

SAT-based Procedures for Temporal Reasoning

Alessandro Armando^{1,2}, Claudio Castellini³, and Enrico Giunchiglia¹

¹ DIST – Università di Genova, Viale Causa 13 – 16145 Genova – Italia

² LORIA-INRIA, 615, rue du Jardin Botanique – 54602 Villers les Nancy – France

³ Div. of Informatics, U. of Edinburgh, 80 South Bridge, Edinburgh EH1 1HN, UK

Abstract. In this paper we study the consistency problem for a set of disjunctive temporal constraints [Stergiou and Koubarakis, 1998]. We propose two SAT-based procedures, and show that—on sets of binary randomly generated disjunctive constraints—they perform up to 2 orders of magnitude less consistency checks than the best procedure presented in [Stergiou and Koubarakis, 1998]. On these tests, our experimental analysis confirms Stergiou and Koubarakis’s result about the existence of an easy-hard-easy pattern whose peak corresponds to a value in between 6 and 7 of the ratio of clauses to variables.

1 Introduction

Temporal reasoning is a traditional area of research, involved in planning (see, e.g., [Allen *et al.*, 1991]) scheduling ([Cheng and Smith, 1994]) and other application areas such as temporal databases ([Brusoni *et al.*, 1996]). One of the most studied problems in temporal reasoning (see, e.g., [Dechter *et al.*, 1991]) involves reasoning about sets of formulas of the form

$$l_1 \leq x - y \leq u_1 \vee \dots \vee l_n \leq x - y \leq u_n, \quad (1)$$

($n \geq 1$) where l_i, u_i are real numbers and x, y are variables ranging over the reals. The *consistency problem* is to determine whether there exists an assignment to variables satisfying all the given formulas. More recently, Stergiou and Koubarakis [Stergiou and Koubarakis, 1998]:

- Extended the framework studied by, e.g., [Dechter *et al.*, 1991], so to allow also *disjunctive temporal constraints*, i.e. formulas of the form

$$x_1 - y_1 \leq u_1 \vee \dots \vee x_n - y_n \leq u_n, \quad (2)$$

($n \geq 1$) where $x_1, y_1, x_2, y_2, \dots, x_n, y_n$ are variables ranging over the reals, and u_1, u_2, \dots, u_n are real numbers. Notice that (1) involves only two variables, while this no longer holds for (2). This added generality may be useful in fields such as scheduling, management of temporal databases and natural language processing ([Koubarakis, 1997, Stergiou and Koubarakis, 1998]).

- Proposed four progressively more efficient algorithms for solving the consistency problem for sets of disjunctive temporal constraints. Essentially, they use (i) (variations of) Smullyan’s tableau procedure for propositional logic [Smullyan, 1968] to incrementally generate sets of formulas of the form $x - y \leq u$, each set propositionally satisfying the given set of disjunctive temporal constraints, and (ii) Chleq’s procedure [Chleq, 1995] to check the consistency of the assignments generated. The proposed algorithms differ in the heuristics adopted for performing the first step.
- On the ground of a theoretical analysis and experimental evaluations conducted using variously generated random tests they concluded that forward checking [Haralick and Elliott, 1980] gives the best performance. The forward checking heuristics amounts to valuating to false (and hence eliminating from the problem at hand) each disjunct whose negation is entailed by the assignment generated so far. In the following, we use SK to denote Stergiou and Koubarakis’s procedure implementing forward checking.
- Finally, by plotting the number of consistency checks performed against the ratio of clauses to variables, they observed an easy-hard-easy pattern centered in between 6 and 7. Furthermore they noticed that the hard region does not coincide with the transition from soluble to insoluble problems.

In this paper we propose a SAT-based procedure for solving the consistency problem for a set of disjunctive temporal constraints. According to the SAT-based methodology [Giunchiglia and Sebastiani, 1996, Giunchiglia *et al.*, 2000], we still first

- *generate* a (possibly partial) valuation for the disjuncts which propositionally satisfies the input clauses, and then
- *test* that the generated valuation is indeed consistent.

However, given that the generation step involves propositional reasoning only, we adopt a state-of-the-art SAT decider for generating valuations. Because of this, we inherit the many optimizations and heuristic strategies (improving the average case behavior) which are implemented in current SAT solvers. Thus, we first propose a procedure, TSAT^{FC} , which differs from SK in that we use a state-of-the-art SAT solver in place of the tableau-like procedure implemented in SK. A comparison with SK reveals that TSAT^{FC} performs up to 1 order of magnitude less consistency checks than SK. Furthermore, by looking at the trace of TSAT^{FC} , we discovered that many of the failed consistency checks are caused by the presence of pairwise mutually exclusive constraints, and observed that such consistency checks (and consequent possible failures) are performed over and over in different parts of the search tree. To avoid such redundant computations, we have implemented a preprocessing routine (called IS(2)) which checks the consistency of each pair of disjuncts, say c_1 and c_2 , involving the same pair of variables: whenever c_1 and c_2 are mutually inconsistent, the clause $(\neg c_1 \vee \neg c_2)$ is added to the problem. The resulting procedure, $\text{TSAT}_{\text{IS}(2)}^{\text{FC}}$, performs up to 1 order of magnitude less consistency checks than TSAT^{FC} , and thus up to 2 orders less than SK.

```

function NAIVETSAT( $\varphi$  : DTP) : boolean
  loop do
    choose a valuation  $\mu$  P-satisfying  $\varphi$ 
    if there is no such valuation then return False;
    if ( $\mu$  is T-consistent) then return True;
  end

```

Fig. 1. NAIVETSAT

The paper is structured as follows. In Section 2, we introduce the basic concepts and terminology that will be used in the paper, and present our SAT-based procedures. In Section 3 we describe our experimental results: we first review the testing methodology presented in [Stergiou and Koubarakis, 1998] and used to evaluate SK; we then carry out a comparative analysis of our procedures against SK using sets of variously generated binary constraints, and show the 2 orders of magnitude improvement. In Section 4 we highlight the differences of the tested procedures, and explain their different behaviors. We end the paper with some concluding remarks in Section 5.

2 SAT-based Procedures for Temporal Reasoning

A *temporal constraint* is a linear inequality of the form $x - y \leq r$, where x and y are variables ranging over the real numbers and r is a real constant. A *disjunctive temporal constraint* is a disjunction of the form $c_1 \vee \dots \vee c_n$ where c_1, \dots, c_n are temporal constraints, and $n \geq 1$. A *disjunctive temporal problem* (DTP) is a finite set of disjunctive temporal constraints to be intended conjunctively. An *assignment* is a function which maps each variable into a real number. An assignment σ *T-satisfies*

- a temporal constraint $x - y \leq r$ if it is indeed the case that $\sigma(x) - \sigma(y) \leq r$;
- a disjunctive temporal constraint if it T-satisfies at least one of the disjuncts occurring in it;
- a DTP if it T-satisfies each of its elements.

We say that a DTP is *T-consistent* if there exists an assignment which T-satisfies it, and that it is *T-inconsistent* if no such an assignment exists. A *literal* is either a temporal constraint, e.g. $x_1 - x_2 \leq 3$, or the negation of a temporal constraint, e.g. $\neg(x_2 - x_1 \leq 5)$. A *clause* is a disjunction of literals. Thus disjunctive temporal constraints are clauses without occurrences of the negation symbol. A *valuation* is a set of literals, e.g., $\{x_1 - x_2 \leq 3, \neg(x_2 - x_1 \leq 5)\}$. If c is a temporal constraint, then \bar{c} abbreviates $\neg c$ and $\neg\bar{c}$ abbreviates c . We say that a valuation μ *P-satisfies* a DTP φ if μ entails φ by propositional reasoning only.

The simplest and most general formulation of a SAT-based decision procedure for checking the consistency of a DTP φ is given by the non-deterministic algorithm NAIVETSAT given in Figure 1. The soundness of the algorithm relies

```

function TSAT( $\varphi$  : DTP,  $\mu$  : VALUATION): boolean
1: if  $\varphi = \{\}$  then return TCONSIST( $\mu$ );
2: if  $\{\} \in \varphi$  then return False;           /* backtrack */
3: if unit_clause?( $\varphi, l$ ) then             /* unit */
    return TSAT(simplify( $\varphi, l$ ),  $\mu \cup \{l\}$ );
4:  $l :=$  choose_literal( $\varphi$ );
5: return TSAT(simplify( $\varphi, l$ ),  $\mu \cup \{l\}$ ) or /* split */
    TSAT(simplify( $\varphi, \bar{l}$ ),  $\mu \cup \{\bar{l}\}$ );

```

Fig. 2. TSAT: a sketch of the algorithm

on the observation that if a valuation P-satisfying φ is T-consistent, then φ is T-consistent. As far as the completeness of the algorithm is concerned, it suffices to notice that if none of the valuations P-satisfying φ is T-consistent, then φ is not T-consistent. The efficiency of the algorithm rests on the way the valuations P-satisfying φ are chosen. Notice that this is essentially the propositional satisfiability problem, for which very efficient procedures are available.

We developed a SAT-based decision procedure for checking the T-satisfiability of DTPs, TSAT, on top of Böhm’s procedure, the winner of a 1992 competition involving various SAT-solvers [Buro and Buning, 1992]¹. An abstract presentation of TSAT is given in Figure 2. Given a DTP φ and a valuation μ (initially set to the empty set), TSAT generates the valuations P-satisfying the input DTP φ , whereas the call to TCONSIST checks the T-consistency of the valuations generated by TSAT. *unit_clause?*(φ, l) checks the existence of a temporal constraint $\{lit\} \in \varphi$ (a “unit clause” in SAT terminology), and assigns the literal *lit* to the variable *l* occurring as its second argument. If a temporal constraint *l* belongs to φ , then any valuation P-satisfying φ must necessarily contain *l*. *simplify*(φ, l) simplifies φ by

1. first removing all the disjunctive temporal constraint containing *l* as a disjunct, and
2. then replacing all the disjunctive temporal constraints of the form $cl \cup \{\bar{l}\}$ with *cl*.

Finally *choose_literal*(φ) selects the literal (from those occurring in φ) which is the best candidate for splitting according to some heuristic criterion.

Forward Checking (FC) can be incorporated into TSAT in various ways. At the abstract level that we have used so far, it is enough to replace the two calls of the form *simplify*(φ, l) in the last two lines, with *fc-simplify*($\varphi, \mu \cup \{l\}$). *fc-simplify*($\varphi, \mu \cup \{l\}$) accomplishes a simplification activity similar to *simplify*’s one, the only difference being that it performs an additional step:

¹ Böhm’s procedure is an efficient implementation of the Davis & Putnam procedure [Davis and Putnam, 1960] which is available at the URL http://www.informatik.uni-koeln.de/lis_juenger/boehm.html.

3. replacing all the disjunctive temporal constraints of the form $cl \cup \{l'\}$ with cl if $\mu \cup \{l\} \cup \{l'\}$ is not T-consistent.

The check for T-consistency can be readily carried out by invoking TCONSIST. *fc-simplify* is computationally more expensive than *simplify* since it requires a number of T-consistency checks roughly equal to the number of literals occurring in φ . However in many cases the stronger form of simplification carried out by *fc-simplify* has the positive effect of pruning the search space in a significant way. We call TSAT^{FC} the TSAT algorithm modified in this way.

A common feature of TSAT and TSAT^{FC} is that all the consistency checks are carried on-line. An alternative is to preprocess the DTP and look for sets of temporal constraints in the input DTP which are not T-consistent. If S is one of such sets, the clause $\bigvee_{c \in S} \neg c$ can be added to the DTP at hand without affecting its consistency. If n is a positive integer, we call $\text{IS}(n)$ the routine which

1. checks for T-consistency all the sets containing up to n temporal constraints taken from those occurring in the input DTP, and
2. extends the input DTP with the clauses encoding the T-inconsistent sets.

Of course, such a preprocessing step makes sense only for small values of n . In our experience, we got benefits for $n = 2$. In this case, the $\text{IS}(2)$ routine can be restricted to check only pairs of temporal constraints which involve the same pair of variables with opposite signs. (Pairs of temporal constraints involving more than two variables are trivially T-consistent.) Also notice that each performed consistency check is very simple, and can be performed in constant time. We call $\text{TSAT}_{\text{IS}(2)}^{\text{FC}}$ the procedure obtained by adding the $\text{IS}(2)$ preprocessing routine to TSAT^{FC} .

3 Experimental evaluation

In order to compare experimentally our algorithms with SK, we adopted the same random generation model employed in [Stergiou and Koubarakis, 1998]. Sets of DTPs are generated in terms of the tuple of parameters $\langle k, n, m, L \rangle$, where k is the number of disjuncts per clause, n is the number of arithmetic variables occurring in the problem, m is the number of clauses, and finally L is a positive integer number such that all the constants are taken from the interval $[-L, L]$. For a given $\langle k, n, m, L \rangle$, a DTP is produced by randomly generating m clauses of length k . Each clause, $x_1 - y_1 \leq r_1 \vee \dots \vee x_k - y_k \leq r_k$, is constructed by generating each disjunct $x_i - y_i \leq r_i$ by randomly choosing x_i and y_i with probability $1/n$ out of n variables (but discharging pairs of identical variables) and taking r_i to be a randomly selected integer in the interval $[-L, L]$. Furthermore, clauses with two (or more) identical disjuncts are discharged.

As in [Stergiou and Koubarakis, 1998] we generated our set of tests by taking $k = 2$ and $L = 100$. The class of binary disjunctive constraints is of particular interest since many problems from planning and scheduling can be mapped in such a class. The results of such experiments are plotted in the graphs of Figure 3

for $n = 10, 12, 15,$ and 20 . Each curve represents the total number of consistency checks versus the ratio of clauses to variables, $r = m/n$, which ranges from 2 to 14. All points are the median value among 100 randomly generated samples.² Each graph has curves for SK³, TSAT^{FC}, and TSAT^{FC}_{IS(2)} in it. For TSAT^{FC}_{IS(2)} we also consider the number of consistency checks performed by IS(2) which grows quadratically as r increases.⁴

As shown by the graphs, the algorithms show qualitatively similar behaviors (which are in turn similar to those given in [Stergiou and Koubarakis, 1998]). However TSAT^{FC} performs uniformly better than SK, and TSAT^{FC}_{IS(2)} performs uniformly better than TSAT^{FC}. Moreover, the gap in performance between the algorithms increases with the difficulty of the problems (i.e. as the values of n increases). Considering the median number of consistency checks performed by the procedures,

- for $n = 10$ and $r > 6$, TSAT^{FC} performs roughly one third of the consistency checks performed by SK, and TSAT^{FC}_{IS(2)} performs roughly half of the consistency checks performed by TSAT^{FC}.
- for $n = 20$ and $r > 6$, TSAT^{FC} performs almost 1 order of magnitude less consistency checks than SK, and TSAT^{FC}_{IS(2)} performs almost 1 order of magnitude less consistency checks than TSAT^{FC}, and hence almost 2 orders less than SK.

Our experimental results show that our procedures (on these tests) present an easy-hard-easy pattern. In most cases, the curves representing our procedures reach their peak when $r = 6$, while for SK the peak is usually obtained when $r = 7$. However, considering the plots corresponding to $n = 20$ (which are much sharper than the others) we agree with Stergiou and Koubarakis’s conjecture

² For practical reasons, we have used a timeout mechanism to stop the execution of a system on a DTP after 1000 seconds of CPU time. Because of this, we cannot plot the value representing the media of the consistency checks performed by the systems.

³ We have used an implementation of SK which has been kindly made available to us by Kostas Stergiou.

⁴ More in detail, assume that the number of temporal constraints N , be equal to $2m$ (which is in turn equal to $2nr$, since $r = m/n$). This assumption is reasonable since the probability of having two occurrences of the same temporal constraint in a DTP is negligible. For instance, when $n = 20$ and $r = 14$ the probability is about 0.015. There are $c = \frac{N^2 - N}{2} = 2r^2n^2 - rn$ possible pairs of temporal constraints, and the probability that two of them involve the same two variables with opposite signs is

$$p_c = \frac{1}{2} \frac{1}{\binom{n}{2}} = \frac{1}{n^2 - n}.$$

Therefore the average number of performed consistency checks is given by

$$c \cdot p_c = \frac{2r^2n - r}{n - 1}.$$

about the existence of an easy-hard-easy pattern whose peak is obtained for a value of r in between 6 and 7. Furthermore, considering all the plots, it seems that

- the 50% of T-satisfiable DTPs is obtained when $5 \leq r \leq 6$, and
- the transition from solvable to unsolvable problems becomes steeper as n increases.

To better highlight these facts, we have run $\text{TSAT}_{\text{IS}(2)}^{\text{FC}}$ on the two additional set of tests corresponding to $n = 25$ and $n = 30$. Results of these test sets are shown in Figure 4.

4 Explaining SK, Tsat^{FC} and $\text{Tsat}_{\text{IS}(2)}^{\text{FC}}$ behaviors

4.1 SK vs. Tsat^{FC}

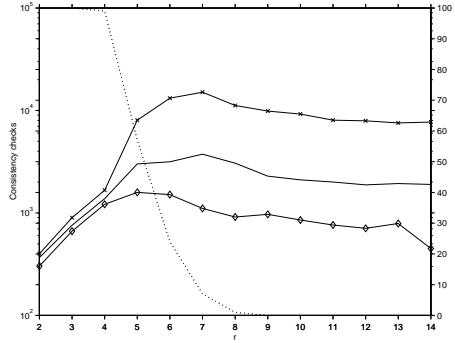
To illustrate why TSAT^{FC} performs better than SK, consider the following DTP φ :

$$\begin{aligned} x_1 - x_2 &\leq 3 \\ x_1 - x_3 &\leq 4 \vee x_4 - x_3 \leq -2 \\ x_2 - x_4 &\leq 2 \vee x_3 - x_2 \leq 1 \\ &\vdots \end{aligned}$$

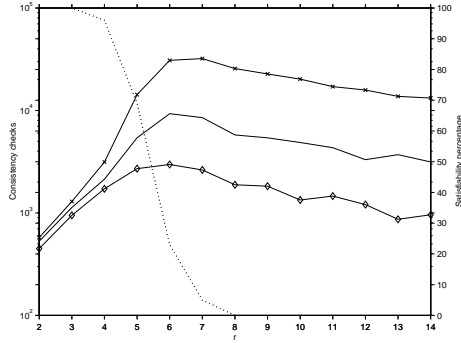
(in the following, for any DTP ψ , $\psi(i)$ denotes the i -th disjunction displayed, and $\psi(i, j)$ the j -th disjunct of the i -th disjunction displayed in ψ . Thus, for example, $\varphi(1)$ is $x_1 - x_2 \leq 3$ and $\varphi(2, 2)$ is $x_4 - x_3 \leq -2$) where the dots stand for further (possibly many) unspecified clauses such that no T-consistent extension of the evaluation $\{\varphi(1, 1), \varphi(2, 1)\}$ P-satisfying the whole DTP exists. Let us consider the behavior of SK and TSAT (TSAT^{FC}) when $\{\varphi(1, 1), \varphi(2, 1)\}$ is the evaluation built so far. Since no T-consistent extension of $\{\varphi(1, 1), \varphi(2, 1)\}$ exists, after some search, failure is necessarily detected and both procedures backtrack and remove $\varphi(2, 1)$ from the current valuation.

The main difference between SK and TSAT (TSAT^{FC}) is that SK goes on with the valuation $\{\varphi(1, 1), \varphi(2, 2)\}$, whereas TSAT (TSAT^{FC}) proceeds with the valuation $\{\varphi(1, 1), \neg\varphi(2, 1)\}$ which is immediately extended (via simplification and unit propagation) to $\{\varphi(1, 1), \neg\varphi(2, 1), \varphi(2, 2)\}$. As we are going to see, working with $\{\varphi(1, 1), \neg\varphi(2, 1), \varphi(2, 2)\}$ in place of $\{\varphi(1, 1), \varphi(2, 2)\}$ may lead to considerable savings. Let us now assume that both procedures extend their own current valuation with $\varphi(3, 1)$. Since $\{\varphi(1, 1), \neg\varphi(2, 1), \varphi(2, 2), \varphi(3, 1)\}$ is T-inconsistent, TSAT (TSAT^{FC}) stops immediately and goes on considering the next disjunct, i.e. $\varphi(3, 2)$. On the other hand SK may waste a big amount of resources in the vain attempt of finding a T-consistent extension of $\{\varphi(1, 1), \varphi(2, 2), \varphi(3, 1)\}$ P-satisfying φ .

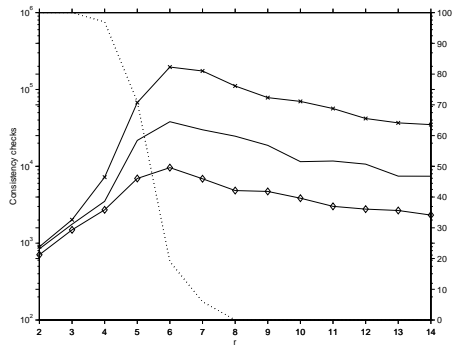
The key observation is that $\{\varphi(1, 1), \varphi(2, 2), \varphi(3, 1)\}$ entails $x_1 - x_3 \leq 3$ and therefore implies $\varphi(2, 1)$. Thus,



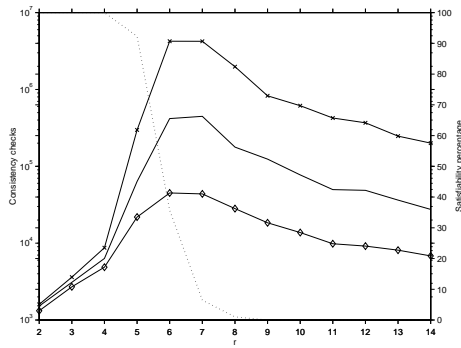
(a) Results for $n=10$.



(b) Results for $n=12$.



(c) Results for $n=15$.



(d) Results for $n=20$.

Legenda:

- Satisfiability percentage
- x— SK
- TSAT^{FC}
- ◇— TSAT^{FC}_{IS(2)}

Fig. 3. SK, TSAT^{FC} and TSAT^{FC}_{IS(2)} median number of consistency checks and percentage of T-satisfiable problems as a function of the ratio of clauses to variables ($r = m/n$).

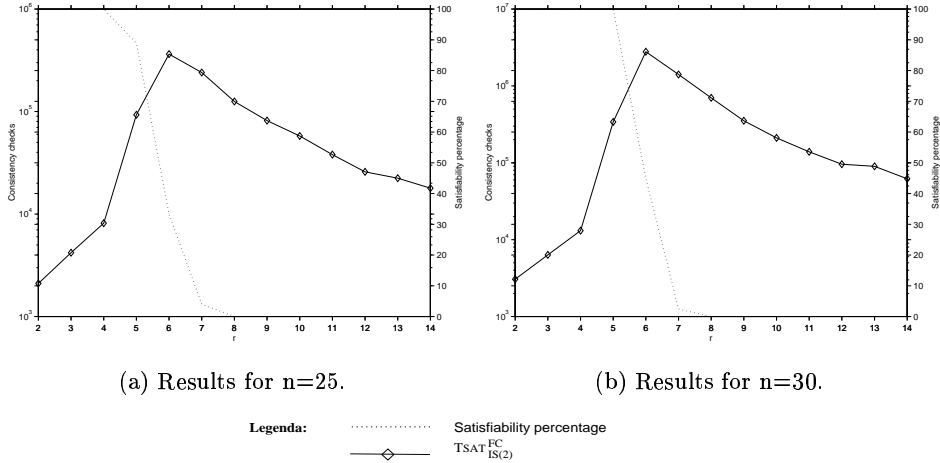


Fig. 4. TSAT^{FC}_{IS(2)} median number of consistency checks and percentage of T-satisfiable problems as a function of the ratio of clauses to variables ($r = m/n$).

- while SK has to (re-)perform some computation in order to determine the T-inconsistency of any valuation (*i*) extending $\{\varphi(1, 1), \varphi(2, 2), \varphi(3, 1)\}$, and (*ii*) P-satisfying the whole DTP,
- TSAT^{FC} does not perform such computation since it considers the valuation $\{\varphi(1, 1), \neg\varphi(2, 1), \varphi(2, 2), \varphi(3, 1)\}$.

At a more abstract level, the fundamental difference between the two search procedures lies in the different form of branching they implement. While SK uses “syntactic branching”:

$$\frac{c_1 \vee c_2}{c_1 \quad c_2}$$

(given a disjunction $(c_1 \vee c_2)$ it first considers the case in which c_1 is true and then the case in which c_2 is true), TSAT (and thus TSAT^{FC} and TSAT^{FC}_{IS(2)}) performs “semantic branching” [Freeman, 1995]:

$$\frac{\quad}{c_1 \quad \neg c_1}$$

i.e., it selects a not yet evaluated constraint c_1 , and first considers the case in which it is true and then the case in which it is false. Notice that in the second case, the conjunction of $\neg c_1$ with $(c_1 \vee c_2)$ yields c_2 as a unit. As already observed in [D’Agostino, 1992, Giunchiglia and Sebastiani, 1996], syntactic branching may lead to redundant exploration of parts of the search space. That this is indeed the case also in the context of temporal reasoning is shown by the above example.

4.2 $\text{Tsatsat}^{\text{FC}}$ vs. $\text{Tsatsat}_{\text{IS}(2)}^{\text{FC}}$

To illustrate the advantages of using the IS(2) heuristics, consider the following disjunctive temporal problem φ :

$$\begin{aligned} x_3 - x_2 \leq 6 \vee x_1 - x_4 \leq 4 \\ x_3 - x_1 \leq 1 \vee x_4 - x_1 \leq -5 \\ x_1 - x_3 \leq -2 \vee x_2 - x_3 \leq -7 \end{aligned} \tag{3}$$

By checking the T-consistency of the 3 pairs of temporal constraints involving the same variables with opposite signs, we discover that $\{x_3 - x_2 \leq 6, x_2 - x_3 \leq -7\}$, $\{x_1 - x_4 \leq 4, x_4 - x_1 \leq -5\}$, and $\{x_3 - x_1 \leq 1, x_1 - x_3 \leq -2\}$ are T-inconsistent. Therefore the IS(2) heuristics adds the following clauses to the initial DTP:

$$\begin{aligned} \neg(x_3 - x_2 \leq 6) \vee \neg(x_2 - x_3 \leq -7) \\ \neg(x_1 - x_4 \leq 4) \vee \neg(x_4 - x_1 \leq -5) \\ \neg(x_3 - x_1 \leq 1) \vee \neg(x_1 - x_3 \leq -2) \end{aligned}$$

Three are the advantages of considering the extended DTP:

- First, if two mutually inconsistent temporal constraints occur in the input DTP, TSAT^{FC} can check and discover such mutual inconsistency over and over again in different part of the search tree. By adding the clauses encoding such mutual exclusion, $\text{TSAT}_{\text{IS}(2)}^{\text{FC}}$ performs such checks only once: adding one of the temporal constraints to the current assignment causes the automatic addition of the negation of the other without performing any other consistency check.
- Second, IS(2) adds binary clauses possibly leading to extensive applications of the unit rule. In the case of the example, it is easy to check that after the first initial splitting step (assigning a truth value to a literal in the extended DTP), all the other literals get assigned by the unit step.
- Third, in (almost) all the DTPs that we randomly generate and use as our tests, there are no two occurrences of a same temporal constraint. As a consequence the heuristic implemented by Böhm’s (and inherited by TSAT^{FC} and $\text{TSAT}_{\text{IS}(2)}^{\text{FC}}$) provides no guidance in the choice of the best literal to use during the splitting step. We recall that Böhm’s procedure chooses one among the literals which have the greatest number of occurrences in the clauses of minimal length [Buro and Buning, 1992] (in SAT terminology, Böhm’s heuristic is a MOMS [Freeman, 1995]). Adding the clauses introduced by IS(2) has the positive effect that $\text{TSAT}_{\text{IS}(2)}^{\text{FC}}$ will choose the literal with the greatest number of occurrences in the newly added clauses, and hence possibly leading to the greatest number of applications of the unit rule.

Notice however that IS(2) is indeed a “greedy” heuristic: it performs some computation without any information about its usefulness. For sure, there are DTPs in which IS(2) may lead to a computationally worse behavior.

5 Conclusions

In this paper we have proposed a SAT-based approach to disjunctive temporal reasoning. We have presented two decision procedures (TSAT^{FC} and $\text{TSAT}_{\text{IS}(2)}^{\text{FC}}$), the second differing from the first in that it uses IS(2), a simple preprocessing routine which greatly improves performances. Our procedures, when applied to sets of randomly generated binary disjunctive temporal constraints, perform less consistency checks than the fastest of the procedures proposed by Stergiou and Koubarakis. We have explained the different computational behaviors of SK and TSAT^{FC} as due to the different types of propositional search performed. In fact, while SK performs syntactic branching, TSAT^{FC} performs semantic branching. We have argued the superiority of semantic branching w.r.t. syntactic branching, using a paradigmatic example as a case study. As for the IS(2) heuristics, we have shown that it is easy to perform, and that it greatly improves performances of TSAT^{FC} .

Our experimental analysis confirms the existence of an easy-hard-easy pattern of the number of consistency checks, if plotted against the ratio of clauses to variables. We agree with Stergiou and Koubarakis's conjecture that the peak is obtained for a value of r in between 6 and 7. Furthermore, our tests show that the 50% of T-satisfiable DTPs is obtained when $5 \leq r \leq 6$. Also this fact agrees with the experiments presented in [Stergiou and Koubarakis, 1998].

In the future, we plan to implement other forms of pruning strategy, like those described in [Prosser, 1993]. An extensive comparative analysis will reveal how these strategies behave and interact in a SAT-based framework.

Moreover, we plan to extend the SAT-based approach to the class of Disjunctive Linear Relation Sets [Jonsson and Bäckström, 1996] which subsumes the DTP framework, but also many other temporal formalisms.

The implementation of the procedures described in this paper can be obtained on request from the second author. The complete test sets we have used to evaluate the procedures (including those presented in this paper) are publicly available on the Internet at the URL: <http://www.mrg.dist.unige.it/~drwho/Tsat>.

Acknowledgments

We are grateful to Silvio Ranise, Roberto Sebastiani and Armando Tacchella for useful discussions related to the subject of this paper. Thanks to Kostas Stergiou for making the SK procedure available to us.

The last author has been supported by the Italian Space Agency under the project "Un sistema Intelligente per la supervisione di robot autonomi nello spazio".

References

- [Allen *et al.*, 1991] J. Allen, H. Kautz, and R. Pelavin (Eds.). *Reasoning about Plans*. Morgan Kaufmann, 1991.

- [Brusoni *et al.*, 1996] V. Brusoni, L. Console, B. Pernici, and P. Terenziani. LaTeR: An Efficient, General Purpose Manager of Temporal Information. *IEEE Software*, 1996.
- [Buro and Buning, 1992] M. Buro and H. Buning. Report on a SAT competition. Technical Report 110, University of Paderborn, Germany, November 1992.
- [Cheng and Smith, 1994] Cheng-Chung Cheng and Stephen F. Smith. Generating feasible schedules under complex metric constraints. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, volume 2, pages 1086–1091, Seattle, Washington, USA, August 1994. AAAI Press/MIT Press.
- [Chleq, 1995] N. Chleq. Efficient algorithms for networks of quantitative temporal constraints. In *Proceedings of CONSTRAINTS95*, pages 40–45, April 1995.
- [D’Agostino, 1992] M. D’Agostino. Are Tableaux an Improvement on Truth-Tables? *Journal of Logic, Language and Information*, 1:235–252, 1992.
- [Davis and Putnam, 1960] M. Davis and H. Putnam. A computing procedure for quantification theory. *Journal of the ACM*, 7:201–215, 1960.
- [Dechter *et al.*, 1991] R. Dechter, I. Meiri, and J. Pearl. Temporal constraint networks. *Artificial Intelligence*, 49(1-3):61–95, January 1991.
- [Freeman, 1995] Jon W. Freeman. *Improvements to propositional satisfiability search algorithms*. PhD thesis, University of Pennsylvania, 1995.
- [Giunchiglia and Sebastiani, 1996] F. Giunchiglia and R. Sebastiani. Building decision procedures for modal logics from propositional decision procedures - the case study of modal K. In *Proc. CADE-96*, Lecture Notes in Artificial Intelligence, New Brunswick, NJ, USA, August 1996. Springer Verlag.
- [Giunchiglia *et al.*, 2000] E. Giunchiglia, F. Giunchiglia, and A. Tacchella. SAT-Based Decision Procedures for Classical Modal Logics, 2000. Accepted for publication in *Journal of Automated Reasoning*. Available at <http://www.mrg.dist.unige.it/~enrico>.
- [Haralick and Elliott, 1980] R. M. Haralick and G. L. Elliott. Increasing Tree Search Efficiency for Constraint Satisfaction Problems. *Acta Informatica*, 14:263–313, 1980.
- [Jonsson and Bäckström, 1996] Peter Jonsson and Christer Bäckström. A linear-programming approach to temporal reasoning. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence and the Eighth Innovative Applications of Artificial Intelligence Conference*, pages 1235–1241, Menlo Park, August 4–8 1996. AAAI Press / MIT Press.
- [Koubarakis, 1997] Manolis Koubarakis. The complexity of query evaluation in indefinite temporal constraint databases. *Theoretical Computer Science*, 171(1–2):25–60, 15 January 1997.
- [Prosser, 1993] Patrick Prosser. Hybrid algorithms for the constraint satisfaction problem. *Computational Intelligence*, 9(3):268–299, 1993.
- [Smullyan, 1968] R. M. Smullyan. *First-Order Logic*. Springer-Verlag, NY, 1968.
- [Stergiou and Koubarakis, 1998] Kostas Stergiou and Manolis Koubarakis. Backtracking algorithms for disjunctions of temporal constraints. In *Proc. AAAI*, 1998.