Curriculum-based Course Timetabling with SAT and MaxSAT

Roberto Asín Achá · Robert Nieuwenhuis

Abstract We introduce novel and strong techniques based on propositional satisfiability (SAT) solvers and optimizers (MaxSAT solvers) for handling the Curriculum-based Course Timetabling problem.

Out of 32 standard benchmark instances derived from the last International Timetabling Competition, our techniques improve the best known solutions for 10 of them (4 of these 10 being optimal), and for another 9 we match the best known solution (8 of them to optimality).

There is still much room for improvement in the encodings we use as well as in the underlying general-purpose SAT and MaxSAT solvers.

Keywords Timetabling · SAT · MaxSAT

1 Introduction

The problem of deciding the satisfiability of propositional formulas (SAT) does not only lie at the heart of the most important open problem in complexity theory (P vs. NP), it is also at the basis of many practical applications in such areas as Electronic Design Automation, Verification, Artificial Intelligence and Operations Research. Thanks to recent advances in SAT-solving technology, propositional solvers are becoming the tool of choice for attacking more and more practical problems by encoding them into SAT.

Example 1 The propositional clause set $\{ \neg x_1 \vee \neg x_2 \vee \neg x_3, \quad x_1, \quad x_2 \vee \neg x_3 \}$ is satisfiable: the assignment $\{x_1, x_2, \neg x_3\}$ is a model of it. If the clause $\neg x_1 \vee x_3$ is added, the set of clauses becomes *unsatisfiable*. A (complete) *SAT solver* is a tool that, given a set of clauses, either finds a model for it or reports its unsatisfiability. □

There exist several optimization versions of the SAT problem. In *MaxSAT* the aim is to find a model that maximizes the number of satisfied clauses. In *Partial MaxSAT* the input consists of two sets of clauses, the *hard* ones and *soft* ones, and the problem

Both authors address: Technical University of Catalonia, Barcelona, `www.lsi.upc.edu/~rasin` and `~roberto`. Both are partially supported by Spanish Min. of Educ. and Science through the LogicTools-2 project (TIN2007-68093-C02-01).

is to find a model for the hard clauses that maximizes the number of satisfied soft clauses. In *Weighted (Partial) MaxSAT* each soft clause has a *weight* and the aim is to minimize the sum of the weights of the falsified soft clauses.

Here we apply novel SAT, Partial MaxSAT, and Weighted (Partial) MaxSAT encodings for handling the Curriculum-based Course Timetabling problem. For this purpose we have used our own general-purpose *Barcelogic* SAT and Partial MaxSAT solvers, as well as several other solvers for (Weighted) Partial MaxSAT. We emphasize that these solvers are based on complete search, that is, they always terminate (when given sufficient resources), in SAT returning a model or an unsatisfiability answer, and in MaxSAT, they always find the optimal solution.

Out of 32 standard benchmark instances derived from the last International Timetabling Competition (Di Gaspero et al (2007)), our techniques improve the best known solutions for 10 of them (4 of these 10 being optimal), and for another 9 we match the best known solution (8 of them to optimality). These facts can be checked at the website `http://tabu.diegm.uniud.it/ctt`. There is still much room for improvement in the encodings we use as well as in the underlying general-purpose SAT and MaxSAT solvers.

This paper is structured as follows. In Section 2 we give basic definitions and background about SAT and SAT solvers, MaxSAT and MaxSAT solvers, about encoding cardinality constraints into (Max)SAT, and we define the Curriculum-based Course Timetabling problem.

In Section 3 we give a first encoding into SAT, where also the soft constraints of the timetabling problem are made hard. Therefore, for those timetabling instances where our Barcelogic SAT solver finds a model, this provides a zero-cost solution. Surprisingly it turns out that this is indeed the case in six of the 32 instances, and in less than 10 seconds.

In Section 4 we give MaxSAT encodings where different soft constraints of the timetabling problems are made soft, and we report on the corresponding experiments.

Section 5 summarizes all our results, and in Sections 6,7 and 8 we discuss related and future work and conclude.

2 Preliminaries

2.1 SAT and SAT Solvers

Let $\mathcal{X}$ be a fixed finite set of propositional variables. If $x \in \mathcal{X}$, then x and $\neg x$ are *literals* of $\mathcal{X}$. The *negation* of a literal l, written $\neg l$, denotes $\neg x$ if l is x, and x if l is $\neg x$. A *clause* is a disjunction of literals $l_1 \vee \ldots \vee l_n$. A (total truth) *assignment* A is a set of literals such that exactly one of $\{x, \neg x\}$ is in A for each x in $\mathcal{X}$. A literal l is *true* in A if $l \in A$ and is *false* in A if $\neg l \in A$. A clause C is true in A (or *satisfied* by A) if at least one of its literals is true in A. An assignment A is a *model* of a set of clauses S if it satisfies all clauses in S. The problem of finding out whether a given clause set S has any model (i.e., is satisfiable) is known as SAT. A (complete) *SAT solver* is a system that, given a clause set S, always terminates returning a model of S or (correctly) reports its unsatisfiability.

Most state-of-the-art SAT solvers (Moskewicz et al, 2001; Goldberg and Novikov, 2002; Een and Sorensson, 2003; Ryan, 2004; Biere, 2008) use *Conflict-driven Clause Learning*, and are originally based on the Davis-Putnam-Logemann-Loveland (DPLL)

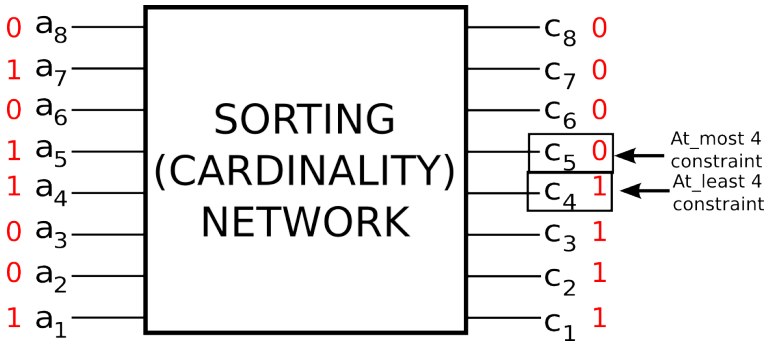

Fig. 1 Sorting network with input $\langle a_1 \dots a_8 \rangle$ and output $\langle c_1 \dots c_8 \rangle$

procedure (Davis and Putnam, 1960; Davis et al, 1962) (see, e.g., (Nieuwenhuis et al, 2006) for details and more references).

2.2 Encoding cardinality constraints into SAT

Although attempts have been made (see e.g., Cadoli and Schaerf (2005)) to define a problem-specification language for automatically generating a SAT encoding, this kind of experiences have shown to be limited in practice. For a given problem many different encodings into propositional clauses may exist, and SAT solvers can behave very differently on each one of them (see Hertel et al (2007)). Specialized encodings for a given problem may perform very well and even become a state-of-the-art technique for the problem, whereas generic ones may not even find a solution. Still, a lot of effort has been put in finding cheap and efficient ways of encoding constraints that appear in many real-world problems.

In particular, this is the case for *cardinality* constraints, saying that (at least, at most or exactly) k of a given set of n literals must be true. The natural straightforward encoding for this kind of constraints is exponential. For example, the following 10 clauses encode that *at most 2* of a set $\{x_1, x_2, x_3, x_4, x_5\}$ of variables are true:

$$\neg x_1 \vee \neg x_2 \vee \neg x_3 \quad \neg x_1 \vee \neg x_2 \vee \neg x_4 \quad \neg x_1 \vee \neg x_2 \vee \neg x_5 \quad \neg x_1 \vee \neg x_3 \vee \neg x_4$$
$$\neg x_1 \vee \neg x_3 \vee \neg x_5 \quad \neg x_1 \vee \neg x_4 \vee \neg x_5 \quad \neg x_2 \vee \neg x_3 \vee \neg x_4 \quad \neg x_2 \vee \neg x_3 \vee \neg x_5$$
$$\neg x_2 \vee \neg x_4 \vee \neg x_5 \quad \neg x_3 \vee \neg x_4 \vee \neg x_5$$

On the other hand, the following ones express *at least 2*:

$$x_1 \vee x_2 \vee x_3 \vee x_4 \quad x_1 \vee x_2 \vee x_3 \vee x_5 \quad x_1 \vee x_2 \vee x_4 \vee x_5$$
$$x_1 \vee x_3 \vee x_4 \vee x_5 \quad x_2 \vee x_3 \vee x_4 \vee x_5$$

The exponential blowup can be avoided by using auxiliary variables. For example, encodings inspired by BDDs , adder networks, and sorting networks have been proposed (Aloul et al, 2002; Bryant, 1986; Bailleux and Boufkhad, 2003).

For the encodings in this paper we have used *Cardinality Networks* (Asín et al, 2009), an improvement over sorting networks, that require $n \log^2 k$ extra-variables and $n \log^2 k$ clauses and have very good propagation properties. The basic idea is that the n input variables are seen as inputs of a circuit (encoded by clauses with auxiliary

variables) that *sorts* them into n output variables, i.e., all 1s of the input come first in the output, and then all 0s. In this way, to express that at least k input variables are true, it suffices to force the kth output variable to 1. Similarly, for "at most k", the $k+1$-th output variable is set to 0 (see figure 1; we refer to Asín et al (2009) and its references for all details, that cannot be included here).

2.3 MaxSAT and core-based MaxSAT solvers

The input to a *Weighted Partial MaxSAT* problem consists of a set of *soft* clauses, each one of them with a real number called its *weight*, and a set of *hard* clauses. The aim is to find a model of the hard clauses that minimizes the sum of the weights of the falsified soft clauses. Here the word "Weighted" is dropped if all soft clauses have the same weight, and the word "Partial" is dropped if there are no hard clauses (note that hard clauses can also be seen as soft clauses with infinite weight).

The state of the art in MaxSAT solvers is still seeing rapid development. The improvements in recent solvers are largely due to new algorithms based on *unsatisfiable cores*. The idea is (very roughly) the following. A (standard) SAT solver can be adapted to return, in case of unsatisfiabilty, a small (or even minimal, wrt. set inclusion) unsatisfiable subset of the initial set of clauses (Zhang and Malik, 2003). Such subsets, called (unsatisfiable) *cores*, are obviously useful (in applications like planning) for generating a small explanation why no feasible solution exists.

For solving (unweighted) MaxSAT, a SAT solver is run on the clause set. If a model is found, then a zero-cost solution has been found. Otherwise, a core is generated, an additional fresh *relaxation* variable is added to each soft clause in the core, and a cardinality constraint is added saying that at most one of the relaxation variables can be true, i.e., at most one of the clauses in the core is allowed to become false. Then the SAT solver is restarted on this extended clause set. This process is iterated until a model is found, which is then provably optimal. Many variants and improvements on this technique exist (that cannot be covered in this paper; see e.g., Fu and Malik (2006); Marques-Silva and Planes (2008); Ansótegui et al (2009); Manquinho et al (2009)). In particular, our new Barcelogic solver we use here for Partial MaxSAT uses a novel concept of *core clusters* to improve the pruning power of the cardinality constraints and never add more than one relaxation variable to any clause.

2.4 The Curriculum-based Course Timetabling Problem

The Curriculum-based Course Timetabling Problem was defined for track 3 of the 2nd International Timetabling Competition (ITC) held in 2007 and its complete description can be found in (Di Gaspero et al, 2007). Briefly, this problem deals with the following objects:

Courses: A *course* is composed of a defined number of lectures on some subject. Each course is associated to one teacher, a number of students and a minimum number of days over which its lectures may be spread. For example, there may exist a 50-student course c_1, taught by teacher t_1, consisting of 5 lectures to be spread over at least 3 distinct days of the week.

Curriculums: A *curriculum* is a set of courses that may share students. For example, curriculum k_1 may consist of four courses: c_1, c_2, c_3 and c_4.

Rooms: Rooms are the spaces in which courses take place. Each room has an associated capacity (number of students, typically from 20 to 1000).
Days and hours: The courses are taught weekly on some day and hour. Each day consists of a specified number of lecture hours in which a course can take place.

The problem is to find an assignment of courses to rooms and hours in such a way that the following *hard* (necessary) and *soft* (desirable) constraints are satisfied.

Hard Constraints:

Curriculum clashes: No two courses belonging to the same curriculum may be scheduled at the same time.
Teacher clashes (Hard): No two courses taught by the same teacher may occur at the same time.
Room clashes: No room must be used for more than one course at the same time.
Hour availability: Teachers may declare themselves as not available for certain hours.
Number of lectures: Exactly the specified number of lectures of every course must be scheduled.

Soft Constraints:

Room capacity: No course may be scheduled to a room with less capacity than the one needed by the course. Each violation of this constraint has a cost of 1 per student that does not fit into the room.
Min working days: The lectures of a given course must be spread over a given minimal number of days. Each day less than the minimum for a course has a cost of 5.
Isolated lectures: Lectures belonging to a curriculum should be adjacent to another lecture of the same curriculum. Each time a lecture belonging to one curriculum is isolated, this violation has cost 2.
Room stability: All lectures of a given course should be scheduled to the same room. Each extra room needed for a course has cost 1.

3 Our Basic SAT encoding

Here we give a first encoding into SAT. All other encodings in this paper are only very slight variants over this (two-page) basic encoding.

In this first encoding also the soft constraints of the timetabling problem are made hard. Therefore, for those timetabling instances where the SAT solver finds a model, this provides a zero-cost solution. Surprisingly it turns out that this is indeed the case in six of the 32 instances, and in less than 10 seconds. In this section we have used our Barcelogic SAT solver, that ranked third in the last SAT Race (2008, Guangzhou, China, see `baldur.iti.uka.de/sat-race-2008`), and first on unsatisfiable problems which is what matters most for optimisation applications.

We define the following propositional variables:

- $ch_{c,h}$: "course c occurs in hour h"
- $cd_{c,d}$: "course c occurs in day d"
- $cr_{c,r}$: "course c occurs in room r"
- $kh_{k,h}$: "curriculum k occurs in hour h"

Once these propositional variables are defined, for the encoding to be correct, that is, admit all correct solutions, and only these ones, we must carefully express every relation between the facts these variables represent.

Relation between ch and cd:

- If some course occurs in hour h, it also occurs in the day corresponding to h. So, for each course c and hour h, the following two-literal clause is needed:

$$\neg ch_{c,h} \vee cd_{c,day(h)}$$

- If some course occurs in a day d, it must also occur in some of the hours of d. So, for each course c and day d consisting of hours $h_1, h_2, \ldots h_n$, the following clause is needed:

$$\neg cd_{c,d} \vee ch_{c,h_1} \vee \ldots \vee ch_{c,h_n}$$

Relation between ch and kh:

- If some course c occurs in hour h, all the curricula $k_1, k_2, \ldots, k_{nk}$, to which c belongs occur in h. So, for each course c, hour h and curricula $k_1, k_2, \ldots, k_n$ that include c, the following clauses are needed:

$$\begin{array}{c} \neg ch_{c,h} \vee kh_{k_1,h} \\ \neg ch_{c,h} \vee kh_{k_2,h} \\ \vdots \\ \neg ch_{c,h} \vee kh_{k_n,h} \end{array}$$

- If some curriculum k occurs in hour h, then at least one of the courses belonging to k must also occur in h. So, for every hour h and curriculum k consisting of courses $c_1, c_2, \ldots, c_n$, the following clauses are needed:

$$\neg ct_{k,h} \vee ch_{c_1,h} \vee \ldots \vee ch_{c_n,h}$$

We now encode the constraints of the timetabling problem. Here we abstract away the concrete encoding of cardinality constraints by simply denoting them by $at\_most(k, S)$, $at\_least(k, S)$ and $exactly(k, S)$, where $k \in \mathbb{N}$ and S is a set of literals. Such an expression represents a set of clauses that are satisfied if, and only if at least (at most, or exactly) k of the nc variables of S are true.

Curriculum clashes: No two courses c and c' belonging to the same curriculum may be scheduled at the same hour. So, for each hour h and for each pair of distinct courses c, c' belonging to the same curriculum, we have:

$$\neg ch_{c,h} \vee \neg ch_{c',h}$$

Teacher clashes: No two courses c and c' with the same teacher may be scheduled to the same hour. So, for each hour h and for each pair of distinct courses c, c' such that $teacher(c) = teacher(c')$, we have:

$$\neg ch_{c,h} \vee \neg ch_{c',h}$$

Room clashes: No two courses c and c' may be scheduled to the same room at the same hour. So, for each room r, hour h, and pair of distinct courses c, c', we need:

$$\neg ch_{c,h} \vee \neg ch_{c',h} \vee \neg cr_{c,r} \vee \neg cr_{c',r}$$

Hour availability: For each course c with forbidden hours $h_1, h_2, \ldots, h_n$, we have the following one-literal clauses:

$$\begin{array}{l} \neg ch_{c,h_1} \\ \neg ch_{c,h_2} \\ \vdots \\ \neg ch_{c,h_n} \end{array}$$

Number of lectures: For each course c exactly $hours(c)$ of the set $ch_{c,h_1}, ch_{c,h_2}, \ldots, ch_{c,h_n}$ must be true:

$$exactly(hours(c), \{ch_{c,h_1}, ch_{c,h_2}, \ldots, ch_{c,h_n}\})$$

Room capacity: Each course must be scheduled to a room in which it fits. So, for each course c with number of students ns and for every room r with capacity cr such that $cr < ns$, we have:

$$\neg cr_{c,r}$$

Min working days: For each course c, at least $working\_days(c)$ literals of the set $cd_{c,d_1}, cd_{c,d_2}, \ldots, cd_{c,d_5}$ should be true:

$$at\_least(working\_days(c), \{cd_{c,d_1}, cd_{c,d_2}, \ldots, cd_{c,d_5}\})$$

Isolated lectures: If some curriculum k occurs in hour h, then k must also occur in an hour before or after in the same day. For each curriculum k:

- For each first hour of a day h:

$$\neg kt_{c,h} \vee kt_{k,h+1}$$

.

- For each last hour of a day h:

$$\neg kt_{c,h} \vee kt_{k,h-1}$$

- For each hour h that is not the first nor the last of a day:

$$\neg kt_{c,h} \vee kt_{k,h-1} \vee kt_{k,h+1}$$

Room stability: Each course must be scheduled to exactly one room. So, if there are rooms $r_1, \ldots, r_n$, for each course c we have[1]:

$$exactly(1, \{cr_{c,r_1}, \ldots, cr_{c,r_n}\})$$

[1] Note that, by including here only the rooms with sufficient capacity, the Room Capacity constraint would get subsumed. Here we have not done this becuase later on the latter constraint will become soft.

Table 1 Solving Times for basic encoding

dataset	vars	clauses	result	time
comp01	12886	62877	-	TO
comp02	110288	575890	UNSAT	2.0
comp03	72749	456172	UNSAT	0.8
comp04	92209	560109	UNSAT	0.6
comp05	34053	179043	UNSAT	0.2
comp06	105728	936770	UNSAT	0.6
comp07	214603	1803516	UNSAT	5.8
comp08	115353	708140	UNSAT	2.3
comp09	105952	603965	UNSAT	0.7
comp10	147673	1211016	UNSAT	3.9
comp11	12537	71919	SAT(=)	1.2
comp12	81659	470203	UNSAT	0.2
comp13	111401	626315	UNSAT	0.7
comp14	113180	675440	UNSAT	0.8
comp15	72749	456172	UNSAT	0.7
comp16	148258	1248224	UNSAT	0.8
comp17	144334	924138	UNSAT	0.6
comp18	40444	174369	UNSAT	0.2
comp19	109236	525817	UNSAT	0.3
comp20	116149	1184210	UNSAT	4.4
comp21	129565	926364	UNSAT	1.0
DDS1	900167	2588788	UNSAT	1.5
DDS2	137688	667470	SAT(=)	1.7
DDS3	60968	305601	SAT(=)	1.1
DDS4	1356276	12842169	UNSAT	25.7
DDS5	556569	3372803	SAT(=)	8.9
DDS6	125029	1001737	**SAT**	10.3
DDS7	124330	612475	SAT(=)	2.4
test1	19406	182328	UNSAT	0.4
test2	34748	213222	UNSAT	0.6
test3	63534	271811	UNSAT	0.2
test4	67539	293208	UNSAT	0.4

3.1 Experiments with the basic SAT encoding

We have tested this encoding over the full set of benchmarks presented in the ITC2007 track3 organizers' web site (`http://tabu.diegm.uniud.it/ctt`). All instances labeled with comp# are competition benchmarks, while the DDS# are bigger ones added after the competition. The test# examples were given for testing. They are small but hard. The web site allows researchers to compare on these benchmarks with the best known results and report new ones that are automatically checked for correctness and cost.

All experiments in this paper are on a 2100Mhz AMD-Opteron with 10000s timeout (TO) and 2GB memory-out (MO). We never include encoding time since it is negligible.

Table 1 lists our results on these benchmarks for the basic SAT encoding: number of variables and clauses, the output SAT/UNSAT, and the time used by our Barcelogic SAT solver. In six instances a model, and thus a zero-cost solution, is found (in 1.1, 1.2, 1.7, 2.4, 8.9, and 10.3 seconds respectively). For five of them, a solution of the same cost (in this case, zero) was already known. In all tables in this paper this is indicated with an = sign. In addition, in very little time (10.3 seconds), we also obtain a new zero-cost solution (DDS6) beating all the reported results over this benchmark.

Table 2 Results for relaxing "isolated lectures" as Partial-MaxSAT

dataset	vars	clauses	cost	barcelogic time	PM2 time
comp01	12886	62877	-	TO	TO
comp02	110288	575890	**24**	3719.0	TO
comp03	72749	456172	-	TO	TO
comp04	92209	560109	36	13.2	60.2
comp05	34053	179043	∞	131.3	89.5
comp06	105728	936770	**28**	540.5	270.7
comp07	214603	1803516	**6**	9625.0	MO
comp08	115353	708140	38	18.1	91.2
comp09	105952	603965	-	TO	TO
comp10	147673	1211016	**4**	145.8	226.6
comp11	12537	71919	0(=)	0.5	2.8
comp12	81659	470203	-	TO	TO
comp13	111401	626315	62	68.8	153.2
comp14	113180	675440	54	111.7	95.3
comp15	72749	456172	-	TO	TO
comp16	148258	1248224	**22**	24.7	84.6
comp17	144334	924138	**60**	700.8	7817.9
comp18	40444	174369	-	TO	TO
comp19	109236	525817	58	173.1	372.4
comp20	116149	1184210	**4**	2305.6	674.4
comp21	129565	926364	**86**	8874.2	TO
DDS1	900167	2588788	∞	1.4	5.3
DDS2	137688	667470	0(=)	0.8	3.0
DDS3	60968	305601	0(=)	0.4	1.9
DDS4	1356276	12842169	-	TO	MO
DDS5	556569	3372803	0(=)	10.4	17.9
DDS6	125029	1001737	0(=)	15.0	14.4
DDS7	124330	612475	0(=)	1.3	4.9
test1	19406	182328	-	TO	MO
test2	34748	213222	16(=)	123.4	59.3
test3	63534	271811	∞	0.3	0.6
test4	67539	293208	-	TO	MO

4 MaxSAT encodings

4.1 Relaxing "isolated lectures" as Partial-MaxSAT

Here we turn the "isolated lectures" constraint into a soft constraint. In the basic SAT encoding, each clause for this constraint encoded exactly one violation of the constraint. So here each one of these clauses that is not satisfied has cost 2: we use a Partial MaxSAT solver that considers these clauses as soft ones, and then each model found represents a solution with as cost twice the number of unsatisfied soft clauses.

Table 2 lists results using our own first partial-maxsat-solver prototype (Barcelogic) and also with PM2 (Ansótegui et al (2009)), the winner of the Industrial Partial MaxSAT track in the latest MaxSAT competition (`www.maxsat.udl.cat/09`). A cost of ∞ indicates that there is no solution if only the isolated lectures constraint is made soft. With respect to the SAT encoding where everything was hard, the number of instances for which we can give a solution grows from 6 to 20. Of the 14 new ones, 8 (indicated in bold) improve the previous best cost reported by the community.

Note that also the 6 zero-cost results of the previous section are obtained with this method and that we also match the best resported result (cost 16) for the test2 instance. In all tables, if the name of the instance is given in bold, this means that our solution is known to be optimal (the website also reports lower bounds).

Table 3 Results for relaxing "min working days" as Weighted-Partial-MaxSAT

dataset	vars	clauses	cost	WPM1 time	MSUNCORE time
comp01	13864	64189	-	TO	TO
comp02	111869	578209	-	MO	TO
comp03	74178	458243	-	MO	TO
comp04	93497	562036	35(=)	7518	TO
comp05	34902	180478	-	MO	MO
comp06	107670	939713	-	MO	TO
comp07	216804	1806892	6(=)	MO	6442
comp08	116948	710560	37(=)	4455	TO
comp09	107448	606143	-	MO	TO
comp10	149513	1213821	4(=)	62	57
comp11	13511	73276	0(=)	1	0.8
comp12	83492	473168	-	MO	TO
comp13	113012	628743	-	MO	TO
comp14	114680	677720	-	TO	TO
comp15	74178	458243	-	MO	TO
comp16	150222	1251187	**18**	615.3	391.6
comp17	146057	926721	-	MO	TO
comp18	41185	175639	-	TO	TO
comp19	110890	528176	-	MO	TO
comp20	118188	1187337	4(=)	325.8	193.3
comp21	131373	929034	-	MO	TO
DDS1	903884	2594511	**48**	645.0	1629.4
DDS2	137850	667646	0(=)	2.6	3.6
DDS3	62282	307644	0(=)	1.7	2.4
DDS4	1360646	12848872	-	MO	MO
DDS5	558714	3376303	0(=)	14.5	22.3
DDS6	126753	1004376	0(=)	22.4	14.5
DDS7	125311	614021	0(=)	4.2	5.0
test1	21022	184475	-	MO	MO
test2	36375	215426	16(=)	27.5	32.3
test3	65415	274319	-	TO	TO
test4	69420	295717	-	TO	TO

4.2 Relaxing "min working days" as Weighted-Partial-MaxSAT

Here we describe a very efficient way of relaxing, in addition to the "isolated lectures" constraint, also the "min working days" constraint. In the basic SAT encoding, we used cardinality networks to encode for each course c the min working days constraint $at\_least(working\_days(c), \{cd_{c,d_1}, cd_{c,d_2}, \ldots, cd_{c,d_5}\})$. For instance, if $working\_days(c)$ is 4, then we set the fourth output out_4 of the network to 1. In order to make this soft, and such that each day less than four has cost 5, we create one soft one-literal clause out_4 with weight 5, and two more weight-5 one-literal clauses for out_3 and out_2.

Table 3 shows results with two state-of-the-art Weighted-Partial MaxSat solvers that did very well in the last MaxSAT competition... *msuncore* (Manquinho et al, 2009) and *WPM1* (Ansótegui et al, 2009) (our Barcelogic solver cannot handle weighted MaxSAT yet). The winner of the industrial division *SAT4J* behaved much worse on these problems. With any of the two solvers, this encoding allows us to again improve the best known cost on two more instances (comp16 and DDS1). We also match the best solutions on two more (comp04 and 08). These solutions are moreover optimal for DDS1, comp04 and comp08.

Table 4 Results for Weighted-Partial-MaxSAT as Partial-MaxSAT

dataset	vars	clauses	cost	PM2 time
comp01	13864	65025	-	TO
comp02	111869	580927	-	TO
comp03	74178	460787	-	TO
comp04	93497	564285	35(=)	111.3
comp05	34902	186066	-	TO
comp06	107670	942715	**27**	4194.4
comp07	216804	1810289	-	MO
comp08	116948	713049	37(=)	158.7
comp09	107448	608874	-	TO
comp10	149513	1216744	4(=)	135.8
comp11	13511	74249	0(=)	1.3
comp12	83492	6479436	-	TO
comp13	113012	631309	59(=)	1867.9
comp14	114680	680244	51(=)	545.5
comp15	74178	460787	-	TO
comp16	150222	1254202	18(=)	143.7
comp17	146057	929587	-	TO
comp18	41185	178063	-	TO
comp19	110890	530718	-	TO
comp20	118188	1190643	4(=)	542.4
comp21	131373	932016	-	TO
DDS1	903884	2603332	48(=)	748.2
DDS2	137850	668420	0(=)	3.1
DDS3	62282	308551	0(=)	2.5
DDS4	1360646	12855538	-	MO
DDS5	558714	3380575	0(=)	16.9
DDS6	126753	1007166	0(=)	25.8
DDS7	125311	616673	0(=)	5.2
test1	21022	185639	-	MO
test2	36375	216742	16(=)	59.3
test3	65415	276191	-	MO
test4	69420	297864	-	TO

4.3 Weighted-Partial-MaxSAT as Partial-MaxSAT

Instead of using Weighted MaxSAT, one can also replicate each soft clause as many times as the cost related with it, and use a solver for unweighted MaxSAT. It turns out that this works very well on these problems because all weights are small. This can be observed in table 4: besides keeping the previous results, we beat one previously improved example's cost (comp06) and achieve another two best reported costs (comp13 and comp14). We did not use our Barcelogic solver because it has no support for replicated clauses.

Table 5 Results summary over the curriculum-based timetabling problem

dataset	Author previously best	Method	previous cost	our cost
comp01	Andrea Schaerf (+ others)	Various	5	-
comp02	Lu And Hao	Tabu Search	29	**24**
comp03	Tomas Muller	Local Search	66	-
comp04	Tomas Muller	Local Search	35	35(=)
comp05	Lu And Hao	Tabu Search	292	-
comp06	Tomas Muller	Local Search	37	**27**
comp07	Tomas Muller	Local Search	7	**6**
comp08	S. Abdullah, H. Turabieh	Other	37	37(=)
comp09	Lu And Hao	Tabu Search	96	-
comp10	Tomas Muller	Local Search	7	**4**
comp11	Andrea Schaerf (+ others)	Various	0	0(=)
comp12	Lu And Hao	Tabu Search	310	-
comp13	Lu And Hao	Tabu Search	59	62
comp14	Gerald Lach	Mathematical Progr.	51	51(=)
comp15	Andrea Schaerf (+ others)	Various	66	-
comp16	Lu And Hao	Tabu Search	23	**18**
comp17	Lu And Hao	Tabu Search	69	**60**
comp18	Lu And Hao	Tabu Search	65	-
comp19	Tomas Muller	Local Search	57	58
comp20	Gerald Lach	Mathematical Progr.	17	**4**
comp21	Tomas Muller	Local Search	89	**86**
DDS1	Gerald Lach	Mathematical Progr.	83	**48**
DDS2	Andrea Schaerf (+ others)	Various	0	0(=)
DDS3	Andrea Schaerf (+ others)	Various	0	0(=)
DDS4	S. Abdullah, H. Turabieh	Evolutionary Comp.	30	-
DDS5	Andrea Schaerf	Tabu Search	0	0(=)
DDS6	Gerald Lach	Mathematical Progr.	4	**0**
DDS7	Andrea Schaerf (+ others)	Various	0	0(=)
test1	Lu And Hao	Tabu Search	224	-
test2	Andrea Schaerf (+ others)	Various	16	16(=)
test3	S. Abdullah, H. Turabieh	Other	67	-
test4	Lu And Hao	Tabu Search	73	-

5 Results summary

Table 5 gives a summary of our achievements in encoding the curriculum-based course timetabling problem into different versions of SAT/MaxSAT. In the column "our cost" all figures in **bold** indicate costs where we have inproved the best known solution (10 of the 32 instances). In the column "previous cost" one can see the previously known best costs, which in some case we improve importantly: for example in DDS1 we improve from cost 83 to 48, which is moreover known to be optimal. For another 9 instances we match the best known solution (8 of them to optimality: again the names of the benchmarks are in bold if the best solution found is known to be optimal).

In the table We also indicate the author of the previously best known costs, the first one in obtaining the result (as it appears on the web site) and add the label (+others) to clarify that others also reached the same results. The column method indicates the technique used to obtain these previously best known results.

Altogher, out of 32 instances, we obtained 19 of the current best known results, 10 of which were improvements over the past known ones.

6 Related work

As it can be seen in Table 5 (see Lü and Hao (2010)) and in the algorithms specifications of the solvers that participated in the two timetabling competitions held until now (see, for example, Kostuch (2004) and Muller (2005)), the timetabling research field has been mostly dominated by local search techniques. In recent years, nevertheless, other techniques like Constraint Programming or Mathematical Programming have been presented with some success. On the other hand, in the specific case of SAT technologies applied to timetabling problems, not much work has been done. In fact, we are not aware of any work on timetabling using MaxSAT.

Using pure SAT some non-competitive (in expressivity and solving time) techniques are described in the two master thesis (Chin-A-Fat, September 2004; Hartog, 2007). The unpublished manuscript (Marić, 2008) deals with non-standard and hence hard-to-compare benchmarks, and uses more naive encodings (e.g., quadratic-size cardinality constraints).

7 Future Work

A lot of room for improvement exists in MaxSAT solvers for this kind of problems. We plan to work on our own solvers, improving our beta Partial MaxSAT and developing a new Weighted version of it. Also, to encourage the MaxSAT community to work on larger and more practically "industrially"-oriented benchmarks, we plan to contribute our encoded instances to the MaxSAT competition.

We also plan to work on new ways for efficiently encoding (and relaxing) some typical constraints in timetabling problems, like "Room allocation" and "Room stability". Furthermore, in instance "comp01" we found that inside timetabling problems, *pigeon-hole-like problems* can appear. As it is well known in the SAT community, these problems are very difficult to handle for SAT Solvers (which is the case of this particular instance). We also plan to search for ways of dealing with this type of constraints, possibly as a *theory* in the SAT Modulo Theories framework (Nieuwenhuis et al, 2006).

8 Conclusions

In this paper we have tackled the curriculum-based course timetabling problem. We have presented several encodings for the problem from pure SAT to Partial and Weighted-Partial MaxSAT. Each encoding has been tried on 32 instances corresponding to those shown in `http://tabu.diegm.uniud.it/ctt/` that belong to the 3rd track of the last International Timetabling Competition. We have tested several current-state-of-the-art SAT an MaxSAT solvers with good results, achieving 19 out of 32 of the current best known results, 10 of which were improvements over the past known ones.

This shows that using SAT and MaxSAT for timetabling is feasible and productive and encourages us to keep working in two main directions: i) to search for better suited MaxSAT solving techniques and ii) to find better encodings for this kind of problems.

Acknowledgements We want to thank Carlos Anstegui for his advise and help providing us access to his PM2 and WPM1 solvers.

References

Aloul FA, Ramani A, Markov IL, Sakallah KA (2002) Generic ilp versus specialized 0-1 ilp: an update. In: Pileggi LT, Kuehlmann A (eds) ICCAD, ACM, pp 450–457

Ansótegui C, Bonet ML, Levy J (2009) Solving (weighted) partial maxsat through satisfiability testing. In: Kullmann (2009), pp 427–440

Asín R, Nieuwenhuis R, Oliveras A, Rodríguez-Carbonell E (2009) Cardinality networks and their applications. In: Kullmann (2009), pp 167–180

Bailleux O, Boufkhad Y (2003) Efficient cnf encoding of boolean cardinality constraints. In: Rossi F (ed) CP, Springer, Lecture Notes in Computer Science, vol 2833, pp 108–122

Biere A (2008) PicoSAT essentials. Journal on Satisfiability, Boolean Modeling and Computation Submitted

Bryant RE (1986) Graph-based algorithms for boolean function manipulation. IEEE Trans Comput 35(8):677–691

Cadoli M, Schaerf A (2005) Compiling problem specifications into SAT. Artificial Intelligence 162(1-2):89–120

Chin-A-Fat K (September 2004) School timetabling using satisfiability solvers. Master's thesis, Technical University Delft, The Netherlands

Davis M, Putnam H (1960) A computing procedure for quantification theory. Journal of the ACM 7:201–215

Davis M, Logemann G, Loveland D (1962) A machine program for theorem-proving. Comm of the ACM 5(7):394–397

Di Gaspero L, McCollum B, Schaerf A (2007) The second international timetabling competition (itc-2007): Curriculum-based course timetabling (track 3). Tech. rep., University of Udine

Een N, Sorensson N (2003) An extensible sat-solver. In: Proceedings of the Sixth International Conference on Theroy and Applications of Satisfiability Testing (SAT), pp 501–518

Fu Z, Malik S (2006) On solving the partial max-sat problem. In: Theory and Applications of Satisfiability Testing, SAT, vol LNCS 4121, pp 252–265

Goldberg E, Novikov Y (2002) BerkMin: A fast and robust SAT-solver. In: Design, Automation, and Test in Europe (DATE '02), pp 142–149

Hartog J (2007) Timetabling on dutch high schools: Satisfiability versus gp-untis. Master's thesis, Technical University Delft, The Netherlands

Hertel A, Hertel P, Urquhart A (2007) Formalizing dangerous sat encodings. In: Marques-Silva J, Sakallah KA (eds) SAT, Springer, Lecture Notes in Computer Science, vol 4501, pp 159–172

Kostuch P (2004) The university course timetabling problem with a three-phase approach. In: Burke EK, Trick MA (eds) PATAT, Springer, Lecture Notes in Computer Science, vol 3616, pp 109–125

Kullmann O (ed) (2009) Theory and Applications of Satisfiability Testing - SAT 2009, 12th International Conference, SAT 2009, Swansea, UK, June 30 - July 3, 2009. Proceedings, Lecture Notes in Computer Science, vol 5584, Springer

Lü Z, Hao JK (2010) Adaptive tabu search for course timetabling. European Journal of Operational Research 200(1):235–244

Manquinho VM, Silva JPM, Planes J (2009) Algorithms for weighted boolean optimization. In: Kullmann (2009), pp 495–508

Marić F (2008) Timetabling based on sat encoding: a case study, faculty of Mathematics, University of Belgrade, Serbia

Marques-Silva J, Planes J (2008) Algorithms for maximum satisfiability using unsatisfiable cores. In: Proceedings of Design, Automation and Test in Europe (DATE 08), pp –

Moskewicz MW, Madigan CF, Zhao Y, Zhang L, Malik S (2001) Chaff: Engineering an Efficient SAT Solver. In: Proc. 38th Design Automation Conference (DAC'01)

Muller T (2005) Constraint-based Timetabling. PhD thesis, PhD thesis, Charles University in Prague, Faculty of Mathematics and Physics, 2005

Nieuwenhuis R, Oliveras A, Tinelli C (2006) Solving SAT and SAT Modulo Theories: from an Abstract Davis-Putnam-Logemann-Loveland Procedure to DPLL(T). Journal of the ACM 53(6):937–977

Ryan L (2004) Efficient Algorithms for Clause-Learning SAT Solvers. Master's thesis, School of Computing Science, Simon Fraser University

Zhang L, Malik S (2003) Validating SAT Solvers Using an Independent Resolution-Based Checker: Practical Implementations and Other Applications. In: 2003 Design, Automation and Test in Europe Conference (DATE 2003), IEEE Computer Society, pp 10,880–10,885