ISSN 2619-1865

Proceedings ICPR Americas 2020

edi UNS

COLECCIÓN CIENCIA Y TECNOLOGÍA

International Conference of Production Research-Americas
(ICPR-Americas). ISSN 2619-1865

Editorial de la Universidad Nacional del Sur
Santiago del Estero 639 – B8000HZK – Bahía Blanca – Argentina
Tel.: 54–0291–4595173 / Fax: 54–0291–4562499
www.ediuns.com.ar | ediuns@uns.edu.ar

Libro Universitario Argentino

Diagramación interior y tapa: Fabián Luzi

No se permite la reproducción parcial o total, el alquiler, la transmisión o la transformación de este libro, en cualquier forma o por cualquier medio, sea electrónico o mecánico, mediante fotocopias, digitalización u otros métodos, sin el permiso previo y escrito del editor. Su infracción está penada por las Leyes 11723 y 25446. Queda hecho el depósito que establece la Ley 11723.

Bahía Blanca, Argentina, marzo 2021

© 2021Ediuns.

Operation Skipping Flow Shop Scheduling and Industry 4.0

Daniel Alejandro Rossit[1,2][0000-0002-2381-4352], Adrián Toncovich[2][0000-0002-1841-8726] Diego Gabriel Rossit[1,2][0000-0002-8531-445X] and Sergio Nesmachnow[3][0000-0002-8146-4012]

[1] Departamento de Ingeniería, Universidad Nacional del Sur, Av. Alem 1253, Bahía Blanca, Buenos Aires CP 8000, Argentina [2] INMABB UNS CONICET, Departamento de Matemática, Av. Alem 1253, Bahía Blanca, Buenos Aires CP 8000, Argentina [3]Facultad de Ingeniería, Universidad de la República, Julio Herrera y Reissig 565, Montevideo 11300, Uruguay;
daniel.rossit@uns.edu.ar, atoncovi@uns.edu.ar, diego.rossit@uns.edu.ar, nesmachnow@gmail.com

Abstract. Industry 4.0 is a modern approach where the connectivity between different stages of the production process and the consumers is enhanced. In this paper a relevant problem for both Industry 4.0 and flow shop literature is addressed: an operation skipping flow shop scheduling problem. A solution method to optimize total tardiness, which is novel to the related literature, is presented. The solution approach consists of two stages. In the first stage, a genetic algorithm is applied to obtain an efficient solution in a permutation fashion. In the second stage, the solution obtained in the previous stage is improved with a simulated annealing algorithm considering a non-permutation strategy. Results show that NPFS solutions outperform PFS solutions for this problem. Furthermore, if the percentage of skipping operations increases the impact of the NPFS approach increases too.

Keywords: Industry 4.0, Non-Permutation Flow Shop, Skipping Operations, Cyber-Physical System.

1 Introduction

The conceptual model Industry 4.0 enhances connectivity between the different components of the production systems, as well as among the components and the decision-making centers [1] [2]. This allows production planners to manage information in real time from the shop floor [3]. 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 [4], allowing designs of innovative products with increasing adaptability. Moreover, given the high level of digitalization and through IIoT (Industrial Internet of Things), it is possible to offer mass customization of products as a business strategy, i.e., the production process can be adjusted so that the final products meet the specific customer requirements. In other words, for a given product

each customer can choose among different combinations of optional features, configurations or characteristics in such a way that the final product is highly personalized [5]. However, these personalized products are translated into specific production activities in the production line. Therefore, efficient scheduling techniques that can manage this variability should be applied in the production line to meet the specifications of each customer in the Industry 4.0 production environment [6].

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 [7]. The resulting output of a machine is transformed into the input of the next machine, and this happens with each of the machines in the cell. Generally, this type of configurations is considered within the flow shop scheduling family of problems [8] [9]. Product customization powered by Industry 4.0 will have a significant impact on this type of problem [5]. Given the freedom provided to the customer when customizing a product, it is very likely that, when following the customer specifications, the operations that must be performed to produce this product imply different uses of the machines in the cell. Thus, some of the operations introduced in the cell design may not be required to achieve the specifications requested by customers [9], these are the missing or skipping operations.

In the related literature, problems of flow shop with skipping operations for many productive environments have been studied [10]. In the vast majority of these works, the problems are solved by considering only permutation solutions (PFS) [11], meaning that the works are processed by each machine in the same order. In terms of search space, this entails finding the best solution in a universe of $n!$ possible solutions, being n the number of jobs to be processed. However, this condition of permutativity is not a technological constraint of the production process, since, in general, machines can process works in any order. It is only a condition introduced for reducing the complexity of the scheduling process. Therefore, another approach to solve these problems is based on relaxing this condition of permutativity, which involves solving the problem under the non-permutation approach (NPFS) [12]. In this approach, the job sequence for each machine can be modified, increasing the search space considerably from n to $n!^m$ solutions, where m is the number of machines [13]. Since NPFS solutions include PFS solutions as a particular case, NPFS solutions are at least as efficient as PFS if not more efficient, and it has been demonstrated that PFS are usually outperformed by NPFS in regard to the objective function values [10] [13]. Despite the industrial interest in this problem, NPFS has not yet been addressed for systems that contemplate Industry 4.0 production environments [3]. This article applies metaheuristics to solve PFS and NPFS, following the aforementioned approach.

Although results in the related literature show that NPFS solutions tend to outperform PFS solutions when considering objective functions related to delivery dates [13], the NPFS with skipping operations considering such optimization criteria has not been studied until now [14]. Therefore, this work makes a contribution and aims at providing a solution approach for considering PFS and NPFS solutions in this kind of problem.

2 Problem Definition

In this section, we properly introduce the problem studied in this work. Firstly, we analyze the impact of Industry 4.0 in manufacturing cells and how this influences the exchange with the customer. Then, we define our conception of the problem, relating our approach to the scheduling literature.

2.1 Industry 4.0 Environment

Industry 4.0 proposes a digital transformation of traditional production systems [2]. This transformation is based on the implementation of cyber-physical systems (CPS), which integrate virtual and physical processes in the same system [15]. Moreover, through IoT, CPS can communicate with each other and with the Decision Support Systems, which in terms of production planning, means to allow directly linking the shop floor with the support systems for production decision making [15] [4]. On the other hand, CPS allows generating a digital twin of the physical system [16], to analyze situations and making decisions more agilely and with more information than in a traditional production system. This results in a much more flexible production system, better adapted to different scenarios [17]. This way, the manufacturing system becomes smart, being able to provide an improved service to the customer, satisfying its needs in a personalized way [1]. This allows bringing the final product closer to the true requirements of the customer, who is actively involved in the design of the product [17].

The customer expresses his/her preferences by specifying different variants of a base product belonging to a given family of products, configuring what is known as a personalized variant of the product [18]. In this way, by making the production system more flexible, and making it smart, Industry 4.0 allows to offer to the customer a much higher level of personalization than in traditional production systems [19] [20].

The problem addressed in our work responds to these conditions imposed by Industry 4.0 in manufacturing cell systems. Given that the focus when solving 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.

2.2 Flow shop Scheduling in Manufacturing

The problem considered in this work can be described as to schedule production for a set of products that share production resources, with the feature that all products use the same resources maintaining the same technological sequence, i.e., a flow shop scheduling problem [8]. However, considering the scenario of intensive personalization promoted by Industry 4.0 [2], each product will have its particularities, a fact that will imply a different use of resources, which is generally expressed in varying processing times. This difference in use can even imply that, directly, the product does not use one of the resources of the manufacturing cell, which means that the product skips that operation (skipping operations flow shop scheduling problem) [21]. Therefore, the objective is to schedule production in such a way that the highest service level is achieved,

complying with the delivery dates. For this, we will seek to minimize the cumulative delays in 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) [9] [14]. In the first case, PFS, the solution search space includes those 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 space of solutions grows to $n!^m$, where m is the number of machines. These problems are identified in the classic notation for scheduling problems [22] as *F|prmu/missing|TTard* and *F|missing|TTard*, respectively.

Therefore, the problem to be solved is to sequence the jobs in such a way that, complying with the precedence relationships among the operations, the objective function is minimized. If π is a permutation sequence, that implies that once job j was processed before job j', then, j will precede job j' on all machines. Therefore, to minimize the objective function, the permutation version seeks that the order π of the jobs, allows the completion times of each job, C_j, to exceed as little as possible the delivery date of each job d_j. In other words, the tardiness for each job j is as small as possible. Where the tardiness for job j is: $T_j = max\{C_j - d_j, 0\}$. Then the objective function can be expressed as:

$$\min_{\pi} \sum_j T_j(\pi) \tag{1}$$

For the non-permutation case, the objective function is the same, but the order π' is considered, which allows changes of precedence relationships among jobs. For instance, for π', job j can precede job j' up to machine i, and from machine $i + 1$ onwards, j' can precede j.

3 Solution approach

To solve the problem under study two metaheuristic algorithms were used in a sequential fashion. First, a Genetic Algorithm (GA) is used to optimize this problem considering only PFS solutions. Then, the best solution obtained by the genetic algorithm is used as the initial solution for the second algorithm. This second phase is a Simulated Annealing Algorithm (SA) that aims at improving the solution by considering NPFS solutions. The main characteristics of both algorithms are presented below.

3.1 Genetic Algorithm

For solving the PFS problem introduced in Section 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 [23] and, thus, in flowshop problems (see, e.g., [24]). The proposed algorithm was implemented in Java, by using JMETAL framework [25] version 4.5.2. This algorithm has the following features.

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

Initialization. The population of size $\#P$ is initialized by applying a random procedure to generate the permutations using 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 interchanges two elements of the permutation. The Mutation operator 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 criterion is based on the fitness, and if two compared individuals have the same fitness, either of the two is chosen with equal probability. The new individual replaces the worst individual in the population if it has better fitness.

Parameters setting. The parametrization was performed with a statistical analysis considering three main parameters: population size $\#P$, 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 using three instances of size n =15 and m = 20, different from the scenarios used for the 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 values follow a normal distribution. Since several of the runs did not adjust to the normal distribution, the medians were analyzed and the final parametric combination that was selected was $\#P$ = 100, p_c = 0.7 and p_m = 0.05.

3.2 Simulated Annealing algorithm

In this section, the simulated annealing algorithm used to solve the problem presented previously is described.

Simulated Annealing (SA) is a local search-based method that was developed from an analogy with the phenomenon of annealing [26] 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. By systematically applying this procedure, in general, 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, toward solutions that worsen the current value of the objective function. SA has shown competitiveness in handling regular flow shop environments [13] and also NPFS with missing operations [8].

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 subsequent procedure. If the newly generated candidate solution (S_C) which is in the vicinity of the current solution ($S_C \in V(S_A)$] has a smaller objective function value than the current solution, i.e., $z(S_C) \leq z(S_A)$, the candidate solution is accepted as the current solution. Conversely, if $z(S_C) > z(S_A)$, a probabilistic test known as the Metropolis criterion is used to determine the probability of accepting a relatively lower quality solution. This means, that this sequence can be accepted with a certain probability $P(A) = e^{-\Delta z/T}$. Thus, $P(A)$ decreases as the difference between the objective function values of both solutions increases. 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 selected randomly and the procedure is repeated.

4 Results

The experimental evaluation was aimed at evaluating appropriately the PFS and NPFS approaches to solve the skipping operations flow shop scheduling problem. This Section presents the instances and the experimental design that were used to assess the algorithms and describes the results obtained by applying both algorithms, making a comparison between the PFS and NPFS solution approaches.

4.1 Instances and Experiment design

For generating the test instances, we have taken as reference previous works that have considered NPFS with missing operation [8] and those that considered due-date related objective functions [27] [28]. The main parameters of the instances are the processing times of each job in each machine ($p_{i,j}$). In general, $p_{i,j}$ values are simulated using a uniform distribution within the interval [1; 99]. However, since this paper considers the possibility of job skipping machines (or operations), it is necessary to incorporate the 0 within that distribution. Furthermore, since this experiment aims at testing the algorithms in instances with a significant proportion of missing operations, a pseudo-uniform distribution is implemented where the probability assigned to $p_{i,j} = 0$ is greater than the probabilities for the rest of the possible interval values. Thus, by defining different values for the probability of occurrence of $p_{i,j} = 0$ within this pseudo-uniform distribution, it is possible to generate diverse instances with different proportions of zeros [8]. In this paper we have considered two values of probability of occurrence of $p_{i,j} = 0$: 5% and 10%. Additionally, for the calculation of the objective function it is necessary to calculate the due-date of each job j (d_j). In this case, literature guidelines presented in [27] were followed, where the due-dates are calculated according to the following formula for a job j: $d_j = \sum_{i \in I} p_{i,j} \cdot (1 + random \cdot 3)$, where *random* is uniformly distributed value between 0 and 1.

To evaluate the experimental results, we have used the Relative Percentage Deviation (RPD) to measure the performance of the algorithms. The RPD is obtained using the following formula:

$$RPD = \frac{S-B}{B} \cdot 100\% \quad (2)$$

Where S is the solution to be evaluated and B is the best known solution for that instance. For the assessment of the algorithms 30 runs where performed for each instance. Thus, we will report the average and standard deviation of the RPD. These values are computed for both algorithms, ssGA and SA. Then, the indicators are identified as $Ave.RPD^{ssGA}$ and $Dev.RPD^{ssGA}$, which represent the average values of RPD and its standard deviation for the ssGA algorithm, and, $Ave.RPD^{SA}$ and $Dev.RPD^{SA}$ in the case of SA, respectively. In order to compare NPFS and PFS solutions, the average relative improvement of the NPFS solution over the PFS solution, $Ave.NPFS$, is calculated. This improvement is measured with the following formula:

$$Ave.NPFS = \frac{sol.PFS - sol.NPFS}{sol.PFS} \cdot 100\% \quad (3)$$

Where *sol.PFS* represents the best PFS solution obtained by means of ssGA, and *sol.NPFS* represents the best solution obtained by means of SA for the same problem. $Ave.NPFS$ expresses the average value of that difference for the 30 runs. In turn, the standard deviation of these differences (between *sol.PFS* and *sol.NPFS*) was also computed, which is indicated as $Dev.NPFS$.

4.2 Experiments results

The values of *Ave.RPD* and *Dev.RPD* for the ssGA and SA algorithms are presented in Tables 1 and 2, respectively. Comparing both algorithms, it can be seen that they have similar performances, both in terms of the RPD average and standard deviation. In the case of RPD and problems with 5% skipping operations, the range of values for both algorithms goes from 4%, for problems with 20 machines and 80 jobs, up to around 40%, for problems with 10 machines and 40 jobs. In regard to the instances with 10% skipping operations, the range of values for both algorithms is around 6% for instances with 20 machines and 80 jobs. In the cases of 20 machines and 40 jobs, the RPDs differ between both algorithms, being 34.7% for ssGA and 39.4% for SA.

From the data in Tables 1 and 2 it can be suggested that, at least for the instances considered in this analysis, when the number of machines is closer to the number of jobs in absolute terms (in these cases, it would be 20 machines and 40 jobs), the RPD is maximum for both algorithms and both percentages of skipping operations. Thus, these instances are more difficult to solve for the algorithms. Another supporting argument of this inference is related to standard deviations. These are larger for instances of 20 machines and 40 jobs, those problems whose matrix of processing times tends to be more square when compared to the other problem sizes (15 machines and 40 jobs, and 20 machines and 80 jobs). However, a larger experimental work is required to draw a thorough conclusion regarding this aspect.

Table 3 presents the results of the solution comparison between the PFS and NPFS approaches. In Table 3 it can be seen that in the majority of cases NPFS solutions are better than PFS, as evidenced in the "*Improvement Freq.*" columns, since this percentage is larger than 96% for all the instances. In regard to the effect of the probability of skipping operations, when it is 10%, it is evident that NPFS is superior in all the cases except for the case of 20 machines and 40 works (in which NPFS is better in 98.7% of

the time). Regarding the improvement in terms of the objective function, on average NPFS is better than PFS, since all the values of the "*Ave.NPFS*" columns are positive. These improvements range from 0.5% for 20 machines and 80 jobs and 5% of skipping operations, to 2.8% for 20 jobs and 40 machines. When considering the instances with 10% of skipping operations, it is observed that this improvement increases, since instead of the 0.5% improvement in the problem of 20 machines and 80 jobs, it goes to 0.9% improvement. For 20 machines and 40 jobs, the improvement goes from 2.8% for 5% of skipping operations, to 3.97% for 10% of skipping operations. The standard deviations of NPFS improvements over PFS ("*Dev.NPFS*" column) also increases going from 5% to 10% of skipping operation.

Table 1. Average and Deviation RPD values for 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%

Table 2. Average and Deviation RPD values for 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%

Table 3. Solution improvement of NPFS approach over PFS approach.

m	*n*	*Ave.NPFS*		*Dev.NPFS*		*Improvement Freq.*	
		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%

5 Conclusions

This work addressed a scheduling problem in Flow shop systems that arises in Industry 4.0 manufacturing environments. This is the scheduling problem with skipping operations that takes into account the increase in personalization of products to meet the specific requirements of customers. To address this problem, this work proposes an innovative way to consider NPFS solutions to find efficient production schedules, seeking to minimize the total tardiness. Two meta-heuristic algorithms are presented that

allow to tackle realistic size instances, a steady state Genetic Algorithm (ssGA) and a Simulated Annealing algorithm (SA). Both algorithms were effective in addressing the instances considered in the experimental study. Specifically, the ssGA was used to optimize the PFS approach, and SA was used to optimize the NPFS. When comparing solutions, it was observed that NPFS tends to outperform PFS solutions in the vast majority of cases. Even when considering different proportions of skipping operations, it was observed that NPFS tends to increase its dominance over PFS when increasing the percentage of skipping operations.

To continue advancing in this line of research, it is proposed to approach problems that consider other types of objective functions, for example, objective functions of the regular type (e.g. makespan), as well as multi-objective problems. Additionally, another future research avenue would be to expand the experimentation by considering instances with larger percentages of skipping operations.

References

1. Hermann, M., Pentek, T., & Otto, B. (2016, January). Design Principles for Industry 4.0 Scenarios. In *2016 49th Hawaii International Conference on System Sciences (HICSS)* (pp. 3928-3937). IEEE.
2. 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.
3. Rossit, D. A., Tohmé, F., & Frutos, M. (2019). Production planning and scheduling in Cyber-Physical Production Systems: a review. *International Journal of Computer Integrated Manufacturing*, *32*(4-5), 385-395.
4. Rossit, D. & Tohmé, F. (2018). Scheduling research contributions to Smart manufacturing. *Manufacturing Letters*. 15 (B), 111-114.
5. 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.
6. 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.
7. 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.
8. 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.
9. 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.
10. 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.
11. Venkataramanaiah, S. (2008). Scheduling in cellular manufacturing systems: an heuristic approach. *International Journal of Production Research*, *46*(2), 429-449.
12. Rossit, D., Tohmé, F., Frutos, M., Bard, J., & Broz, D. (2016). A non-permutation flowshop scheduling problem with lot streaming: A Mathematical model. International Journal of Industrial Engineering Computations, 7(3), 507-516.

13. Liao, C. J., Liao, L. M., & Tseng, C. T. (2006). A performance evaluation of permutation vs. non-permutation schedules in a flowshop. *International Journal of Production Research*, 44(20), 4297-4309.
14. Rossit, D. A., Tohmé, F., & Frutos, M. (2018). The non-permutation flow-shop scheduling problem: a literature review. *Omega*, *77*, 143-153.
15. 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.
16. 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.
17. 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.
18. 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.
19. Rossit, D. A., Tohmé, F., & Frutos, M. (2019). Industry 4.0: Smart Scheduling. *International Journal of Production Research*, *57*(12), 3802-3813.
20. Rossit, D. A., Tohmé, F., & Frutos, M. (2019). An Industry 4.0 approach to assembly line resequencing. *The International Journal of Advanced Manufacturing Technology*, *105*(9), 3619-3630.
21. 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.
22. 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.
23. 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.
24. 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.
25. 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.
26. Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *science*, *220*(4598), 671-680.
27. 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.
28. 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.