Applied Mathematical Modelling 39 (2015) 955–964

Contents lists available at ScienceDirect

Applied Mathematical Modelling

journal homepage: www.elsevier.com/locate/apm

On the optimal modeling and evaluation of job shops with a total weighted tardiness objective: Constraint programming vs. mixed integer programming

Rashed Sahraeian, Mohammad Namakshenas *

Department of Industrial Engineering, College of Engineering, Shahed University, Tehran, Iran

ARTICLE INFO

Article history:
Received 9 May 2013
Received in revised form 3 July 2014
Accepted 21 July 2014
Available online 13 August 2014

Keywords:
Combinatorics
Constraint programming
Job shop
Mixed integer programming
Total weighted tardiness

ABSTRACT

In this study we consider the mapping of the main characteristics, i.e., the structural properties, of a classical job shop problem onto well-known combinatorial techniques, i.e., positional sets, disjunctive graphs, and linear orderings. We procedurally formulate three different models in terms of mixed integer programming (MIP) and constraint programming (CP) paradigms. We utilize the properties of positional sets and disjunctive graphs to construct tight MIP formulations in an efficient manner. In addition, the properties are retrieved by the polyhedral structures of the linear ordering and they are defined on a disjunctive graph to facilitate the formulation of the CP model and to reduce the number of dominant variables. The proposed models are solved and their computational performance levels are compared with well-known benchmarks in the job shop research area using IBM ILog Cplex software. We provide a more explicit analogy of the applicability of the proposed models based on parameters such as time efficiency, thereby producing strong bounds, as well as the expressive power of the modeling process. We also discuss the results to determine the best formulation, which is computationally efficient and structurally parsimonious with respect to different criteria.

1. Introduction

In general, obtaining an optimal solution for a scheduling problem is non-trivial. In this study, we handle the job shop scheduling problem from the viewpoint of a practitioner whose aim is to scrutinize a scheduling problem in a different combinatorial optimization framework. A practitioner with an inappropriate outlook would formulate a scheduling problem as a mixed integer programming (MIP) problem and use the default settings in commercially available software, instead of using problem-specific algorithms. Therefore, we initially focus on obtaining the optimal solution of the job shop scheduling problem with the total weighted tardiness objective (JS-TWT) using mathematically robust procedures such as MIP and constraint programming (CP).

It should be noted that few researchers have revisited the capacity of MIP given the many hardware and software advances that have been made recently, thus we consider these formulations in this context. In particular, the mathematical models used for single machine scheduling problems have received far less attention than heuristics and after nearly 50 years, the composition of the models introduced by Wagner [1], and Manne [2] still retain their original forms but they

* Corresponding author.
E-mail addresses: sahraeian@shahed.ac.ir (R. Sahraeian), m.namakshenas@shahed.ac.ir (M. Namakshenas).

http://dx.doi.org/10.1016/j.apm.2014.07.032
0307-904X/

have also entered neglect. This might be due partially to their low efficiency and/or the absence of high-tech processors, but there are few model development techniques based on these compositions. Nevertheless, the complexity of models depends on the number of jobs and machines, but also on the formulation framework.

The classical job shop problem has many practical applications in processes where a number of tasks need to be scheduled in a specific order. These applications range from production and manufacturing environments, e.g., a semiconductor factory, to service-based industries such as hospitals or transportation systems, e.g., railways. A classical job shop problem that is independent of any objective form is known to be an NP-complete problem [3]. Thus, before the development of an effective algorithm, the different *structural properties* of difficult scheduling problems such as JS-TWT should be addressed based only on the nuances of general-purpose models [4], e.g., mathematical models. This can be achieved by traditional combinatorics, but mapping the mathematical structures of these combinatorial techniques onto JS-TWT is a major challenge.

In the widely used three-field notation of Graham, JS-TWT is written as: $J_m \| \sum w_j T_j$ where $T_j = \max(0, c_j - d_j)$. The basic characteristics of a classical job shop are defined as follows. The processing times are known, fixed, and independent of the sequence. All jobs are ready to be processed at time zero. Recirculation is not allowed, i.e., all jobs should meet the machines only once. Preemption is not permitted, i.e., once an operation has started on a machine, it must be completed before another operation may be started on that machine. Only one job may be processed by a machine at any instant in time and each job should encounter all machines.

The first study to describe the formulation of scheduling problems with a mathematical notation was published in 1959. Early studies mostly addressed single machine scheduling as well as multi-machine scheduling. MIP formulations were developed and treated as powerful combinatorics to investigate the computational aspects and structural properties of scheduling problems. The first attempt to minimize the makespan with three machines in series, which is known as the flow shop problem, was reported by Wagner [1]. In this formulation, the key binary variables are x_{jk}; 1 if job j is placed in the kth position in the sequence. In previous studies, these variables are known as assignment variables or positioning variables. Another early study Manne [2] focused on the job shop problem with the aim of minimizing the makespan. In this formulation, the key binary variables are y_{ij}: 1 if job i precedes j in the schedule. These variables are also referred to as precedence variables and they are used in combination with big-M as a large constant to formulate precedence between jobs or operations (disjunctive constraints). In addition, Baker and Keller [5] used these variables to formulate a MIP model for single machine scheduling with the tardiness objective.

The number of variables and solution spaces are essential features of mathematical or MIP models. Thus, increasing the dimensions of variables or using an ill-defined solution space in the techniques designed to handle MIPs may lead to their collapse and the failure to obtain the optimum as an exponential function of time. In the last three decades, in order to alleviate this problem, computer scientists have developed a combinatorial framework with different variable data structures, graph theory, artificial inference, novel computer programming techniques, and branch and bound trees. This framework is called constraint programming (CP) and it is now used extensively for solving complex scheduling problems. In general, the main features of CP are: (1) the ability to handle heterogeneous constraints, multiple disjunctions, and non-convex solution spaces; (2) it is independent of the problem domain size; and (3) it allows the use of an optimization programming language (OPL) [6]. In general, CP solvers are designed according to two concepts: propagation and filtering (or consistency and backtracking). In the first step, the solver must logically eliminate some of the values in the feasibility set whereas those that are compatible with the model are retained. In the second step, the solver must then challenge its previously selected values and test other values [7]. However, existing CP solvers cannot provide a global view of the problem and they cannot exploit the lower bounds provided by linear relaxations, while in some cases, the artificial intelligence used in CP solvers fails to determine the optimal node within a reasonable amount of time. Thus, there is a trade-off between the careful selection of MIP and CP models during the formulation of different aspects of the problem.

In contrast to job shops with the makespan objective, the solution procedures for the JS-TWT are very limited. We consider that it is convenient to divide previous JS-TWT research into three categories: exact methods, heuristics, and theoretical methods. For example, the branch and bound algorithm developed by Singer and Pinedo [8] belongs to the first category. Heuristics are also divided into the local search approach [9,10] and the shifting bottleneck procedure [11]. Theoretical approaches such as dynamic programming algorithms [12], as well as the first category, may be more appropriate for obtaining deep insights into the structural properties and complexity of the JS-TWT.

The main goal of this study is to map the structural properties of the JS-TWT onto the well-known combinatorics, i.e., positional sets, disjunctive graphs, and linear orderings. The positional sets and the disjunctive graphs inherently possess key properties that precisely define the convex hull of the MIP models. The origin of CP can be traced back to graph theory, thus linear ordering properties are implemented in the present study to formulate a disjunctive graph in an efficient manner and to translate it into an appropriate CP model. The properties are retrieved by the polyhedral structure of the linear ordering and defined on a disjunctive graph, thereby reducing the number of dominant variables in the CP model. In this study, we also test the logic used by the CP solver when the processing times of different jobs sets have few variations and the schedule is tight. A simple study of these models was conducted previously by Namakshenas and Sahraeian [13]. However, we completely revise and tighten the MIP formulations presented in [13] by introducing new binary variables and different constraint sets.

The remainder of this paper is organized as follows. Section 2 procedurally discusses the process of generating MIP formulations based on the aforementioned properties. Section 3 presents the linear ordering properties and the translation of

the graph-based results into the standard OPL expressions. The following section presents a comparison and an analysis of the proposed formulations, and the final section provides a summary and our conclusions.

2. Mixed integer formulation

In this study, we use the following notations to develop the MIP formulations.

Parameters:

n	number of jobs (N: jobs set);
m	number of machines (M: machines set);
p_{jh}	processing time for job j on machine h;
w_j	weight of job j;
d_j	due date of job j;
r_{jlh}	$\begin{cases} 1 & \text{if the } l \text{ th operation of job } j \text{ requires machine;} \\ 0 & \text{o.w.} \end{cases}$

Decision variables:

c_j	completion time of job j;
s_{jh}	starting time of job j on machine h (a continuous non-negative variable);
Z_j	$w_j \max(0, c_j - d_j)$ or the tardiness of job j;
x_{jkh}	$\begin{cases} 1 & \text{if job } j \text{ is scheduled in position } k \text{ on machine } h; \\ 0 & \text{o.w.} \end{cases}$
y_{ijh}	$\begin{cases} 1 & \text{if job } j \text{ follows job } i \text{ on machine } h \text{ (not necessarily immediately);} \\ 0 & \text{o.w.} \end{cases}$

2.1. Disjunctive approach

The disjunctive formulation is one of the multi-purpose formulations that can be applied to many problem types, including single-case and shop problems. This formulation technique employs disjunctive constraints, which utilize big-M and binary variables to decide the best ordering of tasks at each disjunction. This technique was first introduced by Manne [2]. In general, the definition of the solution space is largely dependent on the size of the big-M parameter. The disjunctive (DJ) formulation is closely involved with the disjunctive graph in a job shop. However, the relationship between the disjunctions is implicit, which allows us to define the starting time s_{jh} in combination with a key binary variable, y_{ijh}. Therefore, the disjunctive constraints can be written as follows:

$$(s_{ih} - s_{jh}) \geqslant p_{jh} - M y_{ijh}, \quad \forall i, j \in N, \ \forall h \in M, \tag{2.1}$$

$$(s_{ih} - s_{jh}) \geqslant p_{jh} - M(1 - y_{ijh}), \quad \forall i, j \in N, \ \forall h \in M. \tag{2.2}$$

One of the constraints (2.1) or (2.2) must be relaxed when job i precedes job j and j precedes i on machine k. The operational precedence between the sub-tasks should also be satisfied. In particular, it is necessary to indicate that the processing of operation $l + 1$ for job j on machine h must be started after the completion of operation l. This feature can be characterized by considering the binary parameter r_{jlh}.

$$\sum_{h \in M} r_{jlh}(s_{jh} + p_{jh}) \leqslant \sum_{h \in M} r_{j,l+1,h} s_{jh}, \quad \forall j \in N, \ \forall l \in M - \{m\}. \tag{2.3}$$

The versatility of the binary parameter r_{jlh} should also be emphasized. It can be utilized to produce the starting times of the jobs on the machines, $\sum_{h \in M} r_{jlh} s_{jh}$, or the processing times, $\sum_{h \in M} r_{jlh} p_{jh}$. Therefore, the starting times of the jobs on the last operation should be expressed as $\sum_{h \in M} r_{jmh} s_{jh}$. It should be noted that the constraint set (2.5) is used to convert the max function in the tardiness term into a linear form.

$$\min \sum_{j \in N} Z_j, \tag{2.4}$$

$$\sum_{h \in M} w_j r_{jmh}(s_{jh} + p_{jh}) - w_j d_j \leqslant Z_j, \quad \forall j \in N. \tag{2.5}$$

2.2. Hybrid approach

A simple version of this formulation was presented in Namakshenas and Sahraeian [13]. In our formulation, we attempt to impose tighter constraints by introducing new variables. We constructed this alternative formulation by deploying both the positional variables and the precedence variables, i.e., a hybrid formulation (HB). By defining variable x_{jkh} to position the operations in the sequence, the assignment constraints can be written as follows.

$$\sum_{j\in N} x_{jkh} = 1, \quad \forall k \in N,\ \forall h \in M, \tag{2.6}$$

$$\sum_{k\in N} x_{jkh} = 1, \quad \forall j \in N,\ \forall h \in M. \tag{2.7}$$

In order to establish a consistent model using two sets of variables, we define variable v_{kh} to decide the starting time of the scheduled job in the kth position for processing on machine h. In addition, a set of constraints is needed to set delays according to the processing times of the previously positioned jobs between the starting times of the consecutive positions.

$$v_{kh} - v_{k-1,h} \geqslant \sum_{j\in N} x_{j,k-1,h} p_{jh}, \quad \forall k \in N - \{1\},\ \forall h \in M. \tag{2.8}$$

The operational precedence constraint set can also be written in terms of variable v_{kh} in combination with parameter r_{jlh}.

$$\sum_{k\in M} r_{j,l+1,h}\, v_{kh} - \sum_{k\in M} r_{jlh}\, v_{th} \geqslant \sum_{k\in M} r_{jlh}\, p_{jh}, \quad \forall (j,k,t) \in N,\ \forall l \in M - \{m\}. \tag{2.9}$$

The constraint set (2.9) indicates that there is no relationship between the decision variables v_{kh} and x_{jkh}, which causes a conflict between the constraints. To better illustrate this conflict, consider that job j is placed in position k. Hence, x_{jkh} takes a value of one, whereas variable v_{kh} has a non-negative value for job j' in position k, which has been reserved for job j. To avoid having two or more reserved jobs in a single position, one of the terms must be relaxed in expression (2.10).

$$\neg\left(1 - \sum_{k\in M} r_{jlh}\, x_{jth}\right) \quad \vee \quad \neg\left(1 - \sum_{k\in M} r_{j,l+1,h}\, x_{jkh}\right) \quad \Rightarrow \quad \text{relax constraint set (2.9)}. \tag{2.10}$$

Therefore, using big-M as a large constant and the expression above, the constraint set (2.9) can be modified into the constraint set (2.11).

$$M\left(2 - \sum_{k\in M} r_{jlh}\, x_{jth} - \sum_{k\in M} r_{j,l+1,h}\, x_{jkh}\right) + \left(\sum_{k\in M} r_{j,l+1,h}\, v_{kh} - \sum_{k\in M} r_{jlh}\, v_{th}\right) \geqslant \sum_{k\in M} r_{jlh}\, p_{jh}, \quad \forall (j,k,t) \in N,\ \forall l \in M - \{m\}. \tag{2.11}$$

Two steps are required to define the objective function. First, computing the completion time of job j, i.e., c_j and second, forming the expression $\sum w_j T_j$, where $T_j = \max(0, c_j - d_j)$. The starting time and the processing time of job j for the last operation can be defined as expressions $\sum_{h\in M} x_{jmh}\, v_{mh}$ and $\sum_{h\in M} x_{jmh}\, p_{jh}$, but the first term is nonlinear and this term casts the model as a non-convex model. Thus, the linear form of expression $\sum_{h\in M} x_{jmh}\, v_{mh}$ is written as follows by defining a non-negative variable γ_{jh}:

$$c_j = \sum_{h\in M} \gamma_{jh} + \sum_{h\in M} x_{jmh}\, p_{jh}, \quad \forall j \in N, \tag{2.12}$$

$$\gamma_{jh} = x_{jmh}\, v_{mh}, \quad \forall j \in N, \forall h \in M, \tag{2.13}$$

$$\gamma_{jh} \leqslant v_{mh}, \quad \forall j \in N, \forall h \in M, \tag{2.14}$$

$$\gamma_{jh} \leqslant M x_{jmh}, \quad \forall j \in N, \forall h \in M, \tag{2.15}$$

$$\gamma_{jh} \geqslant v_{mh} - M(1 - x_{jmh}), \quad \forall j \in N, \forall h \in M. \tag{2.16}$$

Then, the objective is written as follows:

$$\min \sum_{j\in N} w_j Z_j. \tag{2.17}$$

$$\text{Subject to:} \quad Z_j \geqslant c_j - d_j, \quad \forall j \in N. \tag{2.18}$$

3. Constraint programming formulation

3.1. Graph translation

A disjunctive graph-based model for the JS-TWT with variables $\overrightarrow{s_{ij}}$ (the starting time of job i on machine j on a digraph) is represented as follows:

$$\text{Objective} \quad \min \sum_{j=1}^{n} Z_j. \tag{3.1}$$

Constraints

$$w_j(\overrightarrow{s_{ij}} + p_{ij} - d_j) \leqslant Z_j, \quad \forall i \in N,\ \forall j \in M, \tag{3.2}$$

$$\overrightarrow{s_{ik}} - \overrightarrow{s_{ij}} \geqslant p_{ij}, \quad \forall i \in N,\ \forall (j,k) \in M,\ \textit{with}\ \langle i,j\rangle \to \langle i,k\rangle, \tag{3.3}$$

$$(\overrightarrow{s_{ih}} - \overrightarrow{s_{jh}}) \vee (\overrightarrow{s_{jh}} - \overrightarrow{s_{ih}}) \geqslant p_{(jh)\vee(ih)}, \quad \forall (i,j) \in N,\ \forall h \in M, \tag{3.4}$$

$$s_{ij}, Z_i \geqslant 0, \quad \forall i \in N,\ \forall j \in M. \tag{3.5}$$

This formulation is conceptually essential when building any graph-based model and each constraint, either conjunctive or disjunctive, can be translated into its own model-building syntax. The model building processes are highly interrelated in graph theory and CP, thus we utilize the linear ordering concept, which is one of the most important combinatorics in graph theory, in order to translate these constraints into an appropriate CP model with the least possible dominant variables and the highest computational efficiency (Fig. 1). In this formulation, the first set of constraints is considered to linearize the objective function. The second set ensures that operation (i,k) cannot start before the completion of operation (i,j). The third set of constraints considers the ordering of the operations of different jobs that have to be assigned to the same resource.

The concept of linear ordering appears in the context of combinatorial problems. Grotschel et al. [14] first utilized this approach to perform a cutting plane algorithm. In another study, Chrissof and Reinelt [15] claimed that this approach can be employed to solve proportionately large instances by inferring that a linear ordering approach yields good results when linear relaxation is performed on large instances, such that the amount of time required to solve the relaxation would be significantly small.

A linear ordering technique with n jobs can be treated as a problem on a complete digraph with n vertices. Suppose that λ_{ij} has a value of one when job i is processed before job j, but zero otherwise. For each pair of vertices, i and j, we want to pick exactly one of two arcs (i,j) or (j,i). Therefore, in any feasible solution, the constraint set $\lambda_{ij} + \lambda_{ji} = 1, \forall (i,j) \in N$, with $i \neq j$ should be imposed on the resulting digraph. Furthermore, there should be no directed cycles in the resulting directed subgraph. This type of acyclic digraph is called a tournament [16]. To avoid directed cycles, it is necessary to prevent the solutions of this form with three jobs $(i,j,k) : \lambda_{ij} = \lambda_{jk} = \lambda_{ki} = 1$. This can be satisfied by the inequality $\lambda_{ij} + \lambda_{jk} + \lambda_{ki} \leqslant 2, \forall (i,j,k) \in N$, with $\{i \neq j,\ j \neq k,\ k \neq i\}$.

To facilitate this procedure and to implement the polyhedral properties of the linear ordering, we consider Fig. 2 as a digraph that represents a job shop problem with three jobs and three machines. Fig. 2 indicates a feasible solution where each job meets each machine once (conjunctions) and each machine should process each decided job (disjunctions). In addition, O_{nm} denotes the operation of job n on machine m. Hence, for machine 1 and subgraph (O_{11}, O_{21}, O_{31}), constraint $\lambda_{12} + \lambda_{13} + \lambda_{23} \leqslant 2$ should be imposed; for machine 2 and subgraph (O_{32}, O_{12}, O_{22}), constraint $\lambda_{31} + \lambda_{32} + \lambda_{12} \leqslant 2$ should be imposed; and for machine 3 and subgraph (O_{33}, O_{23}, O_{13}), constraint $\lambda_{32} + \lambda_{31} + \lambda_{21} \leqslant 2$ should be imposed.

As mentioned above, to observe the precedence of jobs, constraint set $\lambda_{ij} + \lambda_{ji} = 1, \forall (i,j) \in \{1,2,3\}$, with $i \neq j$ must be satisfied. From Fig. 2 and by applying the linear ordering properties on the digraph, two subgraphs λ_{ij} and r_{ilk} can be extracted. The critical path of each job, which satisfies the decided order of the disjunctions and the conjunctions, can be obtained by determining the symmetric difference in these spanning subgraphs. We let $\mathbf{\Lambda}$ denote the incidence matrix of the disjunctive subgraphs, $\mathbf{R_I}$ is a two-dimensional matrix of the conjunctive subgraphs, and $\mathbf{P}$ is the processing time of the jobs on the machine subsets, and we have the following property.

$$\text{(on machine h)}: \quad \partial \mathbf{R_I} \Delta \partial \mathbf{\Lambda} := \partial(\mathbf{R_I} \Delta \mathbf{\Lambda}) \quad \Rightarrow \quad \overrightarrow{s_{jh}} := \max_j[\partial(\mathbf{R_I} \Delta \mathbf{\Lambda})\mathbf{P}]. \tag{3.6}$$

Expression (3.6) implies that the starting time of job j on machine h can be computed as whether we should maximize the multiplication of matrices $\mathbf{\Lambda}$ and $\mathbf{P}$, which only consider the disjunctive subgraphs, or calculating the symmetric difference in the spanning subgraphs on matrices $\mathbf{\Lambda}$ and $\mathbf{R_I}$. For example, we apply this property on subgraph (O_{11}, O_{21}, O_{31}) in Fig. 2.

$$\text{(on machine 1)}: \quad \mathbf{S} := \mathbf{P\Lambda} := (p_{11}, p_{21}, p_{31}).\begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \quad \Rightarrow \quad \mathbf{S} := (0, p_{11}, \overbrace{p_{11} + p_{21}}^{\text{last operation}}).$$

Then, we apply it on machine 2, $\mathbf{S} := (p_{32}, p_{12} + p_{32}, 0)$ and on machine 3, $\mathbf{S} := (p_{23} + p_{33}, p_{33}, 0)$. In this case, all of the starting times of the jobs in their final operations are greater than the other terms in the matrix $\mathbf{S}$. Therefore, it can easily be inferred that there is no need to check the evaluation function based on the resulting expression (3.6) over the complete subset of machines in a job shop. The philosophy that underlies the introduction of this idea is based on reducing the number of dominant variables in the objective function. Hence, expression (3.6) is taken as the starting time criterion of job j on machine h to develop the CP formulation.

3.2. Proposed CP model

In contrast to mathematical models, CP models are highly dependent on the CP packages employed because of the differences in the functional structures of various modeling languages. In the present study, we used ILOG OPL Studio 11.1 syntax

Fig. 1. Translation framework for building the CP model.

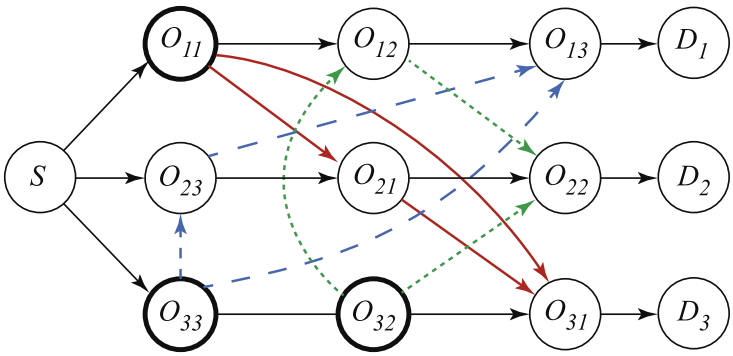

Fig. 2. Illustration of the tournament property of the linear ordering technique for a job shop digraph with three jobs and three machines.

as the modeling language. Table 1 shows the definitions of the syntax used in the OPL model. It should be noted that we completely revised the earlier version of the CP model presented in Namakshenas and Sahraeian [13] based on the properties retrieved by the polyhedral structure of the linear ordering.

A basic OPL modeling framework comprises a set of intervals (jobs) that needs to be assigned to a set of resources, e.g., machines or operators. An interval variable is a decision variable with three components, i.e., start, end, and duration, which are linked together logically. For the interval variables, the search methodology does not enumerate the values in the variables domain and the search space is usually independent of the problem domain size. The data structure of the interval variables is not pseudo-polynomially entangled with the domain of interval variables; therefore, the search space is estimated as the number of possible orderings of the n interval variables, and the complexity of the problem is estimated as $n.log2(n)$. Some properties of the global constraint $\lambda_{ij} + \lambda_{ji} = 1$ and the tournament properties of subgraphs are declared by the terms `precedes` and `requires`. From expression (3.6), which is defined based on the last operation, the objective of the proposed CP formulation is introduced as follows:

$$\min \quad \texttt{cum}\{w[j].\texttt{max}(\overbrace{\texttt{Itv}[\texttt{j},\texttt{m}]}^{\text{last operation}} - \texttt{d}[\texttt{j}],\texttt{0})\} \text{ in } \texttt{j} : \texttt{j} \text{ in } \texttt{r}[\texttt{j},\texttt{m},\texttt{h}]. \tag{3.7}$$

In this OPL formulation, we deploy two key variables, Itv and Opr, as interval variables, and Mac as resource variables or machines. The objective allows the solver to accumulate the sequences of the interval variables and to select the minimum value that corresponds to the combination. The local and global constraints are written as follows.

$$Itv[j,k] \quad \texttt{assigns} \quad Opr[j, nb\_resource].end, \quad \text{for all } (j \text{ in } N, k \text{ in } M), \tag{3.8}$$
$$Opr[j, nb\_resource] \quad \texttt{precedes} \quad Itv[j,k], \quad \text{for all } (j \text{ in } N, k \text{ in } M), \tag{3.9}$$
$$Opr[j,k] \quad \texttt{precedes} \quad Opr[j,k+1], \quad \text{for all } (j \text{ in } N, k \text{ in } M : k \leqslant m-1), \tag{3.10}$$
$$Opr[j,k] \quad \texttt{requires} \quad Mac[Opr[j,k],k], \quad \text{for all } (j \text{ in } N, k \text{ in } M), \tag{3.11}$$
$$Mac[Opr[j,k],k] \quad \texttt{waits} \quad Mac[Opr[j,k],k+1], \quad \text{for all } (j \text{ in } N, k \text{ in } M : k \leqslant m-1), \tag{3.12}$$

The first set of constraints (3.8) assigns the jobs based on their last operations in interval variable Itv from a dynamic resource pool. This concept is applied by defining one of the interval variables as optional, thus the solver can alternate the intervals when branching the nodes (for more information on alternating interval variables, see [17]). These auxiliary variables help the modeler to impose a global deadline constraint on the entire planning horizon. The constraint set (3.9) adjusts the precedences between the operations of the consecutive jobs, i.e., in their final operations, the jobs cannot bypass the interval Itv, and constraint set (3.10) indicates that all interval variables should be dedicated to a *unary* resource variable (Mac). Note that there is no *alternative* resource for interval variables, therefore they must be dedicated to only one source. The last set of constraints suggests that the resource assignments should not overlap.

Table 1
Syntax definitions for the declared OPL local or global constraints.

Syntax	Definition
`cum`	Accumulates over interval variables
`precedes`	Imposes a precedence constraint on interval variables
`requires`	Imposes resource requirements
`wait`	Imposes a precedence constraint on the assignment of resource variables

4. Test problems and experiments

To evaluate the computational performance of the proposed models, we considered that the information retrieved by the solutions of *all-embracing* instances might be useful in a global sense compared with other conditions. In this case, all-embracing instances denote any instances where jobs possess variations in their processing times and various degrees of due date satisfiability. In order to demonstrate this property and to obtain a fair assessment compared with previously reported methods, our benchmarks were derived from well-known test problems, e.g., ABZ05, ABZ06, and MT10. These benchmarks are employed mainly to assess the makespan objective, thus we revised them by adding a due date and a weight vector to each of the jobs according to the following scheme in Pinedo and Singer [11]: 40% of the tasks were given weights of 4 and 1, and the remainder had a weight of 2. Thus, a vector of weights $(4, 4, 2, 2, 2, 2, 2, 2, 1, 1)$ was assumed for an instance of 10 jobs. Pinedo and Singer [11] used Eq. (4.1) to generate the due dates of jobs according to tightness factor β. Thus, the due date of job i was set as equal to the floor of the sum of the processing times on all machines multiplied by the tightness factor.

$$d_i = \lfloor \beta \sum_{k=1}^{m} p_{ik} \rfloor, \quad \beta = 1.3, 1.5, 1.6. \tag{4.1}$$

4.1. Pre-analysis

Table 2 shows the sizes of the MIP and CP formulations generated in terms of the numbers of binary and integer variables, and the constraints for all three alternatives. Among the MIP alternatives, the HB formulation obtained the highest number of variables and constraints. The DJ formulation employed a moderate number of constraints compared with the HB, which appeared to explain its efficiency in solving larger problems. However, the first set of constraints, i.e., the disjunctive constraints, included a larger proportion of the constraints set in the DJ model. This might be a disadvantage because this type of constraint cannot define the solution space in a tight manner. A simple option may be keeping the value of big-M as low as possible. As mentioned earlier, in order to bypass this "loose" behavior, we proposed a hybrid model where the first set of constraints, the assignment constraint set (2.6) and (2.7), was replaced by the disjunctive counterparts in the DJ model. These hard constraints are assumed to define a tighter solution space, but this approach should be assessed based on the computational efficiency to justify whether this property holds.

Table 2 shows clearly that the number of binary variables in the mathematical formulations tended to grow exponentially with the problem size, thereby demonstrating the NP-hard complexity. The data structure of the CP model was not the same as that of the MIP models, thus it cannot be compared explicitly with the ordinary variables developed in the MIP models. Nevertheless, we felt that the simple correspondence between the problem size and the interval variables might be useful information.

Detailed information about the tuning of the CP package is shown in Table 3. The configuration of the CP package did not employ the default settings and we selected the best options based on the following scheme. The "Restart" option was selected as "search strategy" according to Regin [18]. The "long tail phenomenon" caused by "depth-first search" during the ordering of the variables was avoided by randomizing the search, which restarted when the solver began to backtrack [18]. In addition, the high degree of dependency on the "inference level" with the CP option was counter-productive in many cases [17]. The test problems were encoded by the IBM Ilog Cplex 11.1 solver and run on a single PC with a 2.66 GHz Intel Core 2 Quad processor (4 MB cache) and 6 GB of memory. The solver was restricted to a 3600 s time limit, which terminated a specific run if the optimal schedule had not been verified.

4.2. Computational results and post-analysis

Information about the experimental runs is summarized in Tables 4–6 for cases where β = 1.3, 1.5, 1.6. A set of 30 benchmarks was tested for each formulation with different tightness factors for a total of 270 runs. However, measuring the com-

Table 2
Model comparison based on the number of variables generated and the constraints.

Problem size (job × machine)	Variables									Constraints		
	Binary			Integer or interval			Non-negative					
	DJ	HB	CP	DJ	HB	CP	DJ	HB	CP	DJ	HB	CP
4 × 4	24	64	0	0	0	20	21	56	0	64	318	16
6 × 6	90	216	0	0	0	36	43	125	0	216	1374	36
8 × 8	224	512	0	0	0	72	73	197	0	512	4088	64
10 × 10	450	1000	0	0	0	110	110	292	0	1000	9790	100
12 × 12	792	1728	0	0	0	156	157	435	0	1728	20148	144
15 × 15	1575	3375	0	0	0	240	241	678	0	3375	49035	225

Table 3
Detailed information about the CP solver configuration.

Parameter	Setting
Inference level	Medium
Search type	Restart
Restart fail limit	30
Restart growth factor	1.3

Table 4
Summary of the computational results; $\beta = 1.3$.

Model	Problem solved	Average time (s)	Maximum time (s)	Average gap (%)	Maximum gap (%)
DJ	27	985.13	3600+	11.13	30.00
HB	11	3600+	3600+	46.25	100.00
CP	30	748.02	1985.22	0.00	0.00

putational performance of the dedicated models required time-based and gap measurement criteria rather than the convergence rate, or the near-optimal analyses used to investigate heuristics. We focused on five important parameters to highlight various differences with these exact models.

The first criterion represented the number of problems solved within the time limit. Therefore, the overall average and maximum gaps were reported if the number did not reach 30 problems and the optimal schedule could not be found in at least one case. Note that a gap was the difference between the solution returned by a truncated run and the optimum, which was computed as the ratio relative to the optimum. The gaps were also useful for determining the convergence to the optimal schedule by the proposed models. Other important criteria reported in the tables are the average and maximum times (in seconds) for running all 30 benchmarks of a specific case. This was essential for investigating the algorithmic speed-up obtained with each model.

Overall, the results suggest that the HB formulation is very weak and inefficient compared with the other methods. Thus, we tested the hypothesis proposed in the previous section: the set of constraints used to define the solution space is unjustifiable compared with other formulations. Clearly, the CP formulation solved all cases where $\beta = 1.3$, but its computational speed-up declined compared with DJ in terms of the tightness factor. The DJ formulation converged to the optimal schedule almost as fast as the CP when $\beta = 1.3$, but in three cases, it was highly exhaustive with the branch and bound procedure. Nevertheless, the DJ model was the most efficient when $\beta = 1.5, 1.6$. As shown in the tables, the tightness factor had a dramatic effect on the average and the maximum times required to obtain the best schedules, which may be explained by its effect on the due dates. The declining effect of the tightness factor influenced the CP model much more than the DJ model because the growth in the size of the due dates bucket greatly affected the decisions made based on the inherent structures of the interval variables. Clearly, the CP model obtained the desired results in the cases with the tightness factor, thereby supporting its theoretical basis. Hence, a standard CP formulation concept was not sufficient to improve the runtime behavior in a consistent manner. Almost 20% of the cases could be solved using a specific CP procedure. This tightness can be illustrated graphically using a Gantt chart, such as Fig. 3. For example, case ABZ05 appears to be a moderately difficult example in terms of the schedule tightness or the gaps between the scheduled operations.

Overall, these observations suggest that the proposed CP model outperformed the MIP models when there were more disjunctive decision nodes. The complexity of the JS-TWT is pseudo-polynomially locked on the processing times for jobs

Table 5
Summary of the computational results; $\beta = 1.5$.

Model	Problem solved	Average time (s)	Maximum time (s)	Average gap (%)	Maximum gap (%)
DJ	25	1805.78	3600+	18.21	18.00
HB	5	3600+	3600+	46.25	100.00
CP	22	2614.48	3600+	26.54	0.00

Table 6
Summary of the computational results; $\beta = 1.6$.

Model	Problem solved	Average time (s)	Maximum time (s)	Average gap (%)	Maximum gap (%)
DJ	19	2771.36	3600+	21.00	100.00
HB	0	3600+	3600+	100.00	100.00
CP	8	2914.65	3600+	78.00	100.00

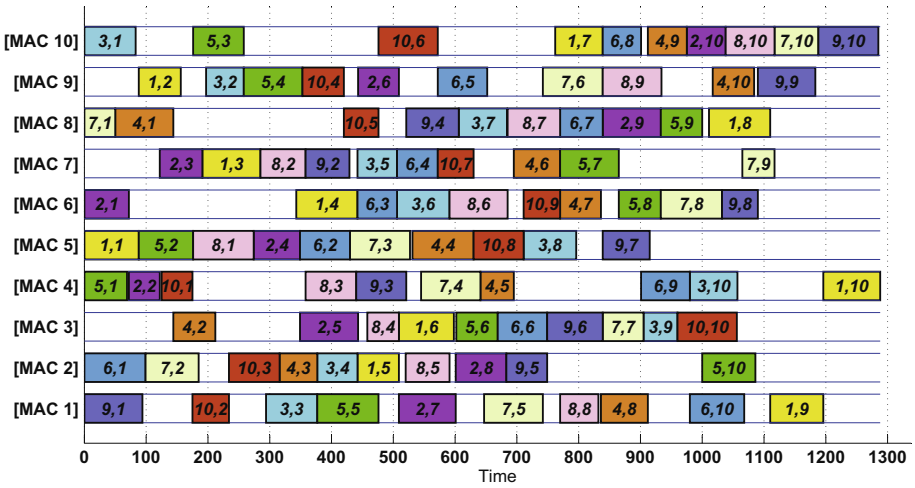

Fig. 3. Gantt chart for ABZ05 and $\beta = 1.5$ with the optimal schedule, which was obtained using the proposed CP model (the first and second indices indicate the job and the operation number, respectively).

[17], thus the CP model is convenient for handling larger problems in terms of the problem domain, such as the processing time (the data structures of CPs are independent of the size of the interval). The expressive power of CP solvers is also of crucial importance because they allow the declaration of global and meta constraints, thereby facilitating the model-building process. However, the inference techniques provided in recent high-tech CP solvers are not reliable for obtaining the optimal schedule in a reasonable amount of time. They also lack algebraic techniques in favor of general-purpose MIP solution methods such as linear relaxation. Thus, the linear relaxation of the HB model provides a stringent and strong bound, although it is inherently inferior to other models. The DJ model also provides a rapid test of the complexity of a specific case in terms of the binary variables and/or the explicit manipulation of the constraints due to their *smooth* nature.

5. Conclusion

Evidently, the implementation of the methods considered in this study remains the crucial step during the problem-solving process. However, the richness of these models and their associated expressive powers are also highly significant. The fact that a scheduling problem can be formulated via mathematical programming should not imply that there is a satisfactory off-the-shelf standard solution procedure. Again, we emphasize that the JS-TWT is still a very hard problem using enumeration methods or heuristics.

In this study, we conducted computational experiments based on two mathematical models and a CP model, which we programmed using a state-of-the-art software package, Cplex 11.1. We tested the efficiency of the proposed formulations with well-known job shop benchmarks, each of which employed different tightness criteria. In the first set, all cases were solved by the CP model within the time limit of 3600 s. The tightness factor and its effect on the due dates deteriorated the computational efficiency of the CP model, but it also provided insights into the merits of the DJ formulation. The most important feature of the CP solution procedure is that its data structure was developed to be pseudo-polynomially independent of the problem domain size. We also found that the formulation based on disjunctive graphs, the DJ model, was highly consistent with our expectations, where it produced the lowest number of variables and constraints compared with the HB formulation and it yielded a tight model. We suggest that modelers should be aware of the capabilities of MIP and CP given the many recent hardware and software advances. For example, the usage of big-M as a large constant should be avoided for two reasons: the solver might require an unacceptable amount of time to prune the branches, i.e., *software failure*, and it might require excessive amounts of memory, i.e., *hardware failure*.

In general, we only recommend using the proposed MIP and CP models with low or moderate numbers of disjunctive nodes. Specialized algorithms may be preferable only for problems with a larger number of disjunctive nodes and longer processing times. Furthermore, a deeper consideration of the CP formulation might be justified if it provides insights into the combinatorial characteristics of the JS-TWT. Finally, we suggest the combination of other graph-based combinatorics to construct models with fewer variables in future research.

References

[1] M.H. Wagner, An integer programming model for machine scheduling, Naval Res. Logist. Q. 6 (2) (1959) 131–140.
[2] S.A. Manne, On the job shop scheduling problem, Oper. Res. 8 (2) (1960) 219–223.
[3] M.L. Pinedo, Scheduling: Theory, Algorithms, and Systems, 3rd ed., Springer, New York, USA, 2008.

[4] A. Jouglet, J. Carlier, Dominance rules in combinatorial optimization problems, Eur. J. Oper. Res. 212 (3) (2011) 433–444.
[5] K.R. Baker, B. Keller, Solving the single-machine sequencing problem using integer programming, Comput. Ind. Eng. 59 (4) (2010) 730–735.
[6] A. Lopez-Ortiz, C.-G. Quimper, J. Tromp, P.V. Beek, A fast and simple algorithm for bounds consistency of the all different constraint, in: Proceedings of the 18th International Joint Conference on Artificial Intelligence, vol. 32, 2003, pp. 245–250.
[7] K.R. Apt, M. Wallace, Constraint Logic Programming Using ECL^{i}PS^e, Cambridge University Press, New York, 2007.
[8] M. Singer, M. Pinedo, A computational study of branch and bound techniques for minimizing the total weighted tardiness in job shops, IIE Trans. 30 (2) (1997) 109–118.
[9] I. Essafi, Y. Mati, S. Dauz ére-Pérès, A genetic local search algorithm for minimizing total weighted tardiness in the job shop scheduling problem, Comput. Oper. Res. 35 (8) (2008) 2599–2616.
[10] Y. Mati, S. Dauz ére-Pérès, C. Lahlou, A general approach for optimizing regular criteria in the job-shop scheduling problem, Eur. J. Oper. Res. 212 (1) (2011) 33–42.
[11] M. Pinedo, M. Singer, A shifting bottleneck heuristic for minimizing the total weighted tardiness in a job shop, Naval Res. Logist. 46 (1) (1999) 1–17.
[12] J.A. Gromicho, J.J. van Hoorn, F.S. da Gama, G.T. Timmer, Solving the job-shop scheduling problem optimally by dynamic programming, Comput. Oper. Res. 39 (12) (2012) 2968–2977.
[13] M. Namakshenas, R. Sahraeian, Toward various exact modeling the job shop scheduling problem for minimizing total weighted tardiness, in: 7th International Conference on Industrial Engineering and Industrial Management, vol. XVII, Congreso de Ingeniera de Organizacion, 2013, pp. 235–244.
[14] M. Grotschel, M. Junger, G. Reinelt, A cutting plane algorithm for the linear ordering problem, Oper. Res. 32 (6) (1984) 1195–1220.
[15] T. Chrissof, G. Reinelt, Low-dimensional linear ordering polytopes, Technical report, IWR Heidelberg, Germany, 1997.
[16] M. Junger, Polyhedral Combinatorics and the Acyclic Subdigraph Problem, Heldermann, Berlin, 1985.
[17] IBM, Detailed Scheduling in IBM ILOG CPLEX Optimization Studio with IBM ILOG CPLEX CP Optimizer, <http://cpoptimizer.ilog.com>, accessed: 2012-10-27, 2012.
[18] J.-C. Regin, Solving Problems with CP: Four Common Pitfalls to Avoid, in: J. Lee (Ed.), 17th International Conference on Principles and Practice of Constraint Programming, vol. 6876, Springer, 2011, pp. 3–11.