Received November 25, 2021, accepted January 10, 2022, date of publication January 28, 2022, date of current version February 8, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3146706

An Effective Optimization Approach to Minimize Waste in a Complex Industrial System

HOUDA HARBAOUI[1] AND SOULEF KHALFALLAH[2]
[1]Higher Institute of Informatics and Telecommunication Technologies ISITCom, University of Sousse, Sousse 4054, Tunisia
[2]MARS Research Laboratory, School of Management of Sousse (ISG), University of Sousse, Sousse 4054, Tunisia

Corresponding author: Houda Harbaoui (houda.harbaoui1@gmail.com)

ABSTRACT Industrial waste is a major environmental concern. This paper proposes an optimization approach to better manage the operations of a food production line, with the aim of minimizing material waste. The configuration considered is inspired by a real situation. It is a two-stage flow shop with dedicated machines at stage 1 and identical parallel machines at stage 2. A mathematical model is first proposed that considers the minimization of the number of idle times exceeding an acceptable threshold as an objective. A positive time lag for the product and machine setup is considered as a constraint. To solve large problem instances, a hybrid genetic algorithm (HGA) is proposed. In addition, we developed a two-phase procedure based on a decision tree to select the parameters of the HGA. The results show that we could achieve the minimization of material waste without altering too much, another industrial goal, which is the total production time.

INDEX TERMS Scheduling, hybrid flow shop, genetic algorithm, waste minimization, time lag, industry, integer program.

I. INTRODUCTION

Many industries are facing two major environmental problems, such as energy consumption and material waste. One example of these industries is pasta production. In this context, energy consumption is mainly due to the drying cabins, while material waste is owed to long idle time of a production line, causing the perishability of pasta stuck in the machine. In this article we consider a shop configuration inspired from the pasta industry. In this context, traditional evaluation criteria such as completion time known as the makespan (Cmax) or maximum tardiness (Tmax), etc. do not reflect the only objectives of the industry. A major concern in this situation is how to minimize the waste of perishable material due to a long machine idle time (IT). In this context, if the IT is below the non-perishability time limit, then the material or pasta stack inside the machine could be recycled, otherwise the material is thrown away. This represents a considerable loss to the industry. In other words, the objective is to minimize the number of occurrences of idle times exceeding the "non-perishability" limit. To our knowledge, as we will explain later in the literature review, such objective was only considered by us in a previous article [1]. This was confirmed by a literature review by Hesran *et al.* [2]. Compared to the work in [1], the MIP (Mixed Integer Program) is modified to result in more acceptable results for the industry, and in addition a metaheuristic is proposed for large size problems.

The production process is a hybrid flow shop (HF) with two stages. Stage one (S1) represents the production phase and contains two unrelated parallel machines, also called dedicated machines. Stage two (S2) represents the drying phase and is made of several identical parallel machines. In fact, S2 contains ten drying cabins in our case. The problem can be defined as follows: a set J of n jobs should be processed on S1, then on S2, where J = j1, j2,…jn. This set is subdivided into two subsets or families, J1 and J2, where J = J1 U J2. Jobs from J1 should be processed on the first dedicated machine M1 at S1; they are no wait jobs. This means that they should be processed on S2 (the drying cabins) immediately when they finish processing on S1. Jobs from J2 should be processed on the second dedicated machine M2 at S1; they are authorized to wait for a maximum time limit before going through S2. This means that we are in a situation with a maximum time lag (lag+). In addition, jobs from either J1 or J2 were subdivided into groups when processed on S1. In fact, a machine setup is needed to

The associate editor coordinating the review of this manuscript and approving it for publication was Shih-Wei Lin.

perform a mold change if two successive jobs, processed on the same machine, are from different groups. Following the $\alpha|\beta|\gamma$ notation of Graham *et al.* [3], this problem can be noted as HF2(PD2,Pm)|$ST_{g,sd}$,$lag+_i$|Nbr_IT+, where $ST_{g,sd}$ designates the constraint sequence-dependent group setup, $lag+_i$ designates the maximum machine time lag for j_i, and Nbr_IT+ is a notation introduced by us to designate the number of IT grater than perishability time limit (θ). In the previous notation; proposed by Yang *et al.* [4], the dedicated machines appear in the α term as PD2, while the identical parallel machines at S2 are represented by Pm. This problem could also be noted HF2 (R2,Pm)|Mj,$ST_{g,sd}$, $lag+_i$|Nbr_IT+; following Ribas *et al.* [5] notations. In this second notation, R in the α field represents unrelated parallel machines, and M_j in the β field indicates machine eligibility for the job. In several industrial cases, a drying cabin is considered a batch machine that can handle multiple tasks in one operation. This could not be applied to our problem, as jobs do not have the same drying characteristics. The main contribution of this paper, compared to our previous work, is an approach to consider an important objective for the industry, which is the minimization of material waste through the minimization of Nbr_IT+ while giving special attention to satisfactory Cmax values. In a previous conference paper study, we investigated the same configuration. In the first article [6], two MIPs were proposed and compared, as well as several LBs, to minimize the Cmax. A second paper [1] presented a MIP for waste minimization without considering acceptable solutions to the industry. In a third study, a genetic algorithm (GA) approach [7] was proposed to minimize Cmax. In the present work, a third MIP and GA are proposed. The objective was to minimize material waste while maintaining a threshold for Cmax.

The remainder of this paper is organized as follows. Section 2 presents a literature review of a two-stage HF with dedicated machines. Section 3 explains the constraints and the objectives considered in this paper. In Section 4, MIP is explained. Section 5 describes the GAs. Computational analysis is presented in Section 6. We close with a conclusion and perspectives.

II. LITERATURE REVIEW

In the following, we first investigate the literature on scheduling with simultaneous constraints. Then, we examined works dealing with scheduling operations for waste reduction.

A. FLOW SHOP SCHEDULING WITH SIMULTANEOUS CONSTRAINTS

In 2009, both Ruiz *et al.* [8] and Goncharov *et al.* [9] published reviews on non-permutation flow shop problem (NPFSP), in which the authors concluded that there was little work considering two simultaneous constraints: no-wait and no_idle_time (N_IT). To our knowledge, the N_IT and no-wait constraints were first studied as separate constraints by Adiri and Pohoryles [10] in a permutation flow shop (PFS) in order to minimize the makespan. Kalczynski and Kamburowski [11] studied the same constraints simultaneously. In addition, Baraz and Mosheiov [12] presented a hybrid greedy algorithm with an enhancement heuristic to solve a NPFS to minimize the Cmax. Shao *et al.* [13] proposed a hybrid mimetic algorithm to solve a NIPFSP. The authors worked on the Cmax criterion and compared their results with those of the existing algorithms. More recently, Wang *et al.* [14] considered the two-stage no-wait HF scheduling problem with the ST_{sd} at S1. The authors proposed a LB based on the Hungarian method, three heuristics, a branch and bound (B&B) approach and a tabu search (TS) algorithm. Khare et Agrawal. [15] proposed a configuration with a no-wait hybrid flowshop and sequence-dependent setup times (SDSTs). They proposed a MIP for small-sized problems and a pairwise iterated greedy (PIG) algorithm to solve medium- and large-size problems. To conclude, the problem considered in this paper, and in two papers published by us ([1], [6]), are the only works considering three constraints simultaneously, that is the no-wait, lag+, and ($ST_{g,sd}$). Compared to previous studies, this study is the first to consider a new objective, which is the minimization of waste due to the occurrence of an idle time exceeding a given threshold (Nbr_IT+), and at the same time, indirectly considering a threshold on a second objective, the Cmax. It is worth mentioning that the objectives and constraints considered by us are inspired by a real situation.

B. WASTE MINIMIZATION IN OPERATIONS SCHEDULING

According to Dewi *et al.* [16], waste is caused by operational activities. The author identified seven types of operational waste: overproduction, waiting, movement transportation, unnecessary processes, inventory, and defects. Le Hesran [2] presented a review of waste minimization in operations scheduling. The author classified 71 articles according to their environmental and economic objectives. The authors identified four domains in which waste management was important. These domains are the cutting stock problem with scheduling aspects, the integrated cutting stock problem, the batch and hoist scheduling problem, and shop floor scheduling. According to this review, waste is mainly due to setup, idle time, operations sequencing, surface treatment, due dates, and machine cleaning. The waste is mainly composed of wastewater and perishable materials, such as food and chemical products. The same author [17] presented a method for the identification and modeling of waste-conscious scheduling problems using flow control. In the study case of hubcap manufacturing, the economic and environmental impacts are quantified, and decision variables for problem solving are provided. Georgios *et al.* [18] considered yoghurt production, a representative food process, in a large-scale dairy facility in Greece. They used rescheduling to avoid demand disruptions, leading to product waste. Table 1 presents a summary of the papers dealing with scheduling problems with simultaneous constraints.

NPFS: No-idle Permutation Flow shop; HF2: Hybrid Flow shop with two stages;

TABLE 1. Contributions on the scheduling problems with simultaneous constrain.

Reference	Problem	PD	ST_{sd}	$ST_{g,sd}$	lag^{+}_{i}	no-wait	no-idle-time	Objective	Solution method
Adiri and Pohoryles [10]	NPFS					*	*	C_i	Proof
Baraz and Mosheiov [12]	NPFS					*	*	Cmax	PIG, heuristic
Shao et al. [13]	NPFS						*	Cmax	HMA
Wang et al. [14]	HF2		*			*		Cmax	B&B, heuristics
Khare et Agrawal. [15]	HF2		*			*		Tw	MIP, algorithms
Our study	HF2	*		*	*	*	*	Cmax, Σ_{Nbr_IT+}	MIP, HGA

PD: Parallel Dedicated Machine;

Cmax: makespan; Tw: Total weighted tardiness; $\sum C_i$: sum of completion times; $\sum$ *Nbr_IT*+: sum of idle−time

HGA: Hybrid genetic algorithm; HMA: Hybrid Memetic Algorithm; PIG: pair−wise iterated greedy: Greedy-based fixing algorithm; B&B: Branch−and−Bound

III. EXPLANATION OF THE CONSTRAINTS AND THE OBJECTIVE

Given that the multitude of concepts considered in this paper can lead to confusion, this paragraph explains the constraints and objectives. In the following, we start with the presentation of the positive time lag (lag+) and the no-wait constraints. Then, we considered the setup constraints $ST_{g,sd}$. Finally, we exhibit the objective Nbr_IT+, that is, the minimization of the number of idle times exceeding a certain limit objective.

A. POSITIVE TIME LAG AND NO-WAIT CONSTRAINTS

A positive time lag (lag+), is an upper bound (UB) on the elapsed time between two successive operations of the same job on S1 and S2. Let C_{j1} be the completion time of j_j on S1 and D_{j2} the starting time of the same job on S2. Then the time lag TL = D_{j2} - C_{j1}. The constraint states that TL $\leq$ lag+. The following graph illustrates a situation with acceptable and non-acceptable TLs when lag+ = 2.

The No-wait constraint occurs when lag+ = 0.

B. THE SETUP CONSTRAINT

For the problem under consideration, the setup constraint is considered when two successive jobs, on the same machine at S1, are from different groups. Thus, if it is possible to process all the jobs belonging to the same group successively, without altering another constraint, the total setup time will be minimized. Figure 2 illustrates this situation. Let J_i and J_j be two jobs processed on M1 of S1, and belong to the same group, and $J_k$and J_l be two jobs processed on the same machine but belong to another group. And let the setup time = 30 min.

C. THE Nbr_IT+ OBJECTIVE

Nbr_IT+ is the number of idle times that exceed a given threshold θ. The idle time (IT) is the elapsed time between the end and starting time of the processing of two successive jobs on the same machine at S1. Let J_i and J_{i+1} be two successive jobs on a given machine in S1. The IT = C_{i1} − $D_{i+1,1}$. If IT > θ, then Nbr_IT+ is incremented by one. The following figure illustrates the situation where $\theta = 1$.

IV. MATHEMATICAL FORMULATION

In a previous paper, Harbaoui *et al.* [6] proposed a MIP model for the problem under consideration to minimize the Cmax. This model (MIP 1) generated either optimal solutions or UBs for small-size problems (up to 20 jobs). This formulation was then extended to minimize Nbr_IT+ (MIP 2). As mentioned by Hesran *et al.* [2], this is the first time that such an objective is considered. In fact, for each idle time exceeding the perishability time limit θ, the material stuck in the machine cannot be recycled. This results in a correlation between Nbr_IT+ and the quantity of waste material. The notations used in both MIPs are as follows:

M_k: set of machines at stage k

P_{ik}: processing time of job i at stage k

S_{ik}: setup time for job i at stage k

L_{i1}: time lag of job i at stage 1

B_{ij}: a binary matrix indicating whether j_i and j_j are from the same group.

The decision variables common to either MIP 1 or MIP 2 are as follows:

t_{ik}:Starting time of job i on stage k

$Cmax$:Completion time of all jobs makespan

$x_{ij}^{m,k}$ = {1 if job j is processed immediately after job i on machine m of stage k;0 otherwise }

$a_{i}^{m,k}$ = {1 if job i is assigned to the machine m of stage k; 0 otherwise }

It is worth mentioning that for S1, $a_{i}^{m,1}$ is known because of the machine eligibility property at that stage. For MIP 2, two additional decision variables were introduced by Harbaoui *et al.* [19]:

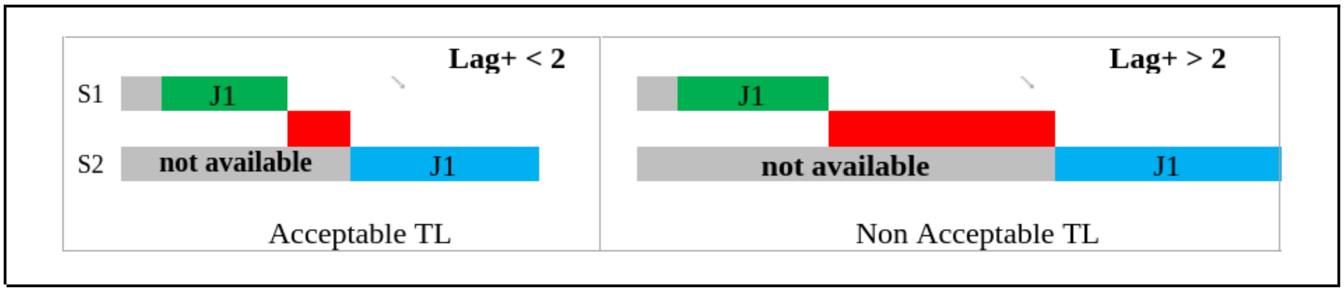


FIGURE 1. Illustration of acceptable and unacceptable TL.

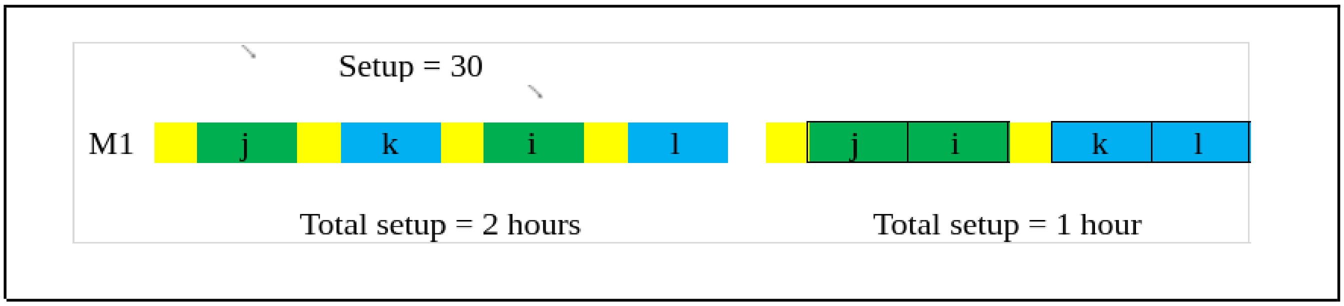


FIGURE 2. Illustration of minimization of setup time.

TM_{ij}^{m1}:The elapsed time between the processing of job i and job j on machine m on stage 1

$v_i^1 = \{$ 1 if there is IT ≥ 0; 0 otherwise $\}$

In MIP 2, two constraints were added, compared to MIP 1. The first ensures that the IT could not be smaller than the time elapsed between the completion and the starting time of two consecutive jobs affected to the same machine. The second one counts Nbr_IT+. Additionally, the Cmax constraint was deleted.

$$\text{Min} \sum_{i=0}^{n} v_i^1$$

$$\text{S.C} \sum_{i=1}^{n+1} x_{i0}^{m,k} = 0, \quad m \in M_k, k \in \{1,2\}, \tag{1}$$

$$\sum_{i=0}^{n} x_{n+1i}^{m,k} = 0, \quad m \in M_k, k \in \{1,2\}, \tag{2}$$

$$x_{ij}^{mk} + x_{ji}^{mk} \leq a_i^{m,k}, \quad i = 1\ldots n, \; j = 1\ldots n, m \in M_k, k = \{1,2\}, i \neq j \tag{3}$$

$$\sum_{\substack{i=0 \\ i \neq j}}^{n} x_{ij}^{m,k} = a_j^{m,k} \quad \forall j = 1\ldots n+1, m \in M_k, \; k \in \{1,2\} \tag{4}$$

$$\sum_{\substack{j=1 \\ j \neq i}}^{n+1} x_{ij}^{m,k} = a_i^{m,k}, \quad \forall i = 0\ldots n, m \in M_k, \; k \in \{1,2\} \tag{5}$$

$$a_0^{m,k} = 1, \quad \forall m \in M_k, k \in \{1,2\} \tag{6}$$

$$a_{n+1}^{m,k} = 1, \quad \forall m \in M_k, k \in \{1,2\}, \tag{7}$$

$$a_i^{m,1} = 1, \quad \forall i = 1..n, m = A_i \tag{8}$$

$$\sum_{m=1}^{m_2} a_i^{m,2} = 1 \quad i = 1\ldots n \tag{9}$$

$$t_{j,k} \geq t_{i,k} + P_{i,k} - M(1 - x_{ij}^{mk}) + S_{j,k}.B_{ij}, \; i = 0..n, j = 1..n, m \in M_k, k = \{1,2\}, i \neq j \tag{10}$$

$$t_{i,2} \leq t_{i,1} + P_{i,1} + l_i, \quad i = 1\ldots n \tag{11}$$

$$t_{i,2} \geq t_{i,1} + P_{i,1}, \quad i = 1\ldots n \tag{12}$$

$$TM_{ij}^{m,1} \geq t_{j,1} - (t_{i,1} + P_{i,1}) - M(1 - x_{ij}^{m,1}), \; i = 1\ldots n, j = 1\ldots n, \forall i \neq j, m \in M_1, \tag{13}$$

$$TM_{ij}^{m,1} \leq M.v_i^1 + \alpha.x_{ij}^{m,1}, \quad i = 1\ldots n, \; j = 1\ldots n \forall i \neq j, m \in M_1, \alpha = 30 \tag{14}$$

$$x_{ij}^{m,k} \in \{0,1\}, m \in M_k, i = 1\ldots n, \; j = 1\ldots n, k = \{1,2\}, i \neq j \tag{15}$$

$$a_i^{m,k} \in \{0,1\}, m \in M_k, i = 1\ldots n, k = \{1,2\} \tag{16}$$

$$v_i^1 \in \{0,1\}, i = 1\ldots n \tag{17}$$

$$TM_{ij}^{m,1}, t_{i,k} \geqslant 0, i = 1\ldots n, \; j = 1\ldots n, m = 1..m_k, k = \{1,2\} \tag{18}$$

In this model, the objective function minimizes the Nbr_IT+. Constraints (1) and (2) ensure that the fictitious jobs j_0 and j_{n+1} have either zero predecessor or zero successor. Constraint (3) ensures that each job has at most one predecessor and one successor on the machine to which it is assigned. Constraints (4) and 5) state that each job must be processed exactly once on one machine for each stage. Constraints (6) and (7) pre-assign fictitious jobs to all machines. Constraint (8) and (9) represent the pre-assignment of jobs on each machine in S1 and S2. Constraint (10) expresses disjunctive property. Constraints (11) and (12) are the technological constraints. Constraints (13) and (14) successively measure IT and the number of IT > θ. Finally, constraints (15) to (17) are the non-negativity and binary constraints. Finally, the Mixed Integer Program 3 (MIP3), proposed in this paper, is based on following principle:

Given that MIP2 generated solutions, sometimes not acceptable for the Cmax criteria, another MIP (MIP3) is created by adding to MIP2 a constraint representing an upper limit on the value of the completion time. This upper limit is derived from MIP 1 and is expressed as follows:

$$t_{i,2} + P_{i,2} \leq \text{UB}, \quad \text{i=1..n}$$

The UB does not concern the objective of the model but it concerns the Cmax.

V. PRESENTATION OF THE GENETIC ALGORITHMS

Metaheuristics have been proposed to solve optimization problems known as NP-hard. Their objective was to generate good solutions within an acceptable time. In the literature, several metaheuristics have been proposed to solve HF. In this work, we chose GAs, as they proved their effectiveness with several problems with a configuration close to the one considered in this study. Indeed, we found several authors affirming the efficiency of GAs in the resolution of HF with different constraints, such as Reeves [20], Ruiz and Maroto, [21], Kahraman *et al.* [22], Jolai *et al.* [23], Besbes *et al.* [24], Sioud *et al.* [[25], [26]] and Wang and Liu [27]. This motivated us to choose the resolution approach.

GAs were first introduced by Holland and then further developed by Goldberg and Deb [28]. They were inspired by Darwin's theory of natural evolution. A GA belongs to the family of evolutionary algorithms that are based on a set of several solutions simultaneously called a population. It evolves a population of solutions gradually from one generation to the other through a process of operators, that are, selection, reproduction, crossover, and mutation, to create new populations with better individuals. This process is repeated through generations until a final population is obtained with individuals of better quality. In this population, the best individuals get as close as possible to the optimal solution. The procedures proposed in this work consider two objectives: the minimization of Cmax and minimization of Nbr_IT+ without explicitly treating the problem as a bi-objective. We propose two hybrid GAs, called GA-C-I and GA-I-C. Figure 4 illustrates the functioning of the two procedures.

The GA-C-I main objective is to minimize the Cmax and evaluate the corresponding Nbr_IT+. For this first GA, the population is first sorted according to the Cmax, and then according to Nbr_IT+. The GA-I-C main objective is to minimize Nbr_IT+ and evaluate the corresponding Cmax. For this second GA, the population is first sorted according to Nbr_IT+, followed by Cmax. Through GA-C-I, which is not the main objective of this work, we will get an idea on how far we are from a more prominent Cmax based solution.

In the next subsections, the solution encoding and the genetic operators are explained.

A. SOLUTION ENCODING

A solution or a chromosome of this GA is represented by a permutation of all jobs. The job that appears first in the sequence is affected first to its dedicated machine. In the second stage, to obtain a full schedule, the jobs are processed in a FIFO manner. It is worth mentioning that the FIFO provides the only feasible solution for the no-wait jobs and a feasible solution for lag+ jobs. Figure 5 illustrates the encoding for a ten jobs' problem.

B. GENERATING THE INITIAL POPULATION

The initial population is composed of 200 individuals divided into three sub-populations:

- The first sub-population, of size six, is generated by dispatching rules; that are:
  - -- SPTS1: "Smallest Processing Time on stage 1"
  - -- SPTS2: "Smallest Processing Time on stage 2"
  - -- LPTS1: "Longuest Processing Time on stage 1
  - -- LPTS2: "Longuest Processing Time on stage 2"
  - -- ITPT: "Increasing Total Processing Time
  - -- DTPT: "Decreasing Total Processing Time
- The second sub-population, whose size depends on the number of job groups, is generated by considering one setup per group of jobs. Thus, the setup time is minimized. This sub-population presents 10% of the initial population, that is, 20 individuals. The principle is to use different group sequences to generate job permutations (see Algorithm 1 [19]). Indeed, we will consecutively launch all the jobs of the same group affected to the same machine to minimize the setup time. The choice of this sub-population is motivated by observations of the structure of the optimal solutions generated by the MIPs.
- The last sub-population, which represents the rest of the individuals, is generated randomly. i.e. 174 individuals.

C. SELECTION, CROSSOVER, MUTATION OPERATOR

In this section, we start by presenting the GA operators, and then explain how we select the best combination of them. The aim of these operators is to create a new generation with globally better fitness's than the previous one.

- Selection
  The selection operators appear in two steps of the GA. The first one is the selection of parents for cross-over and the second one is the selection of individuals that will survive for the next generation. In this study, we experiment with well-known genetic operators. The roulette wheel method and the random selection were tested to choose two parents from the current population for crossover. Then, elitism selection is applied to choose the best individuals among all individuals in order to create the new generation.
- Crossover
  The two parents crossed over to produce two infants. We tested two techniques: SJOX (similar job order

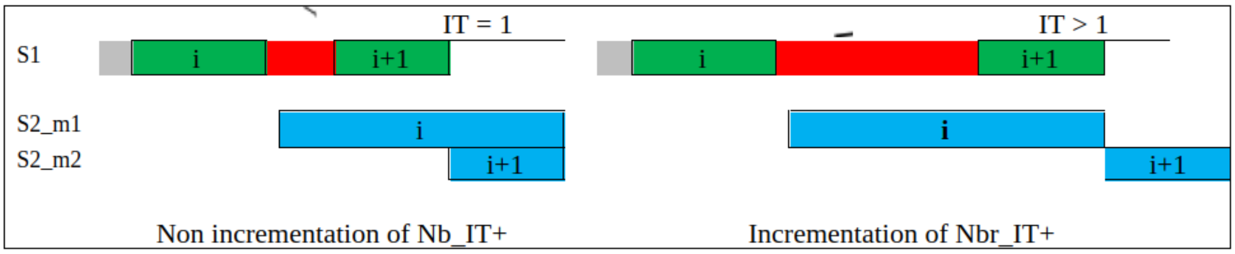


FIGURE 3. Illustration of Nbr_IT+ incrementation.

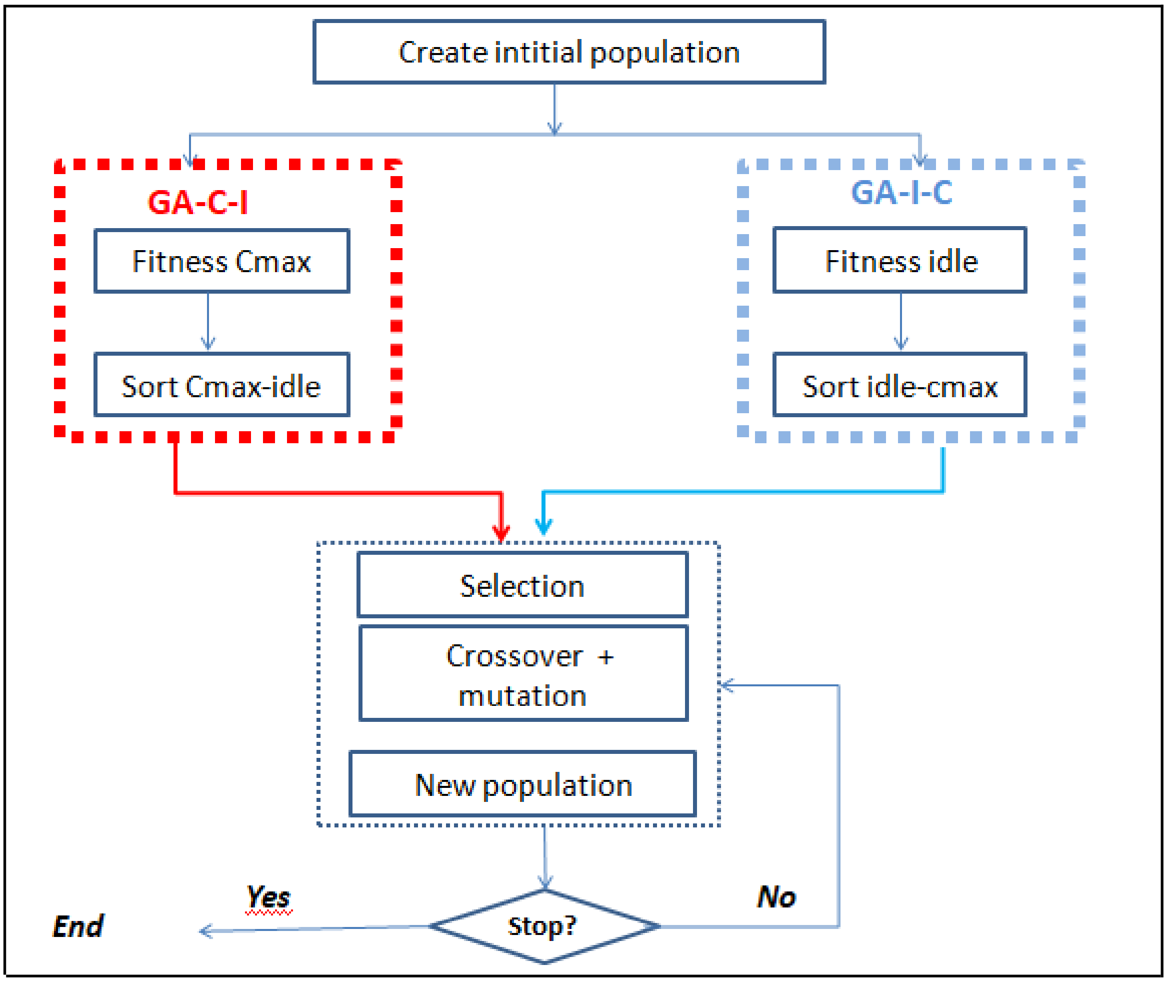


FIGURE 4. Description of the two gas.

crossover) proposed by Ruiz and Maroto [21] and random maximal preservative crossover (RMPX) proposed by Sioud *et al.* [25].

- Mutation
  The mutation brings diversification to the population, in order to efficiently explore the research space and avoid premature convergence. In our study, we tested two techniques frequently used in the literature: swap and insert (FIGURE. 6). Swap involves randomly choosing two jobs, j_i and j_j, and swapping them. The insert involves randomly choosing a job j_i and a position p (different from that of j_i) and then reinserting j_i in position p.

Several combinations of operators were tested, and the best one was chosen experimentally. The best combination is presented in Table 2:

We developed a procedure based on two phases to choose the best combination of operators. Phase one is a decision tree approach, and phase two is based on an Excel pivot table.

-- In phase one, the decision tree tool used in this research (SIPINA) [29], enables us to identify the operators that differentiate the most between the results. This problem turns out to be the crossover operator.

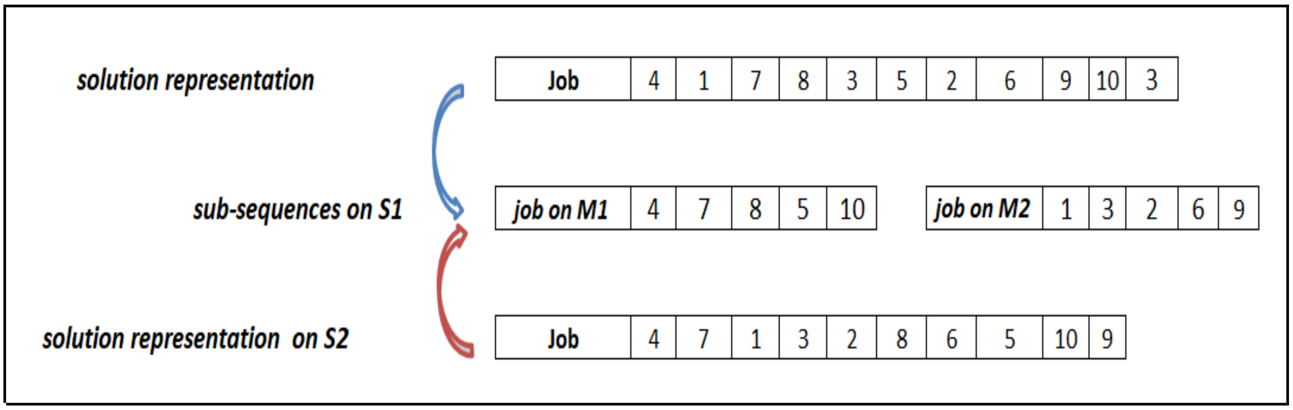


FIGURE 5. Solution encoding.

TABLE 2. The operators' chosen to generate the results.

Operator	Type
Selection	Roulette Wheel
Crossover	RMPX
Mutation	Insert

-- In phase two, a cross analysis of the crossover operator with the other operators reveals the best combination of operators.

In fact, we chose two instances of each family of problems that we tested with all combinations of parameters. It should be noted that the assessment of each instance is based on the deviation of Cmax from the LB (RD). Given the randomness of GA, we do not necessarily obtain the same result by running the same instance multiple times, even if the same settings are maintained. On this basis, the objective is to determine the setting that is likely to produce the largest number of good solutions (top 10%) with relative deviations (RD) and not necessarily the best solution. The method used to calibrate the GA is based on three steps. The first step consists of grouping the RDs into classes, which allows the labeling of the results. The second step is to apply a classification algorithm to determine which operator affects the result the most. The third step is to use Excel's pivot table to determine the RDs' averages based on the crossing of different operators. We illustrate in what follows the procedure and the results for the family of problems with 20 jobs.

1) Step 1: Grouping of results (see Table 3)
   We started by ranking the solutions according to the average deviation (AD) from the best LB (i.e. RD).
2) Step 2: Operator that mostly affects the solutions (see Table 4)
   The second step is to apply a classification algorithm based on a decision tree to determine the operator that most affects the RDs. We used the SIPINA software, which enabled us to conclude that the crossover operator better differentiated between the results (see table below).
   In this context, RMPX was superior to SJOX (low RD values). Figure 7 illustrates the results of the SIPINA application. The fact that the crossover operator, as an attribute, is at the first level of the tree means that it differentiates between RD values better than other operators (mutation, number of generations, size of the population).
3) Step 3: Cross-referencing of results
   The third step is to determine the RD averages for the other operators. Tables 5 and 6 present the results according to the mutation operator and the number of generations. These results were generated using Excel's pivot-table tool.

It appears that the combination of the RMPX crossover operator with the SWAP mutation operator and a population of size 200 produces smaller RDs compared to other combinations in the case of 20 job problems.

Even though the results are slightly better with 500 generations, it is necessary to notice that the running time is quite high. For this, we stopped at G200 and not at G500. On this basis, for the GA, we adopted the parameters shown in the following table.

D. FITNESS EVALUATION

We propose choosing the best individual by introducing a filter function. To do so, in the case of GA-I-C, for each individual, we first compute Nbr_IT+ on S1 and then calculate its Cmax. We used a solution ranking function with two filters in terms of switching from one generation to another. We first sort individuals according to Nbr_IT+; then, we sort; in ascending order of Cmax, the individuals with the same Nbr_IT+. In this way, the population is organized in layers, where each layer contains sorted solutions in ascending order of Nbr_IT+ and Cmax (see Figure 8). The fitness evaluation for GA-C-I follows the same logic by sorting individuals based on Cmax, followed by Nbr_IT+.

Even though the objective was to obtain solutions with zero Nbr_IT + and therefore No-waste, we accepted solutions that do not necessarily satisfy a null number of Nbr_IT +. To do so, we propose a filtering function that allows a gradual

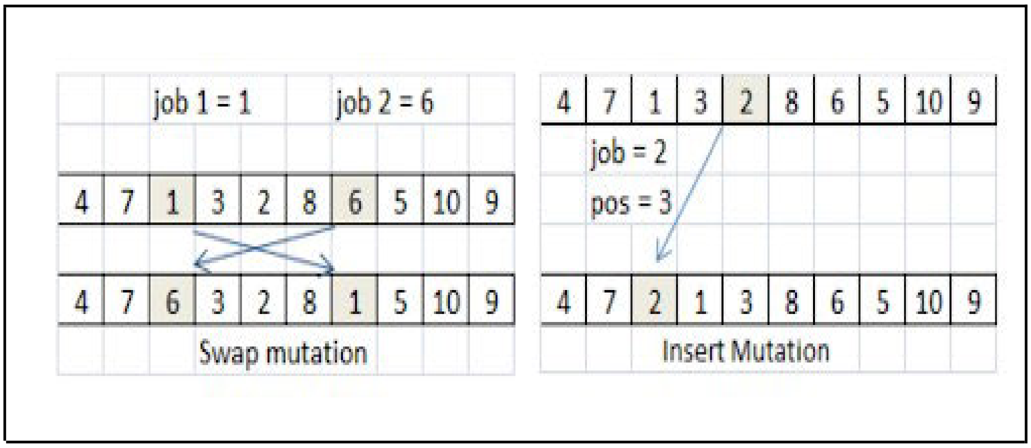


FIGURE 6. The insert and swap.

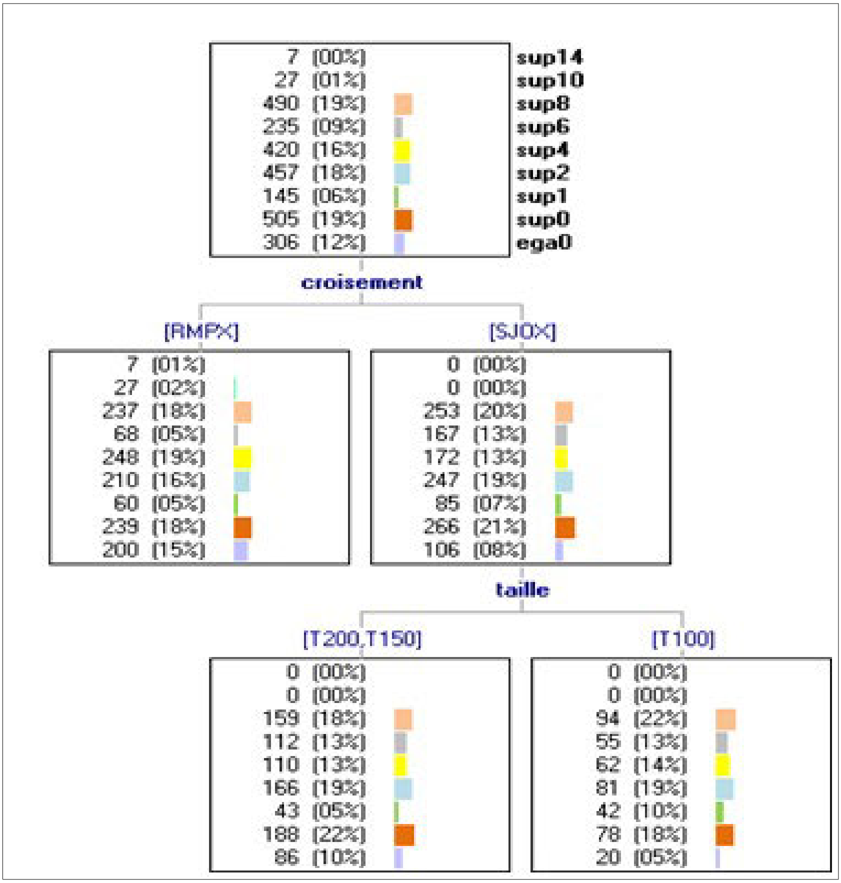
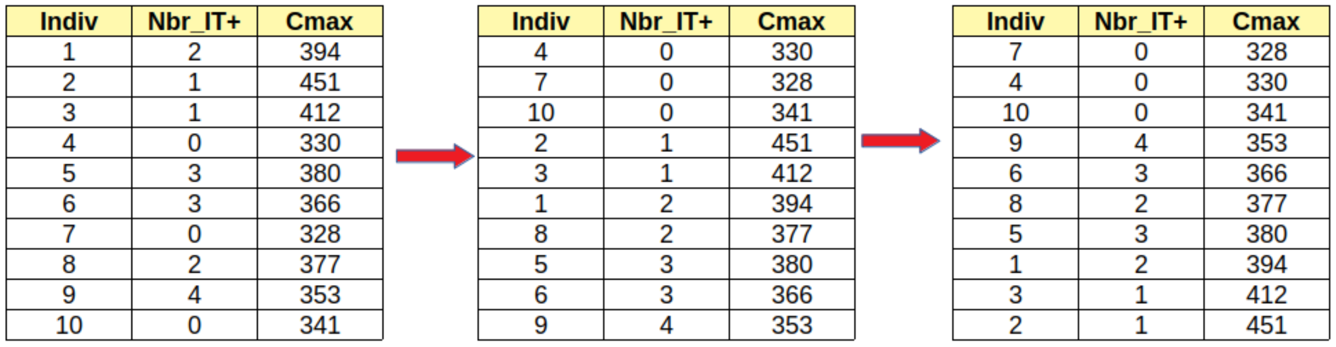


FIGURE 7. The SIPINA decision tree.

transition from less good solutions (dominated solutions) to good-quality solutions (dominant solutions).

E. ITERATIVE LOCAL SEARCH

Although the crossover and mutation operators guarantee the diversification and intensification of the search, their application does not always guarantee obtaining solutions with the desired quality. The introduction of a local search (LS) makes it possible to further expand the search space, thereby obtaining better solutions. An iterative local search iterates the descent method, which is the LS. The LS remove a job from one position and insert it in another position different from the current position. If the new solution found presents a better solution than an update of the best current solution



Indiv	Nbr_IT+	Cmax
1	2	394
2	1	451
3	1	412
4	0	330
5	3	380
6	3	366
7	0	328
8	2	377
9	4	353
10	0	341

Indiv	Nbr_IT+	Cmax
4	0	330
7	0	328
10	0	341
2	1	451
3	1	412
1	2	394
8	2	377
5	3	380
6	3	366
9	4	353

Indiv	Nbr_IT+	Cmax
7	0	328
4	0	330
10	0	341
9	4	353
6	3	366
8	2	377
5	3	380
1	2	394
3	1	412
2	1	451

FIGURE 8. Classification of individuals by two optimization criteria.

TABLE 3. Results classification.

RD	0	>0	>1	>2	>4	>6	>8	>10	>12	>14
Classe	Eg0	Sup0	Sup1	Sup2	Sup4	Sup6	Sup8	Sup10	Sup12	Sup14

TABLE 4. Parameters goodness of fit.

Parameter	Goodness of fit	Acceptance
Crossover	0.0174	Yes
Population size	0.0023	No
Nbr. generations	0.0000	No
Selection	0.0000	No

will be carried out until the stopping criterion is reached. The stopping criterion is the number of iterations without improvement. This was experimentally set to 10. For more details, see Harbaoui [19].

F. EVALUATION AND VALIDATION OF THE MIP AND GA

A tight lower bound (LB) for Cmax permits the evaluation of the quality of a solution. In a previous work, Harbaoui *et al.* [6] tested a different LB and proposed a tight LB that assumes the existence of a waiting constraint and the smallest processing time on S2 is added. The authors added the setup time of each group only once, as if jobs from the same group were processed successively. It is evident that for the objective minimizing of Nbr_IT +, the LB is zero.

VI. COMPUTATIONAL ANALYSIS

This section is dedicated to computational experiments. We start with a presentation of the instances, and then we discuss the results generated by the proposed GAs. The evaluation of the quality of the algorithm is based on a comparison of the results to the LB and to the MIP output.

A. DATA GENERATION AND SETTINGS

Because there is no benchmark available for this problem, we randomly generated 30 instances for each family of problems. These instances depend on the total number of jobs (n), the number of jobs dedicated to M1 of S1 (n1), the number of machines at S2 (n-n1), the way the processing times are generated, the way the $ST_{g,sd}$ and lag+ are generated.

Two families of instances were generated. The first one is inspired by a real situation, where jobs from J1 are no-wait and jobs from J2 have a lag+ constraint. The second family of instances was proposed for theoretical purposes of exploring different instances.

Parameters common to both families of instances:

- Number of stages $|K| = 2$;
- Number of jobs n: (10, 20, 50, and 100);
- Distribution of the setup times: uniform [5…20];
- Distribution of the time lags: uniform [0…30];
- Number of dedicated machines on S1: 2;
- Number of parallel machines on S2: 10;
- Number of jobs dedicated to M1 of S1; n1: (0.5n, 0.6n, 0.7n);

TABLE 5. Crossover operator vs mutation operator.

	Insert			SWAP			All
Operator	0.2	0.5	Av	0.2	0.5	Av	Total
RMPX	3.859	3.839	3.849	3.7	3.854	3.777	3.813
0.7	4.15	4.315	4.233	4.248	4.336	7.292	4.262
0.9	3.794	3.7	3.747	3.494	3.828	3.661	3.704
1.0	3.634	3.502	3..568	3.358	3.397	3.377	3.473
SJOX	3.943	4.031	3.987	4.019	3.907	3.963	3.975
0.7	4.093	4.132	4.113	4.295	4.141	4.218	4.165
0.9	3.891	4.084	3.988	3.865	3.811	3.838	3.913
1.0	3.847	3.876	3.861	3.897	3.768	3.833	3.847
Total	3.901	3.935	3.918	3.859	3.880	3.870	3.894

TABLE 6. Operators vs number generations.

	G100	G200	G500	General
RMPX	3.934	3.894	3.611	3.813
Insert	3.911	3.960	3.677	3.849
swap	3.958	3.828	3.545	3.777
SJOX	3.997	3.964	3.964	3.975
Insert	3.982	3.975	4.004	3.987
Swap	4.011	3.953	3.925	3.963
General	3.965	3.929	3.788	3.894

TABLE 7. Final parameters of the GA.

Operator	Type
Crossover	RMPX 1.0
Mutation	SWAP 0.5
Nbr. Generations	200
Population Size	200

The values obtained by n1 were obtained from Wang and Liu. ([27], [30]) and Yang. [4]. For the other parameters, the datasets depend on the family instances, as presented below.

1) Family 1 data sets: The processing times at S1 and at S2 are generated as following: $P_{i1} \in [50,150]$ and $P_{i2} \in [180,780]$
2) Family 2 data sets: The processing times at S1 and at S2 are generated as following:
   - Class 1: $P_{i1} \in [1, 40]$ and $P_{i2} \in [5, 200]$;
   - Class 2: $P_{i1} \in [40, 80]$ and $P_{i2} \in [200, 400]$;
   - Class 3: $P_{i1} \in [80, 100]$ and $P_{i2} \in [400, 600]$;

These instances were randomly generated using a uniform distribution. The processing time intervals with respect to the ratio, $|M1|/|M2| = 2/10$.

In the following, the results generated by the MIP models are presented, followed by a description of the GA's results. Finally, the results were analyzed and compared.

B. RESULTS GENERATED BY THE MATHEMATICAL MODELS

In the following, we compare first the results generated by the three models. As mentioned above, model 1 (MIP1) is the Cmax minimization, model 2 (MIP2) is the Nbr_IT+ minimization, and model 3 (MIP3) is the Nbr_IT+ minimization with an additional constraint on Cmax.

For MIP1, we recorded for each instance the objective value Cmax and the corresponding Nbr_IT+. For MIP2 and MIP3, we recorded for each instance the objective values Nbr_IT+ and the corresponding Cmax. Table 8 presents the results for the first family of instances, which is inspired by a real situation. Column one represents the model, columns 2 to 4 represent the average (ARD), maximum, and minimum relative deviations. Columns 5 to 7 represent the number of instances with zero Nbr_IT+, with only one Nbr_IT+, and with two or more Nbr_IT+. It is worth mentioning that zero Nbr_IT+ represents the optimal solution for the minimization of Nbr_IT+.

The relative deviation RD is defined as follows:

$$RD_i = \frac{Sol_i - LB}{LB} \times 100$$

Sol_i is the Cmax of the solution under consideration. Given that we generated 30 instances for each configuration, the

TABLE 8. **MIP's results for the family 1 instances.**

	10 job Problems							20 job Problems					
	Cmax			Nbr_IT+				Cmax			Nbr_IT+		
	Avg	Min	Max	"0"	"1"	"2+"		Avg	Min	Max	"0"	"1"	"2+"
MIP1	0,00	0,00	0,00	10	15	5	MIP1	5,63	1,44	16,77	2	6	22
MIP2	0,00	0,00	0,00	29	1	0	MIP2	10,20	3,30	19,40	30	0	0
MIP3	0,00	0,00	0,00	30	0	0	MIP3	7,32	2,20	17,70	30	0	0

TABLE 9. **MIP's results for family 2 - Class 1 instances.**

		10 job's Problems						20 job's Problems					
		Cmax			Nbr_IT+			Cmax			Nbr_IT+		
	n1	Avg	Min	Max	"0"	"1"	"2+"	Avg	Min	Max	"0"	"1"	"2+"
MIP1	0.5 n	0.00	0.00	0.00	15	8	7	12.70	3.01	37.05	15	11	4
MIP2		0.00	0.00	0.00	30	0	0	15.10	10.80	41.23	30	0	0
MIP3		**0.00**	**0.00**	**0.00**	**30**	**0**	**0**	**13.22**	**4.70**	**37.05**	**21**	**9**	**0**
MIP1	0.6 n	0.00	0.00	0.00	15	9	6	11.84	0.00	33.57	15	11	4
MIP2		0.00	0.00	0.00	30	0	0	14.70	0.00	38.70	30	0	0
MIP3		**0.00**	**0.00**	**0.00**	**30**	**0**	**0**	**12.91**	**0.00**	**33.57**	**23**	**7**	**0**
MIP1	0.7 n	0.00	0.00	0.00	15	10	5	13.26	3.54	26.11	9	12	9
MIP2		0.00	0.00	0.00	30	0	0	14.20	9.30	34.23	30	0	0
MIP3		**0.00**	**0.00**	**0.00**	**30**	**0**	**30**	**13.94**	**6.70**	**26.11**	**25**	**4**	**1**

TABLE 10. **MIP's results for family 2 - Class 2 instances.**

		10 job's's Problems						20 job Problems					
		Cmax			Nbr_IT+			Cmax			Nbr_IT+		
	n1	Avg	Min	Max	"0"	"1"	"2+"	Avg	Min	Max	"0"	"1"	"2+"
MIP1	0.5n	0.00	0.00	0.00	19	10	1	3.05	0.00	7.17	16	13	1
MIP2		0.00	0.00	0.00	30	0	0	6.21	1.32	8.63	29	1	0
MIP3		**0.00**	**0.00**	**0.00**	**30**	**0**	**0**	**3.05**	**0.00**	**7.17**	**20**	**10**	**0**
MIP1	0.6n	0.00	0.00	0.00	13	15	2	3.89	0.19	13.41	0	7	23
MIP2		0.00	0.00	0.00	30	0	0	8.28	1.54	16.32	30	0	0
MIP3		**0.00**	**0.00**	**0.00**	**30**	**0**	**0**	**4.82**	**0.20**	**14.56**	**21**	**6**	**3**
MIP1	0.7n	0.00	0.00	0.00	9	18	3	5.88	0.00	11.44	0	4	26
MIP2		0.00	0.00	0.00	30	0	0	7.32	1.94	12.83	29	1	0
MIP3		**0.00**	**0.00**	**0.00**	**30**	**0**	**0**	**6.10**	**0.00**	**11.67**	**27**	**3**	**0**

ARD was defined by the following formula:

$$ARD = \frac{\sum_{i=1}^{30} RD_i}{30}$$

For the 10 job problems, both MIP2 and MIP3 generated permutations corresponding to the optimal solutions in terms of Nbr_IT+ for 59/60 instances. Furthermore, for MIP3, the generated permutations also corresponded to optimal Cmax solutions. This is expected, as the number of parallel machines is equal as the number of jobs.

For the 20 job problems, both MIP2 and MIP3 generate optimal solutions in terms of Nbr_IT+. The Cmax generated by MIP3 was better than that generated by MIP2 for all the instances.

Tables 9 to 11 present the results for family 2 of the instances, with its three classes. For these tables, column 2 represents the number of jobs affected to M1 in

TABLE 11. MIP's results for family 2 - Class 3 instances.

		10 jobs						20 jobs					
		Cmax			Nbr_IT+			Cmax			Nbr_IT+		
		Avg	Min	Max	"0"	"1"	"2+"	Avg	Min	Max	"0"	"1"	"2+"
MIP1	0.5 n	0.00	0.00	0.00	25	5	0	4.39	0.20	8.20	3	17	10
MIP2		0.00	0.00	0.00	30	0	0	6.31	1.62	11.86	25	5	0
MIP3		**0.00**	**0.00**	**0.00**	**30**	**0**	**0**	**4.45**	**0.20**	**8.84**	**22**	**8**	**0**
MIP1	0.6 n	0.00	0.00	0.00	12	18	0	1.85	0.00	4.55	0	4	26
MIP2		0.00	0.00	0.00	30	0	0	3.21	0.00	8.97	28	2	0
MIP3		**0.00**	**0.00**	**0.00**	**30**	**0**	**0**	**1.91**	**0.00**	**4.82**	**30**	**0**	**0**
MIP1	0.7 n	0.00	0.00	0.00	12	17	1	3.23	0.00	11.77	1	4	25
MIP2		0.00	0.00	0.00	30	0	0	5.50	1.61	11.67	23	7	0
MIP3		**0.00**	**0.00**	**0.00**	**30**	**0**	**0**	**4.01**	**0.00**	**13.21**	**25**	**0**	**5**

TABLE 12. Comparison of instances with zero material waste.

	Nbr_IT+ "0"		
Class	MIP1	MIP2	**MIP3**
1	39/90	90/90	**69/90**
2	16/90	88/90	**68/90**
3	4/90	76/90	**77/90**

TABLE 13. Comparison of ARD – Cmax values between MIP1 and MIP3.

	Class1 - ARD		Class 2 -ARD		Class3 - ARD	
n1	MIP1	**MIP3**	MIP1	**MIP3**	MIP1	**MIP3**
0.5 n	12.7	**13.2**	3.0	**3.0**	4.4	**4.5**
0.6 n	11.5	**12.9**	3.9	**4.8**	1.8	**1.9**
0.7 n	13.3	**13.9**	5.9	**6.1**	3.2	**4.0**

S1. We observe that for all classes, in the case of the 10 job problems, both MIP2 and MIP3 generate optimal Nbr_IT+, with better Cmax values for MIP3. This is true for three classes of instances.

We recall that the initial objective of MIP3 is to produce optimal or near-optimal results for material waste, with acceptable Cmax values. For the 20 jobs' problems, MIP2 generates optimal Nbr_IT+ values most of the time, and MIP3 generates optimal solutions 2/3 of the time in the worst case (Table 12).

The results generated by MIP3 for the Nbr_IT+ objective are much better than those generated by MIP1 (Table 12), while the decrease in the value of Cmax is not that important, as summarized in Table 13.

It is clear that MIP3 generates much better results from an industrial perspective, mainly when production is for the stock. In this case, a small increase in the completion time is not that important. It is worth mentioning that the results presented above are based on single solutions generated by the solver. Because we know that a MIP could have more than one optimal solution, we need to keep in mind that there could exist other optimal solutions, with better values of Nbr_IT+, for MIP1, and solutions with better values of Cmax for MIP2. In the following we will show experimentally how the GA can overcome this situation.

C. RESULTS GENERATED BY THE GAs

Tables 14 to 17 compare the results generated by GA-C-I to those generated by GA-I-C. It should be noted that the stopping criterion is the number of generations without improvement, which is fixed to 10, in addition to the running time. The running time is fixed to 60 s for 10 and 20 jobs' problems, 120 s for 50 jobs' problems, and 180 s for 100 and 200 jobs' problems.

1) For family 1 of instances, up to 10 jobs' problems, Nbr_IT+ is relatively high for GA-C-I, while the completion times are slightly better than GA-I-C. For problems with 100 jobs' and 200 jobs', Nbr_IT+ is high for both GA-C-I and GA-I-C.
   It should be mentioned that, in the case of 200 jobs, the number of Nbr_IT+ that exceeds 2 occurs with 4/30 instances only.
2) For the second family of instances, the solutions generated by both GAs are the same for the 10 jobs' problems. In fact, an optimal solution is generated for all instances. This is true for three classes of instances.

For the rest of the problems, the optimal solutions are generated by GA-I-C for the first objective (zero Nbr_IT+), for all instances. However, we observe a decrease in the quality of Cmax most of the time. GA-C-I generated slightly better makespans at the price of more Nbr_IT+ (more material waste).

The ARD is slightly different for the two algorithms when the number of jobs is less than 50. For the remaining problems, the GA-C-I generated somewhat better makespans.

TABLE 14. GA results for family 1.

	GA-C-I						*GA-I-C*					
n	*Cmax*			*Nbr_IT+*			*Cmax*			*Nbr_IT+*		
	Avg	Min	Max	0	1	>=2	Avg	Min	Max	0	1	>=2
10	0.00	0.00	0.00	30	0	0	0.00	0.00	0.00	30	0	0
20	2.07	0.00	14.25	4	11	15	2.21	0.00	14.75	30	0	0
50	1.48	0.00	12.12	5	3	22	1.54	0.00	11.80	30	0	0
100	1.86	0.17	11.57	1	1	28	2.00	0.17	12.33	22	5	3
200	2.24	0.27	7.80	0	0	30	3.11	0.43	10.15	3	6	21*

TABLE 15. GA results for family 2 - Class 1 instances.

		GA-C-I						GA-I-C					
		Cmax			Nbr_IT+			Cmax			Nbr_IT+		
n	n1	Avg	Min	Max	"0"	"1"	"2+"	Avg	Min	Max	"0"	"1"	"2+"
10	0.5 n	0.00	0.00	0.00	30	0	0	0.00	0.00	0.00	30	0	0
20		5.21	0.00	13.4	28	2	0	5.01	0.00	12.58	30	0	0
50		6.07	0.00	18.5	28	1	1	6.02	0.00	17.91	30	0	0
100		6.07	2.08	13.4	24	4	2	6.17	2.32	14.07	30	0	0
200		6.54	2.59	11.6	20	6	4	6.87	3.39	13.22	30	0	0
10	0.6 n	0.00	0.00	0.00	30	0	0	0.00	0.00	0.00	30	0	0
20		4.44	0.00	13.5	27	2	1	4.5	0.00	13.52	30	0	0
50		3.32	0.00	8.72	27	1	2	3.57	0.00	8.92	30	0	0
100		2.77	0.74	8.64	26	1	3	3.25	1.02	8.53	30	0	0
200		3.02	0.66	5.38	23	4	3	4.15	0.42	5.98	30	0	0
10	0.7 n	0.00	0.00	0.00	30	0	0	0.00	0.00	0.00	30	0	0
20		5.01	0.00	20.2	30	0	0	5.01	0.00	20.17	30	0	0
50		2.49	0.89	4.96	30	0	0	2.49	0.89	4.96	30	0	0
100		2.33	0.76	6.31	25	3	2	2.37	0.76	7.14	30	0	0
200		2.01	0.89	4.77	23	5	2	2.93	1.08	7.08	30	0	0

For the instances from the class 2 problem, the observations are similar to those of class 1, except that for the 200 jobs' problems, the GA-I-C did not generate zero Nbr_IT+ for approximately one third of the instances. For some instances, this number exceeded two Nbr_IT+.

For class 3 instances, the decrease in the performance of GA-I-C is observable starting from the problems of 50 jobs. This decrease in performance becomes important as the size of the problem increases. For the 200 jobs' problems, with both GAs, there is more than one Nbr_IT+ most of the time; for GA-I-C, and more than two Nbr_IT+ all the time with GA-C-I. The ARD is nearly the same for the two algorithms when the number of jobs is less than 50 for the remaining problems. GA-C-I resulted in a slightly better Cmax. This will be explained in the last section.

It should be mentioned that, for 100 jobs' problems, only 3/30 instances have a Nbr_IT+ that exceeds 2, for 200 jobs' problems, only 6/30 instances have a Nbr_IT+ exceeding 2. For class 3 instances, the decrease in the performance of GA-I-C is observable starting from the 50 jobs' problems. This decrease in performance becomes important as the size of the problem increases. For the 200 jobs' problems, with both GAs, there are more than one Nbr_IT+ most of the time; for GA-I-C, and more than two Nbr_IT+ all the time with GA-C-I. The Cmax ARD is almost the same for the two algorithms when the number of jobs is less than 50 for the remaining problems. The GA-C-I resulted in a slightly better Cmax. This is explained in the last section.

In the following, we propose to compare and understand the performances of the different classes of problems in terms of Nbr_IT+, mainly for GA-I-C.

It appears from this table that the performance of the algorithm with Class1 instances is better, whether there is an equilibrium of charge, in terms of number of jobs, on S1

TABLE 16. GA results for family 2 - Class 2 instances.

		GA-C-I						GA-I-C					
		Cmax			Nbr_IT+			Cmax			Nbr_IT+		
n	n1	Avg	Min	Max	"0"	"1"	"2+"	Avg	Min	Max	"0"	"1"	"2+"
10	0.5n	0.00	0.00	0.00	30	0	0	0.00	0.00	0.00	30	0	0
20		0.64	0.00	2.15	28	2	0	0.64	0.00	2.15	30	0	0
50		1.68	0.00	4.75	26	4	0	1.68	0.00	4.75	30	0	0
100		2.74	0.61	6.48	25	3	2	2.74	0.61	6.48	30	0	0
200		4.06	1.63	7.36	2	1	27	7.68	2.87	27.4	18	7	5
10	0.6n	0.00	0.00	0.00	30	0	0	0.00	0.00	0.00	30	0	0
20		0.47	0.00	2.03	26	3	1	0.47	0.00	2.03	30	0	0
50		0.32	0.00	1.27	23	6	1	0.32	0.00	1.27	30	0	0
100		0.92	0.10	2.99	23	7	0	0.92	0.10	2.99	30	0	0
200		2.03	0.52	4.52	1	1	28	8.3	0.52	4.52	21	5	4
10	0.7n	0.00	0.00	0.00	30	0	0	0.00	0.00	0.00	30	0	0
20		0.52	0.00	1.56	28	2	0	0.52	0.00	1.56	30	0	0
50		0.26	0.00	0.89	22	6	2	0.26	0.00	0.89	30	0	0
100		0.68	0.00	2.18	19	9	2	0.68	0.00	2.18	30	0	0
200		1.05	0.21	2.16	1	3	26	1.05	0.21	2.16	20	7	3

TABLE 17. GA results for the Class 3 instances.

		GA-C-I						GA-I-C					
		Cmax			Nbr_IT+			Cmax			Nbr_IT+		
n	n1	Avg	Min	Max	"0"	"1"	"2+"	Avg	Min	Max	"0"	"1"	"2+"
10	0.5n	0.00	0.00	0.00	30	0	0	0.00	0.00	0.00	30	0	0
20		1.84	0.00	4.77	30	0	0	1.84	0.00	4.77	30	0	0
50		6.59	4.39	10.37	21	8	1	6.59	4.39	10.37	13	14	3
100		9.51	7.60	11.84	0	0	30	9.51	7.60	11.84	3	12	15*
200		11.11	8.84	13.17	1	0	29	11.11	8.84	13.17	1	1	28*
10	0.6n	0.00	0.00	0.00	30	0	0	0.00	0.00	0.00	30	0	0
20		0.11	0.00	1.03	30	0	0	0.11	0.00	1.03	30	0	0
50		2.19	0.40	4.15	20	6	4	2.19	0.40	4.15	27	3	0
100		5.34	3.52	7.32	1	0	29	5.34	4.21	13.51	10	13	7
200		7.27	6.35	9.04	0	0	30	7.27	6.35	9.04	0	5	25
10	0.7n	0.00	0.00	0.00	30	0	0	0.00	0.00	0.00	30	0	0
20		0.18	0.00	1.15	21	8	1	0.18	0.00	1.15	30	0	0
50		0.85	0.00	2.64	4	12	14	0.85	0.00	2.64	30	0	0
100		2.74	1.20	4.30	0	3	27	2.74	1.20	4.30	29	1	0
200		4.44	3.35	6.16	0	0	30	4.44	3.35	6.16	0	16	14

machines or not. This could be explained by the fact that for class 1, there were some overlaps between the intervals of processing times for both stages. This means that there is a chance to encounter more jobs that are processed on S2 directly after completing (or within the lag+) on S1. Thus, there is no need to delay the launch of subsequent jobs on S1. In fact, it is this delay that results on situations where $IT \geq \theta$.

D. EVALUATION OF MATERIAL WASTE

As mentioned above, material waste is caused by an idle time that exceeds the perishability limit. In fact, in one of the pasta

TABLE 18. GA-I-C performances for the Nbr_IT+.

n1	0.5 n			0.6 n			0.7 n		
n	Class 1	Class 2	**Class 3**	Class 1	Class 2	**Class 3**	Class 1	Class 2	**Class 3**
10	30	30	**30**	30	30	**30**	30	30	**30**
20	30	30	**30**	30	30	**30**	30	30	**30**
50	30	30	**13**	30	30	**30**	30	27	**30**
100	30	30	**3**	30	30	**30**	30	10	**29**
200	30	18	**1**	30	21	**20**	30	0	**0**

TABLE 19. The waste quantities found by GA-C-I and GA-I-C algorithms.

The waste quantities found by GA-C-I and GA-I-C		
n	GA-C-I	GA-I-C (best)
10	00.00	00.00
20	75.11	00.00
50	168.20	00.00
100	311.60	21.44

production lines, it is necessary to clean the machine and remove the material stuck in it if its idle time exceeds 30 mn. The amount of removed pasta or material was estimated to be 50 kg. This quantity cannot be recycled or used for production purposes. During the full production period, the total waste (TW) is estimated as follows:

$$TW = 50 \times Nbr\_IT+$$

Table 19 presents the results presented in Table 14 in terms of the quantity of material waste.

We observe that with the GA-I-C algorithm, zero waste was realized for all instances (90 instances) corresponding to problems of small and medium size (10, 20, and 50 jobs). It is worth mentioning that a 50 jobs 'instance corresponds to one week of production. This objective was achieved while maintaining a satisfactory makespan. It should be noted that GA-C-I causes a quantity of waste between 75 kg and 168 kg for large instances (100 and 200 jobs). GA-I-C maintains the average amounts of waste low (less than 100 kg) compared to those obtained by GA-C-I (more than 300 kg). From an industrial point of view, the results found by GA-I-C seem more interesting than those found by GA-C-I because the wasted material is divided by three while maintaining a satisfactory production time.

VII. CONCLUSION

The configuration presented in this paper that distinguishes this work from other studies is the objective of minimizing waste resulting from the occurrences of machine idle times exceeding the perishability limit (Nbr_IT+) while simultaneously considering several temporal constraints. A MIP incorporating the new objective and two hybrid GAs was proposed. The MIP generated better results than the previous ones without a significant increase in Cmax. Within the proposed GAs, we incorporated many priority rules and an iterative local search, with a filtering function for both Nbr_IT+ and Cmax. The effectiveness of the proposed approach was validated through a numerical analysis. The results indicate that the hybrid GA generates either optimal or near-optimal solutions. The generated solutions satisfied the industrial expectations as material waste was very limited or null, with acceptable completion times. Future research may be conducted to consider industrial objectives, especially those related to the due date, such as total tardiness. Another future improvement direction may be the use of other resolution methods, such as tree-based heuristics or constraint satisfaction optimization.

REFERENCES

[1] H. Harbaoui, S. Khalfallah, and O. Bellenguez-Morineau, "A case study of a hybrid flow shop with no-wait and limited idle time to minimize material waste," in *Proc. IEEE 15th Int. Symp. Intell. Syst. Informat. (SISY)*, Sep. 2017, pp. 000207–000212.

[2] C. Le Hesran, A.-L. Ladier, V. Botta-Genoulaz, and V. Laforest, "Operations scheduling for waste minimization: A review," *J. Cleaner Prod.*, vol. 206, pp. 211–226, Jan. 2019.

[3] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. R. Kan, "Optimization and approximation in deterministic sequencing and scheduling: A survey," *Ann. Math.*, vol. 5, pp. 287–326, 1979.

[4] J. Yang, "Hybrid flow shop with parallel machines at the first stage and dedicated machines at the second stage," *Ind. Eng. Manage. Syst.*, vol. 14, no. 1, pp. 22–31, Mar. 2015.

[5] I. Ribas, R. Leisten, and J. M. Framiñan, "Review and classification of hybrid flow shop scheduling problems from a production system and a solutions procedure perspective," *Comput. Oper. Res.* vol. 37, no. 8, pp. 1439–1454, 2010.

[6] H. Harbaoui, O. Bellenguez- Morineau, and S. Khalfallah, "Scheduling a two-stage hybrid flow shop with dedicated machines, time lags and sequence-dependent family setup times," in *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC)*, Oct. 2016, pp. 002990–002995.

[7] H. Harbaoui, S. Khalfallah, and O. Bellenguez-Morineau, "A novel hybrid GA for the assignment of jobs to machines in a complex hybrid flow shop problem," in *Proc. Int. Conf. Intell. Syst. Design Appl.*, 2017, pp. 640–649.

[8] R. Ruiz and A. Allahverdi, "New heuristics for no-wait flow shops with a linear combination of makespan and maximum lateness," *Int. J. Prod. Res.*, vol. 47, no. 20, pp. 5717–5738, Oct. 2009.

[9] Y. Goncharov and S. Sevastyanov, "The flow shop problem with no-idle constraints: A review and approximation," *Eur. J. Oper. Res.*, vol. 196, no. 2, pp. 450–456, Jul. 2009.

[10] I. Adiri and D. Pohoryles, "Flowshop/no-idle or no-wait scheduling to minimize the sum of completion times," *Nav. Res. Logistics Quart.*, vol. 29, no. 3, pp. 495–504, 1982.

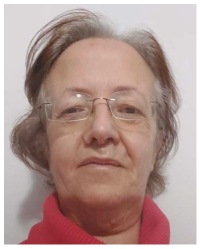
[11] P. Kalczynski and J. Kamburowski, "On the NEH heuristic for minimizing the makespan in permutation flow shops," *Omega*, vol. 35, no. 1, pp. 53–60, 2007.

[12] D. Baraz and G. Mosheiov, "A note on a greedy heuristic for flow-shop makespan minimization with no machine idle-time," *Eur. J. Oper. Res.*, vol. 184, no. 2, pp. 810–813, Jan. 2008.

[13] W. Shao, D. Pi, and Z. Shao, "Memetic algorithm with node and edge histogram for no-idle flow shop scheduling problem to minimize the makespan criterion," *Appl. Soft Comput.*, vol. 54, pp. 164–182, May 2017.

[14] S. Wang, X. Wang, and L. Yu, "Two-stage no-wait hybrid flow-shop scheduling with sequence-dependent setup times," *Int. J. Syst. Sci., Oper. Logistics*, vol. 7, no. 3, pp. 291–307, Jul. 2020.
[15] A. Khare and S. Agrawal, "Scheduling hybrid flowshop with sequence-dependent setup times and due Windows to minimize total weighted earliness and tardiness," *Comput. Ind. Eng.*, vol. 135, pp. 780–792, Sep. 2019.
[16] S. K. Dewi, D. M. Utama, and R. N. Rohman, "Minimize waste on production process using lean concept," in *Proc. J. Phys., Conf.*, 2021, vol. 1764, no. 1, pp. 012–201.
[17] C. Le Hesran, A. Ladier, V. Laforest, and V. Botta-Genoulaz, "Using flow assessment to identify a scheduling problem with waste reduction concerns: A case study," in *Proc. 6th Int. EurOMA Sustain. Oper. Supply Chains Forum*, Gothenburg, Sweden, Mar. 2019.
[18] G. P. Georgiadis, G. M. Kopanos, A. Karkaris, H. Ksafopoulos, and M. C. Georgiadis, "Optimal production scheduling in the dairy industries," *Ind. Eng. Chem. Res.*, vol. 58, no. 16, pp. 6537–6550, Apr. 2019.
[19] H. Harbaoui, "Ordonnancement d'un système de production industriel complexe: Flow shop hybride avec des machines dédiées soumis à différentes contraintes temporelles," Ph.D. dissertation, Ecole Nationale Supérieure Mines-Télécom Atlantique; Inst. Supérieur d, Nantes, France, 2018.
[20] C. R. Reeves, "A genetic algorithm for flowshop sequencing," *Comput. Oper. Res.*, vol. 22, no. 1, pp. 5–13, 1995.
[21] R. Ruiz and C. Maroto, "A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility," *Eur. J. Oper. Res.*, vol. 169, no. 3, pp. 781–800, 2006.
[22] C. Kahraman, O. Engin, İ. Kaya, and M. Kerim Yilmaz, "An application of effective genetic algorithms for solving hybrid flow shop scheduling problems," *Int. J. Comput. Intell. Syst.*, vol. 1, no. 2, pp. 134–147, May 2008.
[23] F. Jolai, S. Sheikh, M. Rabbani, and B. Karimi, "A genetic algorithm for solving no-wait flexible flow lines with due window and job rejection," *Int. J. Adv. Manuf. Technol.*, vol. 42, nos. 5–6, pp. 523–532, May 2009.
[24] W. Besbes, T. Loukil, and J. Teghem, "A two-stage flow shop with parallel dedicated machines," in *Proc. 8th Int. Conf. Modeling Simulation (MOSIM)*, 2010, pp. 1–9.
[25] A. Sioud, M. Gravel, and C. Gagné, "A hybrid genetic algorithm for the single machine scheduling problem with sequence-dependent setup times," *Comput. Oper. Res.*, vol. 39, no. 10, pp. 2415–2424, Oct. 2012.
[26] A. Sioud, M. Gravel, and M. Gagné, "A genetic algorithm for solving a hybrid flexible flowshop with sequence dependent setup times," in *Proc. IEEE Congr. Evol. Comput.*, Cancun, Mexico, vol. 40, Jun. 2013, pp. 1064–1075.
[27] S. Wang and M. Liu, "A genetic algorithm for two-stage no-wait hybrid flow shop scheduling problem," *Comput. Oper. Res.*, vol. 40, no. 4, pp. 1064–1075, Apr. 2013.
[28] D. E. Goldberg and K. A. Deb, "A comparative analysis of selection schemes used in genetic algorithms," *Found. Genetic Algorithms*, vol. 1, pp. 69–93, Dec. 1991.
[29] *Sipina–Arbres de Décision–Data Mining*. Accessed: Jan. 24, 2022. [Online]. Available: http://sipina-arbres-de-decision.blogspot.com/
[30] S. Wang and M. Liu, "A heuristic method for two-stage hybrid flow shop with dedicated machines," *Comput. Oper. Res.*, vol. 40, no. 1, pp. 438–450, Jan. 2013.

HOUDA HARBAOUI received the master's degree in management IT from the High Institute of Management of Sousse, University of Sousse, Tunisia, and the Ph.D. degree in computer science from IMT Atlantique, Nantes, France, and the University of Sousse. Her main research interests include operations research, evolutionary computation, optimization, metaheuristics, and decision support systems.



SOULEF KHALFALLAH received the M.S. degree in industrial engineering from Louisiana State University, Baton Rouge, LA, USA, in 1987, and the Ph.D. degree in industrial engineering from École Centrale Paris, France, in 2003. She worked at the Municipality of Sousse, Tunisia, and the Ministry of Research, Tunisia, from 1987 to 2003, before joining as a Faculty Member at the School of Management of Sousse, Tunisia, as an Assistant Professor. She is currently a Researcher at the MARS Research Laboratory, University of Souse. She has supervised many research works and published several research articles in the area of scheduling and data envelopment analysis.