IIETA International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Journal Européen des Systèmes Automatisés
Vol. 54, No. 4, August, 2021, pp. 591-597
Journal homepage: http://iieta.org/journals/jesa

Comparing Mixed-Integer Programming and Constraint Programming Models for the Hybrid Flow Shop Scheduling Problem with Dedicated Machines

Asma Ouled Bedhief[1,2]

[1] Department of Industrial Engineering, National Engineering school of Tunis, University of Tunis El Manar, Tunis 1002, Tunisia
[2] ESPRIT School of Engineering, Tunis 1002, Tunisia

Corresponding Author Email: asma.ouledbedhief@enit.utm.tn

https://doi.org/10.18280/jesa.540408

Received: 22 March 2021
Accepted: 20 July 2021

Keywords:
hybrid flow shop scheduling, dedicated machines, mixed-integer programming, constraint programming, Cplex, CP optimizer

ABSTRACT

The paper considers a two-stage hybrid flow shop scheduling problem with dedicated machines and release dates. Each job must be first processed on the single machine of stage 1, and then, the job is processed on one of the two dedicated machines of stage 2, depending on its type. Moreover, the jobs are available for processing at their respective release dates. Our goal is to obtain a schedule that minimizes the makespan. This problem is strongly NP-hard. In this paper, two mathematical models are developed for the problem: a mixed-integer programming model and a constraint programming model. The performance of these two models is compared on different problem configurations. And the results show that the constraint programming outperforms the mixed-integer programming in finding optimal solutions for large problem sizes (450 jobs) with very reasonable computing times.

1. INTRODUCTION

In many industrial applications, dedicated machines consist in producing products of different types. First, all products go through the same main operations of the production process, and then they are processed on dedicated machines specific to each product. Such applications include pharmaceutical industries [1], woodworking industries [2], label sticker manufacturing [3], furniture assembly and mass customization [4].

Hybrid Flow Shop (HFS) with dedicated machines with two or more stages is strongly NP-hard [5, 6].

In this paper, we deal with a two-stage HFS with dedicated machines and release dates. We focus on the case where there exists one machine in stage one, denoted as M_c, and two dedicated machines $D_k, k = \{1,2\}$ in the second stage. Each job j must first be processed on M_c and then, depending on its type, it will be processed on a dedicated machine D_k. Furthermore, we assume that each job j is available for processing at its release date r_j. The aim is to minimize the makespan (maximum completion time) denoted as C_{max}. Following the notation $\alpha|\beta|\gamma$ of Graham et al. [7], we denote such a problem as $2FHD\left|1,2,\ r_j\right|C_{max}$.

The rest of the paper is organized as follows:

Section 2 presents a brief literature review on exact, heuristic and metaheuristic methods that have been proposed for the HFS with dedicated machines. In Section 3, we deploy several notations and assumptions. Sections 4 and 5 describe respectively the mixed-integer programming (MIP) and the constraint programming (CP) models that we propose for the purpose of our paper. To assess the performance of these formulations, several test problems are solved, and the results are reported in Section 6. Finally, Section 7 concludes the paper.

2. LITERATURE REVIEW

In the academic literature, few exact methods have been employed to solve the HFS with dedicated machines.

Riane et al. [8] developed a dynamic program for the case of two stages, where two dedicated machines are in the second stage. The goal is to minimize the maximum completion time. The results show that the developed dynamic program is baffled for problems of more than 15 jobs.

Moseiov and Sarig [9] developed an integer programming model for the two-stage HFS with m dedicated machines and due dates. Besbes et al. [10] considered a two-stage HFS with parallel-dedicated machines in both stages. They proposed a mixed-integer linear programming model to minimize the maximum completion time among all jobs. This model is based on the mathematical formulation of Guinet et al. [11] and is solved by Cplex. The authors proved that Cplex can obtain an optimal solution for only small sizes. With the objective of minimizing the makespan, Hadda et al. [12] also proposed a branch and bound algorithm for the case of m dedicated machines in the second stage. The authors used an elimination rule to improve even more the performance of their algorithm. The experimental results demonstrated that many big size instances are solved, and the elimination rule has contributed to discard up to 50% of the nodes. Chikhi et al. [13] studied the case of the two-stage robotic flow shop scheduling problem with the objective of minimizing the makespan. There are two dedicated machines in stage 1 and only one machine in stage 2. Depending on the job type, each job is firstly processed on a dedicated machine and is then transported by a robot, to be processed on the single machine. The authors developed a mixed-integer programming model, which is using valid inequalities based on a set of lower bounds. Nabli et al. [14] proposed two mixed-integer

programming models (M1 and M2) for solving the hybrid flow shop scheduling problem with parallel machines in the first stage and two dedicated machines in the second stage. The first model (M1) is based on time indexed variables, while the second model (M2) is based on ordering variables. In their paper, the authors compared the computational time performances of M1 and M2, and they found that M2 (linear ordering variables) is faster than M1 (time index variables).

Other studies have developed heuristic and metaheuristic approaches for solving the hybrid flow shop with dedicated machines. For example, Lin et al. [3] developed a heuristic approach with combined rules for the two-stage HFS with setup times and due dates in a label sticker manufacturing company. The objective was to minimize the weighted maximum tardiness of jobs. Oguz et al. [15] developed a heuristic approach, which is based on the Johnson's algorithm [16] for minimizing the makespan.

Yang [17] proposed an optimal solution for the case where the processing times on the common machine are identical. The aim is to minimize the total completion time. Besbes et al. [10] considered respectively *s* and *m* dedicated machines in stages one and two. The authors developed two approaches. The first approach is based on the Johnson's algorithm [16], while the second one consists of a genetic algorithm. Huang and Lin [18] considered the case of two stages with setup time where a single machine is in the first stage and two dedicated machines are in the second stage. Their aim is to obtain a schedule that has the minimum makespan. They investigated the case where the processing schedules of the two types of jobs are fixed. To solve the problem, the authors proposed a polynomial-time dynamic programming algorithm.

Nabli et al. [19] studied a two-stage hybrid flow shop with dedicated machines and release dates. There are two parallel machines in stage one and two dedicated machines in stage two. The objective is to minimize the makespan. To solve the problem, the authors proposed three heuristic and two lower bounds. Harbaoui et al. [20] compared the performance of two metaheuristic: a tabu search and a genetic algorithm for the no-wait hybrid flow shop problem with dedicated machines under makespan minimization. The authors [21] studied a hybrid flow shop with dedicated machines, sequence dependent setup and time lags, and they developed a genetic algorithm to minimize the makespan.

Dealing with the minimization of the makespan for the case of *m* dedicated machines in the second stage, many heuristics were also developed in the references [22, 23].

For the case of more than two stages, Riane et al. [2] considered a three-stage HFS with two dedicated machines in the second stage and one single machine in stages one and three. They developed a dynamic programming-based heuristic and a branch and bound based heuristic. Ouled Bedhief et al. [1] studied the case of three-stage HFS with dedicated machines in the third stage. The authors studied a set of cases and proposed a heuristic approach for the general problem that is denoted by IH-DP. For the same problem, Ouled Bedhief et al. [24] proposed an improved genetic algorithm. Many computational experiments demonstrated its efficiency with a mean percentage deviation from the lower bound that does not exceed 0.5%, and a very reasonable computational time.

The literature review shows that most of the available studies on the hybrid flow shop with dedicated machines are focused on the development of heuristic approaches. However, few exact methods have been employed to solve this kind of problems.

Furthermore, most of these exact methods are mixed-integer linear programming models. To the best of our knowledge, there is no report on constraint programming (CP) models for the HFS scheduling problem with dedicated machines.

Based on logic programming, the constraint programming is, also, an efficient exact method that has been widely used for solving industrial scheduling problems [25-27].

It is further applied for solving many other types of scheduling problems such as: scheduling problems in operating theatres [28] or medical resident scheduling problems [29].

In this paper, we compare a constraint programming model (CP) to a mixed-integer programming model (MIP) for the $2FHD|1,2,\ r_j|C_{max}$ scheduling problem.

3. NOTATIONS AND ASSUMPTIONS

For convenience and readability, we will use the following notations for our problem $2FHD|1,2,\ r_j|C_{max}$:

- n: Number of jobs
- J: Set of n jobs, $J = \{1,2,\dots n\}$
- J_k: Subset of jobs of type $k, k = \{1,2\}$, such that $J_1 \cup J_2 = J$ and $J_1 \cap J_2 = \emptyset$
- M_c: Common machine of the first stage
- D_k: Dedicated machine of type $k, k = \{1,2\}$ of the second stage
- a_j: Processing time of job j on M_c
- b_{jk}: Processing time of job j on D_k , $k = \{1,2\}$
- r_j : Release date of job j, $j \in J$

Furthermore, we assume that each job should be processed on exactly one machine at the same time and each machine processes one job at a time. The transportation time between machines is zero and no preemption is allowed. Also, the job does not visit the same machine twice.

4. MIXED-INTEGER PROGRAMMING MODEL (MIP)

In this section, we propose a mixed-integer programming model for the $2FHD|1,2,\ r_j|C_{max}$ problem. This MIP model is composed only of linear equations. The goal is to minimize the makespan ($i.e$ maximum completion time of jobs).

At the beginning, we define the decision variables of our MIP model as follows:

Binary variables:

- $X_{0j} = 1$ if j is the first job processed on M_c, 0 otherwise; $j \in J$;
- $X_{ij} = 1$ if job i is scheduled before job j on M_c, 0 otherwise; $i, j \in J$ and $i \neq j$;
- $X_{0j}^{(k)} = 1$ if j is the first job processed on D_k, 0 otherwise; $j \in J_k, k = \{1,2\}$;
- $X_{ij}^{(k)} = 1$ if job i is scheduled before job j on D_k, 0 otherwise; $i, j \in J_k$, $k = \{1,2\}$ and $i \neq j$;

Continuous variables:

- C_j: Completion time of job j on M_c, $j \in J$;
- $C_j^{(k)}$: Completion time of job j on D_k, $j \in J_k, k = \{1,2\}$;

Our proposed MIP model can now be given as:
Objective function: $min\ C_{max}$.
Subject to:

$$\sum_{j=1}^{n} X_{0j} = 1 \ \ j \in J \tag{1}$$

$$\sum_{i=0, i\neq j}^{n} X_{ij} = 1 \quad j \in J \tag{2}$$

$$\sum_{i=1, i\neq j}^{n} X_{ji} \leq 1 \ j \in J \tag{3}$$

$$\sum_{j=1}^{n_k} X_{0j}^{(k)} = 1 \ \ j \in J_k; k \in \{1,2\} \tag{4}$$

$$\sum_{i=0, i\neq j}^{n_k} X_{ij}^{(k)} = 1 \ \ j \in J_k; k \in \{1,2\} \tag{5}$$

$$\sum_{i=1, i\neq j}^{n_k} X_{ji}^{(k)} \leq 1 \quad j \in J_k; k \in \{1,2\} \tag{6}$$

$$C_j \geq r_j + a_j \ \ j \in J \tag{7}$$

$$C_i + a_j + (X_{ij} - 1)M \leq C_j \ \ j \in J \tag{8}$$

$$C_j^{(k)} \geq C_j + b_{jk} \ j \in J_k; k \in \{1,2\} \tag{9}$$

$$C_i^{(k)} + b_{jk} + (X_{ij}^{(k)} - 1)M \leq C_j^{(k)} \ i,j \in J_k; \ i \neq j; \ k \in \{1,2\} \tag{10}$$

$$C_i \leq C_j + M\ (1 - X_{ij}^{(k)}) \ \ i,j \in J_k; \ k \in \{1,2\} \tag{11}$$

$$C_{\ max} \geq C_j^{(k)} \ \ j \in J_k; k \in \{1,2\} \tag{12}$$

$$C_j \geq 0 \ \ j \in J \tag{13}$$

$$C_j^{(k)} \geq 0 \ j \in J_k; k \in \{1,2\} \tag{14}$$

$$X_{ij} \in \{0,1\} \ i,j \in J \tag{15}$$

$$X_{ij}^{(k)} \in \{0,1\} \ \ i,j \in J_k; \ k \in \{1,2\} \tag{16}$$

$$X_{0j} \in \{0,1\} \ j \in J \tag{17}$$

$$X_{0j}^{(k)} \in \{0,1\} \tag{18}$$

In this mixed-integer programming model, the objective function is to minimize the maximum completion time C_{max} among all jobs.

M is a sufficiently large number (an upper bound on the completion time on both stage 1 (Constraint set (8)) and stage 2 (Constraint set (10)).

Constraint set (1) (resp. (4)) ensures that every job-sequence on M_c (resp.D_k) begins with exactly one job. Constraint set (2) (resp. (5)) assures that every job should have a predecessor, otherwise, it is the first job processed on M_c (resp. D_k). Constraint sets (3) and (6) state that every job must have at most only one direct successor. Constraint set (7) insures the respect of the release dates of jobs. Constraint set (8) (resp. (10)) states that no job can be processed on M_c (resp. D_k) before the completion time of the current job. Constraint set (9) ensures that the second operation can only begin when the first one is completed.

Constraint set (11) guarantees the obtaining of permutation solutions. In fact, a permutation schedule is defined when the processing order of jobs on the machines is the same.

The maximum completion time is defined through constraint set (12). Finally, constraint sets from (13) … to (18) define the decision variables of our model.

In general, the efficiency of mixed-integer programming is ostensibly not guaranteed, especially, when the problem size increases. However, constraint programming (CP) provides suitable modeling techniques to optimize several complex problems, which are most often qualified by logical and non-linear constraints.

5. CONSTRAINT PROGRAMMING MODEL (CP)

Unlike the previous model (MIP), Constraint Programming model uses a CP language which has a more advanced descriptive power than traditional linear programming language, and which allows non-linear equations to be incorporated.

In our CP model, operations on machines M_c and D_k are represented by interval variables instead of binary variables in MIP model. An interval variable represents an interval of time during which an operation is performed. We note here that the processing of job j on machine M_c ($resp. D_k, k = \{1,2\}$) is referred to as $operation\ [j]$ ($resp. Operation_k[j]$, $k = \{1,2\}$) and its duration is a_j ($resp. b_{jk}, k = \{1,2\}$).

The definition of interval variables for operations is given as follows:

$$INTERVAL\ Operation\ [j\ in\ J] size\ a_j, j \in J;$$
$$INTERVAL\ Operation_k\ [j\ in\ J_k]\ size\ b_{jk} \quad j \in J_k\ \forall\ k \in \{1,2\};$$

Accordingly, the constraint programming (CP) model will be as follows:
Objective function: $minimize\ C_{max}$.
Subject to:

$$endOf\ (Operation\ [j]) \geq r_j + sizeOf\ (Operation\ [j]) \quad j \in J \tag{19}$$

$$endOf\ (Operation\ [i]) \neq endOf\ (Operation\ [j]) \quad i,j \in J; i \neq j \tag{20}$$

$$endOf\ (Operation_k\ [j]) \geq endOf\ (Operation\ [j]) + sizeOf\ (Operation_k\ [j]) \quad j \in J_k; k \in \{1,2\} \tag{21}$$

$$endOf\ (Operation_k\ [i\ ]) \neq endOf\ (Operation_k\ [j\ ]) \quad i,j \in J_k;\ i \neq j;\ k \in \{1,2\} \tag{22}$$

$$C_{max} \geq endOf\ (Operation_k[j])\ j \in J_k; k \in \{1,2\} \tag{23}$$

In this constraint programming model, constraint set (19) ensures that completion time of job j on M_c is greater than or

equal to its release date plus its processing time. Constraint set (20) (resp. (22)) states that every two jobs i and j cannot be completed on M_c (resp. $D_k, k = 1,2$) at the same time. Constraint set (21) imposes that second operation of job j can only begin when the first one is completed.

Finally, the maximum completion time C_{max} is defined by the constraint set (23).

In our CP model, two jobs must not overlap since machines cannot occur simultaneously. To model this, we have also used two constructs:

- The sequence decision variables.
- The noOverlap scheduling constraints.

A sequence variable represents a total order over a set of interval variables. The definition of sequence variables in our CP model is given as below:

$$SEQUENCE\ Schedule\ in\ all\ (j\ in\ J)\ Operation\ [j];$$
$$SEQUENCE\ Schedule_k\ in\ all\ (j\ in\ J_k)\ \ Operation_k\ [j]$$
$$\forall\ k \in \{1,2\};$$

NoOverlap constraints are added to constrain the intervals in sequences such that they:

- are ordered in time corresponding to the order in the sequence.
- do not overlap.
- respect transition times.

The definition of NoOverlap constraints in our CP model is given by:

$$noOverlap\ (Schedule);$$
$$noOverlap\ (Schedule_k\ )\ \forall\ k \in \{1,2\};$$

6. COMPARISON OF THE TWO MODELS

In this section, we compare the performance of the two formulated models MIP and CP to assess which is the most successful for solving the $2FHD|1,2,\ r_j|C_{max}$ problem. The criteria that we use for this purpose are:

- the problem size that the model can solve.
- the computational time for solving this problem size.

To achieve this objective, a set of numerical experiments were performed on a personal computer with an Intel 2.50 GHz CPU and 1.96 GB RAM.

The Mixed-Integer Programming model (MIP) is coded in ILOG OPL STUDIO V6 and solved by Cplex 12.

Cplex is a major product release that incorporates the latest enhancements in both solution speed and flexibility for mathematical programming. While constraint programming model (CP) is coded in ILOG OPL STUDIO V6 and solved by CP Optimizer that is included in IBM ILOG CPLEX Optimizers.

Furthermore, four classes of test problems are generated to perform the computational analysis. In each class problems, processing times in the first stage are random integers from a uniform distribution from 1 to 20, denoted by $a_j \sim U[1,20]$. However, the balance between average workloads of dedicated machines and the total workload on the single machine is important driving factor to easily find a satisfying solution for the studied problem. Hence, the processing times of jobs on the dedicated machines are generated randomly from the following uniform distributions:

CLASS 1: $b_{jk} \sim U[1,20] \forall k \in \{1,2\}$

CLASS 2: $b_{jk} \sim U[1,40] \forall k \in \{1,2\}$

CLASS 3: $b_{jk} \sim U[20,40] \forall k \in \{1,2\}$

CLASS 4: $b_{jk} = a_j + 5$ with $k \in \{1,2\}$

In fact, for CLASS 1, the processing times of jobs are of the same order of magnitude. Thus, the total load on each of the dedicated machines is less than that of the single machine. As for CLASS 2 and CLASS 3, the processing times of jobs on dedicated machines are greater than those on M_c, which tends to balance the global load of different machines. Finally, for CLASS 4, we intend to study specific instances, which are more difficult to schedule.

We further note that release dates of jobs are generated randomly from two uniform distributions:

$$r_j \sim U[0, 100] \text{and } r_j \sim U[0,\ n * 10].$$

For each class of test problems, several problem sizes are evaluated. For each combination of parameters (r_j, n), we randomly generated 20 instances and provided the average computation time of each proposed model for finding the optimal solution.

6.1 Results analysis for MIP model

Table 1 presents the average computational time (seconds) that is needed by Cplex for finding the optimal solution, for $r_j \in [0.100]$ and $r_j \in [0.n * 10]$.

The sign (-) indicates that Cplex solver failed to obtain an optimal solution.

The results show that Cplex provides an optimal solution within a reasonable time for only small sizes that do not exceed 9 jobs.

In fact, mixed-integer programming performs poorly with the growth of problem size, increasing computing time. This can also be observed from Figures 1 and 2 below.

However, the running time of Cplex does not exceed 9.1 seconds when the number of jobs is less than 8.

6.2 Results analysis for CP model

Table 2 presents the average computational time (seconds) that is needed by CP Optimizer for finding the optimal solution, for $r_j \in [0.100]$ and $r_j \in [0.n * 10]$.

The results reveal that constraint programming model outperforms the other formulation (MIP model) in finding optimal solutions for large problem sizes.

In fact, CP optimizer can obtain an optimal solution for almost all cases of test problems for which the problem size does not exceed 450 jobs. Moreover, such an optimal solution can be obtained within a very reasonable time that is less than three minutes for n=450 jobs. This can also be observed from Figures 3 and 4 below.

Table 1. Results analysis for MIP model (Cplex solver)

$r_j \in [0, 100]$						
	$n = 5$	$n = 6$	$n = 7$	$n = 8$	$n = 9$	$n = 10$
CLASS 1	0.02	2.94	4.54	4.66	7200	-
CLASS 2	0.03	3.66	5.90	7.27	7920	-
CLASS 3	2.00	2.49	5.71	9.04	-	-
CLASS 4	3.70	4.74	4.90	8.60	-	-
$r_j \in [0, n * 10]$						
	$n = 5$	$n = 6$	$n = 7$	$n = 8$	$n = 9$	$n = 10$
CLASS 1	0.02	0.32	1.53	2.90	10800	-
CLASS 2	0.04	0.90	2.42	3.06	11520	-
CLASS 3	0.03	0.95	1.30	3.37	-	-
CLASS 4	0.34	1.22	4.59	9.03	-	-

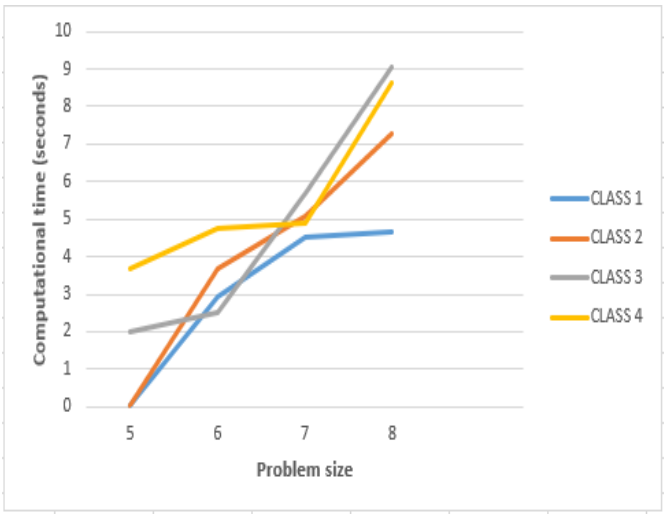

Figure 1. Average CPU time of Cplex for $r_j \in [0, 100]$

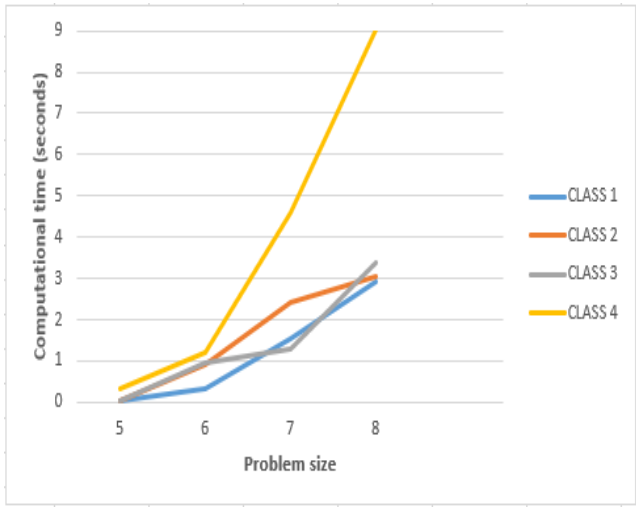

Figure 2. Average CPU time of CP Optimizer for $r_j \in [0. n * 10]$

Table 2. Results analysis for CP model (CP Optimizer)

$r_j \in [0. 100]$						
	$n = 50$	$n = 100$	$n = 150$	$n = 250$	$n = 350$	$n = 450$
CLASS 1	0.03	1.50	4.37	19.1	59.9	201
CLASS 2	0.04	1.57	4.71	18.6	64.3	196
CLASS 3	0.03	1.28	4.11	15.6	112	204
CLASS 4	0.03	1.60	4.96	22.3	163	-
$r_j \in [0. n * 10]$						
	$n = 50$	$n = 100$	$n = 150$	$n = 250$	$n = 350$	$n = 450$
CLASS 1	0.02	0.75	1.54	3.40	17.2	45.6
CLASS 2	0.02	0.63	1.07	3.15	19.9	57.8
CLASS 3	0.03	0.71	1.53	4.52	25.5	80.7
CLASS 4	0.02	0.83	1.42	7.56	16.3	61.9

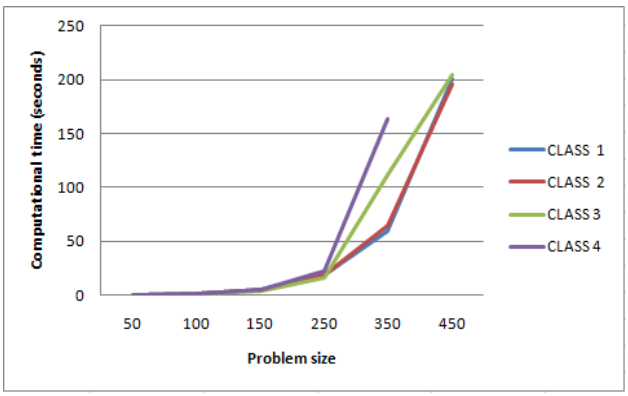

Figure 3. Average CPU time of CP Optimizer for $r_j \in [0, 100]$

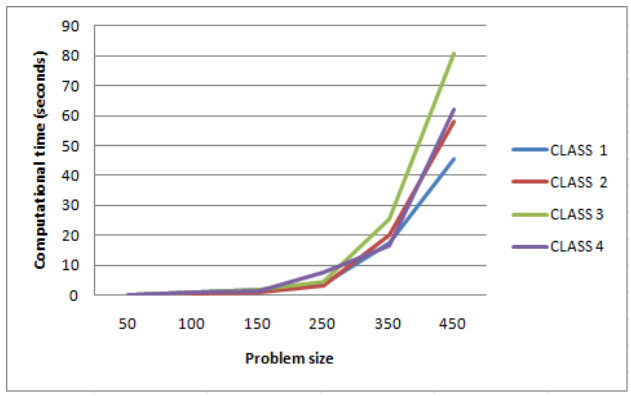

Figure 4. Average CPU time of CP Optimizer for $r_j \in [0. n * 10]$

Considering the results per problem class, we can see from Table 2 and from Figures 3 and 4 that average CPU time of CP Optimizer increases with the problem size. But it is, on average, similar from one problem class to another. This is to say that the performance of CP Optimizer for obtaining an optimal solution for the HFS with dedicated machines is not very sensitive to data variations and their influences on machine workloads.

These results may be explained by two factors:

- The first factor is that the number of variables in MIP model increases more quickly than in CP model.

- The second factor consists in constraint propagation along with domain filtering, which are fundamental in constraint programming strategy. Indeed, the propagation phase can reduce the size of the search space and avoid exploring an exponential size space. In the initial constraint propagation stage, the CP engine uses domain filtering techniques to eliminate infeasible values from the domains of variables. A filtering algorithm is applied to make the CP model arc consistent. This means, for any value of a given variable in a constraint, there exists at least a value for the other variables to satisfy this constraint. All constraints must be filtered, and filtering is repeated until no domain is changed. This reduces considerably the search space.

Consequently, as an advantage, CP model allows various constraints to be taken into consideration, because the solver takes parcels out of the search space. Contrariwise, the MIP model does not let many complicated constraints which are found in real life to be integrated without convoluting the model since it becomes difficult to build, generalize and solve.

However, constraint programming remains a tree-based searching method. On some large problem sizes, this tree traversal becomes very combinatorial and penalizing in terms of execution time.

7. CONCLUSION

In this paper, we have dealt with the two-stage hybrid flow shop with dedicated machines and releases dates. The objective is to minimize the makespan. Such a problem is NP-hard.

To obtain an optimal solution, we have proposed two mathematical models for the problem, namely a mixed integer-programming model and a constraint programming model. A set of computational experiments was conducted to evaluate the performance of the developed models. The results showed that constraint programming dominates the other formulation (mixed-integer programming) in finding optimal solutions. In fact, with constraint programming model, we can obtain an optimal solution for large problem sizes, which can reach 450 jobs, within reasonable computational time ($\leq 3\ minutes$). On the other hand, we find that mixed integer programming can solve only small sizes (≤ 9 jobs) within more than 192 minutes (when the number of jobs is 9).

Going forward, we intend to assess the performance of these two models on more realistic situations such as multiple stages with dedicated machines. Moreover, we are interested to compare the developed models using various parameterizations of search methods in CP Optimizer.

REFERENCES

[1] Ouled Bedhief, A., Dridi, N. (2019). Minimizing makespan in a three-stage hybrid flow shop with dedicated machines. International Journal of Industrial Engineering Computations, 10(2): 161-176. http://dx.doi.org/10.5267/j.ijiec.2018.10.001

[2] Riane, F., Artiba, A., Elmaghraby, S.E. (1998). A hybrid three-stage flow shop problem: Efficient heuristics to minimize makespan. European Journal of Operational Research, 109(2): 321-329. https://doi.org/10.1016/S0377-2217(98)00060-5

[3] Lin, H.T, Liao, C.J. (2003). A case study in a two-stage hybrid flow shop with setup time and dedicated machines. International Journal of Production Economics, 86(2): 133-143. http://dx.doi.org/10.1016/S0925 5273(03)00011-2

[4] Cheng, T.C.E., Lin, B.M.T., Tian, Y. (2009). Scheduling of a two-stage differentiation flow shop to minimize weighted sum of machine completion times. Computers and Operations Research, 36(11): 3031-3040.

[5] Herrmann, J.W., Lee, C.Y. (1992). Three-machine look-ahead scheduling problems. Research Report No. 92-93, Department of Industrial Engineering, University of Florida, FL.

[6] Lin, B.M.T. (1999). The strong NP-hardness of two-stage flow shop scheduling with a common second-stage machine. Computers & Operations Research, 26(7): 695-698. https://doi.org/10.1016/S0305-0548(98)00080-X

[7] Graham, R.L., Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G. (1979). Optimization and approximation in deterministic sequencing and scheduling: A survey. Annals of Discrete Mathematics, 5: 287-326. http://dx.doi.org/10.1016/S0167-5060(08)70356-X

[8] Riane, F., Artiba, A., Elmaghraby, S.E. (2002). Sequencing a hybrid two-stage flow shop with dedicated machines. International Journal of Production Research, 40(17): 4353-4380. https://doi.org/10.1080/00207540210159536

[9] Mosheiov, G., Sarig, A. (2010). Minimum weighted number of tardy jobs on an m-machine flow shop with a critical machine. European Journal of Operational Research, 201(2): 404-408. http://dx.doi.org/10.1016/j.ejor.2009.03.018

[10] Besbes, W., Loukil, T., Teghem, J. (2010). A two-stage flow shop with parallel dedicated machines. 8th International Conference of Modeling and Simulation, MOSIM'10- May 10-12, 2010- Hammamet- Tunisia.

[11] Guinet, A., Solomon, M.M., Kedia, P.K., Dussauchoy, A. (1996). A computational study of heuristics for two-stage flexible flowshops. International Journal of Production Research, 34(5): 1399-1415. http://dx.doi.org/10.1080/00207549608904972

[12] Hadda, H., Dridi, N., Hajri-Gabouj, S. (2014). Exact resolution of the two-stage hybrid flow shop with dedicated machines. Optimization Letters, 8(8): 2329-2339. http://dx.doi.org/10.1007/s11590-014-0741-y

[13] Chicki, N., Abbas, M., Benmansour, R., Bekrar, A., Hanafi, S. (2015). A two-stage flow shop scheduling problem with transportation considerations. 4OR, A Quarterly Journal of Operations Research, 13(4): 381-402.

[14] Nabli, Z., Korbaa, O., Khalfallah, S. (2016). Mathematical programming formulations for hybrid flow shop scheduling with parallel machines at the first stage and two dedicated machines at the second stage. In 2016 IEEE International Conference on Systems, Man, and

Cybernetics (SMC). pp. 4389-4393. http://dx.doi.org/10.1109/SMC.2016.7844921
[15] Oguz, C., Lin, B.M.T., Cheng, T.C.E. (1997). Two-stage flow shop scheduling problem with a common second-stage machine. Computers & Operations Research, 24(12): 1169-1174. https://doi.org/10.1016/S0305-0548(97)00023-3
[16] Johnson, S.M. (1954). Optimal two- and three-stage production schedules with setup times included. Naval Research Logistics Quarterly, 1(1): 61-68. http://dx.doi.org/10.1002/nav.3800010110
[17] Yang, J. (2015). Minimizing total completion time in two-stage hybrid flow shop with dedicated machines at the first stage. Computers & Operations Research, 58: 1-8. http://dx.doi.org/10.1016/j.cor.2014.11.012
[18] Huang, T.C., Lin, B.M.T. (2013). Batch scheduling in differentiation flow shops for makespan minimization. International Journal of Production Research, 51(17): 5073-5082.
[19] Nabli, Z., Khalfallah, S., Korbaa, O. (2018). A two-stage hybrid flow shop problem with dedicated machine and release date. Procedia Computer Science, 126: 214-223. https://doi.org/10.1016/j.procs.2018.07.235
[20] Harbaoui, H., Khalfallah, S. (2020). Tabu-search optimization approach for no-wait hybrid flow-shop scheduling with dedicated machines. Procedia Computer Science, 176: 706-712. https://doi.org/10.1016/j.procs.2020.09.043
[21] Harbaoui, H., Bellenguez-Morineau, O., Khalfallah, S. (2016). Scheduling a two-stage hybrid flow shop with dedicated machines, time lags and sequence-dependent family setup times. In 2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp. 002990-002995. https://doi.org/10.1109/SMC.2016.7844695
[22] Dridi, N., Hadda, H., Hajri-Gabouj, S. (2009). Méthode heuristique pour le problème de flow shop hybride avec machines dédiées. RAIRO Operations Research, 43(4): 421-436. http://dx.doi.org/10.1051/ro/2009024
[23] Wang, S., Liu, M. (2013). A heuristic method for two-stage hybrid flow shop with dedicated machines. Computer & Operations Research, 40(1): 438-450. http://dx.doi.org/10.1016/j.cor.2012.07.015
[24] Ouled Bedhief, A., Dridi, N. (2020). A genetic algorithm for three-stage hybrid flow shop scheduling problem with dedicated machines. Journal Européen des Systèmes Automatisés, 53(3): 357-368. https://doi.org/10.18280/jesa.530306
[25] Baptiste, P., Le Pape, C., Nuijten, W. (2001). Constraint-based scheduling: applying constraint programming to scheduling problems. Springer Science & Business Media.
[26] El Khayat, G., Langevin, A., Riopel, D. (2006). Integrated production and material handling scheduling using mathematical programming and constraint programming. European Journal of Operational Research, 175(3): 1818-1832. https://doi.org/10.1016/j.ejor.2005.02.077
[27] Harjunkoski, I., Grossmann, I.E. (2002). Decomposition techniques for multistage scheduling problems using mixed-integer and constraint programming methods. Computers & Chemical Engineering, 26(11): 1533-1552. http://dx.doi.org/10.1016/S0098-1354(02)00100-X
[28] Wang, T., Meskens, N., Duvivier, D. (2012). A comparison of mixed-integer programming and constraint programming models for scheduling problem in operating theatres. In 2012 International Conference on Information Systems, Logistics and Supply Chain.
[29] Topaloglu, S., Ozkarahan, I. (2011). A constraint programming-based solution approach for medical resident scheduling problems. Computers & Operations Research, 38(1): 246-255. http://dx.doi.org/10.1016/j.cor.2010.04.018