A Comparative Study of MIP and CP Formulations for the B2B Scheduling Optimization Problem

Gilles Pesant[1,2], Gregory Rix[1,2], and Louis-Martin Rousseau[1,2]

[1] École Polytechnique de Montréal, Montréal, Canada,
[2] Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation, Montréal, Canada,
{gilles.pesant,greg.rix,louis-martin.rousseau}@polymtl.ca

Abstract. The Business-to-Business Meeting Scheduling Problem was recently introduced to this community. It consists of scheduling meetings between given pairs of participants to an event while taking into account participant availability and accommodation capacity. The challenging aspect of this problem is that breaks in a participant's schedule should be avoided as much as possible. In an earlier paper, starting from two generic CP and Pseudo-Boolean formulations, several solving approaches such as CP, ILP, SMT, and lazy clause generation were compared on real-life instances. In this paper we use this challenging problem to study different formulations adapted either for MIP or CP solving, showing that the `cost_regular` global constraint can be quite useful, both in MIP and CP, in capturing the problem structure.

1 Introduction

Business-to-business events consist of scheduling meetings between pairs of participants with similar interests. The business-to-business scheduling optimization problem (B2BSOP) was formally introduced in [2], with application to the *4th Forum of the Scientific and Technological Park of the University of Girona*[3]. Several constraints must be respected including participant availability by time, accommodation capacity, and fairness constraints between participants. The challenging aspect of this problem is that breaks in a participant's schedule should be avoided as much as possible. The objective is therefore to minimize the number of breaks assigned to participants, cumulatively over all participants.

The authors of [2] mentioned that solving this problem allowed the organizers to significantly improve their efficiency, in particular because the models were able to handle side constraints, such as fairness, and extend partly fixed schedules. Adopting a "model-and-solve" approach with two fairly straightforward models but state-of-the-art solvers using different combinatorial optimization methods, they wondered whether more dedicated SAT or MIP models would

[3] `http://www.forumparcudg.com`

perform better. Additionally their CP model did not make use of global constraints, which are known to play an important role in the success of CP solvers. These motivated us to look more closely at the B2BSOP in a attempt to capture its underlying structure and to better exploit it in MIP and CP approaches.

In this paper we thus derive and compare empirically several MIP and CP formulations of the problem. Our main observation is that the use of the `cost_regular` constraint is powerful in capturing the structure of the problem, producing stronger lower bounds than simpler MIP formulations, scaling better to larger instances by finding better feasible solutions, and providing a more effective cost-aware branching heuristic for CP.

The remainder of this paper is organized as follows. Section 2 recalls the definition of the B2BSOP. Section 3 focuses on the CP approach, presenting the models, computational results and comparisons. Section 4 presents, evaluates and compares four MIP models. Section 5 then discusses the overall results with respect to previously published methodologies.

2 Problem Definition, Instance Description, and Experimental Setting

Let P be the set of participants, M the set of meetings between pairs of participants, $M_p \subseteq M$ the set of meetings involving participant p, L the set of locations for meetings, and T the set of time slots. The problem is to assign the meetings in M to time slots in T and locations in L so that no participant is in more than one meeting at a time and at most $|L|$ meetings are held at any time. A participant's schedule can then be seen as a vector of length T in which each element is either a meeting in M_p (each appearing only once), or a free period ("0"). We refer to a sequence of 0s that occurs between two meetings as a *break*.

A few other constraints must hold. The set of meetings M_p for participant p can be refined into meetings M_p^{AM} that can only be held in the morning and meetings M_p^{PM} that can only be held in the afternoon. These two sets typically have meetings in common, as some meetings are unrestricted. T^{AM} and T^{PM} respectively represent the morning and afternoon time slots. We also define $F_p \subset T$ representing the forbidden time slots for p. We can derive $T_m \subseteq T$, the set of allowed time slots for meeting m, which takes into account the forbidden time slots of both participants as well as any morning or afternoon requirements of meeting m.

The above defines a feasibility problem for B2B scheduling that could be modeled and solved simply as a graph colouring problem. However an important aspect of the problem is to minimize the number of breaks assigned to all participants cumulatively; this is added as an objective function. Additionally, a schedule must be observed to be fair from the point of view of any participant: too many breaks in an individual schedule relative to another participant is viewed as unfair; we define $b^{\max}$ to be the maximum deviation in the number of breaks between any two participants.

Table 1: Features of the B2BSOP instances from [2].

feature	**T2a**	**T2c**	**T3a**	**T3b**	**T3c**	**F3a**	**F3b**	**F3c**	**F4a**
#meetings	125	125	180	184	180	154	195	154	302
#participants	42	42	47	46	47	70	76	70	78
#locations	21	16	21	21	19	14	14	12	22
#time slots	8	8	10	10	10	21	21	21	22
#AM slots	0	0	0	0	0	13	13	13	12

Table 1 lists the main features of the instances from [2]. There are two sets **T**$\star$ and **F**$\star$, each originating from an actual event with a mix of real and synthetic instances. The first set has fewer participants, fewer time slots, no meeting restricted to morning or afternoon slots, and no forbidden time slots for participants. The second set is richer and more difficult to solve.

The MIP and CP models were executed on Dual core AMD 2.1 GHz processors with 8 GB of RAM, running IBM ILOG Solver 6.7 as the CP solver and Gurobi 6.0 as the MIP solver. As in [2] we used a 2-hour time limit.

3 CP Models

A natural CP model defines variables $\{s_{pt} : p \in P, t \in T\}$ to represent what participant p is scheduled to do at time t, value 0 corresponding to no meeting. An alternative model could have defined one variable per meeting with the set of allowed time slots for its domain, as proposed in [2]. However such a representation does not allow us to express constraints directly on the sequence of meetings for a participant, which is important to evaluate the cost of an individual schedule and ultimately the objective we seek to minimize: the latter authors had to define auxilliary variables similar to ours and channel them to their main variables.

Here is our model for the feasibility subproblem:

$$\texttt{gcc}(\{s_{p\star}\},\ \langle |T| - |M_p|, 1, \ldots, 1\rangle) \qquad p \in P \tag{1}$$

$$\texttt{gcc}(\{s_{\star t}\},\ \langle \{|P| - 2|L|, \ldots, |P|\}, \{0,2\}, \ldots, \{0,2\}\rangle) \qquad t \in T \tag{2}$$

$$s_{pt} = 0 \qquad p \in P, t \in F_p \tag{3}$$

$$s_{pt} \in M_p^{\text{AM}} \cup \{0\} \qquad p \in P, t \in T^{\text{AM}} \tag{4}$$

$$s_{pt} \in M_p^{\text{PM}} \cup \{0\} \qquad p \in P, t \in T^{\text{PM}} \tag{5}$$

Constraints (1) use a global cardinality constraint [8] on the decision variables of a given participant to ensure that each of his meetings appears once (the first component of the vector of occurrences, corresponding to value 0, indicates the number of time slots without a meeting). Constraints (2) use a global cardinality constraint on the decision variables of a given time slot to express two things: the first component says that the number of participants not having a meeting

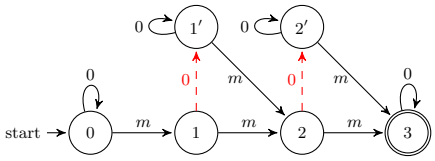

Fig. 1: Automaton $\mathcal{A}_1$ for a participant with three meetings. Arc label "m" stands for any meeting and label "0" for no meeting. Only the red dashed arcs carry a cost, of one unit, to mark the start of a break.

must be at least $|P| - 2|L|$ because we can hold at most $|L|$ meetings and each meeting appears twice (once for each participant); the other components, for each meeting, say that the two participants to a given meeting must attend it in the same time slot and therefore a meeting occurs twice or not at all. Constraints (3) ensure that there is no meeting scheduled for a participant during one of his forbidden time slots.

3.1 Modeling Break Patterns

We now model the optimization component of our problem. We define a variable b_p for each participant p giving the number of breaks in his schedule and seek to minimize the total number of breaks in the schedule. In order to link the b_p variables to the main s_{pt} variables we need to consider the sequence of values taken by the decision variables of a participant: each subsequence of zeros in between scheduled meetings for p corresponds to a break and b_p represents how many such breaks there are in the sequence. For example, patterns $0\star\star00\star00$ and $\star000\star0\star0$ for eight time slots and three meetings feature respectively one and two breaks.

To express this globally we could enumerate each possible pattern, associate its number of breaks and use a `table` constraint. For given T, M_p, and maximum number of breaks $b^\star$ this makes

$$\sum_{i=0}^{b^\star} \binom{|M_p| - 1}{i} \cdot \binom{|T| - |M_p| + 1}{i+1}$$

patterns. Even if we restrict ourselves to at most $b^\star = 2$ breaks, the number of patterns is in $\Theta(|M_p|^2(|T| - |M_p|)^3)$ which, when the number of meetings is about half of the number of time slots, simplifies to $\Theta(|M_p|^5)$. Considering that the largest instance has 22 time slots with some participants holding 11 meetings, we could end up generating hundreds of thousands of patterns.

A much more compact way to express this uses an automaton on $2|M_p|$ states that recognizes precisely these patterns. Figure 1 presents such an automaton for a participant with three meetings. Observe however that by concentrating on patterns without distinguishing between meetings we may miss some inferences.

For example any assignment from the sequence of domains $\langle\{m_1, m_2, m_3, 0\}, \{m_1, m_2, m_3, 0\}, \{m_4, 0\}, \{m_4, 0\}, \{m_1, m_2, m_3, 0\}, \{m_1, m_2, m_3, 0\}\rangle$ corresponding to four meetings being scheduled over six time slots will necessarily introduce at least one break but such an automaton will not recognize it. To catch this, a more fine-grain automaton distinguishing between meetings will have $2^{|M_p|+1}-2$ states essentially representing all subsets of meetings (see Figure 2). Because this automaton has significantly more states we will refrain from using it when the number of meetings is greater than a certain threshold $m^{\max}$. Here is the rest of our model:

$$\min \sum_{p \in P} b_p \qquad \text{s.t.} \tag{6}$$

$$b_p - \min_{p' \in P} b_{p'} \leq b^{\max} \qquad p \in P \tag{7}$$

$$b_p = 0 \qquad p \in P : \ |M_p| \in \{0, 1, |T|\} \tag{8}$$

$$\texttt{cost\_regular}(\langle s_{p\star} \rangle, \mathcal{A}_2, b_p) \qquad p \in P : \ 1 < |M_p| \leq m^{\max} \tag{9}$$

$$\texttt{cost\_regular}(\langle s_{p\star} \rangle, \mathcal{A}_1, b_p) \qquad p \in P : \ m^{\max} < |M_p| < |T| \tag{10}$$

$$b_p \in \mathbb{N} \qquad p \in P \tag{11}$$

As in [2] we ensure some fairness between individual schedules by requiring that the number of breaks among individual schedules differ by at most $b^{\max} = 2$ (see Constraints (7)). Constraints (8) fix b_p to zero for participants who trivially have no break in their schedule (e.g. they have a single meeting or as many meetings as there are time slots). Note that because in every instance considered there are always such participants, individual schedules will feature at most two breaks. The `cost_regular` constraint (10) on automaton $\mathcal{A}_1$ maintains a layered digraph on $2|M_p|(|T|+1)$ vertices and makes variable b_p equal to the sum of the costs of the arcs on the path corresponding to the values taken by the sequence of variables $\langle s_{p\star} \rangle$ [4]. The upper bound on b_p limits the feasible paths in the digraph and possibly removes arcs (i.e. filters values in a domain) that do not belong to any feasible path. Conversely the smallest cost of the possible paths given the current domains of the variables provides a lower bound on b_p. Constraints (9) work similarly but on the larger automaton $\mathcal{A}_2$.

3.2 Branching Heuristics

For the most part, CP branching heuristics are feasibility-driven: an optimization problem is solved as a succession of feasibility problems from which each new improved solution provides a tighter bound on the objective value, formulated as a constraint that is added to the model. A generic heuristic such as max regret [1] does take into account the objective but it is only effective if the cost of a solution can be decomposed at the level of individual variable assignments. For the B2BSOP one needs to consider neighbouring assignments in the sequence or even the whole sequence in order to assess the impact of a particular variable-value assignment on the number of breaks. To make matters worse the objective is a sum, reputed to back propagate poorly.

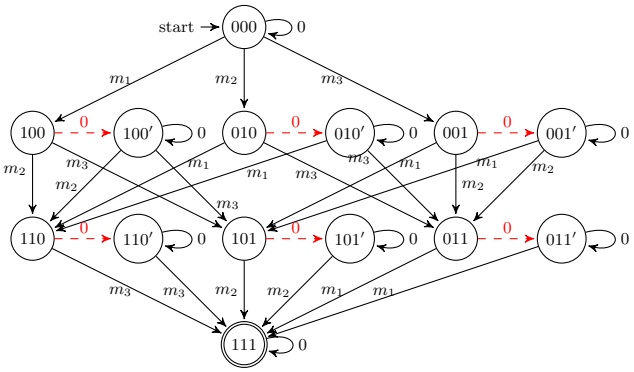

Fig. 2: Automaton $\mathcal{A}_2$ for a participant with three meetings. Arc label "m_i" stands for that particular meeting and label "0" for no meeting. Only the red dashed arcs carry a cost, of one unit, to mark the start of a break.

Table 2: Number of breaks in best solution found within the time limit using different heuristics branching on s_{pt} variables for chronological backtrack search on the same CP model. For reference we add the performance reported in [2] for the CP solver they used (Gecode), albeit on a different model.

heuristic	**T2a**	**T2c**	**T3a**	**T3b**	**T3c**	**F3a**	**F3b**	**F3c**	**F4a**
dom	1	1	27	13	31	64	–	–	81
IBS	0	6	0	0	19	72	–	68	100
maxSD	10	7	–	5	–	–	–	–	–
maxSD*	0	0	3	0	–	1	–	13	23
Gecode [2]	0	–	0	0	–	–	–	–	–

Table 2 reports the performance of a few standard generic branching heuristics on the s_{pt} variables of our CP model. Smallest-domain-first (dom), impact-based search (IBS)[7], and counting-based search (maxSD)[6] perform poorly, as expected, since they are not driven at all by the objective. Some of the smaller instances are solved well by IBS but generally the solutions obtained are quite far from the optimal or best-known solutions and for some instances we get no solution at all within the time limit. This is consistent with the observations of Bofill et al. [2] albeit on a different CP model and running a different CP solver: we report their findings on the last line of the table. We also investigated branching on the b_p variables and trying lower values first, which did help generate somewhat better-quality solutions but not competitive with what we describe below.

Heuristic maxSD, previously shown to be very effective on feasibility problems [6], performs particularly poorly on this optimization problem. It branches on the variable-value pair that has the highest *solution density*, which is the

Table 3: Number of breaks in best solution found within the time limit using different heuristics branching on s_{pt} variables for limited-discrepancy search on the same CP model.

heuristic	**T2a**	**T2c**	**T3a**	**T3b**	**T3c**	**F3a**	**F3b**	**F3c**	**F4a**
dom	0	0	0	0	8	35	73	61	80
IBS	0	0	1	0	15	72	–	76	123
maxSD	0	0	11	0	–	–	–	–	–
maxSD*	0	0	0	0	11	0	23	9	25

proportion of solutions to an individual constraint that feature this given assignment. Because it does not discriminate between these solutions in terms of their quality with respect to the objective, such information can be misleading. We experimented with a cost-aware version of this approach: instead of considering solutions of any quality, we compute solution densities among the solutions of lowest cost, thereby integrating optimization into our branching recommendation. For the B2BSOP we retrieve solution densities from the `cost_regular` constraints, the combinatorial substructure from which the cost (i.e. the number of breaks) associated with individual participants can be computed. Instead of considering all paths in the layered digraph maintained by the constraint, we only consider shortest paths. We branch on the largest cost-aware solution density for scheduling a meeting (values in M). We denote that generic heuristic as maxSD* in Table 2. As anticipated it performs much better than the others, producing higher-quality solutions, though not producing any solution in two instances.

It is well known for backtrack search that regardless of a branching heuristic's quality, chronological backtracking may take a very long time to undo a bad branching decision made early on. There are a few devices commonly used to add robustness to heuristics, e.g. randomized restarts and limited-discrepancy search (LDS) [5]. The latter modifies the order in which the leaves of a search tree are visited according to how often the corresponding path deviates from the search heuristic's recommendation: first the leaf with 0 deviation, then those with $1k$ deviations, followed by those with $2k$ deviations, and so forth, for a given parameter k. This has the effect of more quickly changing decisions close to the root. We add LDS to our branching heuristics (see Table 3). We note a general improvement of solution quality and of robustness, and maxSD* now finds good-quality solutions to every instance.

The $m^{\max}$ threshold, determining when a participant has too many meetings to use the increased-inference but also increased-time-and-space-consumption fine-grain automaton $\mathcal{A}_2$, has an impact on solution quality. Table 4 reports our findings for a few values of that threshold. The first line shows that never using that automaton produces lower-quality solutions for the larger instances. Otherwise a moderate threshold of 5 already makes a difference and the approach

Table 4: Number of breaks in best solution found within the time limit using different $m^{\max}$ thresholds to select which automaton to use in each `cost_regular` constraint. The maxSD* branching heuristic is used together with LDS.

$m^{\max}$	**T2a**	**T2c**	**T3a**	**T3b**	**T3c**	**F3a**	**F3b**	**F3c**	**F4a**
0	0	0	0	0	11	7	24	12	27
5	0	0	0	0	11	0	22	10	26
10	0	0	0	0	11	0	23	9	25
12	0	0	0	0	11	0	21	10	25

Table 5: Applying Large Neighbourhood Search on the CP model for the six instances without a proof of optimality after one minute of CP backtrack search.

approach	**T3a**	**T3c**	**F3a**	**F3b**	**F3c**	**F4a**
maxSD*	0	11	0	23	9	25
LNS with maxSD* (avg of 10 runs)	0	4.2	2.4	16.8	3.8	17.1
LNS with maxSD* (best of 10 runs)	0	4	0	13	3	13

does not appear too sensitive to the actual value of the threshold. In our other experiments we used $m^{\max} = 10$.

3.3 Applying Large Neighbourhood Search to the CP Model

Local search is often the method of choice to solve large combinatorial optimization problems. Large Neighbourhood Search (LNS) is a natural way to perform local search on a solution space defined by a CP model [9]: it iteratively freezes part of the current solution and explores the remaining solution space (a potentially large neighbourhood) by applying a (usually incomplete) CP tree search, benefiting from the usual inference and search heuristics.

We evaluated a simple implementation of this idea: we run our exact CP algorithm for one minute in order to get a fair initial solution and then iteratively freeze the schedule of a randomly-selected subset of the participants whose schedule does not contain any break (but we disregard participants who trivially cannot have any break). We explore each neighbourhood with the same exact CP algorithm, stopping at the first improving solution or until 20 seconds have passed. We use the same overall time limit as before.

Table 5 reports the performance of our LNS with respect to our previous CP backtrack search. Since that approach is not deterministic, we give the average and the best objective value out of ten runs. We see a significant improvement of the solutions (except on **F3a** for which an optimal solution was not obtained consistently). In particular an optimal solution to **T3c** (see Section 4) is finally found and we also obtain our best overall solution for **F4a**.

4 MIP Models

We compare four mathematical formulations that exactly solve the B2BSOP. Each formulation contains a common subset of binary variables that fix a meeting to a particular time slot, and a common subset of constraints that enforce a feasible schedule without consideration of the objective value of break minimization. The difference in the formulations is with respect to the added variables and constraints that cumulate this objective value, as well as use the objective function to enforce fairness constraints on the resulting schedule.

The first formulation uses a set of binary variables that determine if a time slot is the terminating time slot of a break period before a participant begins a meeting. These variables are linked to the feasibility problem through a series of linearized logical constraints. The logical constraints to be linearized are those derived in the pseudoboolean model of [2]. The second formulation defines a network flow problem for each participant, with the flow representing the journey of the participant between breaks and meetings. The final two formulations are derived using the `cost_regular` constraint, with two unique deterministic finite automatons (DFAs) that cumulate the cost of a schedule for each participant. The first DFA is the previously defined automaton $\mathcal{A}_1$, whereas the second additionally enforces the fairness constraints. However, this final automaton requires a fixed maximum on the number of breaks in a schedule. As mentioned in Section 3.1, all considered instances contain participants who must have 0 breaks; hence the maximum number of breaks is equivalent to the maximum deviation $b^{\max}$. Hence this formulation is applicable in this context and will be used for comparative purposes, but does not solve the B2BSOP as defined in the general sense. We also note that preliminary experimentation showed that the number of variables associated with the automaton $\mathcal{A}_2$ proved too large to be competitive with the other formulations.

In order to apply the `cost_regular` constraints to the MIP formulations, we use the approach of [3]. We first create the layered graph of the `cost_regular` constraint consistency algorithm. From this layered graph we derive a network flow formulation, where the source node is the initial state of the DFA, and the sink node is connected to every final state. The network flow is then easily translated to a set of linear variables and constraints.

We first give the subproblem present in all considered formulations that enforces the feasibility of a solution. We let x_{mt} be a binary variable that equals 1 if meeting m is held at time t, and enforce the following constraints.

$$\sum_{t \in T} x_{mt} = 1 \qquad m \in M \tag{12}$$

$$\sum_{m \in M_p} x_{mt} \leq 1 \qquad p \in P, t \in T \tag{13}$$

$$\sum_{m \in M} x_{mt} \leq |L| \qquad t \in T \tag{14}$$

$$x_{mt} = 0 \qquad m \in M, t \in T \setminus T_m \tag{15}$$

Constraints (12) force each meeting to be held. Constraints (13) force each participant to be in at most one meeting at a time. Constraints (14) force the number of meetings at any time to be at most the number of locations. Constraints (15) force each meeting to be held in an allowed time slot. The following subsections give the formulation-specific variables and constraints that cumulate the objective function derived from the schedule of each participant.

4.1 Logical

We define the following variables:

$b' \in \mathbb{Z}$	Variable cumulating the minimum number of breaks assigned to any participant,
$y_{pt} \in \{0,1\}$	Indicator if participant p has a meeting at time t,
$z_{pt} \in \{0,1\}$	Equals 1 for time t starting from participant p's first meeting,
$h_{pt} \in \{0,1\}$	Indicator if time t terminates an idle period for participant p.

The variables h_{pt} contribute a value of 1 to the objective function. The necessary constraints linking these variables to the model appear below.

$$\sum_{t \in T} y_{pt} = |M_p| \qquad p \in P \tag{16}$$

$$x_{mt} \leq y_{pt} \qquad p \in P, m \in M_p, t \in T \tag{17}$$

$$y_{pt} \leq z_{pt} \qquad p \in P, t \in T \tag{18}$$

$$z_{pt} \leq z_{p,t+1} \qquad p \in P, t \in T, t \leq |T| - 1 \tag{19}$$

$$y_{p(t+1)} - h_{pt} \leq y_{pt} + 1 - z_{pt} \qquad p \in P, t \in T \tag{20}$$

$$\sum_{t \in T} h_{pt} \geq b' \qquad p \in P \tag{21}$$

$$\sum_{t \in T} h_{pt} \leq b' + b^{\max} \qquad p \in P \tag{22}$$

Constraints (16) through (19) link the y and z variables to the formulation. Constraints (20) then link the h variables to the formulation, forcing a break to be counted after an idle period. Constraints (21) and (22) are fairness constraints that limit the difference in the number of breaks per participant. We then define the problem (P1) as the minimization of the objective function

$$\sum_{p \in P} \sum_{t \in T} h_{pt}$$

subject to constraints (12) through (15) and (16) through (22).

4.2 Network Flow

We define the space-time network $G_p = (N_p, A_p)$ for each participant p, with node set $\{source, sink\} \bigcup (T \times \{0,1\})$. We define v_{tq} to be one such node for

time slot t and binary index q. The index $q = 0$ corresponds to the participant on break at time t, and $q = 1$ corresponds to the participant in a meeting. We create an arc from the source to each v_{t1}, from each v_{t1} to the sink, and from each v_{tq} to each $v_{(t+1)q'}$.

Variables y_{pt} are added to the formulation, with the same definition as in problem (P1), and constraints (16) and (17) link these variables to the variables x_{mt}. Additionally, the binary variable f_a^p is defined for all arcs in G_p to represent the source-to-sink flow. This flow is defined by the following constraints.

$$\sum_{(source,v)\in A_p} f^p_{(source,v)} = 1 \qquad p \in P \tag{23}$$

$$\sum_{(v,sink)\in A_p} f^p_{(v,sink)} = 1 \qquad p \in P \tag{24}$$

$$\sum_{(v,w)\in A_p} f^p_{(v,w)} - \sum_{(w,v)\in A_p} f^p_{(w,v)} = 0 \qquad p \in P, v \in T \times \{0,1\} \tag{25}$$

$$f^p_{(source,v_{t1})} + f^p_{(v_{(t-1)0},v_{t1})} + f^p_{(v_{(t-1)1},v_{t1})} = y_{pt} \qquad p \in P, t \in T \tag{26}$$

$$\sum_{t\in T, t\le |T|-1} f^p_{(v_{t1},v_{(t+1)0})} \ge b' \qquad p \in P \tag{27}$$

$$\sum_{t\in T, t\le |T|-1} f^p_{(v_{t1},v_{(t+1)0})} \le b' + b^{\max} \qquad p \in P \tag{28}$$

Constraints (23) through (25) define a source-to-sink flow of 1 unit, for each participant. Constraints (26) link this network flow to the variables y_{pt}. Constraints (27) and (28) are fairness constraints that limit the difference in the number of breaks per participant. We then define the problem (P2) as the minimization of the objective function

$$\sum_{p\in P} \sum_{t\in T, t\le |T|-1} f^p_{(v_{t1},v_{(t+1)0})}$$

subject to constraints (12) through (15), (16), (17), and (23) through (28).

4.3 Cost Regular

We create a MIP `cost_regular` constraint for each participant $p \in P$ using the previously defined automaton $\mathcal{A}_1$ as illustrated at Figure 1, which links to the variables y_{pt}. This constraint ensures that a covering of the meetings M_p is a word recognized by the DFA, with the objective function measured as the cost of the word. We then define the problem (P3) as the minimization of the objective function defined by the set of these `cost_regular` constraints, subject to constraints (12) through (15), (16), (17), and the `cost_regular` constraints defined by the DFA $\mathcal{A}_1$. We enforce the fairness constraints in an analogous manner to the previous two models, bounding the appropriate subsets of the objective function.

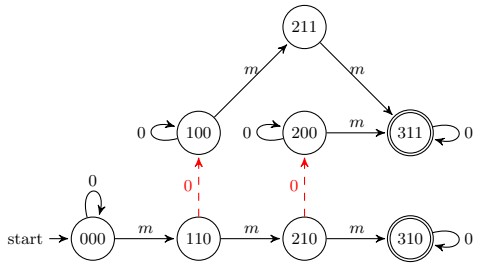

Fig. 3: Automaton $\mathcal{A}_3$ integrating fairness for a participant with three meetings. Arc label "m" stands for any meeting and label "0" for no meeting. Only the red dashed arcs carry a cost, of one unit, to mark the start of a break. Notice that there is no transition labeled "0" out of state "211" because of the limit of one break.

4.4 Cost Regular with Integrated Fairness Constraints

We again define a DFA for each participant $p \in P$. Each state corresponds to the triple (n, w, h), where parameter w is 1 if the previous activity was a meeting and 0 otherwise, and parameter h represents the number of breaks taken on the schedule thus far. The DFA presented in Figure 3 is for a participant with three meetings and a maximum of one break. All arcs have zero cost, except again those denoted with a dashed line.

We define the problem (P4) to be the minimization of the derived objective function, subject to constraints (12) through (15), (16), (17), and the MIP `cost_regular` constraints defined by the DFAs $\mathcal{A}_3$. Note that for the CP approach this augmented automaton is redundant because we already consider fairness in the CP `cost_regular` constraints by restricting the b_p variables: paths in the digraph will already be of length at most $b^{\max}$.

4.5 Empirical Results

We resolved each of the nine problem instances considered in [2] with each MIP formulation. This was executed for the parameter $b^{\max}$ set to values of both 2 (as used in [2]) and 1. For each resolution, we give the number of variables and constraints present in the formulation, the objective value of the root node linear relaxation, the value of the best found solution by the branch-and-bound, the branch-and-bound lower bound associated with this best solution, and the runtime in seconds. The results for $b^{\max}$ values 2 and 1 appear in Tables 6 and 7, respectively. The formulation that yields either the best objective value or the best runtime is bolded.

All but one formulation are able to solve only the first 6 instances to optimality, with the remaining 3 instances timing out. The exceptions are (P3) and (P4), which additionally proved optimality on instance **F3c** when $b^{\max} = 2$. Under this

Table 6: MIP Results with $b^{\max} = 2$

Model	Result	T2a	T2c	T3a	T3b	T3c	F3a	F3b	F3c	F4a
P1	$\|V\|$	**1967**	1967	3164	3175	3164	7575	9231	7575	**11715**
	$\|C\|$	**3427**	3427	5627	5687	5627	12675	15277	12675	**21160**
	Root	**0**	0	0	0	0	0	3	0	**2**
	Objective	**0**	0	0	0	4	0	12	6	**9**
	Bound	**0**	0	0	0	4	0	5	0	**2**
	Time	**7.62**	15.48	27.04	167.01	303.44	942.61	7200	7200	**7200**
P2	$\|V\|$	2173	2173	**4007**	**3645**	4007	13245	14923	13245	17961
	$\|C\|$	3043	3043	**5361**	**5196**	5361	12955	15563	12955	21398
	Root	0	0	**0**	**0**	2	0	3.0198	0	2
	Objective	0	0	**0**	**0**	4	0	45	2	16
	Bound	0	0	**0**	**0**	4	0	5	0	2
	Time	14.43	10.04	**9.44**	**156.28**	822.97	1545.7	7200	7200	7200
P3	$\|V\|$	2225	**2225**	4503	4013	4503	22507	**25952**	**22507**	34408
	$\|C\|$	3297	**3297**	6105	5764	6105	20313	**23955**	**20313**	33003
	Root	0	**0**	0	0	4	0	**3.4317**	**0**	2
	Objective	0	**0**	0	0	4	0	**9**	**1**	26
	Bound	0	**0**	0	0	4	0	**4**	**1**	2
	Time	22.52	**0.66**	27.24	566.54	37.27	573.12	**7200**	**2930.74**	7200
P4	$\|V\|$	3419	3419	7017	6204	**7017**	**94027**	102023	94027	110778
	$\|C\|$	4328	4328	8126	7567	**8126**	**84051**	91004	84051	97026
	Root	0	0	0	0	**4**	**0**	3.5125	0	2
	Objective	0	0	0	0	**4**	**0**	25	1	-
	Bound	0	0	0	0	**4**	**0**	5	1	2
	Time	9.19	7.89	555.38	281.64	**24.78**	**506.25**	7200	3225.67	7200

setting, (P3) also finds the best solution to the instance **F3b** and hence scales the best with problem size while also being competitive in time to optimality on the smaller instances. On the other hand, when $b^{\max} = 1$, formulations (P3) and (P4) are competitive on the 6 smaller instances, proving optimality in the shortest runtime in 2 and 4 instances, respectively. However formulation (P4) is superior to (P3) on all 3 of the more difficult instances, and the best overall formulation under this parameter setting.

While the formulations derived from the `cost_regular` constraints have the greatest number of binary variables, they are tighter formulations that scale well to large problem sizes. Hence they better capture the structure of the B2BSOP. The root node lower bound, derived from a stronger relaxation and the MIP cuts applied by Gurobi, is observably stronger on instances **T3c** and **F3b**. It would be interesting to further observe this on data sets without a trivial optimal lower bound of 0. However, the number of states to be represented in the DFA $\mathcal{A}_3$ increases quickly with the parameter $b^{\max}$; this correlates with an increased number of variables in the MIP formulation. Therefore the simpler DFA appears to be the most appropriate when fairness is less tight.

Table 7: MIP Results with $b^{\max} = 1$

Model	Result	T2a	T2c	T3a	T3b	T3c	F3a	F3b	F3c	F4a
P1	$\|V\|$	1967	1967	3164	3175	3164	7575	**9231**	**7575**	11715
	$\|C\|$	3427	3427	5627	5687	5627	12675	**15277**	**12675**	21160
	Root	0	0	0	0	0	0	**3**	**0**	2
	Objective	0	0	0	0	4	0	**12**	**2**	30
	Bound	0	0	0	0	4	0	**3**	**0**	2
	Time	1.22	29.07	401.76	193.24	647.63	833.11	**7200**	**7200**	7200
P2	$\|V\|$	2173	2173	4007	3645	4007	13245	14923	13245	17961
	$\|C\|$	3043	3043	5361	5196	5361	12955	15563	12955	21398
	Root	0	0	0	0	2	0	3.0475	0	2
	Objective	0	0	0	0	4	0	-	4	-
	Bound	0	0	0	0	4	0	5	0	2
	Time	10.77	3.4	289.17	243.64	236.33	2313.79	7200	7200	7200
P3	$\|V\|$	**2225**	**2225**	4503	4013	4503	22507	25952	**22507**	34408
	$\|C\|$	**3297**	**3297**	6105	5764	6105	20313	23955	**20313**	33003
	Root	**0**	**0**	0	0	4	0	3.5140	**0**	2
	Objective	**0**	**0**	0	0	4	0	20	**2**	-
	Bound	**0**	**0**	0	0	4	0	5	**1**	2
	Time	**1**	**0.93**	469.77	370.42	136.69	1034.5	7200	**7200**	7200
P4	$\|V\|$	3039	3039	**5973**	**5344**	**5973**	**68359**	**74052**	**68359**	**80402**
	$\|C\|$	4030	4030	**7368**	**6930**	**7368**	**63235**	**68735**	**63235**	**74547**
	Root	0	0	**0**	**0**	**4**	**0**	**3.6699**	**0**	**2**
	Objective	0	0	**0**	**0**	**4**	**0**	**12**	**2**	**28**
	Bound	0	0	**0**	**0**	**4**	**0**	**5**	**1**	**2**
	Time	1.17	1.92	**266.28**	**3.15**	**132.14**	**552.43**	**7200**	**7200**	**7200**

5 Discussion

Table 8 summarizes the best results obtained with our MIP, CP, and LNS approaches, and recalls those of the two best approaches in [2]: SBDD and clasp are respectively an SMT solver representing the objective function as a binary decision diagram and a conflict-driven answer set solver.

Looking at the different instances, we noticed that although the **T⋆** set are generally easy, **T3c** is challenging for most of the computational approaches considered. It is quickly solved to optimality by MIP but CP and SBDD only find suboptimal solutions and clasp, none at all. It is interesting that the only difference between instances **T3a** and **T3c** is that the latter has fewer available locations: that type of restriction does not seem to be handled well by those approaches. On the harder **F⋆** set there is no clear winner but MIP, LNS, and SBDD are generally performing better than the other two.

For CP to be competitive on the harder instances, having a good model did not seem to be sufficient and it required branching heuristics geared toward optimization. This was the case even though an optimization-based global constraint was used in the formulation.

Table 8: An empirical comparison of the computational approaches proposed in this paper as well as some of the previous proposals. For MIP we report the best solution out of the four models presented and for LNS, the best solution out of the ten runs. We give the time in seconds to find the solution value reported, except for SBDD and clasp on non-optimal solutions since these were not provided.

Instance	Approach	Obj.	Time
T2a	MIP	0	7.6
	CP	0	0.5
	LNS	0	0.5
	SBDD	0	2.7
	clasp	0	0.1
T2c	MIP	0	0.7
	CP	0	3.9
	LNS	0	3.9
	SBDD	0	235.4
	clasp	0	977.9
T3a	MIP	0	9.4
	CP	0	98.5
	LNS	0	64.0
	SBDD	0	65.2
	clasp	0	2.1
T3b	MIP	0	156.3
	CP	0	12.0
	LNS	0	12.0
	SBDD	0	24.1
	clasp	0	2.1
T3c	MIP	4	24.8
	CP	11	87.8
	LNS	4	850.6
	SBDD	8	–
	clasp	–	–

Instance	Approach	Obj.	Time
F3a	MIP	0	506.3
	CP	0	1514.9
	LNS	0	5442.9
	SBDD	0	3128.1
	clasp	0	52.4
F3b	MIP	9	5833.0
	CP	23	4795.0
	LNS	13	3189.0
	SBDD	12	–
	clasp	24	–
F3c	MIP	1	2930.7
	CP	9	5070.3
	LNS	3	2507.4
	SBDD	20	–
	clasp	–	–
F4a	MIP	9	7172.0
	CP	25	1296.5
	LNS	13	5996.0
	SBDD	7	–
	clasp	–	–

The main conclusion that can be drawn is that the `cost_regular` structure is quite useful in building efficient MIP and CP models. Compared to the MIP and CP formulations proposed in [2], models using such structure are more efficient on the instances considered, and often state-of-the-art. For the MIP approach one could ask whether it is possible to tighten the more compact formulations (P1) or (P2) with valid constraints derived from the `cost_regular` constraints. As an avenue of future research, we believe that embedding some of these constraints into a branch-and-cut algorithm could result in a more robust exact algorithm for the B2BSOP.

Acknowledgments

We wish to thank the reviewers for their constructive criticism. This research was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

1. Egon Balas and Matthew J. Saltzman. An algorithm for the three-index assignment problem. *Operations Research*, 39(1):150–161, 1991.
2. Miquel Bofill, Joan Espasa, Marc Garcia, Miquel Palahí, Josep Suy, and Mateu Villaret. Scheduling B2B Meetings. In Barry O'Sullivan, editor, *Principles and Practice of Constraint Programming - 20th International Conference, CP 2014, Lyon, France, September 8-12, 2014. Proceedings*, volume 8656 of *Lecture Notes in Computer Science*, pages 781–796. Springer, 2014.
3. Marie-Claude Côté, Bernard Gendron, and Louis-Martin Rousseau. Modeling the Regular Constraint with Integer Programming. In Pascal Van Hentenryck and Laurence A. Wolsey, editors, *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, 4th International Conference, CPAIOR 2007, Brussels, Belgium, May 23-26, 2007, Proceedings*, volume 4510 of *Lecture Notes in Computer Science*, pages 29–43. Springer, 2007.
4. Sophie Demassey, Gilles Pesant, and Louis-Martin Rousseau. A Cost-Regular Based Hybrid Column Generation Approach. *Constraints*, 11(4):315–333, 2006.
5. William D. Harvey and Matthew L. Ginsberg. Limited Discrepancy Search. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, IJCAI 95, Montréal Québec, Canada, August 20-25 1995, 2 Volumes*, pages 607–615. Morgan Kaufmann, 1995.
6. Gilles Pesant, Claude-Guy Quimper, and Alessandro Zanarini. Counting-Based Search: Branching Heuristics for Constraint Satisfaction Problems. *J. Artif. Intell. Res. (JAIR)*, 43:173–210, 2012.
7. Philippe Refalo. Impact-Based Search Strategies for Constraint Programming. In Mark Wallace, editor, *CP*, volume 3258 of *Lecture Notes in Computer Science*, pages 557–571. Springer, 2004.
8. J.-C. Régin. Generalized Arc Consistency for Global Cardinality Constraint. In *Proceedings of the Thirteenth National/Eighth Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence, AAAI-98/IAAI-98*, volume 1, pages 209–215, 1996.
9. Paul Shaw. Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems. In Michael J. Maher and Jean-Francois Puget, editors, *Principles and Practice of Constraint Programming - CP98, 4th International Conference, Pisa, Italy, October 26-30, 1998, Proceedings*, volume 1520 of *Lecture Notes in Computer Science*, pages 417–431. Springer, 1998.