Solving Disjunctive Temporal Problems with Preferences using Maximum Satisfiability *

Marco Maratea [a,**] and Luca Pulina [b]

[a] *DIBRIS, University of Genova, Viale F. Causa 15, Genova, Italy*
E-mail: marco@dist.unige.it
[b] *POLCOMING, University of Sassari, Viale Mancini 5, Sassari, Italy*
E-mail: lpulina@uniss.it

The Disjunctive Temporal Problem (DTP) involves conjunction of DTP constraints, each DTP constraint being a disjunction of difference constraints of the form $x - y \leq c$, where x and y range over a domain of interpretation, and c is a numeric constant. The DTP is recognized to be an expressive framework for constraints modeling and processing. The addition of preferences, in the form of weights associated to difference constraints for their satisfaction, needs methods for aggregating preferences among and within DTP constraints to compute meaningful and high quality solutions, while further enhancing DTP expressivity and applicability.

In this paper we consider an utilitarian aggregation of DTP constraints weights, and a prominent semantic for aggregating such weights from its difference constraints weights that considers the *maximum* among the weights associated to satisfied difference constraints in it. We present a novel approach that reduces the problem to Maximum Satisfiability of DTPs (Max-DTPs). In this way, we can employ off-the-shelf Max-DTP solvers with different solution methods, ranging from Satisfiability Modulo Theories (SMT), to interval-based and Boolean optimization-based solvers. We then compare the performance of our approach with different back-end solvers on both randomly generated and real-world benchmarks, in comparison with Maxilitis, the best solver that can deal with DTPs with preferences using the aggregation methods considered. Results show that the Yices SMT solver is the best, and that Yices and the TSAT# solver based on Boolean optimization can be orders of magnitude faster than Maxilitis.

Keywords: Disjunctive Temporal Problems, Optimization

1. Introduction

The Disjunctive Temporal Problem (DTP), introduced in [SK98], is defined as a finite conjunction of DTP constraints, each DTP constraint being a finite disjunction of difference constraints of the form $x - y \leq c$, where x and y are arithmetic variables ranging over a domain of interpretation (the set of real numbers $\mathcal{R}$ or the set of integers $\mathcal{Z}$), and c is a numeric constant. The goal is to find an assignment to the variables of the problem such that all DTP constraints are satisfied. The DTP is recognized to be a good compromise between expressivity and efficiency, given that the arithmetic consistency of a set of difference constraints can be checked in polynomial time, and has found applications in many areas such as planning, scheduling, hardware and software verification, see, e.g., [ORC10,BBC+05].

With the addiction of preferences [PP04,MP05, NO06,Mof11,MPP11], e.g., in the form of weights associated to difference constraints for their satisfaction, methods for aggregating preferences among and within DTP constraints have to be considered to compute meaningful and high quality solutions, at the same time enhancing temporal constraints expressivity and applicability, see, e.g., [MVSO04,BGPYS07,BGPYS11]. In the context of temporal preferences, the mainly considered aggregation methods of constraints preferences are the *maximin* aggregation [KMMV03], which considers the minimum value among the satisfied constraints, and the *utilitarian* aggregation [MMK+04,Kum04,PP05,Mof11], where the solution quality is computed by the sum of the weights of satisfied constraints. While the maximin

*This paper is an extended and revised version of [MPP11].

**Corresponding author.

AI Communications
ISSN 0921-7126, IOS Press.

aggregation method can be proper in some situations [KMMV03], the second is often preferred given that it considers the global level of satisfaction of the whole problem.

In this paper we consider an *utilitarian* aggregation of DTP constraints weights, and a prominent semantic for aggregating such weights from the difference constraints weights: the *max* semantic. Such semantic, employed in, e.g., [MP05,Mof11] (where the whole problem is called DTPP) assigns as DTP constraint weight the *maximum* among the weights associated to satisfied difference constraints in it.

We present a novel approach that reduces the problem of finding utilitarian optimal solutions of DTPs under the *max* semantic to Maximum Satisfiability of DTPs (Max-DTPs), a problem already studied in, e.g., [NO06], we dealt with in [MPP11]. Our reduction allows us to employ off-the-shelf Max-DTP solvers with different solution methods, ranging from one based on Boolean optimization [MPP11] with the TSAT# solver, to Satisfiability Modulo Theories (SMT), e.g., YICES [DdM06], and interval-based (HYSAT [FHT+07]) solvers.

We finally evaluate the performance of our approach on both randomly generated DTPs, which is the most widely analyzed domain of benchmarks in the literature of DTP with preferences, using a well-known generation method from [SK00], and non-random benchmarks from the SMT-LIB[1] that include DTPs with real- and integer-valued variables (e.g., the QF_IDL logic), both augmented with randomly generated weights. The analysis considers our approach with the back-end solvers mentioned above and MAXILITIS [MP05,MP06, Mof11], the best solver that can deal with DTPs with preferences[2] using the aggregation methods considered. Results show that YICES is the best among the back-end solvers employed, followed by TSAT#, and that their performances can be orders of magnitude better than MAXILITIS.

To sum up the main contributions of this paper:

- We define a new reduction from DTPs with preferences, as defined in this paper, to Max-DTPs;
- We implement the reduction, allowing for the use of various back-end systems with different input formats;
- We run a wide experimental analysis involving both random and non-random benchmarks, our approach with different back-end solvers and the state-of-the-art solver MAXILITIS.

The rest of the paper is structured as follows. Section 2 introduces needed preliminaries about DTPs and DTPs augmented with weights such as Max-DTPs. Section 3 presents the reduction from our problem of interest to Max-DTP. Next, benchmarks and solvers employed are outlined in Section 4, while the experimental analysis is presented in Section 5. The paper ends by providing a discussion about the related work in Section 6, and some conclusions and topics for future research in Section 7.

2. Formal Background

Problems involving disjunction of temporal constraints have been introduced in [SK98], as an extension of the Simple Temporal Problem (STP) [DMP91], which consists of conjunction of different constraints. The problem was referred for the first time as Disjunctive Temporal Problem (DTP) in [ACG99].

Syntax. Let $\mathcal{V}$ be a set of symbols, called *variables*. A *difference constraint* is an expression of the form $x - y \leq c$, where $x, y \in \mathcal{V}$, and c is a numeric constant. A *DTP constraint*, or simply *constraint*, is a disjunctively intended set of difference constraints. A *DTP formula*, or simply *formula*, is a conjunctively intended set 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 method employed and defined below. Finally, a *constraint literal*, or simply *literal*, is either a difference constraint or its negation. If a is a difference constraint, then $\overline{a}$ abbreviates $\neg a$ and $\overline{\neg a}$ stands for a. A *conjunctive DTP constraint* is a conjunctively intended set of constraint literals.

[1] `http:www.smtlib.org`.

[2] MAXILITIS can solve problems with more complex preferences than the one we consider in this paper, restricted to integer-valued variables, see e.g., [Mof11].

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

- $\sigma \models x - y \leq c$ if and only if $\sigma(x) - \sigma(y) \leq c$;
- $\sigma \models \neg\phi$ 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
- $\sigma \models (\vee_{i=1}^{n}\phi_i)$ 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 almost always treated uniformly. Therefore, from now on, we will drop the distinction (and we will re-introduce it only when needed).

Example 1. The following formula, where $\mathcal{D}$ is $\mathcal{Z}$

$$(x - y \leq 7 \vee z - x \leq -20) \wedge z - y \leq 10$$

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

2.1. Max-DTP

Consider now a DTP formula ϕ consisting of both hard and soft DTP constraints, as an extension of the formula defined earlier. Intuitively, in this case the goal is to find an assignment that satisfies all hard DTP constraints and maximizes the sum of weights associated to satisfied soft DTP constraints.

The problem is called maximum satisfiability of DTP, i.e., "Max-DTP" (resp. Weighted Max-DTP) in case all DTP constraints are soft and their weights are uniform (resp. non-uniform). The same distinction holds for "Partial Max-DTP" and "Partial Weighted Max-DTP" in case the formula contains both hard and soft DTP constraints.[3]
We now formally define the more general type of DTP formula we have described. A Partial Weighted Max-DTP is 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 DTP constraints to positive integer numbers.

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

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

where d is a soft DTP constraint. We also refer to σ' as an *optimal solution* for ϕ.

Example 2. For simplicity, we herewith consider arithmetic operators other than $\leq$. We remind that all arithmetic operators can be easily recasted in $\leq$.

The following formula ϕ, where $\mathcal{D}$ is $\mathcal{Z}$

$$d_1 : (x - y \leq 7 \vee z - x \leq -20) \wedge$$

$$d_2 : x - y \geq 10 \wedge$$

$$d_3 : z - x \geq 0$$

is not satisfiable.
Consider $w(d_1)$=3, $w(d_2)$=1 and $w(d_3)$=1. σ_1 is an optimal solution for ϕ as well as, e.g., σ_2 that assigns $x = 30$, $y = 2$ and $z = 10$, given that for both assignments the corresponding value of f is 4.

In the following, we will always use Max-DTP to refer to the optimization problem at hand, while it will be clear from the context which is the particular optimization problem being solved.

[3] The names of the problems are borrowed from the terminology used in the Max-SAT Evaluations and Competitions, see, e.g., [ALMP08]. In related works, e.g., [dM, NO06] always Max-SMT (meaning, e.g., Max-DTP in case the optimization is defined on a DTP) is used.

2.2. DTP with preferences

In this paper we consider a more general problem than Max-DTP: the DTP with preferences is 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 (possibly partial) function that maps difference constraints to positive integer numbers.

We consider an utilitarian method for aggregating soft DTP constraints weights: our goal is thus to find an assignment σ' for ϕ that

1. satisfies all hard DTP constraints; and
2. maximizes the sum of weights associated to satisfied soft DTP constraints, i.e., maximizes the linear objective function (1).

It is left to define how weights are aggregated within soft DTP constraints to define their weights $w(d)$ in (1). In our work we consider a prominent semantics for this purpose.

The *max* semantic [MP05,Mof11] defines the weight $w(d)$ of a satisfied soft DTP constraint d as the maximum among the weights of satisfied difference constraints in d, i.e.

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

Example 3. For simplicity, we again consider arithmetic operators other than $\leq$. Consider the same formula ϕ as in Example 2.

With d_{11} and d_{12} we refer to the first and second different constraints in d_1, respectively. Similarly for d_2 and d_3. Consider $w(d_{11})$=2, $w(d_{12})$=3, $w(d_{21})$=1 and $w(d_{31})$=1. σ_1 is not anymore an optimal solution for ϕ, while σ_2 is still an optimal solution.

Note that the considered problem generalizes the one we dealt with in [MPP11] in which

(i) all DTP constraints are soft; and
(ii) the weights associated to difference constraints in a soft DTP constraint are the same,

i.e., the Weighted Max-DTP.

3. Reduction to Max-DTP

As we said before, our main idea is to reduce the problem of solving DTPs with preferences to Max-DTPs. Hard DTP constraints remain unchanged in our reduction, while soft DTP constraints need special treatment.

We remind that the aggregation of preferences within DTP constraints consider the max semantic: intuitively, our idea is to express a soft DTP constraint d using soft conjunctive DTP constraints that force the highest weight associated to satisfied difference constraints in d to be assigned as weight for d.

Consider a soft DTP constraint

$$d := dc_1 \vee ... \vee dc_k \tag{2}$$

where $\{dc_1, \ldots, dc_k\}$ is the set of difference constraints in d. Further consider an ordering on the difference constraints $\{dc_1, \ldots, dc_k\}$ induced by their weights, i.e., an ordering $\prec$ is which:

$$dc_i \prec dc_j \text{ iff } w(dc_i) \geq w(dc_j), 1 \leq i, j \leq k, i \neq j.$$

For simplicity, from now on we consider the difference constraints in d to be re-ordered according to $\prec$, i.e., in Eq. (2) dc_1 is the difference constraint whose $w(dc_1)$ is maximum among the weights in d, i.e.,

$$w(dc_1) \geq w(dc_i), 2 \leq i \leq k$$

while $w(dc_k)$ is such that

$$w(dc_k) \leq w(dc_i), 1 \leq i \leq k-1.$$

Consider w to be also defined on conjunctive DTP constraints. Then, d is expressed by the following k conjunctive DTP constraints: for each $z = 1 \ldots k$

$$c_z := \wedge_{i=1}^{z-1} \neg dc_i \wedge dc_z, w(d) = w(c_z) = w(dc_z) \tag{3}$$

The constraints are to be intended in disjunction, and are mutually exclusive: considering an assignment, at most one of the constraints can be satisfied, and the relative weight is assigned to d.

Example 4. Consider the soft DTP constraint d:

$$x - y \leq 5 \vee x - z \leq 7 \vee q - x \leq 15$$

where $x, y, z, q \in \mathcal{Z}$, and with $w(x - y \leq 5) = 4, w(x - z \leq 7) = 2$, and $w(q - x \leq 15) = 9$, d is expressed with the following 3 conjunctive constraints and related weights:

$$c_1 := q - x \leq 15, w(c_1) = 9$$

$$c_2 := \neg(q - x \leq 15) \wedge x - y \leq 5, w(c_2) = 4$$

$$c_3 := \neg(q - x \leq 15) \wedge \neg(x - y \leq 5) \wedge x - z \leq 7, w(c_3) = 2$$

This starting point for our reduction seems to lead to a quite natural problem representation, but we have to consider that it generates conjunctive constraints: this is not a problem in general, but given that our ultimate goal is to evaluate our reduction method by using off-the-shelf solvers, and that most (if not all) systems can only deal with conjunction of DTP constraints, this reduction can not be directly applied to compute optimal solutions.

The transformation of a set of conjunctive DTP constraints as defined before in (3) to Conjunctive Normal Form (CNF) leads to the following k soft DTP constraints: for each $z = 1 \dots k$

$$c'_z := \vee_{i=1}^{z} dc_i \tag{4}$$

The intuition is that each new constraint c'_z is the result of applying a plain CNF transformation to the first z conjunctive constraints in (3), i.e., to the set $\{c_i : 1 \leq i \leq z\}$.

The problem is now to define what are the weights associated to each newly defined soft difference constraint, in order to reflect the semantic of our problem. In the previous formulation (2) the difference constraints occurred only once positively, but now there are possibly several positive occurrences in the corresponding soft DTP constraints in (4) that influence constraints weights adaptation and definition. Our solution starts from the following fact: if the constraint c'_k, i.e., the constraint that is equivalent to d in Eq. (2) is satisfied, we know that it contributes for at least the weight of the difference constraint with minimal weights, i.e., $w(dc_k)$. Satisfying the constraint c'_{k-1} contributes for $w(dc_{k-1}) - w(dc_k)$, and given that a constraint c'_z implies all constraints $c'_{z'}$, $z' > z$, these two soft constraints together contribute for $w(dc_{k-1})$. This method is recursively applied up to dc_1, whose weight is $w(dc_1) - w(dc_2)$ and, given it implies all other introduced soft DTP constraints, satisfying dc_1 correctly corresponds to assign a weight $w(dc_1)$ to d.

More formally, for each $z = 1 \dots k$

$$w(c'_z) = \begin{cases} w(dc_k) & z = k \\ w(dc_z) - w(dc_{z+1}) & 1 \leq z < k \end{cases} \tag{5}$$

and, given an assignment σ'

$$w(d) = \sum_{z \in \{1 \dots k\}, \sigma' \models c'_z} w(c'_z)$$

Example 5. The soft DTP constraints that express the constraint d with the associated weights to its difference constraints from Example 4 are:

$$c'_1 := q - x \leq 15, w(c'_1) = 5$$

$$c'_2 := q - x \leq 15 \vee x - y \leq 5, w(c'_2) = 2$$

$$c'_3 := q - x \leq 15 \vee x - y \leq 5 \vee x - z \leq 7, w(c'_3) = 2$$

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 soft DTP constraints in the reduced formula ϕ', e.g., if the same difference constraint has highest weight in more than one constraint in 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 constraints in ϕ. Thus, in this case, a solution is to consider a *single instance* of the resulting soft constraint in ϕ' whose weight is the sum of the weights of the various occurrences.

Moreover, the reduction is still correct assuming that the difference constraints weights in a DTP constraint induce a total order on these difference constraints, i.e., if all weights are different as in Example 4. Instead, our problem formulation allows for repeated weights in a soft constraint, and a further problem can arise with our reduction: if there are at least two difference constraints with the same weight in d, our reduction would contain a DTP constraint with weight equals to 0, as shown in Example 6, whose semantic is not clear given the definition of a Maximum Satisfiability problem.

Example 6. Consider the soft DTP constraint d in Example 4 but with $w(x - z \leq 7) = 4$. The soft DTP constraints that, given our encoding, would express the constraint d with the associated weights are:

$$c'_1 := q - x \leq 15, w(c'_1) = 5$$

$$c'_2 := q - x \leq 15 \vee x - y \leq 5, w(c'_2) = 0$$

$$c'_3 := q - x \leq 15 \vee x - y \leq 5 \vee x - z \leq 7, w(c'_3) = 2$$

which contains a constraint (c'_2) with weight 0.

In this case, the correct reduction works as follows. Consider Dc to be a set of difference constraints with the same weight. Thus, d can now be expressed as

$$d := \vee_{dc \in Dc_1} dc \vee ... \vee \vee_{dc \in Dc_{k'}} dc \tag{6}$$

where dc is a difference constraint, and $k' \leq k$.

The reduced formula contains k' soft DTP constraints: for each $z = 1 \dots k'$

$$c'_z := \vee_{i=1}^{z} \vee_{dc \in Dc_i} dc$$

Further extending w to be defined on sets of difference constraints, given such a set Dc we define

$$w(Dc) = w(dc), dc \in Dc$$

The definition of $w(c'_z)$ is the same as (5) but with Dc in place of dc, and k' in place of k. Note that the case where all difference constraints have the same weight is mapped directly into a single soft Max-DTP constraint.

Example 7. Consider the soft DTP constraint in Example 4, but with the following weights: $w(x - y \leq 5) = 4, w(x - z \leq 7) = 4$, and $w(q - x \leq 15) = 9$. Thus, $Dc_1 = \{q - x \leq 15\}$, $Dc_2 = \{x - y \leq 5, x - z \leq 7\}$, $w(Dc_1) = 9$ and $w(Dc_2) = 4$.

d is expressed with the following 2 constraints and related weights:

$$c'_1 := q - x \leq 15, w(c'_1) = 5$$

$$c'_2 := q - x \leq 15 \vee x - y \leq 5 \vee x - z \leq 7, w(c'_2) = 4$$

Satisfying $x - y \leq 5$ or $x - z \leq 7$ contributes for a weight of 4, which is the weight of the whole constraint if $q - x \leq 15$ is not satisfied.

3.1. An alternative formulation

The formulation we have so far employed assigns weights to soft DTP constraints of the reduced formula. Given that not all systems can directly deal with this formulation, we herewith present an alternative formulation that can be useful at implementation level: the idea of this alternative is to

- add to each resulting soft DTP constraint a "constraint selector" to take into account its (un)satisfaction; and
- apply the weights over such selectors, instead over soft constraints.

For simplicity we use Boolean variables as selectors, but they can be simulated with dummy difference constraints: thus, there is no need to adapt the syntax of our problem, and/or to extend the definition of assignment and function w. Consider first the simple setting in which (i) it is not the case that the reduced formula contains the same DTP constraint more than once, and (ii) all weights associated to difference constraints of the same soft DTP constraints are different.

Each soft DTP constraint d is now expressed by the following k constraints: for $z = 1 \dots k$

$$c'_z := s_z \vee \vee_{i=1}^{z} dc_i$$

and

$$w(s_z) = w(c'_z)$$

as in (5).

The goal is to find an assignment σ' that minimizes[4] the following linear objective function:

$$\sum_{c'_z \in \phi', \sigma' \models s_z} w(s_z) \tag{7}$$

Intuitively, in case a soft constraint $\vee_{i=1}^{z} dc_i$ is not satisfied, in order for c'_z to be satisfied the related constraint selector must be true and the related weight is counted, otherwise is false.

[4]We can still maximize such function, by considering "negated" constraint selectors in the formulation, but in this case we have to update the definition of DTP constraints by allowing for constraint literals.

Example 8. Consider the formula in Example 5. The soft DTP constraints that express the constraint d with the associated weights to selectors are:

$$c'_1 := s_1 \vee q - x \leq 15, w(s_1) = 5$$

$$c'_2 := s_2 \vee q - x \leq 15 \vee x - y \leq 5, w(s_2) = 2$$

$$c'_3 := s_3 \vee q - x \leq 15 \vee x - y \leq 5 \vee x - z \leq 7, w(s_3) = 2$$

The formulation can be (easily) extended to the more general cases in which (i) and (ii) do not hold as we have done in the first formulation.

4. Solvers and Benchmarks

In our experimental analysis, we consider a pool of off-the-shelf solvers – some of them not tailored to deal with DTPs with constant preferences as defined in this paper. The reduction described in the previous section allow us to use Max-DTP solvers to cope with the reduced Max-DTPs problem. We selected the following solvers:

HySAT [FHT+07] (ver. 0.86): it is an interval-based solver. Its core algorithm is composed by a tight integration between state-of-the-art SAT solving techniques and Interval Constraints Propagation [Dav87]. HySAT also features optimization, i.e., it can be used to determine a solution which minimizes the value of a given variable in the input formula, which describes the whole optimization.

Maxilitis [MP05,MP06,Mof11]: it is an extension of the DTP solver Epilitis [TP03], and it is able to directly deal with DTPs with preferences. It is based on a branch and bound algorithm on top of which there are several pruning techniques, e.g., forward checking, removal of subsumed variables, and semantic branching.

TSAT# [MPP11]: it is an extension of the TSAT++ solver [ACG+05] targeted to handle preferences expressed as weights on DTP constraints. In order to do that, it features several systems in the generation phase of candidate solutions able to solve Boolean optimization problems like Max-SAT, Pseudo-Boolean (PB), and Answer Set Programming (ASP) [GL88,GL91]. Such systems are used as back-end instead of the SAT solver simo [GMT03] of TSAT++. It also features optimizations like IS_2 pre-processing and model reduction, see, e.g., [ACG+05,MPP11]. The employed Boolean Optimization solver in the following analysis is the Max-SAT solver IncWMaxSatz [LS07,LSL08].

Yices [DM06]: it is a SMT solver integrating a Davis-Putnam-Logemann-Loveland (DPLL)-based SAT solver with several specialized first-order theories solvers. It also natively includes the feature to solve weighted MAX-SMT formulas.

The HySAT solver relies on the alternative formulation of Section 3.1.

We evaluated the solvers listed above using both randomly generated benchmarks and non-random ones from SMT-LIB. Random benchmarks are the mostly widely used domain in the literature of DTPs with preferences (see, e.g., [MP05, SPSP05,MP06,NK08,Mof11]). We used the well-established generation method from [SK98,SK00] extended with randomly generated weights. Thus, DTPs with preferences are randomly generated by fixing

- the number k of disjuncts per constraint;
- the number n of arithmetic variables;
- the number m of DTP constraints;
- a positive integer L such that all constants are taken in $[-L, L]$; and
- a positive integer w such that the different constraints weights are taken in $[1, v]$.

In our experiments, we set $k = \{2, 3\}$, $L = 100$, $v = 100$, $n = \{5, 10, 15\}$. For $k = 2$,[5] we generated benchmarks having the ratio m/n varying from 2 to 14, while considering $k = 3$, m/n ranges from 2 to 20. A weight w is generated and associated to each difference constraint in a soft DTP constraint (unless the difference constraint has been already generated in other soft DTP constraints, in this case it already has an associated weight).

For each tuple of values of the parameters, 10 instances are generated. The range of m/n has been used to obtain both satisfiable and unsatisfiable DTPs. For each instance, 4 mutations are generated in which the first 0%, 25%, 50% and 75% of

[5] Note that while 2-CNF is a polynomial class of SAT, it is not the case for DTPs with $n = 2$.

the DTP constraints are hard, and the remaining are soft.

Considering the non-random benchmarks from SMT-LIB, first we investigated formulas in QF_IDL, in order to select formulas that could be solved by all the considered systems: we remind that MAXILITIS does not support real-valued variables, and parses only $\leq$ as arithmetic operator. The second constraint in our choice is that we select benchmarks for which only simple syntactic transformations are needed to satisfy our first constraint. The main goal of the non-random benchmarks is to test MAXILITIS and our back-end solvers on structured benchmarks. In conclusion, we mainly focus on two families from SMT-LIB, namely DIAMONDS [SSB02], and JOBSHOP – 120 formulas related to the basic formulation[6] of the Job Shop Scheduling Problem in [PS82]. Also for these problems, for each instance the previously mentioned 4 mutations are generated.

5. Experimental Analysis

The analysis is presented in two parts. The first subsection contains the results for problems with soft constraints only, which is the most common setting in the literature of DTPs with preferences. The second subsection reports the results obtained on problems with both hard and soft constraints.

5.1. Benchmarks with soft constraints only

The experiments here presented ran on PCs equipped with a processor Intel Core 2 Duo running at 2.13 GHz, with 2 GB of RAM, and running GNU Linux Ubuntu 2.6.32. The timeout for each instance has been set to 900s. To prevent memory swapping, we also set a memory limit at 2GB.

As first experiment, we evaluated the solvers on random benchmarks having $k = 2$, and the results are shown in Figure 1, in which the median of the CPU times among the 10 generated instances is plotted against m/n. Looking at the figure, and considering the results related to the benchmarks with integer-valued variables (left-most plots), we can see that YICES is the only solver that never reaches the time limit. In particular, considering $n = 5$ (top-left plot in Figure 1), we can see that HYSAT is not able to solve formulas beyond $m/n = 6$, while performance of the remaining solvers are in the same ballpark. From $m/n = 10$ to $m/n = 14$, both MAXILITIS and YICES are about one order of magnitude faster that TSAT#. Considering now the results related to $n = 10$ (middle-left plot), we can see that benchmarks with $m/n > 4$ are too hard for HYSAT. We also can see that again only YICES solves all benchmarks without to exhaust its time resources: MAXILITIS stops to $m/n = 10$, while TSAT# reaches the time limit for $m/n = 13$. We also report that the relationship between the performance of MAXILITIS and YICES change substantially: while for $n = 5$ they perform similarly, in this case, for $m/n > 4$ YICES is at least one order of magnitude faster. Also the relationship between MAXILITIS and TSAT# performances changes in this setting. Finally, notice that YICES is not always the best solver in this case: for $m/n = 9$ TSAT# is faster by a factor of 3. Looking now at the results related to $n = 15$ (bottom-left plot), we can see that the benchmarks turn to be very difficult for the considered solvers. In particular, HYSAT stops to $m/n = 3$ and MAXILITIS to $m/n = 6$. Despite the fact that TSAT++ stops to $m/n = 8$, we can see that it is the best solver for $m/n = \{5, 6, 7\}$, while YICES is the best on higher values of m/n.

Considering now the results related to benchmarks with real-valued variables, looking at Figure 1 (right-most plots), we can see that no results are reported for MAXILITIS because we remind it can only solve benchmarks with integer-valued variables. Considering $n = 5$ (top-right plot), we can see that HYSAT stops now at $m/n = 10$, while considering both TSAT# and YICES, the picture is similar as the related case for integer-valued variables. Considering $n = 10$, we can report analogous conclusions as the integer-valued variables case. On the other hand, for $n = 15$ (bottom-right plot), we can see that no solver is able to solve benchmarks related to all values of the ratio m/n: HYSAT stops to $m/n = 3$, TSAT# stops to $m/n = 9$, while YICES fails to solve benchmarks for $m/n = \{11, 13, 14\}$. Notice that both HYSAT and TSAT# show slightly better performances with respect to the integer-valued variables case, while it is the converse for YICES.

Detailed results for these benchmarks containing, for each solver and ratio, the number of solved instances, and the sum of their solving CPU times, are reported in Table 1.

[6] Personal communications with Hyondeuk Kim.

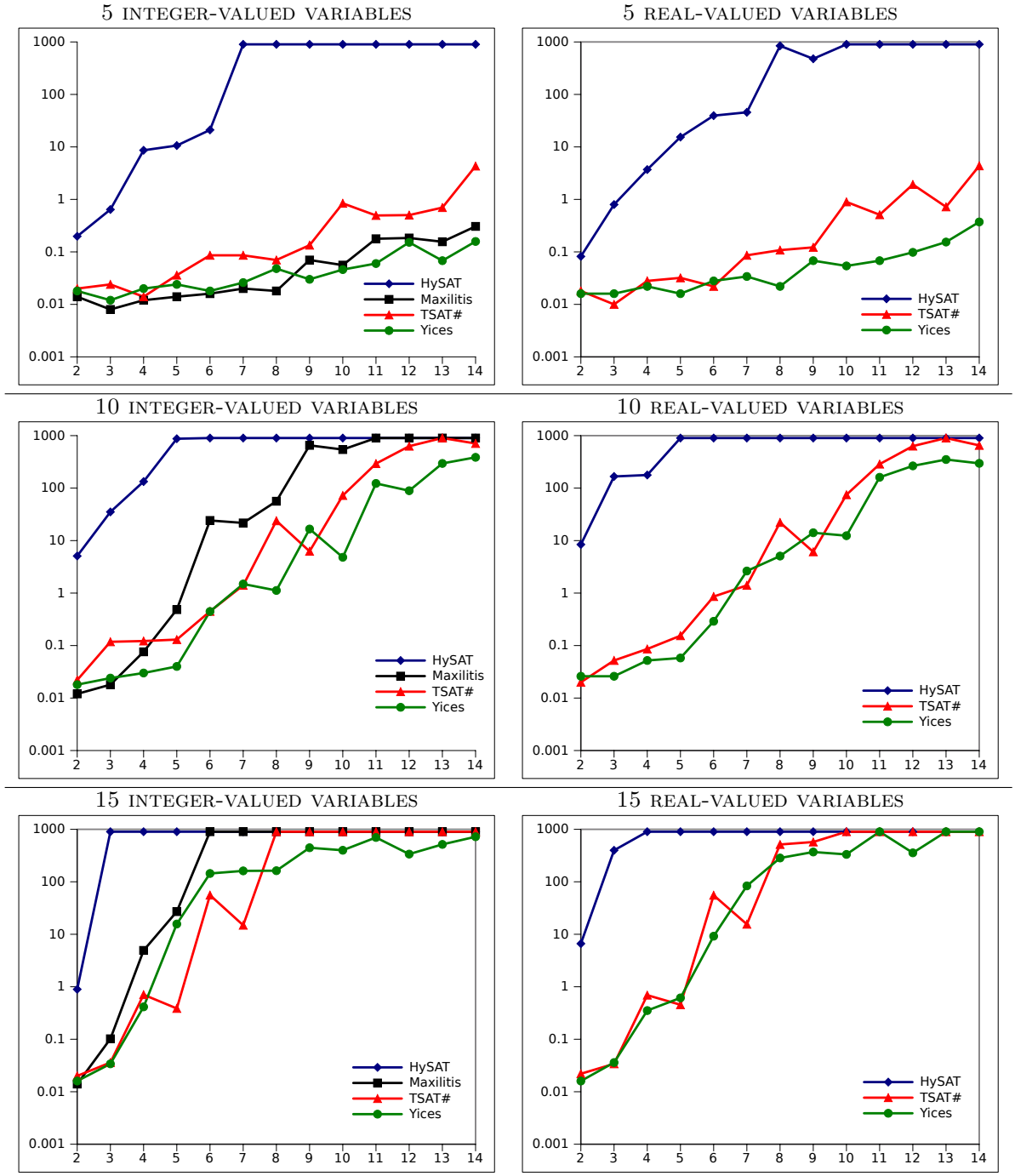

Fig. 1. Results of the evaluated solvers on random DTPs with preferences with $k = 2$. For each plot, in the x axis is shown the m/n ratio, while in the y axis (in logarithmic scale) the related median CPU time (in seconds). HySAT performance is depicted by blue diamonds, Maxilitis by using black boxes, while TSAT# and Yices results are denoted by red triangles and green circles, respectively. Plots in the left-most column are related to random DTPs with preferences encoded with integer-valued variables, while plots in the right-most column are related to random DTPs with preferences encoded with real-valued variables. Plots in the same row are related to random DTPs with preferences having the same value of n, i.e., plots in the top-most row reports the results for $n = 5$, while plots in second and third row report the results related to $n = 10$ and $n = 15$, respectively.

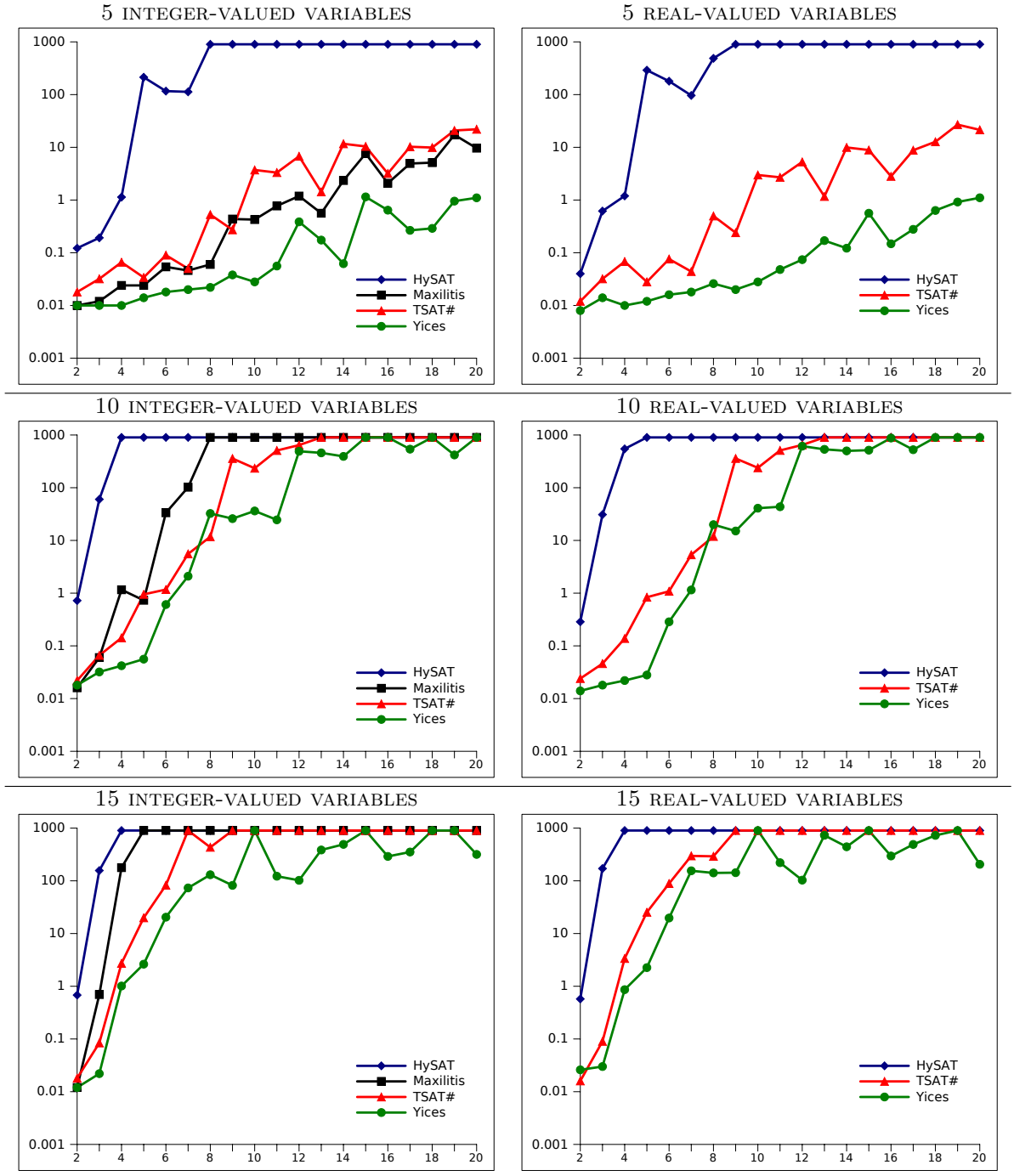

Fig. 2. Results of the evaluated solvers on random DTPs with preferences with $k = 3$. Plots are organized as in Figure 1.

In our next experiment we evaluated the solvers on random benchmarks having $k = 3$, and the results are depicted in Figure 2. Looking at the figure, and considering the results related to the benchmarks with integer-valued variables (left-most plots), we can see that YICES is confirmed to be the best solver. Considering $n = 5$ (top-left plot in Figure 2), we can see that it is faster than all other solvers along all the values of the m/n ratio. HYSAT reaches the CPU time limit

# VARS	m	HySAT				Maxilitis		TSAT#				Yices			
		INT-VALUED		REAL-VALUED		INT-VALUED		INT-VALUED		REAL-VALUED		INT-VALUED		REAL-VALUED	
		#	Time	#	Time	#	Time	#	Time	#	Time	#	Time	#	Time
5	10	10	1.36	10	1.80	10	0.11	10	0.16	10	0.20	10	0.18	10	0.18
	15	10	10.35	10	21.88	10	0.09	10	0.26	10	0.23	10	0.16	10	0.18
	20	10	45.44	10	75.85	10	0.10	10	0.20	10	0.21	10	0.18	10	0.24
	25	10	65.73	10	85.64	10	0.12	10	0.41	10	0.44	10	0.21	10	0.18
	30	10	1453.72	10	1793.20	10	0.13	10	0.76	10	0.76	10	0.26	10	0.24
	35	6	131.54	6	267.56	10	0.23	10	1.02	10	1.05	10	0.31	10	0.26
	40	5	1815.10	6	2125.40	10	0.30	10	1.41	10	1.40	10	0.33	10	0.31
	45	2	164.94	2	396.88	10	0.50	10	2.83	10	2.74	10	0.40	10	0.50
	50	1	35.46	1	60.20	10	0.62	10	3.15	10	3.29	10	0.52	10	0.49
	55	–	–	–	–	10	1.38	10	5.80	10	5.76	10	0.75	10	0.75
	60	–	–	–	–	10	1.75	10	12.56	10	11.72	10	0.96	10	1.09
	65	–	–	–	–	10	2.48	10	16.87	10	17.40	10	1.57	10	1.72
	70	–	–	–	–	10	3.37	10	17.75	10	17.52	10	2.00	10	2.15
10	20	10	13.99	10	42.14	10	0.15	10	0.21	10	0.20	10	0.18	10	0.19
	30	10	483.39	10	545.82	10	0.24	10	0.75	10	0.79	10	0.26	10	0.25
	40	6	1239.34	5	753.75	10	1.42	10	1.53	10	1.38	10	0.36	10	0.38
	50	2	1744.84	1	598.71	10	11.79	10	3.22	10	3.26	10	1.35	10	1.31
	60	–	–	–	–	10	73.24	10	6.15	10	6.26	10	4.68	10	4.84
	70	–	–	–	–	10	311.00	10	22.07	10	21.84	10	14.30	10	15.88
	80	–	–	–	–	10	1438.54	10	109.17	10	111.80	10	64.77	10	57.85
	90	–	–	–	–	6	1310.63	10	416.96	10	409.45	10	244.56	10	227.56
	100	–	–	–	–	2	671.17	9	353.62	9	370.92	10	600.00	10	520.23
	110	–	–	–	–	4	2932.86	10	1630.88	10	1646.45	10	1509.33	10	1408.65
	120	–	–	–	–	–	–	7	1471.99	7	1459.08	10	2513.06	10	2482.44
	130	–	–	–	–	–	–	4	1350.77	4	1284.93	10	2324.59	10	2493.96
	140	–	–	–	–	–	–	4	2251.60	4	2131.79	10	4136.29	9	3403.79
15	30	10	32.73	10	34.37	10	0.16	10	0.26	10	0.26	10	0.23	10	0.19
	45	6	202.63	7	1210.79	10	6.07	10	1.16	10	1.19	10	0.49	10	0.61
	60	–	–	–	–	10	200.91	10	5.98	10	6.21	10	2.76	10	2.84
	75	1	83.37	1	105.69	7	920.17	10	31.27	10	29.95	10	49.35	10	33.06
	90	–	–	–	–	3	1045.34	10	399.24	10	402.08	10	920.89	10	624.74
	105	–	–	–	–	1	489.08	9	553.22	9	566.27	10	1140.13	10	1213.47
	120	–	–	–	–	–	–	7	1951.63	7	1949.68	9	2458.48	10	3428.45
	135	–	–	–	–	–	–	2	475.80	2	465.05	10	3761.93	10	3629.82
	150	–	–	–	–	–	–	–	–	–	–	9	4344.51	8	2322.13
	165	–	–	–	–	–	–	1	248.94	1	249.05	6	1931.01	7	3291.59
	180	–	–	–	–	–	–	–	–	–	–	7	3146.96	7	2412.29
	195	–	–	–	–	–	–	–	–	–	–	6	2129.40	5	2405.48
	210	–	–	–	–	–	–	–	–	–	–	8	4759.54	8	4002.70

Table 1

Performance of the selected solvers on random DTPs with preferences with k = 2. The first columns reports the total number of variables for each group of DTPs (column "# VARS"), while the second one reports the number of constraints m. It is followed by four groups of columns, and the label is the solver name. Each group is composed by four columns – with the noticeable exception of the group related to Maxilitis – reporting the number of instances solved within the time limit ("#") and the total CPU time ("Time") spent on the solved formulas, both in the case of integer- and real-valued variables (groups "INT-VALUED" and "REAL-VALUED", respectively). In case a solver does not solve any instance, "–" is reported.

at $m/n = 7$, while performance of MAXILITIS and TSAT# are comparable, with the noticeable exception of $m/n = \{10, 11, 12\}$, for which we report that MAXILITIS performance is significantly better. Looking now at the results related to $n = 10$ (middle-left plot), HYSAT stops at $m/n = 3$, MAXILITIS at $m/n = 8$, while TSAT++ stops at $m/n = 12$. Notice that, with increasing values of the ratio, such benchmarks are hard for YICES too: it exhausts its CPU time for four values of m/n, namely 15, 16, 18, and 20. We also report that YICES is not the best solver along all benchmarks: for $m/n = 8$, TSAT# is the best one.

Looking now at the results related to benchmarks with real-valued variables (Figure 2 right-most plots), first we can see that the picture is almost the same as the one described in the case of integer-valued variable, with the noticeable exception that HYSAT performance stops at $m/n = 9$ instead to 8. Also for values of n equal to both 10 and 15 we can draw analogous conclusions to the integer-valued variables case.

Detailed results for the random benchmarks having $k = 3$ are reported in Table 2, which is organized similarly to Table 1.

Concerning non-random benchmarks, in our next experiment we first tested the solvers on the 40 smallest JOBSHOP benchmarks in the pool picked-up from the QF_IDL logic of SMT-LIB and we translated them – with minimal syntactical modifications – to the related solvers format. Table 3 shows the results of such analysis. Looking at the table, we can see that the picture obtained by the previous experiments is confirmed. YICES is the most efficient tool to solve Max-DTPs among the ones considered, followed by TSAT#. From the table, we can also see that MAXILITIS is confirmed to have good performances in the smallest instances: its performance is comparable with the ones obtained by both TSAT# and YICES. About the diamonds benchmarks, we report that the instances available having integer-valued variables (augmented with random weights to each difference constraints) are quite easy for all the tested systems – i.e., all of them are solved very easily. Thus, their results are not shown in detail.

5.2. Benchmarks with both hard and soft constraints

Problems with only soft DTP constraints is largely the most evaluated case in the literature of DTPs with preferences. However, it is also of interest to evaluate DTPs with both hard and soft constraints, generated as described in Section 4. The analysis is performed with YICES, given it clearly showed best performance considering the results in the previous subsection.

All the experiments here presented ran on PCs equipped with a processor Intel Core 2 Duo running at 2.20 GHz, with 2 GB of RAM, and running GNU Linux Ubuntu 3.0.0. Time and memory limits are the same used in the experiments described in Section 5.1.

Here we present only the results obtained with random DTPs having 15 integer-valued variables: all plots and tables related to the remaining configurations can be found in the web appendix available at `http://www.star.dist.unige.it/~marco/Tsat/appendix-aicom-tsat.pdf`.

In Figure 3 we can see the results related to the performance of YICES on random DTPs with 15 integer-valued variables with both hard and soft constraints. In general, we can see that increasing the percentage of hard constraints in the formulas leads to easier formulas: this seems to be due, at least partly, to the fact that as the percentage increases, "easy" unsatisfiable formulas are introduced. Looking more in details at the figure, and considering the results related to the benchmarks with $k = 2$ (left-most plot), we can see that the performance of YICES on benchmarks with no hard constraints is always at least one order of magnitude slower with respect to the performance reported on the same benchmarks with hard constraints – with the noticeable exception of benchmarks with 25% of hard constraints for $m/n = \{6, 10, 13, 14\}$.

Considering now the results related to $k = 3$ – right-most plot on Figure 3 – we can see a slightly different picture. YICES is able to solve all benchmarks only in the case they have 75% of hard constraints, which is again the simplest setting. We can also note that random DTPs with soft constraints only are not anymore the most difficult ones. The results of our experiments show that, in this case, random DTPs with 25% of hard constraints are not solved for $m/n \geq 10$, while YICES does not solve random DTPs with 50% of hard constraints for $m/n \geq 12$ – with the noticeable exception of $m/n = 15$ and $m/n = 20$. One difference wrt the case with $k = 2$ is that here there is not anymore a relevant number of "easy" unsat-

# VARS	m	HySAT				Maxilitis		TSAT#				Yices			
		INT-VALUED		REAL-VALUED		INT-VALUED		INT-VALUED		REAL-VALUED		INT-VALUED		REAL-VALUED	
		#	Time	#	Time	#	Time	#	Time	#	Time	#	Time	#	Time
5	10	10	1.55	10	0.81	10	0.16	10	0.16	10	0.14	10	0.10	10	0.09
	15	10	19.58	10	14.39	10	0.17	10	0.28	10	0.26	10	0.10	10	0.11
	20	10	44.45	10	44.47	10	0.21	10	0.36	10	0.33	10	0.10	10	0.13
	25	10	745.05	10	757.07	10	0.25	10	0.78	10	0.75	10	0.13	10	0.14
	30	9	1371.35	9	906.93	10	0.39	10	1.02	10	0.96	10	0.16	10	0.17
	35	8	1267.36	8	1254.11	10	0.56	10	1.73	10	1.56	10	0.18	10	0.17
	40	3	269.49	3	205.75	10	1.64	10	3.72	10	3.21	10	0.24	10	0.24
	45	–	–	1	678.14	10	2.87	10	7.64	10	6.32	10	0.40	10	0.41
	50	–	–	–	–	10	4.58	10	12.46	10	10.24	10	0.49	10	0.49
	55	–	–	–	–	10	7.76	10	22.51	10	18.06	10	1.90	10	2.08
	60	–	–	–	–	10	11.26	10	33.35	10	27.93	10	1.52	10	1.63
	65	–	–	–	–	10	16.24	10	137.86	10	116.57	10	3.10	10	2.92
	70	–	–	–	–	10	23.23	10	114.07	10	99.11	10	2.28	10	2.38
	75	–	–	–	–	10	42.13	10	208.16	10	194.15	10	4.63	10	4.88
	80	–	–	–	–	10	37.08	10	157.06	10	147.05	10	3.80	10	4.03
	85	–	–	–	–	10	65.19	10	426.42	10	439.18	10	5.74	10	6.00
	90	–	–	–	–	10	67.53	10	188.84	10	236.76	10	5.09	10	5.10
	95	–	–	–	–	10	105.58	10	377.46	10	475.11	10	9.10	10	8.14
	100	–	–	–	–	10	127.78	10	1343.67	10	1398.79	10	9.67	10	9.73
10	20	10	17.68	10	7.36	10	0.14	10	0.32	10	0.30	10	0.21	10	0.14
	30	10	454.36	10	197.49	10	0.38	10	0.80	10	0.62	10	0.26	10	0.17
	40	6	2335.41	6	1624.25	10	6.24	10	2.66	10	2.16	10	0.56	10	0.27
	50	2	426.78	3	1092.40	10	29.47	10	7.25	10	6.60	10	1.43	10	0.57
	60	–	–	1	894.44	10	228.32	10	14.98	10	13.59	10	3.48	10	1.47
	70	–	–	–	–	10	1221.70	10	80.84	10	68.60	10	18.41	10	10.13
	80	–	–	–	–	6	1924.68	10	578.70	10	578.82	10	128.38	10	78.01
	90	–	–	–	–	2	1070.29	10	2023.55	10	2018.22	10	213.21	10	152.49
	100	–	–	–	–	2	1061.79	9	1052.45	9	1073.69	10	359.95	10	495.64
	110	–	–	–	–	–	–	7	1902.56	7	1917.36	10	1811.45	10	1400.65
	120	–	–	–	–	–	–	6	1436.90	6	1457.58	10	2420.55	10	2914.64
	130	–	–	–	–	–	–	–	–	–	–	9	4347.10	9	3932.29
	140	–	–	–	–	–	–	–	–	–	–	8	3580.30	10	5573.39
	150	–	–	–	–	–	–	–	–	–	–	8	3366.58	9	4881.64
	160	–	–	–	–	–	–	–	–	–	–	7	3626.04	7	3789.90
	170	–	–	–	–	–	–	–	–	–	–	8	3767.18	10	5272.14
	180	–	–	–	–	–	–	1	0.02	1	0.03	7	3418.81	4	1822.88
	190	–	–	–	–	–	–	–	–	–	–	4	1880.34	4	2374.17
	200	–	–	–	–	–	–	–	–	–	–	4	1788.35	5	2112.90
15	30	10	52.84	10	32.87	10	0.16	10	0.30	10	0.31	10	0.16	10	0.20
	45	9	2042.86	8	1841.60	10	8.75	10	1.18	10	1.29	10	0.24	10	0.28
	60	1	145.27	1	127.76	10	526.96	10	11.56	10	13.76	10	3.00	10	2.72
	75	–	–	–	–	6	1078.61	10	70.34	10	89.63	10	12.17	10	10.16
	90	–	–	–	–	2	1394.12	10	823.85	10	868.09	10	220.44	10	194.18
	105	–	–	–	–	–	–	5	1442.68	8	2872.92	10	1235.27	10	1347.94
	120	–	–	–	–	–	–	2	864.75	4	2191.27	10	1993.83	10	1966.26
	135	–	–	–	–	–	–	1	647.31	1	354.63	10	1756.30	10	1737.47
	150	–	–	–	–	–	–	–	–	–	–	7	1274.67	7	2290.38
	165	–	–	–	–	–	–	–	–	–	–	8	1832.83	8	2471.95
	180	–	–	–	–	–	–	–	–	–	–	10	2085.46	10	2018.39
	195	–	–	–	–	–	–	–	–	–	–	9	3825.19	5	2251.18
	210	–	–	–	–	–	–	–	–	–	–	8	3222.01	8	2347.85
	225	–	–	–	–	–	–	–	–	–	–	7	2945.63	9	3358.87
	240	–	–	–	–	–	–	–	–	–	–	9	3086.62	9	3053.98
	255	–	–	–	–	–	–	–	–	–	–	5	1968.85	6	2554.17
	270	–	–	–	–	–	–	–	–	–	–	7	2592.21	8	3064.55
	285	–	–	–	–	–	–	–	–	–	–	8	2355.08	7	2440.89
	300	–	–	–	–	–	–	–	–	–	–	8	2484.74	8	2064.53

Table 2
Performance of the selected solvers on random DTPs with preferences with k = 3. The table is organized as Table 1.

isfiable formulas generated, and the very few generated (see, e.g., the very last number in the last row of Table 5) are not that easy.

Detailed results for random benchmarks with 15 integer-valued variables having both hard and soft constraints are reported in Tables 4 and 5. For each setting, the number of solved instances (out of 10) with Yices and the sum of their CPU times

Instance	N	HySAT		Maxilitis		TSAT#		Yices	
		#	Time	#	Time	#	Time	#	Time
jobshop2*	4	4	0.17	4	0.03	4	0.05	4	0.04
jobshop4*	4	4	604.70	4	0.13	4	0.25	4	0.07
jobshop6*	4	1	704.94	3	618.15	4	5.76	4	0.20
jobshop8*	4	–	–	–	–	4	597.19	4	41.55
jobshop10*	4	–	–	–	–	3	1262.99	4	722.02
jobshop12*	4	–	–	–	–	–	–	4	989.49
jobshop14*	4	–	–	–	–	–	–	4	1047.87
jobshop16*	4	–	–	–	–	–	–	4	1292.09
jobshop18*	4	–	–	–	–	–	–	4	1623.10
jobshop20*	4	–	–	–	–	–	–	4	1910.31

Table 3

Performance of the selected solvers on a pool of JOBSHOP benchmarks. The table consists of ten columns: the first one reports the group of benchmarks (column "Instances"), while the second one reports the total number of formulas within the group (column "N"). It is followed by four groups of columns, and the label is the solver name. Each group is composed by two columns, reporting the number of instances solved within the time limit ("#") and the total CPU time ("Time") spent on the solved formulas. In case a solver reaches the time limit on all instances, "–" is reported.

is reported. Moreover, a separate analysis taking account unsatisfiable formulas is also shown.

Our last experiment concerns the JOBSHOP benchmarks presented in Section 4. In Table 6 we report the results of the experiment. Looking at the table, we notice that the general pattern identified for random benchmarks is maintained, with some exceptions on the bigger instances.

6. Related Work

Maxilitis [MP05,Mof11], WeightWatcher [MP06] and ARIO [SPSP05] implement different approaches for solving DTP with preferences as defined in [PP04]. Maxilitis is a direct extension of the DTP solver Epilitis [TP03], while WeightWatcher uses an approach based on Weighted Constraints Satisfaction problems, even if the two methods are similar (as mentioned in, e.g., [MP06]). ARIO, instead, relies on an approach based on Mixed Logical Linear Programming (MLLP) problems. Such systems deal with more complex preferences than the one we deal with in this paper. In our analysis we have used Maxilitis because the results in, e.g., [Mof11] clearly indicates its superior performance. As we already said, the extension for DTPs defined in [PP04] is different wrt the one in this paper: in [PP04] preferences are not limited to be constants. Moreover, in this paper we have tested the version of Maxilitis that uses a branch and bound approach for reaching the optimal solution, called Maxilitis-BB in [Mof11], and that on our benchmarks is consistently better than Maxilitis-IW, a further version of Maxilitis (IW staying for Iterative Weakening) that searches for solutions with a progressively increasing number of violated constraints. formulation is that we can

About the relation of our approach wrt Maxilitis and WeightWatcher, we would like to notice first that we do not deal with a subset of the problems they can deal with given that, e.g., we can solve problems with real-world variables. Moreover, our approach has a number of advantages: first, it can be easily extended to deal with generic Boolean combination of difference constraints, where a DTP constraints is a set of constraint literals (all our back-end solvers can deal with these problems), while the solvers defined above are tailored for conjunction of disjunctions of difference constraints. Then, we can take easily advantage from the availability of (more) efficient solvers that can solve Max-DTPs, by adding to our back-end a new solver, or updating an already existing solver with a new version. Last, our framework seems to be more flexible to include same further extensions, e.g., to allow for both hard and soft difference constraints in DTP constraints, with the related update in the definition. Maxilitis, however, allows for soft difference constraints only.

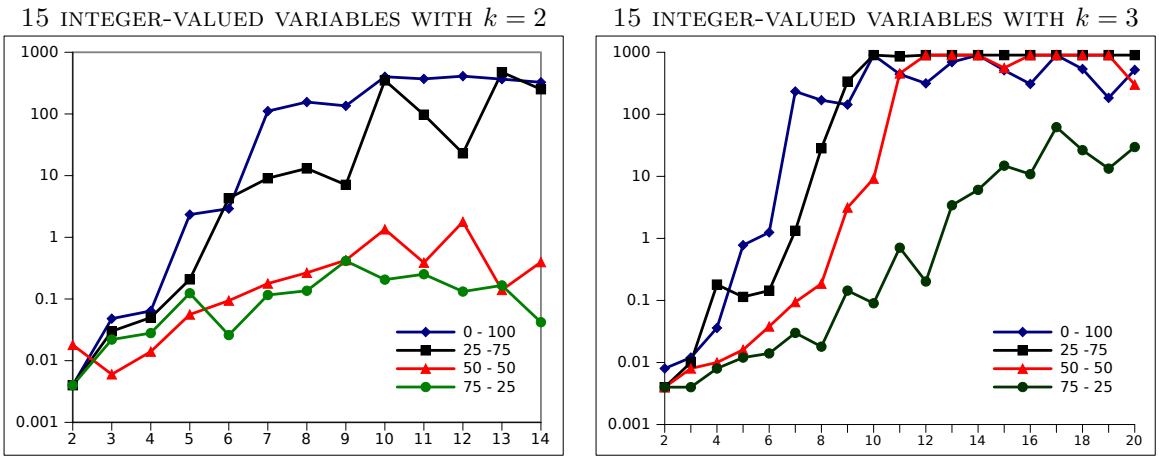

Fig. 3. Results of YICES on random DTPs – 15 integer-valued variables – with both hard and soft constraints. For each plot, in the x axis is shown the m/n ratio, while in the y axis (in logarithmic scale) the related median CPU time (in seconds). YICES performance on benchmarks without hard constraints ("0–100") is depicted by blue diamonds. Performance of YICES on benchmarks having 25% of hard constraints is depicted by using black boxes ("25–75"), while performance on benchmarks having an equal portion of hard and soft constraint is denoted by red triangles ("50–50"). Finally, green circles denotes performance of YICES on benchmarks having 75% of hard constraints ("75–25"). The left-most plot shows the results on benchmarks with $k = 2$, while in the right-most plot we report the results related to $k = 3$.

m	0–100		25–75						50–50						75–25					
			TOTAL		OPT		UNSAT		TOTAL		OPT		UNSAT		TOTAL		OPT		UNSAT	
	#	Time	#	Time	#	Time	#	Time	#	Time	#	Time	#	Time	#	Time	#	Time	#	Time
30	10	0.04	10	0.04	10	0.04	–	–	10	0.10	10	0.10	0	0.00	10	0.08	10	0.08	–	–
45	10	0.21	10	0.17	10	0.17	–	–	10	0.05	10	0.05	0	0.00	10	0.12	10	0.12	–	–
60	10	1.27	10	0.38	10	0.38	–	–	10	0.22	10	0.22	0	0.00	10	0.15	10	0.15	–	–
75	10	26.65	10	2.40	10	2.40	–	–	10	0.76	10	0.76	0	0.00	10	0.50	10	0.50	–	–
90	10	666.66	10	22.45	10	22.45	–	–	10	1.48	10	1.48	0	0.00	10	0.91	7	0.41	3	0.50
105	10	757.33	10	51.23	10	51.23	–	–	10	5.57	10	5.57	0	0.00	10	1.64	6	0.59	4	1.04
120	10	2385.65	10	494.38	10	494.38	–	–	10	5.18	9	5.06	1	0.11	10	1.66	4	0.83	6	0.83
135	10	2224.69	10	1461.37	10	1461.37	–	–	10	7.99	8	5.87	2	2.12	10	3.02	1	0.12	9	2.89
150	10	3368.71	10	1486.14	10	1486.14	–	–	10	7.87	9	7.79	1	0.08	10	1.46	–	–	10	1.46
165	9	3345.03	9	1885.58	9	1885.58	–	–	10	7.53	7	6.76	3	0.77	10	1.46	–	–	10	1.46
180	10	3847.37	8	1627.48	8	1627.48	–	–	10	9.72	4	5.84	6	3.87	10	1.88	–	–	10	1.88
195	7	2046.42	8	735.21	8	735.21	–	–	10	3.30	3	1.12	7	2.18	10	1.98	–	–	10	1.98
210	9	3373.26	8	1128.82	8	1128.82	–	–	10	8.90	1	0.43	9	8.47	10	1.22	–	–	10	1.22

Table 4

Results of YICES on random DTPs with both hard and soft constraints, having $k = 2$ and 15 integer-valued variables. The first column reports the number of constraints m, and it is followed by four group of columns. The first one reports the performance of YICES on benchmarks without hard constraints ("0–100"), while the second one ("25–75") reports performance of YICES on benchmarks having 25% of hard constraints. Last two groups report performance of YICES on benchmarks having an equal portion of hard and soft constraint, and on benchmarks having 75% of hard constraints ("75–25"), respectively. Considering last three groups, for each one we report the global performance (group "TOTAL"), and we detail the performance of YICES both when an optimum has been found (group "OPT") and when the instance is unsatisfiable (group "UNSAT"). As in Tables 1 and 2, columns "#" and "Time" report the number of instances solved within the time limit and the total CPU time, respectively. In case a solver reaches the time limit on all instances, "–" is reported.

About the random benchmarks set, note that the DTP is already a "difficult" problem, and the

m	0–100		25–75						50–50						75–25					
			TOTAL		OPT		UNSAT		TOTAL		OPT		UNSAT		TOTAL		OPT		UNSAT	
	#	Time	#	Time	#	Time	#	Time	#	Time	#	Time	#	Time	#	Time	#	Time	#	Time
30	10	0.08	10	0.04	10	0.04	–	–	10	0.04	10	0.04	–	–	10	0.04	10	0.04	–	–
45	10	0.17	10	0.09	10	0.09	–	–	10	0.07	10	0.07	–	–	10	0.06	10	0.06	–	–
60	10	3.33	10	0.56	10	0.56	–	–	10	0.11	10	0.11	–	–	10	0.10	10	0.10	–	–
75	10	24.90	10	0.94	10	0.94	–	–	10	0.19	10	0.19	–	–	10	0.11	10	0.11	–	–
90	10	240.53	10	3.92	10	3.92	–	–	10	0.45	10	0.45	–	–	10	0.15	10	0.15	–	–
105	10	1246.08	10	95.22	10	95.22	–	–	10	1.85	10	1.85	–	–	10	0.23	10	0.23	–	–
120	10	2165.72	10	986.05	10	986.05	–	–	10	7.59	10	7.59	–	–	10	0.31	10	0.31	–	–
135	10	1977.29	8	2603.17	8	2603.17	–	–	10	125.10	10	125.10	–	–	10	1.50	10	1.50	–	–
150	7	1366.11	3	945.66	3	945.66	–	–	10	933.56	10	933.56	–	–	10	1.53	10	1.53	–	–
165	8	1982.32	3	1919.72	3	1919.72	–	–	9	2167.42	9	2167.42	–	–	10	11.63	10	11.63	–	–
180	10	2299.02	3	1653.00	3	1653.00	–	–	7	2392.98	7	2392.98	–	–	10	23.06	10	23.06	–	–
195	9	4117.52	1	626.02	1	626.02	–	–	3	1191.33	3	1191.33	–	–	10	43.57	10	43.57	–	–
210	8	3368.95	1	600.03	1	600.03	–	–	–	–	–	–	–	–	10	105.56	10	105.56	–	–
225	7	3090.40	2	1526.14	2	1526.14	–	–	4	1692.46	4	1692.46	–	–	10	165.96	10	165.96	–	–
240	9	3537.82	1	731.47	1	731.47	–	–	3	1324.35	3	1324.35	–	–	10	176.86	10	176.86	–	–
255	5	2099.09	–	–	–	–	–	–	4	1274.11	4	1274.11	–	–	10	236.21	10	236.21	–	–
270	7	2694.83	–	–	–	–	–	–	5	1868.65	5	1868.65	–	–	10	280.25	10	280.25	–	–
285	8	2461.61	–	–	–	–	–	–	8	2006.49	8	2006.49	–	–	10	152.43	10	152.43	–	–
300	8	2609.72	–	–	–	–	–	–	6	1669.38	6	1669.38	–	–	10	506.92	8	197.07	2	309.85

Table 5

Results of YICES on random DTPs with both hard and soft constraints, having $k = 3$ and 15 integer-valued variables. The table is organized as Table 4.

Instance	0–100		25–75		50–50		75–25	
	#	Time	#	Time	#	Time	#	Time
`jobshop2*`	4	0.01	4	0.01	4	0.00	4	0.00
`jobshop4*`	4	0.04	4	0.02	4	0.02	4	0.03
`jobshop6*`	4	0.16	4	0.16	4	0.08	4	0.27
`jobshop8*`	4	0.23	4	0.37	4	0.20	4	4.22
`jobshop10*`	4	324.69	4	596.45	4	5.94	4	0.51
`jobshop12*`	4	1480.24	4	1865.46	4	576.43	4	22.17
`jobshop14*`	4	962.49	4	1512.17	4	1752.04	4	5.46
`jobshop16*`	3	897.51	4	1407.23	4	1249.45	3	1.68
`jobshop18*`	4	1445.48	4	1339.54	1	346.75	4	691.39
`jobshop20*`	4	1620.85	4	1506.04	3	1453.52	3	30.75

Table 6

Results of YICES on a pool of JOBSHOP benchmarks with both hard and soft constraints. The table consists of nine columns: the first one reports the group of benchmarks (column "Instance"), and it is followed by four groups of columns, organized similarly to Table 4.

analysis in literature on DTPs has been performed on problems with few tens of variables for the setting used in this paper: adding preferences further increase the difficulty. An exception to this trend is [NK08], where the CIRCUITTSAT solver is presented. CIRCUITTSAT is based on a circuit-based representation of the problem in CNF, and on solving with SAT solvers. In [NK08] DTPs with many variables n are used, but the analysis in focused on problems with $k > 2$ having ratios m/n such that

the majority of the instances are satisfiable. On these instances, CircuitTSAT performs faster than competing solvers (i.e., TSAT# and Yices without optimizations). CircuitTSAT does not deal with preferences, and relies on approximation when dealing with real-valued variables.

In the following, we describe in more details the back-end solvers we evaluated in this paper.

TSAT# [MPP11] is an extension of the DTP solver TSAT++ to deal with Max-DTPs. It implements a classical generate and test approach, but the generation of candidate solutions is performed with Boolean optimization (i.e., Max-SAT, PB and ASP) solvers, instead of SAT solvers, to take into account constraints weights. An advantage of TSAT# is that it can fruitfully relying on continuous improvements in the performance of its back-end solvers, thanks to Max-SAT, PB, and ASP competitions[7] (see, e.g., [ALMP08,CIR+11] for recent reports). The solvers we tested in our analysis are: minisat+ ver. 1.14 [ES06], MiniMaxSat ver. 1.0 [HLO08], IncWMaxSatz [LS07,LSL08], PBclasp ver. 1.0 and akmaxsat [Kug10], the version submitted to the last Max-SAT 2010 Competition.[8] minisat+ solves the problem by compilation to a SAT problem, and iteratively calling the SAT solver minisat [ES03] with improved solution until an unsatisfiable formula is found. MiniMaxSat is built on top of minisat+ ver 1.13, with theory specific pre-processing techniques. IncWMaxSatz extends the WMaxSatz [LMMP09] and the MaxSatz [LS07] Max-SAT solvers. akmaxsat is based on an algorithm with improved detection of disjoint inconsistent sub-formulas, while PBclasp compiles a PB problem into an ASP problem and calls the ASP solver clasp [GKNS07] ver. 1.3.6 .

The HySAT solver [FHT+07] core algorithm is composed by a tight integration between the DPLL algorithm [DP60,DLL62] and Interval Constraints Propagation [Dav87] in order to mainly deal with non-linear arithmetic constraints, also involving transcendental functions. It can also be seen as an SMT solver mainly designed for problems falling in the QF_NRA logic of SMT competitions (SMT-COMP)[9] involving non-linear constraints.

About SMT solvers, Barcelogic [NO06,NOT06], Yices [DdM06] and Z3 [dMB08] can solve problems involving Boolean combinations of difference constraints, in conjunction with maximum satisfiability. They all feature the lazy SAT-based approach [Seb07]. We have only used Yices in our analysis because Z3 is limited to solve Partial Max-DTPs, and the version of Barcelogic that deals with Max-DTPs is currently not available.[10]

7. Conclusions and Future Work

In this paper we have presented a new approach for solving DTPs with constant preferences expressed on different constraints whose aggregation among DTP constraints considers an utilitarian method, and the aggregation within DTP constraints follows the max semantic. Our approach solves the problem by reducing it to a weighted maximum satisfiability of DTPs, we dealt with in [MPP11]. The reduction has been implemented, allowing for a wide set of off-the-shelf systems that can deal with Max-DTPs to be used as back-end. Our experimental analysis, conducted on both randomly generated and real-world problems, shows that the Yices SMT solver is the best back-end solver among those analyzed, followed by TSAT#, and that with both systems benchmarks can be solved orders of magnitude faster than with Maxilitis, the state-of-the-art systems for solving DTPs with preferences.

The future work includes extending our approach to deal with more complex forms of preferences, e.g., the one outlined in the previous section about DTP constraints having both hard and soft difference constraints, and/or the one defined in [PP04].

[7] See, e.g., `http://www.maxsat.udl.cat/`, `http://www.cril.univ-artois.fr/PB10/` and `https://www.mat.unical.it/aspcomp2011/` for references to related Evaluations and Competitions.

[8] See `http://www.lsi.upc.edu/~fheras/docs/m.tar.gz`, `http://minisat.se/MiniSat+.html`,`http://www.uni-ulm.de/fileadmin/website_uni_ulm/iui.inst.190/Mitarbeiter/kuegel/testdata.tgz`, `http://www.cs.sysu.edu.cn/~lh/IncWMaxSatz.zip` and `http://potassco.svn.sourceforge.net/viewvc/potassco/trunk/pbclasp.tar.gz`.

[9] `http://www.smtcomp.org/` and see, e.g. [BDOS08].

[10] Personal communications with Albert Oliveras.

Acknowledgments. 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 authors would like to thank the anonymous reviewers of RCRA 2011 for comments and criticisms on the early version of the paper, Enrico Giunchiglia for fruitful discussions on the topic of the paper, and Maurizio Pianfetti for the implementation of TSAT#. We would also like to thank Adrian Kügel, Michael D. Moffitt, Bart Peintner, Christian Herde, Bruno Dutertre and Albert Oliveras for discussions about, and/or getting support to, their solvers, and Hyondeuk Kim for details about the JOBSHOP benchmarks.

References

[ACG99] 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.

[ACG+05] Alessandro Armando, Claudio Castellini, Enrico Giunchiglia, Massimo Idini, and Marco Maratea. TSAT++: An open platform for satisfiability modulo theories. *Electronic Notes in Theoretical Compututer Science*, 125(3):25–36, 2005.

[ALMP08] Josep Argelich, Chu Min Li, Felip Manyà, and Jordi Planes. The first and second Max-SAT evaluations. *Journal of Satisfiability, Boolean Modelling and Computation*, 4(2-4):251–278, 2008.

[BBC+05] 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.

[BDOS08] Clark Barrett, Morgan Deters, Albert Oliveras, and Aaron Stump. Design and results of the 3rd annual Satisfiability Modulo Theories competition (SMT-COMP 2007). *International Journal on Artificial Intelligence Tools*, 17(4):569–606, 2008.

[BGPYS07] Pauline M. Berry, Melinda T. Gervasio, Bart Peintner, and Neil Yorke-Smith. A preference model for over-constrained meeting requests. In *Proc. of the AAAI 2007 Workshop on Preference Handling for Artificial Intelligence*, pages 7–14, 2007.

[BGPYS11] Pauline M. Berry, Melinda T. Gervasio, Bart Peintner, and Neil Yorke-Smith. PTIME: Personalized assistance for calendaring. *ACM Transaction on Intelligent Systems and Technology*, 2(4):40, 2011.

[CIR+11] Francesco Calimeri, Giovambattista Ianni, Francesco Ricca, Mario Alviano, Annamaria Bria, Gelsomina Catalano, Susanna Cozza, Wolfgang Faber, Onofrio Febbraro, Nicola Leone, Marco Manna, Alessandra Martello, Claudio Panetta, Simona Perri, Kristian Reale, Maria Carmela Santoro, Marco Sirianni, Giorgio Terracina, and Pierfrancesco Veltri. The third answer set programming competition: Preliminary report of the system competition track. In James P. Delgrande and Wolfgang Faber, editors, *Proc. of the 11th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR 2011)*, volume 6645 of *Lecture Notes in Computer Science*, pages 388–403. Springer, 2011.

[Dav87] Ernest Davis. Constraint propagation with interval labels. *Artificial Intelligence*, 32(3):281–331, 1987.

[DdM06] Bruno Dutertre and Leonardo de Moura. The Yices SMT solver. Technical report, Stanford Research Institute, 2006.

[DLL62] Martin Davis, George Logemann, and Donald Loveland. A machine program for theorem proving. *Communications of the ACM*, 5(7):394–397, 1962.

[dM] Leonardo de Moura. `http://yices.csl.sri.com/`.

[DM06] Bruno Dutertre and Leonardo De Moura. A fast linear-arithmetic solver for DPLL (T). In Thomas Ball and Robert B. Jones, editors, *Proc. of the 18th International Conference on Computer Aided Verification (CAV 2006)*, volume 4144 of *Lecture Notes in Computer Science*, pages 81–94. Springer, 2006.

[dMB08] Leonardo de Moura and Nikolaj Bjørner. Z3: An efficient SMT solver. In C. R. Ramakrishnan and Jakob Rehof, editors, *Proc. of the 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2008)*, volume 4963 of *Lecture Notes in Computer Science*, pages 337–340. Springer, 2008.

[DMP91] Rina Dechter, Itay Meiri, and Judea Pearl. Temporal constraint networks. *Artificial Intelligence*, 49(1-3):61–95, 1991.

[DP60] Martin Davis and Hilary Putnam. A computing procedure for quantification theory. *Journal of the ACM*, 7:201–215, 1960.

[ES03] Niklas Eén and Niklas Sörensson. An extensible SAT-solver. In Enrico Giunchiglia and Ar-

mando Tacchella, editors, *Proc. of the 6th International Conference on the Theory and Applications of Satisfiability Testing (SAT 2003)*, volume 2919 of *Lecture Notes in Computer Science*, pages 502–518. Springer, 2003.

[ES06] Niklas Eén and Niklas Sörensson. Translating pseudo-Boolean constraints into SAT. *Journal on Satisfiability, Boolean Modeling and Computation*, 2:1–26, 2006.

[FHT+07] Martin Fränzle, Christian Herde, Tino Teige, Stefan Ratschan, and Tobias Schubert. Efficient solving of large non-linear arithmetic constraint systems with complex Boolean structure. *Journal on Satisfiability, Boolean Modeling and Computation*, 1:209–236, 2007.

[GKNS07] Martin Gebser, Benjamin Kaufmann, André Neumann, and Torsten Schaub. Conflict-driven answer set solving. In Manuela M. Veloso, editor, *Proc. of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007)*, pages 386–392. Morgan Kaufmann Publishers, 2007.

[GL88] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In Robert Kowalski and Kenneth Bowen, editors, *Proc. of the 5th International Conference on Logic Programming (ICLP/SLP 1988)*, pages 1070–1080. MIT Press, 1988.

[GL91] Michael Gelfond and Vladimir Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9:365–385, 1991.

[GMT03] Enrico Giunchiglia, Marco Maratea, and Armando Tacchella. (In)Effectiveness of look-ahead techniques in a modern SAT solver. In Francesca Rossi, editor, *Proc. of the 9th International Conference on Principles and Practice of Constraint Programming (CP 2003)*, volume 2833 of *Lecture Notes in Computer Science*, pages 842–846. Springer, 2003.

[HLO08] Federico Heras, Javier Larrosa, and Albert Oliveras. MiniMaxSAT: A new weighted Max-SAT solver. *Journal of Artificial Intelligence Research*, 31:1–32, 2008.

[KMMV03] Lina Khatib, Paul H. Morris, Robert A. Morris, and Kristen Brent Venable. Tractable pareto optimization of temporal preferences. In Georg Gottlob and Toby Walsh, editors, *Proc. of the 18th International Joint Conference on Artificial Intelligence (IJCAI 2003)*, pages 1289–1294. Morgan Kaufmann, 2003.

[Kug10] Adrien Kuger. Improved exact solver for the weighted Max-SAT problem. In *Proc. of the 2nd Pragmatics of SAT (PS-10) workshop*, 2010.

[Kum04] T. K. Satish Kumar. A polynomial-time algorithm for simple temporal problems with piecewise constant domain preference functions. In Deborah L. McGuinness and George Ferguson, editors, *Proc. of the 19th National Conference on Artificial Intelligence (AAAI 2004)*, pages 67–72. AAAI Press / The MIT Press, 2004.

[LMMP09] Chu Min Li, Felip Manyà, Nouredine Ould Mohamedou, and Jordi Planes. Exploiting cycle structures in Max-SAT. In Oliver Kullmann, editor, *Proc. of the 12th International Conference on Theory and Applications of Satisfiability Testing (SAT 2009)*, volume 5584 of *Lecture Notes in Computer Science*, pages 467–480. Springer, 2009.

[LS07] Han Lin and Kaile Su. Exploiting inference rules to compute lower bounds for Max-SAT solving. In Manuela M. Veloso, editor, *Proc. of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007)*, pages 2334–2339, 2007.

[LSL08] Han Lin, Kaile Su, and Chu Min Li. Within-problem learning for efficient lower bound computation in Max-SAT solving. In Dieter Fox and Carla P. Gomes, editors, *Proc. of the 23rd AAAI Conference on Artificial Intelligence (AAAI 2008)*, pages 351–356. AAAI Press, 2008.

[MMK+04] Paul H. Morris, Robert A. Morris, Lina Khatib, Sailesh Ramakrishnan, and Andrew Bachmann. Strategies for global optimization of temporal preferences. In Mark Wallace, editor, *Proc. of the 10th International Conference on Principles and Practice of Constraint Programming (CP 2004)*, volume 3258 of *Lecture Notes in Computer Science*, pages 408–422. Springer, 2004.

[Mof11] Michael D. Moffitt. On the modelling and optimization of preferences in constraint-based temporal reasoning. *Artificial Intelligence*, 175(7-8):1390–1409, 2011.

[MP05] 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.

[MP06] 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.

[MPP11] Marco Maratea, Maurizio Pianfetti, and Luca Pulina. Maximum satisfiability of disjunctive temporal problems with Boolean optimization. In Marco Gavanelli and Toni Mancini, editors, *Proc. of the 18th RCRA International Workshop on "Experimental Evaluation of Algorithms for Solving Problems with Combinatorial Explosion" (RCRA 2011)*, 2011.

[MVSO04] Pragnesh Jay Modi, Manuela M. Veloso, Stephen F. Smith, and Jean Oh. CMRadar: A

personal assistant agent for calendar management. In Deborah L. McGuinness and George Ferguson, editors, *Proc. of the 19th National Conference on Artificial Intelligence (AAAI 2004)*, pages 1020–1021. AAAI Press / The MIT Press, 2004.

[NK08] Blaine Nelson and T. K. Satish Kumar. CircuitTSAT: A solver for large instances of the disjunctive temporal problem. In Jussi Rintanen, Bernhard Nebel, J. Christopher Beck, and Eric A. Hansen, editors, *Proc. of the 18th International Conference on Automated Planning and Scheduling (ICAPS 2008)*, pages 232–239. AAAI Press, 2008.

[NO06] Robert Nieuwenhuis and Albert Oliveras. On SAT modulo theories and optimization problems. In Armin Biere and Carla P. Gomes, editors, *Proc. 9th International Conference on Theory and Applications of Satisfiability Testing (SAT 2006)*, volume 4121 of *Lecture Notes in Computer Science*, pages 156–169. Springer, 2006.

[NOT06] Robert Nieuwenhuis, Albert Oliveras, and Cesare Tinelli. Solving SAT and SAT modulo theories: From an abstract Davis–Putnam–Logemann–Loveland procedure to DPLL(). *Journal of the ACM*, 53(6):937–977, 2006.

[ORC10] Angelo Oddi, Riccardo Rasconi, and Amedeo Cesta. Project scheduling as a disjunctive temporal problem. In Helder Coelho, Rudi Studer, and Michael Wooldridge, editors, *Proc. of the 19th European Conference on Artificial Intelligence (ECAI 2010)*, volume 215 of *Frontiers in Artificial Intelligence and Applications*, pages 967–968. IOS Press, 2010.

[PP04] 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.

[PP05] Bart Peintner and Martha E. Pollack. Anytime, complete algorithm for finding utilitarian optimal solutions to STPPs. In Manuela M. Veloso and Subbarao Kambhampati, editors, *Proc. of the 20th National Conference on Artificial Intelligence (AAAI 2005)*, pages 443–448. AAAI Press / The MIT Press, 2005.

[PS82] Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall, 1982.

[Seb07] Roberto Sebastiani. Lazy satisability modulo theories. *Journal of Satisfiability, Boolean Modeling and Computation*, 3(3-4):141–224, 2007.

[SK98] 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.

[SK00] Kostas Stergiou and Manolis Koubarakis. Backtracking algorithms for disjunctions of temporal constraints. *Artificial Intelligence*, 120(1):81–117, 2000.

[SPSP05] 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.

[SSB02] Ofer Strichman, Sanjit A. Seshia, and Randal E. Bryant. Deciding Separation formulas with SAT. In Ed Brinksma and Kim Guldstrand Larsen, editors, *Proc. of the 14th International Conference on Computer Aided Verification (CAV 2002)*, volume 2404 of *Lecture Notes in Computer Science*, pages 209–222. Springer, 2002.

[TP03] Ioannis Tsamardinos and Martha Pollack. Efficient solution techniques for disjunctive temporal reasoning problems. *Artificial Intelligence*, 151:43–89, 2003.