Efficient DTPP solving with a reduction-based approach [1]

Marco Maratea [??,*], and Luca Pulina [??]
[a] *DIBRIS - University of Genova, Viale F. Causa 15, Genova, Italy. Email: marco@dibris.unige.it*
[b] *POLCOMING - University of Sassari, Viale Mancini 5, 07100 Sassari, Italy. Email: lpulina@uniss.it*

Abstract.
Disjunctive Temporal Problems with Preferences (DTPPs) extend DTPs with piece-wise constant preference functions associated to each constraint of the form $l \leq x - y \leq u$, where x, y are (real or integer) variables, and l, u are numeric constants. The goal is to find an assignment to the variables of the problem that maximizes the sum of the preference values of satisfied DTP constraints, where such values are obtained by aggregating the preference functions of the satisfied constraints in it under a "max" semantic. The state-of-the-art approach in the field, implemented in the DTPP solver MAXILITIS, extends the approach of the DTP solver EPILITIS.

In this paper we present an alternative approach that reduces DTPPs to Maximum Satisfiability of a set of Boolean combination of constraints of the form $l \bowtie x - y \bowtie u$, $\bowtie \in \{<, \leq\}$, that extends previous work that dealt with constant preference functions only. Results obtained with the Satisfiability Modulo Theories (SMT) solver YICES on randomly generated DTPPs show that our approach is competitive to, and can be faster than, MAXILITIS.

Keywords: Disjunctive Temporal Problems, Optimization

1. Introduction

Temporal constraint networks [?] provide a convenient formal framework for representing and processing temporal knowledge. Along the years, a number of extensions to the framework have been presented to deal with, e.g. more expressive preferences. Disjunctive Temporal Problems with Preferences (DTPPs) is one of such extensions, that is useful in planning and scheduling domains. DTPPs extend DTPs, i.e. conjunctions of disjunctions of constraints of the form $l \leq x - y \leq u$, where x, y are (real or integer) variables, and l, u are numeric constants, with piece-wise constant preference functions associated to each constraint. The goal is to find an assignment to the variables of the problem that maximizes the sum of the preference values of satisfied disjunctions of constraints (called DTP constraints), where such values are obtained by aggregating the preference functions of the satisfied constraints in it. We consider an utilitarian aggregation of such DTP constraints values, and a "max" semantic for aggregating preference values within DTP constraints: given a (candidate) solution of a DTPP, the preference value of a DTP constraint is defined to be the maximum value achieved by any of its satisfied disjuncts (see, e.g. [?]). The current state-of-the-art approach that considers such aggregation methods is implemented in the DTPP solver MAXILITIS, and is based on an extension of the approach of the DTP solver EPILITIS [?] to deal with piece-wise constant preference functions. Various other approaches have been designed in the literature to deal with DTPPs [?,?,?,?], possibly relying on alternative preference aggregation methods (see, e.g. [?,?]).

[1]This is an extended and revised version of a paper presented to the 5th Italian Workshop on Planning and Scheduling, which appears in [?].

[*]Corresponding author. E-mail: marco@dibris.unige.it.

In this paper we present an alternative approach that reduces DTPPs to Maximum Satisfiability of a set of Boolean combination of constraints of the form $l \bowtie x - y \bowtie u$, where $\bowtie \in \{<, \leq\}$. We have first considered a very natural modeling of the problem where, starting from each DTP constraint, the generated constraints are mutually exclusive, and each is weighted by a preference value in order to maximize the degree of satisfaction of the DTP constraint. A second solution we propose is, instead, obtained by extending previous work that dealt with constant preference functions only [**?**], and reduces each DTP constraint to a set of disjunctions of constraints, and a non-trivial interplay among their preference values to maximize, as before, the preference value that get assigned to the DTP constraint. In order to test the effectiveness of our proposals, we have randomly generated DTPPs, following the method originally developed in [**?**] and then employed in all other papers on DTPPs. In our framework, each problem is then represented as a Satisfiability Modulo Theory (SMT) formula, and the YICES SMT solver [**?**], that is able to deal with optimization issue, is employed[1]. An experimental analysis conducted on a wide set of benchmarks, using also the same benchmarks setting already employed in past papers, shows that (*i*) the first solution is impractical, and (*ii*) the second approach is competitive to, and can be faster than, MAXILITIS.

To summarize the main contributions of the paper:

- We defined two approaches for solving DTPPs, by means of reduction to a Maximum Satisfiability problem of a set of Boolean combination of constraints of specific form.
- We implemented both reductions.
- We evaluated such reductions employing the SMT solver YICES w.r.t. the state-of-the-art DTPP solver MAXILITIS on random DTPPs.

The rest of the paper is structured as follows. Section **??** introduces preliminaries about DTPs, DTPPs and Maximum Satisfiability. Then, in Section **??** we present our reductions from DTPPs to Maximum Satisfiability of Boolean combination of constraints, while the experimental analysis is presented in Section **??**. The paper ends by providing a discussion about the related work in Section **??** and some conclusions in Section **??**.

2. Formal Background

Problems involving disjunctions of temporal constraints have been introduced in [**?**], as an extension of the Simple Temporal Problem (STP) [**?**], which consists of conjunction of constraints. The problem was referred for the first time as Disjunctive Temporal Problem (DTP) in [**?**], and is presented in the first subsection. The remaining subsections introduce Maximum Satisfiability of DTPs and DTPPs.

2.1. DTP

Let $\mathcal{V}$ be a set of symbols, called *variables*. A *constraint* is an expression of the form $l \bowtie x - y \bowtie u$, where $\bowtie \in \{<, \leq\}$, $x, y \in \mathcal{V}$, and l, u are numeric constants. A *DTP constraint* is a disjunction of constraints having $\bowtie = \leq$ (equivalently seen as a disjunctively intended finite set of constraints). A *DTP formula*, or simply *formula*, is a finite conjunction of DTP constraints. A DTP constraint can be either *hard*, i.e. its satisfaction is mandatory, or *soft*, i.e. its satisfaction is not necessary but preferred, and in case of satisfaction it contributes to the generation of high quality solutions according to the aggregation methods employed and defined later. A *DTP^A constraint* is a Boolean combination of constraints.

About the semantics, let the set $\mathcal{D}$ (*domain of interpretation*) be either the set of the real numbers $\mathcal{R}$ or the set of integers $\mathcal{Z}$. An *assignment* is a total function mapping variables to $\mathcal{D}$. Let σ be an assignment and ϕ be a formula composed by hard DTP constraints only. Then, $\sigma \models \phi$ (σ satisfies *a formula* ϕ) is defined as follows

- $\sigma \models l \leq x - y \leq u$ if and only if $l \leq \sigma(x) - \sigma(y) \leq u$;
- $\sigma \models \neg\phi$ if and only if it is not the case that $\sigma \models \phi$;
- $\sigma \models (\wedge_{i=1}^{n} \phi_i)$ if and only if for each $i \in [1, n]$, $\sigma \models \phi_i$; and

[1]YICES showed the best performance in [?] among a number of alternatives, and it is the only SMT solver able to cope with (Partial Weighted) Maximum Satisfiability problems.

– $\sigma \models (\vee_{i=1}^{n} \phi_i)$ if and only if for some $i \in [1, n]$, $\sigma \models \phi_i$.

If $\sigma \models \phi$ then σ is also called a *model* of ϕ. We also say that a formula ϕ is *satisfiable* if and only if there exists a model for ϕ. The DTP is the problem of deciding whether a formula ϕ is satisfiable or not in the given domain of interpretation $\mathcal{D}$. Notice that the satisfiability of a formula depends on $\mathcal{D}$, e.g. the formula

$$x - y > 0 \wedge x - y < 1$$

is satisfiable if $\mathcal{D}$ is $\mathcal{R}$ but unsatisfiable if $\mathcal{D}$ is $\mathcal{Z}$. However, the problems of checking satisfiability in $\mathcal{Z}$ and $\mathcal{R}$ are closely related and will be treated uniformly.

Example 1. Consider a DTP formula ϕ_1 composed by only one constraint $dc_1 \vee dc_2$, where $dc_1 : 1 \leq x - y \leq 10$ and $dc_2 : 5 \leq z - q \leq 15$; $\mathcal{D}$=$\mathcal{Z}$.

ϕ_1 is satisfiable and a model σ_1 for it assigns $x = 8$, $y = 2$, $z = 10$, and $q = 0$.

2.2. Max-DTP

Consider now a DTP^A formula ϕ consisting of hard DTP constraints and soft DTP^A constraints. Intuitively, in this case the goal is to find an assignment to the variables in ϕ that satisfies all hard DTP constraints and maximizes the sum of the weights associated to satisfied soft DTP^A constraints. The problem is called Partial Weighted Maximum Satisfiability of DTP^A, and is formally defined as a pair $\langle \phi, w \rangle$, where

1. ϕ is a DTP^A formula consisting of both hard DTP and soft DTP^A constraints, and
2. w is a function that maps DTP^A constraints to positive integer numbers.

More precisely, the goal is to find an assignment σ' for ϕ that satisfies all hard DTP constraints and maximizes the following linear objective function f

$$f = \sum_{d \in \phi, \sigma' \models d} w(d) \tag{1}$$

where d is a soft DTP^A constraint. In the following, for simplicity, we will use Max-DTP to refer to the Partial Weighted Maximum Satisfiability problem of mixed DTP and DTP^A constraints as defined above.

Example 2. Consider a soft DTP^A formula $\phi_2 := d_1 \wedge d_2 \wedge d_3$, where d_1 is the formula ϕ_1 in Example 1, $d_2 := -1 \leq x - y \leq 0$ and $d_3 := 1 \leq z - q \leq 4$; moreover, $w(d_1) = 3$, $w(d_2) = 2$ and $w(d_3) = 1$.

An assignment σ_2 that maximizes the objective function (**??**) assigns $x = 0$, $y = 1$, $z = 10$, $q = 0$, and the related value for f is 5.

2.3. DTPP

DTPP is an extension of DTP, and it is defined as a pair $\langle \phi, w' \rangle$, where

1. ϕ is a DTP formula consisting of both hard and soft DTP constraints, and
2. w' is a function that maps constraints in soft DTP constraints to piece-wise constant preference functions.

We consider, as before, an utilitarian method for aggregating soft DTP constraints weights: the goal is now to find an assignment σ' for ϕ that (i) satisfies all hard DTP constraints, and (ii) maximizes the sum of weights associated to satisfied soft DTP constraints, i.e. maximizes a linear objective function as (**??**).

It is left to define how weights, corresponding to preference values, are aggregated within soft DTP constraints to "define" their weights $w(d)$ in (**??**). In our work we consider a prominent semantic for this purpose: the *max* semantic.

Given a constraint $dc := l \leq x - y \leq u$, its preference function $w'(dc)$ is in general defined as:

$$w'(dc) : t \subseteq [l, u] \rightarrow [0, R^+]$$

mapping every feasible temporal interval t to a preference value expressing its weight. The *max* semantic [**?**,**?**] defines the weight $w(d)$ of a satisfied soft DTP constraint d as the maximum among the possible preference values of satisfied constraints in d, i.e. given an assignment σ'

$$w(d) := max\{w'(t_{dc}) : dc \in d, \sigma' \models dc\}$$

where t_{dc} is the interval $[\sigma'(x), \sigma'(y)]$.

Example 3. Consider the DTP formula in Example 1 to be soft instead of hard, and the following piece-wise constant preference functions:

$$f(dc_1) = \begin{cases} 1 & 1 \leq x - y \leq 3 \\ 2 & 3 < x - y \leq 7 \\ 1 & 7 < x - y \leq 10 \end{cases} \tag{2}$$

and

$$f(dc_2) = \begin{cases} 2 & 5 \leq z - q \leq 8 \\ 4 & 8 < z - q \leq 10 \\ 2 & 10 < z - q \leq 15 \end{cases} \tag{3}$$

Of course, in this case the maximum possible value (4) can be assigned to the DTP constraint d, e.g. with the assignment σ_1 of Example 1.

3. Reducing DTPPs to Max-DTPs

As we said before, our main idea is to reduce the problem of solving DTPPs to solving Max-DTPs. Hard DTP constraints remain unchanged in our reduction, while soft DTP constraints need special treatment. Given a soft DTP constraint d, for each constraint $dc := l \leq x - y \leq u$ in d, let L_{dc} be a set of pairs, each pair $\langle DC, v \rangle$ being composed by (i) a set DC of pairs $(\overline{l}, \overline{u})$, representing the end points of intervals, such that $[\overline{l}, \overline{u}] \subseteq [l, u]$, and (ii) the preference value v of the constraints of the type $\overline{l} \bowtie x - y \bowtie \overline{u}$, $\bowtie \in \{\leq, <\}$ from DC, where the variables x, y are obtained from the constraint name. If the preference function is a constant v', L_{dc} is composed by only one pair $\langle \{(l, u)\}, v' \rangle$, i.e. the interval $[l, u]$, representing the constraint $l \leq x - y \leq u$, and its preference value v'.

We need now to "aggregate" the preference values corresponding to different levels of the piece-wise constant functions in the various constraints in order to implement our reduction. The idea is to "merge" the pairs $\langle DC, v \rangle$, representing preference functions of constraints, in the same soft DTP constraint; intuitively, this means that, if the candidate solution satisfies at least one of the constraints obtained from DC at preference value v, then a possible preference value for d is v.

More formally, consider aggregating L_{dc_1} and L_{dc_2}, coming from two constraints dc_1 and dc_2 in d, respectively.

$L_{dc_1 \vee dc_2}$:=MERGE(L_{dc_1}, L_{dc_2}) is an operator that

- contains the preference values that are in the preference functions of dc_1 or dc_2; and
- if the preference functions of dc_1 and dc_2 have a common preference value, i.e. L_{dc_1} contains a pair $\langle DC_i, v_i \rangle$, L_{dc_2} contains a pair $\langle DC_j, v_j \rangle$ and $v_i = v_j$, these pairs are merged and $L_{dc_1 \vee dc_2}$ contains a pair $\langle DC_i \cup DC_j, v_i \rangle$.

Moreover, during MERGE a subscript is attached to the pairs $(\overline{l}, \overline{u})$, from which we deduce the ordered pair of variables involved in the constraint it represents.

The operator MERGE can be easily generalized to an arbitrary finite number of constraints.

Consider a soft DTP constraint

$$d := dc_1 \vee ... \vee dc_k. \tag{4}$$

Further, consider an ordering on the k pairs in L_d of a dc in d induced by the preference values, i.e. an ordering $\prec$ is which $\langle DC_i, v_i \rangle \prec \langle DC_j, v_j \rangle$ iff $v_i < v_j, 1 \leq i, j \leq k, i \neq j$. For simplicity, from now on we consider the pairs in L_d to be re-ordered according to $\prec$, i.e. DC_1 is the set whose v_1 is maximum among the weights in d, i.e. $v_1 > v_i, 2 \leq i \leq k$, while the set DC_k is such that $v_k < v_i, 1 \leq i \leq k-1$.

In the following subsections, we present the two reductions. The first reduction corresponds to a very natural representation of soft DTP constraints, where preference values are directly applied to conjunctions of (possibly negated) constraints. In the second reduction, instead, the resulting formula has a Conjunctive Normal Form (CNF) structure, but preference values need to be aggregated.

3.1. First reduction

The first attempt we considered for our reduction is to express a soft DTP constraint d using soft DTP^A constraints that force the highest preference value associated to satisfied constraints in d to be assigned as weight for d, in a direct way. First, we apply the operator MERGE to all the constraints in d, and related piece-wise constant preference functions, i.e. L_d :=MERGE$(L_{dc_1}, \ldots, L_{dc_k})$.

Then, starting from L_d, d and its preference value are expressed by the following $|L_d|$ soft DTP^A constraints: for each $z = 1 \dots |L_d|$

$$c_z := \wedge_{i=1}^{z-1} \neg(\vee_{p \in DC_i} dc_p) \wedge (\vee_{p \in DC_z} dc_p) \quad (5)$$

$$w(d) = w(c_z) = v_z$$

where dc_p is a constraint built from the pair p (we recall that the subscript identifies the variables involved in the constraint, and in which order they appear). The set of constraints is mutually exclusive: considering an assignment, at most one of the constraints in (**??**) can be satisfied, and the relative value is assigned to d. If a constraint in (**??**) is satisfied, this is the constraint leading to the maximum achievable value (according to the candidate solution considered).

This is done for each soft DTP constraint in the formula.

Example 4. Consider the DTPP formula in Example 3. We report again the preference functions for convenience, and then the related representations.

The piece-wise constant preference function associated to dc_1 is

$$f(dc_1) = \begin{cases} 1 & 1 \leq x - y \leq 3 \\ 2 & 3 < x - y \leq 7 \\ 1 & 7 < x - y \leq 10 \end{cases} \quad (6)$$

and can be represented with

$$L_{dc_1} = \{\langle\{(1,3),(7,10)\},1\rangle, \langle\{(3,7)\},2\rangle\}.$$

Regarding dc_2, its preference function is

$$f(dc_2) = \begin{cases} 2 & 5 \leq z - q \leq 8 \\ 4 & 8 < z - q \leq 10 \\ 2 & 10 < z - q \leq 15 \end{cases} \quad (7)$$

represented with

$$L_{dc_2} = \{\langle\{(5,8),(10,15)\},2\rangle, \langle\{(8,10)\},4\rangle\}.$$

We now "merge" L_{dc_1} and L_{dc_2} into

$$L_{dc_1 \vee dc_2} := merge(L_{dc_1}, L_{dc_2})$$

whose result is

$$\begin{aligned} &\{\ \langle\{(1,3)_1,(7,10)_1\},1\rangle, \\ &\langle\{(3,7)_1,(5,8)_2,(10,15)_2\},2\rangle, \\ &\langle\{(8,10)_2\},4\rangle\ \}. \end{aligned} \quad (8)$$

Following (**??**), the reduction is in Figure **??**.

Further note that the preference functions we have considered are characterized by having the left-most sub-interval with both bounds included, while the remaining sub-intervals have only the right bound included: to correctly reproduce the reduction from the set L, we have further assumed that with the subscript we can recognize the left-most sub-interval of each constraint.

This first reduction corresponds to a very natural way of expressing soft DTP constraints; unfortunately, experiments show that it is inefficient (see Section **??** for details).

3.2. Second reduction

A second reduction transforms each soft DTP constraint d to $|L_d|$ soft DTP^A constraints as follows: for each $z = 1 \dots |L_d|$

$$c'_z := \vee_{i=1}^{z} \vee_{p \in DC_i} dc_p \quad (9)$$

The problem is now to define what are the weights associated to each newly defined soft DTP^A constraint, in order to reflect the semantic of our problem. In the previous reduction (**??**), the constraints occurred positively only once; now there can be many occurrences in the corresponding soft DTP^A constraints in (**??**) that influence constraints weights adaptation and definition. Our solution starts from the following fact: if the constraint $c'_{|L_d|}$ is satisfied, it contributes for at least the minimum preference value $v_{|L_d|}$, i.e. the one associated to the set $DC_{|L_d|}$, from which $c'_{|L_d|}$ is constructed. Satisfying the constraint $c'_{|L_d|-1}$ contributes for $v_{|L_d|-1} - v_{|L_d|}$, and given that a constraint c'_z implies all constraints $c'_{z'}$, $z' > z$, these two soft DTP^A constraints together contribute for $v_{|L_d|-1}$. This method is recursively applied up to the set of constraints constructable from DC_1, i.e. c'_1, whose preference value is $v_1 - v_2$ and, given that c'_1 implies all other introduced soft DTP^A

$$c_1 : (8 < z - q \leq 10),\ w(c_1) = 4$$

$$c_2 : \neg c_1 \wedge ((3 < x - y \leq 7) \vee (5 \leq z - q \leq 8) \vee (10 < z - q \leq 15)),\ w(c_2) = 2$$

$$c_3 : \neg c_1 \wedge \neg c_2 \wedge (1 \leq x - y \leq 3 \vee 7 < x - y \leq 10),\ w(c_3) = 1$$

Fig. 1. First reduction applied to Example 4.

constraints, satisfying c'_1 correctly corresponds to assign a weight v_1 to d.

More formally, for each $z = 1 \ldots |L_d|$

$$w(c'_z) = \begin{cases} v_{|L_d|} & z = |L_d| \\ v_z - v_{z+1} & 1 \leq z < |L_d| \end{cases} \tag{10}$$

and, given an assignment σ',

$$w(d) = \sum_{z \in \{1,\ldots,|L_d|\}, \sigma' \models c'_z} v_z$$

.

The soft DTP^A constraints that express Example 4 are in Figure **??**.

Such reduction works correctly if we consider a single soft DTP constraint. However, considering a formula ϕ, given our reduction, it is possible to have repeated DTP^A constraints in the reduced formula ϕ'. In this case, intuitively, we want each single occurrence in ϕ' to count "separately", given that they take into account different contributions from different soft DTP constraints in ϕ. A solution is to consider a *single occurrence* of the resulting soft DTP^A constraint in ϕ' whose weight is the sum of the weights of the various occurrences. The same applies to the first reduction.

4. Experimental Analysis

We have implemented both reductions, and expressed the resulting formulas as SMT formulas with optimization, then solved with YICES ver. 1.0.38. The experimental analysis will be divided into two parts: the first part compares the two reductions (showing that the first reduction is not competitive), while the second part aims at comparing the performance of our second reduction, called DTPPYICES, with two versions of the MAXILITIS solver, namely MAXILITIS-IW and MAXILITIS-BB. MAXILITIS-IW (IW standing for Iterative Weakening) searches for solutions with a progressively increasing number of violated constraints; MAXILITIS-BB uses a branch-and-bound approach for reaching the optimal solution.

Our experiments aim at comparing the considered solvers on three dimensions, namely (*i*) the size of the benchmarks; (*ii*) the number of preference levels in the piece-wise constant preference functions; and (*iii*) the constraint density, i.e. the ratio of constraints to arithmetic variables, all used in past papers on DTPPs.

For randomly generating the benchmarks the main parameters considered are:

1. the number k of disjuncts per DTP constraint;
2. the number n of arithmetic variables;
3. the number m of DTP constraints;
4. the number l of levels in the preference functions.

Furthermore, we also investigated the performance of the solvers considering two different settings related to the generation of the preference values in the piece-wise preference functions. The preference functions considered are semi-convex piece-wise constant:[2] starting from the lower and upper bounds of the constraints, intervals corresponding to higher preference levels are randomly put within the interval of the immediate lower level, with a reduction factor, up to an highest level. For details see, e.g. [**?**].

In particular, we consider

[2]These are the type of preference functions resulting from the generation method employed in the literature to evaluate DTPP solvers.

$$c'_1 := 8 < z - q \leq 10,\ w(c'_1) = 2$$

$$c'_2 := c'_1 \vee (3 < x - y \leq 7 \vee 5 \leq z - q \leq 8 \vee 10 < z - q \leq 15),\ w(c'_2) = 1$$

$$c'_3 := c'_1 \vee c'_2 \vee (1 \leq x - y \leq 3 \vee 7 < x - y \leq 10),\ w(c'_3) = 1$$

Fig. 2. Second reduction applied to Example 4.

- a setting where, given the i-th level l, $w(l) = i$, i.e. the setting used in [?] ("Model A" in the following);
- a setting where $w(l)$ is a positive integer number randomly generated in the range [1, 100], still ensuring to maintain the same shape for preference functions ("Model B" in the following).

In Figure **??** we report an example related to a constraint, with possible preference values generated by the two models; Figure **??** depicts their graphical versions.

The experiments reported in the following ran on PCs equipped with a processor Intel Core i5 at 3.20 GHz, with 4 GB of RAM, and running GNU Linux Ubuntu 12.04. The timeout for each instance has been set to 300s.

For each tuple of values of the parameters, 25 instances have been generated, and the median CPU time is showed (unsolved instances are counted 300s).

As a first experiment, we randomly generated benchmarks[3] by varying the total amount of constraints, with the following parameters: k=2, $m \in \{10, \ldots, 100\}$, n=0.8 $\times$ m, l=5, lower and upper bounds of each constraint taken in $[-50, 100]$.

The next experiment aims to evaluate the solvers by varying the number of levels in the preference functions, with the following parameters: k=2, n=24, m=30, $l \in \{2, \ldots, 8\}$, lower and upper bounds of each constraint taken in $[-50, 100]$.

The last experiment aims to evaluate the solvers with respect to the constraint density m/n. In order to do that, we generated benchmarks having k=2, n=30, l=5 and ratio m/n in {3, 6, 12, 15, 20, 30, 36}. Also in this case, we set lower and upper bounds of each constraint in the range $[-50, 100]$.

[3]We have used and extended the program provided by Michael D. Moffitt, author of MAXILITIS.

These experiments correspond to the dimensions reported above, and for each of such setting the preference values of piece-wise constant preference functions are generated with model A and model B (for a total of six groups of instances).

4.1. Comparing the two reductions

As we mentioned in the Introduction, the first reduction is impractical from a performance point of view. Considering the first experiment, we report that the implementation of the first reduction was able to solve only benchmarks having m = 10. In the case of model A, we report that it was able to solve all the 25 instances in 74.22 CPU seconds, while considering model B, it tops at 18 solved benchmarks in 2162.67 seconds. As we will show in detail in the next section, DTPPYICES is orders of magnitude faster.

Considering the second experiment, we report that the implementation of the first reduction did not solve any instance in the CPU time limit. Finally, concerning the last experiment, we report that it was only able to solve benchmarks having m/n = 3. Also in this case, DTPPYICES is faster: the median CPU time related to the first reduction is 182.16 for model A, and 18.81 for model B, while the same for DTPPYICES is 65.33 and 1.14, respectively.

We conjecture that the differences in performances between the two reductions are mainly due to the structure of the resulting formulas: YICES follows the lazy approach to SMT solving [?], which is based on abstracting the problem into a CNF formula to be processed by a SAT solver. However, a more detailed analysis is needed to corroborate this conjecture.

4.2. Comparison with the state-of-the-art

The results obtained in the experiments are shown in Figure **??**, whose organization corre-

$$L_{dc_A} = \{\langle\{(-49,-33),(77,90)\},1\rangle,\langle\{(-33,-14),(65,77)\},2\rangle,\langle\{(-14,65)\},3\rangle\}.$$

$$L_{dc_B} = \{\langle\{(-49,-33),(77,90)\},17\rangle,\langle\{(-33,-14),(65,77)\},55\rangle,\langle\{(-14,65)\},68\rangle\}.$$

Fig. 3. Examples of a constraint dc, whose preference values are generated with model A (L_{dc_A}) and with model B (L_{dc_B}).

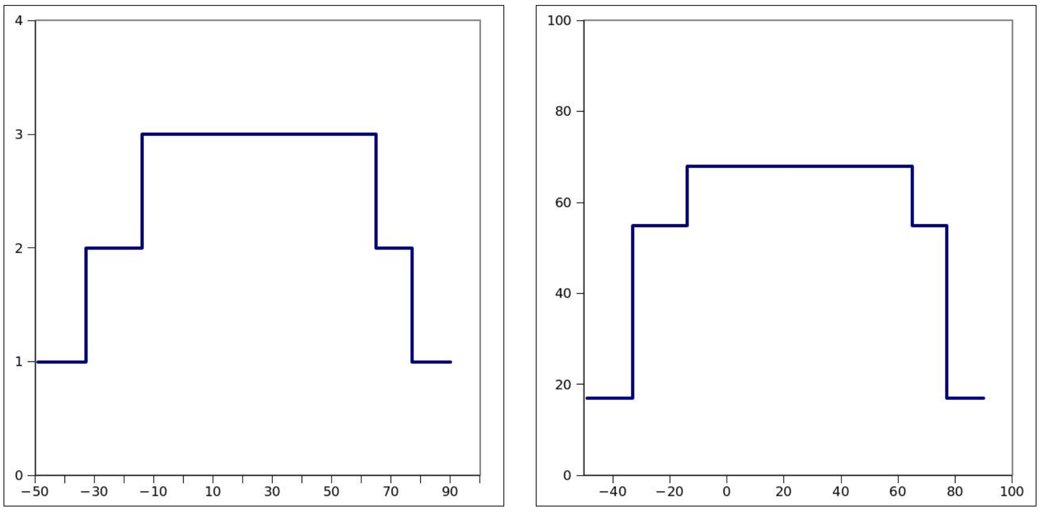

Fig. 4. Piece-wise constant preference functions related to L_{dc_A} (left) and L_{dc_B} (right) from Figure **??**. In the x-axis is reported the interval where the constraint is defined, while in the y-axis the preference values are reported.

sponds to the six groups of instances outlined above. Concerning the top-most plots, in the x-axis we show the total amount of constraints, while in the middle plots the total amount of levels of the piece-wise constant preference functions is reported. In the plots in the bottom we show the ratio of constraints to arithmetic variables. In the y-axis (in log scale), it is shown the CPU time (in seconds). MAXILITIS-BB's performance is depicted by a dotted line with blue triangles, MAXILITIS-IW's by using a dashed line with orange upside down triangles, and DTPPYICES performance is denoted by a solid line with black circles. Plots in the left-most column are related to model A, while plots in the right-most column are related to model B. Finally, points in the plots denote the related median CPU time (timeout values are not depicted).

Looking at Figure **??**, and considering the top-left plot, we can see that the median time of MAXILITIS-BB on benchmarks with 100 constraints runs into timeout. We can also see that up to $m = 80$, MAXILITIS-IW is one order of magnitude of CPU time faster that DTPPYICES, while for $m > 80$ the performance of the solvers are in the same ballpark. Now, considering the same analysis in the case of model B, we can see (top-right plot) that the picture changes in a noticeable way. Benchmarks are harder than before: MAXILITIS-BB and MAXILITIS-IW are not able to efficiently cope with benchmarks with $m > 30$. In this case, DTPPYICES is the best solver, and we report that it is able to deal with benchmarks up to $m = 60$.

Looking at the middle-left plot of Figure **??**, we can see that MAXILITIS-IW is the best solver up to $l = 7$, while for $l = 8$ we report that DTPPYICES is faster. Also in this case MAXILITIS-BB does not efficiently deal with the most difficult benchmarks in the suite. Looking now at the plot in the middle-right, we can see the same picture related to the bottom-right plot: the performance of both versions of MAXILITIS are very similar, while DTPPYICES is the fastest solver: the median CPU time of both MAXILITIS-BB and MAXILITIS-IW runs in timeout for $l > 5$, while DTPPYICES solves the majority of the benchmarks within the

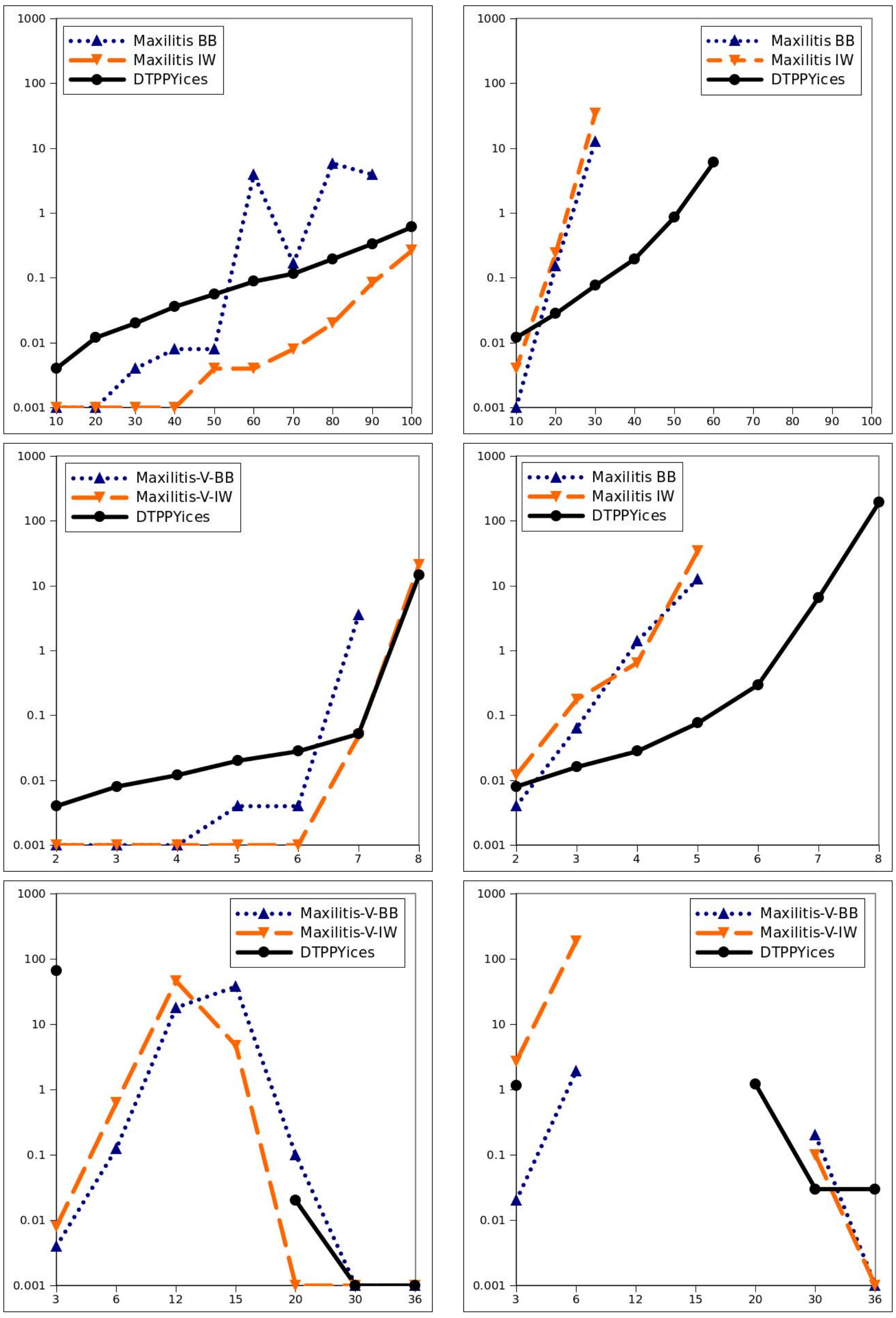

Fig. 5. Results of the evaluated solvers on random DTPPs considering the size of the benchmarks (top-most plots), number of preference levels (middle), and constraint density (bottom). Left-most plots are related to model A, while right-most plots depict the results related to model B.

time limit for all levels. Along with the previous results, this reveals that MAXILITIS may have specialized techniques to deal with DTPPs generated with model A. Finally, concerning our last experiment, we report that both versions of MAXILITIS are generally faster than DTPPYICES. Looking at the plot reporting the performance on model A (bottom-left), we can see that MAXILITIS is faster especially looking at $m/n < 20$. The picture changes with benchmarks generated with model B (bottom-right), where we can see that m/n equals to 12 and 15 represent a hard setting for all the considered solvers. Moreover, DTPPYICES is the only solver that can solve the majority of the benchmarks with $m/n = 20$.

Detailed results related to the plots in Figure **??** are reported in Tables **??**, **??**, and **??**.

5. Related Work

DTP is a well studied problem in literature: Along the years several systems that can solve DTPs have been developed, e.g., SK [**?**], TSAT [**?**], CSPI [**?**], EPILITIS [**?**], TSAT++ [**?**], and MATHSAT [**?**]. Moreover, the competition of solvers for Satisfiability Modulo Theories (SMT-COMP)[4] has two logics that include DTPs (called QF_RDL and QF_IDL, respectively), thus all SMT solvers that can deal with these logics can act as a DTP problems (note that TSAT++ and MATHSAT have been originally developed in the SMT field).

Then, the addition of preferences (see, e.g. [**?**]) has led to the definition of solvers for dealing with DTPPs. MAXILITIS [**?**,**?**] (that extends the DTP solver EPILITIS), WEIGHTWATCHER [**?**] and ARIO [**?**] (originated in the SMT field) implement different approaches for solving DTPPs as defined in [**?**]. MAXILITIS is a direct extension of the DTP solver EPILITIS [**?**], while WEIGHTWATCHER uses an approach based on Weighted Constraints Satisfaction problems. ARIO, instead, relies on an approach based on Mixed Logical Linear Programming problems. In our analysis we have used MAXILITIS because the results in, e.g. [**?**] clearly indicate its superior performance.

About the comparison to MAXILITIS, our solution is easy, yet efficient, and has a number of advantages w.r.t. the approach of MAXILITIS. On the modeling side, it allows to consider (with no modifications) both integer and real variables, while MAXILITIS can deal with integer variables only. Moreover, our implementation provides an unique framework for solving DTPPs, while the techniques proposed in [**?**] are implemented in two separate versions of MAXILITIS. Finally, our solution is modular, i.e. it is easy to rely on different back-end solvers (or, on a new version of YICES), thus taking advantages on new algorithms and tools for solving our formulas of interest.

DTPPs with a restricted form of preference functions have been tackled in [**?**] with a similar approach: in [**?**], only constant preference functions have been considered.

A previous version of this paper has been presented to the 5th Italian Workshop on Planning and Scheduling[5], and appears in [**?**]: in comparison, the current paper (*i*) introduces the important concepts in a more clear way, by relying on examples to illustrate such concepts, (*ii*) implements and evaluates the first reduction, (*iii*) includes more benchmarks, and (*iv*) extends the related work section.

6. Conclusions

In this paper we have introduced a general reduction-based approach for solving DTPPs, that reduces these problems to Maximum Satisfiability of a set of Boolean combination of constraints. An experimental analysis performed with the YICES SMT solver on randomly generated DTPPs shows that our approach is competitive to, and sometimes faster than, the specific implementations of the MAXILITIS solver. As a future work, we first plan to evaluate techniques (e.g. pre-processing) to further improve the performance of our approach, especially on Model A, and to run more detailed experiments to corroborate the conjectures (about the performances of the two reductions, and the differences in performances between Model A and Model B) in the paper.

The executable of our solver can be found at

`http://www.star.dist.unige.it/~marco/DTPPYices/`.

Acknowledgment. This research has been partially supported by the Autonomous Region of Sardinia (Italy) and the Port Authority of Cagliari (Sardinia, Italy) under grant agreement L.R. 7/2007, Tender 16 2011, CRP-49656 "Metodi innovativi per il supporto alle decisioni riguardanti l'ottimizzazione delle attività in un terminal container". The anonymous reviewers are thanked for

[4]`http://www.smtcomp.org/`.

[5]`http://ips2013.istc.cnr.it/`.

N	**M**	MAXILITIS-BB				MAXILITIS-IW				DTPPYICES			
		Mod. A		Mod. B		Mod. A		Mod. B		Mod. A		Mod. B	
		#	Time	#	Time	#	Time	#	Time	#	Time	#	Time
8	10	25	0.00	25	0.04	25	0.00	25	0.31	25	0.10	25	0.26
16	20	25	0.17	25	16.28	25	0.01	25	60.84	25	0.31	25	0.80
24	30	25	5.61	18	588.56	25	0.02	21	913.16	25	0.50	25	4.45
32	40	24	67.60	3	80.83	25	0.06	9	905.42	25	0.78	25	23.44
40	50	22	27.53	1	105.45	25	0.29	3	327.23	25	1.48	24	245.32
48	60	17	247.24	–	–	25	4.24	–	–	25	2.41	21	403.39
56	70	21	57.79	–	–	25	0.71	–	–	25	3.58	12	1355.01
64	80	16	151.82	–	–	25	7.13	–	–	25	4.75	6	414.62
72	90	17	399.10	–	–	25	15.30	–	–	25	6.72	5	674.02
80	100	12	774.92	–	–	25	57.90	–	–	25	10.72	3	164.91

Table 1

Performance of the selected solvers on random DTPPs with different sizes. The first columns ("N") reports the total amount of variables for each pool of DTPPs, while the second one ("M") reports the total amount of constraints. It is followed by three groups of columns, and the label is the solver name. Each group is composed of four columns, reporting the total amount of instances solved within the time limit ("#") and the total CPU time – i.e., the sum of CPU time related to the solved instances – in seconds ("Time") spent, in the case of model A and B (groups "Mod. A" and "Mod. B", respectively). In case a solver does not solve any instance, "–" is reported.

L	MAXILITIS-BB				MAXILITIS-IW				DTPPYICES			
	Mod. A		Mod. B		Mod. A		Mod. B		Mod. A		Mod. B	
	#	Time	#	Time	#	Time	#	Time	#	Time	#	Time
2	25	0.01	25	1.68	25	0.01	25	2.836	25	0.10	25	0.20
3	25	0.01	25	7.05	25	0.01	25	47.261	25	0.20	25	0.38
4	25	0.01	21	395.00	25	0.01	25	203.523	25	0.32	25	0.87
5	25	5.62	18	589.33	25	0.04	21	914.453	25	0.50	25	4.44
6	24	32.66	10	673.87	25	4.80	10	608.038	25	1.82	25	17.28
7	21	230.50	2	68.01	23	129.72	2	59.572	23	56.05	21	528.05
8	12	434.55	2	216.89	17	598.42	2	303.079	15	228.16	13	487.39

Table 2

Performance of the selected solvers on random DTPPs with different levels. In column "L" we report the total amount of levels, while the rest of the table is organized similarly to Table **??**.

suggestions and criticisms. The authors would like also to thank Michael D. Moffitt for providing his solvers and the program for generating random benchmarks, and Bruno Dutertre for his support with YICES. Jean-Rémi Bourguet is thanked for having provided a first implementation of the second reduction.

References

[1] Alessandro Armando, Claudio Castellini, and Enrico Giunchiglia. SAT-based procedures for temporal reasoning. In S. Biundo and M. Fox, editors, *Proc. of the 5th European Conference on Planning (ICAPS 1999)*, volume 1809 of *Lecture Notes in Computer Science*, pages 97–108. Springer, 1999.

[2] Alessandro Armando, Claudio Castellini, Enrico Giunchiglia, and Marco Maratea. The SAT-based approach to Separation Logic. *Journal of Automated*

m/n	Maxilitis-BB				Maxilitis-IW				dtppYices			
	Mod. A		Mod. B		Mod. A		Mod. B		Mod. A		Mod. B	
	#	Time	#	Time	#	Time	#	Time	#	Time	#	Time
3	25	0.01	25	0.04	25	0.24	25	73.02	14	312.17	25	71.88
6	25	1.73	26	28.58	25	36.72	14	1548.19	–	–	1	174.40
12	24	1391.20	1	140.84	19	882.03	–	–	3	318.15	–	–
15	15	258.68	2	516.95	20	338.29	1	73.92	11	10.06	6	497.41
20	21	63.28	8	863.85	25	3.61	7	589.64	25	3.08	24	186.87
30	25	0.03	24	338.19	25	0.01	24	147.92	25	0.45	25	0.94
36	25	0.02	25	0.55	25	0.00	25	0.53	25	0.42	25	0.80

Table 3

Performance of the selected solvers on random DTPPs with different constraint density. In column "m/n" we report the ratio of constraints to arithmetic variables, while the rest of the table is organized similarly to Table **??**.

Reasoning, 35(1-3):237–263, 2005.

[3] Jean-Rémi Bourguet, Marco Maratea, and Luca Pulina. A reduction-based approach for solving disjunctive temporal problems with preferences. In Matteo Baldoni, Cristina Baroglio, Guido Boella, and Roberto Micalizio, editors, *Proc. of the 13th International Conference of the Italian Association for Artificial Intelligence (AI*IA 2013): Advances in Artificial Intelligence*, volume 8249 of *Lecture Notes in Computer Science*, pages 445–456. Springer, 2013.

[4] Marco Bozzano, Roberto Bruttomesso, Alessandro Cimatti, Tommi A. Junttila, Peter van Rossum, Stephan Schulz, and Roberto Sebastiani. MathSAT: Tight integration of SAT and mathematical decision procedures. *Journal of Automated Reasoning*, 35(1-3):265–293, 2005.

[5] Rina Dechter, Itay Meiri, and Judea Pearl. Temporal constraint networks. *Artificial Intelligence*, 49(1-3):61–95, 1991.

[6] Bruno Dutertre and Leonardo de Moura. The Yices SMT solver. Technical report, Stanford Research Institute, 2006.

[7] Marco Maratea and Luca Pulina. Solving disjunctive temporal problems with preferences using maximum satisfiability. *AI Commununications*, 25(2):137–156, 2012.

[8] Michael D. Moffitt. On the modelling and optimization of preferences in constraint-based temporal reasoning. *Artificial Intelligence*, 175(7-8):1390–1409, 2011.

[9] Michael D. Moffitt and Martha E. Pollack. Partial constraint satisfaction of disjunctive temporal problems. In Ingrid Russell and Zdravko Markov, editors, *Proc. of the 18th International Conference of the Florida Artificial Intelligence Research Society (FLAIRS 2005)*, pages 715–720. AAAI Press, 2005.

[10] Michael D. Moffitt and Martha E. Pollack. Temporal preference optimization as weighted constraint satisfaction. In *Proc. of the 21st National Conference on Artificial Intelligence (AAAI 2006)*. AAAI Press, 2006.

[11] A. Oddi and A. Cesta. Incremental forward checking for the disjunctive temporal problem. In *Proceedings of the 14th European Conference on Artificial Intelligence (ECAI-2000)*, pages 108–112, Berlin, 2000.

[12] Bart Peintner, Michael D. Moffitt, and Martha E. Pollack. Solving over-constrained disjunctive temporal problems with preferences. In Susanne Biundo, Karen L. Myers, and Kanna Rajan, editors, *Proc. of the 15th International Conference on Automated Planning and Scheduling (ICAPS 2005)*, pages 202–211. AAAI, 2005.

[13] Bart Peintner and Martha E. Pollack. Low-cost addition of preferences to DTPs and TCSPs. In Deborah L. McGuinness and George Ferguson, editors, *Proc. of the 19th National Conference on Artificial Intelligence (AAAI 2004)*, pages 723–728. AAAI Press / The MIT Press, 2004.

[14] Roberto Sebastiani. Lazy satisability modulo theories. *Journal of Satisfiability, Boolean Modeling and Computation*, 3(3-4):141–224, 2007.

[15] Hossein M. Sheini, Bart Peintner, Karem A. Sakallah, and Martha E. Pollack. On solving soft temporal constraints using SAT techniques. In Peter van Beek, editor, *Proc. of the 11th International Conference on Principles and Practice of Constraint Programming (CP 2005)*, volume 3709 of *Lecture Notes in Computer Science*, pages 607–621. Springer, 2005.

[16] Kostas Stergiou and Manolis Koubarakis. Backtracking algorithms for disjunctions of temporal constraints. In Howard E. Shrobe, Tom M. Mitchell, and Reid G. Smith, editors, *Proc. of the 15th National Conference on Artificial Intelligence (AAAI 1998)*, pages 248–253. AAAI Press / The MIT Press, 1998.

[17] Kostas Stergiou and Manolis Koubarakis. Backtracking algorithms for disjunctions of temporal constraints. *Artificial Intelligence*, 120(1):81–117, 2000.

[18] Ioannis Tsamardinos and Martha Pollack. Efficient solution techniques for disjunctive temporal reasoning problems. *Artificial Intelligence*, 151:43–89, 2003.