Mixed-integer programming models for flowshop scheduling problems minimizing the total earliness and tardiness*

Débora P. Ronconi † Ernesto G. Birgin ‡

April 29, 2010

Abstract

Scheduling problems involving both earliness and tardiness costs have received significant attention in recent years. This type of problem became important with the advent of the just-in-time (JIT) concept, where early or tardy deliveries are highly discouraged. In this work we examine the flowshop scheduling problem with no storage constraints and with blocking in-process. In this latter environment, there are no buffers between successive machines; therefore intermediate queues of jobs waiting in the system for their next operations are not allowed. Performance is measured by the minimization of the sum of earliness and tardiness of the jobs. Mixed-integer models that represent these scheduling flowshop problems are presented. The models are evaluated and compared in several problems using commercial known software.

Key words: Flowshop scheduling, earliness and tardiness, blocking in-process, mixed-integer programming formulations.

1 Introduction

This paper addresses the flowshop scheduling problem. This environment is characterized by n jobs being processed on m machines always in the same order, that is, the k-th operation of every job must be conducted on machine k. We will consider the case with the same job sequence in all machines, known as permutation schedule. As pointed out in [7], permutation schedules do not always include the optimal schedule for a given problem, but their importance should not be underestimated, since only permutation schedules are feasible in most real-life situations. A static and deterministic environment is assumed here, where the processing times and due dates are known and all jobs are available for processing since the beginning. Preemptions are not allowed, that is, when a job starts to be processed on a machine, it cannot be interrupted. Among its several applications, one of the most relevant applications of flowshop can be found in the chemical industry [3, 13]. Since specific equipment for chemical processes is expensive, there is a strong motivation to optimize this production environment.

*This work was supported by PRONEX-CNPq/FAPERJ (E-26/171.1510/2006-APQ1), FAPESP (Grants 2006/53768-0, 2006/03496-3 and 2009/10241-0), CNPq (308000/2009-9 and 304484/2007-5).

†Department of Production Engineering, EP-USP, University of São Paulo, Av. Prof. Almeida Prado, 128, Cidade Universitária, 05508-900, São Paulo SP, Brazil. e-mail: dronconi@usp.br

‡Department of Computer Science, IME-USP, University of São Paulo, Rua do Matão, 1010, Cidade Universitária, 05508-090 São Paulo, SP - Brazil. e-mail: egbirgin@ime.usp.br

Two different storage conditions are considered in this work: (a) unlimited buffer and (b) blocking in-process. In the flowshop scheduling problem with blocking there is no buffer storage between machines, and then queues of jobs waiting in the system for their next operation are not allowed. A job completed on one machine blocks it until the next machine is available for processing. Note that this environment is different from the no-wait flowshop environment. In the latter, there is no machine blocking: once a job is started on the first machine, it must be continuously processed (without interruption) until its completion on the last machine. As mentioned in [5], blocking can be related to the production process itself. Some examples of blocking can be found in concrete block manufacturing, which does not allow stock in some stages of the manufacturing process [4]. Another example appears in a robotic cell, where a job may block a machine while waiting for the robot to pick it up and move it to the next stage [16].

The considered objective function is the minimization of the total deviation of job completion times in relation to their corresponding due dates. Meeting due dates is a common objective for many manufacturing processes. Tardy jobs may generate contractual penalties and loss of credibility, causing damages to the company's image and loss of clients [15]. Early jobs were discouraged since the advent of Just-in-Time approaches due to the costs generated by those jobs, such as tied-up capital and inventory costs [1].

Mixed-integer linear programming (MILP) models, can be used to generate optimal schedules for moderate size problems. With the evolution of computers, in addition to the development of specific software to solve this kind of problem, research in this field has been greatly leveraged. See, for example, [11, 17, 18, 22]. The main goal of this paper is to evaluate, in terms of computational cost, mixed-integer linear programming formulations for the job scheduling problem in the flowshop environment with unlimited and zero buffer, for the minimization of total earliness and tardiness. The rest of this work is organized as follows. In Section 2, models for the unlimited buffer environment are presented. Models for the blocking in-process situation are introduced in Section 3. Section 4 is devoted to the numerical evaluation of the different formulations. Concluding remarks are given in Section 5.

2 MILP models for flowshop with unlimited buffer

The production environment with unlimited buffer is characterized by the existence of intermediate buffers between all machines with no constraint in terms of their capacity. At first, we introduce three formulations to be analyzed for this environment. According to [11], the flowshop environment with unlimited buffer minimizing the makespan is better represented by Wagner's [20], Wilson's [21], and Manne's [10] formulations. Therefore, these first three introduced models are based on them. Note that, differently from the problem considered in [10, 20, 21], in the present work, the insertion of idle time in the schedule can be advantageous. See [6] for a literature review about scheduling with inserted idle time.

The problem parameters for all the formulations studied in the rest of this work are (a) n: the number of jobs, (b) m: the number of machines, (c) p_{ik}: the processing time of job i on machine k, and (d) d_i: the due date of job i.

The model for the minimization of total earliness and tardiness based on the Wagner's formulation [20] with some modifications suggested in [17], called MUB1 from now on, is presented below. Variables of the model, with their corresponding bound constraints, are: $x_{ij} \in \{0,1\}, i = 1,\ldots,n, j = 1,\ldots,n$, $E_j \geq 0, j = 1,\ldots,n$, $T_j \geq 0, j = 1,\ldots,n$, $W_{jk} \geq 0, j = 1,\ldots,n, k =$

$1, \ldots, m-1$, $I_{jk} \geq 0, j = 1, \ldots, n-1, k = 1, \ldots, m$, and $C_{jm}, j = 1, \ldots, n$. The meanings of the variables are: x_{ij} is equal to 1 if job i is in j-th position of the sequence, 0 otherwise; T_j is the tardiness of the j-th job, E_j is the earliness of the j-th job, C_{jm} is the completion time of the j-th job on machine m, I_{jk} is the idle time between the j-th and the $(j+1)$-th jobs on machine k, and W_{jk} is the waiting time of the j-th job in the buffer between machines k and $k+1$.

$$\text{Minimize } \sum_{j=1}^{n} E_j + T_j \tag{1}$$

$$\text{subject to } T_j \geq C_{jm} - \sum_{i=1}^{n} x_{ij} d_i, \qquad j = 1, \ldots, n, \tag{2}$$

$$E_j \geq \sum_{i=1}^{n} x_{ij} d_i - C_{jm}, \qquad j = 1, \ldots, n, \tag{3}$$

$$C_{1m} = \sum_{k=1}^{n-1} (\sum_{i=1}^{n} x_{i1} p_{ik} + W_{1k}) + \sum_{i=1}^{n} x_{i1} p_{im}, \tag{4}$$

$$C_{jm} = C_{j-1,m} + I_{j-1,m} + \sum_{i=1}^{n} x_{ij} p_{im}, \qquad j = 2, \ldots, m, \tag{5}$$

$$I_{jk} + \sum_{i=1}^{n} x_{i,j+1} p_{ik} + W_{j+1,k} = W_{jk} + \sum_{i=1}^{n} x_{ij} p_{i,k+1} + I_{j,k+1}, \qquad j = 1, \ldots, n-1, \quad k = 1, \ldots, m-1, \tag{6}$$

$$\sum_{i=1}^{n} x_{ij} = 1, \qquad j = 1, \ldots, n, \tag{7}$$

$$\sum_{j=1}^{n} x_{ij} = 1, \qquad i = 1, \ldots, n. \tag{8}$$

Constraint (2), in conjunction with the non-negativity constraint $T_j \geq 0$, correspond to the linearization of $T_j = \max\{C_{jm} - \sum_{i=1}^{n} x_{ij} d_i, 0\}$ and provide us with the value of the individual tardiness of each job. Constraint (3), in conjunction with the non-negativity constraint $E_j \geq 0$, play the analogous role related to the individual earliness of each job. Constraints (4) and (5) are involved in obtaining the completion times of each job on machine m. Constraint (4) applies only to the job ordered in the first position, whose completion time depends on its own processing times and the waiting time between the machines. On the other hand, constraint (5) is general, based on the sum of the completion time of the previous job, the idle time of the last machine between two consecutive jobs and the processing time of the job. Constraints (7) and (8) ensure that a job can only be allocated to a sequence position and that each sequence position can only be related to one job. Constraint (6) expresses the relationship among several times, such as processing, waiting and idle times, among machines and jobs. Figure 1, based on Gantt chart, illustrates these relationships.

The following formulation, based on Wilson's model [21] and called MUB2 from now on, uses neither variables of idle time nor waiting time, but it introduces the variables of processing start time. Variables of the model, with their corresponding bound constraints, are: $x_{ij} \in \{0, 1\}, i =$

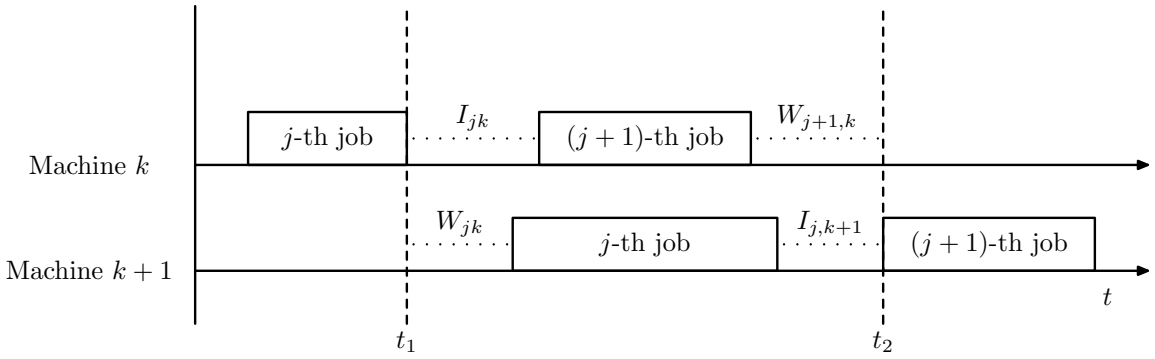

Figure 1: Graphical representation of the MUB1 model constraint (6) that expresses the relationship among processing, waiting and idle times.

$1, \ldots, n, j = 1, \ldots, n$, $E_j \geq 0, j = 1, \ldots, n$, $T_j \geq 0, j = 1, \ldots, n$, and $S_{jk}, j = 1, \ldots, n, k = 1, \ldots, m$. $C_{jm}, j = 1, \ldots, n$, defined in (12), representing the completion time of each job on the last machine, are not variables but intermediate values used to simplify the earliness and tardiness expressions in (10) and (11). Variables S_{jk} represent the starting time of the j-th job on machine k. The other variables have the previously defined meaning.

$$\text{Minimize} \sum_{j=1}^{n} E_j + T_j \tag{9}$$

$$\text{subject to } T_j \geq C_{jm} - \sum_{i=1}^{n} x_{ij} d_i, \qquad j = 1, \ldots, n, \tag{10}$$

$$E_j \geq \sum_{i=1}^{n} x_{ij} d_i - C_{jm}, \qquad j = 1, \ldots, n, \tag{11}$$

$$C_{jm} = S_{jm} + \sum_{i=1}^{n} x_{ij} p_{im}, \qquad j = 1, \ldots, m, \tag{12}$$

$$S_{j+1,k} \geq S_{jk} + \sum_{i=1}^{n} x_{ij} p_{ik}, \qquad j = 1, \ldots, n-1, k = 1, \ldots, m, \tag{13}$$

$$S_{j,k+1} \geq S_{jk} + \sum_{i=1}^{n} x_{ij} p_{ik}, \qquad j = 1, \ldots, n, k = 1, \ldots, m-1, \tag{14}$$

$$S_{11} \geq 0, \tag{15}$$

$$\sum_{i=1}^{n} x_{ij} = 1, \qquad j = 1, \ldots, n, \tag{16}$$

$$\sum_{j=1}^{n} x_{ij} = 1, \qquad i = 1, \ldots, n. \tag{17}$$

Constraints (13–15) impose the rules for the starting time of each job on each machine; while, as mentioned above, equality (12) gives the completion time of each job on the last machine as a

function of its start time and its processing time. Constraint (13) says that between the starting times of consecutive jobs on a machine, there must be enough time for the first (of the two jobs) to be processed. Constraint (14) indicates that between the starting times of a job on two consecutive machines, there must enough time for the job to be processed on the first machine. Constraint (15) simply says that the starting time of the first job on the first machine must be non-negative, i.e., the insertion of idle time at the beginning of the schedule is allowed. The remaining constraints were already explained before.

The next formulation, based on Manne's model [10] and called MUB3 from now on, uses a different definition of decision binary variables, based on whether a job occurs before others: z_{ij} is equal to 1 if job i precedes job j in the sequence (not necessarily immediately before it), 0 otherwise. In addition, the completion time C_{jk} now refers to job j and not to the j-th job in the sequence as defined earlier. Constant M, used in the formulation, represents a very large positive number. The formulation is presented below. Variables of the model, with their corresponding bound constraints, are: $z_{ij} \in \{0,1\}, i = 1,\ldots,n-1, j = i+1,\ldots,n$, $E_i \geq 0, i = 1,\ldots,n$, $T_i \geq 0, i = 1,\ldots,n$, and $C_{ik}, i = 1,\ldots,n, k = 1,\ldots,m$.

$$\text{Minimize } \sum_{i=1}^{n} E_i + T_i \tag{18}$$

$$\text{subject to } T_i \geq C_{im} - d_i, \qquad i = 1,\ldots,n, \tag{19}$$

$$E_i \geq d_i - C_{im}, \qquad i = 1,\ldots,n, \tag{20}$$

$$C_{i1} \geq p_{i1}, \qquad i = 1,\ldots,n, \tag{21}$$

$$C_{ik} \geq p_{ik} + C_{i,k-1}, \qquad i = 1,\ldots,n, k = 2,\ldots,m, \tag{22}$$

$$C_{ik} \geq p_{ik} + C_{jk} - Mz_{ij}, \qquad i = 1,\ldots,n-1, j = i+1,\ldots,n, k = 1,\ldots,m, \tag{23}$$

$$C_{jk} \geq p_{jk} + C_{ik} - M(1 - z_{ij}), \qquad i = 1,\ldots,n-1, j = i+1,\ldots,n, k = 1,\ldots,m. \tag{24}$$

Constraints (21) and (22) deal with completion times of individual jobs. Constraint (21) indicates that the completion time of a job on the first machine must be greater than or equal to the processing time of the job on the first machine. Constraint (22) says that between the completion time of a job on two consecutive machines there must be enough time to process the job on the first of the two machines. Constraints (23) and (24) ensure that only one operation is processed on each machine at any given time and that some order must exist between different jobs on the same machine. The remaining constraints were already explained before.

Finally, we elaborate on an improvement to Manne's model proposed by Liao [9]. Constraints (23) and (24) can be re-written as

$$C_{ik} - C_{jk} - p_{ik} + Mz_{ij} \geq 0, \tag{25}$$

$$-C_{ik} + C_{jk} + p_{ik} - Mz_{ij} \geq p_{jk} + p_{ik} - M, \tag{26}$$

respectively. Defining

$$q_{ijk} = C_{ik} - C_{jk} - p_{ik} + Mz_{ij},$$

inequalities (25) and (26) reduces to

$$0 \leq q_{ijk} \leq M - p_{ik} - p_{jk}. \tag{27}$$

Note that inequalities in (27) are in fact a lower and upper bound constraint for q_{ijk}. Thus, to solve subproblems of the branch and bound tree, the bounded simplex method can be used and achieving an optimal solution may be easier. It is worth noticing that with these new relationships, there is a reduction of one constraint for each pair of jobs in each machine, but it leads to an increase in variables q_{ijk} in the same amount.

Model MUB4 below incorporates the Liao's suggestion into model MUB3. Variables of the model, with their corresponding bound constraints, are: $z_{ij} \in \{0,1\}, i = 1,\ldots,n-1, j = i+1,\ldots,n$, $E_i \geq 0, i = 1,\ldots,n$, $T_i \geq 0, i = 1,\ldots,n$, $C_{ik}, i = 1,\ldots,n, k = 1,\ldots,m$, and $0 \leq q_{ijk} \leq M - p_{ik} - p_{jk}, i = 1,\ldots,n-1, j = i+1,\ldots,n, k = 1,\ldots,m$.

$$\text{Minimize } \sum_{i=1}^{n} E_i + T_i \tag{28}$$

$$\text{subject to } T_i \geq C_{im} - d_i, \qquad i = 1,\ldots,n, \tag{29}$$

$$E_i \geq d_i - C_{im}, \qquad i = 1,\ldots,n, \tag{30}$$

$$C_{i1} \geq p_{i1}, \qquad i = 1,\ldots,n, \tag{31}$$

$$C_{ik} \geq p_{ik} + C_{i,k-1}, \qquad i = 1,\ldots,n, k = 2,\ldots,m, \tag{32}$$

$$q_{ijk} = C_{ik} - C_{jk} - p_{ik} + Mz_{ij}, \qquad i = 1,\ldots,n-1, j = i+1,\ldots,n, \; k = 1,\ldots,m. \tag{33}$$

Summing up the characteristics of the four models presented in this section for the flowshop scheduling problem with unlimited buffer and minimizing earliness and tardiness, Table 1 shows the sizes of each model, expressed by their number of constraints, binary and continuous variables.

Model	binary variables	continuous variables	constraints
MUB1	n^2	$nm + 2n - m$	$nm + 3n - 1$
MUB2	n^2	$nm + 3n$	$2nm + 3n - m$
MUB3	$n(n-1)/2$	$nm + 2n$	$n^2m + 2n$
MUB4	$n(n-1)/2$	$n^2m/2 - nm/2 + 2n$	$n^2m/2 + nm/2 + 2n$

Table 1: Number of variables and constraints of the unlimited-buffer formulations.

3 MILP models for flowshop with zero buffer

The zero buffer environment is characterized by not having intermediary buffers between machines. Therefore, a situation known as blocking can occur, when a machine interrupts its production cycle (and it can also interrupt the production of previous machines), even though it has completed a job, because the next machine is not free. Moreover, when minimizing tardiness *and* earliness, it may be convenient for a job to stay on a machine after being completed even if the next machine is ready to process it.

A model, based on MUB1 (for the unlimited buffer environment) and on Ronconi's model [14] for the zero-buffer situation minimizing the total tardiness, is presented below. This model will be called MZB1 from now on. Variables of the model, with their corresponding bound constraints,

are: $x_{ij} \in \{0,1\}, i = 1,\ldots,n, j = 1,\ldots,n$, $E_j \geq 0, j = 1,\ldots,n$, $T_j \geq 0, j = 1,\ldots,n$, $B_{jk} \geq 0, j = 1,\ldots,n, k = 1,\ldots,m$, $I_{jk} \geq 0, j = 1,\ldots,n, k = 1,\ldots,m$, and $D_{jm}, j = 1,\ldots,n$. The new variable B_{jk} stands for the blocking time of the j-th job that, after being completed on machine k, stays on the machine. D_{jm} represents the departure time of the j-th job from the last machine.

$$\text{Minimize } \sum_{j=1}^{n} E_j + T_j \tag{34}$$

$$\text{subject to } T_j \geq D_{jm} - \sum_{i=1}^{n} x_{ij} d_i, \qquad j = 1,\ldots,n, \tag{35}$$

$$E_j \geq \sum_{i=1}^{n} x_{ij} d_i - D_{jm}, \qquad j = 1,\ldots,n, \tag{36}$$

$$D_{1m} = \sum_{k=1}^{m} (\sum_{i=1}^{n} x_{i1} p_{ik} + B_{1k}), \tag{37}$$

$$D_{jm} = D_{j-1,m} + I_{j-1,m} + \sum_{i=1}^{n} x_{ij} p_{im} + B_{jm}, \qquad j = 2,\ldots,n, \tag{38}$$

$$I_{jk} + \sum_{i=1}^{n} x_{i,j+1} p_{ik} + B_{j+1,k} = I_{j,k+1} + \sum_{i=1}^{n} x_{ij} p_{i,k+1} + B_{j,k+1}, \quad j = 1,\ldots,n-1, \quad k = 1,\ldots,m-1, \tag{39}$$

$$\sum_{i=1}^{n} x_{ij} = 1, \qquad j = 1,\ldots,n, \tag{40}$$

$$\sum_{j=1}^{n} x_{ij} = 1, \qquad i = 1,\ldots,n. \tag{41}$$

Constraints (35) and (36), which are used to identify the tardiness and the earliness of each job, are analogous to those of the unlimited buffer models, replacing the completion time of a job on a machine by its departure time. Constraint (37) states that the time when the first job leaves the last machine is equal to the sum of its processing and blocking times in each machine. For the departure time of the other jobs from the last machine, we have constraint (38), which also involves the idle times of the last machine. Constraint (38) says that the departure time of a job from the last machine is equal to the departure time of the previous job, plus the idle time between both jobs, plus the processing time of the job plus the time the job blocks the machine. Constraint (39) establishes the relationships required to keep the consistency between machines idle times and machines blocking times. These relationships are illustrated in Figure 2. Constraints (40) and (41) are those already presented in previous models, which ensure that each job will only have one position in the sequence and that each position will be taken only by one job.

Model MZB1 presented above uses variables to measure times related to two different states of the machines: empty and busy. When a machine k is empty, its idle time I_{jk} between every pair of consecutive jobs in positions j and $j+1$ is used in the formulation. When machine k is being

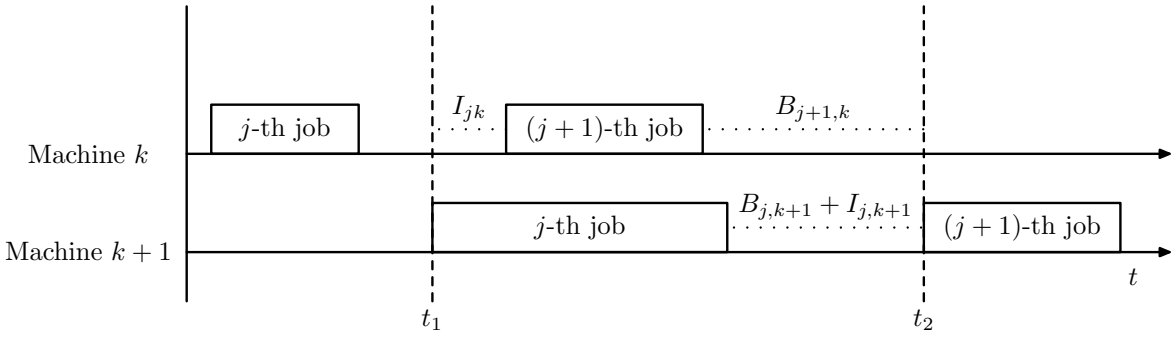

Figure 2: Graphical representation of model MZB1 constraint (39) that expresses the relationship among processing, blocking and idle times.

occupied by the j-th job, the time is divided into the time needed to process the j-th job, given by

$$\sum_{i=1}^{n} x_{ij} p_{i,k},$$

and the time B_{jk} during which the j-th job blocks machine k after having been processed.

In fact, it is not hard to see that there is no need to divide the time on this way, as the j-th job can be processed on machine k any time between its arrival time and its departure time. In this way, we eliminate the unnecessary imposition of processing the job as soon as it arrives. Therefore, the job can arrive to the machine, block it for a while, then be processed and finally block the machine again before leaving it. The alternative formulation below, called MZB2 from now on, models the problem by imposing very intuitive relations between the departure times of the jobs from the machines.

Variables of the model, with their corresponding bound constraints, are: $x_{ij} \in \{0,1\}, i = 1,\ldots,n, j = 1,\ldots,n$, $E_j \geq 0, j = 1,\ldots,n$, $T_j \geq 0, j = 1,\ldots,n$, and $D_{jk} \geq 0, i = 1,\ldots,n, k = 1,\ldots,m$. D_{jk} stands for departure of the j-th job from machine k. The other variables have the same meanings defined before.

$$\text{Minimize } \sum_{j=1}^{n} E_j + T_j \tag{42}$$

$$\text{subject to } T_j \geq D_{jm} - \sum_{i=1}^{n} x_{ij} d_i, \qquad j = 1, \ldots, n, \tag{43}$$

$$E_j \geq \sum_{i=1}^{n} x_{ij} d_i - D_{jm}, \qquad j = 1, \ldots, n, \tag{44}$$

$$D_{j1} \geq \sum_{i=1}^{n} x_{ij} p_{i1}, \qquad j = 1, \ldots, n, \tag{45}$$

$$D_{jk} \geq D_{j,k-1} + \sum_{i=1}^{n} x_{ij} p_{ik}, \qquad j = 1, \ldots, n, k = 2, \ldots, m, \tag{46}$$

$$D_{jk} \geq D_{j-1,k} + \sum_{i=1}^{n} x_{ij} p_{ik}, \qquad j = 2, \ldots, n, k = 1, \ldots, m, \tag{47}$$

$$D_{jk} \geq D_{j-1,k+1}, \qquad j = 2, \ldots, n, k = 1, \ldots, m-1, \tag{48}$$

$$\sum_{i=1}^{n} x_{ij} = 1, \qquad j = 1, \ldots, n, \tag{49}$$

$$\sum_{j=1}^{n} x_{ij} = 1, \qquad i = 1, \ldots, n. \tag{50}$$

Constraints (45) and (46) indicate that a job must be processed on a machine before leaving it. Constraint (45) is a special case for the first machine and says that the departure time of a job from the first machine must be greater than or equal to its processing time on the machine. Constraint (46) indicates that the departure time of a job from a machine must be greater than or equal to its arrival time (i.e, departure time from the previous machine) plus its processing time. Constraint (47) says that between the departure times of two consecutive jobs from a given machine, there must be enough time for the second job to be processed. In other words, the departure time of a job from a machine minus its processing time (that is something greater than or equal to its arrival time to the machine) must be greater than or equal to the departure time of the previous job from the machine. Constraint (48), deeply related to the zero-buffer situation, says that a job can departure from a machine, and arrive to the next one, only when the previous job in the sequence leaved the machine. Note that, for most of the combinations of indices j and k, constraints (46–48) are the linearization of

$$D_{jk} \geq \max \left\{ D_{j,k-1} + \sum_{i=1}^{n} x_{ij} p_{ik},\ D_{j-1,k} + \sum_{i=1}^{n} x_{ij} p_{ik},\ D_{j-1,k+1} \right\}. \tag{51}$$

That is, for each D_{jk}, if $D_{j,k-1} + \sum_{i=1}^{n} x_{ij} p_{ik} \neq D_{j-1,k} + \sum_{i=1}^{n} x_{ij} p_{ik} \neq D_{j-1,k+1}$, only one constraint will be active. It is worth noticing that inequality (51) resembles a formulation presented by Leisten [8] for the minimization of the makespan in the flowshop problem with blocking.

Summing up the characteristics of the two models presented in this section for the flowshop scheduling problem with zero buffer and minimizing earliness and tardiness, Table 2 shows the sizes of each model, expressed by their number of constraints, binary and continuous variables.

Model	binary variables	continuous variables	constraints
MZB1	n^2	$2nm + 3n$	$nm + 4n - m + 1$
MZB2	n^2	$nm + 2n$	$3nm + 3n - 2m + 1$

Table 2: Number of variables and constraints of the zero-buffer formulations.

4 Numerical experiments

In this section, we aim to compare the presented formulations based on the performance of the commercial solver CPLEX when applied to a set of instances varying the number of jobs n, the number of machines m, and considering different scenarios for the jobs due dates.

In the numerical experiments, we considered instances with

$$(n, m) \in \{(5, 3), (10, 3), (10, 7), (10, 10), (15, 3), (15, 7), (15, 10), (20, 3)\}.$$

Processing times were uniformly distributed between 1 and 99 (as suggested in [19]). Due dates were uniformly distributed between $P(1 - TF - DR/2)$ and $P(1 - TF + DR/2)$ (as suggested in [12]), where TF and DR are the tardiness factor of jobs and dispersion range of due dates, respectively, while P is a lower bound of the makespan on the flowshop with unlimited buffer [19] defined as:

$$P = \max\left\{ \max_{1\leq v\leq m}\left\{ \sum_{k=1}^{n} p_{kv} + \min_{1\leq k\leq n}\left\{ \sum_{q=1}^{v-1} p_{kq} \right\} + \min_{1\leq k\leq n}\left\{ \sum_{q=v+1}^{m} p_{kq}\} \right\} \right\}, \max_{1\leq k\leq n}\left\{ \sum_{v=1}^{m} p_{kv} \right\} \right\}.$$

The scenarios represent different configurations by varying TF and DR, as follows:

Scenario 1: low tardiness factor $TF = 0.2$ and small due date dispersion range $DR = 0.6$,

Scenario 2: low tardiness factor $TF = 0.2$ and wide due date range $DR = 1.2$,

Scenario 3: high tardiness factor $TF = 0.4$ and small due date range $DR = 0.6$,

Scenario 4: high tardiness factor $TF = 0.4$ and wide due date range $DR = 1.2$.

For each combination of n, m and scenario, we considered ten different randomly generated instances. Therefore, as a whole, the test set consists of $8 \times 4 \times 10 = 320$ problems. We set the value of the big M parameter in formulations MUB3 and MUB4 as $M = 100 \sum_{j=1}^{n} \sum_{k=1}^{m} p_{jk}$.

Formulations were written in AMPL (IBM ILOG AMPL 12.1.0) modelling language and solved using CPLEX 12.1. The experiments were performed in a 2.4GHz Intel Core2 Quad Q6600 with 4.0GB of RAM memory and Linux Operating System. The computer implementation of the models as well as the data sets used in our experiments and the solutions found are publicly available for benchmarking purposes at [23].

Table 3 shows the results of solving formulations MUB1, MUB2, MUB3 and MUB4 for all the instances in the test set. In the table, "CPU Time" means computational CPU time in seconds used by CPLEX to solve the problem, "Simplex It" is the total number of simplex iterations, and "B&B nodes" is the total number of explored nodes in the branch and bound tree. Each cell in the table represents 40 instances (4 scenarios and 10 instances per scenario). The mean was computed

excluding 2 instances from the top and 2 instances from the bottom tails of the CPU times. From the table, it can be seen that formulations MUB1 and MUB2 are very similar, when referring to the used CPU time. On the other hand, solving instances of formulations MUB3 and MUB4 is very time consuming. As MUB3 and MUB4 have half the number of integer variables of MUB1 and MUB2, the experiment confirms that, at least for the present set of test instances, it is not true that the computational effort needed to solve a MILP problem is proportional to the number of binary variables of the model, as claimed in [11]. When comparing the effort needed to solve MUB3 and MUB4, it is not possible to confirm either the claim that replacing regular constraints by new variables with lower and upper bounds (as in the modification to the Manne's model suggested by Liao [9]) makes the subproblems easier to be solved using CPLEX. This result may be related, as pointed out in [18], to the chosen value for M that greatly affects the solver performance.

A note on solving the large instances using models MUB3 and MUB4 is in order. Table 3 shows that those models need one order of magnitude more computational time to be solved when $n = 10$ and $m = 10$. Moreover, in instance with $n = 10$, $m = 10$, scenario 2, 8-th seed of both models, CPLEX solver with its default parameters reported as final point a non-integer solution with objective function value 1148.99678 (the optimal value of this instance is 1162). As pointed out in [18], the performance of the solver when applied to solve those models, is strongly related to the choice of the big M parameter in (23), (24) and (33). In [18], the authors mention that better performances were achieved for large values of M. However, our numerical experiments suggested that, due to rounding issues and scaling, large values of M prevent the solver from finding optimal integer solutions in many cases. Therefore, for using models MUB3 and MUB4, a user should face the problem of setting an adequate value for parameter M capable of providing optimal solutions in a reasonable computational time.

		Model MUB1			Model MUB2		
n	m	CPU Time	Simplex It	B&B nodes	CPU Time	Simplex It	B&B nodes
5	3	0.03	131.03	14.17	0.02	111.89	10.33
10	3	0.61	7,517.03	533.08	0.49	7,341.31	497.03
10	7	1.94	25,943.47	1,082.28	1.65	23,295.50	1,093.39
10	10	4.59	66,336.97	2,065.44	3.65	51,774.64	2,037.33
15	3	10.81	194,452.58	8,230.92	9.63	159,715.39	7,343.39
15	7	91.25	1,501,181.78	34,280.94	75.50	1,207,768.81	36,477.86
15	10	281.25	4,506,249.72	86,486.28	227.75	3,566,364.53	91,435.28
20	3	105.90	1,719,383.14	68,800.81	89.59	1,456,790.72	54,698.81

		Model MUB3			Model MUB4		
n	m	CPU Time	Simplex It	B&B nodes	CPU Time	Simplex It	B&B nodes
5	3	0.05	182.31	22.17	0.05	331.08	33.17
10	3	2.11	37,891.92	2,956.08	3.28	58,925.89	5,075.83
10	7	27.90	316,057.22	28,688.97	39.45	461,485.92	44,965.47
10	10	40.17	417,254.75	32,041.25	61.87	635,131.11	50,661.58

Table 3: Numerical evaluation of formulations MUB1, MUB2, MUB3 and MUB4.

Table 4 shows the results of solving formulations MZB1 and MZB2 for all the instances in the

test set. From the table, it can be seen that these formulations are very similar, MZB1 been a slightly easier to be solved in the smaller cases while the opposite situation occurs in the larger instances. The quotients between the number of branch and bound nodes used to solve models MZB1 and MZB2 are 1.32, 1.10, 0.99, 1.00, 1.10, 0.97, 0.94 and 1.13, for each of the instances dimensions listed in the table, respectively. Their average is 1.07, showing that 7% more nodes need to be explored in the branch and bound tree to solve model MZB1 in comparison with MZB2. The average number of simplex iterations needed to solve each node of the branch and bound tree is 38.91 for model MZB1 and 32.81 for model MZB2. Both models have the same binary variables, while model MZB2 has fewer continuous variables and more constraints than model MZB1. It seems that this situation helps to reduce the number of explored nodes and the number of simplex iterations needed to solve each node.

		Model MZB1			Model MZB2		
n	m	CPU Time	Simplex It	B&B nodes	CPU Time	Simplex It	B&B nodes
5	3	0.02	147.56	14.72	0.04	154.50	11.17
10	3	0.79	12,171.28	753.50	0.94	12,850.72	685.78
10	7	2.67	39,232.25	1,561.97	3.08	37,722.58	1,578.14
10	10	6.06	101,268.08	3,058.47	6.86	90,915.25	3,073.31
15	3	20.33	392,951.31	14,895.53	21.10	331,480.81	13,569.50
15	7	203.06	3,334,785.25	71,643.31	188.79	2,740,222.69	74,074.36
15	10	645.71	9,658,414.00	179,901.81	629.83	8,440,086.14	190,989.08
20	3	558.15	8,680,575.50	299,238.36	458.06	6,320,653.61	263,867.06

Table 4: Numerical evaluation of formulations MZB1 and MZB2.

5 Concluding remarks

In [2], it is stated that the problem solution time using branch and bound algorithms is unpredictable and that it depends on the numerical values of data. In the tests conducted, it could be noted that several factors interact, in addition to the number of binary variables, so that favorable or adverse conditions can be established to solve the model. That is, the time to solve a branch and bound algorithm does not have a direct relationship with the size of the model, although it is one of the factors involved in this relationship.

Observing the numerical results, it can be seen that the number of binary variables by itself does not indicate precisely the level of difficulty in solving mixed integer problems, contradicting the claims in [11]. This can be confirmed by the analysis of Table 3, which make it clear that formulations with a small number of binary variables (MUB3 and MUB4) took longer to achieve an optimal solution in the various scenarios analyzed.

Our numerical study strongly suggests that models based on n^2 positioning binary variables x_{ij}, named MUB1, MUB2, MZB1 and MZB2, are easier to be solved than models based on $n(n-1)/2$ precedence binary variables z_{ij}, named MUB3 and MUB4. The experiments also seem to suggest that, for models with the same number of binary variables, the ones with similar or less number of continuous variables and more constraints (MUB2 versus MUB1 in the unlimited buffer case, and MZB2 versus MZB1 in the zero buffer case) are slightly easier to be solved.

References

[1] D. Biskup and M. Feldmann, Benchmarks for scheduling on a single machine against restrictive and unrestrictive common due dates, *Computers & Operations Research* 28, pp. 787–801, 2001.

[2] P. Brandimarte, Neighbourhood search-based optimization algorithms for production scheduling: a survey, *Running Series - Designing Production Order Scheduling Tools* 5(2), 1992.

[3] R. A. Dudek, S. S. Panwalkar and M. L. Smith, The lessons of flowshop scheduling research, *Operations Research* 40, pp. 7–13, 1992.

[4] J. Grabowski and J. Pempera, The permutation flow shop problem with blocking: A Tabu Search approach, *Omega* 35, pp. 302–311, 2007.

[5] N. G. Hall and C. Sriskandarajah, A survey of machine scheduling problems with blocking and no-wait in process, *Operations Research* 44, pp. 510–525, 1996.

[6] J. J. Kanet and V. Sridharan, Scheduling with inserted idle time: problem taxonomy and literature review, *Operations Research* 48, pp. 99–110, 2000.

[7] Y. D. Kim, Minimizing total tardiness in permutation flowshops, *European Journal of Operational Research* 85, pp. 541–555, 1995.

[8] R. Leisten, Flowshop sequencing with limited buffer storage, *International Journal of Production Research* 28, pp. 2085–2100, 1990.

[9] C. J. Liao and C. T. You, An improved formulation for the job-shop scheduling problem, *Journal of the Operational Research Society* 43, pp. 1047–1054, 1992.

[10] A. S. Manne, On the job-shop scheduling problem, *Operations Research* 8, pp. 219–223, 1960.

[11] C. H. Pan, A study of integer programming formulations for scheduling problems, *International Journal of Systems Science* 28, pp. 33–41, 1997.

[12] C. N. Potts CN and L. N. van Wassenhove, A decomposition algorithm for the single machine total tardiness problem, *Operations Research Letters* 1, pp. 177–181, 1982.

[13] G. V. Reklaitis, Review of scheduling of process operations, *AIChE Symposium Series* 78, pp. 119–133, 1982.

[14] D. P. Ronconi, *A contribution to the study of the flowshop problem minimizing the total tardiness*, Ph.D. Dissertation (in portuguese), UNICAMP, Campinas, Brazil, 1997.

[15] T. Sen and S. K. Gupta, A state-of-art survey of static scheduling research involving due dates, *Omega* 12, pp. 63–76, 1984.

[16] S. P. Sethi, C. Sriskandarajah, G. Sorger, J. Blazewicz and W. Kubiak, Sequencing of parts and robot moves in a robotic cell, *International Journal of Flexible Manufacturing Systems* 4, pp. 331–358, 1992.

[17] E. F. Stafford, On the development of a mixed-integer linear programming model for the flowshop sequencing problem, *Journal of the Operational Research Society* 39, pp. 1163–1174, 1988.

[18] E. F. Stafford, F. T. Tseng and J. N. D. Gupta, Comparative evaluation of MILP flowshop models, *Journal of the Operational Research Society* 56, pp. 88–101, 2005.

[19] E. Taillard, Benchmarks for basic scheduling problems, *European Journal of Operational Research* 64, pp. 278–285, 1993.

[20] H. M. Wagner, An integer linear-programming model for machine scheduling, *Naval Research Logistic* 6, pp. 131–140, 1959.

[21] J. M. Wilson, Alternative formulations of a flow-shop scheduling problem, *Journal of the Operational Research Society* 40, pp. 395–399, 1989.

[22] Z. Zhu and R. B. Heady, Minimizing the sum of earliness/tardiness in multimachine scheduling: a mixed integer programming approach, *Computers & Industrial Engineering* 38, pp. 297–305, 2000.

[23] *http://www.ime.usp.br/∼egbirgin/*