor x^>_{1^-3^-}) && (x^<_{1^-3^+}) && (x^<_{1^+3^-} \lor x^>_{1^+3^-}) && (x^<_{1^+3^+}) \\
& && (x^>_{2^-3^-}) && (x^<_{2^-3^+}) && (x^>_{2^+3^-}) && (x^<_{2^+3^+}) \\
&\text{AMO:} && (\neg x^<_{1^-3^-} \lor \neg x^>_{1^-3^-}) && && (\neg x^<_{1^+3^-} \lor \neg x^>_{1^+3^-}) \\
&\text{CON:} && (\neg x^<_{1^-1^+} \lor \neg x^<_{1^+3^-} \lor \neg x^>_{1^-3^-})\ (\neg x^>_{1^+2^-} \lor \neg x^>_{2^-3^-} \lor \neg x^<_{1^+3^-}) \\
&\text{FOR:} && (\neg x^<_{1^-3^-} \lor \neg x^<_{1^-3^+} \lor \neg x^>_{1^+3^-} \lor \neg x^<_{1^+3^+})
\end{aligned}
$$

Fig. 7. A point-based direct encoding of the running example.

4.3 The SAT Log Encoding

A compact version of the direct encoding is the *log* encoding [8, 22]. Here, a Boolean variable x^l_i is true iff the corresponding CSP variable X_i is assigned a value in which the l-th bit of that value is 1. We can linearly derive log encoded IA instances from direct encoded IA instances by replacing each Boolean variable in the direct encoding with its *bitwise representation*. As a single instantiation of the underlying CSP variable is enforced by the bitwise representation, ALO and AMO clauses are omitted. However, extra prohibited (PRO) clauses are added (if necessary) to prevent undesired bitwise representations from being instantiated. For example, if the domain of variable X has three values then we have to add the clause $\neg x^0_3 \lor \neg x^1_3$ to prevent the fourth value from assigning to X. Another way to handle redundant bitwise representations is to treat them as equivalent to a valid representation. However, this *binary* encoding [4] tends to generate exponentially more conflict clauses than the log encoding and hence is not considered in this study.

$$
\begin{aligned}
&\text{PRO:} \quad (\neg x^1_{12})\ (\neg x^1_{23}) \\
&\text{CON}_l: (x^1_{12} \lor x^1_{23} \lor x^1_{13})
\end{aligned}
$$

Fig. 8. An interval-based log encoding of the running example.

$$\begin{aligned}
\text{PRO:} \quad & (\neg x^1_{1-1+}) && (\neg x^1_{2-2+}) && (\neg x^1_{3+3+}) \\
& (\neg x^1_{1-2-}) && (\neg x^1_{1-2+}) && (\neg x^1_{1+2-}) && (\neg x^1_{1+2+}) \\
& && (\neg x^1_{1-3+}) && && (\neg x^1_{1+3+}) \\
& (\neg x^1_{2-3-}) && (\neg x^1_{2-3+}) && (\neg x^1_{2+3-}) && (\neg x^1_{2+3+}) \\
\text{CON}_l\text{:} \quad & (x^1_{1-1+} \vee x^1_{1+3-} \vee \neg x^1_{1-3-}) \; (x^1_{1+2-} \vee x^1_{2-3-} \vee x^1_{1+3-}) \\
\text{FOR:} \quad & (x^1_{1-3-} \vee x^1_{1-3+} \vee \neg x^1_{1+3-} \vee x^1_{1+3+})
\end{aligned}$$

Fig. 9. A point-based log encoding of the running example.

5 The Phase Transition of SAT-encoded IA instances

As our SAT translations were theoretically proved sound and complete, we expected that the following properties would also be true for our SAT-encoded IA instances:

i) The phase transition of SAT-encoded instances happens at the same critical value of the average degree parameter d as for the original IA instances; and

ii) The performance of SAT solvers on SAT-encoded instances is proportionally similar to the performance of temporal backtracking algorithms on the original IA instances.

To verify these properties, we conducted a similar experiment to that reported in Nebel's study [13]. We generated an extensive benchmark test set of $A(n, d, 6.5)$ IA instances by varying the average degree d from 1 to 20 (in steps of 0.5 from 8 to 11 and in steps of 1 otherwise) and n from 20 to 50 (in steps of 5).[3] We generated 500 instances for each n/d data point to obtain a set of $23 \times 7 \times 500 = 80,500$ test instances. We then ran two variants of Nebel's backtracking algorithm [13], NBT$_\mathcal{I}$ and NBT$_\mathcal{H}$, on these instances and zChaff [12] on the corresponding SAT-encoded instances. NBT$_\mathcal{I}$ instantiates each arc with an atomic relation in $\mathcal{I}$, whereas NBT$_\mathcal{H}$ assigns a relation in the set $\mathcal{H}$ of ORD-Horn relations to each arc. The other heuristics used in Nebel's backtracking algorithm were set to default and all solvers were timed out after one hour.

As expected, the probability of satisfiability for our SAT-encoded instances was the same as the probability of satisfiability for the original IA instances, regardless of the SAT translation method. This is illustrated in Figure 10 which shows that the phase transition happens around $d = 9.5$ for $s = 6.5$ regardless of instance size or representation. These results are consistent with the earlier work of Nebel [13].

However, the performance of zChaff on our six different SAT encodings was relatively significantly different from the performance of NBT$_\mathcal{I}$ and NBT$_\mathcal{H}$ on the native IA representations. As graphed in Figure 10, the median runtime of NBT$_\mathcal{I}$ and NBT$_\mathcal{H}$ both peaked where the phase transition happens, i.e. $d = 9.5$. In contrast, the runtime peaks of zChaff on the SAT instances were shifted away from the phase transition. The graphs in the middle row of Figure 10 show that the median CPU time of zChaff on the point-based 1-D support, direct and log instances peaked around $d = 9$, 8 and 6, respectively. In addition, the CPU time of zChaff on instances surrounding these peaks was relatively similar, regardless of which SAT encoding scheme was used.

---

[3] These instances were generated by Nebel's generator, which is available at ftp://ftp.informatik.uni-freiburg.de/documents/papers/ki/allen-csp-solving.programs.tar.gz

Fig. 10. The phase transition and hardness distribution of $NBT_{\mathcal{I}}$ and $NBT_{\mathcal{H}}$ on the native $A(n, d, 6.5)$ IA instances and zChaff on the corresponding SAT-encoded instances (500 instances per data point).

This result is further supported when we take into account the performance of zChaff on the interval-based SAT instances. The graphs in the bottom row of Figure 10 show the median CPU time of zChaff on the corresponding interval-based 1-D support, direct and log instances. Here we can see that the runtime peaks of zChaff are shifted away from the phase transition in exactly the same way as they were on the point-based SAT instances, regardless of which SAT encoding scheme was used. In fact, the CPU time of zChaff on the interval-based direct and log instances peaked at the same points as their corresponding point-based instances, i.e. at $d = 8$ and 6, respectively. The only exception is the runtime of zChaff on the interval-based 1-D support instances which peaked at $d = 8$, i.e. even further away than for the point-based 1-D support instances.

These results are quite surprising and contrast with the results of previous studies on the phase transition behaviour of IA networks [13] and random problems [2, 16]. Intuitively, the further left we move from the phase transition, the more solutions an instance has and, as a consequence, the easier this instance should be to solve. However, this conjecture is not true for our SAT encoding schemes. The empirical results clearly show that the hard region, where instances take significantly more time to solve, does not always happen around the phase transition. In contrast, the representation or encoding of the problem instance plays an important role in determining where the hard region will occur.

6   An Empirical Comparison Among SAT Encodings

The graphs in Figure 10 provide strong supporting evidence for the following conjectures:

i) A point-based formulation produces better results than an interval-based formulation, regardless of how IA instances are generated (in terms of the number of nodes n, the average degree d or the average label size s) or the SAT encoding employed.

ii) The 1-D support encoding scheme produces the best results, followed by the direct and log encoding schemes, regardless of how IA instances are generated (in terms of the number of nodes n, the average degree d or the average label size s) or the formulation method employed.

iii) Among the six encoding schemes considered, the point-based 1-D support encoding is the most suitable for translating IA instances into SAT formulae.

The superior performance of the 1-D support encoding can be partly explained by the significantly smaller number of clauses generated (on average about ten times less than for direct or log encoded instances). However, it should be noted that the search space (i.e. the number of variables) of 1-D support and direct encoded instances are the same, whereas the search space of log encoded instances is $O(n \times (|s| - log|s|))$ times smaller [8]. A further possible reason for the superiority of the 1-D support encoding (suggested by Gent [5]) is the reduced bias to falsify clauses, i.e. the numbers of positive and negative literals in the support encoding are more balanced than in a direct encoding and hence this may prevent the search from resetting variables to false shortly after they are set to true.

7   Empirical Evaluation of SAT versus Existing Approaches

The final question to address in this study is how our SAT approach compares to the existing state-of-the-art specialised approaches. We generated another benchmark test set of $A(n, d, 6.5)$ IA instances by varying the average degree d from 1 to 20 (in steps of 0.5 from 8 to 11 and in steps of 2 otherwise) across nine values of n varied from 60 to 100 (in steps of 5). We generated 100 instances for each n/d data point to obtain a set of $16 \times 9 \times 100 = 14,400$ test instances. This test set allowed us to take a closer look at the performance of different approaches around the phase transition while still providing a general view across the entire distribution. We then ran $\text{NBT}_\mathcal{H}$ on these instances and zChaff on the corresponding point-based 1-D support SAT instances. All solvers were timed out after one hour for $n < 80$ and four hours for $n \geq 80$.

As shown in Figure 11, the mean CPU time of zChaff was significantly better than the mean CPU time of $\text{NBT}_\mathcal{H}$ (around 4.35 times at $n = 100$). In addition, when the test instances became bigger (e.g. $n \geq 80$), the time curves of $\text{NBT}_\mathcal{H}$ were exponentially increased while the time curves of zChaff remained nearly linear. These observations led us to conjecture that zChaff performs better than $\text{NBT}_\mathcal{H}$ on hard instances and that its performance scales better as the size of the test instances grows. A more thorough analysis of the results produced further evidence to support this conjecture: with a one hour time limit, zChaff was unable to solve 32 of the entire benchmark set of $14,400$

instances, while 323 instances remained unsolved for $NBT_{\mathcal{H}}$ (see Figure 11). When the time limit was raised to four hours, only 2 instances remained unsolvable for zChaff in comparison with 103 for $NBT_{\mathcal{H}}$. This means that the performance of zChaff scaled 51.5 times better than $NBT_{\mathcal{H}}$ on these extremely hard instances.

Fig. 11. The CPU time and the probability of failure of zChaff on the point-based 1-D support instances and $NBT_{\mathcal{H}}$ on the native IA instances.

8 Summary

In summary, we have proposed six different methods to formulate IA networks into SAT formulae and provided the theoretical proofs of completeness of these transformation techniques. Although our empirical results confirmed that the phase transition of IA networks mapped directly into these SAT encodings, they also showed that the hard regions of these problems were surprisingly shifted away from the phase transition areas after transformation into SAT. Evaluating the effects of these SAT encodings, we found that the point-based 1-D support scheme is the best among the six IA-to-SAT schemes examined. Our results also revealed that zChaff combined with our point-based 1-D support scheme could solve IA instances significantly faster than existing IA solvers working on the equivalent native IA networks.

In future work we anticipate that the performance of our SAT-based approach can be further improved by exploiting the special structure of IA problems in a manner analogous to the work on TSAT [17]. The possibility also opens up of integrating our approach to temporal reasoning into other well known real world problems such as planning. Given the success of SAT solvers in many real world domains, our work promises to expand the reach of temporal reasoning approaches for IA to encompass larger and more practical problems.

Acknowledgments

References

1. James F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26:832–843, 1983.
2. Peter Cheeseman, Bob Kanefsky, and William M. Taylor. Where the really hard problems are. In *IJCAI-91*, pages 331–337, 1991.
3. Eugene C. Freuder. Synthesizing constraint expressions. *Communication of ACM*, 21(11):958–966, 1978.
4. Alan M. Frisch and Timothy J. Peugniez. Solving non-Boolean satisfiability problems with stochastic local search. In *IJCAI-01*, pages 282–290, 2001.
5. Ian P. Gent. Arc consistency in SAT. In *ECAI-02*, pages 121–125, 2002.
6. K. Ghiathi and G. Ghassem-Sani. Using satisfiability in temporal planning. *WSEAS Transactions on Computers*, 3(4):963–969, 2004.
7. Martin C. Golumbic and Ron Shamir. Complexity and algorithms for reasoning about time: A graph-theoretic approach. *Journal of ACM*, pages 1108–1133, 1993.
8. Holger H. Hoos. SAT-encodings, search space structure, and local search performance. In *IJCAI-99*, pages 296–302, 1999.
9. Henry Kautz, David McAllester, and Bart Selman. Encoding plans in propositional logic. In *KR-96*, pages 374–384, 1996.
10. Peter Ladkin and Alexander Reinefeld. Effective solution of qualitative interval constraint problems. *Artificial Intelligence*, 57(1):105–124, 1992.
11. Alan K. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, 8(1):99–118, 1977.
12. Matthew W. Moskewicz, Conor F. Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik. Chaff: Engineering an efficient SAT solver. In *DAC-01*, pages 530–535, 2001.
13. Bernhard Nebel. Solving hard qualitative temporal reasoning problems: Evaluating the efficiency of using the ORD-Horn class. *Constraints*, 1(3):175–190, 1997.
14. Bernhard Nebel and Hans-Jürgen Bürckert. Reasoning about temporal relations: A maximal tractable subclass of Allen's Interval Algebra. *Journal of ACM*, 42(1):43–66, 1995.
15. Steven Prestwich. Local search on SAT-encoded colouring problems. In *SAT-03*, pages 105–119, 2003.
16. Barbara M. Smith and Martin E. Dyer. Locating the phase transition in binary constraint satisfaction problems. *Artificial Intelligence*, 81(1-2):155–181, 1996.
17. John Thornton, Matthew Beaumont, Abdul Sattar, and Michael Maher. A local search approach to modelling and solving Interval Algebra problems. *Journal of Logic and Computation*, 14(1):93–112, 2004.
18. Peter van Beek. Reasoning about qualitative temporal information. *Artificial Intelligence*, 58:297–326, 1992.
19. Peter van Beek and Robin Cohen. Exact and approximate reasoning about temporal relations. *Computational Intelligence*, 6:132–144, 1990.
20. Peter van Beek and Dennis W. Manchak. The design and experimental analysis of algorithms for temporal reasoning. *Journal of Artificial Intelligence Research*, 4:1–18, 1996.
21. Marc Vilain and Henry Kautz. Constraint propagation algorithms for temporal reasoning. In *AAAI-86*, pages 377–382, 1986.
22. Toby Walsh. SAT v CSP. In *CP-00*, pages 441–456, 2000.