International Journal of Computational Intelligence Systems (2022) 15:63
https://doi.org/10.1007/s44196-022-00121-5

RESEARCH ARTICLE

The Impact of Implied Constraints on MaxSAT B2B Instances

Miquel Bofill[1] · Jordi Coll[2] · Jesús Giráldez-Cru[3] · Josep Suy[1] · Mateu Villaret[1]

Received: 26 February 2022 / Accepted: 25 July 2022

Abstract
The B2B scheduling optimization problem consists of finding a schedule of a set of meetings between pairs of participants, minimizing their number of idle time periods. Recent works have shown that SAT-based approaches are state-of-the-art on this problem. One interesting feature of such approaches is the use of implied constraints. In this work, we provide an experimental setting to study the impact of using these implied constraints in MaxSAT B2B instances. To this purpose and due to the reduced number of existing real-world B2B instances, we propose a random B2B instance generation model, which reproduces certain features of these problems. In our experimental analysis, we show that the impact of using some implied constraints in the MaxSAT encodings depends on the characteristics of the problem, and we also analyze the benefits of combining them. Finally, we give some insights on how a MaxSAT solver is able to exploit these implied constraints.

Keywords Timetabling · MaxSAT · Implied constraints

Abbreviations

B2B	Business-to-business
B2BSOP	B2B scheduling optimization problem
`noimp`	B2BSOP MaxSAT encoding with no implied constraints
`imp1`	B2BSOP MaxSAT encoding with implied constraint 1 (see Eq. 17)
`imp2`	B2BSOP MaxSAT encoding with implied constraint 2 (see Eq. 18)
`imp12`	B2BSOP MaxSAT encoding with both implied constraints
PAR	Penalized average runtime

✉ Jesús Giráldez-Cru
jgiraldez@ugr.es

Miquel Bofill
mbofill@imae.udg.edu

Jordi Coll
jordi.coll@lis-lab.fr

Josep Suy
suy@imae.udg.edu

Mateu Villaret
villaret@imae.udg.edu

[1] IMAE, Universitat de Girona, Girona, Spain

[2] Aix Marseille Univ, Université de Toulon, CNRS, LIS, Marseille, France

[3] DaSCI, DECSAI, Universidad de Granada, Granada, Spain

1 Introduction

Business-to-business (B2B) events consist of meetings between people with similar interests. They are typically held in business networking events. The B2B scheduling problem is the problem of finding a feasible schedule for a set of requested meetings between pairs of participants, subject to participants' availability and accommodation capacity. To the best of our knowledge, there are a few works dealing with this problem [4–8, 12].

The B2B scheduling optimization problem (B2BSOP) considered in this work was introduced in [5]. It consists of finding a feasible B2B schedule that minimizes the number of idle time periods in the participants' schedule. Side constraints are added to ensure fairness among participants, e.g., by restricting the maximum difference in the number of idle time periods between participants' schedules. A further refinement could consist of (additionally or alternatively) minimizing the duration of those idle time periods.

The B2BSOP has been successfully addressed by means of relatively straightforward CP and Pseudo-Boolean formulations [5], and further investigated with specialized CP and MIP formulations [12], and with partial MaxSAT specialized encodings [6]. A thorough comparison of all these models is analyzed in [4].

In this work, we focus on the MaxSAT encoding for the B2BSOP [4, 6]. This encoding represents B2B instances as partial MaxSAT formulas [10], where some clauses are

Published online: 11 August 2022

marked as hard whereas others are marked as soft, and the goal is to find an assignment to the Boolean variables of the propositional formula that satisfies all hard clauses and falsifies the minimum number of soft clauses. In this encoding, the falsification of a soft clause represents the existence of an idle time period for some participant. This MaxSAT encoding clearly dominates the other existing formulations on real-world instances, in which, moreover, the addition of implied constraints appears to be beneficial [4].

There exist some works in the literature showing the benefits of using implied constraints or redundant encodings in SAT [2, 3, 9]. However, the lack of varied real-world B2B instances makes it difficult to hold this claim for the problem at hand. In this paper, we try to overcome this limitation by providing a parameterized random generator of B2B problem instances. The development of random problem generators sharing the majority of real-world instance features was already stated in [13] as one of the most important challenges in propositional search. The main goal of these generators is two-sided. On the one hand, these generators allow to increment the set of available instances sharing the main characteristics of the existing ones. On the other hand, these random models allow us to create instances with any desired property given an appropriate parametrization.

In this work, we present a generator of random B2B instances, which is parameterized in several ways to reproduce certain characteristics of real-world B2B problems. This allows us to get some insights on the benefits of using implied constraints to solve B2B instances, depending on the characteristics of the problem. In particular, we show that the impact on the performance of the MaxSAT solver (in terms of solving time) when those implied constraints are used, depends on the *density* and the *shape* of the problem. By *density* we refer to the ratio between the number of meetings and the accommodation capacity. By *shape* we refer to the configuration of the accommodation capacity, i.e., the ratio between time slots and locations. Following the insights observed in random B2B instances, we also analyze the impact that using those implied constraints has on solving real-world MaxSAT B2B instances. Finally, we study the performance of a MaxSAT solver solving B2B problems and observe that the existence of these implied constraints helps to improve its performance, since its branching heuristic focuses on the variables of those implied constraints, and this reduces the solving times. This is a revisited and extended version of the work presented in [7]. In particular, the analysis of the impact of implied constraints on real-world B2B instances and on the performance of the MaxSAT solver is completely new.

The rest of this work is organized as follows. In Sect. 2 we provide the definition of the B2BSOP and a MaxSAT encoding for such a problem. In Sect. 3 we present the proposed random generation model of B2B instances. Section 4 describes the results of our experimental evaluation. Finally, we conclude in Sect. 5.

2 Definitions

In this section we precisely reproduce the definition of the B2BSOP and a MaxSAT encoding of it [4].

Definition 1 (B2BSOP-h) Given a set of participants $\mathscr{P}$, a list of time slots $\mathscr{T}$, a set of available locations $\mathscr{L}$ and a set of meetings $\mathscr{M} \subset \mathscr{P} \times \mathscr{P}$ between pairs of participants to be scheduled, the *B2B scheduling Optimization Problem with homogeneity h* consists of finding a total mapping from $\mathscr{M}$ to $\mathscr{T} \times \mathscr{L}$, minimizing the total number of idle time periods and such that:

(i) Each participant has at most one meeting scheduled in each time slot.
(ii) At most one meeting is scheduled in a given time slot and location.
(iii) The difference between the number of idle time periods among all participants is at most h, where by an *idle time period* we refer to a group of consecutive idle time slots between two consecutive meetings involving the same participant.

Notice that this definition does not consider as idle time periods any free time slot before (resp. after) the first (resp. last) meeting scheduled for a participant.

Further constraints like to consider meetings fixed in a certain time slot or session (i.e., group of consecutive time slots), meeting precedences, and forbidden time slots for a certain participant, among others, are analyzed in [4]. In this work, however, we mainly focus on the basic definition of the B2BSOP as stated above.[1]

Parameters. Each instance is defined by the following parameters: $|\mathscr{M}|$ is the number of meetings, $|\mathscr{T}|$ is the number of available time slots, $|\mathscr{L}|$ is the number of available locations, and $|\mathscr{P}|$ is the number of participants. The function $meetings : \mathscr{P} \to \mathbb{P}(\mathscr{M})$ represents the set of meetings involving each participant, i.e., $meetings(p) = \{m \in \mathscr{M} \mid p \in m\}$ for any participant $p \in \mathscr{P}$ (where $\mathbb{P}(\mathscr{M})$ is the powerset of $\mathscr{M}$).

Variables. We define the following propositional variables:

- $schedule_{i,j}$: meeting i is held in time slot j.
- $usedSlot_{p,j}$: participant p has a meeting scheduled in time slot j.

[1] Some real-world B2B instances analyzed in Sect. 4 do have fixed meetings and forbidden time slots for some participants.

- $meetingHeld_{p,j}$: participant p has a meeting scheduled at or before time slot j.
- $endHole_{p,j}$: participant p has an idle time period finishing at time slot j.

We also use some auxiliary variables that will be introduced when needed.

In what follows, we use several Boolean global constraints with the following meaning: $exactly(k, S)$ states that exactly k variables in S are set to true, $atMost(k, S)$ states that at most k variables in S are set to true, and $sortingNetwork(S, sl)$ states that sl is the sorted list of Boolean variables of S. These constraints can be encoded in several ways into SAT, being *cardinality networks* [1] the most promising approach on this problem, according to the experiments in [4].

Constraints. All constraints are hard if not explicitly defined as soft constraints.

At most one meeting involving the same participant is scheduled in each time slot.

$$atMost(1, \{schedule_{i,j} \mid i \in meetings(p)\}) \quad \forall p \in \mathscr{P}, j \in \mathscr{T} \tag{1}$$

Each meeting is scheduled in exactly one time slot.

$$exactly(1, \{schedule_{i,j} \mid j \in \mathscr{T}\}) \qquad \forall i \in \mathscr{M} \tag{2}$$

The number of meetings scheduled in a give time slot is bounded by the number of available locations.

$$atMost(|\mathscr{L}|, \{schedule_{i,j} \mid i \in \mathscr{M}\}) \qquad \forall j \in \mathscr{T} \tag{3}$$

To be able to minimize the number of idle time periods we introduce channeling constraints between the variables *schedule*, *usedSlot* and *meetingHeld*.

If a meeting is scheduled in a certain time slot, then that time slot is used by both meeting participants.

$$schedule_{i,j} \rightarrow (usedSlot_{p_i^1,j} \wedge usedSlot_{p_i^2,j}) \;\; \forall i \in \mathscr{M}, j \in \mathscr{T} \tag{4}$$

where p_i^1 and p_i^2 are the two participants of meeting i.

In the reverse direction, if a time slot is used by some participant, then one of the meetings of that participant is scheduled in that time slot.

$$usedSlot_{p,j} \rightarrow \bigvee_{i \in meetings(p)} schedule_{i,j} \qquad \forall p \in \mathscr{P}, j \in \mathscr{T} \tag{5}$$

For each participant p and time slot j, $meetingHeld_{p,j}$ is true if and only if participant p has had a meeting at or before time slot j.

$$\neg usedSlot_{p,1} \rightarrow \neg meetingHeld_{p,1} \forall p \in \mathscr{P} \tag{6}$$

$$(\neg meetingHeld_{p,j-1} \wedge \neg usedSlot_{p,j}) \rightarrow \neg meetingHeld_{p,j} \forall p \in \mathscr{P}, j \in \mathscr{T} \setminus \{1\} \tag{7}$$

$$usedSlot_{p,j} \rightarrow meetingHeld_{p,j} \forall p \in \mathscr{P}, j \in \mathscr{T} \tag{8}$$

$$meetingHeld_{p,j-1} \rightarrow meetingHeld_{p,j} \forall p \in \mathscr{P}, j \in \mathscr{T} \setminus \{1\} \tag{9}$$

Optimization. Minimization of the number of idle time periods is achieved by means of soft constraints. An idle time period can be identified when a participant has no meeting in a certain time slot, has a meeting in the next time slot, and has had a meeting before. We reify this situation with the *endHole* variables.

Variable $endHole_{p,j}$ is true if and only if participant p has an idle time period finishing at time slot j.

$$endHole_{p,j} \leftrightarrow (\neg usedSlot_{p,j} \wedge meetingHeld_{p,j} \wedge usedSlot_{p,j+1}) \qquad \forall p \in \mathscr{P}, j \in \mathscr{T} \setminus \{|\mathscr{T}|\} \tag{10}$$

Variables $sortedHole_{p,1}, \ldots, sortedHole_{p,\lfloor(|\mathscr{T}|-1)/2\rfloor}$ are the unary representation of the number of idle time periods of each participant p.[2]

$$sortingNetwork([endHole_{p,j} \mid j \in \mathscr{T}], [sortedHole_{p,j} \mid j \in \mathscr{T}]) \qquad \forall p \in \mathscr{P} \tag{11}$$

[Soft Constraint] *A participant does not have idle time periods.*

$$\neg sortedHole_{p,j} \qquad \forall p \in \mathscr{P}, j \in 1..\lfloor(|\mathscr{T}| - 1)/2\rfloor \tag{12}$$

Homogeneity. We find the maximum and minimum number of idle time periods among all participants, and enforce homogeneity by bounding their difference.

(An approximation to) The unary representation of the maximum and minimum number of idle time periods among all participants are respectively $max_1, \ldots, max_{\lfloor(|\mathscr{T}|-1)/2\rfloor}$ and $min_1, \ldots, min_{\lfloor(|\mathscr{T}|-1)/2\rfloor}$, as defined by the following constraints.

$$sortedHole_{p,j} \rightarrow max_j \quad \forall p \in \mathscr{P}, j \in 1..\lfloor(|\mathscr{T}| - 1)/2\rfloor \tag{13}$$

$$\neg sortedHole_{p,j} \rightarrow \neg min_j \quad \forall p \in \mathscr{P}, j \in 1..\lfloor(|\mathscr{T}| - 1)/2\rfloor \tag{14}$$

The difference between the maximum and minimum number of idle time periods can be at most h.

[2] Notice that the number of possible idle time periods of each participant is strictly smaller than the half of time slots since every idle time period requires two meetings scheduled before and after it.

$$dif_j \leftrightarrow min_j \; XOR \; max_j \qquad \forall j \in 1..\lfloor(|\mathscr{T}| - 1)/2\rfloor \tag{15}$$

$$atMost(h, \{dif_j \mid j \in 1..\lfloor(|\mathscr{T}| - 1)/2\rfloor\}) \tag{16}$$

Implied Constraints: From the constraints defined for the problem, the following two sets of implied constraints on *usedSlot* variables are identified:

(1) The number of meetings of a participant p as derived from $usedSlot_{p,j}$ *variables must match the total number of meetings of p.*

$$exactly(|meetings(p)|, \{usedSlot_{p,j} \mid j \in \mathscr{T}\}) \;\; \forall p \in \mathscr{P} \tag{17}$$

(2) The number of participants having a meeting in a given time slot is bounded by twice the number of available locations.

$$atMost(2 \times |\mathscr{L}|, \{usedSlot_{p,j} \mid p \in \mathscr{P}\}) \;\; \forall j \in \mathscr{T} \tag{18}$$

Although other implied constraints can be defined [4], their impact in practice seems to be limited. In particular, the implied constraints analyzed in this work help to solve a number of B2B problems (that would remain unsolved otherwise), whereas additional implied constraints only help to (slightly) reduce their solving times. We conjecture that this phenomenon is due to the ability of the analyzed implied constraints to detect that a partial schedule is already infeasible in earlier stages. For this reason, we do not consider other additional constraints in our study.

As shown by the experimental evaluation presented in [4], using the implied constraints defined in Equations 17 and 18 is in general beneficial. However, it remains unclear the precise contribution of each implied constraint in the observed improvements on the performance of the MaxSAT solver, due to the limited number of existing B2B instances and their heterogeneity.

3 A Random B2B Instances Generator

In this section we present a model for the generation of random B2B problems. To model these problems, we consider a number of participants $P = |\mathscr{P}|$ and a number of meetings $M = |\mathscr{M}|$. A key question is how these M meetings are distributed among these P participants.

The set of B2B instances provided in [6] consists of 20 instances: five instances obtained from real-world data, plus 15 crafted instances derived from the real ones. As we cannot draw general conclusions about probability distributions from this reduced set of instances, we present a simple model to generate random B2B instances. This model, called *B2Brand*, is based on an uniform probability distribution of meetings between pairs of participants in $\mathscr{P}$.

Definition 2 (B2Brand) Let P be a number of participants, with $P > 1$, and M, T and L the number of meetings, time slots, and locations, respectively. A random B2B instance from the B2Brand model is a set of M distinct meetings, independently selected by randomly choosing with uniform probability two distinct participants, whose identifiers range from 1 to P. Additionally, the instance must satisfy the following necessary feasibility conditions.

- The number of meetings M is bounded by the maximum combinations of two participants:

$$M \leq \binom{P}{2} \tag{19}$$

- The number of meetings M is bounded by the event capacity:

$$M \leq T \cdot L \tag{20}$$

- Each participant can have at most T meetings:

$$|meetings(p)| \leq T \qquad \forall p \,.\, 1 \leq p \leq P \tag{21}$$

  where $|meetings(p)|$ is the number of meetings of participant p.
- The number of meetings M is bounded by the combinations of participants and time slots:

$$M \leq \frac{P \cdot T}{2} \tag{22}$$

For the sake of simplicity, we assume a single meeting for each pair of participants.

Notice that the last condition can be derived from Eq. (21) by summing up the number of meetings of every participant.

The model is parametric in the number of participants P and in the number of meetings M. However, it also needs the number of time slots T and locations L to check the feasibility conditions defined. Notice that if any of those conditions is not satisfied, the B2B instance is trivially unsatisfiable. In particular, we define these feasibility conditions within the model because if the number of meetings is close to $\binom{P}{2}$ but $T \ll P$, then there would exist (with high probability) a number of participants requesting more meetings than available time slots, and hence it would not be possible to schedule all of them i.e., the problem will be unsatisfiable. Forbidding this case, the extreme instance with $M \approx P \cdot T/2$ and $P \gg T$ will contain many participants requesting T meetings, i.e., many observations close to the maximum. In our experience, we have found this kind of participant to be very frequent in the real instances from [6].

Note also that the previous feasibility conditions do not guarantee the instance to be feasible. Let us consider for example a B2B instance with three participants, the three possible meetings between them, two time slots, and two

locations. In this problem, the four feasibility conditions are satisfied. However, it is easy to see that the instance is unfeasible.

In what follows, we use this model to generate random families of B2B instances, and we study the performance of the solving process on the use of the implied constraints. This study aims to check whether it is beneficial to use them or not, and in which situations it is more useful to use one or another. In particular, we focus our study on two features of the problem: the *density* and the *shape*.

Definition 3 (Density) Given a positive number of meetings M, time slots T and locations L, the density d of a B2B instance is the relation between the number of meetings M and the accommodation capacity $T \cdot L$,

$$d = \frac{M}{T \cdot L} \tag{23}$$

Notice that if the density of a B2B instance is greater than one, the instance is necessarily unfeasible. The opposite is not true. For example, the density of the instance mentioned before with 3 participants, the three possible meetings between them, two time slots, and two locations is $d = 3/4$, but this instance is unfeasible. In fact, increasing the number of locations, i.e., decreasing the density, does not modify its feasibility. Therefore, unfeasible instances with density smaller than one do exist.

Definition 4 (Shape) Given the accommodation capacity $T \cdot L$, the shape s of a B2B instance is defined as the relation between the number of time slots T and the number of locations L,

$$s = \frac{T}{L} \tag{24}$$

Finally, we remark that this model does not represent all features of real-world instances. For instance, using this model we cannot generate a large number of participants requesting a number of meetings far from the mean. In particular, there could exist *passive* participants in some B2B events. These participants are characterized by requesting no meetings (i.e, they attend the event because other participants request meetings with them). In this case, using a different probability distribution to randomly select the two participants of each meeting, rather than uniform, would probably be a more suitable choice. However, our model seems adequate to model B2B instances similar to the known real-world ones.

Another simplification of the B2Brand model is related with secondary requirements of the problem, such as forbidden time slots and fixed meetings or sessions, among others. In random B2B instances generated with the B2Brand model, we do not consider this kind of additional constraints.

4 Experimental Evaluation

In this section, we evaluate the impact on the solver performance of the use of implied constraints in MaxSAT B2B instances. To do so, we first generate a benchmark of random B2B instances,[3] and we solve them with and without using implied constraints. The goal of these experiments is to identify those cases for which the use of some implied constraint is beneficial. Then, we validate these observations of random B2B instances on real-world instances. Finally, we conjecture the reasons for the success of using implied constraints based on some observations from the MaxSAT solver.

4.1 Experimental Setup

In the experiments, we generated a set of random B2B instances differing in the configuration of their density and shape. Each configuration of density and shape contains 10 random instances, and a total of 36 different configurations were generated, resulting in a total of 1440 different random MaxSAT B2B instances. All experiments are run with a timeout of 2 h (7200 s). As in [6], a value of homogeneity $h = 2$ was used.[4] Each B2B instance was encoded without using implied constraints, as well as with implied constraint 1 (Eq. 17), implied constraint 2 (Eq. 18), and both of them. From now on, we refer to these methods as `noimp`, `imp1`, `imp2` and `imp12`, respectively. In our analysis, we use the Open-WBO [11] MaxSAT solver. This is a well-known MaxSAT solver which has been ranked as one of the best (non-portfolio) solvers in the industrial partial MaxSAT tracks of the last MaxSAT Evaluations. The experiments have been executed on Intel Xeon E3-1220v2 machines at 3.10 GHz with 8 GB of RAM.

In our analysis, we represent the Penalized Average Runtime 1 (PAR1), which is the average of the runtime used to solve the set of instances in a configuration, assigning to unsolved instances the used timeout (i.e., 7200 s). We use PAR1 due to the low number of timeouts. For each configuration, we depict a box-and-whisker plot, which represents the maximum, minimum, median, and quartiles 1 and 3 of the runtimes. In the analysis of real-world B2B instances, where the number of timeouts increases, we also consider the PAR10 runtime (i.e., assigning 10 times the timeout to unsolved instances).

[3] The generator can be found at https://www.ugr.es/~jgiraldez/.

[4] We also tested, instead of using homogeneity (i.e., the maximum difference between the number of idle periods of every two participants), to use an upper-bound on that number, but we did not observe any difference neither in the optimums nor in the solver performance.

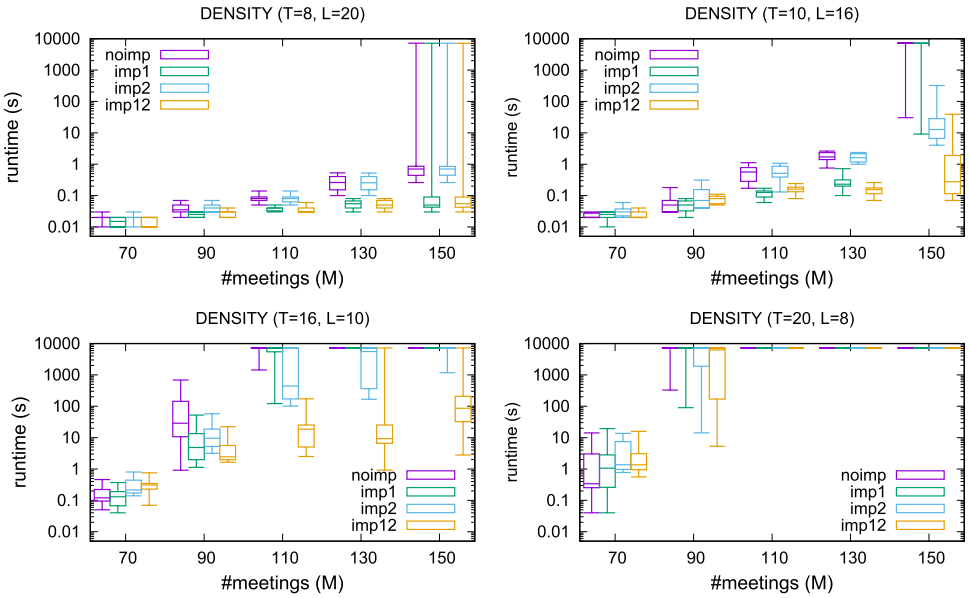

Fig. 1 PAR1 (in seconds) of solving some random B2B families of instances, with and without using implied constraints, varying their density d. Instances are generated using the B2Brand model with $P = 40$, $M = \{70, 90, 110, 130, 150\}$, and (i) $T = 8$ and $L = 20$ (top left), (ii) $T = 10$ and $L = 16$ (top right), (iii) $T = 16$ and $L = 10$ (bottom left), and (iv) $T = 20$ and $L = 8$ (bottom right)

4.2 Random B2B Instances

Based on the definitions of density and shape, we can easily create families of random B2B problems by setting different values to the parameters of the B2Brand model. In all the experiments, the random B2B instances have $P = 40$ participants, which is a realistic number of participants according to existing real-world B2B instances [4]. In a first batch of experiments, we analyze the hardness of random B2B problems with respect to their density. To this purpose, we fix the number of time slots T and the number of locations L, and generate several families of B2B random instances varying the number of meetings M. We do the same experiment for four different combinations of T and L (i.e., four different shapes), where the product $T \cdot L$ is the same. Therefore, we represent the same densities in the four plots (since we evaluate the same values of M and the product $T \cdot L$ is always the same). In particular, we generate the following B2B random instances: $P = 40, M = \{70, 90, 110, 130, 150\}$, and (i) $T = 20$ and $L = 8$, (ii) $T = 16$ and $L = 10$, (iii) $T = 10$ and $L = 16$, and (iv) $T = 8$ and $L = 20$. Figure 1 shows the results of this experiment.

We first observe that the lower the density is (i.e., the lower number of meetings M), the easier the instance becomes for the four encodings. This phenomenon is more clear as the shape gets higher (i.e., with a high number of time slots T w.r.t. the number of locations L). This happens in the four shapes analyzed. This is expected since finding the optimum schedule of a lower number of meetings seems to be an easier task.

A second observation related with the encodings is that the encoding without implied constraints `noimp` always performs worse than the other three, whereas the encoding with both implied constraints `imp12` always shows the best performance.

Interestingly, for the encodings with only one implied constraint (`imp1` and `imp2`), we can distinguish some cases where one performs better than the other. In particular, when the density is low, the encoding with the first implied constraint `imp1` seems to perform faster. See, for instance, the case with $M = 90$, $T = 16$ and $L = 10$ (bottom left plot in Fig. 1), for which `imp1` is clearly faster than `imp2`. On the contrary, when the density is high, `imp2` seems to outperform `imp1`. See, for instance, the case with $M = 150$, $T = 10$ and $L = 16$ (top right plot in Fig. 1), where `imp2` solves the 10 instances of the family whilst `imp1` only solves two. We summarize these results in Observation 1.

Observation 1 Using the implied constraint 1 (`imp1`) seems to be more interesting when the density is small. On the contrary, the implied constraint 2 (`imp2`) seems to be more relevant when the density is high. Overall, using both implied constraints (`imp12`) is always beneficial.

In a second batch of experiments, we analyze the hardness of random B2B instances with respect to their shape. To this purpose, we fix the number of meetings M and the number of time slots T, and generate several families of B2B random instances varying the number of locations L. We do the same experiment for four different values of M (i.e., four different densities). Notice that we represent the same shapes in the four plots. In particular, we consider the following cases: $P = 40$, $T = 20$, $L = \{8, 12, 16, 20\}$, and (i) $M = 80$, (ii) $M = 100$, (iii) $M = 130$, and (iv) $M = 150$. Figure 2 shows the results of this experiment.

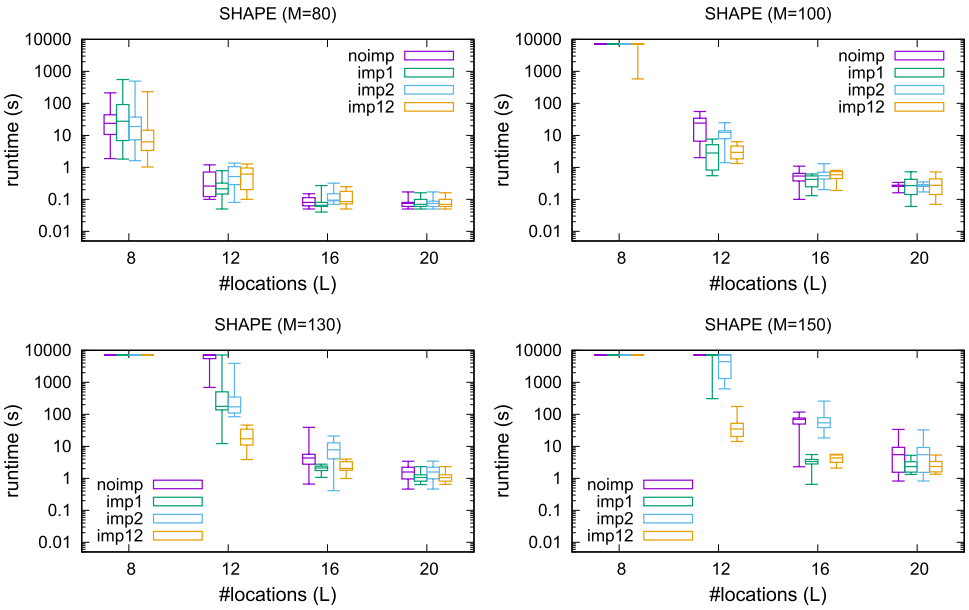

Fig. 2 PAR1 (in seconds) of solving some random B2B families of instances, with and without using implied constraints, varying their shape s. Instances are generated using the B2Brand model with $P = 40$, $T = 20$, $L = \{8, 12, 16, 20\}$, and (i) $M = 80$ (top left), (ii) $M = 100$ (top right), (iii) $M = 130$ (bottom left), and (iv) $M = 150$ (bottom right)

We observe that the higher the density is (i.e., the lower number of locations L), the harder the B2B instance becomes. This happens in the four cases analyzed, although it occurs more clearly as the density gets higher (i.e., with a high number of meetings M). This is expected since reducing the number of available locations reduces the possible parallelism among meetings, possibly creating some idle periods for some participants. Thus, finding the optimum schedule may be a harder task.

Again, using no implied constraints (`noimp`) is always worse than using any, whereas using both (`imp12`) is always better than the other three options.

Interestingly, as in the previous analysis, in this second experiment of B2B random instances (varying their shape) we can also distinguish some configurations in which using only one implied constraint performs better than using the other. In particular, when the shape is high, the encoding with the first implied constraint `imp1` seems to perform faster. See, for instance, the case with $L = 16$ and $M = 130$ (bottom left plot of Fig. 2), for which `imp1` is clearly faster than `imp2`. On the contrary, when the shape is low, `imp2` seems to outperform `imp1`. See, for instance, the case with $L = 12$ and $M = 150$ (bottom right plot of Fig. 2), where `imp2` solves most of the B2B instances of the family, but `imp1` only solves one. We summarize these results in Observation 2.

Observation 2 Using the implied constraint 1 (`imp1`) seems to be more interesting when the shape is high. On the contrary, the implied constraint 2 (`imp2`) seems to be more relevant when the shape is low. Overall, using both implied constraints (`imp12`) is always beneficial.

The intuition behind these observations may be connected to the relation between the implied constraints and the number of locations and time slots. In particular, the first implied constraint depends on the number of meetings of each participant, and this number is bounded by the number of time slots. The second implied constraint depends on the number of locations. Notice that since we are using cardinality networks to encode these constraints, the smaller the number of time slots is, the better for the encoding of the first implied constraint, and similarly with the number of locations and the second implied constraint. Overall, using both implied constraints seems to facilitate finding and propagating contradictions faster either when a participant has been scheduled more meetings than their actual meetings (i.e., implied constraint 1), or when a time slot has been scheduled more meetings than the number of available locations (i.e., implied constraint 2).

4.3 Real-World B2B Instances

Here we want to check if the previous observations are also valid in real-world B2B instances. Notice that in the case of real-world instances, the combinations of T and L are limited (to obtain feasible non-trivial instances). Therefore, the number of perturbed problems (from the original ones) is smaller. In this experiment, we use the 20 real-world B2B instances from [6].

In Fig. 3 we represent the runtime of solving these real-world B2B instances varying the density d, with a fixed shape s. To fix the shape of these problems, we use the original numbers of time slots and locations, and we increase both of them in the same proportion. Based on Observation 1, we

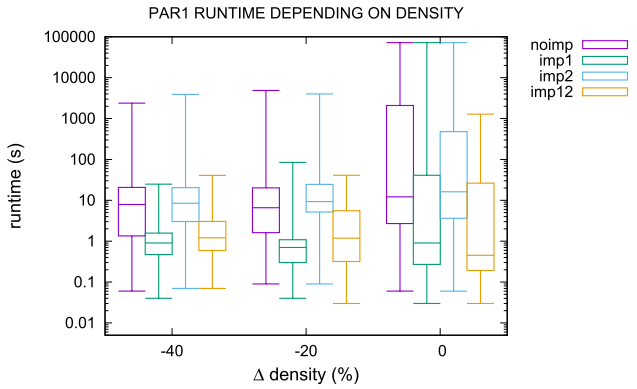

Fig. 3 PAR10 (in seconds) of solving real-world B2B instances, with and without using implied constraints, varying their density d, and with a fixed shape, i.e., $\Delta T = \Delta L$

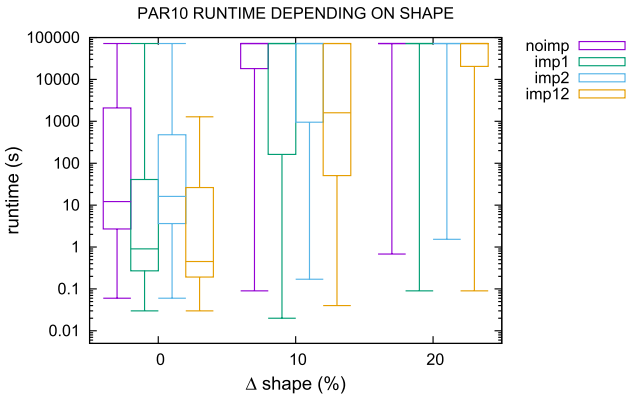

Fig. 4 PAR10 (in seconds) of solving real-world B2B instances, with and without using implied constraints, varying their shape s, and with a fixed density, i.e., fixed $T \cdot L$

predicted that `imp1` is more beneficial with small densities, `imp2` for high density, and `imp12` shows good performance in all cases. From the results, we observe that this observation is also valid in the set of real-world instances. For example, when $\Delta d = -40\%$, the fastest encoding in solving all instances is `imp1`. Also, `imp2` solves one instance more than `imp1` when $\Delta d = 0\%$. Finally, `imp12` is, in general, the best choice. To support this claim, in Table 1 we report some statistics for this experiment. We provide the PAR1 and PAR10 runtimes. Notice that the penalization in PAR1 is *small*, thus it is useful for comparing the runtime of configurations where the majority of instances were solved. On the other hand, using PAR10 especially penalizes timeouts, and thus it is useful when some instances were not solved. From these results, we observe that `imp12` is the fastest method in solving hard instances (see $\Delta d = 0\%$), but it is also a good choice when the instances are easy (see $\Delta d = -40\%$). In this last case, the differences between `imp1` (the fastest method) and `imp12` are small. Therefore, this observation seems to remain valid on this benchmarks.

In Fig. 4 we represent the runtime of solving some real-world B2B instances varying the shape s, with a fixed density d. To do so, we increase/decrease T and L in the same proportion.[5] The prediction from Observation 2 is that `imp1` seems to be more useful than `imp2` for small shapes, and vice-versa, while `imp12` is always beneficial. This claim can be observed from the plot. Also, in Table 2 we report some statistical results. When the number of timeouts is small, we consider more appropriate to compare the PAR1 scores. We observe that the PAR1 of `imp1` is very close to the PAR1 of `imp2` for small values of Δs (see $\Delta s = 0\%$), even when it solves one instance less, suggesting that it is much faster in the rest. However, as ΔT increases, `imp2` tends to be faster than `imp1`. Finally, we can observe than `imp12` dominates the other encodings. Therefore, this second observation also seems to be valid in real-world B2B problems.

4.4 Performance of the MaxSAT Solver

Finally, we conjecture why the use of implied constraints is beneficial to improve the performance of the solver. In the following figures, all Boolean variables are grouped into the high-level variables and constraints they encode (see the horizontal lines). The high-level variables are the following: *schedule*, *usedSlot* and *meetingHeld* are directly the ones of the encoding; *exactlyOne* are the auxiliary variables encoding that each meeting is scheduled in a time slot exactly once; *tableCount* are the auxiliary variables encoding that at most one meeting is scheduled in a time slot and location; `imp1` and `imp2` are the auxiliary variables encoding the implied constraints; the rest are the auxiliary variables to deal with optimization and homogeneity.

In Fig. 5 we represent the branching variables on which the solver decided along its execution, for the encodings `noimp` (top left), `imp1` (top right), `imp2` (bottom left) and `imp12` (bottom right), for a random B2B instance generated with the B2Brand model and low density ($P = 40$, $M = 90$, $T = 16$ and $L = 10$). For simplicity, we only represent the results of a single instance. However, we have found the same behavior in all instances we have analyzed. According to Observation 1, instances with low density are solved faster by `imp1` (and `imp12`). The runtime and the number of decisions of each encoding are reported in Table 3.

Since this instance is solved by `imp12` in 126,770 decisions, we only represent the first 130,000 decisions for the

[5] For distinct values of T and L, increasing/decreasing them in the same proportion does not ensure that its product continues being the same. However, this does happen in our real-world instances.

Table 1 Statistics of solving real-world B2B instances, varying their density d, for a fixed shape s (with $\Delta T = \Delta L$)

	$\Delta d = -40\%$			$\Delta d = -20\%$			$\Delta d = 0\%$		
	#s	PAR1	PAR10	#s	PAR1	PAR10	#s	PAR1	PAR10
noimp	20	3777.3	3777.3	**20**	7194.7	7194.7	18	30048.6	159648.7
imp1	**20**	**62.7**	**62.7**	**20**	154.3	154.3	18	14704.3	144304.3
imp2	20	4527.5	4527.5	**20**	5527.8	5527.8	19	12937.8	77737.8
imp12	**20**	110.1	110.1	**20**	**114.9**	**114.9**	**20**	**2060.5**	**2060.5**

#s stands for the number of solved instances. The best results are marked in bold

Table 2 Statistics of solving real-world B2B instances, varying the shape s, for a fixed density d (with $T \cdot L$ fixed)

	$\Delta s = 0\%$			$\Delta s = 10\%$			$\Delta s = 20\%$		
	#s	PAR1	PAR10	#s	PAR1	PAR10	#s	PAR1	PAR10
noimp	18	30048.7	159648.7	5	108228.3	1080228.3	3	122503.3	1224103.3
imp1	18	14704.3	144304.3	8	92829.1	870429.1	3	122409.9	1224009.9
imp2	19	12937.8	77737.8	9	87409.0	800209.0	3	127373.3	1228973.3
imp12	**20**	**2060.5**	**2060.5**	**11**	**68465.1**	**651665.1**	**5**	**113262.9**	**1085262.9**

#s stands for the number of solved instances, and the best results are marked in bold

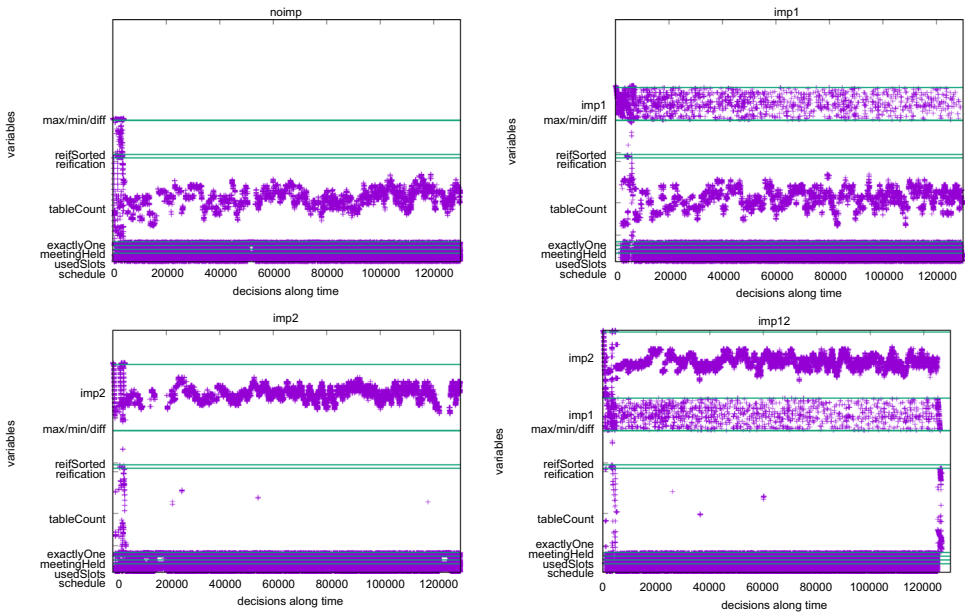

Fig. 5 Branching variables decided by the solver along its execution, for a random B2B instance with low density ($P = 40, M = 90, T = 16, L = 10$) during their first 130, 000 decisions (solved by `imp12` in 126, 770 decisions)

Table 3 Runtime and number of decisions required to solve the B2B problem with low density of Fig. 5: a random B2B instance with $P = 40, M = 90, T = 16, L = 10$

Encoding	Runtime	No. of decisions
No implied constraints (`noimp`):	175.42 s	2,362,611 dec.
Implied constraint 1 (`imp1`):	7.69 s	223,729 dec.
Implied constraint 2 (`imp2`):	36.08 s	521,272 dec.
Implied constraints 1 and 2 (`imp12`):	**5.07** s	**126,770** dec.

four encodings. However, the behavior shown in the plots is the same during the whole execution.

We remark that Open-WBO uses a CDCL SAT solver internally (we used its default version, with Glucose 3.0). In CDCL solvers, after each conflict, the variables involved in it are increased their activity, and these activity counters are used by the branching heuristic to select the next decision variable. Therefore, these decisions give an intuition about where the search was performed.

We observe that in four encodings, variables *schedule*, *usedSlot*, *meetingHeld* and *exactlyOne* are very active during the whole execution. This is expected since they represent the most important variables of the problem. When

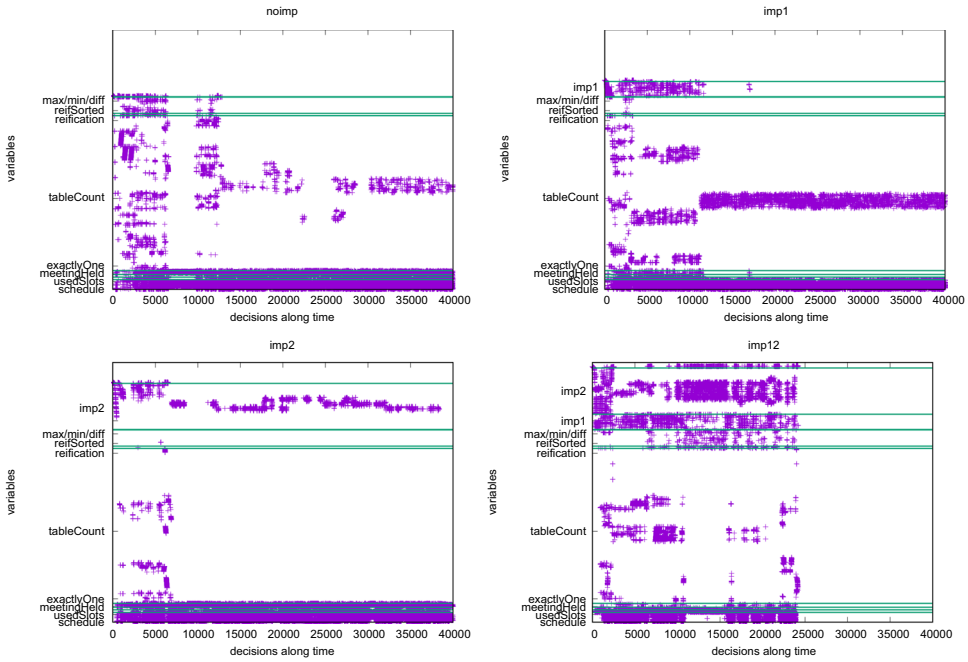

Fig. 6 Branching variables decided by the solver along its execution, for a random B2B instance with high density ($P = 40, M = 150$, $T = 10, L = 16$), during their first 40, 000 decisions (solved by `imp12` in 24, 169 decisions)

we use one implied constraint only, this implied constraint is very active. Also, when we use both implied constraints, they reinforce their activity mutually and, therefore, the performance of the solver improves. This phenomenon is clear from the plot, but we can analyze it in detail during the whole execution. For instance, in the encoding `imp2`, 25.08% of the decisions correspond to this implied constraint. When using the encoding `imp12`, the percentage of decisions on the variables of the implied constraint 2 is 22.24% of the total. Therefore, the use of `imp1` reduces the need of using `imp2`, and hence, the instance is solved faster.

In Fig. 6 we represent the results of the same experiment using an instance with high density ($P = 40$, $M = 150$, $T = 10$ and $L = 16$). Again, we only represent results for a single instance but conclusions are general for the family. According to Observation 1, instances with high density are solved faster by `imp2` (and `imp12`). This is also the case of this instance, whose runtime and number of decisions are reported in Table 4.

In this case, it is worth noticing that, although `imp2` needs a larger number of decisions to solve the formula, it solves it much faster. This suggests that `imp1` spends much more time in propagations, to find a conflict. In fact, it can be observed in Fig. 6 that the variables of this implied constraint are almost never decided during the whole execution (except at the beginning of the search). On the contrary, `imp2` makes more conflicting decision, but they propagate faster. As a consequence, the instance is solved much faster by `imp2`. Also, the encoding `imp2` took 8.62% of its decisions on the Boolean variables from this implied constraint. When using both implied constraints `imp12`, the decisions on the variables of the second implied constraint represent the 22.26% of the total number of decisions.

We also checked these effects on some real B2B instances. The encoding `imp1` is more efficient for the instance `forum-14`, taking 4.45% of its decisions on this constraint. When using both implied constraints, this increases up to 6.03% of the decisions. Similarly, the encoding `imp2` is more efficient solving the instance `tic-13crafc`, with a 15.45% of decisions on this constraint. When using both implied constraints, these decisions are 47.80%. This suggests that the previous hypothesis is also valid in real-world B2B instances.

5 Conclusions and Future Work

In this work, we have provided an experimental study of the impact of using implied constraints in B2B scheduling problems using MaxSAT-based encodings.

Table 4 Runtime and number of decisions required to solve the B2B problem with high density of Fig. 6: a random B2B instance with $P = 40, M = 1500, T = 10, L = 16$

Encoding	Runtime	No. of decisions
No implied constraints (`noimp`):	51.55 s	820,327 dec.
Implied constraint 1 (`imp1`):	51.35 s	778,518 dec.
Implied constraint 2 (`imp2`):	36.26 s	1,462,374 dec.
Implied constraints 1 and 2 (`imp12`):	**0.56** s	**24,169** dec.

Due to the limited number of real-world instances, we have proposed a random B2B instances generator, using a model based on an uniform probability of a participant requesting a meeting with another participant. Using this generator, we have generated families of random B2B instances, and studied the strengths and weaknesses of using implied constraints depending on the characteristics of the instance. We have focused the analysis on the density (i.e., the ratio between the number of meetings and the accommodation capacity) and the shape (i.e., the configuration of the accommodation capacity, expressed as the ratio between time slots and locations).

We have observed that there exists a duality in the benefits of using the two implied constraints studied in this work. For small densities or high shapes, we have seen that it is more useful to use the implied constraint `imp1`. On the contrary, for high densities or low shapes, the second implied constraint `imp2` is more beneficial. Overall, the use of both implied constraints results in a very good performance in all cases.

As future work, we propose to extend this analysis in three different directions. First, we plan to extend our random B2B model to incorporate *passive* participants. As stated before, this can be achieved using a different probability distribution when choosing the two participants of each meetings, rather the uniform distribution used in the current model. Additionally, the random model can be extended by introducing secondary constraints, like forbidden time slots and fixed sessions (e.g., morning and afternoon meetings). Second, we propose to study solver heuristics that, for instance, prioritize taking decisions on the most relevant variables of the model (e.g., the variables that encode the implied constraints `imp1` or `imp2`), depending on the characteristics of the problem, like the density or the shape. Finally, we plan to study further cardinality constraints to solve B2B problems, and their impact on the characteristics of the instances (such as density and shape).

Author Contributions All authors have equally contributed to this work.

Funding Miquel Bofill, Josep Suy and Mateu Villaret are partially funded by grant RTI2018-095609-B-I00 (MICINN/FEDER, UE). Jordi Coll is partially funded by the French Agence Nationale de la Recherche, reference ANR-19-CHIA-0013-01. JesÚs Giráldez-Cru is supported through the Juan de la Cierva program, fellowship IJC2019-040489-I, funded by MCIN and AEI.

Availability of Data and Material All authors give their consent for the publication of this work.

Declarations

Conflict of Interest Not applicable.

Ethics Approval and Consent to Participate Not applicable.

Consent for Publication Not applicable.

References

1. Abío, I., Nieuwenhuis, R., Oliveras, A., Rodríguez-Carbonell, E.: A parametric approach for smaller and better encodings of cardinality constraints. In Proceedings of the 19th International Conference on Principles and Practice of Constraint Programming (CP'13), vol. 8124, pp. 80–96 (2013)
2. Alsinet, T., Béjar, R., Cabiscol, A., Fernández, C., Manyà, F.: Minimal and redundant SAT encodings for the all-interval-series problem. In: Proceedings of the 5th International Conference of the Catalan Association for Artificial Intelligence CCIA 2002, pp. 139–144 (2002)
3. Ansótegui, C., del Val, A., Dotú, I., Fernández, C., Manyà, F.: Modeling choices in quasigroup completion: SAT vs. CSP. In: Proceedings of the 19th National Conference on Artificial Intelligence (AAAI'04), pp. 137–142 (2004)
4. Bofill, M., Coll, J., Garcia, M., Giráldez-Cru, J., Pesant, G., Suy, J., Villaret, M.: Constraint solving approaches to the business-to-business meeting scheduling problem. J. Artif. Intell. Res. (in press) (2022)
5. Bofill, M., Espasa, J., Garcia, M., Palahí, M., Suy, J., Villaret, M.: Scheduling B2B meetings. In: Proceedings of the 20th international conference on principles and practice of constraint programming (CP'14), vol. 8656, pp. 781–796 (2014)
6. Bofill, M., Garcia, M., Suy, J., Villaret, M.: MaxSAT-based scheduling of B2B meetings. In: Proceedings of the 12th International Conference on Integration of AI and OR Techniques in Constraint Programming (CPAIOR'15), vol. 9075, pp. 65–73 (2015)
7. Bofill, M., Giráldez-Cru, J., Suy, J., Villaret, M.: A study on implied constraints in a maxsat approach to B2B problems. In: Proceedings of the 22nd International Conference of the Catalan Association for Artificial Intelligence (CCIA'19), vol. 319, pp. 183–192 (2019)
8. Gebser, M., Glase, T., Sabuncu, O., Schaub, T.: Matchmaking with answer set programming. In: Proceedings of the 12th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'13), pp. 342–347 (2013)
9. Kautz, H. A., Ruan, Y., Achlioptas, D., Gomes, C. P., Selman, B., Stickel, M. E.: Balance and filtering in structured satisfiable problems. In: Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI'01), pp. 351–358 (2001)
10. Li, C. M., Manyà, F.: Handbook of satisfiability, chapter MaxSAT. In: Hard and Soft Constraints, pp. 613–631. IOS Press (2009)
11. Martins, R., Manquinho, V.M., Lynce, I.: Open-WBO: a modular MaxSAT solver. In: Proceedings of the 17th International

Conference on Theory and Applications of Satisfiability Testing (SAT'14), pp. 438–445 (2014)
12. Pesant, G., Rix, G., Rousseau, L.: A comparative study of MIP and CP formulations for the B2B scheduling optimization problem. In: Proceedings of the 12th International Conference on Integration of AI and OR Techniques in Constraint Programming (CPAIOR'15). LNCS, vol. 9075, pp. 306–321. Springer (2015)
13. Selman, B., Kautz, H.A., McAllester, D.A.: Ten challenges in propositional reasoning and search. In: Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI'97), pp. 50–54 (1997)