Contents lists available at GrowingScience

Journal of Project Management

homepage: www.GrowingScience.com/jpm

Solving a flow shop scheduling problem with missing operations in an Industry 4.0 production environment

Daniel Alejandro Rossit[a,b*], Adrián Toncovich[a], Diego Gabriel Rossit[a,b] and Sergio Nesmachnow[c]

[a]*Departamento de Ingeniería, Universidad Nacional del Sur, Av. Alem 1253, Bahía Blanca, Buenos Aires CP 8000, Argentina*
[b]*INMABB UNS CONICET, Departamento de Matemática, Av. Alem 1253, Bahía Blanca, Buenos Aires CP 8000, Argentina*
[c]*Facultad de Ingeniería, Universidad de la República, Julio Herrera y Reissig 565, Montevideo 11300, Uruguay*

C H R O N I C L E

Article history:
Received: July 21, 2020
Received in revised format: September 10 2020
Accepted: September 19, 2020
Available online: October 15, 2020

Keywords:
Non-Permutation Flow Shop
Total Tardiness
Industry 4.0
Missing Operation
Genetic Algorithm
Simulated Annealing
Scheduling

A B S T R A C T

Industry 4.0 is a modern approach that aims at enhancing the connectivity between the different stages of the production process and the requirements of consumers. This paper addresses a relevant problem for both Industry 4.0 and flow shop literature: the missing operations flow shop scheduling problem. In general, in order to reduce the computational effort required to solve flow shop scheduling problems only permutation schedules (PFS) are considered, i.e., the same job sequence is used for all the machines involved. However, considering only PFS is not a constraint that is based on the real-world conditions of the industrial environments, and it is only a simplification strategy used frequently in the literature. Moreover, non-permutation (NPFS) orderings may be used for most of the real flow shop systems, i.e., different job schedules can be used for different machines in the production line, since NPFS solutions usually outperform the PFS ones. In this work, a novel mathematical formulation to minimize total tardiness and a resolution method, which considers both PFS and (the more computationally expensive) NPFS solutions, are presented to solve the flow shop scheduling problem with missing operations. The solution approach has two stages. First, a Genetic Algorithm, which only considers PFS solutions, is applied to solve the scheduling problem. The resulting solution is then improved in the second stage by means of a Simulated Annealing algorithm that expands the search space by considering NPFS solutions. The experimental tests were performed on a set of instances considering varying proportions of missing operations, as it is usual in the Industry 4.0 production environment. The results show that NPFS solutions clearly outperform PFS solutions for this problem.

1. Introduction

The Industry 4.0 conceptual model enhances connectivity among the different components of the production systems, as well as among the components and the decision-making centers (Hermann et al., 2016; Zhong et al., 2017). This allows production planners to manage information obtained in real time from the "shop floor" (Monostori, 2014; Rossit et al., 2019a). This communication increases the flexibility of the production system, since the machines have the capacity to communicate with each other and with the decision-making centers (Zhong et al., 2017; Rossit & Tohmé, 2018), allowing designs of innovative products and the development of personalization capabilities. Moreover, given the increased level of digitalization and with the help of IIoT (Industrial Internet of Things), now it is possible to achieve the business strategy of mass customization of products (i.e., the products can be adapted to individual customer preferences). In other words, this means that for a given product each client can choose among different combinations of options, configurations or characteristics in such a way that the final product is personalized (Zheng et al., 2019). However, in order to be able to produce these personalized products, it is important to implement an efficient sequencing approach to accommodate the specifications of each customer in these Industry 4.0 production environments (Dolgui et al., 2019). This constitutes the motivation for addressing this sequencing problem in the present work.

* Corresponding author.

E-mail address: daniel.rossit@uns.edu.ar (D. A. Rossit)

doi: 10.5267/j.jpm.2020.10.001

Currently, one of the most widespread and efficient production configurations is the manufacturing cell. The manufacturing cell represents a set of production resources or machines that are grouped by the type of products they produce and organized in sequential mode (Rajendran, 1994). The output of a machine in the cell is the input for the next machine, and this happens with each machine of the cell. Generally, this type of configuration is included within the flow shop scheduling family of problems (Rajendran & Ziegler, 2001; Henneberg & Neufeld, 2016). Product customization powered by Industry 4.0 will have a significant impact on the scheduling function. Given the freedom provided to the clients when customizing their products, it is very likely that, when responding to a client's product order, the operations required to produce this product imply different uses of the machines in the cell. Thus, some operations included in the cell design may not be required to meet the specifications requested by some clients (Tseng et al., 2008). The design of manufacturing cells is aimed at grouping operations together that allow, for example, producing a given family of products within a certain range of variation. This arrangement can provide flexibility and a better service to the clients (Pugazhendi et al., 2003). However, when the client selects the options for the product and defines its specifications, it is very likely that he or she does not use all the available options. Therefore, there are operations that must not be performed, and some jobs will skip them, which gives rise to the problem of flow shop scheduling with missing operations (Glass et al., 1999; Pugazhendi et al., 2004a; Dios et al., 2018).

Several flow shop scheduling problems with missing operations in different production environments have been addressed in the related literature (Ribas et al., 2010). In the vast majority of these works, the problems are solved by considering only permutation sequences (PFS) (Venkataramanaiah, 2008), meaning that each machine processes all the jobs in the same order. In terms of the size of search space, this implies finding the best possible solution in a universe with $n!$ possible sequences, where n is the number of jobs to be processed. However, the permutation schedules are not required as a technological condition or restriction in the manufacturing process, since, in general, each machine can process jobs in any order. Another approach to solve this problem involves relaxing the permutation schedule condition, which requires solving the problem under the non-permutation approach (NPFS) (Rossit et al., 2020). In this approach, the job sequence for each machine can be modified, increasing the size of the search space considerably from n to $(n!)^m$ solutions, where m is the number of machines. Since PFS solutions are comprised in the set of NPFS solutions as a particular case, NPFS solutions are at least as efficient as PFS, and improvements have been obtained using different objective measures (Pugazhendi et al., 2003; Tseng et al., 2008; Benavides & Ritt, 2016; Benavides & Ritt, 2018; Rossit et al., 2019b). Despite the industrial significance of this problem, NPFS has not yet been addressed for systems that possess Industry 4.0 production characteristics (Liu et al., 2018; Rossit et al., 2019c). Moreover, there is evidence in the relevant literature that NPFS solutions tend to outperform PFS solutions when considering objective functions related to due dates (Tseng et al., 2008), though the NPFS problem with missing operations considering such optimization criteria has not been studied yet (Rossit et al., 2018). Therefore, this work makes a contribution in that regard. For this purpose, this article introduces the Industry 4.0 environment and discusses the importance of considering problems with missing operations in this new scenario. An approach applying metaheuristics is proposed to solve both variants of the problem: Genetic Algorithm (GA) for PFS and Simulated Annealing (SA) for NPFS. The results reported in the literature for both problem variants indicate that NPFS solutions improves over PFS solutions, for problems with missing operations, considering the makespan as objective function. Improvements are larger than those obtained for regular flow shop (no missing operations considered) and makespan as objective function (Benavides & Ritt, 2016; Benavides & Ritt, 2018).

This work is organized as follows. In Section 2, we present briefly the related literature to show our contribution to fill the research gap. In Section 3, we describe in detail the addressed problem, and in Section 4 the solution approaches are introduced. Finally, in Section 5 we describe the experimental work undertaken, as well as the corresponding results.

2. Related Research Work

In this section, we analyze the works that are more directly related to the research proposed in this paper, considering the following topics: Scheduling in Industry 4.0 systems and NPFS with missing operations.

2.1. Scheduling in Industry 4.0

Due to the potential impact of Industry 4.0 technologies on production systems in recent years, the scientific community has focused its efforts on generating contributions that allow exploiting this potential. Many of those contributions consider the scheduling decision-making process in industries of the fourth industrial revolution. Evidence of this effort is the recent reviews published in the last two years (Dolgui et al., 2019; Ivanov et al., 2018; Liu et al., 2018; Uhlmann & Frazzon, 2018; Rossit et al., 2019c). Subsequently, the most outstanding works in flow shop systems are reviewed. To delve into other scheduling problems in Industry 4.0 (e.g., job shop problems) refer to the aforementioned revisions. Luo et al. (2015) analyze a hybrid flow shop scheduling problem where the jobs to be sequenced arrive dynamically and the work in the machines is not continuous. The information is provided in real time by RFID technologies. To solve the problem, the authors propose a multi-period hierarchical scheduling mechanism to divide the planning horizon into smaller time intervals. In this way, the decisions to be made by the shop floor (shorter time) and the stage manager (larger time horizon) are also divided hierarchically. In each level, the decision makers optimize their schedule permutatively. Wang et al. (2016) introduces a scheduling model based on Petri Net for manufacturing in a permutation IoT-enabled hybrid flow shop. In this way, they can address the problem of scheduling in real time. To solve the problem, they propose a modified Ant Colony Optimization algorithm, which controls pheromone levels to avoid stagnation in the solution search process. Shim et al. (2017) study a flexible permutation flow shop problem with sustainability considerations in Industry 4.0 and sequence-dependent setups. The authors propose heuristic algorithms based on dispatching rules and the economic lot size model to minimize the total tardiness of jobs. Framinan et al. (2017) study the impact of having information in real time to regenerate schedules in permutation flow shop configurations, trying to minimize the makespan. The study tries to quantify the advantages of collecting data in real time on the actual time of completion of the jobs in the shop to re-sequence the jobs that remain to be processed. Their results show that the effectiveness of Reschedule versus non Reschedule (maintaining the initial schedule) is highly dependent on the variability of the processing times. The greater the variability of the processing times, the lower the benefits of carrying out the Reschedule. The concept of this study was deepened and broadened by Framinan et al. (2019), and reschedule strategies were proposed that would allow information to be extracted in real time. Within the proposed strategies, the one that obtained the best performance uses the critical path as a rule of analysis. If the difference between the processing times obtained in real time and the standard or expected ones (the ones used to generate the initial schedule) affects the critical path, Rescheduling is performed; otherwise, it is not. Fu et al. (2018) studies a bi-objective permutation flow shop scheduling problem with time-dependent processing times and uncertainty in Industry 4.0-based manufacturing system. To solve the problem, they use a Fireworks algorithm, where specially designed operators are applied.

2.2. Non-permutation flow shop with missing operations

Next, we present the main contributions of studies on flow shop scheduling that consider production scenarios with missing or skipping operations and non-permutation solutions. This section allows us to frame our work in the context of the non-permutation flow shop with missing operation literature. The first work to analyze on flow shop problems with missing operations considering non-permutation solutions is Rajendran & Ziegler (2001). In this work, the authors minimized the total flow time using heuristics and dispatching rules. Pugazhendhi et al. (2003) address the same problem by proposing insertion heuristics designed to optimize total flow time. In this work, the authors address problems involving a maximum of 40 jobs and 50 machines. This study is deepened in Pugazhendhi et al. (2004 a) where they analyze non-permutation heuristics and dispatch rules against permutation solutions. In Pugazhendhi et al. (2004 b) a flow shop problem with missing operation considering sequence-dependent setup times is studied. In this work, they optimize the makespan and the total weighted flow time in a mono-objective way, solving instances of up to 30 jobs and 20 machines. Tseng et al. (2008) extend the analysis of missing operations to hybrid flow shops (where there is more than one machine per production stage). They analyze a stainless-steel factory and seek to optimize the makespan of the scheduling problem. The solutions considered in this work include non-permutation solutions through a heuristic that uses permutation solutions as input. Henneberg & Neufeld (2016) address the problem of flow shop with missing operations proposing an improved version of Pugazhendi et al. (2003) heuristics, and a two-step Simulated Annealing algorithm. The studies carried out by these authors include instances of up to 100 jobs and 30 machines.

2.3. Comments about the related research work

From the reviewed papers in Industry 4.0, we could not find articles that address flow shop systems with missing operations. While scheduling in Industry 4.0 is a subject that is taking its first steps (Liu et al., 2018; Rossit et al., 2019c), it is clear that it is necessary to study these problems considering the type of impact Industry 4.0 will have on manufacturing with the increasing degree of personalization of the products (Zhong et al., 2017). We will expand on this issue in the next section. Regarding the works that have studied flow shops with missing operations considering non-permutation solutions, these have shown the importance of including non-permutation solutions in the search process, allowing to improve the efficiency of the schedules (Rajendran & Ziegler 2001; Pugazhendi et al., 2003; Henneberg & Neufeld, 2016). However, all these works, at least as far as we are concerned, have worked only considering regular objective functions, e.g., makespan, total flowtime, etc. We have not found articles on flow shop with missing operations considering non-permutation solutions in systems that optimize criteria related to due-dates. For a deeper and wider state of the art review refer to the paper on NPFS by Rossit et al. (2018). The due-date related objectives are of great importance for systems that are oriented to generate a better service by offering mass customization of production, such as the ones considered in this work. Consequently, this work constitutes a contribution to both scheduling in Industry 4.0 production systems, as well as to non-permutation flow shop literature since it addresses a new problem.

3. Problem statement

In this section, we properly introduce the problem that we address in this paper. Firstly, we analyze the impact of Industry 4.0 in manufacturing cells, and how this influences the interaction with the customer. Then, we formalize our conception of the problem, relating our approach to the scheduling literature, and, finally, we model the problem considered in this work as mixed-integer programming formulations.

3.1. Industry 4.0 environment

Industry 4.0 proposes a digital transformation of traditional production systems (Zhong et al., 2017). This transformation is based on the implementation of cyber-physical systems (CPS), which integrate virtual and physical processes in the same system (Lee, 2008). In turn, through IoT the CPS can communicate with each other and with the Decision Support Systems, which in terms of production planning, allows directly linking the shop floor with the support systems for production decision making (Almada-Lobo, 2016; Rossit et al., 2019a). On the other hand, CPS allow generating a digital twin of the physical system (Lee et al., 2015), making it possible to analyze situations and making more informed decisions more agilely than in a traditional production system. This results in a much more flexible production system, better suited to different scenarios (Lu et al., 2019; Wang et al., 2019). In this way, the manufacturing system becomes a smart manufacturing system capable of providing an improved service to the customer, satisfying its needs in a personalized way (Yu et al., 2018; Rossit et al., 2019d). This allows to bring the final product closer to the true requirements of the client, who is actively involved in the design of his product (Zheng et al., 2017; Lu & Xu, 2019). The customer expresses his/her preferences by specifying different variants of a base product belonging to a given family of products (Vollmann et al., 2005), constructing what is known as a personalized variant of the product (Simpson et al., 2006). In this way, by making the production system more flexible, and making it smart, Industry 4.0 allows to offer the customer a much higher level of personalization than traditional production systems (Yao & Lin, 2016; Lu & Xu, 2019).

The problem addressed in our work responds to these conditions imposed by Industry 4.0 in manufacturing cell systems. Given that the focus of the problem is on providing the best possible level of service to the customer, it is appropriate to study scheduling performance metrics related to service level, such as total tardiness.

3.2. Scheduling in Manufacturing Cells: Mathematical Formulations

The problem considered in this work can be stated as finding the production schedule for a set of products that share production resources, with the feature that all products use the same resources respecting the same technological sequence, i.e., a flow shop scheduling problem. However, considering the scenario of intensive personalization promoted by Industry 4.0 (Zhong et al 2017; Wang et al., 2019), each product has its particularities, a fact that will imply a different use of resources, which is generally expressed in variable processing times. This difference in resource usage can even imply that the product directly does not use one of the resources of the manufacturing cell, which means that the product misses that operation (missing operations flow shop scheduling problem). Therefore, the objective is to schedule production in such a way that the highest level of service is achieved, complying with the due dates. To this end, we will seek to minimize the total tardiness related to the delivery of the products. On the other hand, this problem can be addressed by considering only permutation solutions (Permutation Flow Shop, PFS), or by considering non-permutation solutions (Non-Permutation Flow Shop, NPFS). In the first case, PFS, the solution search space includes sequences represented by the possible permutations of the *n* jobs, that is, a total of *n*! feasible solutions. Whereas if the job ordering is allowed to be modified for each stage of the flow shop, i.e. NPFS, the solution space size grows to $n!^m$, where *m* is the number of machines. These problems are identified in the classic notation for scheduling problems (Graham et al., 1979) as $F|prmu/missing|TTard$ and $F|missing|TTard$, respectively.

In the next section, the mixed integer programming model for the PFS is presented. This model is extended in Section 3.2.2. to allow the consideration NPFS solutions.

3.2.1. PFS model

Sets

Jobs *J* indexed by $\{j\}$, $j = 1,\ldots,n$

Machines *I* indexed by {i} , $i = 1,\ldots,m$

Parameters

$p_{i,j}$	Processing time of product *j* on machine *i*
Ω	large positive number
d_j	due date of job *j*

Variables

$C_{i,j}$	Completion time of job *j* on machine *i*
$x_{j',j}$	Binary variable: 1 if job *j'* is processed before job *j*
T_j	Tardiness of job *j*

$min \sum_j T_j$

subject to:

$$C_{i,j} \geq C_{i-1,j} + p_{i,j}\,, \qquad \forall j, i > 1 \tag{1}$$

$$C_{i,j} \geq C_{i,j'} + p_{i,j} - \left(1 - x_{j',j}\right) \cdot \Omega\,, \qquad \forall i, j' \neq j \tag{2}$$

$$C_{i,j'} \geq C_{i,j} + p_{i,j'} - x_{j',j} \cdot \Omega\,, \qquad \forall i, j' \neq j \tag{3}$$

$$x_{j',j} + x_{j,j'} = 1 , \qquad j \neq j' \tag{4}$$

$$T_j = max\{C_{m,j} - d_j; 0\} , \qquad \forall j \tag{5}$$

$$C_{i,j} > 0; \; x_{j,j'} \; \{0,1\} \tag{6}$$

The objective function to be minimized is the total tardiness, and it is defined as the sum of tardiness of all jobs. Constraint (1) forces the precedence of operations, i.e. a job must be completed on the current machine before it passes to the next one. Constraints (2) and (7) work together indicating the ordering of jobs. If job *j'* is processed before job *j*, then $x_{j',j}$ becomes 1 and constraint (2) becomes active, while constraint (3) turns redundant. In expression (4) the logical order is respected: if job *j'* is processed before job *j*, the converse cannot be valid. Constraint (5) defines the tardiness of each job *j*, which requires the comparison between the completion time of job *j* in the last machine *m* ($C_{m,j}$) with its corresponding due date d_j.While (6) enforces the non-negativity and binary conditions on the decision variables.

3.2.2. NPFS model

The NPFS model is similar to the PFS model, the main difference being that the job sequence may vary from machine to machine. For this, the variable *x* becomes now indexed by the set *m* of machines, as follows:

$x_{j',j,i}$ Binary variable: 1 if job *j'* is processed before job *j* on machine *m*

Then the equations that are modified are (6), (7) and (8) and we get,

$$C_{i,j} \geq C_{i,j'} + p_{i,j} - (1 - x_{j',j,i}) \cdot \Omega , \qquad \forall i, j' \neq j \tag{7}$$

$$C_{i,j'} \geq C_{i,j} + p_{i,j'} - x_{j',j,i} \cdot \Omega , \qquad \forall i, j' \neq j \tag{8}$$

$$x_{j',j,i} + x_{j,j',im} = 1 , \qquad j \neq j', \forall i \tag{9}$$

Here, constraints (6) and (8) are analogous to (2) and (3), but now the sequence of jobs may change at each machine of the system. Constraint (9) is a similar logical condition as (4) but now it is evaluated on every machine.

4. Solution approach

In NP-hard combinatorial optimization problems, as the one addressed is this work, where exact optimization methods require long execution times for solving realistic instances, metaheuristics are viable options to find good-quality approximate solutions (Ruiz & Stutzle 2008; Nesmachnow 2014; Framinan et al., 2019). For solving the problem at hand, a solution procedure that combines two metaheuristic approaches is used. Initially, a Genetic Algorithm is applied to solve the PFS problem with missing operations. Then, the resulting solution is improved in a second stage using a Simulated Annealing algorithm, which explores the problem as a NPFS.

The Genetic Algorithm is described in Section 4.1 and the Simulating Annealing algorithm is described in Section 4.2.

4.1. Genetic Algorithm

For solving the PFS problem presented in Section 3.2, a steady state Genetic Algorithm (ssGA) is proposed. ssGA is known to be a valid choice in problems where fitness evaluation is computationally expensive (Altiparmak et al., 2009) having been used in other flow shop problems (see, e.g., Kellegöz et al., 2010). The proposed algorithm was implemented in Java, by using JMETAL framework (Durillo et al., 2006) version 4.5.2.

Solution representation. Solutions are encoded as a permutation of integers of length equal to number of jobs *n*. Each index in the vector represents the processing order in the (first) machine and the corresponding integer value represents one of the jobs.

Initialization. The population of size *#P* is initialized by applying a random procedure to generate the permutations with a uniform distribution.

Genetic operators. The recombination operator is the Partially Mapped Crossover (PMX) applied over two selected individuals with probability p_c. The mutation operator is based on Swap Mutation and it interchanges two elements of the permutation. Mutation is applied to an individual with probability p_m. The proposed operators guarantee the feasibility of the solution.

Selection, replacement, and fitness assignment. Tournament selection is applied, with tournament size of two solution representations. The tournament criteria is based on the fitness, and if two compared individuals have the same fitness, any of them is chosen with probability 0.5. The new individual replaces the worst individual in the population if it has a better fitness.

Parameters setting. The parametrization was performed with a statistical analysis by varying three main parameters: population size #P, with values 100 and 200, crossover probability p_c and mutation probability p_m. The values considered were 100 and 200 for #P; 0.7, 0.8, 0.9 and 1 for p_c; and 0.05 and 1 for p_m. The maximum number of evaluations used for the parametrization was 5000. The parameter setting analysis was made over three instances of size n =15 and m = 20, different from the scenarios used for computational experimentation of Section 5. For each instance and each parametric combination (#P, p_c, and p_m), 30 independent runs where performed. Shapiro-Wilk test was applied to assess if the fitness results follow a normal distribution. Since several of the runs did not adjust to normal distribution, the medians were analyzed and the selected parametric combination was #P = 100, p_c = 0.7 and p_m = 0.05.

4.2. Simulated Annealing algorithm

This section describes the simulated annealing algorithm used to solve the NPFS problem variant. Simulated Annealing is a local search-based method that was developed from an analogy with the phenomenon of annealing (Kirkpatrick et al., 1983) to solve complex optimization problems. Local search methods look for the solution with the best value of the chosen criterion in the neighborhood of the current solution, accept it as the current solution, and repeat this step until it is not possible to improve the solution in the explored neighborhood. In general, by systematically applying this procedure a local optimum for the problem is obtained. To avoid getting trapped at a local optimum, a diversifying mechanism should be incorporated with the aim of exploring the entire solution space. In the simulated annealing metaheuristic, the diversifying strategy allows moves, with a certain probability, towards solutions that worsen the current value of the objective function. SA has shown capability of handling regular flow shop environments (Osman & Potts, 1989; Low, 2005; Vahedi Nouri et al., 2013), and particularly the NPFS with missing operations (Henneberg & Neufeld 2016). In a minimization problem, the simulated annealing algorithm evolves from one candidate solution to the next, considering the behavior of the objective function value following to the procedure described below:

1. If the new randomly generated candidate solution (S_C) in the neighborhood of the current solution ($S_C \in V(S_A)$] has an objective function value, z (S_C), better than the current z (S_A), the candidate solution is accepted as the current solution.
2. On the other hand, if the candidate solution has a higher objective function value than the current one, a probabilistic test known as the Metropolis criterion (Metropolis et al., 1953) is used to determine the probability of acceptance of a relatively lower quality solution. This test establishes that this solution can be accepted with a certain probability, $P(A) = e^{-\Delta z/T}$, that decreases as a function of the increase in the difference between the objective function values of both solutions, Δz. T is the control parameter that simulates the role of the temperature in the physical process of annealing. If the candidate solution is not accepted, another sequence is randomly selected and the procedure is repeated until the stopping criteria are met.

The classic parameters of the simulated annealing algorithm SA were incorporated in the algorithm:

- T: Control parameter (temperature), positive real value that varies from an initial higher value, T_0, to another lower value, T_f, during the execution of the algorithm.
- N_T: Number of iterations performed by the algorithm for a certain value of T.
- α: function in T, $\alpha = \alpha(T)$, which determines the variation of T. In general: $\alpha(T) = \alpha T$, in practice: $\alpha \in [0.8; 0.99]$.
- N_{stop}: Maximum number of iterations allowed without improvement.

The method described in the pseudo-code in Algorithm 1 is applied to obtain an approximation (S_{BEST}) to the optimal solution.

Algorithm 1: Pseudo-code of the proposed SA algorithm for NPFS

```
i. Start
    A constructive procedure is applied to generate an initial solution, $S_0$
    Evaluate $z(S_0)$
    $S_0$ is accepted as the current solution: $S_A \leftarrow S_0$, $S_{BEST} \leftarrow S_0$, $N_{cont} = 0$, $t = 1$, $T = T_0$
ii. While $N_{cont} < N_{stop}$ and $T < T_f$
    Iteration t
        A solution is randomly generated in the neighborhood of $S_A$, $S_C \in V(S_A)$
        Evaluate $z(S_C)$
        Calculate $\Delta z = z(S_C) - z(S_A)$
            If $\Delta z \le 0$, the new solution is accepted: $S_A \leftarrow S_C$, $N_{cont} = 0$
            Otherwise, $S_C$ is accepted with the following probability $PA = e^{-\Delta z/T}$
            A random number $\xi$ uniformly distributed in [0,1] is generated:
            $\begin{cases} \text{if } \xi \le PA, S_A \leftarrow S_C, N_{cont} = 0, \\ \text{if } \xi > PA, S_A \leftarrow S_A, N_{cont} = N_{cont} + 1. \end{cases}$
        If $z(S_A) < z(S_{BEST})$ then $S_{BEST} \leftarrow S_A$
    $t \leftarrow t + 1$
    If t is a multiple of $N_T$, then $T = \alpha T$, otherwise the value of T is maintained
End While
Return $S_{BEST}$
```

5. Experimental analysis

This section describes and analyzes the results of the computational experimentation performed to test the proposed algorithms. Particularly, this section presents the tests instances used in the experimentation, the assessment of the computational performance of the algorithms, the comparison between the results of the permutation and the non-permutation instances and, finally, an overall analysis of the competitiveness of the proposed approach based on the experimental results.

5.1. Description of the tests Instances

For generating the test instances, we considered as reference previous works that have addressed the NPFS problem with missing operations (Pugazhendi et al., 2004a; Henneberg & Neufeld, 2016) and those that considered due-date related objective functions (Ruiz & Stützle, 2008; Toncovich et al., 2019). As it was explained in the mathematical formulation section, some parameters must be set to define each scenario within each instance. In the case of processing times ($p_{i,j}$) which are always integer values, the data are obtained from a pseudo-uniform distribution in the interval [0;100]. Since the problem addressed here contemplates missing operations, the zero value is included in the processing time distribution. In the pseudo-uniform distribution, the zero value (i.e., $p_{i,j} = 0$), which is the probability of skipping an operation, has a special probability and then rest of the integer values from the interval [1;100] have all the same probability. Two groups of instances were built, one group in which the special probability of skipping an operation is set to 5% and another group where this probability is set to 10%. The relatively large probability assigned to the zero value aims to amplify the impact of skipping operations in the scheduling problem. Then, in regard to the due date of the jobs, we followed the guidelines given in Ruiz and Stützle (2008), where the due dates are calculated using the following expression: $d_j = r_j + \sum_{i \in I} p_{i,j} \cdot (1 + random \cdot 3)$. This representation contemplates all the operations that job j must pass through for its completion plus an additional time considered by the term of the *random number*, which is a random number uniformly distributed in the interval [0;1]. Then, d_j is rounded to the nearest integer.

Depending on the number of jobs and machines, three sizes of instances are considered: small, medium and large. Small instances include those in which $n = 40$ jobs and $m = 15$ machines. Medium instances are those in which n can be either 40, 80, or 120 while m is 20. Finally, large instances are those in which n can be either 120 or 150 while m is 30. To the best of our knowledge, the set of large instances, are the largest in terms of number of machines and jobs used in the non-permutation flow shop scheduling with missing operation literature. In turn, this experimental design allows to analyze different sizes of instances modifying only one of the sets (jobs or machines). That is, for the case where $m = 20$, there is the possibility of analyzing the impact of the number of jobs in isolation, since m is kept constant for $n = 40, 80, 120$. The reciprocal analysis can also be performed in our experimental design, that is, setting $n = 40$ and analyzing $m = 15, 20$, or the same for $n = 120$ with $m = 20, 30$.

5.2. Performance of the proposed algorithms

To evaluate the performance of the algorithms the Relative Percentage Deviation (RPD) is used as a performance metric. RPD evaluates the percentage deviation of a particular solution S obtained by the algorithm in comparison to the best solution B obtained for that problem. The RPD is calculated according to equation (10).

$$RPD = \frac{S - B}{B} \cdot 100\% \tag{10}$$

Therefore, the first indicator that is used is *Ave.*RPD, which is the average RPD for all the solutions S obtained for a given instance. The second indicator is the *Dev.*RPD, which is the standard deviation of that sample of solutions S with respect to the average *Ave.* RPD. This indicator evaluates the robustness of the algorithms. Since the purpose of these metrics is to evaluate the performance of the algorithm, we identify the indicator with the algorithm with a super index "ssGA" and "SA" for ssGA and SA, respectively.

Table 1
Results for PFS with missing operation solved by ssGA.

m	***n***	***Ave.* RPD^{ssGA}**		***Dev.*RPD^{ssGA}**	
		5%	**10%**	**5%**	**10%**
15	40	20.0%	15.5%	8.6%	9.5%
20	40	38.7%	34.7%	25.4%	20.0%
	80	4.3%	6.0%	2.3%	3.2%
	120	2.5%	2.8%	1.3%	1.4%
30	120	3.6%	4.4%	1.8%	2.1%
	150	2.7%	2.7%	1.3%	1.4%
Average		12%	11%	6.8%	6.3%

Initially, the behavior of the proposed ssGA algorithm is analyzed. This information is presented in Table 1, where the *Ave.*RPD^{ssGA} values after 30 independent executions of the ssGA for each of the six data sets are shown. Each dataset has two variants that differ in the probability of appearance of zero values in the processing times: 5% and 10% respectively. From Table 1, it is possible to state that the ssGA tends to be more accurate as the size of the instance grows. For example, for instances with 40 jobs the *Ave*.RPD^{ssGA} is larger than 20% while for instances of more than 80 jobs this value falls below 5%. A similar behavior can be depicted when considering *Dev*.RPD$^{ssGA:}$ the larger the instance (in terms of jobs and machines) the lower the standard deviation, reaching values below 2% for instances with 120 jobs or larger. Regarding the effect of the probability of skipping an operation (5% and 10%), there is not a significant impact on the algorithm performance, except for the global values, i.e., total averages indicated in the last row of Table 1. The cases with 10% of probability of missing operations tend to be more precise than the cases with 5%. An interesting fact of Table 1 is that for the same number of works *n*, the number of machines *m* affects the performance of the algorithm. For example, for *n* = 120 and *m* = 20, *Ave*.RPD^{ssGA} = 2.5% but when *m* = 30, then *Ave*.RPD^{ssGA} = 3.6%. This behavior is maintained for the other instances of Table 1. While, in the reciprocal case of fixing *m*, the larger the number of jobs, the smaller is the indicator *Ave*.RPD^{ssGA}.

Regarding the global values of Table 1, the main conclusion is that the ssGA has an acceptable performance when compared with other similar works of the literature (Pugazhendi et al., 2004a; Henneberg & Neufeld, 2016) which are also slightly above 10%. However, it is important to highlight that in those papers the authors worked with regular objective functions (makespan, total flow time, etc.) so the comparison is only valid for acquiring a general perception of performance.

Table 2
Results for NPFS with missing operation solved by SA.

m	***n***	***Ave*.RPD^{SA}**		***Dev*.RPD^{SA}**	
		5%	**10%**	**5%**	**10%**
15	40	17.9%	17.3%	10.4%	10.0%
20	40	40.2%	39.4%	25.6%	22.9%
	80	4.6%	6.4%	2.4%	3.3%
	120	2.5%	2.8%	1.3%	1.4%
30	120	4.4%	4.5%	2.4%	2.2%
	150	2.9%	2.7%	1.3%	1.5%
Average		12.1%	12.2%	7.2%	6.9%

Table 2 presents the same analysis, but the algorithm evaluated is the SA. Again, it is observed that the greater the size of the instance (in terms of machines and jobs), the lower the *Ave*.RPD^{SA}. For example, for the instances with 5% probability of missing operation *Ave.*RPD^{SA} starts from a value of 40.2% for the instance with 20 machines and 40 jobs up to 2.9% for the instance with 30 machines and 150 jobs. In regard to *Dev*.RPD^{SA}, the dispersion of results also decreases considerably as the size of the problem grows. For example, in the instances of 10% of missing operations, *Dev*.RPD^{SA} has values above 20% for 20 machines and 40 jobs, while it descends to values below 2% for 30 machines and 150 jobs. Regarding the analysis of results when *n* is constant and *m* varies and when *m* is constant and *n* varies, both situations have a similar behavior. When *n* is constant, as *m* increases the variability in the performance of the algorithm also increases (for the instance with 120 jobs and 5% of missing operations, *Ave*.RPD^{SA} goes from 2.5% to 4.4%). In the reciprocal case, when *m* is constant, the larger *n,* the smaller *Ave*.RPD^{SA}. Regarding the *Dev*.RPD^{SA}, the behavior follows a similar trend. Considering the global values of Table 2, expressed in the last row, we can again see values that are similar to others in the literature. However, again the direct comparison is not valid because different optimization metrics were used.

Fig. 1 summarizes the results obtained for the problem variants and algorithms, considering a 5% of zero values in the processing times. Similarly, Fig. 2 summarizes the results obtained for the problem variants and algorithms, considering a 10% of zero values in the processing times

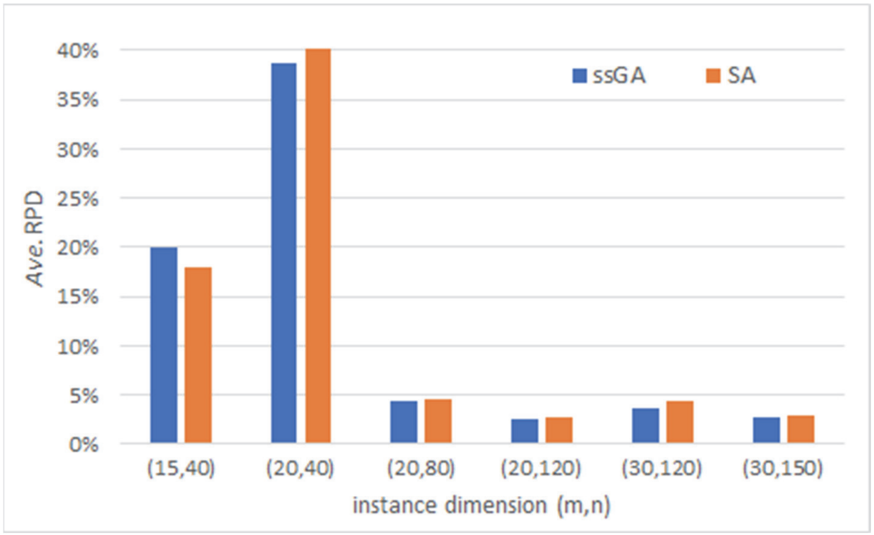

Fig. 1. *Ave.* RPD^{ssGA} results for ssGA in PFS problem variant and SA in NPFS problem variant, considering a 5% of zero values in the processing times

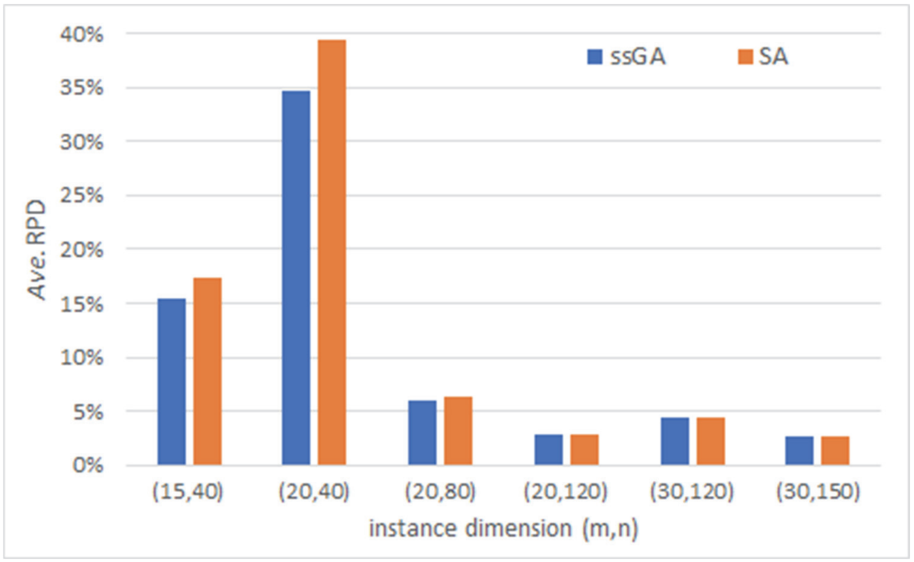

Fig. 2. *Ave.* RPD^{PFS} results for ssGA in PFS problem variant and SA in NPFS problem variant, considering a 10% of zero values in the processing times

5.3. Comparison of PFS and NPFS solutions

This section analyzes the impact of considering non-permutation solutions when solving the flow shop scheduling problem with missing operations (compared with the case of only considering permutation solutions). The runs performed for NPFS had a time limit of 3600 seconds whereas for PFS the stopping condition was set to 300,000 number of evaluations of the objective function. The comparative results between PFS and NPFS solutions are presented in Table 3. This Table shows the average improvement values for 30 independent runs for each instance and each dataset, the dispersion of this relative improvement (considering the standard deviation) and the percent frequency when NPFS solution outperforms the initial PFS solution. In order to calculate the relative improvement, the best solution obtained in PFS, i.e., *sol.*PFS, and the best solution obtained in NPFS, i.e., *sol.*NPFS, for the same data set were considered. The calculation of relative improvement (*Ave.NPFS^{impr}*) is performed with Eq. (11).

$$\boldsymbol{Ave.NPFS^{impr}} = \frac{sol.PFS - sol.NPFS}{sol.PFS} \cdot 100\% \qquad (11)$$

The standard deviation, *Dev.NPFS^{impr}*, was calculated from the deviations of the *Ave.NPFS^{impr}*. This data is depicted in Table 3. For all the instances the NPFS solutions allowed to improve, on average, the PFS solution of the problem. These average improvements are greater for smaller instances (in terms of jobs and machines). For example, the values that are greater than 3.5% for instances of 40 jobs and 10% of missing operations decrease, as the number of machines and jobs increase, to below 1%. The same happens for instances of 5% missing operations, although in these cases the improvement percentages are lower than those obtained for instances with 10% of missing operations in their datasets. This is clearer if the overall values of improvement of Table 3 (last row) are compared.

Table 3
NPFS solution improvement over PFS solution.

m	***n***	***Ave.NPFS^{impr}***		***Dev.NPFS^{impr}***		***Frequency***	
		5%	**10%**	**5%**	**10%**	**5%**	**10%**
15	40	1.83%	3.53%	1.73%	2.20%	96.7%	100%
20	40	2.80%	3.97%	3.36%	3.74%	96.0%	98,7%
	80	0.50%	0.95%	0.43%	0.50%	99.3%	100%
	120	0.53%	0.82%	0.27%	0.38%	99.3%	100%
30	120	0.58%	0.90%	0.27%	0.44%	98.9%	100%
	150	0.55%	0.81%	0.23%	0.33%	100%	100%
Average		1.1%	1.8%	1.0%	1.3%	98.4%	99.8%

For the cases of 10% of missing operation, average improvements of 1.8% are achieved, while when the percentage of missing operation is 5%, the average improvement obtained through the NPFS approach is 1.1%. Regarding the dispersion of these improvements (*Dev.NPFS^{impr}*), the larger the size of instances, the smaller is the dispersion of the values. Finally, NPFS improvements for different percentages of missing operations are compared. In global terms, it can be said that the greater percentages of missing operations, the greater the values of *Ave.NPFS^{impr}*. This trend is consistent for all problem sizes, being more noticeable for smaller problems. For example, in the case of m = 15 and n = 40, when considering 5% of missing operations *Ave.NPFS*^{impr} is 1.83%, while if 10% of missing operations is considered, the value of *Ave.NPFS*^{impr} is almost twice that value (3.53%). For larger problems, these differences are less perceptible, for example for the largest problem of $m = 30$ and $n = 150$, for 5% of missing operations, *Ave.NPFS*^{impr} = 0.55%, while if the percentage of missing operations is of 10%, *Ave.NPFS*^{impr} = 0.81%. These results support the observation raised as a motivation for this paper, that for this type of problems the NPFS approach is advantageous. Fig. 3 graphically reports the average NPFS improvements, considering 5% and 10% of zero values in the processing times.

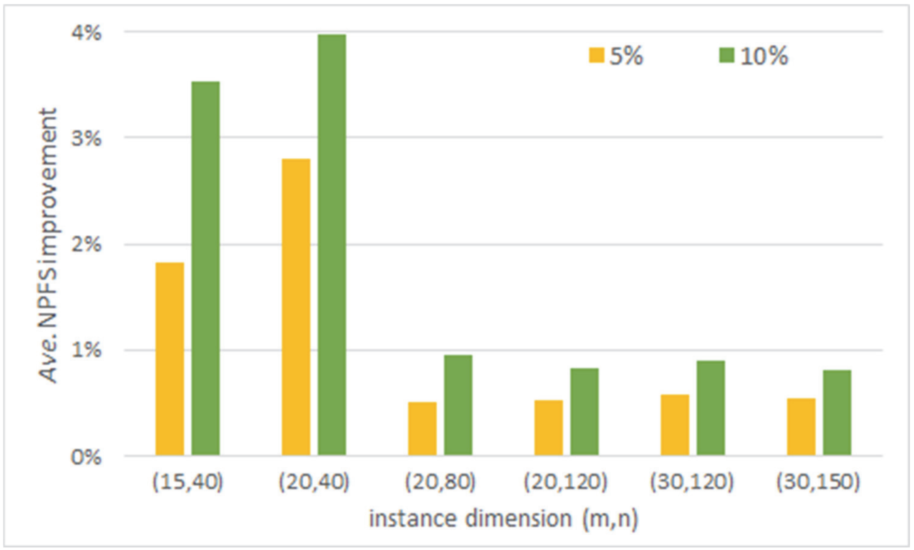

Fig. 3. *Ave.* NPFS improvements, considering 5% and 10% of zero values in the processing times

A noteworthy fact is the number of cases in which NPFS manages to improve the initial PFS solution, which is shown in the last two columns of Table 3. For none of the instances, the percentage of cases in which NPFS manages to improve PFS is less than 96% (reaching 100% of the cases in several of the instances). In fact, analyzing these values it can be seen that as the amount of missing operations grows, the probability that NPFS obtains a better solution than PFS is almost 100%. Therefore, for production systems with a high proportion of missing operations, not only NPFS is very likely to outperform the PFS, but also these improvements might be significant in terms of total tardiness.

6. Conclusions

This work addresses the non-permutation flow shop scheduling problem considering missing operations in the context of the Industry 4.0 paradigm. Therefore, it integrates a traditional problem of the scheduling literature within the modern production requirements of customization that arise in the fourth industrial revolution, what it is not common in the related literature.

This article contributes with a novel mixed-integer mathematical formulation and a solution approach for a non-permutation scheduling problem considering missing operations with the aim of reducing the total tardiness. The devised solution approach combines two metaheuristics algorithms. In first phase, a steady state Genetic Algorithm is applied to obtain an initial solution of the problem considering only PFS solutions. The obtained solution is then improved in the second phase by means of a Simulated Annealing Algorithm, which also considers NPFS solutions. The experimental tests showed that the Genetic Algorithm was able to generate high quality solutions with a smaller variability for the larger instances. The Simulated Annealing phase was effective in improving the quality of PFS solutions through the exploration of the feasible space of NPFS solutions. Overall, it can be concluded that the NPFS approach allowed significant improvements of the PFS solutions and, thus, it was able to increase customer service level through reductions of total tardiness values. Another relevant result is that the improvements of the NPFS solutions over the PFS solutions were larger for the instances with a larger amount of missing operations.

One of the main lines for future work would be to extend the analysis to other types of algorithms tailored specifically for this problem, such as memetic algorithms with local search, since the selected algorithms show the potential of the NPFS solutions for this problem. In addition, it would be interesting to incorporate other types of features to the problem, such as sequence-dependent setups times or learning effects, which would allow to better represent real world production systems.

References

Almada-Lobo, F. (2016). The industry 4.0 revolution and the future of manufacturing execution systems (MES). *Journal of Innovation Management, 3*(4), 16–21.

Altiparmak, F., Gen, M., Lin, L., & Karaoglan, I. (2009). A steady-state genetic algorithm for multi-product supply chain network design. *Computers & Industrial Engineering*, *56*(2), 521-537.

Benavides, A. J., & Ritt, M. (2016). Two simple and effective heuristics for minimizing the makespan in non-permutation flow shops. *Computers & Operations Research*, *66*, 160-169.

Benavides, A. J., & Ritt, M. (2018). Fast heuristics for minimizing the makespan in non-permutation flow shops. *Computers & Operations Research*, *100*, 230-243.

Dios, M., Fernandez-Viagas, V., & Framinan, J. M. (2018). Efficient heuristics for the hybrid flow shop scheduling problem with missing operations. *Computers & Industrial Engineering*, *115*, 88-99.

Dolgui, A., Ivanov, D., Sethi, S. P., & Sokolov, B. (2019). Scheduling in production, supply chain and Industry 4.0 systems by optimal control: fundamentals, state-of-the-art and applications. *International Journal of Production Research*, *57*(2), 411-432.

Durillo, J. J., Nebro, A. J., Luna, F., Dorronsoro, B., & Alba, E. (2006). jMetal: A java framework for developing multi-objective optimization metaheuristics. Departamento de Lenguajes y Ciencias de la Computación, University of Málaga, ETSI Informática, Campus de Teatinos, Tech. Rep. ITI-2006-10.

Framinan, J. M., Perez-Gonzalez, P., & Escudero, V. F. V. (2017, December). The value of real-time data in stochastic flowshop scheduling: A simulation study for makespan. In *2017 Winter Simulation Conference (WSC)* (pp. 3299-3310). IEEE.

Framinan, J. M., Fernandez-Viagas, V., & Perez-Gonzalez, P. (2019). Using real-time information to reschedule jobs in a flowshop with variable processing times. *Computers & Industrial Engineering*, *129*, 113-125.

Fu, Y., Ding, J., Wang, H., & Wang, J. (2018). Two-objective stochastic flow-shop scheduling with deteriorating and learning effect in Industry 4.0-based manufacturing system. *Applied Soft Computing*, *68*, 847-855.

Garey, M. R., Johnson, D. S., & Sethi, R. (1976). The complexity of flowshop and jobshop scheduling. *Mathematics of Operations Research, 1(2), 117-129.*

Glass, C. A., Gupta, J. N., & Potts, C. N. (1999). Two-machine no-wait flow shop scheduling with missing operations. *Mathematics of Operations Research*, *24*(4), 911-924.

Graham, R. L., Lawler, E. L., Lenstra, J. K., & Kan, A. R. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. In *Annals of discrete mathematics* (Vol. 5, pp. 287-326). Elsevier.

Henneberg, M., & Neufeld, J. S. (2016). A constructive algorithm and a simulated annealing approach for solving flowshop problems with missing operations. *International Journal of Production Research*, *54*(12), 3534-3550.

Hermann, M., Pentek, T., & Otto, B. (2016, January). Design Principles for Industrie 4.0 Scenarios. In *2016 49th Hawaii International Conference on System Sciences (HICSS)* (pp. 3928-3937). IEEE.

Ivanov, D., Sethi, S., Dolgui, A., & Sokolov, B. (2018). A survey on control theory applications to operational systems, supply chain management, and Industry 4.0. *Annual Reviews in Control*, *46*, 134-147

Kellegöz, T., Toklu, B., & Wilson, J. (2010). Elite guided steady-state genetic algorithm for minimizing total tardiness in flowshops. *Computers & Industrial Engineering*, *58*(2), 300-306.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *science*, *220*(4598), 671-680.

Lee, E. A. (2008, May). Cyber physical systems: Design challenges. In Object oriented real-time distributed computing (isorc), *2008 11th ieee international symposium on* (pp. 363-369). IEEE

Lee, J., Bagheri, B., & Kao, H. A. (2015). A cyber-physical systems architecture for industry 4.0-based manufacturing systems. *Manufacturing Letters*, *3*, 18-23.

Liu, Y., Wang, L., Wang, X. V., Xu, X., & Zhang, L. (2018). Scheduling in cloud manufacturing: state-of-the-art and research challenges. *International Journal of Production Research*, 1-26.

Low, C. (2005). Simulated annealing heuristic for flow shop scheduling problems with unrelated parallel machines. *Computers & Operations Research*, *32*(8), 2013-2025.

Lu, Y., Peng, T., & Xu, X. (2019). Energy-efficient cyber-physical production network: Architecture and technologies. *Computers & Industrial Engineering*, *129*, 56-66.

Lu, Y., & Xu, X. (2019). Cloud-based manufacturing equipment and big data analytics to enable on-demand manufacturing services. *Robotics and Computer-Integrated Manufacturing*, *57*, 92-102.

Luo, H., Fang, J., & Huang, G. Q. (2015). Real-time scheduling for hybrid flowshop in ubiquitous manufacturing environment. *Computers & Industrial Engineering*, *84*, 12-23.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., & Teller, E. (1953). Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, *21*(6), 1087-1092.

Monostori, L. (2014). Cyber-physical production systems: Roots, expectations and R&D challenges. *Procedia CIRP*, *17*, 9-13.

Nesmachnow, S. (2014). An overview of metaheuristics: accurate and efficient methods for optimisation. *International Journal of Metaheuristics, 3*(4), 320-347.

Osman, I. H., & Potts, C. N. (1989). Simulated annealing for permutation flow-shop scheduling. *Omega*, *17*(6), 551-557.

Ponnambalam, S. G., & Reddy, M. (2003). A GA-SA multiobjective hybrid search algorithm for integrating lot sizing and sequencing in flow-line scheduling. *The International Journal of Advanced Manufacturing Technology*, *21*(2), 126-137.

Pugazhendhi, S., Thiagarajan, S., Rajendran, C., & Anantharaman, N. (2003). Performance enhancement by using non-permutation schedules in flowline-based manufacturing systems. *Computers & Industrial Engineering*, *44*(1), 133-157.

Pugazhendhi, S., Thiagarajan, S., Rajendran, C., & Anantharaman, N. (2004 a). Relative performance evaluation of permutation and non-permutation schedules in flowline-based manufacturing systems with flowtime objective. *The International Journal of Advanced Manufacturing Technology*, *23*(11-12), 820-830.

Pugazhendhi, S., Thiagarajan, S., Rajendran, C., & Anantharaman, N. (2004 b). Generating non-permutation schedules in flowline-based manufacturing sytems with sequence-dependent setup times of jobs: a heuristic approach. *The International Journal of Advanced Manufacturing Technology*, *23*(1-2), 64-78.

Rajendran, C. (1994). A heuristic for scheduling in flowshop and flowline-based manufacturing cell with multi-criteria. *The International Journal of Production Research*, *32*(11), 2541-2558.

Rajendran, C., & Ziegler, H. (2001). A performance analysis of dispatching rules and a heuristic in static flowshops with missing operations of jobs. *European Journal of Operational Research*, *131*(3), 622-634.

Ribas, I., Leisten, R., & Framiñan, J. M. (2010). Review and classification of hybrid flow shop scheduling problems from a production system and a solutions procedure perspective. *Computers & Operations Research*, *37*(8), 1439-1454.

Rossit, D. & Tohmé, F. (2018). Scheduling research contributions to Smart manufacturing. *Manufacturing Letters, 15* (B), 111-114.

Rossit, D. A., Tohmé, F., & Frutos, M. (2018). The non-permutation flow-shop scheduling problem: a literature review. *Omega*, *77*, 143-153.

Rossit, D. A., Tohmé, F., & Frutos, M. (2019a). Industry 4.0: Smart Scheduling. *International Journal of Production Research*, *57*(12), 3802-3813.

Rossit, D. A., Vásquez, Ó. C., Tohmé, F., Frutos, M., & Safe, M. D. (2019b). A combinatorial analysis of the permutation and non-permutation flow shop scheduling problems. *European Journal of Operational Research*. doi.org/10.1016/j.ejor.2019.07.055

Rossit, D. A., Tohmé, F., & Frutos, M. (2019c). Production planning and scheduling in Cyber-Physical Production Systems: a review. *International Journal of Computer Integrated Manufacturing*, *32*(4-5), 385-395.

Rossit, D. A., Tohmé, F., & Frutos, M. (2019d). An Industry 4.0 approach to assembly line resequencing. *The International Journal of Advanced Manufacturing Technology*, *105*(9), 3619-3630.

Rossit, D., Tohmé, F., Frutos, M., & Safe, M. (2020). Critical paths of non-permutation and permutation flow shop scheduling problems. *International Journal of Industrial Engineering Computations*, *11*(2), 281-298.

Ruiz, R., & Stützle, T. (2008). An Iterated Greedy heuristic for the sequence dependent setup times flowshop problem with makespan and weighted tardiness objectives. *European Journal of Operational Research, 187*(3), 1143-1159.

Shim, S. O., Park, K., & Choi, S. (2017). Innovative production scheduling with customer satisfaction based measurement for the sustainability of manufacturing firms. *Sustainability*, *9*(12), 2249.

Simpson, T. W., Siddique, Z., & Jiao, R. J. (Eds.). (2006). *Product platform and product family design: methods and applications*. Springer Science & Business Media.

Toncovich, A., Rossit, D. A., Frutos, M. & Rossit, D. G. (2019). Solving a multi-objective manufacturing cell scheduling problem with the consideration of warehouses using a simulated annealing based procedure. *International Journal of Industrial Engineering Computations*, *10*(1), 1-16.

Tseng, C. T., Liao, C. J., & Liao, T. X. (2008). A note on two-stage hybrid flowshop scheduling with missing operations. *Computers & Industrial Engineering*, *54*(3), 695-704.

Uhlmann, I. R., & Frazzon, E. M. (2018). Production rescheduling review: Opportunities for industrial integration and practical applications. *Journal of Manufacturing Systems*, *49*, 186-193.

Vahedi Nouri, B., Fattahi, P., & Ramezanian, R. (2013). Hybrid firefly-simulated annealing algorithm for the flow shop problem with learning effects and flexible maintenance activities. *International Journal of Production Research*, *51*(12), 3501-3515.

Venkataramanaiah, S. (2008). Scheduling in cellular manufacturing systems: an heuristic approach. *International Journal of Production Research*, *46*(2), 429-449.

Vollmann, Thomas E., Berry, William L., Whybark, D. C. & Jacobs R. (2005). *Manufacturing Planning and Control for Supply Chain Management*. McGraw-Hill/Irwin. 5th Edition

Wang, M., Zhong, R. Y., Dai, Q., & Huang, G. Q. (2016). A MPN-based scheduling model for IoT-enabled hybrid flow shop manufacturing. *Advanced Engineering Informatics*, *30*(4), 728-736.

Wang, Y., Zheng, P., Xu, X., Yang, H., & Zou, J. (2019). Production planning for cloud-based additive manufacturing—A computer vision-based approach. *Robotics and Computer-Integrated Manufacturing*, *58*, 145-157.

Yao, X., & Lin, Y. (2016). Emerging manufacturing paradigm shifts for the incoming industrial revolution. *The International Journal of Advanced Manufacturing Technology*, *85*(5-8), 1665-1676.

Yu, C., Mou, S., Ji, Y., Xu, X., & Gu, X. (2018). A delayed product differentiation model for cloud manufacturing. *Computers & Industrial Engineering*, *117*, 60-70.

Zheng, P., Yu, S., Wang, Y., Zhong, R. Y., & Xu, X. (2017). User-experience based product development for mass personalization: A case study. *Procedia CIRP*, *63*, 2-7.

Zheng, P., Lin, Y., Chen, C. H., & Xu, X. (2019). Smart, connected open architecture product: an IT-driven co-creation paradigm with lifecycle personalization concerns. *International Journal of Production Research*, *57*(8), 2571-2584.

Zhong, R. Y., Xu, X., Klotz, E., & Newman, S. T. (2017). Intelligent manufacturing in the context of industry 4.0: a review. *Engineering*, *3*(5), 616-630.