Minimizing Working-Group Conflicts in Conference Session Scheduling Through Maximum Satisfiability

Sami Cherif, Heythem Sattoutah, Chu-Min Li, Corinne Lucet, Laure Brisoux-Devendeville

Minimizing Working-Group Conflicts in Conference Session Scheduling Through Maximum Satisfiability

Sami Cherif
MIS UR 4290, Université de Picardie Jules Verne, Amiens, France

Heythem Sattoutah
MIS UR 4290, Université de Picardie Jules Verne, Amiens, France

Chu-Min Li
MIS UR 4290, Université de Picardie Jules Verne, Amiens, France

Corinne Lucet
MIS UR 4290, Université de Picardie Jules Verne, Amiens, France

Laure Brisoux-Devendeville
MIS UR 4290, Université de Picardie Jules Verne, Amiens, France

Abstract

This paper explores the application of Maximum Satisfiability (Max-SAT) to the complex problem of conference session scheduling, with a particular focus on minimizing working-group conflicts within the context of the ROADEF conference, the largest French-speaking event aimed at bringing together researchers from various fields such as combinatorial optimization and operational research. A Max-SAT model is introduced then enhanced with new variables, and solved through state-of-the-art solvers. The results of applying our formulation to data from ROADEF demonstrate its ability to effectively compute session schedules, while enabling to reduce the number of conflicts and the maximum number of parallel sessions compared to the handmade solutions proposed by the organizing committees. These findings underscore the potential of Max-SAT as a valuable tool for optimizing conference scheduling processes, offering a systematic and efficient solution that ensures a smoother and more productive experience for attendees and organizers alike.

2012 ACM Subject Classification Computing methodologies → Planning and scheduling; Computing methodologies → Logic programming and answer set programming

Keywords and phrases Maximum Satisfiability, Scheduling, Modeling

Digital Object Identifier 10.4230/LIPIcs.CP.2024.34

Category Short Paper

Supplementary Material *Other (Source Code, Data, Benchmark)*:
`https://github.com/satoutahhaithem/ROADEF_SCHEDULING` [12]
archived at `swh:1:dir:7083377094f69163d37d30b77d740c72c562139d`

1 Introduction

The Maximum Satisfiability (Max-SAT) problem is a natural optimization extension of the well-known satisfiability (SAT) problem [9]. While SAT consists in verifying whether all the clausal constraints in a given propositional formula can be satisfied by an assignment of the variables, the goal in Max-SAT shifts to determining the maximum number of clausal constraints that can be satisfied. Although Max-SAT is NP-hard, the last decade has known major breakthroughs in Max-SAT theory and solving. Indeed, SAT-based algorithms [2, 18, 27] are able to harness the power of modern SAT solvers and particularly their ability to effectively compute cores, i.e. unsatisfiable subsets of the formula. More

recently, Branch and Bound algorithms for Max-SAT have also gained a huge leap in competitiveness with respect to other approaches through the introduction of a foundational learning process tailored to Max-SAT [11, 22, 23]. This was coupled with new insights and understanding of Max-SAT proof theory, paving the way for efficient solver certification for Max-SAT [7, 29].

These recent advancements have made Max-SAT a powerful formalism that can be used to model and effectively solve many real-world problems. In particular, Max-SAT has been used to tackle applications in various domains such as security [31], hardware and software debugging [25], data analysis [8] and bio-informatics [19] among many others. In particular, in the context of scheduling, Max-SAT has been used for tackling Curriculum-based Course Timetabling in [4] and [14], mainly for optimizing isolated lectures or resources in the schedule. It was also used in [10] to avoid idle time periods in the scheduling of bilateral meetings as well as in [15] to solve staff scheduling problems while taking into account requested and unpreferred shifts. A Max-SAT formulation for scheduling tasks in overloaded real-time systems by enforcing task deadlines as soft constraints is introduced in [24]. More recently, an iterative and incremental Max-SAT approach has been introduced to solve train scheduling optimization problems in [20]. These successful applications show the high potential of Max-SAT as an effective paradigm to solve problems in the context of scheduling and beyond.

In this paper, we address a conference scheduling problem through the lens of Max-SAT. In particular, we aim to tackle a problem which has arisen during the organization of the ROADEF conference, the largest French-speaking event aimed at bringing together researchers from various domains, mainly the combinatorial optimization and operational research communities. Our work addresses a crucial gap in the organization of the esteemed conference where, despite its status as a gathering of premier scheduling experts, the planning process remains reliant on manual efforts, requiring significant human and time resources, even after 25 editions. Our objective is thus to provide an effective way for optimizing the conference scheduling process, thus enhancing the overall experience for conference attendees and organizers alike. To this end, we introduce a Max-SAT model to minimize the working-group conflicts in the scheduling of parallel sessions during the conference. Furthermore, we show how our model can be enhanced by introducing new variables. Our experimental evaluation shows that our formulation offers a systematic and efficient solution to the intricate challenges associated with our problem, ensuring a smoother and more productive experience for attendees and organizers as it enables to reduce the number of conflicts as well as the maximum number of parallel sessions compared to the handmade solutions proposed by the organizing committees.

This paper is organized as follows. Section 2 includes the necessary definitions and notations. The problem context and parameters are presented in Section 3. Our dedicated Max-SAT model and its enhanced formulation are introduced in Section 4. An experimental evaluation on data from ROADEF is presented in Section 5. Finally, we conclude and discuss future work in Section 6.

2 Preliminaries

Let V be the set of propositional variables. A literal l is a variable $x \in V$ or its negation $\overline{x}$. A clause is a disjunction of literals. A formula in Conjunctive Normal Form (CNF) is a conjunction of clauses. An assignment $\alpha : V \to \{true, false\}$ maps each variable to a Boolean value and can be represented as a set of literals. A literal l is satisfied by an assignment α if $l \in \alpha$, else it is falsified by α. A clause is satisfied by an assignment α if

at least one of its literals is satisfied by α, otherwise it is falsified by α. The cost of an assignment α, denoted $cost_\alpha(\phi)$, is the number of clauses in the formula ϕ falsified by α. Solving the (plain) Max-SAT problem [6] consists in determining the maximum number of clauses that can be satisfied by an assignment of a given CNF formula ϕ or, equivalently, the minimum number of clauses that each assignment must falsify, i.e. $opt(\phi) = \min_{\alpha} cost_\alpha(\phi)$. In our formulation, we will particularly rely on the partial Max-SAT problem [21], which is a well-known variant taking as input a partial CNF formula, i.e. a bipartite set of clauses $\phi = H \cup S$ where H is the set of hard clauses that must be satisfied and S is the set of soft clauses to be optimized as in plain Max-SAT. The goal thus shifts to computing $opt(S)$ such that all the clauses in H are satisfied. Finally, we note that, in the subsequent Max-SAT formulation, we will use Pseudo-Boolean (PB) constraints which enforce a constant bound on a weighted sum of literals as in $(\sum_{i=1}^{h} a_i * l_i) \circ k$ where $a_i, k \in \mathbb{N}$ and $\circ \in \{\leq, =, \geq\}$. In the case where all literals in the sum are variables in V and all the weights are set to 1 (and thus can be discarded), such constraints are more commonly referred to as cardinality constraints. PB and cardinality constraints can be efficiently encoded in CNF form [30] and are typically used in Max-SAT solvers to enforce relevant bounds during the search [21]. For instance, the sorting network encoding [16] enables to encode PB and cardinality constraints using $O(\sum_{i=1}^{h} a_i \, log^2(\sum_{i=1}^{h} a_i))$ new variables and clauses, i.e. $O(h \, log^2(h))$ for cardinality constraints as $m = h$. Independently, the cardinality network encoding [3] can be more suited for cardinality constraints as it is in $O(h \, log^2(k))$ while the BDD encoding [1], which is in $O(h^3 \, log(max_{1 \leq i \leq h} a_i))$, can be more suitable for PB constraints.

3 ROADEF Scheduling

The ROADEF[1] conference is the largest French-speaking event aimed at bringing together researchers from various domains, including combinatorial optimization, operational research, constraint programming and industrial engineering. This event is organized annually and welcomes around 600 participants, with the aim to foster exchanges and collaborations between researchers and industry professionals, as well as to contribute to the training of young researchers and encourage them to present their work to the national community. ROADEF includes plenary sessions, tutorials in semi-plenary sessions, multiple parallel sessions, industrial experience feedback sessions and the ROADEF general assembly. The conference also involves many working groups consisting of researchers collaborating on a national and potentially international level on specific themes covered by the conference, with each parallel session usually being organized by one or more of these working groups. In this paper, we focus on the scheduling of ROADEF parallel sessions into available time slots while avoiding clashes among research working groups. This problem arises each year during the event organization as each parallel session is usually scheduled to occur simultaneously with others, potentially causing overlaps and conflicts. Indeed, such overlaps not only prevent researchers from assisting to the sessions organized by their respective working groups but they may also pose constraints regarding session chair assignments.

Let S be the set of conference sessions, G be the set of working groups and C be the set of available slots[2]. A parallel session can be allocated to multiple slots with a different number of papers allocated to each slot. We denote L the set of authorized quantity of papers that can be allocated to a specific slot for each parallel session. We set n as the maximum number

[1] `https://www.roadef.org/roadef-le-congres-annuel`
[2] To simplify, each session, slot, or working group is referred to by a unique natural number identifier.

Table 1 Summary of ROADEF Data for the last four years. #X denotes the number of X.

Year	2024	2023	2022	2021
#Sessions	40	47	42	27
#Working-groups	20	24	24	17
#Slots	7	7	8	11
Paper range L	{3,4,5,6}		{3,4,5}	
#Papers	307	358	311	182

of parallel sessions for the available slots, which is fixed each year depending on the available resources. Note that, in general, a higher value for this parameter entails more complex logistics and more resources to manage the parallel sessions. For each session $s \in S$, we denote $np(s)$ and $W(s) \subseteq G$ respectively the number of papers and the set of working groups associated to session s. Finally, for each slot $c \in C$, we denote $npMax(c)$ the maximum number of authorized papers for each parallel session in slot c.

For instance, in 2024, the 25^{th} edition of ROADEF[3] was organized over three days in the city of Amiens. The conference comprised 40 programmed sessions[4]. It also involved the participation of 20 distinct working groups. Over the course of the three-day conference, there were seven parallel session slots. The organizing committee has decided that each parallel session was required to host at least three papers and at most six. Indeed, since the allocated presentation and question time per paper is 20 minutes, a session with less than 3 papers, i.e. less than one hour, would be too short while, one with more than 6 would be too long since a break would be expected after 2 hours. The maximum number of parallel sessions per slot can go up to 15 with respect to the available resources and mainly the available rooms in the conference site. For the lack of space, we do not report the maximum number of authorized papers for each slot nor the values of $np(s)$ and $W(s)$ for $s \in S$. However, we mention that there were a total of 307 papers accepted in the 2024 edition, split over the different sessions. To give an example, for session $s = 40$ dedicated to constraint programming, we have $np(s) = 8$ and $W(s) = \{7, 15\}$. The summary of Data from ROADEF over the last four years is reported in Table 1. Finally, we note that additional constraints for specific sessions may arise. In particular, the chairs of session 34 in the 2024 edition requested to assign their session presentations on the last day of the conference covered by slots 5, 6 and 7.

4 Maximum Satisfiability for Conference Session Scheduling

4.1 Basic Model

Our aim is to minimize the conflicts among working groups in the conference schedule while respecting all the schedule-related constraints. We start first by defining our main variables. More specifically, we set two types of variables to represent the allocation of parallel sessions and to compute the conflicts as follows:

- $x_{(s,c,l)}$ is set to true if session $s \in S$ is allocated to slot $c \in C$ with $l \in L$ papers
- $y_{(s_1,s_2,c,g)}$ is set to true if there is a conflict in slot $c \in C$ for working group $g \in G$ associated to the pair of sessions $(s_1, s_2) \in S^2$ such that $s_1 < s_2$ and $g \in W(s_1) \cap W(s_2)$

[3] `https://roadef2024.sciencesconf.org/`

[4] 50 parallel sessions were initially programmed in 2024 but 6 were canceled for lack of submissions and 4 were merged with other sessions for insufficient number of submissions.

Since we want to reduce the number of group conflicts in the conference, we clearly need to minimize the number of satisfied y variables. We can therefore define the set of soft clauses accordingly as follows:

$$\phi_{soft} = \bigwedge_{\substack{(s_1,s_2,c,g)\in S\times S\times C\times G \\ s1<s2 \\ g\in W(s_1)\cap W(s_2)}} \overline{y_{(s_1,s_2,c,g)}}$$

Next, we state the constraints that will be added to the hard clause set ϕ_{hard} of the formula:

- At most one quantity of papers can be chosen for each (session , slot) pair:

$$\sum_{l\in L} x_{(s,c,l)} \leq 1 \qquad \forall (s,c) \in S \times C \tag{1}$$

- The subdivision of a session into slots covers all the papers in the session:

$$\sum_{\substack{c\in C \\ l\in L}} x_{(s,c,l)} * l = np(s) \qquad \forall s \in S \tag{2}$$

- The allocation of a session to a slot c is valid if $npMax(c)$ is not exceeded:

$$\bigwedge_{\substack{l\in L \\ l>npMax(c)}} \overline{x(s,c,l)} \qquad \forall (s,c) \in S \times C \tag{3}$$

- The maximum number of parallel sessions per slot is not exceeded:

$$\sum_{\substack{s\in S \\ l\in L}} x_{(s,c,l)} \leq n \qquad \forall c \in C \tag{4}$$

- Two sessions associated to the same group and allocated to the same slot generate a conflict:

$$\bigwedge_{(l_1,l_2)\in L^2} \overline{x_{(s_1,c,l_1)}} \vee \overline{x_{(s2,c,l_2)}} \vee y_{(s_1,s_2,c,g)} \tag{5}$$

$$\forall (s_1, s_2, c, g) \in S^2 \times C \times G \text{ s.t } s_1 < s_2 \text{ and } g \in WG(s_1) \cap WG(s_2)$$

- Session-specific constraints such as the one which arose for session 34 in the 2024 edition can be written as follows:

$$\bigwedge_{\substack{c\in C\setminus\{5,6,7\} \\ l\in L}} \overline{x_{(34,c,l)}} \tag{6}$$

4.2 Enhanced Model

In order to enhance our model, we first remark that Constraint (5) can be written differently as stated in the following proposition which can be easily proven by the distributive properties of logical conjunction and disjunction.

▶ **Proposition.** *Constraint (5) is logically equivalent to the following constraint:*

$$(\bigwedge_{l\in L} \overline{x_{(s_1,c,l)}}) \vee (\bigwedge_{l\in L} \overline{x_{(s_2,c,l)}}) \vee y_{(s_1,s_2,c,g)}$$

$$\forall (s_1, s_2, c, g) \in S^2 \times C \times G \ s.t \ s_1 < s_2 \ and \ g \in WG(s_1) \cap WG(s_2)$$

As such, we can clearly observe that the information represented by the conjunctions in the clauses may occur redundantly in the constraint. Note that, for a given (session,slot) pair, these conjunctions represent the absence of a session from a given slot. Using these observations, we introduce new variables to our model as follows:

- $z_{(s,c)}$ is set to $true$ if session $s \in S$ is not allocated to slot $c \in C$

To enforce the semantic meaning of these variables, we add the following constraint which can be easily rewritten in CNF form using the CNF rewriting of logical equivalence and the distributivity laws:

- When a session is not allocated to a given slot, the corresponding z variable is set to $true$, otherwise to $false$:

$$z_{(s,c)} \Leftrightarrow \bigwedge_{l \in L} \overline{x_{(s,c,l)}} \qquad \forall (s,c) \in S \times C \tag{7}$$

Now, we can rewrite some of the previous constraints as shown below.

- We can optimize the left-hand side of constraint (4) through the z variables as follows:

$$\sum_{s \in S} \overline{z_{(s,c)}} \leq n \qquad \forall c \in C \tag{4*}$$

- We can also simply remodel constraint (5) based on the rewriting established in the proposition above and by relying on the newly introduced z variables:

$$z_{(s_1,c)} \vee z_{(s_2,c)} \vee y_{(s_1,s_2,c,g)} \quad \forall (s_1,s_2,c,g) \in S^2 \times C \times G, s_1 < s_2 \text{ and } g \in W(s_1) \cap W(s_2) \tag{5*}$$

- Finally, the z variables can be also used to effectively encode session-specific constraints as follows:

$$\bigwedge_{c \in C \setminus \{5,6,7\}} z_{(34,c)} \tag{6*}$$

5 Experimental Evaluation

We have encoded our Max-SAT models[5] using the PySAT[6] library [17]. To compare our enhanced formulation to the basic model, we used the Sorting Network (SortN) encoding which is available for both PB and cardinality constraints in PySAT. Furthermore, to show that our enhanced model can be further improved, we also test a CardN+PBbest combination, with a cardinality encoding for cardinality constraints and the "best" option set for PB constraints in PySAT, which typically invokes the BDD encoding. Using data extracted from the last four years of ROADEF, we have generated a total of 72 instances, by choosing 7 varying values of n for each year. This enables us to not only minimize the number of working-group conflicts for a given n value but to also seek an optimal value for the maximum number of parallel sessions. The features of the generated instances in terms of number of variables and clauses are reported in Table 2. We have solved the generated instances using different state-of-the-art Max-SAT solvers including RC2 [18], OpenWBO (WBO) [26], MaxCDCL [23] and MaxHS [13]; all first-ranked solvers in previous Max-SAT evaluations.

[5] Our code, data and benchmark are available in the following **GitHub repository**
[6] `https://pysathq.github.io/`

Table 2 Summary of instance features and computed optimal values for the whole benchmark. # X denotes number of X. An interval is indicated when there are several values possibles. For the sake of comparison, we report the values obtained through the handmade scheduling by the organizing committees between "()" in the last two rows.

Year	2024		2023		2022		2021	
Encoding	SortN	CardN+PBbest	SortN	CardN+PBbest	SortN	CardN+PBbest	SortN	CardN+PBbest
# Var	45112	11007	49917	[12309-15235]	44212	[15810-12466]	44612	[9337-10173]
# Hard	67754	20208	74926	[22640-27029]	65950	[26122-21106]	66807	[16209-17463]
# Soft	308		518		880		264	
# Conflicts	4 (6)		9 (35)		29 (38)		0 (3)	
Optimum n	10 (11)		13 (14)		11 (11)		5 (5)	

For the sake of comparison, we have also implemented our models using OR-Tools to perform tests with the constraint programming solver CP-SAT [28]. The tests were performed on a machine equipped with an Intel Core i7 processor clocked at 3.80GHz, under Ubuntu 22.04. A timeout of 3600s was set for each instance. The results in terms of solving time are described in Table 3.

First, it is crucial to highlight the substantial enhancements our solutions have achieved over the manually crafted schedules by the organizing committees in recent years. More specifically, our models have remarkably enabled to improve both the number of conflicts and the management of parallel sessions. Indeed, for the years 2024 to 2021, our automated solutions reduced the number of working-group conflicts respectively by 2, 26, 9, and 3. This reduction emphasizes the relevance of our computational approach over traditional manual methods. Furthermore, our formulation has successfully decreased the maximum number of parallel sessions required in the 2024 and 2023 schedules, by reducing it from 11 to 10 in 2024 and from 14 to 13 in 2023. These results enable to significantly lessen the operational complexity and the resources required for organizing the conference and to ensure a smoother and more productive experience for attendees and organizers alike.

Next, we compare our enhanced formulation to the basic one. As showcased in Table 3, the comparison between the two models with Max-SAT solvers for a similar encoding (SortN) clearly supports the relevance of introducing z variables into the model. Across all years, the enhanced model consistently outperforms the basic one in terms of solving times. Notably, in terms of total solving time across all MaxSAT solvers, it achieves a gain of 62.9%, 67.2% and 44.7% respectively for years 2024, 2023 and 2021, with a total gain of 66.3% over the three years. In particular, the solving times were significantly improved for specific solvers as, for instance, more than 93% gain in solving time is achieved for the solver RC2 in 2023 with our enhanced formulation. Furthermore, the enhanced model's ability to produce better feasible solutions is particularly pronounced. Indeed, for the instances of 2023, the model not only enables the MaxCDCL solver to solve some instances within the timeout but also to achieve a feasible solution (which is optimal) for the remaining ones, unlike the basic model where it fails on all the instances. For the instances of 2022, although the optimal solutions are not found within the timeout[7], it is crucial to emphasize that the enhanced model consistently improves the best feasible solutions compared to the basic model. In fact, the optimal values for the number of conflicts is in [27,29] with $n = 11$ showing that the enhanced model is able to achieve a tight optimal solution interval for 2022, with a feasible solution of 29 achieved by

[7] Note that we did not change the internal behavior of the solvers and, in particular, MaxHS which may stop and return the best feasible solution before the timeout if it finds the problem hard to solve and does not manage to achieve optimality

Table 3 Solving times in seconds for the Basic Model with SortN and the Enhanced Model with SortN and CardN+PBbest. For each one, the best solving time for each instance is highlighted in bold. The best overall results for each instance are also underlined. T denotes the occurrence of a timeout. If the solver is not able to compute the optimal solution, the best obtained feasible solution is reported between "()" and, if a lower bound is provided, it is indicated between "[]".

		Basic Model					Enhanced Model					Enhanced Model with CardN+PBbest			
Year	n	RC2	WBO	MaxCDCL	MaxHS	CP-SAT	RC2	WBO	MaxCDCL	MaxHS	CP-SAT	RC2	WBO	MaxCDCL	MaxHS
2024	15	**3.428**	7.811	5.358	16.086	26.800	1.259	1.180	**0.989**	17.933	7.683	0.767	**0.300**	0.467	9.925
	14	**3.054**	12.095	5.716	46.308	26.291	1.300	1.130	**0.818**	14.633	6.677	0.716	0.301	**0.290**	11.236
	13	**3.288**	6.989	5.591	46.206	20.622	1.199	1.404	**0.992**	18.585	7.713	0.770	**0.279**	0.353	12.476
	12	**3.540**	6.036	5.147	95.241	22.691	1.315	1.742	**0.805**	20.607	10.056	0.718	**0.309**	0.313	5.972
	11	**3.652**	12.384	8.746	153.335	22.594	1.287	2.472	**1.100**	71.655	10.390	0.707	0.392	0.395	26.609
	10	**9.902**	1398.111	511.043	869.119	88.831	**9.707**	341.225	150.416	271.440	35.460	**1.544**	411.766	213.257	500.964
2023	17	197.515	**119.674**	T (-)	1187.325	T (-)	**9.687**	17.769	T (9)	357.906	T (9)	10.370	**6.136**	3371.304	353.515
	16	355.627	**71.472**	T (-)	755.975	T (9)	**12.602**	20.701	3058.418	395.716	T (9)	12.145	**4.792**	2866.853	426.965
	15	139.625	**99.332**	T (-)	1106.281	T (9)	**23.467**	41.042	T (9)	408.854	T (9)	15.247	**6.696**	2686.484	405.923
	14	107.202	**102.947**	T (-)	1036.948	T (9)	**14.011**	22.683	T (9)	413.160	T (9)	19.751	**6.706**	T (9)	417.693
	13	250.308	**199.549**	T (-)	853.876	T (9)	**8.006**	14.962	3136.008	439.812	T (9)	8.733	**4.162**	T (9)	438.877
	12	T [10]	T (-)	T (-)	T (28)[8]	T (-)	T (-)[10]	T (-)	T (-)	T (23)[8]	T (10)	T [10]	T (-)	T (-)	T (30)[9]
2022	16	T (-)[24]	T (127)	**T (29)**	385.625 (46)[18]	**T (29)**	T (-)[25]	T (146)	**T (29)**	496.061 (42)[22]	**T (29)**	T (-)[22]	T (83)	**T (29)**	1024.785 (40)[25]
	15	T (-)[24]	T (161)	**T (29)**	380.048 (37)[26]	**T (29)**	T (-)[24]	T (134)	**T (29)**	322.627 (51)[18]	**T (29)**	T (-)[24]	T (58)	**T (29)**	1059.200 (33)[25]
	14	T (-)[25]	T (196)	**T (29)**	T (37)[26]	**T (29)**	T (-)[27]	T (132)	**T (29)**	473.504 (38)[23]	**T (29)**	T (-)[24]	T (47)	**T (29)**	**448.613 (29)[23]**
	13	T (-)[24]	T (222)	**T (29)**	392.299 (53)[18]	**T (29)**	T (-)[27]	T (146)	**T (29)**	348.256 (56)[17]	**T (29)**	T (-)[25]	T (50)	**T (29)**	330.284 (59)[18]
	12	T (-)[24]	T (324)	**T (29)**	856.923 (50)[24]	**T (29)**	T (-)[26]	T (135)	**T (29)**	980.692 (30)[25]	**T (29)**	T (-)[24]	T (71)	**T (29)**	328.658 (59)[18]
	11	T (-)[25]	T (543)	**T (29)**	T (40)[27]	**T (29)**	T (-)[27]	T (436)	**T (29)**	713.466 (56)[18]	**T (29)**	T (-)[24]	T (287)	**T (29)**	650.517 (48)[18]
2021	10	0.670	1.397	**0.503**	0.653	1.356	0.656	0.748	0.179	**0.132**	1.307	0.178	0.139	0.050	**0.027**
	9	1.210	1.380	**0.276**	0.685	1.324	0.309	0.812	0.170	**0.155**	1.316	0.133	0.136	0.038	**0.026**
	8	1.232	1.352	**0.437**	0.682	1.327	0.654	0.775	**0.278**	0.454	1.272	0.132	0.148	0.039	**0.030**
	7	1.189	1.364	**0.262**	0.619	1.333	0.649	0.791	**0.256**	0.448	1.276	0.126	0.132	0.035	**0.030**
	6	0.646	1.384	**0.414**	0.695	1.307	0.350	0.776	**0.281**	0.470	1.274	0.135	0.146	0.035	**0.031**
	5	1.142	1.724	**0.425**	0.696	1.347	0.661	0.885	**0.279**	0.459	1.280	0.142	0.200	0.035	**0.027**

MaxCDCL. Overall, the results provide strong empirical evidence supporting the importance of introducing z variables in the scheduling model. This observation remains true for the CP-SAT solver which also shows superior performance with the enhanced formulation.

Note how CP-SAT achieves a timeout on all the instances of 2022 and 2023 both with the basic and enhanced models and does not manage to outperform the best MaxSAT solver in 2021 and 2024 for all the instances, which highlights the relevance of our MaxSAT approach in managing large-scale scheduling tasks. In addition, for the sake of enhancing MaxSAT solver performance, we have tested the CardN+PBbest combination with the enhanced model and we have obtained promising outcomes. Specifically, this combination demonstrates notable improvements in the best solving times compared to SortN, with a gain of 67.5%, 59.9% and 88.2% respectively for 2024, 2023 and 2021, thus averaging to more than 70% gain over the three years. Finally, note how the instances become particularly difficult for the smallest n within each year. In fact, the choices of the n values were not arbitrary as a formal lower bound can be established by calculating an upper bound of the number of papers for a given n as follows: $UBP_n = \sum_{c \in C} n * npMax(c)$. For instance, in 2024, we have $UBP_9 = 288 < \sum_{s \in S} np(s) = 307$ and, therefore, the the set of hard constraints becomes unsatisfiable for $n = 9$ as it is impossible to schedule all the sessions so as to achieve the total amount of papers. In fact, we have also generated the instances for $n = 9$ in 2024, $n = 11$ in 2023, $n = 10$ in 2022 and $n = 4$ in 2021 totaling to 12 unsatisfiable instances. Unsurprisingly, while CP-SAT is able to prove the infeasibility of these instances, all the MaxSAT solvers failed within the timeout as these solvers are scarcely confronted to instances where the set of hard clauses is unsatisfiable [5]. This underscores the importance of enlarging the benchmarks in Max-SAT evaluations with such instances and, specifically, ones extracted from real-life scenarios such as ours to keep up the driving force in improving modern solvers.

6 Conclusion

In this paper, we addressed an optimization problem in conference session scheduling. More specifically, we presented a Max-SAT formulation for the minimization of working-group conflicts in the ROADEF conference, which we then enhanced by introducing new variables. Our results indicate that the number of conflicts and the maximum number of parallel sessions could be reduced and a scheduling could be computed in an effective time frame. Our work thus holds great promise in simplifying the scheduling task, streamlining logistics and optimizing resources in subsequent editions of the conference. As future work, we plan to expand our research by conducting experiments on additional datasets, both within and beyond the scope of ROADEF. Additionally, we aim to enhance our scheduling approach by incorporating considerations for author conflicts so as to develop a more comprehensive optimization framework that accounts for the intricate relationships among participants.

References

1 Ignasi Abío, Robert Nieuwenhuis, Albert Oliveras, and Enric Rodríguez-Carbonell. BDDs for Pseudo-Boolean Constraints - Revisited. In Karem A. Sakallah and Laurent Simon, editors, *14th International Conference on Theory and Applications of Satisfiability Testing - SAT 2011, Ann Arbor, MI, USA, June 19-22, 2011. Proceedings*, volume 6695 of *Lecture Notes in Computer Science*, pages 61–75. Springer, 2011. doi:10.1007/978-3-642-21581-0_7.

2 Carlos Ansótegui, Maria Luisa Bonet, and Jordi Levy. SAT-based MaxSAT algorithms. *Artif. Intell.*, 196:77–105, 2013. doi:10.1016/J.ARTINT.2013.01.002.

3 Roberto Asín, Robert Nieuwenhuis, Albert Oliveras, and Enric Rodríguez-Carbonell. Cardinality Networks and Their Applications. In Oliver Kullmann, editor, *Theory and Applications of Satisfiability Testing - SAT 2009, 12th International Conference, SAT 2009, Swansea, UK, June 30 - July 3, 2009. Proceedings*, volume 5584 of *Lecture Notes in Computer Science*, pages 167–180. Springer, 2009. doi:10.1007/978-3-642-02777-2_18.

4 Roberto Javier Asín Achá and Robert Nieuwenhuis. Curriculum-based course timetabling with SAT and MaxSAT. *Ann. Oper. Res.*, 218(1):71–91, 2014. doi:10.1007/S10479-012-1081-X.

5 Fahiem Bacchus, Matti Järvisalo, and Ruben Martins. MaxSAT Evaluation 2018: New Developments and Detailed Results. *J. Satisf. Boolean Model. Comput.*, 11(1):99–131, 2019. doi:10.3233/SAT190119.

6 Fahiem Bacchus, Matti Järvisalo, and Ruben Martins. Maximum satisfiability. In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability - Second Edition*, volume 336 of *Frontiers in Artificial Intelligence and Applications*, pages 929–991. IOS Press, 2021. doi:10.3233/FAIA201008.

7 Jeremias Berg, Bart Bogaerts, Jakob Nordström, Andy Oertel, and Dieter Vandesande. Certified Core-Guided MaxSAT Solving. In Brigitte Pientka and Cesare Tinelli, editors, *Automated Deduction - CADE 29 - 29th International Conference on Automated Deduction, Rome, Italy, July 1-4, 2023, Proceedings*, volume 14132 of *Lecture Notes in Computer Science*, pages 1–22. Springer, 2023. doi:10.1007/978-3-031-38499-8_1.

8 Jeremias Berg, Antti Hyttinen, and Matti Järvisalo. Applications of MaxSAT in Data Analysis. In Daniel Le Berre and Matti Järvisalo, editors, *Proceedings of Pragmatics of SAT 2015, Austin, Texas, USA, September 23, 2015 / Pragmatics of SAT 2018, Oxford, UK, July 7, 2018*, volume 59 of *EPiC Series in Computing*, pages 50–64. EasyChair, 2018. doi:10.29007/3QKH.

9 A. Biere, M. Heule, and H. van Maaren. *Handbook of Satisfiability: Second Edition*. Frontiers in Artificial Intelligence and Applications. IOS Press, 2021. URL: https://books.google.fr/books?id=dUAvEAAAQBAJ.

10 Miquel Bofill, Marc Garcia, Josep Suy, and Mateu Villaret. MaxSAT-Based Scheduling of B2B Meetings. In Laurent Michel, editor, *Integration of AI and OR Techniques in Constraint Programming - 12th International Conference, CPAIOR 2015, Barcelona, Spain, May 18-22, 2015, Proceedings*, volume 9075 of *Lecture Notes in Computer Science*, pages 65–73. Springer, 2015. doi:10.1007/978-3-319-18008-3_5.

11 Mohamed Sami Cherif, Djamal Habet, and André Abramé. Understanding the power of max-sat resolution through up-resilience. *Artif. Intell.*, 289:103397, 2020. doi:10.1016/J.ARTINT.2020.103397.

12 Sami Cherif, Heythem Sattoutah, Chu-Min Li, Corinne Lucet, and Laure Brisoux-Devendeville. ROADEF_SCHEDULING. Software, swhId: swh:1:dir:7083377094f69163d37d30b77d740c72c562139d (visited on 2024-08-16). URL: https://github.com/satoutahhaithem/ROADEF_SCHEDULING.

13 Jessica Davies and Fahiem Bacchus. Solving MAXSAT by Solving a Sequence of Simpler SAT Instances. In Jimmy Ho-Man Lee, editor, *Principles and Practice of Constraint Programming - CP 2011 - 17th International Conference, CP 2011, Perugia, Italy, September 12-16, 2011. Proceedings*, volume 6876 of *Lecture Notes in Computer Science*, pages 225–239. Springer, 2011. doi:10.1007/978-3-642-23786-7_19.

14 Emir Demirovic and Nysret Musliu. MaxSAT-based large neighborhood search for high school timetabling. *Comput. Oper. Res.*, 78:172–180, 2017. doi:10.1016/J.COR.2016.08.004.

15 Emir Demirovic, Nysret Musliu, and Felix Winter. Modeling and solving staff scheduling with partial weighted maxSAT. *Ann. Oper. Res.*, 275(1):79–99, 2019. doi:10.1007/S10479-017-2693-Y.

16 Niklas Eén and Niklas Sörensson. Translating Pseudo-Boolean Constraints into SAT. *J. Satisf. Boolean Model. Comput.*, 2(1-4):1–26, 2006. doi:10.3233/SAT190014.

17 Alexey Ignatiev, António Morgado, and João Marques-Silva. PySAT: A Python Toolkit for Prototyping with SAT Oracles. In Olaf Beyersdorff and Christoph M. Wintersteiger, editors, *Theory and Applications of Satisfiability Testing - SAT 2018 - 21st International Conference, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 9-12, 2018, Proceedings*, volume 10929 of *Lecture Notes in Computer Science*, pages 428–437. Springer, 2018. doi:10.1007/978-3-319-94144-8_26.

18 Alexey Ignatiev, António Morgado, and João Marques-Silva. RC2: an Efficient MaxSAT Solver. *J. Satisf. Boolean Model. Comput.*, 11(1):53–64, 2019. doi:10.3233/SAT190116.

19 Gerold Jäger, Sharlee Climer, and Weixiong Zhang. The complete parsimony haplotype inference problem and algorithms based on integer programming, branch-and-bound and Boolean satisfiability. *J. Discrete Algorithms*, 37:68–83, 2016. doi:10.1016/J.JDA.2016.06.001.

20 Alexandre Lemos, Filipe Gouveia, Pedro T. Monteiro, and Inês Lynce. Iterative Train Scheduling under Disruption with Maximum Satisfiability. *J. Artif. Intell. Res.*, 79:1047–1090, 2024. doi:10.1613/JAIR.1.14924.

21 Chu Min Li and Felip Manyà. MaxSAT, Hard and Soft Constraints. In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability - Second Edition*, volume 336 of *Frontiers in Artificial Intelligence and Applications*, pages 903–927. IOS Press, 2021. doi:10.3233/FAIA201007.

22 Chu-Min Li, Zhenxing Xu, Jordi Coll, Felip Manyà, Djamal Habet, and Kun He. Combining clause learning and branch and bound for maxsat. In Laurent D. Michel, editor, *27th International Conference on Principles and Practice of Constraint Programming, CP 2021, Montpellier, France (Virtual Conference), October 25-29, 2021*, volume 210 of *LIPIcs*, pages 38:1–38:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPICS.CP.2021.38.

23 Chu-Min Li, Zhenxing Xu, Jordi Coll, Felip Manyà, Djamal Habet, and Kun He. Boosting branch-and-bound MaxSAT solvers with clause learning. *AI Commun.*, 35(2):131–151, 2022. doi:10.3233/AIC-210178.

24 Xiaojuan Liao, Hui Zhang, Miyuki Koshimura, Rong Huang, and Wenxin Yu. Maximum Satisfiability Formulation for Optimal Scheduling in Overloaded Real-Time Systems. In Abhaya C. Nayak and Alok Sharma, editors, *PRICAI 2019: Trends in Artificial Intelligence - 16th Pacific Rim International Conference on Artificial Intelligence, Cuvu, Yanuca Island, Fiji, August 26-30, 2019, Proceedings, Part I*, volume 11670 of *Lecture Notes in Computer Science*, pages 618–631. Springer, 2019. doi:10.1007/978-3-030-29908-8_49.

25 Hratch Mangassarian, Andreas G. Veneris, and Farid N. Najm. Maximum circuit activity estimation using pseudo-boolean satisfiability. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, 31(2):271–284, 2012. doi:10.1109/TCAD.2011.2169259.

26 Ruben Martins, Vasco M. Manquinho, and Inês Lynce. Open-WBO: A Modular MaxSAT Solver. In Carsten Sinz and Uwe Egly, editors, *Theory and Applications of Satisfiability Testing - SAT 2014 - 17th International Conference, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 14-17, 2014. Proceedings*, volume 8561 of *Lecture Notes in Computer Science*, pages 438–445. Springer, 2014. doi:10.1007/978-3-319-09284-3_33.

27 António Morgado, Federico Heras, Mark H. Liffiton, Jordi Planes, and João Marques-Silva. Iterative and core-guided MaxSAT solving: A survey and assessment. *Constraints An Int. J.*, 18(4):478–534, 2013. doi:10.1007/S10601-013-9146-2.

28 Laurent Perron and Frédéric Didier. Cp-sat. URL: https://developers.google.com/optimization/cp/cp_solver/.

29 Matthieu Py, Mohamed Sami Cherif, and Djamal Habet. Proofs and Certificates for Max-SAT. *J. Artif. Intell. Res.*, 75:1373–1400, 2022. doi:10.1613/JAIR.1.13811.

30 Olivier Roussel and Vasco M. Manquinho. Pseudo-Boolean and Cardinality Constraints. In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability - Second Edition*, volume 336 of *Frontiers in Artificial Intelligence and Applications*, pages 1087–1129. IOS Press, 2021. doi:10.3233/FAIA201012.

31 Ahmad Shabani and Bijan Alizadeh. PMTP: A MAX-SAT-Based Approach to Detect Hardware Trojan Using Propagation of Maximum Transition Probability. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, 39(1):25–33, 2020. doi:10.1109/TCAD.2018.2889663.