Modelling and Solving Temporal Reasoning as Propositional Satisfiability *

Duc Nghia Pham, John Thornton, and Abdul Sattar

Institute for Integrated and Intelligent Systems
Griffith University, Queensland, Australia
{d.n.pham, j.thornton, a.sattar}@griffith.edu.au

Abstract. Recent research has shown that it is often preferable to encode real-world problems as propositional satisfiability (SAT) problems, and then solve using general purpose solvers. In this way the efficiencies of SAT technology can be exploited, and the development of specialised solution techniques can be avoided. However, in the interval algebra (IA) domain of temporal reasoning, the state-of-the-art still involves the use of specialised techniques that exploit the particular structure of interval relations.
In this paper we investigate the feasibility of representing and solving IA problems as SAT problems. We firstly introduce two methods of representing IA as a constraint satisfaction problem (CSP), and then use three SAT-encoding schemes to produce six different IA to SAT representations. In an empirical study, we examine the performance of existing SAT local and complete search solvers on these SAT representations, and perform a comparison with solvers that operate on native IA representations. Our results show that the best performance over a range of algorithms is produced using a support SAT encoding of a point algebra-based CSP. The results also show that a state-of-the-art complete SAT solver (zChaff) can solve these instances significantly faster than existing IA solvers working on equivalent native IA representations.

1 Introduction

Representation and reasoning with time information, or *temporal reasoning*, is a fundamental research area in computer science and artificial intelligence. Basic tasks in this domain include the design and development of efficient reasoning methods for determining consistency of temporal representations, and effectively answering temporal queries. More generally, results from temporal reasoning research have been successfully applied in many real world AI applications such as planning, plan recognition, natural language understanding, and medical diagnosis [1].

In this paper we are specifically concerned with the interval algebra (IA) representation of the temporal reasoning problem [2]. This is firstly because IA offers considerable expressivity in terms of representing qualitative information and secondly because it is the most popular and well studied temporal reasoning formalism. Existing IA temporal reasoning techniques are generally based on the backtracking approach (proposed by

* We would like to thank Peter van Beek and Jochen Renz for helpful comments on the earlier version of this paper.

Ladkin and Reinefeld [3]), which uses path consistency as forward checking. Although this approach has been further improved [1, 4], 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 complete search heuristics (such as unit propagation look ahead in Satz [5] or no-good recording and non-chronological backtracking in Chaff [6]) 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 [7, 8]. This has motivated research in applying stochastic local search techniques (SLS) to the IA problem, to see if performance increases over complete search observed in other SAT and CSP domains can be translated to IA. The first step in this direction was taken by Thornton *et al.* [8], with the development of the *end-point ordering* model, specifically designed to represent IA problems in a form suitable for processing by SLS. In this research the TSAT local search algorithm was shown to significantly outperform an existing complete search technique on a set of larger, more difficult IA problems. However, the end-point ordering model, like the native IA model, has a specialised structure that is carefully exploited by the TSAT algorithm. This means it is not a suitable representation for the application of general purpose techniques.

In this paper we ask the question whether the representation of IA problems using specialised models that require specialised algorithms is necessary in the general case. Given 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 propositional satisfiability problem. This enables us to apply a range of state-of-art SAT solvers and to compare the performance of these with the existing native IA approaches. According to our understanding, it appears that no explicit and thorough work has tried to formulate temporal problems as SAT instances. Nebel and Bürckert [9] 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]

A second issue addressed by the paper is the discovery of the best SAT representation for IA. This question divides into two parts: firstly the development of an appropriate CSP representation of IA (i.e. one suitable for the efficient application of general purpose solution techniques), and secondly the best choice of a CSP to SAT encoding. One of the main contributions of the paper is the development of a *point-based* CSP encoding of the IA problem which uses point algebra-based relations [11] but retains the full expressivity of IA. This is compared to a more straightforward interval-based model. We also extend the literature on the relative performance of existing CSP to SAT encodings by giving an empirical comparison of three encodings for each of our CSP models and for both complete and local search approaches.

The remainder of the paper is structured as follows: in the next section we review the basic definitions of IA, and in Section 3 we introduce the two methods to transform

[1] Recent independent work [10] has proposed representing IA as SAT, but the authors do not specify the transformation in detail, and fail to provide an adequate empirical evaluation.

IA instances to CSP instances. Using these two transformation methods, combined with three CSP to SAT encodings, six IA to SAT encodings are presented. In Section 4.1 we describe the generation of our test set instances and in Sections 4-4.5 we present an empirical study to evaluate the performance of these SAT encodings relative to each other, and the performance of existing complete and SLS SAT solvers in comparison to the native IA backtracking and TSAT solvers. Finally, Section 5 presents the conclusions and the future research suggested by this study.

2 Interval Algebra

Interval Algebra [2] 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 X can happen either before or after event Y"* can be expressed as $X\{b, bi\}Y$. Hence there are a total of $2^{|\mathcal{I}|}$ = 8,192 possible relations between pairs of temporal intervals.

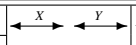
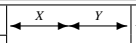
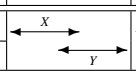
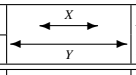
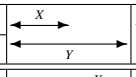
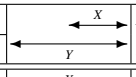
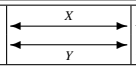
Atomic relation	Symbol	Diagram of meaning	PA representation
X before *Y*	b		$X^- < Y^-, X^- < Y^+$
Y after *X*	bi		$X^+ < Y^-, X^+ < Y^+$
X meets *Y*	m		$X^- < Y^-, X^- < Y^+$
Y met by *X*	mi		$X^+ = Y^-, X^+ < Y^+$
X overlaps *Y*	o		$X^- < Y^-, X^- < Y^+$
Y overlapped by *X*	oi		$X^+ > Y^-, X^+ < Y^+$
X during *Y*	d		$X^- > Y^-, X^- < Y^+$
Y includes *X*	di		$X^+ > Y^-, X^+ < Y^+$
X starts *Y*	s		$X^- = Y^-, X^- < Y^+$
Y started by *X*	si		$X^+ > Y^-, X^+ < Y^+$
X finishes *Y*	f		$X^- > Y^-, X^- < Y^+$
Y finished by *X*	fi		$X^+ > Y^-, X^+ = Y^+$
X equals *Y*	eq		$X^- = Y^-, X^- < Y^+$
			$X^+ > Y^-, X^+ = Y^+$

Table 1. The thirteen IA atomic relations

The four operators of IA: *union* (denoted by $\cup$), *intersection* (denoted by $\cap$), *inversion* (denoted by $^{-1}$), and *composition* (denoted by $\circ$), can be defined as follows:

$$\forall\, X, Y : X(R_1 \cup R_2)Y \leftrightarrow (XR_1Y \vee XR_2Y)$$

$$\forall\, X, Y : X(R_1 \cap R_2)Y \leftrightarrow (XR_1Y \wedge XR_2Y)$$

$$\forall\, X, Y : X(R_1^{-1})Y \leftrightarrow Y R_1 X$$

$$\forall\, X, Y : X(R_1 \circ R_2)Y \leftrightarrow \exists\, Z : (X R_1 Z \wedge Z R_2 Y).$$

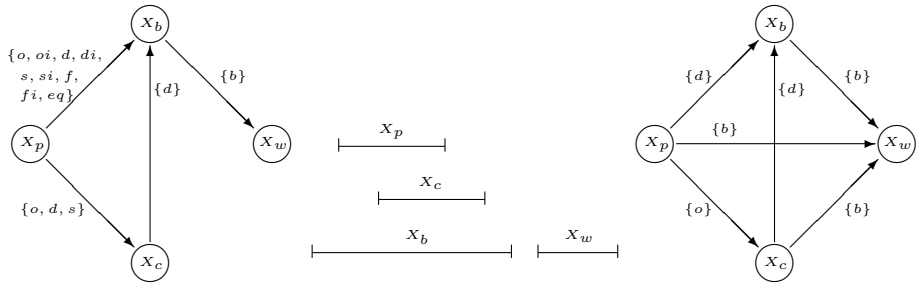
Hence, the *intersection* and *union* of any two temporal relations (R_1, R_2) are simply the standard set-theoretic intersection and union of the two sets of atomic relations describing R_1 and R_2, respectively. The *inversion* of a temporal relation R is the union of the inversion of each atomic relation $r_i \in R$. 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 are available in [2].

An IA network can be modelled as a temporal CSP (TCSP), where each interval event is a CSP variable with a domain of ordered pairs of real numbers and each binary constraint C_{ij} is labelled with the interval relations between the i^{th} and j^{th} intervals [1]. In general, the solution for a standard CSP is an assignment of domain values to all variables such that all the constraints are satisfied. Unfortunately, as the domain of an IA interval is infinite, this representation is not appropriate for a discrete domain CSP or SAT solver. However, the problem can be relaxed such that an $\mathcal{I}$-instantiation of a given IA network is an assignment of interval relations to all binary constraints in the corresponding TCSP. An $\mathcal{I}$-instantiation is *singleton*, also known as a *scenario*, iff each binary constraint is assigned with exactly a single atomic relation. An IA network with n interval variables is *globally consistent* iff it is strongly n-consistent [12]. Hence, the ISAT problem of determining whether a given IA network is satisfiable, becomes the problem of determining whether a *globally consistent* $\mathcal{I}$-instantiation of the corresponding TCSP exists [2, 1]. 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 [13] and it is one of the most important tasks in practical applications [4].

Figure 1(a) shows an example of an IA network expressing the situation: *"Fred was reading the paper while eating his breakfast. He put the paper down and drank the last of his coffee. After breakfast he went for a walk."*.[2] In this example, variables X_p, X_b, X_c and X_w represent the interval that Fred is *reading the paper*, *eating the breakfast*, *drinking coffee* and *walking* respectively. Figure 1(b) shows a consistent solution of this network on the timeline and figure 1(c) shows the corresponding $\mathcal{I}$-instantiation of that solution.

As ISAT is known to be NP-complete [11], the application of some sort of *exhaustive search method* is generally required to determine the satisfiability of a full IA network. Ladkin and Reinefeld [3] proposed an efficient backtracking approach to solve the ISAT problem by enforcing path consistency as forward checking at every branching node. This allows the elimination of relations that are path inconsistent with the current partial solution. They also pointed out that the instantiation of each constraint can be extended from atomic relations to any set of relations for which path consistency guarantees global consistency, and hence considerably reduced the branching factor of the algorithm. In addition, various variable and value ordering techniques were developed and empirically shown to significantly improve overall performance [7, 14, 1].

[2] This example was originally used in [7].



(a) IA network (b) timeline solution (c) consistent scenario

Fig. 1. An example of an IA network and its consistent solution.

Given the implicit constraints of the current IA formalism can only be enforced by path consistency, state-of-the-art search techniques from the CSP and SAT domains cannot be easily employed, i.e. it would require the embedding of a path consistency heuristic in the reasoning mechanism of the solver. To address this issue, Thornton *et al.* [8] developed a new transformation method, called *end-point ordering*, that reformulates IA networks into CSPs. In this model the variables are interval end-points and the constraints are the end-point relations as defined in Table 1. The domain of each end-point variable is defined as the integer value position or rank of that variable within the *total ordering of all end-points* [8]. To solve these problems, a specialised TSAT local search algorithm was developed that exploits the structure of the end-point domains and constraints. The main difficulty with this approach is the generation of very large variable domains (representing all possible orderings of each interval). Without the special TSAT pruning heuristics a standard general purpose solver would not prove competitive. Therefore, neither native IA or the end-point ordering models are appropriate for use with a general purpose SAT or CSP solver. For this reason we decided to explore the development of alternative representations that could produce variables with reasonable domain sizes and easily represented constraints.

3 Encoding IA into SAT

Recent research has shown that modelling and solving hard combinatorial problems as SAT instances can produce significant performance benefits over solving problems in their original form [15–17]. This at least indicates that encoding and solving IA problems as SAT instances using state-of-the-art SAT solvers is a promising line of enquiry.

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 of IA networks into SAT. This results in six different approaches to encode IA networks into SAT.

3.1 The Interval-Based Transformation Method

A straightforward method to formulate IA networks as CSPs is to represent each arc between a pair of intervals in the original IA network as a CSP variable. We then limit the domain values of each CSP 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 CSP variable from 2^{13} to a maximum of 13 values. In addition, an instantiation of the new model is now a singleton $\mathcal{I}$-instantiation, as a property of a CSP is that one and only one value can be assigned to a variable at any given time. Hence, the global constraint that an $\mathcal{I}$-instantiation has to be *globally consistent* becomes equivalent to it being *path consistent*, as a singleton $\mathcal{I}$-instantiation is globally consistent iff it is path consistent [7].

In his original work, Allen [2] proposed a path consistency algorithm for a regular $\mathcal{I}$-instantiation that repeatedly computes

$$R_{ik} = R_{ik} \cap (R_{ij} \circ R_{jk})$$

for all triples of intervals (i, j, k) until no further change occurs or until $R_{ik} = \emptyset$. These operations remove all the relations that cause an inconsistency between any triple (i, j, k) of intervals. If $R_{ik} = \emptyset$, then the original $\mathcal{I}$-instantiation is path inconsistent.

For a singleton $\mathcal{I}$-instantiation, the algorithm can be simplified, without loss of completeness, so that we only need to check

$$R_{ik} \subset (R_{ij} \circ R_{jk})$$

for all triples of interval $(i < j < k)$ once. The intersection ($\cap$) operation is unnecessary as R_{ik} is instantiated with exactly one atomic relation.

Formally, using this *interval*-based reduction method, the corresponding CSP of a given IA network is defined as follows:

Definition 1. *Given an IA network with n intervals, the corresponding interval-based CSP is (X, D, C), where $X = \{v_{ij} \mid i, j \in [1..n], i < j\}$; each variable v_{ij} represents a relation between two intervals i and j, having a domain $D_{ij} = R_{ij}$; and C consists of the following constraints:*

$$v_{ij} = x \wedge v_{jk} = y \Longrightarrow v_{ik} \in \{z_1, ..., z_m\}, \quad (i, j, k \in [1..n], i < j < k) \qquad (1)$$

where $\{z_1, ..., z_m\} = D_{ik} \cap (x \circ y)$. Note that R_{ij} is the IA relation between i and j; and $x, y \in R_{ij}$.

This complete interval-based reduction method requires $O(n^3)$ time, where n is the number of intervals.

3.2 The Point-Based Transformation Method

Vilain and Kautz [11] proposed the Point Algebra (PA) to model qualitative information between time points. PA consists of a set of 3 atomic relations $\mathcal{P} = \{<, =, >\}$ and four operators defined in the similar manner to IA. As an interval event X can be

represented as an ordered pair of end-points (X^-, X^+) where $X^- < X^+$, it follows that the relations between two intervals can be expressed as the relations between their end-points.

Although each atomic IA relation can be uniquely represented by a combination of relations on these points (see Table 1), 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, $(X^- \neq Y^-) \wedge (X^- < Y^+) \wedge (X^+ \neq Y^-) \wedge (X^+ < Y^+)$, represents not only $X\{b, d\}Y$ but also $X\{b, d, o\}Y$. Therefore, PA can only cover 2% of IA [1].

However, we can simply introduce new constraints to prevent the representation of undesired IA relations in the CSP model. Such constraints are formed using the negation of the PA representation of the undesired atomic IA relation.[3] For instance, to disallow $X\{o\}Y$ in the above example, we would include the constraint:

$$\neg(X^- < Y^-) \vee \neg(X^- < Y^+) \vee \neg(X^+ > Y^-) \vee \neg(X^+ < Y^+)$$

Formally, using this *point*-based reduction method, the corresponding CSP of a given IA network is defined as follows:

Definition 2. *Let μ^r_{st} be the PA representation for an atomic IA relation r between two intervals s and t. Given an IA network with n intervals, the corresponding point-based CSP is (X, D, C), where $X = \{v_{ij} \mid i, j \in [1..2n], i < j\}$; each variable v_{ij} represents a relation between two points i and j, having a domain $D_{ij} \in \mathcal{P}$; and C consists of the following constraints:*

$$v_{ij} = x \wedge v_{jk} = y \Longrightarrow v_{ik} \in \{z_1, ..., z_m\}, \quad (i, j, k \in [1..n], i < j < k) \quad (2)$$

$$\neg\mu^r_{st}, \quad (r \notin R_{st}) \quad (3)$$

where $\{z_1, ..., z_m\} = D_{ik} \cap (x \circ y)$. Note that x, y, z are PA atomic relations; s, t are intervals in the original problem and R_{st} is the relation between them.

3.3 Mapping CSP Representations of IA into SAT

Using either of the above CSP formulations, an IA network can be easily encoded as a SAT instance, where each Boolean variable v^r_{ij} represents an assignment of domain value r to a CSP variable v_{ij}. Two sets of at-least-one (ALO) and at-most-one (AMO) clauses can then be used to ensure that each CSP variable can only be instantiated with exactly one value at any time. It is common practice to encode a general CSP into SAT without the AMO clauses, thereby allowing CSP variables to be instantiated with more than one value [18]. A CSP solution can then be extracted by taking any single SAT-assigned value for each CSP variable. However, our two CSP reduction methods depend on the fact that each CSP variable can only be instantiated with exactly one value at any time. This maintains the completeness of formulation by ensuring not only the correctness of our translation of path consistency constraints but also the

[3] Table 1 shows the PA representations of 13 IA atomic relations.

global consistency of the solution. Hence, the AMO clauses cannot be removed from our translation.

A natural way to encode the path consistency constraints, i.e. constraints (1) and (2) above, is to add the following support (SUP) clause for each pair of domain values (x, y) for the two CSP variables $(\upsilon_{ij}, \upsilon_{jk})$:

$$\neg\upsilon_{ij}^{x} \vee \neg\upsilon_{jk}^{y} \vee \upsilon_{ik}^{z_1} \vee ... \vee \upsilon_{ik}^{z_m}$$

where $\{z_1, ..., z_m\} = D_{ik} \cap (x \circ y)$. 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 [11].

For the extra, but essential, constraints that forbid undesired IA relations in the point-based model, i.e constraints of type (3) that cannot be excluded in a standard PA representation, we add a forbidden (FOR) clause for each of undesired IA relations involving the CSP variable in the original problem. The FOR clause of an atomic relation r is defined based on the negation of the PA representation of that relation. For example, given the PA representation of $X\{o\}Y$ is

$$(X^- < Y^-) \wedge (X^- < Y^+) \wedge (X^+ > Y^-) \wedge (X^+ < Y^+)$$

then the forbidden clause to rule out the relation $X\{o\}Y$ is

$$\neg\upsilon_{X^-Y^-}^{<} \vee \neg\upsilon_{X^-Y^+}^{<} \vee \neg\upsilon_{X^+Y^-}^{>} \vee \neg\upsilon_{X^+Y^+}^{<}$$

We call the above the *support* encoding scheme as it encodes the support values of the original problem. In Gent's support encoding scheme [19], 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 $(i < j < k)$, and not for *all* permutation orders of this triple.

Formally, the SAT support encoding scheme for IA networks is defined as follows:

Proposition 1. *The support SAT-encoded instance of a given interval-based representation of an IA network consists of appropriate ALO, AMO and SUP clauses. The support SAT-encoded instance of a point-based representation is defined in a similar manner with the use of extra FOR clauses.*

Another way of representing CSP constraints as SAT clauses is to encode the conflict values, e.g. nogoods, between any pair of CSP variables [16, 18]. This *direct* encoding scheme for IA networks can be derived from our support encoding scheme by replacing the SUP clauses with the conflict (CON) clauses. If we represent SUP clauses between a triple of intervals $(i < j < k)$ as a 3D array of allowable values for the CSP variable υ_{ik} given the values of υ_{ij} and υ_{jk}, then the CON clauses can be defined as:

$$\neg\upsilon_{ij}^{x} \vee \neg\upsilon_{jk}^{y} \vee \neg\upsilon_{ik}^{z_m}$$

where $z_m \in \{D_{ik} - \{x \circ y\}\}$. The *multivalued* encoding [17] 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 the IA transformations the AMO clauses play a necessary role.

Proposition 2. *The direct SAT-encoded instance of a given IA network is derived from the support SAT-encoded instance of that network by replacing SUP clauses with the appropriate CON clauses.*

A more compact version of the direct encoding is the *log* encoding [16, 18]. Here, a Boolean variable x_{ij} is true iff the corresponding CSP variable X_i is assigned a value in which the j-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, the AMO and ALO clauses can be omitted. However, extra bitwise prevention (PRE) clauses are needed (if necessary) to prevent bitwise representations of undesired Boolean variables from being instantiated. For example, if the domain of variable X is 3 then we have to add the clause $\neg x_{30} \vee \neg x_{31}$ to prevent the fourth value from assigning to X. As with the direct encoding, there is a version of the log encoding that allows the original CSP variable to have more than one instantiation in the SAT encoding. This *binary* encoding [20] was not considered for the same reason that the multi-valued encoding was rejected previously.

Proposition 3. *The log SAT-encoded instance of a given IA network is derived from the direct SAT-encoded instance of that network by replacing each Boolean variable with its bitwise representation, removing all AMO and ALO clauses and adding PRE clauses as necessary.*

4 Experimental Study

4.1 Generating SAT-encoded Test Instances

As a large collection of IA benchmarks is not available, the majority of work in the area has relied on randomly generated IA problems [4, 1]. This reliance on randomly generated problems has the advantage of allowing the average difficulty of a problem set to be controlled. We therefore based our empirical study on random problems generated using Nebel's $A(n, d, s)$ model [1]. The resulting instances are temporal constraint graphs with n nodes (i.e. intervals) and an average degree of d constrained arcs (i.e. interval relations). Constrained arcs are then labelled with an average of s IA atomic relations, where $1 \leq s \leq 12$. Unconstrained arcs are labelled with all 13 IA atomic relations.

Unlike the $S(n, d, s)$ model used in earlier studies (e.g. [8]), the $A(n, d, s)$ model allows us to control the difficulty of generated problems. Nebel [1] suggested that the problem instances can be generated in the phase transition using the $A(n, d, s)$ model with the values of d and s fixed to 9.5 and 6.5, respectively. It is worthy to note that the phase transition of a problem type is the border between two regions: one is where most of the problems have many solutions and it is relatively easy to solve; and one is where most of the problems are unsatisfiable and it is also relatively easy to prove [21]. In addition, research has empirically showed that instances in the phase transition are harder to solve for both complete and incomplete search solvers. We therefore used these settings to limit the study to only consider harder instances.

To investigate the performance effects of our six different SAT-encoding schemes, we followed Nebel's original study and randomly generated three test sets of 20, 30 and 40 nodes, each containing 100 satisfiable instances. We then pre-processed these instances using the path consistency algorithm before encoding them into SAT. We found this pre-processing significantly reduced the problem size of SAT-encoded IA instances both in terms of the number of variables and clauses.

Table 2 shows the average problem size of these instances together with the average time required to encode them into SAT instances. These results indicate the point-based support encoding can produce the smallest SAT-encoded instances within the shortest time window.

Problem	Reduction	Encoding	#vars	#clauses	time (secs)
$n = 20$	Interval	support	$1,685$	$75,667$	0.06
$d = 9.5$	based	direct	$1,685$	$695,713$	0.19
$s = 6.5$		log	639	$688,180$	0.43
	Point	support	$1,954$	$42,157$	0.02
	based	direct	$1,954$	$81,094$	0.03
		log	$1,299$	$79,264$	0.03
$n = 30$	Interval	support	$4,484$	$371,486$	0.19
$d = 9.5$	based	direct	$4,484$	$3,697,416$	0.83
$s = 6.5$		log	$1,577$	$3,675,095$	2.37
	Point	support	$4,741$	$173,746$	0.06
	based	direct	$4,741$	$340,366$	0.10
		log	$3,146$	$335,800$	0.14
$n = 40$	Interval	support	$8,421$	$1,019,320$	0.45
$d = 9.5$	based	direct	$8,421$	$10,415,055$	2.26
$s = 6.5$		log	$2,877$	$10,351,622$	6.80
	Point	support	$8,637$	$443,354$	0.14
	based	direct	$8,637$	$873,715$	0.23
		log	$5,744$	$865,344$	0.36

Table 2. Interval versus Point Based Encodings: A comparison on problem generation.

4.2 SAT Solver Selection

Table 3 shows the results of zChaff [6], PAWS [22] and MV-PAWS [23] for the three test sets generated in Section 4.1.[4] We ran PAWS and MV-PAWS for $1,000$ runs on each 20 node instance and 100 runs on each 30 and 40 instance. All three methods were timed out after $1,200$ seconds for each run.

[4] All our experiments were performed on a Sun supercomputer with $8 \times$ Sun Fire V880 servers, each with $8 \times$ UltraSPARC-III 900MHz CPU and 8GB memory per node.

We chose PAWS as our SLS solver [22], as PAWS was recently shown to be one of the most competitive SAT solvers on a range of larger and more difficult problems, while also requiring considerably less effort in terms of parameter tuning than other comparable weighting schemes. For complete search, we chose zChaff [6] version 2004.11.25 as it has won the championship in the SAT competition for three consecutive years. In addition, we included a recently developed version of PAWS, MV-PAWS [23], which implements the same local search heuristic as PAWS, but is specifically designed to exploit CSP structure in SAT-encoded instances. MV-PAWS does this by automatically recognising the structure of CSP variables in a SAT problem and ensuring that each underlying CSP variable is only ever instantiated with a single value during the search. This built-in MV-PAWS mechanism means it can discard all ALO and AMO clauses, as they are now redundant.

			zChaff	PAWS			MV-PAWS		
Problem	Reduction	Encoding	$\frac{mean}{time}$	Param	$\frac{mean}{time}$	$\frac{mean}{flips}$	Param	$\frac{mean}{time}$	$\frac{mean}{flips}$
$n = 20$	Interval	support	0.17	33	1.42	67,441	100	0.85	15,955
$d = 9.5$	based	direct	2.66	29	18.04	133,572	n/a	n/a	n/a
$s = 6.5$		log	1,200	67	65.82	190,436	n/a	n/a	n/a
	Point	support	0.07	52	1.13	124,388	100	0.26	11,696
	based	direct	0.17	50	3.53	246,886	n/a	n/a	n/a
		log	3.49	68	1.16	54,734	n/a	n/a	n/a
$n = 30$	Interval	support	2.17	40	42.83	533,668	100	20.59	134,887
$d = 9.5$	based	direct	84.54	31	147.94	382,124	n/a	n/a	n/a
$s = 6.5$		log	1,200	69	271.55	278,265	n/a	n/a	n/a
	Point	support	0.53	61	23.28	941,819	100	3.53	67,462
	based	direct	1.95	60	78.80	1,691,735	n/a	n/a	n/a
		log	104.90	70	28.57	440,037	n/a	n/a	n/a
$n = 40$	Interval	support	14.61	40	210.38	1,274,465	100	120.13	389,524
$d = 9.5$	based	direct	354.99	n/a	n/a	n/a	n/a	n/a	n/a
$s = 6.5$		log	1,200	n/a	n/a	n/a	n/a	n/a	n/a
	Point	support	3.58	80	179.14	2,821,202	100	34.36	287,986
	based	direct	13.74	n/a	n/a	n/a	n/a	n/a	n/a
		log	914.36	n/a	n/a	n/a	n/a	n/a	n/a

Table 3. A comparison of Interval versus Point Based Encodings.

4.3 Results for Different Encodings

Overall, the results in Table 3 and graphed in Figure 2 clearly show that the point-based transformation produced better results that the interval-based encoding, regardless of problem size, solver or the SAT encoding method employed. Similarly, the support

encoding produced the best performance of all three SAT encoding schemes, regardless of problem size, solver[5] or transformation method (this is consistent with earlier work of [19]). Putting this together leads to the strong conclusion that a combination of point-based transformation and support encoding produces the best results, at least for the randomly generated IA problems considered.

The superior performance of the 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 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 [16]. A further possible reason for the superiority of the support encoding (suggested by Gent [19]) is the reduced bias to falsify clauses, i.e. the numbers of positive and negative literals in support encoding are more balanced than in direct encoding and hence this may prevent the search from resetting variables to false shortly after they are set to true.

Although our experiments confirmed the superiority of direct over log encoding (as per Hoos [16]) for the interval-based transformation, our results differed for the point-based transformation, where log encoding was found to be better than direct encoding, both in terms of runtimes and flips. To investigate this further, we used two features of the search space originally developed in [16]'s study: the standard deviation of the objective function (*sdnclu*) and the local minima branching along SLS trajectories (*blmin*) (we did not look at the solution density as this was the same for both encodings). Intuitively, the larger the *sdnclu* and *blmin* values are, the more effective the SLS solver. However, this hypothesis did not fit with our results. This may be explained by the reduced size of the point-based log clauses, resulting from the smaller number of PA atomic relations (in comparison to IA).

4.4 Results for SAT Solvers

As with the encoding results, a comparison of the relative performance of the three solvers produces a fairly unambiguous picture: zChaff outperforms MV-PAWS, and MV-PAWS outperforms PAWS, with relative performance being fairly independent of the transformation or SAT encoding method employed. The superior performance of MV-PAWS over PAWS confirms the results of [23], where MV-PAWS had a similar advantage on a range of SAT-encoded non-temporal CSP instances. This also confirms the results of earlier studies showing that SAT algorithms which exploit CSP structure generally produce better performance (e.g. [20, 24]).

However, the overall domination of zChaff is more surprising, as the earlier TSAT study [8] indicated that local search has an advantage over complete search in the temporal domain. In these results, zChaff is not only better across the board, but appears to be scaling as well or better than both of the local search approaches. It should also be considered that there is a version of zChaff [24] (analogous to MV-PAWS) that also exploits underlying CSP structure, and we would expect this solver, when released, to produce even better performance on these SAT-encoded instances.

[5] As the support encoding was so strongly favoured by PAWS and zChaff we did not test MV-PAWS on the other encodings.

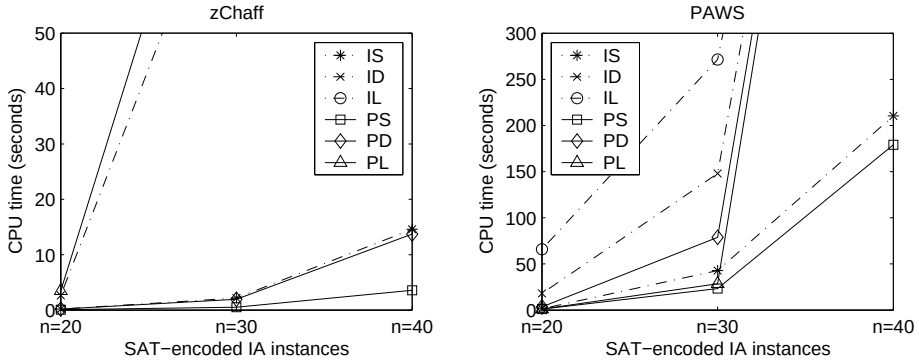


Fig. 2. Scalability of zChaff and PAWS on different SAT encodings.

4.5 SAT versus Existing Approaches

The experimental results in the previous section have clearly shown a point-based transformation using a support SAT encoding is the better encoding, that MV-PAWS is the better local search solver, and that zChaff is the better overall solver. The final question to address is how these SAT approaches compare to the existing state-of-the-art specialised IA solvers, namely TSAT and Nebel's backtracking/path consistency algorithm.

For this study we generated three test sets of 80, 90 and 100 nodes using the same method discussed in Section 4.1 (with each set containing 10 satisfiable instances). In addition to using zChaff and MV-PAWS on support encoded point-based transformations, we used TSAT on its end point ordering representation and used two variants of Nebel's backtracking algorithm (NBT+$\mathcal{I}$ and NBT+$\mathcal{H}$), on the native IA representations. 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. Other heuristics used in Nebel's backtracking algorithm were set to default.

The results in Table 4 provide strong evidence that the approach of modelling IA problems as SAT has met with success. In particular, zChaff was the only method capable of solving all instances in the problem set within 1 hour. While Nebel's NBT+$\mathcal{H}$) was superior on the 80 node problems, it failed to scale up on the 90 and 100 node instances, and while TSAT remained competitive with zChaff in terms of median time, it proved unable to consistently solve all instances (falling to 76% success on the 100 node problems). Of secondary interest is that TSAT does appear to have the advantage over MV-PAWS, especially on the larger instances.

5 Conclusions and Future Work

In conclusion, the experiments indicate that our new point-based support encoding is the most suitable scheme to encode IA problems into SAT instances. The results further show that running a state-of-the-art complete SAT solver on such representations can

		Success	Time (secs)	
Problem	Solver	100%	$median$	$mean$
$n = 80$	zChaff	100	191.46	244.04
$d = 9.5$	MV-PAWS	90	416.41	925.41
$s = 6.5$	TSAT	87	27.01	594.66
	NBT+$\mathcal{H}$	100	64.81	210.06
	NBT+$\mathcal{I}$	50	3,309.31	3,024.78
$n = 90$	zChaff	100	393.77	576.67
$d = 9.5$	MV-PAWS	85	944.91	1,364.66
$s = 6.5$	TSAT	82	539.88	915.25
	NBT+$\mathcal{H}$	60	1,792.86	2,238.46
	NBT+$\mathcal{I}$	30	3,600.00	2,581.89
$n = 100$	zChaff	100	1,132.23	1,120.03
$d = 9.5$	MV-PAWS	64	2,059.51	2,229.51
$s = 6.5$	TSAT	76	1,307.80	1,525.73
	NBT+$\mathcal{H}$	30	3,600.00	2,571.44
	NBT+$\mathcal{I}$	20	3,600.00	3,270.14

Table 4. A comparison of SAT versus Existing Approaches.

produce superior results to two of the fastest specialised IA solvers reported in the current literature. This suggests that a SAT approach to solving larger and more difficult IA problems may be preferable to developing specialised representations and algorithms.

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. The possibility also opens up of integrating our approach to temporal reasoning into other well known real world problem domains such as planning. Given the success of SAT solvers in many other real world domains, our work promises to expand the reach of temporal reasoning approaches for IA to encompass larger and more practical problems.

References

1. Nebel, B.: Solving hard qualitative temporal reasoning problems: Evaluating the efficiency of using the ORD-Horn class. Constraints **1** (1997) 175–190
2. Allen, J.: Maintaining knowledge about temporal intervals. Communications of the ACM **26** (1983) 832–843
3. Ladkin, P., Reinefeld, A.: Effective solution of qualitative interval constraint problems. Artificial Intelligence **57** (1992) 105–124
4. van Beek, P., Manchak, D.: The design and experimental analysis of algorithms for temporal reasoning. Journal of Artificial Intelligence Research **4** (1996) 1–18
5. Li, C., Anbulagan: Look-ahead versus look-back for satisfiability problems. In: Proceedings of the Third International Conference on Principles and Practice of Constraint Programming (CP-97). (1997) 341–355

6. Moskewicz, M., Madigan, C., Zhao, Y., Zhang, L., Malik, S.: Chaff: Engineering an efficient SAT solver. In: Proceedings of the 38th Design Automation Conference (DAC-01), Las Vegas (2001)
7. van Beek, P.: Reasoning about Qualitative Temporal Information. Artificial Intelligence **58** (1992) 297–326
8. Thornton, J., Beaumont, M., Sattar, A., Maher, M.: A local search approach to modelling and solving Interval Algebra problems. Journal of Logic and Computation **14** (2004) 93–112
9. Nebel, B., Bürckert, H.J.: Reasoning about temporal relations: A maximal tractable subclass of Allen's Interval Algebra. Journal of ACM **42** (1995) 43–66
10. Ghiathi, K., Ghassem-Sani, G.: Using satisfiability in temporal planning. WSEAS Transactions on Computers **3** (2004) 963–969
11. Vilain, M., Kautz, H.: Constraint propagation algorithms for temporal reasoning. In: Proceedings of the Fifth National Conference on Artificial Intelligence (AAAI-86). (1986) 377–382
12. Mackworth, A.K.: Consistency in networks of relations. Artificial Intelligence **8** (1977) 99–118
13. Golumbic, M., Shamir, R.: Complexity and algorithms for reasoning about time: A graph-theoretic approach. Journal of ACM (1993) 1108–1133
14. Ladkin, P., Reinefeld, A.: Fast algebraic methods for interval constraint problems. Annals of Mathematics and Artificial Intelligence **19** (1997) 383–411
15. Kautz, H., McAllester, D., Selman, B.: Encoding plans in propositional logic. In: Proceedings of the Fifth International Conference on Principles of Knowledge Representation and Reasoning (KR-96). (1996) 374–384
16. Hoos, H.: SAT-encodings, search space structure, and local search performance. In: Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99). (1999) 296–302
17. Prestwich, S.: Local search on SAT-encoded colouring problems. Lecture Notes in Computer Science (2003)
18. Walsh, T.: SAT v CSP. In: Proceedings of the Sixth International Conference on Principles and Practice of Constraint Programming (CP-00). (2000) 441–456
19. Gent, I.: Arc consistency in SAT. In: Proceedings of the Fifteenth European Conference on Artificial Intelligence (ECAI-02). (2002) 121–125
20. Frisch, A., Peugniez, T.: Solving non-Boolean satisifiability problems with stochastic local search. In: Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-01). (2001)
21. Cheeseman, P., Kanefsky, B., Taylor, W.: Where the really hard problems are. In: Proceedings of the Twelveth International Joint Conference on Artificial Intelligence (IJCAI-91). (1991) 331–337
22. Thornton, J., Pham, D.N., Bain, S., Ferreira Jr., V.: Additive versus multiplicative clause weighting for SAT. In: Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-04). (2004) 191–196
23. Pham, D.N., Thornton, J., Sattar, A., Ishtaiwi, A.: SAT-based versus CSP-based constraint weighting for satisfiability. In: Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-05). (2005) to appear
24. Ansótegui, C., Larrubia, J., Manyà, F.: Boosting chaff's performance by incorporating CSP heuristics. In: Proceedings of the Ninth International Conference on Principles and Practice of Constraint Programming (CP-03). (2003) 96–107