

Article

Discrete-Event Simulation Integrates an Improved NEH Algorithm for Practical Flowshop Scheduling Problems in the Satellite Industry

Guangzhen Li ^{1,2,3} and Lei Zhang ^{1,2,3,*}¹ Chang Guang Satellite Technology Co., Ltd., Changchun 130032, China; li.guangzhen@foxmail.com² Changchun Institute of Optics, Fine Mechanics and Physics, Chinese Academy of Sciences, Changchun 130033, China³ University of Chinese Academy of Sciences, Beijing 100049, China

* Correspondence: 18686344285@163.com

Abstract: The production of multiple types of satellites based on a common manufacturing platform represents a permutation flowshop scheduling problem (PFSP) with complex constraints. This is a highly complex scheduling problem, yet there is still a gap between theoretical research and practical application, particularly in the satellite industry. Therefore, we propose a more practical method that integrates discrete-event simulation modelling and an improved NEH algorithm to solve a more realistic PFSP. The discrete-event simulation-based method includes the following three main components: a flexible PFSP simulation modelling approach, an improved NEH algorithm, and an interaction mechanism between the simulation model and the optimisation algorithm. The proposed method allows automatic and flexible simulation modelling according to the characteristics of the actual satellite manufacturing workshop, which determines the practical nature of the approach proposed in this paper and then achieves excellent scheduling results based on the special interaction mechanism. The computational results demonstrate that this is a 9.18% improvement over the initial NEH algorithm and a 1.40% improvement over the best current improved NEH algorithm.

Keywords: discrete-event simulation; simulation modelling; satellite batch production; flowshop scheduling; NEH algorithm; Plant Simulation



Citation: Li, G.; Zhang, L. Discrete-Event Simulation Integrates an Improved NEH Algorithm for Practical Flowshop Scheduling Problems in the Satellite Industry. *Appl. Sci.* **2024**, *14*, 9755. <https://doi.org/10.3390/app14219755>

Academic Editor: Arkadiusz Gola

Received: 4 October 2024

Accepted: 21 October 2024

Published: 25 October 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

With the increasing maturity of satellite technology and the rapid development of commercial space, companies such as One Web, SpaceX, and Leosat all have plans to build large constellations of hundreds or thousands of satellites [1], including China's Chang Guang satellite technology company. These include SpaceX's Starlink constellation program of up to 42,000 [2]. To accommodate such a large number of satellites, there has been a tremendous deal of interest in the study of satellite batch production [3–6]. One of the future batch challenges we face is the production of multiple types of satellites based on a common manufacturing platform, which is a permutation flowshop scheduling problem (PFSP) with complex constraints.

PFSPs are undoubtedly the most frequently investigated problem in scheduling theory [7]. The problem can be described as follows: Each of n jobs is available at time zero and must be processed on m machines sequentially, e.g., n satellites are to be assembled on m machines in a generalised manufacturing platform. No machine can process more than one job at a time, and no job pre-emptions are allowed. The sequence of m machines is fixed, but the order of n jobs is variable. The goal of PFSPs is to find a job sequence that minimizes makespan. PFSPs are common in many industries, such as the automotive manufacturing, integrated circuit fabrication, petrochemical, and pharmaceutical industries [8], and is also the constituent form of many industrial productions but has not yet been applied to satellite

applications. It has been proven that some variants of PFSP are NP-hard problems [9]. Therefore, PFSPs have a wide range of engineering application backgrounds and high theoretical research value.

With the depth of research and the need to solve practical problems, more variants of PFSP are being studied, for example, multi-objective PFSPs [10], no-idle and no-wait PFSPs [11], PFSPs with sequence-dependent setup time [12], PFSPs with limited buffers [13], and variations and combinations of these problems [14,15]. Most studies are based on a mathematical approach, like some new algorithm or new search method [16,17]. However, due to the complexity and randomness of the actual environment, it is difficult to accurately establish a mathematical model for the objective functions. With the advent of small-batch and customised production modes, the mathematical approach of PFSPs will make it increasingly difficult to meet future research needs. According to the published PFSP literature, a very important phenomenon is that theoretical research on PFSPs has rarely been applied, which indicates that there is still a considerable gap between theoretical research and practical application in the PFSP field. The first reason for this phenomenon is that it is difficult to describe a complex real system with mathematical models accurately. The second reason may be that most researchers have only focussed on one particular problem in the PFSP at a time. At present, a more pragmatic method for solving a realistic PFSP is still lacking.

With the development of computers, discrete-event simulation technology has become an important decision-making tool for solving various discrete-event problems [18,19]. In addition, the authenticity of discrete-event simulation is more prominent compared to mathematical methods. For example, assessing the maturity level of Industry 4.0 [20], determining the logistics scheduling strategies [21], determining the production layout [22], allocating the resources in production logistics systems [23], evaluating the eco-efficiency of railway systems [24], identifying and alleviating bottlenecks [25], rationalising the allocation of production resources in conjunction with deep reinforcement learning [26], evaluating fossil fuel boiler electrification for decarbonisation [27], and conducting supply chain management research [28].

PFSPs are also a type of discrete-event problem, and there is a relatively small amount of research that combines discrete-event simulation with scheduling [29]. NEH is one of the most efficient algorithms for solving PFSPs, and its initials are kept in honour of its three creators [30]. However, it has not yet been used in discrete-event simulation optimisation. Jinping Zhou [31] suggested a flexible simulation modelling approach for an ideal PFSP and studied the sequencing of jobs using a genetic algorithm (GA) software package. However, he did not consider more practical constraints, and the GA did not produce very good solutions. The literature [29] shows similarities with Jinping Zhou's work, which has similar problems. Ištoković et al. [32] built a simulation model of a real-batch PFSP and solved the problem using the GA software package, despite lacking efficient modelling procedures and optimisation algorithms. Allaoui and Artiba [33] developed a simulation model of a hybrid PFSP with maintenance constraints and coded the simulated annealing algorithm (SAA) in the software environment to solve this real-world problem, demonstrating that the algorithm outperforms the classical NEH algorithm under certain conditions. Similarly, more efficient simulation modelling methods and more efficient algorithms are not considered in the literature [33].

Based on the above analysis, it can be concluded that the relevant studies have demonstrated the feasibility of discrete-event simulation applied to the permutation flowshop scheduling problem. Although it is possible to solve for a satellite flowshop based on the method described above, the following shortcomings also exist:

- (1) Only the flexible simulation modelling method for the ideal PFSP is considered without considering more complex constraints, which lack practicality.
- (2) The GA and SAA used in the literature are of lower quality, lacking more efficient optimisation algorithms.

This paper is mainly concerned with solving the two problems above, and the rest of this paper is organised as follows. In Section 2, we propose the discrete-event-simulation-based method, which can be introduced in three main components. In Section 3, computational results are presented to demonstrate the effectiveness of the proposed method for solving PFSPs, also proving that the new heuristic outperforms the best NEH-based heuristics reported in the literature. Section 4 presents concluding remarks.

2. Discrete-Event-Simulation-Based Method

2.1. Research Methodology

Figure 1 shows the research methodology of this paper. It uses Plant Simulation software to integrate all the methods. This research includes two main parts: the model and the algorithm. The model considers two kinds of constraints and achieves fully automated modelling, which can model the constraints of real plants, such as the number of machines, buffers, processing time, and waiting time. The algorithm proposes an improved NEH algorithm and proves that it is superior to several currently known improved NEH algorithms. Figure 1 shows how the approach taken in this paper differs from traditional mathematical approaches. It uses a model with real constraints to calculate completion times, making the results more practical. This is also the significance of using a simulation model with optimisation.

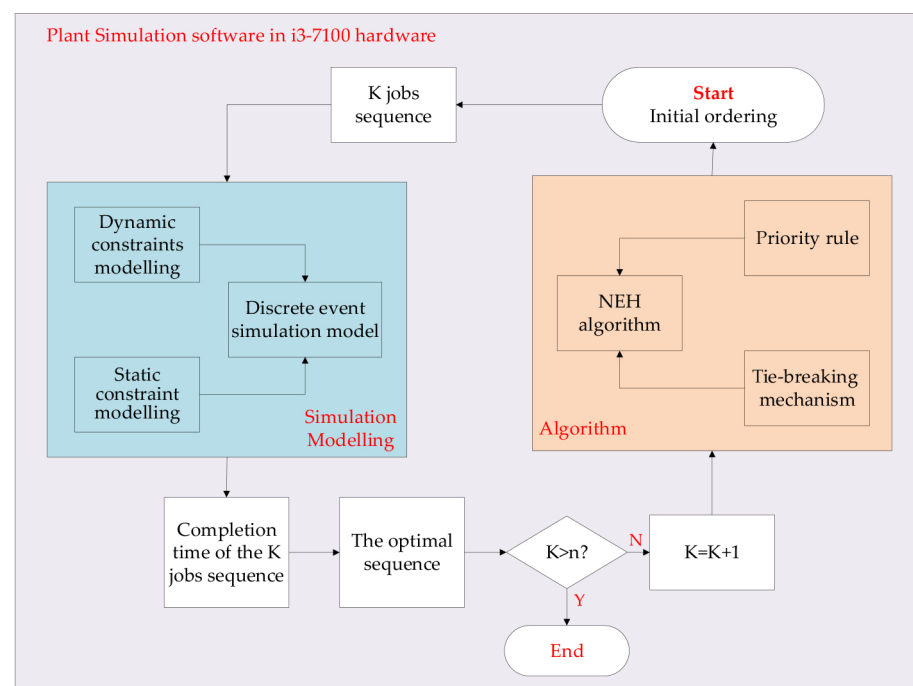


Figure 1. Research methodology of this article.

2.2. Flexible Simulation Modelling Approach

First, we need to make a classification of all the constraints in the real workshop:

(1) Dynamic constraints: The value of this constraint changes with the dynamic production process, for example, the processing time associated with the sequence, the time that the workpiece is on different machines, and the varying durations that the machine needs to process different workpieces.

(2) Static constraints: The specific value of the constraint does not change with the dynamic production process. For example, the buffer capacity is determined in advance according to the production needs and space size before the workshop is designed and built, and the overall buffer capacity does not change during the production process.

According to the literature review, the sequence-dependent setup time and buffer capacity are the most common constraints. The object-oriented programming language

SimTalk in Plant Simulation software provides the possibility of flexible modelling. Using some constraints as a prerequisite for modelling, using the SimTalk language to create entities and a connection between the entities, and setting constraints to complete the PFSP modelling can greatly simplify the modelling process. Then, a fast automated modelling program for each of these two constraints was created in Plant Simulation software with the pseudo-code shown in Figures 2 and 3.

Procedure 1 Model construction (Dynamic constraints)

1: **Inputs:**

Global variable *m* = number of machines
Global variable *boolean-a* = switch control of unique buffer capacity mode
Table *buffer-capacity* = a vector representing the capacity of each buffer

2: **Initialize:**

Clear all object in the previous model
Local integer variable *i* = 0, object variable *obj-a* = void

3 Create object source in (40, 240) position of the current layer and let *obj-a* = source

4: **for** *i* = 1 to *m* **do**

5: **if** *i* > 1 **then**

6: Create buffer in (100**i*, 200) position of the current layer and rename the buffer to BF_{*i*}

7: **if** *boolean-a* = true **then**

8: Set BF_{*i*} capacity to table *buffer-capacity* [*i*, 1]

9: **else**

10: Set BF_{*i*} capacity to infinity

11: **end if**

12: Connect BF_{*i*} and M_{*i-1*}

13: *obj-a* = BF_{*i*}

14: Create machine in (100**i* + 40, 200) position of the current layer and rename the machine to M_{*i*}

15: Embed **Procedure 2** to the entrance-ctrl of M_{*i*}

16: Connect *obj-a* and M_{*i*}

17: **end if**

18: **end for**

19: Create object drain in (100**m* + 100, 240) position of the current layer and let *obj-a* = drain

20: Connect *obj-a* and M_{*i*}

Figure 2. Pseudo-codes of dynamic constraints in the simulation environment.

Procedure 2 Set machine (Static constraint)

1: **Inputs:**

Global variable *boolean-b* = switch control of setup time mode
Global variable *boolean-c* = switch control of recovery time mode
Table *process-time* = a matrix representing the process time of each job on each machine
Table *setup-time* = a matrix representing the setup time of each job on each machine
Table *recovery-time* = a matrix representing the recovery time of each job on each machine

2: **Definition:**

Table *result-table1* = a matrix recording the entry time of each job on each machine

Table *result-table2* = a matrix recording the departure time of each job on each machine

3: When jobs are about to enter machine

4: *i* = index of current job

5: *j* = index of current machine

6: The process time of current job on current machine = *process-time* [*j*, *i*]

7: **if** *boolean-b* = true **then**

8: Set setup time of current machine to table *setup-time* [*j*, *i*]

9: **else**

10: Set setup time of current machine to 0

11: **end if**

12: **if** *boolean-c* = true **then**

13: Set recovery time of current machine to table *recovery-time* [*j*, *i*]

14: **else**

15: Set recovery time of current machine to 0

16: **end if**

17: Record the entry time of current job on current machine to table *result-table1* [*j*, *i*]

18: Record the departure time of current job on current machine to table *result-table2* [*j*, *i*]

Figure 3. Pseudo-codes of static constraints in the simulation environment.

2.3. An Improved NEH Algorithm

The basic steps of the classical NEH algorithm can be described as follows:

(1) Sort the n workpieces in descending order of the average processing time of each job on m machines to obtain an initial sequence of these n workpieces to be processed.

(2) Select the first $k - 1$ jobs from the initial sequence as the current sequence (k starts at 3). Insert the i th job of the current sequence (i from k to n) into each of the k possible positions of the current sequence to obtain a sequence of k partial jobs and calculate the maximum completion time corresponding to these k partial sequences. The idea of generating a sequence of k partial jobs in this step is shown in Figure 4.

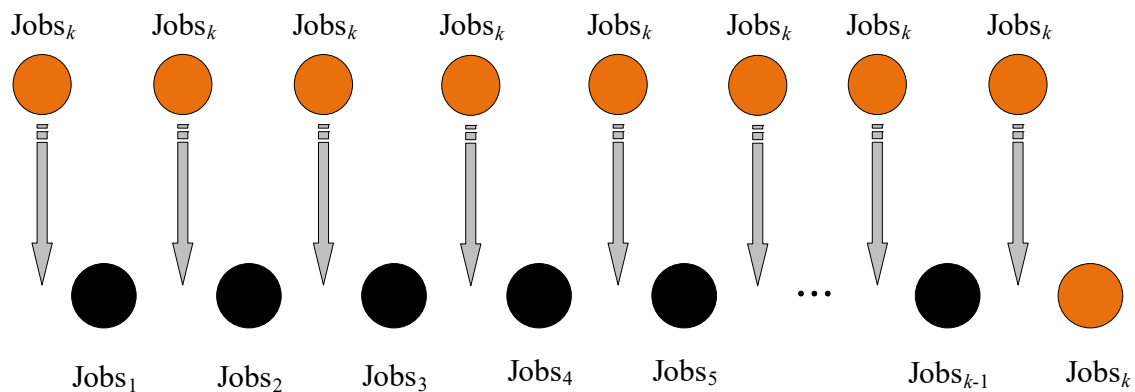


Figure 4. The idea of creating a sequence of k partial jobs.

The improvement method of the NEH algorithm generally lies in two points: the priority rule and the tie-breaking mechanism. In the reported literature, NEHLJP1 is the most effective NEH-based algorithm, and PRLJP1 and TBLJP1 contained in NEHLJP1 have been proven to be the best priority rule and tie-breaking mechanism, respectively. Although we tried several new tie-breaking mechanisms, their performance never exceeded that of TBLJP1. As a result, we simply proposed a new priority rule and used TBLJP1. The new improved NEH algorithm presented in this paper is represented by NEHL (PRL combined with TBLJP1).

Valid priority rules such as PRO [30], PRD [34], and PRLJP1 [8] are all based on a metric to determine the initial job orders. These metrics can be regarded as the sum of two parts: the mean value, which describes the overall level of the data, and the standard deviation or skewness, which describes the degree of data chaos. According to these priority rules and their computational results, the higher the data mean and the more chaotic the data, the higher the job order should be. Based on this conclusion, kurtosis is introduced in this paper to further evaluate the degree of data chaos for the first time.

Skewness and kurtosis are two indices used to describe the degree of data chaos in two directions. Our additional computational results show that a better solution quality can be obtained by multiplying the two together rather than adding them. Skewness and kurtosis are only significant for sufficiently large samples. For the PFSP with fewer machines, the statistical significance of skewness and kurtosis is not pronounced enough. Therefore, we introduce a coefficient represented by q before the product of skewness and kurtosis and set $q = 1$ when the number of machines is greater than 10; otherwise, we set $q = 0.5$. Then, all jobs can be ordered by the non-increasing sum of

$$f = AVG_i + STD_i + q \times abs(SKE_i) \times KUR_i \quad (1)$$

where AVG_i is the average processing time of job i , STD_i stands for the standard deviation of job i , $abs(SKE_i)$ represents the absolute value of the skewness of job i , and KUR_i stands

for the kurtosis of job i . The parameters are defined as follows, and the pseudo-code for the proposed algorithm is shown in Figure 5.

$$AVG_i = \frac{1}{m} \sum_{k=1}^m t_{i,k} \quad (2)$$

$$STD_i = \sqrt{\frac{1}{m-1} \sum_{k=1}^m (t_{i,k} - AVG_i)^2} \quad (3)$$

$$q = \begin{cases} 0.5 & m \leq 10 \\ 1 & m > 10 \end{cases} \quad (4)$$

$$SKE_i = \frac{\frac{1}{m} \sum_{k=1}^m (t_{i,k} - AVG_i)^3}{\left(\sqrt{\frac{1}{m} \sum_{k=1}^m (t_{i,k} - AVG_i)^2} \right)^3} \quad (5)$$

$$KUR_i = \frac{\frac{1}{m} \sum_{k=1}^m (t_{i,k} - AVG_i)^4}{\left(\sqrt{\frac{1}{m} \sum_{k=1}^m (t_{i,k} - AVG_i)^2} \right)^4} \quad (6)$$

Procedure 3 Simulation optimization method

1: Inputs:

table *process-time*, a matrix of n jobs, m machine

2: Definition:

U : $\{1 \dots n\}$ set of jobs ordered by decreasing total processing time value

seq : partial sequence

seq' : permutation sequence of a new placed job

C_{max} : makespan

$P_{i,j}$: the process time of jobs i on machine j

3: for $i = 1$ to n do

4: for $j = 1$ to m do

5: $P_i = \text{sum of } P_{i,j}$

6: end for

7: end for

8: Sort n jobs in decreasing order of P_i , and obtain U

9: for $i = 2$ to n do

10: $seq = \text{the first } i \text{ job}$

11: Move job i to k possible position in seq , thus obtaining k new seq'

12: Calculate C_{max} value of the k new seq' by running simulation model

13: the first i job of $U = \text{the } seq' \text{ with minimum makespan among } k \text{ } seq'$

14: end for

Figure 5. Pseudo-codes of NEH algorithm under simulation environment.

2.4. Interaction Mechanism Between the Simulation Model and NEH

The NEHL algorithm proposed in Section 2.2 is encoded in a method object named “NEH_L”. The NEH_L object is placed in the same structural layer as the simulation

environment established in Section 2.1. The discrete-event simulation procedure is illustrated in Figure 6, and a PFSP simulation meta-model is shown in Figure 7. The specific procedures and ideas of the simulation–optimisation mechanism presented in this paper can be described as follows:

Step 1: Establish the initial model according to the number of machines, processing time, and other initial constraints, and selectively modify the model according to the actual situation to obtain the final simulation model.

Step 2: Arrange the jobs in descending order of the PRL metric and select the first k jobs (starting at 3) as the current sequence. Insert the job k into the k possible slots of the current sequence to generate k job sub-sequences. Input these k job sub-sequences into the final simulation model.

Step 3: Run the simulation model by calling the EventController in Plant Simulation software, and the absolute time of entry and departure of each job on each machine in the above k job sub-sequences can be obtained, which can be used to calculate the makespan and TBLJP1 metric of the corresponding sub-sequence. Then, select the optimal sequence among the above k job sub-sequences by TBLJP1's mechanism. Add k to 1 and assign the optimal sequence to the new current sequence and enter the next loop. End the loop and outputs the final job sequence when $k = n$.

In summary, the automated dynamic discrete-event modelling method enables rapid modelling of real workshops with complex constraints, ensuring the practicality of the proposed method. The NEH algorithm provides a near-optimal solution to the sorting problem, ensuring the effectiveness of the proposed method.

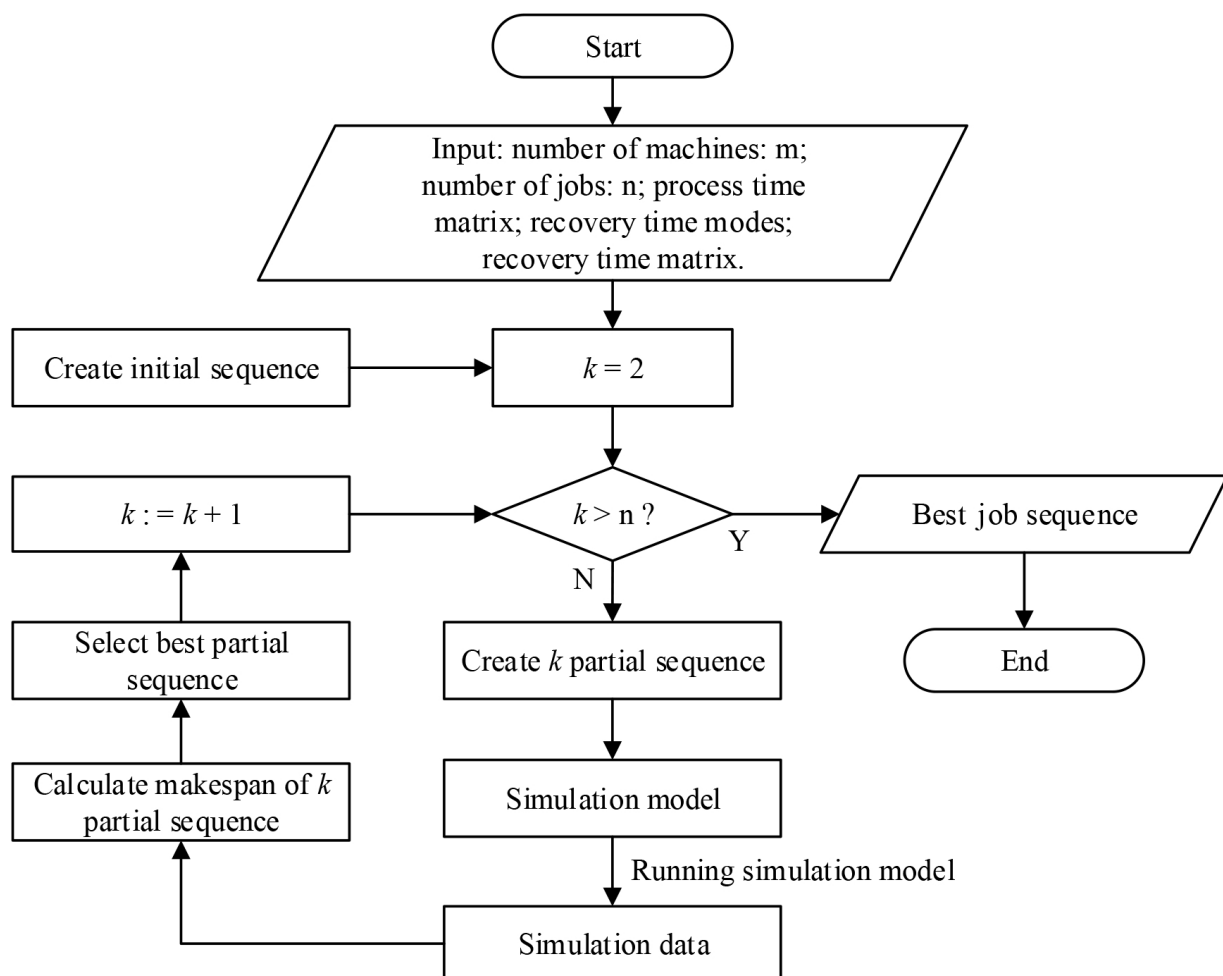


Figure 6. Simulation–optimisation procedure.

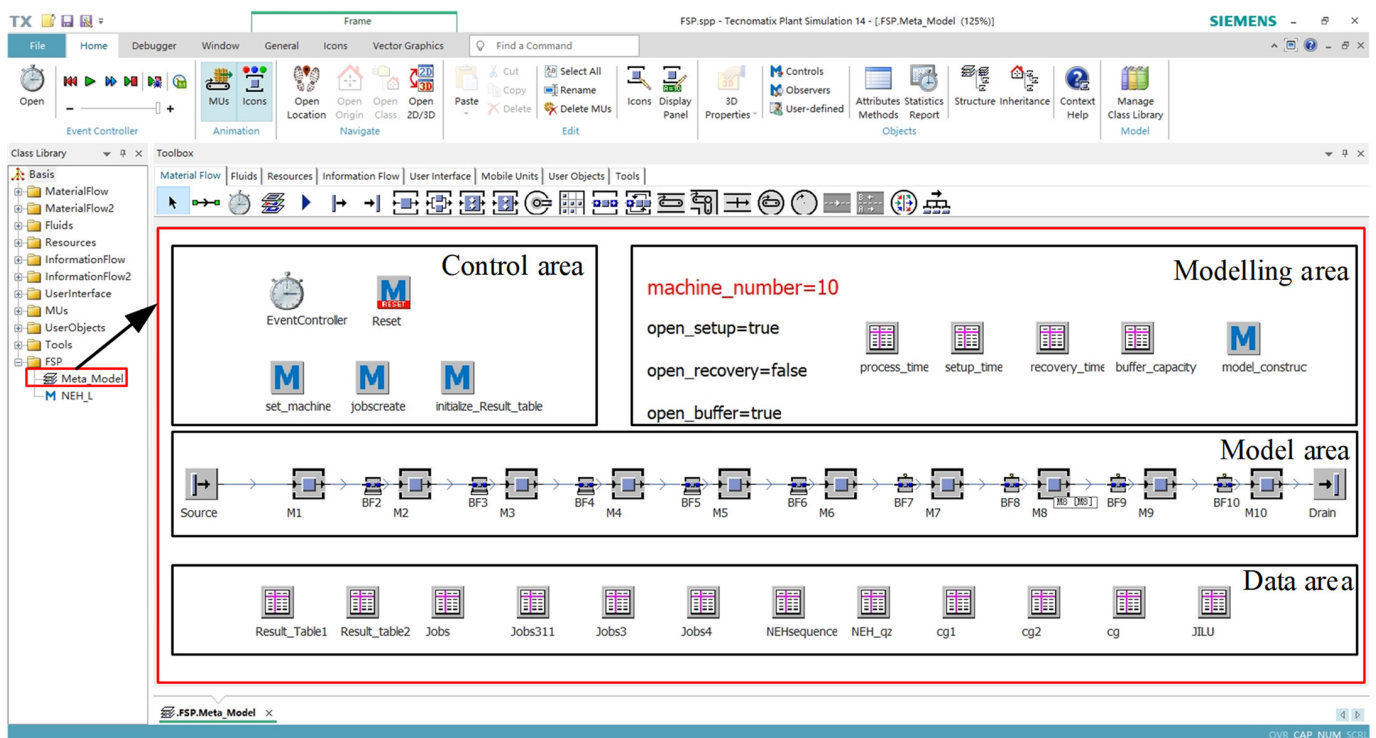


Figure 7. Simulation interface in Plant Simulation software.

3. Computational Results

In this section, we confirm the effectiveness of the new priority rule and test the performance of the new heuristic and the effectiveness of the proposed method in solving PFSPs. The priority rules including PRO, PRLJP1, PRL, and the tie-breaking mechanism including TBO, TBFF [35], and TBLJP1 are taken as references.

We combined different priority rules and tie-breaking mechanisms to obtain nine combination algorithms and tested them on the VRF benchmark [36]. All algorithms were coded in Plant Simulation 14.0 and ran on a CPU i3-7100 computer with 4.00G memory. As the time required to solve large-scale examples was much longer than we expected and the demands on the computer's performance were very high, we only calculated the first 37 instances of the VRF benchmark. This is also sufficient for justifying our conclusion, as it is unlikely that a practical production workshop would have more than 500 different products.

To measure the solution quality of each algorithm, the relative percentage deviation (RPD) is employed as the performance measure:

$$RPD = \frac{Cmax_p - UB_p}{UB_p} \times 100\% \quad (7)$$

where $Cmax_p$ represents the value obtained via heuristics on problem instance p , and UB_p is the upper bound provided by the VRF benchmark.

Table 1 shows the RPD values of nine algorithms on the VRF benchmark. As shown in Table 1, the average RPD values of the nine algorithms are 3.92, 3.76, 3.70, 3.80, 3.67, 3.61, 3.76, 3.69, and 3.56, respectively. The newly proposed algorithm NEHL (PRL with TBLJP1) outperforms all existing algorithms with an RPD value of 3.56, achieving the best on 10/37 problems of the VRF benchmark. The new priority rule PRL achieves the best on 18/37 problems compared to PRO and PRLJP1. Only comparing the NEHL and NEHLJP1 algorithms, the average RPD values of the two algorithms are 3.56 and 3.61, respectively, achieving the best results on 27/37 problems and 15/37 problems, respectively.

Table 1. RPD values of each heuristic on VRF benchmark.

	PR _O			PR _{LJP1}			PR _L		
	TB _O	TB _{FF}	TB _{LJP1}	TB _O	TB _{FF}	TB _{LJP1}	TB _O	TB _{FF}	TB _{LJP1}
10*5	2.18	2.47	2.09	1.51	1.69	1.39	1.51	1.47	1.39
10*10	1.91	2.15	1.74	1.58	2.02	1.69	1.58	2.02	1.69
10*15	1.53	1.31	1.51	2.27	2.22	2.33	2.14	2.13	2.14
10*20	1.99	1.99	1.97	1.62	1.76	1.78	1.62	1.76	1.78
20*5	1.51	1.31	1.84	2.82	2.7	2.35	2.77	2.65	2.35
20*10	4.82	4.79	3.81	4.8	5.05	4.59	4.8	5.05	4.59
20*15	4.33	4.04	4.08	3.89	3.74	3.62	3.39	3.29	3.38
20*20	4.12	3.9	4	3.51	3.38	3.59	3.5	3.2	3.47
30*5	1.43	1.22	1.3	1.54	0.9	1.67	1.55	0.96	1.81
30*10	5.26	5.44	5	4.73	4.59	4.69	4.71	4.66	4.62
30*15	5.83	5.32	5.73	5.24	5.26	4.9	5.51	5.21	4.77
30*20	5.41	5.3	5.32	5.57	5.31	5.23	5.3	5.38	5.08
40*5	1.09	0.72	0.99	0.93	0.81	1.04	0.82	0.77	0.89
40*10	4.97	4.38	4.52	4.13	4.1	4.25	4.79	4.41	4.15
40*15	6.05	5.54	5.95	5.74	5.73	5.23	5.77	5.83	5.54
40*20	5.14	5.25	4.86	5.38	5.23	5.34	4.7	5.23	4.92
50*5	0.55	0.59	0.75	0.8	0.72	0.66	0.87	0.82	0.63
50*10	4.58	4.62	3.78	4.26	4.02	3.38	4.53	4.11	4.08
50*15	6.52	6.2	6.16	6.21	6.3	6.01	6.21	6.48	5.5
50*20	5.96	6.2	5.8	5.7	5.74	5.51	5.78	5.63	5.67
60*5	0.89	0.86	0.86	0.55	0.54	0.63	0.44	0.55	0.74
60*10	3.96	4.03	4.05	4.39	3.64	3.61	4.01	3.8	3.88
60*15	5.79	5.15	5.7	5.95	5.62	5.85	5.91	5.39	5.63
60*20	6.45	6.5	6.09	6.57	6.13	6.24	5.83	6.09	5.67
100*20	5.71	5.67	5.12	5.44	5.29	5.55	5.62	5.56	5.34
100*40	5.47	5.07	5.2	5.25	5.31	5.32	5.37	5.36	5.16
100*60	4.95	4.77	4.85	4.7	4.67	4.5	4.68	4.88	4.75
200*20	4.23	4.04	3.87	4.13	4.06	3.92	4.17	3.78	3.79
200*40	4.71	4.53	4.56	4.56	4.3	4.29	4.67	4.45	4.49
200*60	4.55	4.24	4.37	4.45	4.18	4.2	4.49	4.3	4.28
300*20	3	2.88	2.64	3	2.72	2.5	2.9	2.9	2.49
300*40	4.08	3.79	3.86	3.97	3.68	3.6	3.88	3.84	3.52
300*60	3.93	3.94	3.83	3.85	3.73	3.9	3.89	3.92	3.62
400*20	2.56	2.22	2.17	2.44	2.14	2.22	2.36	2.06	2.09
400*40	3.66	3.43	3.21	3.57	3.35	3.17	3.53	3.37	3.08
400*60	3.56	3.45	3.34	3.36	3.25	3.23	3.46	3.23	3.08
500*20	2.27	1.87	1.78	2	1.86	1.67	2.14	1.87	1.75
AVG	3.92	3.76	3.7	3.8	3.67	3.61	3.76	3.69	3.56

The bold numbers are the best RPD value for each problem.

As shown in Table 1, in the columns of TBO, the priority rules PRO and PRLJP1 and the new rule PRL achieve 3.92, 3.80, and 3.76, respectively. Similarly, the effectiveness of each priority rule can be determined by comparing each sub-column. By using the new PRL priority rule, the performance of TBO and TBLJP1 is significantly improved compared to the other priority rules, PRO and PRLJP1. This is a 9.18% improvement over the initial NEH algorithm [30] and a 1.40% improvement over the best current improved NEH algorithm [8].

To further compare the computational efficiency of nine algorithms, the average CPU time (ACT) is employed as a measure:

$$ACT_p = \frac{\sum_j CPU_{j,p}}{J} \quad (8)$$

where $CPU_{j,p}$ represents the computation time on instance j by heuristic p , and J is the total number of instances.

As shown in Table 2, the computational efficiency of the nine algorithms in our solution framework is very slow, which differs greatly from the results in other literature using mathematical methods. The reason is that in the literature, the ACT time almost exclusively depends on the algorithm's complexity. However, in the discrete-event-simulation-based method presented in this paper, the total complexity is composed of the simulation model's complexity and the algorithm's complexity, while the simulation model complexity accounts for the majority. There are two reasons why the simulation model complexity dominates: The software's simulation strategy is based on the process interaction method, which deals with the change and continuation of all states of each event point along the whole time axis. The programming environment of this software has no array or matrix, so data access and processing can only be achieved through tables.

Table 2. ACT time of each heuristic on VRF benchmark.

Tie-Breaking	PR _O	PR _{LJP1}	PR _L
TB _O	1555.15	1565.20	1577.75
TB _{FF}	1615.86	1623.12	1635.08
TB _{LJP1}	1562.80	1574.15	1575.48
Mean	1577.94	1587.49	1596.1

To compare the comprehensive performance of each heuristic, the figures of RPD vs. ACT are also given, as shown in Figure 8, which shows that NEHL has the best solution quality but relatively moderate computational efficiency among the nine algorithms. It is worth noting that the average computation time of the nine algorithms is always of the same magnitude on the VRF benchmark.

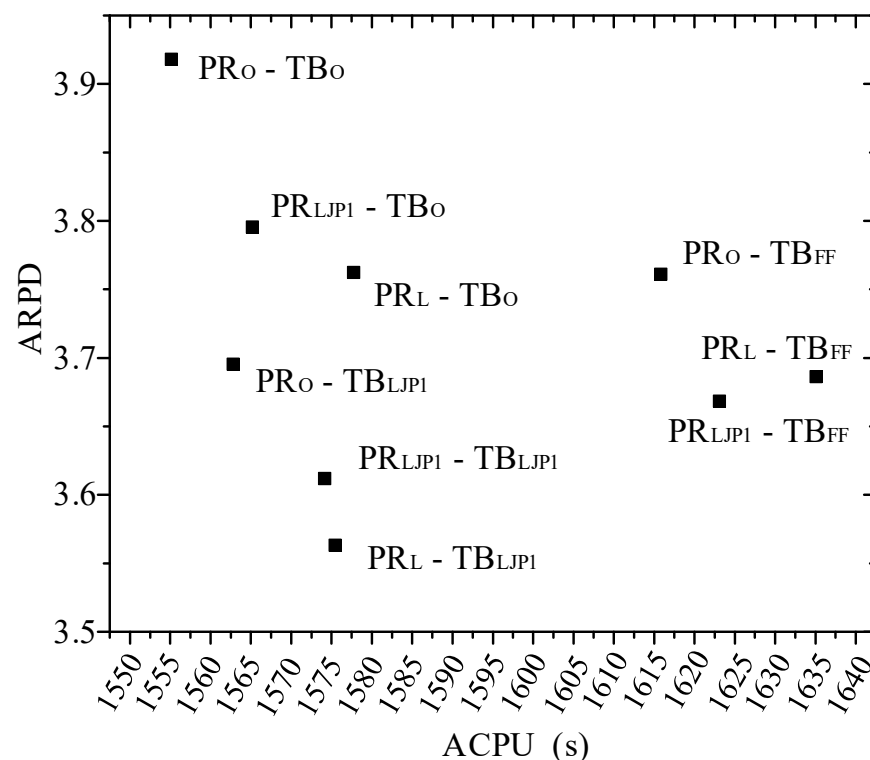


Figure 8. APRD vs. ACT of each algorithm on VRF benchmark.

4. Conclusions

This paper proposes a discrete-event-simulation-based method for batch production of multiple satellite models based on a common manufacturing platform, which is a practical PFSP. We combine the advantages of discrete-event simulation modelling and

heuristic algorithms and propose a more practical method, including a flexible modelling methodology for the PFSP simulation meta-model and an improved NEH algorithm. The computational results demonstrate that the new heuristic outperforms the best NEH-based heuristics reported in the literature in terms of solution quality, and this is a 9.18% improvement over the initial NEH algorithm and a 1.40% improvement over the best current improved NEH algorithm. This means that a better and more practical approach is proposed for the problem of satellite batch production in the flowshop. Furthermore, this approach is generally applicable to similar flowshop problems in other fields.

The main difference between our method and the traditional mathematical method lies in the innovative calculation approach of related objective functions. The reported literature on NEH-based algorithms uses a mathematical model to perform calculations, but this paper collects and processes data by running a simulation model. Secondly, the simulation model is simple to reconstruct, so it is ready to add or change various constraints according to an actual problem. Under these constraints, running the simulation model can easily obtain various production data, such as makespan and total flow time, which are difficult to achieve with traditional mathematical methods. In addition, the simulation model is straightforward to use and maintain because of the object-oriented nature of the simulation meta-model, so the ordinary worker can use software packages developed based on this method after simple training.

Although this paper proposes a more pragmatic approach, we have not been able to provide a more robust case study due to the pace of batch production planning. The limitations of the algorithms and models in this paper also lie in the fact that the computation time is too long compared to mathematical methods. Therefore, this method is currently not applicable to optimisation scenarios with high real-time requirements. How to improve the speed of optimisation by combining simulations and algorithms is also one of the important problems that needs to be solved in the future.

Author Contributions: Material preparation, data collection, and analysis were performed by G.L. The entire research work was planned by L.Z., and he completed the review and revision of the article before submission. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no formal funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The raw data supporting the conclusions of this paper will be made available by the authors on reasonable request.

Acknowledgments: We would like to thank the journal editors and reviewers, whose suggestions have greatly improved this paper.

Conflicts of Interest: Authors Guangzhen Li and Lei Zhang were employed by the company Chang Guang Satellite Technology Co., Ltd. Both authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Abbreviations

NEH	Three initials: Nawaz, M., Jr.; E.E.E.; and Ham, I.A.
PFSP	Permutation flowshop scheduling problem
AVG	Average processing time
STD	Standard deviation
SKE	Skewness
KUR	Kurtosis
RPD	Relative percentage deviation
ACT	Average CPU time

References

- Jiang, Y.H.; Li, Q.; Li, W.K.; Li, F. Improvement of leak detection technology for small satellite mass production. *Spacecr. Environ. Eng.* **2020**, *37*, 81–88. [\[CrossRef\]](#)
- Xing, Q. Analysis on construction and influence of large-scale LEO constellation. *Aerosp. China* **2019**, *42*, 43–47.
- Hao, J.; Yang, H.; Zeng, C. Research on construction of batch intelligent production line for micro/nano satellite. In Proceedings of the 6th International Conference on Signal and Information Processing, Networking and Computers, Guiyang, China, 13–16 August 2019; Springer Nature: Singapore, 2020; pp. 226–236. [\[CrossRef\]](#)
- Juan, G.P. Study of the Benefits and Applications of LEO (Low Earth Orbit) for Communications and Definition of Space New Business Models: Oneweb Case. Bachelor's Thesis, Universitat Politècnica de Catalunya, Barcelona, Spain, 2021.
- Zhou, Y.; Liang, C.; Ji, J. A Full Lifecycle-Oriented Flexible Method for Satellite Design. In Proceedings of the Wireless and Satellite Systems: 12th EAI International Conference, Harbin, China, 31 July–2 August 2021; Springer Nature: Singapore, 2022; pp. 135–141. [\[CrossRef\]](#)
- Qiu, T.; Zhang, Q.; Guo, L. Study on the Application of Robot-Assisted Assembly in Satellite Board Assembly. In Proceedings of the 10th International Conference on Signal and Information Processing, Networking and Computers, Xi'ning, China, July 2022; Springer Nature: Singapore, 2023; pp. 858–865. [\[CrossRef\]](#)
- Kalczynski, P.J.; Kamburowski, J. An empirical analysis of the optimality rate of flow shop heuristics. *Eur. J. Oper. Res.* **2009**, *198*, 93–101. [\[CrossRef\]](#)
- Liu, W.; Jin, Y.; Price, M. A new improved neh heuristic for permutation flowshop scheduling problems. *Int. J. Prod. Econ.* **2017**, *193*, 21–30. [\[CrossRef\]](#)
- Anjana, V.; Sridharan, R.; Kumar, P.N.R. Metaheuristics for solving a multi-objective flow shop scheduling problem with sequence-dependent setup times. *J. Sched.* **2020**, *23*, 49–69. [\[CrossRef\]](#)
- Della Croce, F.; Grosso, A.; Salassa, F. Minimizing total completion time in the two-machine no-idle no-wait flow shop problem. *J. Heuristics* **2019**, *27*, 159–173. [\[CrossRef\]](#)
- Jiang, E.-D.; Wang, L. An improved multi-objective evolutionary algorithm based on decomposition for energy-efficient permutation flow shop scheduling problem with sequence-dependent setup time. *Int. J. Prod. Res.* **2018**, *57*, 1756–1771. [\[CrossRef\]](#)
- Thürer, M.; Ma, L.; Stevenson, M. Workload Control order release in general and pure flow shops with limited buffer size induced blocking: An assessment by simulation. *Int. J. Prod. Res.* **2020**, *59*, 2558–2569. [\[CrossRef\]](#)
- Benda, F.; Braune, R.; Doerner, K.F.; Hartl, R.F. A machine learning approach for flow shop scheduling problems with alternative resources, sequence-dependent setup times, and blocking. *OR Spectr.* **2019**, *41*, 871–893. [\[CrossRef\]](#)
- Laribi, I.; Yalaoui, F.; Belkaid, F.; Sari, Z. Heuristics for solving flow shop scheduling problem under resources constraints. *IFAC-PapersOnLine* **2016**, *49*, 1478–1483. [\[CrossRef\]](#)
- De Sousa Junior, W.; Barra Montevechi, J.A.; de Carvalho Miranda, R.; Teberga Campos, A.T. Discrete simulation-based optimization methods for industrial engineering problems: A systematic literature review. *Comput. Ind. Eng.* **2019**, *128*, 526–540. [\[CrossRef\]](#)
- Razali, F.; Nawawi, A. Optimization of Permutation Flowshop Scheduling Problem (PFSP) using First Sequence Artificial Bee Colony (FSABC) Algorithm. *Prog. Eng. Appl. Technol.* **2024**, *5*, 369–377.
- Ding, C.; Qiao, F.; Zhu, G. Solving a many-objective PFSP with reinforcement cumulative prospect theory in low-volume PCB manufacturing. *Neural Comput. Appl.* **2023**, *35*, 20403–20422. [\[CrossRef\]](#)
- Gajsek, B.; Marolt, J.; Rupnik, B.; Lerher, T.; Sternad, M. Using maturity model and discrete-event simulation for industry 4.0 implementation. *Int. J. Simul. Model.* **2019**, *18*, 488–499. [\[CrossRef\]](#)
- Gomes, L.; Natário, D.; Costa, A.; Barros, J.-P.; Campos-Rebello, R. Event-Based Modeling of Input Signal Behaviors for Discrete-Event Controllers. *Appl. Sci.* **2024**, *14*, 5289. [\[CrossRef\]](#)
- Yang, S.; Xu, Z.; Li, G.; Wang, J. Assembly transport optimization for a reconfigurable flow shop based on a discrete event simulation. *Adv. Prod. Eng. Manag.* **2020**, *15*, 69–80. [\[CrossRef\]](#)
- Zhang, Z.; Wang, X.; Wang, X.; Cui, F.; Cheng, H. A simulation-based approach for plant layout design and production planning. *J. Ambient. Intell. Humaniz. Comput.* **2019**, *10*, 1217–1230. [\[CrossRef\]](#)
- Li, G.; Yang, S.; Xu, Z.; Wang, J.; Ren, Z.; Li, G. Resource allocation methodology based on object-oriented discrete event simulation: A production logistics system case study. *CIRP J. Manuf. Sci. Technol.* **2020**, *31*, 394–405. [\[CrossRef\]](#)
- Amorim, G.A.; Lopes, L.A.S.; Junior, O.S.S. Discrete event-based railway simulation model for eco-efficiency evaluation. *Int. J. Simul. Model.* **2020**, *19*, 375–386. [\[CrossRef\]](#)
- Li, G.; Xu, Z.; Yang, S.; Wang, H.; Bai, X.; Ren, Z. Bottleneck identification and alleviation in a blocked serial production line with discrete event simulation: A case study. *Adv. Prod. Eng. Manag.* **2020**, *15*, 125–136. [\[CrossRef\]](#)
- Yang, S.; Wang, J.; Xin, L.; Xu, Z. Verification of intelligent scheduling based on deep reinforcement learning for distributed workshops via discrete event simulation. *Adv. Prod. Eng. Manag.* **2023**, *17*, 401–412. [\[CrossRef\]](#)
- Krenczyk, D. Deep Reinforcement Learning and Discrete Simulation-Based Digital Twin for Cyber-Physical Production Systems. *Appl. Sci.* **2024**, *14*, 5208. [\[CrossRef\]](#)
- Chowdhury, N.I.; Gopalakrishnan, B.; Adhikari, N.; Li, H.; Liu, Z. Evaluating Electrification of Fossil-Fuel-Fired Boilers for Decarbonization Using Discrete-Event Simulation. *Energies* **2024**, *17*, 2882. [\[CrossRef\]](#)

28. Kim, S.; Choi, Y.; Kim, S. Simulation Modeling in Supply Chain Management Research of Ethanol: A Review. *Energies* **2023**, *16*, 7429. [[CrossRef](#)]
29. Tang, H.T.; Chen, M.; Jiang, W.G. Simulation of flow shop sequence-dependent group scheduling problem. *Light Ind. Mach.* **2015**, *33*, 56–64.
30. Nawaz, M.; Enscore, E.E.; Ham, I. A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *Omega* **1983**, *11*, 91–95. [[CrossRef](#)]
31. Zhou, J.P. *Simulation of Production System*; Electronic Industry Press: Beijing, China, 2011.
32. Ištoković, D.; Perinić, M.; Doboviček, S.; Bazina, T. Simulation framework for determining the order and size of the product batches in the flow shop: A case study. *Adv. Prod. Eng. Manag.* **2019**, *14*, 166–176. [[CrossRef](#)]
33. Allaoui, H.; Artiba, A. Integrating simulation and optimization to schedule a hybrid flow shop with maintenance constraints. *Comput. Ind. Eng.* **2004**, *47*, 431–450. [[CrossRef](#)]
34. Dong, X.; Huang, H.; Chen, P. An improved neh-based heuristic for the permutation flowshop problem. *Comput. Oper. Res.* **2008**, *35*, 3962–3968. [[CrossRef](#)]
35. Fernandez-Viagas, V.; Framinan, J.M. On insertion tie-breaking rules in heuristics for the permutation flowshop scheduling problem. *Comput. Oper. Res.* **2014**, *45*, 60–67. [[CrossRef](#)]
36. Vallada, E.; Ruiz, R.; Framinan, J.M. New hard benchmark for flowshop scheduling problems minimising makespan. *Eur. J. Oper. Res.* **2015**, *240*, 666–677. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.