Solving an integrated employee timetabling and job-shop scheduling problem via hybrid branch-and-bound

Christian Artigues[1], Michel Gendreau[2], Louis-Martin Rousseau[2] and Adrien Vergnaud[2]

[1]LAAS–CNRS
7 avenue du Colonel Roche, 31077 Toulouse, France
christian.artigues@laas.fr
[2]CRT – Université de Montréal
C.P. 6128, succursale Centre-ville, Montréal, QC H3C 3J7 Canada
{michelg,louism}@crt.umontreal.ca

Abstract

We propose exact hybrid methods based on integer linear programming and constraint programming for an integrated employee timetabling and job-shop scheduling problem. Each method we investigate uses a constraint programming (CP) formulation associated with a linear programming (LP) relaxation. Under a CP framework, the LP-relaxation is integrated into a global constraint using in addition reduced cost-based filtering techniques. We propose two CP formulations of the problem yielding two different LP relaxations. The first formulation is based on a direct representation of the problem. The second formulation is based on a decomposition in intervals of the intervals of the possible operation starting times. We show the theoretical interest of the decomposition-based representation compared to the direct representation through the analysis of dominant schedules. Computational experiments on a set of randomly generated instances confirms the superiority of the decomposition-based representation. In both cases, the hybrid methods outperforms pure constraint programming for employee cost minimization while it is not the case for makespan minimization. The experiments also investigate the interest of the proposed integrated method compared to a sequential approach and show its potential for multiobjective optimization

1 Introduction

In production systems, the decisions related to scheduling jobs on the machines and the decisions related to employee timetabling are often made in a sequential process. The objective of job scheduling is to minimize the production costs whereas the objective of employee timetabling is to maximize employee satisfaction (or to minimize labor costs). Either the employee timetabling is first established and then the scheduling of jobs must take employee availability constraints into account or the scheduling of jobs is done at first and the employees must then adapt to cover the machine loads. It is well known that optimizing efficiently an integrated process would allow to balance production costs and employee satisfaction in a better way. However, the resulting problem has generally been considered as too complex to be used in practical situations. Some attempts have been made [DM94, BAL95, DHM96, AB97, Che04, HCM04, DMS04] but mostly considering an oversimplified version of the employee timetabling problem. Nevertheless the integration of task scheduling and employee timetabling has been successfully developed in complex transportation systems [BGAB83, DDI$^+$98, HF99, CSSD01, KJN02, FHW03, MCS05]. In the context of production or project scheduling, even basic integration processes, considering manpower availability in an aggregated way, have shown however substantial cost savings [BAL95].

There is consequently a potential on developing models that consider detailed timetabling of individual employees as well as detailed scheduling of job operations on machines.

In the classical employee timetabling problem [EJKS04, DPR05], the employee horizon is decomposed in shifts and the employee timetabling problem aims at assigning to each employee a single activity during each shift while minimizing the employee cost, each employee having skills for a limited number of activities. The classical job-shop scheduling problem [BDP96] considers a set of jobs and a set of machines. Each job is a chain of operations, each operation being defined by its duration and its assigned machine. The job-shop scheduling problem aims at assigning a start time to each operation such that the precedence constraints between the operations of the same job are satisfied and such that the processing intervals of any two operations assigned to the same machine do not overlap. The job-shop scheduling objective is to minimize the makespan. Integrating employee timetabling and job-shop scheduling has to be performed by defining how the employee activity demand is determined by the schedule of job operations on the machines.

In [DMS04], the assumption is made that each job operation has to be performed by its machine, plus a set of employees assigned to the operation during its entire processing time. This amounts to consider employees as additional parallel machines, each machine being able to process only a subset of operations because of limited employee skills. However in practice, the decomposition in shifts of the employee timetabling horizon leads to the case where the set of employees assigned to the operation can be modified while the operation is in process, generally at a shift change. Drezet and Billaut [DB06] propose a more complex model, in the context of resource-constrained project scheduling, taking account of this characteristic, among others. In [HBBF05], Haït *et al.* propose a general model for integrated production scheduling and employee timetabling, based on the concepts of load center, configuration, employee assignment and sequence. A so-called load center is a subset of machines that can be managed simultaneously by a single employee. A configuration is a set of load centers defining a partition of a subset of machines. At each scheduling time period, a single configuration can be active. Hence, the number of load centers in a configuration gives the number of active employees. An employee assignment is an assignment of each load center of a configuration to a different employee. The authors define the configuration graph in which each node corresponds to a possible configuration and there is an arc between two configurations that can be consecutive in time with a weight giving the cost of the configuration changeover. This model emphasizes that in production systems employee activities are more associated with machine than with operations. Furthermore, it allows to represent the simultaneous work of an employee on several machines, e.g. for supervision activities.

In this paper we propose a simple model making a synthesis of the previously defined models, in the context of a job-shop. This model basically links the employee activity to the resource usage during the employee shift, while the required skills are determined by the operations actually scheduled on the machine during this shift. The model allows also an employee to work simultaneously on several machines. We consider further flexibility in the operation schedule/activity demand mapping by allowing an employee shift to cover more than one scheduling period, aggregating the demand of the operations scheduled during the shift. We consider a lexicographic optimization problem where the makespan minimization is the primary objective whereas the employee cost minimization is the secondary objective. The problem is presented in Section 2.

The integration of two NP-hard optimization problems is unlikely to give an easily solvable problem. On one hand, the vast literature on job-shop scheduling shows that exact makespan minimization methods are based on efficient constraint propagation techniques [BPN01]. On the other hand, employee cost minimization methods, especially in the airline crew scheduling problems, are often based on integer linear programming methods [EJKS04]. It follows that

the integrated employee timetabling and job-shop scheduling problem appears as a potential application of hybrid methods, incorporating elements from both constraint programming (CP) and integer linear programming (ILP) as already successfully developed for other integrated planning and scheduling problems [FLN00, Hoo05]. The exact hybrid methods we investigate use a CP formulation associated with a linear programming (LP) relaxation. The LP relaxation is integrated into a global constraint using reduced cost-based filtering [FLM99]. We propose two CP formulations of the problem yielding two different LP relaxations. The first formulation, described in Section 3, is based on a direct representation of the problem. The second formulation, given in Section 4, is based on a decomposition in intervals of the set of possible operation start time values. We show in Section 5 the theoretical interest of the decomposition-based representation compared the direct representation through the analysis of dominant schedules. Computational experiments on a set of randomly generated instances are given in Section 6. Concluding remarks are drawn in Section 7.

2 A flexible model for integrated employee timetabling and job-shop scheduling

We consider a set of machines $\{M_k\}_{k=1,\ldots,m}$ and a set of jobs $\{J_i\}_{i=1,\ldots,n}$, each job J_i being made of a chain of operations $\{O_{ij}\}_{j=1,\ldots,m}$. Each operation O_{ij} is defined by its assigned machine $m_{ij} \in \{M_1, \ldots, M_m\}$ and its duration $p_{ij} > 0$. All durations are assumed to be integers. The set of operations assigned to machine M_k is denoted $\mathcal{O}_k$. H denotes the scheduling horizon, an upper bound of the schedule completion time.

We also consider a set of employees $\{E_e\}_{e=1,\ldots,\mu}$, a set of activities $\{A_a\}_{a=1,\ldots,\alpha}$ and a set of shifts $\{s\}_{s=0,\ldots,\sigma-1}$. All shifts s are assumed to have the same duration $\pi \geq 1$. The timetabling horizon σ is defined such that $H = \sigma \times \pi$. The cost of assigning employee E_e to activity A_a during shift s is denoted c_{eas}. $\mathcal{A}_e$ denotes the set of activities an employee is able to perform. We assume there is an additional activity A_0 representing employee inactivity and such that $A_0 \in \mathcal{A}_e$ for each employee E_e. Employee E_e is assumed to be available for a subset of shifts $\mathcal{T}_e$. Each operation O_{ij} is assumed to require during its processing an integer number $b_{ija} \geq 0$ employees for activity a.

Solving the problem lies in assigning values to the following decision variables. S_{ij} denotes the start time of operation O_{ij}. C_{ij} denotes the completion time of operation O_{ij}. $C_{\max}$ denotes the makespan of the schedule. x_{eas} is a binary variable equal to 1 if employee E_e is assigned to activity a during shift s. θ_{ijs} is a binary variable equal to 1 if operation O_{ij} is in process during shift s. δ_{as} is the number of employees required for activity a during shift s (demand). The problem can be formulated as follows, mixing for convenience linear and logic formulations.

- *Objective function:*

$$\min Lex \left(C_{\max}, \sum_{e=1}^{\mu} \sum_{a=1}^{\alpha} \sum_{s=0}^{\sigma-1} c_{eas} x_{eas} \right) \tag{1}$$

- *Job-shop specific constraints:*

$$C_{\max} \geq C_{im} \qquad i = 1, \ldots, n \tag{2}$$

$$C_{ij} = S_{ij} + p_{ij} \qquad i = 1, \ldots, n \quad j = 1, \ldots, m \tag{3}$$

$$S_{ij} \geq C_{i(j-1)} \qquad i = 1, \ldots, n \quad j = 2, \ldots, m \tag{4}$$

$$(S_{ij} \geq C_{kl}) \vee (S_{kl} \geq C_{ij}) \qquad O_{ij} \neq O_{kl}, m_{ij} = m_{kl} \tag{5}$$

$$S_{ij} \geq 0 \qquad i = 1, \ldots, n \quad j = 1, \ldots, m \tag{6}$$

- *Employee timetabling specific constraints:*

$$\sum_{e=1}^{\mu} x_{eas} \geq \delta_{as} \qquad a = 1, \ldots, \alpha \quad s = 0, \ldots, \sigma - 1 \tag{7}$$

$$\sum_{a \in \mathcal{A}_e} x_{eas} = 1 \qquad e = 1, \ldots, \mu \quad s = 0, \ldots, \sigma - 1 \tag{8}$$

$$x_{e0s} + x_{e0(s+1)} + x_{e0(s+2)} \geq 2 \qquad s = 0, \ldots, \sigma - 4 \tag{9}$$

$$\sum_{s=0}^{\sigma-1} x_{eas} = 0 \qquad e = 1, \ldots, \mu \quad a \notin \mathcal{A}_e \tag{10}$$

$$x_{e0s} = 1 \qquad e = 1, \ldots, \mu \quad s \notin \mathcal{T}_e \tag{11}$$

$$x_{eas} \in \{0, 1\} \qquad e = 1, \ldots, \mu \quad a = 0, \ldots \alpha \quad s = 0, \ldots, \sigma - 1 \tag{12}$$

- *Coupling constraints:*

$$S_{ij} < (s+1)\pi \wedge C_{ij} > s\pi \implies \theta_{ijs} = 1 \qquad i = 1, \ldots, n \quad j = 1, \ldots, m \quad s = 0, \ldots, \sigma - 1 \tag{13}$$

$$S_{ij} \geq (s+1)\pi \vee C_{ij} \leq s\pi \implies \theta_{ijs} = 0 \qquad i = 1, \ldots, n \quad j = 1, \ldots, m \quad s = 0, \ldots, \sigma - 1 \tag{14}$$

$$\delta_{as} = \sum_{k=1}^{m} \max_{O_{ij} \in \mathcal{O}_k} \theta_{ijs} b_{ija} \qquad a = 1, \ldots, \alpha \quad s = 0, \ldots, \sigma - 1 \tag{15}$$

For the job shop part, constraints (2) defines the makespan. Constraints (3) link the start and the completion time of each operation. Constraints (4) state that each operation cannot start before the completion of its job predecessor. Constraints (5) are the so-called disjunctive constraints preventing two activities assigned to the same machine from being processed in parallel. Each disjunctive constraint is non linear.

For the employee timetabling part, fully linear, constraints (7) are the standard demand cover constraints for each activity and each shift. Constraints (8) state that each employee has to be assigned exactly one activity (including "rest" activity A_0) during each shift. Constraints (9) are regulation constraints that state that each employee can work at most 1 shift during each gliding window of 3 shifts. Note that other more complex (and possibly non linear) regulation constraints could be necessary for practical applications. Constraints (10) fix assignment variables to 0 due to employee skills. Constraints (11) fix assignment variables to activity A_0 because of employee unavailability during some shifts.

The coupling part describes how operation schedule S and activity demand δ are linked. Constraint (13) and (14) set binary variable θ_{ijs} to 1 if the processing interval of operation O_{ij}, equal to $[S_{ij}, C_{ij}[$, intersects shift s interval equal to $[s\pi, (s+1)\pi[$, and to 0 otherwise. It is assumed that the employee assigned to an activity A_a distinct from A_0 during his shift is also assigned to a machine and has to perform only activity A_a on this particular machine during this shift. Therefore, suppose two operations O_{ij} and O_{kl} are in process during a shift s and require each one employee for the same activity A_a. In this case, two employees able to perform A_a are required during shift s if O_{ij} and O_{kl} are assigned to different machines while only one employee is needed if $m_{ij} = m_{kl}$. Constraints (15) compute the demand δ_{as} for each activity and each shift, summing up the demands on each machine for the same shift. The machine demand on a shift for an activity is equal to the highest demand of operations in process during this shift.

Let us consider a small example (the EJS1 instance) comprising 6 jobs, 4 machines, 4 activities 15 employees and 6 shifts. Shift duration is $\pi = 8$. Hence the scheduling horizon is $H = 48$. In Table 1 the job data (machines and durations) is provided. The durations have been randomly generated between 1 and 10. For the operations requests for activities (values b_{ija}) we consider the special case where operations require a single employee and where there is a mapping between activities and machines. Therefore, we have $b_{ijm_j} = 1$ and $b_{ija} = 0$, for all $a \neq m_{ij}$.

INSERT TABLE 1 ABOUT HERE

Employee data is displayed in Table 2. Each row corresponds to an activity the employee is able to perform. The assignment costs of the employee for this activity are displayed for all shifts. They have been randomly generated between 1 and 5 assuming they correspond to employee preferences.

INSERT TABLE 2 ABOUT HERE

An optimal solution of the problem is displayed in Figure 1. On top of the Figure, a Gantt diagram displays the start times of the operations. At the bottom, the employee timetable is displayed, giving, for each employee and each shift, the activity (machine) performed and the corresponding cost. A makespan of 45 is obtained with a total cost equal to 29. In this simple example, an employee is necessary on a machine for a shift whenever the processing interval of an operation assigned to the machine intersects the shifts. We see that only shifts 0, 2 and 5 require less than 4 employees thanks to the idleness of machines 4, 3 and 4, respectively.

INSERT FIGURE 1 ABOUT HERE

3 Direct CP formulation and LP relaxation

The job-shop scheduling part of the problem (2-6) can be modeled easily through a constraint-based scheduling library such as ILOG scheduler. The operations and the machines they require are declared as high level objects having internal decision variables. Each operation O_{ij} has a duration $O_{ij}.p$, a start variable $O_{ij}.S$ with domain $\{0, \ldots, H - O_{ij}.p\}$ and a completion variable $O_{ij}.C$ verifying $O_{ij}.C = O_{ij}.S + O_{ij}.p$. Each machine M_k is declared as a unary resource, i.e. a discrete resource with availability 1. The Makespan $C_{\max}$ is a variable with domain $\{0, \ldots, H\}$. Two types of constraints are defined on operations and machines. Binary temporal constraints are used to model the precedence constraints (4) between operations. Disjunctive resource constraints representing constraints (5) are defined on each machine M_k. In Constraint Programming, constraints can be expressed in a very general way through so-called global constraints (such as the disjunctive constraint), with the counterpart that ad-hoc constraint propagation algorithms have to be defined for each global constraint. Efficient constraint propagation algorithms exist for both precedence and disjunctive constraints. The temporal constraint propagation, disjunctive resource constraint propagation and edge-finding algorithms allow to reduce

the domains of start and completion variables during the search process. The start (completion) time domain of an operation is denoted $\{ES_i, \ldots, LS_i\}$ ($\{EC_i, \ldots, LC_i\}$). We refer to [BP96, BPN01] for more details on the filtering algorithms associated with constraint-based job-shop scheduling.

The employee timetabling part (7-12) can also be modeled in terms of constraint programming. The model involves an assignment variable A_{es} with domain $\mathcal{A}_e$ if $s \in \mathcal{T}_s$ and $\{A_0\}$ otherwise, associated with each employee E_e and each shift s. Variable A_{es} is equivalent to the 0-1 variable x_{eas} used in (7-12). We have indeed the following relation $A_{es} = \sum_{a=1}^{\alpha} a \cdot x_{eas}$. A demand variable δ_{as} with domain $\{0, \ldots, \mu\}$ is defined for each activity and each shift. We represent the timetabling problem with global constraints `element`, `distribute` and `sequence` as performed in previous work, see e.g. [DPR05].

The coupling part involves an additional demand variable δ_{ijas} for each operation O_{ij}, each activity A_a and each shift s with domain $\{0, \ldots, b_{ija}\}$.

Given the above defined operations, machines and decision variables, the considered CP formulation can be stated as the following constraint optimization problem.

$$\min Lex \left(C_{\max}, \sum_{e=1}^{\mu} \sum_{s=0}^{\sigma-1} c_{eA_{es}s} \right) \tag{16}$$

$$O_{im}.C \preceq C_{\max} \qquad i = 1, \ldots, n \tag{17}$$

$$O_{ij}.C \preceq O_{i(j+1)}.S \qquad i = 1, \ldots, n \quad j = 1, \ldots, m-1 \tag{18}$$

$$O_{ij}\,.\texttt{require}(M_k) \qquad k = 1, \ldots, m \quad O_{ij} \in \mathcal{O}_k \tag{19}$$

$$\texttt{distribute}((\delta_{\texttt{as}})_{\texttt{a}=1,\ldots,\alpha}, (\texttt{A}_{\texttt{a}})_{\texttt{a}=1,\ldots,\alpha}, (\texttt{A}_{\texttt{es}})_{\texttt{e}=1,\ldots,\mu}) \qquad s = 0, \ldots \sigma - 1 \tag{20}$$

$$\texttt{sequence}((\texttt{A}_{\texttt{es}})_{\texttt{s}=0,\ldots,\sigma-1}, \{\texttt{A}_0\}, 3, 2, 2\}) \qquad e = 1, \ldots, \mu \tag{21}$$

$$O_{ij}.S < (s+1)\pi \wedge O_{ij}.C > s\pi \implies \bigwedge_{a=1}^{\alpha} (\delta_{ijas} = b_{ija}) \qquad i = 1, \ldots, n \quad j = 1, \ldots, m$$
$$s = 0, \ldots \sigma - 1 \tag{22}$$

$$O_{ij}.S \geq (s+1)\pi \vee O_{ij}.C \leq s\pi \implies \bigwedge_{a=1}^{\alpha} (\delta_{ijas} = 0) \qquad i = 1, \ldots, n \quad j = 1, \ldots, m$$
$$s = 0, \ldots \sigma - 1 \tag{23}$$

$$\delta_{as} = \sum_{k=1}^{m} \max_{O_{ij} \in \mathcal{O}_k} \delta_{ijas} \qquad a = 1, \ldots, \alpha \quad s = 0, \ldots, \sigma - 1 \tag{24}$$

The objective function is the lexicographic minimization of the makespan and the employee total cost. The total cost is expressed as an `element` constraints as the decision variable A_{es} is used as an index in $c_{eA_{es}s}$ which is in CP the standard way of expressing a cost associated to a discrete decision variable [TO02]. Constraints (17) and (18) are the binary temporal constraints involved in the job-shop sub-problem. Constraints (19) are the disjunctive constraints. Constraints (20) represent demand satisfaction through the global cardinality constraint `distribute` [Rég96] which states that for a given shift s and for each activity A_a exactly δ_{as} variables must have value A_a in the activity assignment vector $(A_{es})_{e \in \mathcal{E}}$ of employees during period s. Constraints (21) represent regulation constraints (9) by means of the `sequence` constraint [BC94] which states that in each sub-sequence of q variables of the sequence $(A_{es})_{s=0,\ldots,\sigma-1}$, at most 2 and at least 2 variables must have the value A_0. Note that this constraint could be alternatively represented by constraint `regular` which would be also suitable to represent more complex regulation rules [Pes04, DPR05, vPRS06]. Constraints (22-24) are a direct transcription of coupling constraints (13-15) using demand variable δ_{ijas} instead of binary variable θ_{ijs}.

The CP model is solved by a backtrack search on the variable values after transforming the optimization problem into a series of decision (satisfaction) problems. The lexicographic minimization problem is solved by first finding the minimum makespan through binary search. The makespan is then fixed to its minimal feasible value and a second binary search is performed to minimize the total employee cost. At each node of the involved search trees, constraint propagation algorithms are used to detect inconsistencies and reduce the domains of the variables. As previously stated, each predefined global constraint is associated with a specific constraint propagation algorithm.

In addition, we propose to integrate a LP relaxation of the model. The considered LP relaxation is reduced to the timetabling problem with variables x_{aes}. It is precisely the LP relaxation of the ILP $\min \sum_{e=1}^{\mu} \sum_{a=1}^{\alpha} \sum_{s=0}^{\sigma-1} c_{eas} x_{eas}$ subject to $(7-12)$, except that variable δ_{as} is replaced by the smallest value in its current domain in the demand cover constraint (7). Variables x_{eas} are preprocessed by analyzing the current domain of variable A_{es}: each variable x_{eas} such that $a \notin A_{es}$ is set to 0 and each variable x_{eas} such that $A_{es} = \{A_a\}$ is set to 1.

The resolution of the LP relaxation is embedded into the constraint propagation algorithm of a global constraint `demandCover`. During search, the constraint propagation algorithm of `demandCover` is called each time the lower bound of any variable δ_{as} is changed, which corresponds to a modification of the demand cover constraint, or when the domain of any assignment variable A_{es} is modified, which corresponds to fixing some variables x_{aes}. Different rules for propagation of the `demandCover` constraint are investigated in Section 6.

For the makespan minimization phase, each time the LP is unfeasible, `demandCover` fails and the current node can be pruned. For the total employee cost minimization phase, further reductions can be performed with reduced-cost based variable fixing techniques [FLM99]. Besides infeasibility, `demandCover` also fails if the optimal cost of the LP relaxation is not lower than the current upper bound Z of the total cost value (set by the binary search). If the LP is feasible then let $\tilde{C}_{eas}$ denote the reduced cost of an activity assignment variable x_{eas} and let TC^* denote the current optimal LP solution value. The reduced-cost based domain reduction technique states that if $TC^* + \tilde{C}_{eas} > Z$ then activity A_a can be removed from the domain of A_{es}.

4 Decomposition-based CP formulation and LP relaxation

The CP formulation and the LP relaxation defined in the previous Section have the drawback that the job-shop scheduling part of the problem is absent of the LP relaxation. In this Section, we propose a second formulation based on a decomposition of the domains of the operation start time variables into disjoint intervals such that the selection of an interval for an operation determines its demand for each activity on all the employee shifts.

When an operation O_{ij} starts at time t it intersects all shifts s such that $(s+1)\pi > t$ and $s\pi < t + p_{ij}$. This allows to compute for each time point $t \in \{ES_i, \ldots, LS_i\}$ the demand profile $\delta_{ijas}(t)$ for each activity a and each shift s. Then all (consecutive) time points $t1, t2 \in \{ES_i, LS_i\}$ such that $\delta_{ijas}(t_1) = \delta_{ijas}(t_2)$ for all shifts $s = 0, \ldots, \sigma-1$ can be grouped into the same *demand interval*. With this process, the domain of the start time of each operation O_{ij} can be partitioned into consecutive demand intervals $I_1, I_2, \ldots, I_{v_{ij}}$. Each interval I_q corresponds to a demand b_{ijqas} for each activity a and each shift s. Demand interval I_q comprises values $\{ES_{ijq}, \ldots, LS_{ijq}\}$ such that $LS_{ijq} + 1 = ES_{ij(q+1)}$ for $1 \leq q < v_{ij}$, $ES_{ij} = ES_{ij1}$ and $LS_{ij} = LS_{ijv_{ij}}$. Note that the number of demand intervals v_{ij} cannot exceed $2\sigma + 1$ where σ is the number of shifts.

An illustrative example is displayed in Figure 2 where the start time domain of an operation is partitioned into 5 intervals. The demand profiles of the operation for each interval and for an activity a such that $b_{ija} = 1$ are displayed at the bottom of the Figure.

INSERT FIGURE 2 ABOUT HERE

We modify the constraint programming model presented in the preceding Section by introducing a decision variable I_{ij} for each operation O_{ij} with domain $\{I_1, \ldots, I_{v_{ij}}\}$. The job-shop scheduling and employee timetabling part of the CP model (16-21) do not change but the following constraints (comprising `element` constraints) are added to enforce the correspondence between the start times and the demand intervals

$$O_{ij}.S \succeq ES_{ijI_{ij}} \qquad i = 1 \ldots, n \quad j = 1 \ldots, m \tag{25}$$
$$O_{ij}.S \preceq LS_{ijI_{ij}} \qquad i = 1 \ldots, n \quad j = 1 \ldots, m \tag{26}$$

Furthermore the coupling constraints (22-23) are replaced by the following constraints, also comprising `element` constraints:

$$\delta_{ijas} = b_{ijI_{ij}as} \qquad i = 1 \ldots, n \quad j = 1 \ldots, m \quad a = 1, \ldots, \alpha \tag{27}$$

Constraint (24) is left unchanged.

Besides the simplification of the coupling constraints, this new model allows to introduce the scheduling part in the LP relaxation by defining a new binary variable y_{ijq} equal to 1 if O_{ij} is assigned to demand interval I_q and to 0 otherwise. The variable corresponds to the discrete variable I_{ij}. Hence the new variables are preprocessed as follows. Each variable y_{ijq} such that $I_q \notin I_{ij}$ is set to 0 and each variable y_{ijq} such that $I_{ij} = \{I_q\}$ is set to 1. A continuous demand variable d_{ask} giving the demand for activity A_a on shift s issued from machine M_k is defined as an intermediate variable.

The model comprises constraints (8-12). The cover constraint (7) is removed and replaced by the following constraints:

$$\sum_{q=1}^{v_{ij}} y_{ijq} = 1 \qquad i = 1, \ldots, n \quad j = 1, \ldots, m \tag{28}$$
$$d_{ask} \geq \sum_{q=1}^{v_{ij}} y_{ijq} b_{ijqas} \qquad a = 1, \ldots, \alpha \quad s = 0, \ldots, \sigma - 1 \quad k = 1, \ldots, m \quad O_{ij} \in \mathcal{O}_k \tag{29}$$
$$\sum_{e=1}^{\mu} x_{eas} \geq \sum_{k=1}^{m} d_{ask} \qquad a = 1, \ldots, \alpha \quad s = 0, \ldots, \sigma - 1 \tag{30}$$
$$y_{ijq} \in \{0.1\} \qquad i = 1, \ldots, n \quad j = 1, \ldots, m \quad q = 1, \ldots, v_{ij} \tag{31}$$

Constraints (28) state that only one interval has to be selected for each operation. Constraints (29) states that demand d_{ask} of machine M_k for activity A_a on shift s is not lower than the demand for activity A_a on shift s of each operation O_{ij} assigned to machine k, which is set by the assigned interval. Constraints (30) enforce the employees assigned to activity A_a on shift s to cover the total demand obtained by summing up all machine demands.

In addition, scheduling constraint can be added to the LP. First a makespan variable can be introduced. Let $LPC_{\max}$ denote this variable. We have:

$$LPC_{\max} \geq \sum_{q=1}^{v_{ij}} ES_{ijq} y_{ijq} + p_{ij} \qquad i = 1, \ldots, n \quad j = 1, \ldots, m \tag{32}$$

Second, fixing variables y_{ijq} assigns a time interval to the start time of operation O_{ij}. We can apply the principle of energetic reasoning feasibility checks [ELT89, BPN01] to the set of operations scheduled on the same machine. Let $P_{ijq}(t_1, t_2)$ denote the minimal part of operation O_{ij} scheduled in interval $\{t_1, \ldots, t_2\}$ with $t_2 > t_1$ when the operation is assigned to interval I_q. This minimal part, although referred to as minimal energy consumption, is defined as

$$P_{ijq}(t_1, t_2) = \min\left(P^-_{ijq}(t_1, t_2), P^+_{ijq}(t_1, t_2)\right)$$

where $P^-_{ijq}(t_1, t_2)$ is the minimal energy consumption of O_{ij} when it is left shifted in interval I_q and $P^+_{ijq}(t_1, t_2)$ is the minimal energy consumption of O_{ij} when it is right shifted. We have:

$$P^-_{ijq}(t_1, t_2) = \min\left(t_2 - t_1, \max(ES_{ijq} + p_{ij} - t_1, 0)\right)$$
$$P^+_{ijq}(t_1, t_2) = \min\left(t_2 - t_1, \max(t_2 - LS_{ijq}, 0)\right)$$

Then energetic reasoning feasibility check for time interval $\{t_1, \ldots, t_2\}$ can be written as follows on each machine M_k.

$$\sum_{O_{ij} \in \mathcal{O}_k} \sum_{q=1}^{v_{ij}} P_{ijq}(t_1, t_2) y_{ijq} \leq t_2 - t_1$$

The feasibility check is valid for each interval $\{t_1, \ldots, t_2\}$. We propose to add to the linear relaxation the following constraints, selecting only intervals corresponding to shifts.

$$\sum_{O_{ij} \in \mathcal{O}_k} \sum_{q=1}^{v_{ij}} P_{ijq}(s\pi, (s+1)\pi) y_{ijq} \leq \pi \qquad \forall s = 0, \ldots, \sigma - 1 \tag{33}$$

The LP relaxation of the new CP formulation is the relaxation of the ILP $\min f$ subject to (8-12), (28-31), (32), (33) where f is either $C_{\max}$ or $\sum_{e=1}^{\mu} \sum_{a=1}^{\alpha} \sum_{s=0}^{\sigma-1} c_{eas} x_{eas}$ depending on the optimization phase.

The LP formulation is embedded in a new global constraint `demandIntervalCover`. The constraint propagation algorithm of `demandIntervalCover` is called under the same conditions as `demandCover` except that the LP resolution can also be called when a value is assigned to a demand interval variable I_{ij}. Several variants of the propagation rules are compared in Section 6. After solving the LP relaxation, the algorithm performs additional reduced cost-based domain filtering on variables I_{ij} using the reduced-costs of variables y_{ijq} under the same principle as for variables x_{eas}.

5 Search method and dominance properties

To solve the problem with the two CP formulations, the search method has to be specified. As already explained the CP formulations are both solved by backtrack search on the variable values.

For the direct CP formulation, the assignment of variables is made sequentially as follows (a backtrack point being set each time a decision is taken):

1. Rank the operations on the machines.
2. Set the start times of the operations.
3. Assign the activities to employees.

For the decomposition-based CP formulation, the assignment of variables includes additionally the assignment of demand interval variables:

1. Rank the operations on the machines
2. Assign the demand intervals to the operations.
3. Assign the activities to employees
4. Set the start times of the operations

For both models, the first step lies in ordering the operations on the machines. In any feasible solution, the operations sharing the same machine are ordered because of the corresponding disjunctive constraint. Predefining the order reduces the search space for feasible start times. Furthermore, this increases the domain reduction performed by the disjunctive and edge-finding constraint propagation algorithms which reason on operation time windows [BPN01]. The ranking of the operation is performed by the algorithm `RankForward` of the ILOG scheduler library.

The start time and activity assignment steps are reversed for two models. It is intuitively preferable that the demand is fixed when assigning activities to employee. In the first model, the demand is fixed when the start times of the operations are defined or sufficiently reduced. In the second model, the demand is fixed when the demand intervals are assigned, which is performed in the second step of the search process.

We now try to answer the following question. For the start time assignment process, do we have to enumerate all possible values in the start time domains of each operation ? For the standard job-shop problem it is well-known that left-shifted (also called semi-active) schedules are dominant. A simpler counter-example displayed in Figure 3 shows this is unfortunately not true for the considered problem. The illustrative instance has 2 jobs, 2 machines 2 employees. There is a mapping between the machines and the activities and the operation requires one employee each. Each employee cost is unitary. Employee E_1 is only able to perform activity M_1. Employee E_2 is only able to perform activity M_2. In the left of the Figure, a semi-active schedule is displayed. The schedule is of minimal makespan and has a total employee cost equal to 4 (all machines are used in each shift). As displayed in the right part of the Figure, right shifting operation O_{21} until the second shifts saves one cost unit.

INSERT FIGURE 3 ABOUT HERE

The negative consequence is that for the first model we cannot use an algorithm restricting the search to left-shifted schedules as performed by the ILOG scheduler algorithm `setTimes`. It follows that for search step 2 of the first model, an enumeration of all the possible values of the start times is necessary. However the following theorem holds.

Theorem 1 *If each operation has an assigned domain interval, the semi-active schedules are dominant*

Proof. Let us assume that all operations are assigned to a demand interval and consider a non semi-active schedule. Since the schedule is not semi active there exists an operation O_{ij} such that replacing S_{ij} by $S_{ij} - 1$ yields a feasible solution. Since this modification keeps by definition the operation in its assigned demand interval, the total demand is not modified and the employee activity assignment remains feasible. Hence the total employee cost does not change while the makespan cannot increase by the modification. It follow that from any non semi-active schedule

we can obtain a semi-active schedules by applying a set of elementary modifications without deteriorating the considered cost. ■

Consequently, for the last search step of the second model, only left-shifted schedules can be considered. The search algorithm `setTimesForward` of the ILOG scheduler library is used.

For the assignment of domain interval to operations, the enumeration is performed in the chronological order of the zones.

For the assignment of activities to employee we follow the following principle. We search for the first period p for which the minimal demand for an activity a is not covered by the current assignment of employees. This occurs when the number of employees for which variable A_{es} is fixed and set to a is lower than the smallest value in the domain of variable δ_{as}. In this case we search for the unassigned variable A_{es} that includes a in its domain that induces the minimal cost c_{eas}. If no variable is found, a fail occurs. Otherwise, the branching is done by selecting value A_a for A_{es} in the left branch and by removing value a for domain s in the right branch. In the case the minimal demand is covered, the branching is done by assigning a rest activity A_0 to the unassigned variable A_{es} with the smallest domain in the left branch and by removing the rest activity for this variable in the right branch.

Several variants of the methods are compared on the following Section.

6 Instance generation and computational experiments

Since no previous work exists for the considered integration of the job shop problem and the employee timetabling problem, we have developed an instance generator. The instance generator has been made as a java applet and is available on-line at the address `http://www.crt.umontreal.ca/~adrien/instances.htm`. The generator applet takes as input the problem parameters as displayed in Figure 4. The *number of extra employees parameters* defines the subset of "extras" employees able to perform all activities at a high cost. Such employees can be added to allow the feasibility of the minimal makespan schedule and/or to obtain a feasible instance. Indeed in this paper we do not consider instances where the schedule feasibility is limited by the employee availability. To solve such instances with our method, fictitious extra employees with arbitrarily large costs have to be added. All costs concerning the regular employees are generated between 1 and 5 to represent the preferences. All cost of extra employee are equal to 9. Note these extra employees could in fact be outsourced personel or regular staff working overtime

INSERT FIGURE 4 ABOUT HERE

To test the proposed methods, we have generated several sets of instances. For each set the parameters are displayed in Table 3. We have generated 4 set of instances of increasing size where a mapping between the resources and the activities exists and for which each operations requires exactly one employee. The illustrative instance ejs1 belongs to this set. We have generated a set of instances where the mapping does not exists, defining 6 jobs, 4 machines, 5 activities and 50 employees where each operation may require at most 2 activities simultaneously and at most 2 employees for each activity. Except for series ejs, all instances are such that employees have the required skills for 4 activities.

INSERT Table 3 ABOUT HERE

As stated in Section 3, the lexicographic minimization problem is solved by first finding the minimum makespan through binary search. The makespan is then fixed to its minimal feasible value and a second binary search is performed to minimize the total employee cost. A maximal CPU time limit of 1800 seconds is set to each backtrack search (i.e. to each iteration of the binary searches). For constraint based scheduling we use ILOG Solver 6.1 and Scheduler 6.1. For LP resolution we use ILOG Cplex 9.1. All programs are coded in C++ under Linux on a 64-bit AMD architecture.

Several variants of the method are tested. First, a pure CP resolution involving no LP relaxation can be applied to the problem. The results of the resolution of the proposed direct and decomposition-based CP formulations are displayed in Table 4 for a selection of 11 instances of the 4 categories. For both the direct and the decomposition-based formulations, column M/C gives the minimal makespan solution and its total employee cost. Columns CPU(F) gives the CPU times in seconds and the total number of fails needed to obtain the corresponding result. Note that for both formulation the minimal makespan solution is found for all instances in very fast computational times. Therefore, we do not need to test any hybrid method for the makespan minimization on the instances we selected. Column C^* gives the best upper and lower bound of the total cost obtained for the second phase. For both formulations and on all instances, the method is unable to improve the initial employee cost. Indeed, the time limit of 1800 seconds is reached for each instance and each formulation (TL indication in the CPU(F) column). Both CP formulations appears inefficient to minimize the employee costs.

In the remaining, 12 variants of hybrid methods are compared to solve the employee cost minimization at minimal makespan. The variants are obtained by the different possible combinations of the values the following parameters. The first parameter is the used formulation (direct or decomposed). The second parameter is linked to reduced-cost based filtering, that can be switched on or off. The last parameter corresponds to the frequency of calls of the propagation algorithm of the LP global constraint (`demandCover` or `demandIntervalCover`). The propagation algorithms of the LP global constraint can be called at specified events, defining the frequency of the LP resolution (and possibly the reduced cost-based filtering algorithms). The following propagation events (a), (b) and (c) are tested corresponding to increasing call frequencies:

(a) when the lower bound of the per shift activity demand δ_{as} changes,

(b) when (a) holds or when the domain of an employee/activity assignment variable A_{es} is changed,

(c) when (b) holds or when a demand interval variable I_{ij} is fixed for an operation (only for `demandIntervalCover`).

Intuitively, calling more frequently the LP resolution (and the reduced cost-based filtering algorithms) yields better pruning of the search tree at the possible expense of extra CPU times. Hence a good compromise has to be found.

The results of the direct and the decomposed formulations, with and without reduced cost filtering, are given in Table 5 for propagation rule (a) and in Table 6 for propagation rule (b). Last the results of the decomposition-based formulation only, with and without reduced cost filtering, are given in Table 7 for propagation rule (c). In each Table and for each parameter combination, column C^* gives the optimal cost found by the method or the upper and lower bound in the case the time limit is reached. Column CPU(F) gives the CPU time and the number of fails needed to obtain the optimal solution. TL indicates when the time limit is reached before getting the optimal solution. For each instance, the best result is displayed in bold (optimal found at minimal CPU time or best lower and upper bounds found).

The proposed methods are able to solve all instances but one (ejs10x10_3) to optimality. The theoretical superiority of the decomposition-based CP formulation and of its relaxation is verified in practice since the decomposed formulation obtains the best results on 9 instances out of 11. For the direct formulation, the reduced cost-based filtering technique is very efficient since the method can solve only 2 instances if this filtering method is switched off. With reduced cost-based filtering, the direct formulation solves all instances but one with propagation rule (a) only. The decomposition-based formulation solves also all instances but one with propagation rules (b) and (c), with and without reduced cost-based filtering. However the best results for the decomposed formulation are obtained without reduced cost-based filtering for propagation rule (b). More precisely, for the decomposition formulation, using reduced cost-based filtering always increases the CPU times, except for one instance (ejs8x8_1) with propagation rules (b) and (c) while it seems to reduce the CPU time for the solved instances with propagation rule (a). This could be interpreted by the fact that each reduction of an activity assignment of interval variable domain made by reduced cost-based filtering calls again the propagation of the global LP constraints which calls again reduced-cost-based filtering, and so on. This process, carrying out too many calls of the propagation algorithm, could be limited by solving the LP relaxation and performing reduced-cost based filtering only once at each node of the search tree.

The experiments clearly show that the minimum cost is a lot smaller that the cost of the initially found solution. However, this observation alone is not enough to argue in favor of integrated model: it lacks a third quantity: the minimal cost obtained if, in phase 2, not only the makespan was fixed, but all the scheduling variables. If this last quantity is close to the initial cost solution, integration is valuable. If it is close to the min cost solution produced by phase 2 of the model, then integration is not valuable. This method is the one usually applied in practice, that is to schedule first production and then the human resources

Table 8 shows the employee costs obtained by a sequential approach, after fixing for phase 2, the start time variables of the first minimal makespan solution. The solutions are obtained by the decomposed formulation without reduced cost based filtering and propagation rule (b). The interest of integration appears clearly for 7 problems out of 11 for which the cost obtained by the non-integrated approach varies from 4.8% to 36.5% above the cost obtained by the integrated approach.

So far, we only considered the lexicographic order for the objective function. This order is not always the best choice, because if they were an almost-minimal makespan schedule with a far smaller employee cost, it would certainly be preferable. Our last experiment, reported in Table 9, shows the solutions obtained by the integrated approach when the makespan cannot exceed 5% of the optimal makespan. We used the decomposed formulation without reduced cost based filtering and propagation rule (b). Although the problem becomes much more difficult to solve for our approach (only 7 instances out of 11 are solved to optimality), the result show the method can be helpful in a multiobjective framework using the epsilon-constraint method. With a slight makespan increase (see column M) substantial cost gains (from 9.4% up to 29.5%) are obtained on 6 instances ou of 11. We note especially the large gains obtained for the instances of the series ejs6x4x2x2 where there is no mapping between operations and activities, which yields more flexibility for operator assignment. For the large instances however no solutions with improving cost could be found.

We also point out that the epsilon-contraint method can be used to solve problems for which there is no possibility too hire extra employees. In this case, as stated above, extra employees can be added with an arbitrarily large cost and the makespan can be progressively increased until no extra employee is present in the solution.

7 Concluding remarks

The obtained results illustrate the potential cost savings brought by integrating employee timetabling and production scheduling. Indeed there is a significant difference for all instances between the cost of the initial makespan-minimal solution (column M/C of Table 4) and the final obtained optimal cost. There is also substantial employee cost improvements brought by the integrated method in comparison with a sequential approach. The method also showed its potential for multiobjective optimisation. Further research should focus on developing efficient heuristics for problems of realistic size. We hope the exact method we developed in this study will be useful to evaluate the quality of the heuristics.

The computational experiments we have carried out on a set of randomly generated instances confirms the theoretical interest of the proposed decomposition-based CP formulation of the integrated employee timetabling and production scheduling problem and of its associated LP relaxation. The key point of the decomposition lies in the possibility of restricting to semi-active schedules as soon as intervals have been assigned to operations. The main interest of the proposed LP relaxation is to incorporate aggregated scheduling decisions in the LP model with the help of energetic reasoning concepts.

Reduced cost-based filtering techniques confirm their interest for integrated planning and scheduling problems. However, depending on the used formulation and LP relaxation, the frequency of calls of the propagation algorithms have to be tuned carefully. In all cases, the hybrid method were able to obtain optimal solutions for the employee-cost minimization problem at minimal makespan while pure CP-based method could not improve the initial cost. The results we obtained illustrate that (hybrid) methods have to be designed taking account of the structure and properties of the particular problem with the help of scheduling theory.

References

[AB97] H. Alfares and J. Bailey. Integrated project task and manpower scheduling. *IIE Transactions*, 29(9):711–717, 1997.

[BAL95] J. Bailey, H. Alfares, and W. Lin. Optimization and heuristic models to integrate project task and manpower scheduling. *Computers and Industrial Engineerings*, 29:473–476, 1995.

[BC94] N. Beldiceanu and E. Contejean. Introducing global constraints in chip. *Journal of Mathematical and Computer Modelling*, 20(12):97–123, 1994.

[BDP96] J. Blazewicz, W. Domschke, and E. Pesch. The job shop scheduling problem: Conventional and new solution techniques. *European Journal of Operational Research*, 93:1–33, 1996.

[BGAB83] L. Bodin, B. Golden, A. Assad, and M. Ball. Routing and scheduling of vehicles and crews the state of the art. *Computers and Operations Research*, 10(2):63–211, 1983.

[BGHS05] P. Baptiste, V. Giard, A. Haït, and F. Soumis, editors. *Gestion de production et ressources humaines*. Presses Internationales Polytechnique, 2005.

[BP96] Ph. Baptiste and C. Le Pape. Disjunctive constraints for manufacturing scheduling: Principles and extensions. *International Journal of Computer Integrated Manufacturing*, 9(4):306–310, 1996.

[BPN01] Ph Baptiste, C. Le Pape, and W. Nuijten. *Constraint-Based Scheduling*. Kluwer, 2001.

[Che04] Z. Chen. Simulataneous job scheduling and resource allocation on parallel machines. *Annals of Operations Research*, 129:135–153, 2004.

[CSSD01] J.-F. Cordeau, G. Stojković, F. Soumis, and J. Desrosiers. Benders decomposition for simultaneous aircraft routing and crew scheduling. *Transportation science*, 35:375–388, 2001.

[DB06] L.-E. Drezet and J.-C. Billaut. Predictive and proactive approaches for RCPSP with labour constraints. In *12th IFAC Symposium on Information Control Problems in Manufacturing (INCOM06)*, Saint-Etienne, 2006.

[DDI+98] G. Desaulniers, J. Desrosiers, I. Ioachim, M.M. Solomon, F. Soumis, and D. Villeneuve. A unified framework for deterministic time constrained vehicle routing and crew scheduling problems. In T. Crainic and G. Laporte, editors, *Fleet management and logistics*, pages 57–93. Kluwer, 1998.

[DHM96] R. L. Daniels, B. J. Hoopes, and J. B. Mazolla. Scheduling parallel manufacturing cells with resource flexibility. *Management science*, 42:1260–1276, 1996.

[DM94] R. L. Daniels and J. B. Mazolla. Flow shop scheduling with resource flexibility. *Operations Research*, 42(3):504–522, 1994.

[DMS04] R. L. Daniels, J. B. Mazolla, and D. Shi. Flow shop scheduling with partial resource flexibility. *Management Science*, 50(5):658–669, 2004.

[DPR05] S. Demassey, G. Pesant, and L.-M. Rousseau. Constraint programming based column generation for employee timetabling. In *7th International Conference on Integration of AI and OR techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR'05)*, Prague, 2005.

[EJKS04] A.T. Ernst, H. Jiang, M. Krishnamoorthy, and D. Sier. Staff scheduling and rostering: A review of applications, methods and models. *European Journal of Operational Research*, 153:3–27, 2004.

[ELT89] J. Erschler, P. Lopez, and C. Thuriot. Scheduling under time and resource constraints. In *Proc. of Workshop on Manufacturing Scheduling, 11th IJCAI*, Detroit, USA, 1989.

[FHW03] R. Freling, D. Huisman, and A. Wagelmanss. Models and algorithms for integration of vehicle and crew scheduling. *Journal of Scheduling*, 6(1):63–85, 2003.

[FLM99] F. Focacci, A. Lodi, and M. Milano. Cost-based domain filtering. In *Fifth International Conference on the Principles and Practice of Constraint Programming CP'99*, volume 1713 of *Lecture Notes in Computer Science*, pages 189–203. Springer-Verlag, 1999.

[FLN00] F. Focacci, P. Laborie, and W. Nuijten. Solving scheduling problems with setup times and alternative resources. In *AIPS*, pages 92–111, 2000.

[HBBF05] A. Haït, P. Baptiste, N. Brauner, and G. Finke. Approches intégrées à court terme, 2005. chapter 6 in [BGHS05].

i	m_{ij}				p_{ij}			
1	3	2	0	1	3	7	2	9
2	3	0	1	2	8	10	3	1
3	2	1	3	0	3	4	8	6
4	1	3	0	2	10	3	3	9
5	2	0	3	1	10	8	4	7
6	2	1	0	3	2	3	10	4

Table 1: Job data

[HCM04] F. Huq, K. Cutright, and C. Martin. Employee scheduling and makespan minimization in a flow shop with multi-processor work stations: a case study. *Omega*, 32:121–129, 2004.

[HF99] K. Haase and C. Friberg. An exact algorithm for the vehicle and crew scheduling problem. In N. Wilson, editor, *Computer-Aided Transit Scheduling*, volume 471 of *Lecture Notes in Economics and Mathematical Systems*, pages 63–80. Springer, 1999.

[Hoo05] J. N. Hooker. A hybrid method for planning and scheduling. *Constraints*, 10:385–401, 2005.

[KJN02] D. Klabjan, E.L. Johnson, and G.L. Nemhauser. Airline crew scheduling with time windows and plane count constraints. *Transportation science*, 36:337–348, 2002.

[MCS05] A. Mercier, J.-F. Cordeau, and F. Soumis. A computational study of benders decomposition for the integrated aircraft routing and crew scheduling problem. *Computers and Operations Research*, 32:1451–1476, 2005.

[Pes04] G. Pesant. A regular language membership constraint for finite sequences of variables. In *Tenth International Conference on Principles and Practice of Constraint Programming -CP'04*, volume 3258 of *Lecture Notes in Computer Science*, pages 482–495. Springer-Verlag, 2004.

[Rég96] J.-C. Régin. Generalized arc consistency for global cardinality constraint. In *AAAI/IAAI*, pages 209–215. AAAI Press/The MIT Press, 1996.

[TO02] E. S. Thorsteinsson and G. Ottoson. Linear relaxations and reduced-cost based propagation of continuous variable subscripts. *Annals of operations research*, 115:15–29, 2002.

[vPRS06] W.-J van Hoeve, G. Pesant, L.-M. Rousseau, and A. Sabharwal. Revisiting the sequence constraint. In *CP-06. Proceedings of the 12th International Conference on Principles and Practice of Constraint Programming*, Nantes, 2006. to appear.

e	a	c_{eas}							
1	2	5	4	2	1	4	1	5	5
	4	3	4	4	4	2	4	4	4
2	1	4	2	2	1	2	4	3	1
	3	5	4	4	4	1	1	2	2
3	1	2	2	1	1	5	3	2	1
	4	4	5	1	2	4	3	5	1
4	2	5	2	4	1	5	3	1	4
	3	1	2	5	4	3	3	4	3
5	2	4	4	4	2	4	5	2	5
	3	5	5	2	1	3	3	2	2
6	1	2	4	4	5	4	2	5	5
	4	5	2	1	5	4	3	1	4
7	3	5	2	1	3	1	4	3	1
	4	2	3	5	3	5	4	5	3
8	1	5	5	1	1	5	4	3	2
	2	1	4	1	5	3	1	5	1
9	1	3	3	3	3	3	2	2	1
	2	1	5	5	2	5	3	3	5
10	3	3	5	1	3	4	4	4	1
	4	5	5	1	5	4	1	3	1
11	2	5	4	2	4	1	3	2	1
	3	5	1	5	2	5	2	3	1
12	1	3	2	3	4	2	1	2	4
	4	4	5	5	1	3	1	2	5
13	1	5	5	4	2	3	5	4	5
	4	2	3	2	3	1	3	4	2
14	2	5	3	3	5	1	2	3	5
	3	5	4	2	5	2	5	5	3
15	1	4	1	2	2	2	1	5	4
	4	1	2	5	3	5	4	1	4

Table 2: Employee data

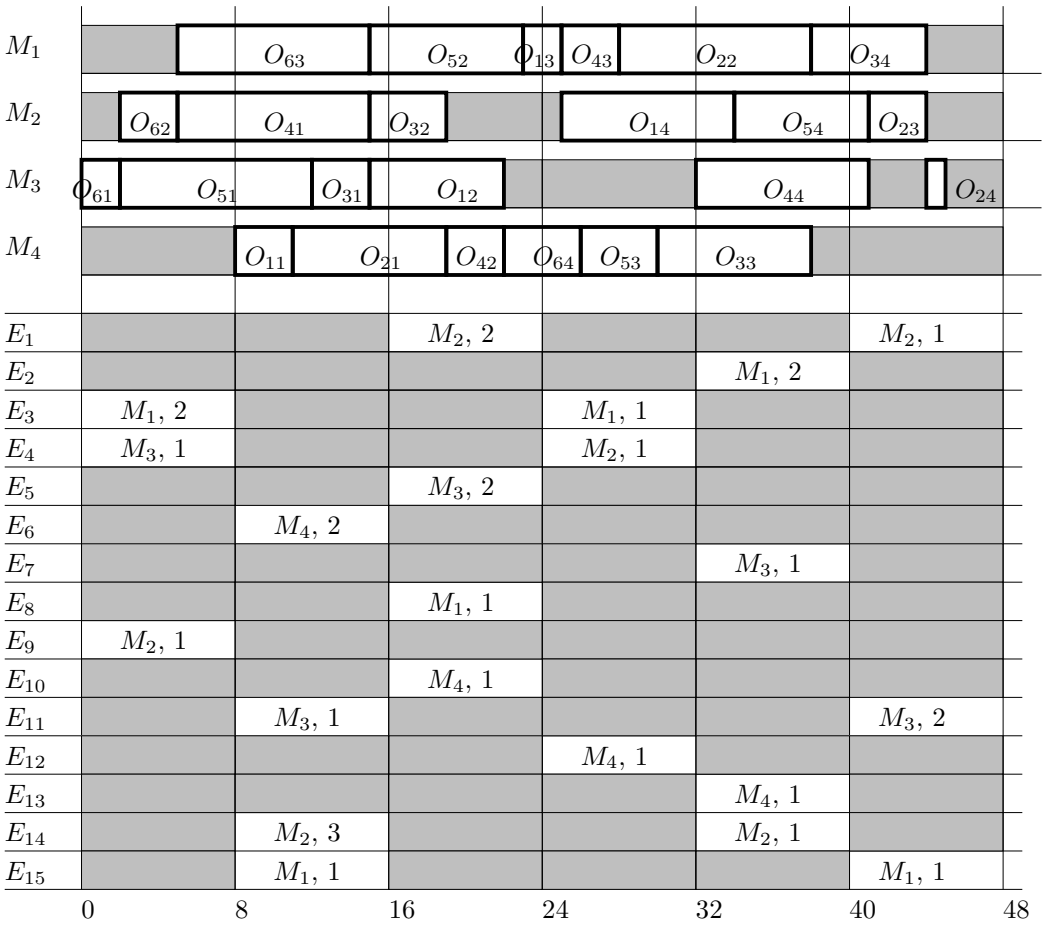

Figure 1: Optimal solution of the EJS1 instance

Inst name	n	m	μ	extra	α	mapping	σ	π	max act per oper	max empl per act	max skill per empl
ejs	6	4	25	10	4	yes	8	8	1	1	2
ejs8x8	8	8	40	20	8	yes	8	8	1	1	4
ejs10x10	10	10	50	20	10	yes	10	10	1	1	4
ejs6x4x2x2	6	4	50	20	5	no	9	10	2	2	4

Table 3: Parameters of the generated instances

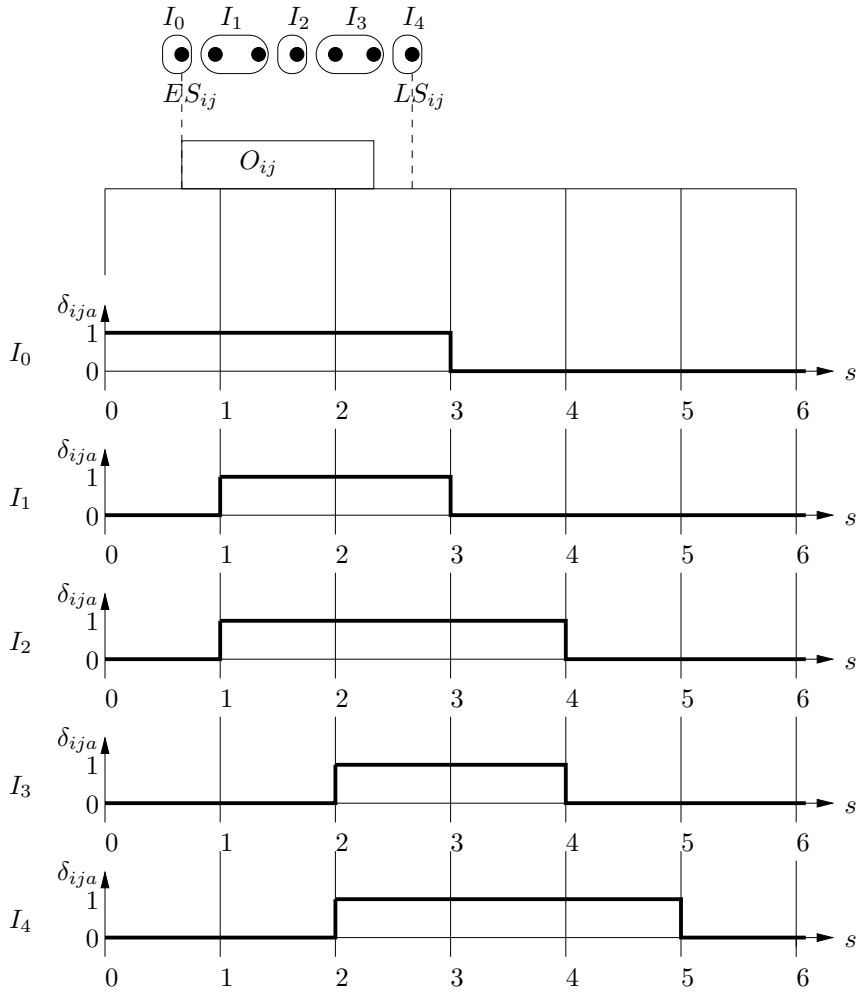

Figure 2: Decomposition of a start time domain in demand intervals

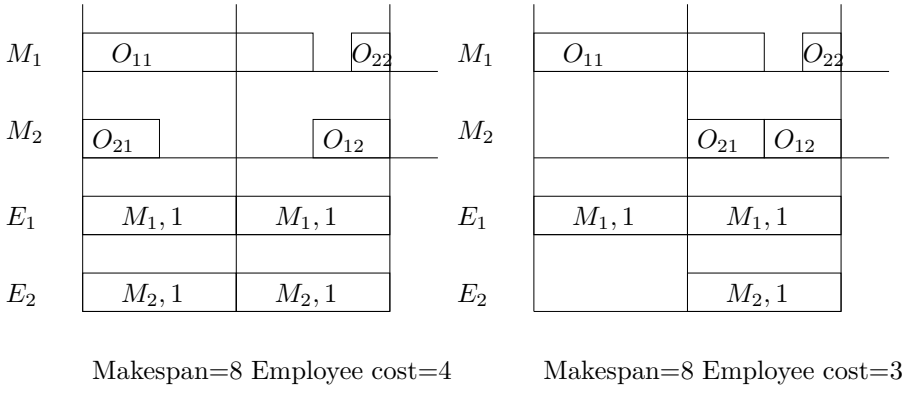

Figure 3: Non dominance of semi-active schedules

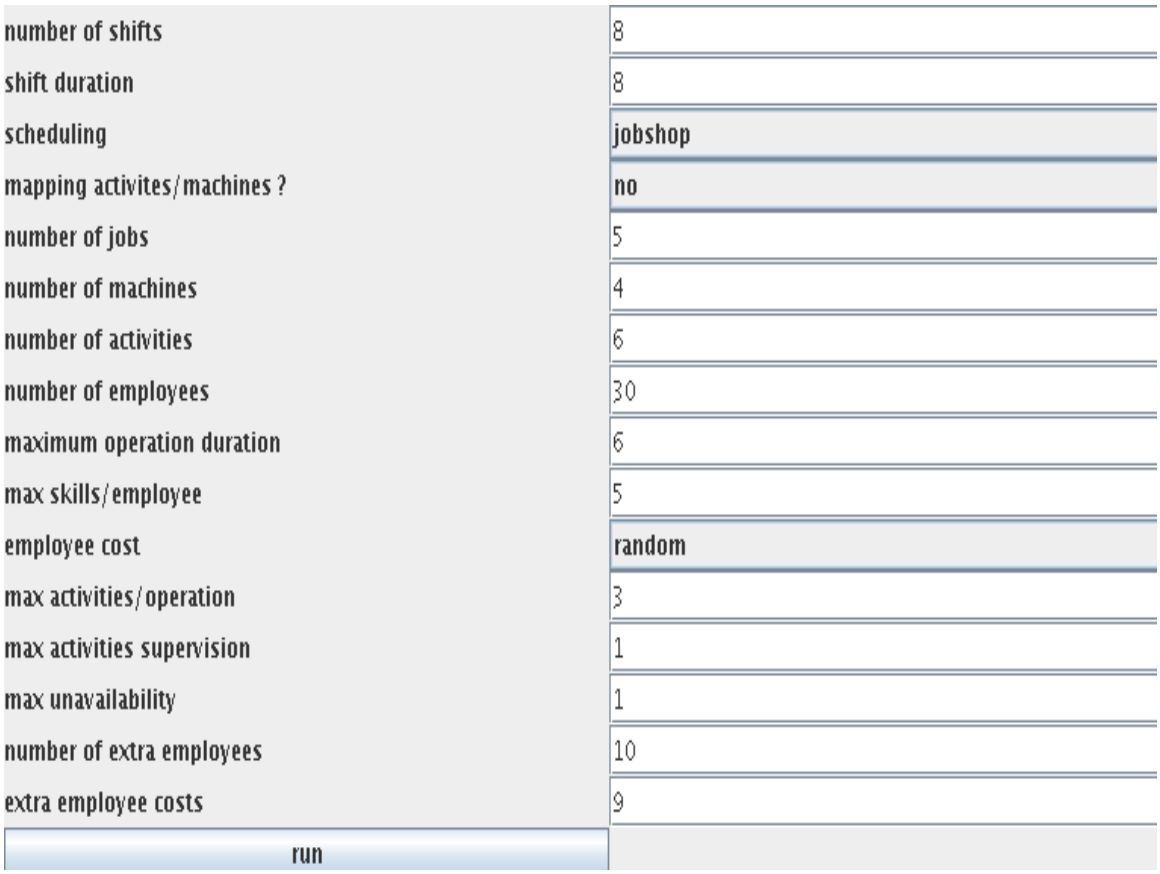

Figure 4: Instance generator applet

	Direct				Decomposition			
Inst	M/C	CPU(F)	C*	CPU(F)	M/C	CPU(F)	C*	CPU(F)
ejs4	40/27	0.1(3)	27(13)	TL	40/27	0.1/3	27(14)	TL
ejs9	49/44	0.2(4)	44(0)	TL	49/44	0.2(4)	44(0)	TL
ejs10	48/38	0.2(3)	38(0)	TL	48/38	0.2(3)	38(0)	TL
ejs8x8_1	44/135	0.7(65)	135(0)	TL	44/135	0.7(65)	135(0)	TL
ejs8x8_2	51/161	0.7 (6)	161(0)	TL	51/161	0.7(6)	161(0)	TL
ejs8x8_3	44/123	0.7(24)	123(0)	TL	44/123	0.7(24)	123(0)	TL
ejs10x10_1	80/182	1.8 (145)	182(0)	TL	80/182	1.6(145)	182(0)	TL
ejs10x10_3	84/150	7.4(2164)	150(0)	TL	84/150	12(5268)	150(0)	TL
ejs6x4x2x2_1	46/223	0.64 (18)	223(0)	TL	46/223	0.64(18)	223(0)	TL
ejs6x4x2x2_2	54/182	0.6(9)	184(0)	TL	54/182	0.6(9)	184(0)	TL
ejs6x4x2x2_3	42/240	0.7(16)	240(0)	TL	42/240	0.7(16)	240(0)	TL

Table 4: Results of pure CP methods

	Direct + `demandCover`				Decomposition + `demandIntervalCover`			
	no RC		with RC		no RC		with RC	
Inst	C*	CPU(F)	C*	CPU(F)	C*	CPU(F)	C*	CPU(F)
ejs4	23	2269(441635)	23	1.0(4)	25(23)	TL	23	2.0 (2587)
ejs9	44(0)	TL	24	135(6120)	24	194 (52328)	**24**	**50.4 (7770)**
ejs10	38(13)	TL	23	5.1(159)	38(16)	TL	23	27.1(22838)
ejs8x8_1	135(69)	TL	78	133(1225)	135(77)	TL	135 (77)	TL
ejs8x8_2	161(86)	TL	96	155(869)	161(92)	TL	161 (92)	TL
ejs8x8_3	123(67)	TL	83	52(203)	123(72)	TL	123(72)	TL
ejs10x10_1	182(92)	TL	124	2690(14176)	182(102)	TL	182(102)	TL
ejs10x10_3	150(76)	TL	150(15)	TL	150(15)	TL	150 (15)	TL
ejs6x4x2x2_1	223(16)	TL	132	50(902)	223(40)	TL	223(40)	TL
ejs6x4x2x2_2	184(0)	TL	74	225(2431)	184(22)	TL	184(22)	TL
ejs6x4x2x2_3	240(122)	TL	148	29(217)	240(133)	TL	240(133)	TL

Table 5: Results of CP with LP propagation on bound change of δ_{as}

	Direct + `demandCover`				Decomposition + `demandIntervalCover`			
	no RC		with RC		no RC		with RC	
ejs4	23	0.7(8)	**23**	**0.7(4)**	23	0.8(8)	23	0.8(4)
ejs9	44(0)	TL	44(0)	TL	24	61(11128)	24	121(7465)
ejs10	38(13)	TL	38(13)	TL	**23**	**2.6(219)**	23	3.8 (159)
ejs8x8_1	135(69)	TL	135(69)	TL	78	90(3310)	**78**	**81.2(1437)**
ejs8x8_2	123(86)	TL	104(86)	TL	**96**	**103(1679)**	96	107(848)
ejs8x8_3	83	40(1602)	83	64(736)	**83**	**29(227)**	83	34(204)
ejs10x10_1	182(92)	TL	182(92)	TL	**124**	**926(24125)**	124	1518(19224)
ejs10x10_3	**150(76)**	**TL**	150(0)	TL	150(15)	TL	150(15)	TL
ejs6x4x2x2_1	223(16)	TL	223(16)	TL	**132**	**38(1304)**	132	40.3(977)
ejs6x4x2x2_2	184(0)	TL	184(0)	TL	**74**	**157(2781)**	74	194(2398)
ejs6x4x2x2_3	181(122)	TL	181(122)	TL	**148**	**24(332)**	**148**	**24(224)**

Table 6: Results of CP with LP propagation on bound change of δ_{as} and on assignment of A_{es}

	Decomposition + `demandIntervalCover`			
	no RC		with RC	
Inst	C*	CPU(F)	C*	CPU(F)
ejs4	23	1.0(8)	23	1.0(4)
ejs9	24	79(9643)	24	141(6120)
ejs10	23	3.55(217)	23	5.08(159)
ejs8x8_1	78	156(3207)	78	134(1225)
ejs8x8_2	96	156(1679)	96	172(869)
ejs8x8_3	83	46(227)	83	53(203)
ejs10x10_1	124	1893(21314)	124	2832(14176)
ejs10x10_3	150(15)	1811(9202)	150(15)	1812(8203)
ejs6x4x2x2_1	132	50(1304)	132	50(902)
ejs6x4x2x2_2	74	173(2777)	74	225(2431)
ejs6x4x2x2_3	148	29(332)	148	29(217)

Table 7: Results of CP with propagation of LP constraint on δ_{as} bound change, A_{es} and I_{ij} assignment

	Decomposition + `demandIntervalCover`		
	no RC		
Inst	C^*	CPU(F)	loss
ejs4	23	0.95(8)	0%
ejs9	24	5.13(755)	0%
ejs10	28	2.3(127)	21%
ejs8x8_1	91	11.6(196)	16.6%
ejs8x8_2	110	159(2968)	14.6%
ejs8x8_3	98	11.07(69)	18%
ejs10x10_1	130	157(1902)	4.8%
ejs10x10_3	150(83)	1927(29840)	0%
ejs6x4x2x2_1	132	27.8(654)	0%
ejs6x4x2x2_2	101	7.3(71)	36.5%
ejs6x4x2x2_3	170	8.28(113)	14.8%

Table 8: Results of the sequential approach

	Decomposition + `demandIntervalCover`			
	no RC			
Inst	M	C^*	CPU(F)	gain
ejs4	40	23	0.95(8)	0%
ejs9	49	24	67(11128)	0%
ejs10	48	23	15(1589)	0%
ejs8x8_1	46	68(61)	3207(114975)	12.8%
ejs8x8_2	52	87(79)	2675(122407)	9.4%
ejs8x8_3	46	72	5027(141410)	13.3%
ejs10x10_1	80	182(19)	1801(30777)	-46%
ejs10x10_3	84	150(14)	1811(47248)	0%
ejs6x4x2x2_1	48	93	49(1010)	29.5%
ejs6x4x2x2_2	56	58	718(16134)	21.6%
ejs6x4x2x2_3	43	118	745(19228)	20.2%

Table 9: Results of the integrated approach for a makespan within 5% of the minimal makespan