applied sciences

MDPI

Article

A Data-Driven Robust Scheduling Method Integrating Particle Swarm Optimization Algorithm with Kernel-Based Estimation

Peng Zheng [1], Peng Zhang [2], Ming Wang [3] and Jie Zhang [2,*]

[1] School of Mechanical Engineering, Shanghai Jiao Tong University, Shanghai 200240, China; pengzheng@sjtu.edu.cn
[2] Institute of Artificial Intelligence, Donghua University, Shanghai 201620, China; zhangp88@dhu.edu.cn
[3] College of Mechanical Engineering, Donghua University, Shanghai 201620, China; 1199075@mail.dhu.edu.cn
* Correspondence: mezhangjie@dhu.edu.cn

Abstract: The assembly job shop scheduling problem (AJSSP) widely exists in the production process of many complex products. Robust scheduling methods aim to optimize the given criteria for improving the robustness of the schedule by organizing the assembly processes under uncertainty. In this work, the uncertainty of process setup time and processing time is considered, and a framework for the robust scheduling of AJSSP using data-driven methodologies is proposed. The framework consists of obtaining the distribution information of uncertain parameters based on historical data and using a particle swarm optimization (PSO) algorithm to optimize the production schedule. Firstly, the kernel density estimation method is used to estimate the probability density function of uncertain parameters. To control the robustness of the schedule, the concept of confidence level is introduced when determining the range of uncertain parameters. Secondly, an interval scheduling method constructed using interval theory and a customized discrete PSO algorithm are used to optimize the AJSSP with assembly constraints. Several computational experiments are introduced to illustrate the proposed method, and these were proven effective in improving the performance and robustness of the schedule.

Keywords: robust scheduling; particle swarm optimization algorithm; kernel density estimation; interval theory

Citation: Zheng, P.; Zhang, P.; Wang, M.; Zhang, J. A Data-Driven Robust Scheduling Method Integrating Particle Swarm Optimization Algorithm with Kernel-Based Estimation. *Appl. Sci.* **2021**, *11*, 5333. https://doi.org/10.3390/app11125333

Academic Editor: Paolo Renna

Received: 2 May 2021
Accepted: 31 May 2021
Published: 8 June 2021

1. Introduction

Scheduling is one of the core issues in production management, which is very important for improving the production efficiency and resource utilization of the manufacturing system [1]. In real-world manufacturing systems, different types of uncertainties naturally appear in the production process, causing fluctuations in the production performance [2,3]. In this context, the robustness of a schedule is of great importance in addition to the schedule's quality. With the rapid development of the industrial informatics technologies, a large amount of historical data are now routinely captured and collected in many enterprises. Historical data can be used to analyze the characteristics of uncertainty, and this provides a new view for robust scheduling under uncertainty [4].

The motivation of this study is derived from scheduling problems encountered by many assembly manufacturing enterprises [5,6], where a variety of different products are obtained through multiple components by multistage machining and assembly processes (shown in Figure 1). Due to the tree-like assembly relationships, higher level components cannot be assembled until all preceding lower levels components are completed. Therefore, compared with a single machining process, fluctuations in the processing time and setup time of any part may cause the waiting of subsequent and parallel components, and this can ultimately lead to the delay of the final product [7]. In order to ensure sufficient lead time for preparing production auxiliary materials and tools, the manager should decide on the complete schedule in advance. In most real-life manufacturing systems, managers employ

dispatching rules and optimization methods to generate a baseline schedule, and they then insert extra time redundancy into the schedule to ensure robustness [8,9]. However, the assessment of uncertainty in this schedule is subjective and largely depends on the experience of the manager [10].

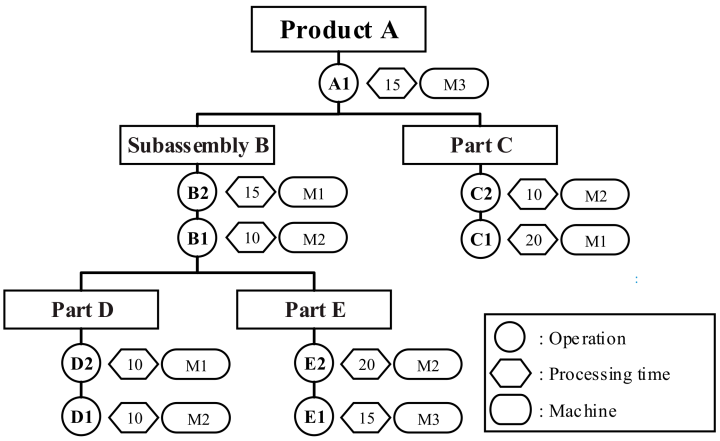

Figure 1. Tree-like assembly structure of product A.

In real-life manufacturing systems, schedules are often confronted with uncertain factors. For example, resource shortages and machine breakdowns can delay a schedule's completion time. In this work, we mainly consider the uncertainty of process setup time and processing time, which are also the two most frequent and influential uncertainties in the assembly production system. To hedge against uncertainties in the production process, traditional optimization methods, such as stochastic programming (SP), fuzzy programming (FP), and robust optimization (RO), have proposed different mathematical expressions of uncertainty [11–13]. The sources and types of uncertain parameters considered by these methods are different. For example, the FP method assumes that the uncertain parameters belong to a specific perturbation interval under different probabilities, and it establishes a fuzzy set based on the membership function to describe the uncertain scenarios. The SP method establishes a representative scenario tree by enumerating the possible values of uncertain parameters. These methods usually combine a priori reasoning with assumption on the uncertainty to motivate the construction of the optimization models. However, in the real world, the probability distribution of uncertainty is closely related to the specific production environment, and some complex uncertainties are even difficult to describe explicitly. Thus, traditional optimization methods are often criticized for their inflexibility or ineffectiveness in practical applications [14].

In addition to traditional optimization methods, many metaheuristic algorithms have been extensively applied to shop scheduling problems under uncertainty [15]. Among them, the particle swarm optimization algorithm is one of the most commonly used metaheuristic methods because of its flexibility and global optimization capabilities [16,17]. Han et al. [18] studied integrated production planning and scheduling with fuzzy startup time and processing time. They developed a fuzzy bi-level decision-making technique based on the PSO algorithm and a heuristic method. Jamrus et al. [19] developed a hybrid approach integrating the particle swarm optimization algorithm with genetic operators for solving a fuzzy job shop scheduling problem with an uncertain processing time. Panadero et al. [20] considered a permutation flow shop problem with stochastic processing times. They proposed a biased-randomized heuristic algorithm and extended it to a metaheuristic method by adding a variable neighborhood descent framework. Obviously, in the above research, different uncertain parameters were described and modeled by fuzzy and stochastic theories. In other studies, uncertain parameters are usually regarded as bounded uncertainties, which can be defined as a continuous interval or a finite set of discrete values. Feng et al. [21] studied a similar problem but proposed a min–max regret

scheduling model with interval scenarios of processing times. They designed both exact and heuristic algorithms to solve this problem. Wang et al. [22] considered a job-shop scheduling problem with an uncertain processing time, described by discrete scenarios. Lu et al. [23] addressed a single-machine scheduling problem with uncertain processing times and sequence-dependent setup times.

All the previous studies focused on the classic flow-shop or job-shop scheduling problems under uncertainty and did not consider the assembly process. In fact, the production process with both machining and assembly is very commonly witnessed in real life. Some examples include production flows of molds, ball valves, and various complex electromechanical products [24,25]. Normally, production scheduling that considers the assembly process is more complicated, and the schedule is more sensitive to the effects of uncertainty. In addition, most of the previous studies define uncertainty based on prior assumptions or artificial experience, rather than from an objective data-driven perspective. With the recently emerging industrial informatics technologies such as the Internet of Things (IoT), sensors, and artificial intelligence, the historical data of the production process can be continuously collected and can be a new source for scheduling optimization [26,27]. These data have motivated a shift from a priori reasoning and assumptions to a new data-driven paradigm in dealing with uncertainties in the production process, which is also the focus of this research.

In the actual production process, fluctuations in the setup time and processing time of the process may cause the completion time to deviate from the expected time, and this can ultimately lead to delays in subsequent processes. Motivated by realistic needs, this paper considers two types of uncertain parameters that have an important impact on the production scheduling of the assembly manufacturing system, namely process processing time and setup time, and from there we aim to develop a data-driven robust scheduling framework. Briefly, the proposed framework consists of statistical analysis and interval scheduling methods used to describe the distribution of uncertainty parameters. On this basis, a PSO algorithm considering assembly constraints is designed to hedge against the risk of system performance degradation in uncertain environments.

The rest of this paper is organized as follows. In Section 2, we give the problem description and the notations. In Section 3, considering that the prior distribution of process setup time and processing time is unknown, we design a nonparametric estimation method based on kernel density estimation to obtain the probability density function of these two parameters. The proposed particle swarm optimization algorithm based on interval scheduling is then elaborated in detail, which is followed by a series of computational experiments and discussion. A summary covering the conclusions is presented in the last section.

2. Problem Description

Suppose there are several types of products with tree-like assembly structures to be produced. Each product is composed from a set of components. Those basic components produced with only machining operations are termed *parts*, while those assembled from other parts are termed *subassemblies*. The term *job* refers to a part or a subassembly. A final product is assembled from a set of subassemblies and parts with a sequential constraint relationship in a tree-like operation diagram. The operation tree example of a product is shown in Figure 1. The nodes in the tree operation represent the operations, while the edges represent the partial order relationship of the operation constraints. An assembly operation can only start after all required parts are completed. Each individual operation can be processed by multiple alternative machines/assembly group. For convenience, we call the above problem an assembly job shop scheduling problem.

Other assumptions made are listed as follows:

- Each operation can be machined/assembled by at most one machine/assembly group at a time;
- Each machine/assembly group can handle at most one operation at a time;

- Only one alternative assignment is chosen for any operation;
- Any started operation cannot be interrupted before it is finished;
- All parts and machines are available at the beginning of the scheduling period.

In the production process, various uncertain factors need to be considered in the scheduling problem. The robustness of a schedule means that it is insensitive to various uncertainties, given a certain control policy. In this study, we assumed that managers adopt the most commonly used right-shift policy [28] to hedge against the effect of uncertainty on the baseline schedule. We used s to represent a feasible schedule that specifies the operations assigned to each machine and the corresponding sequence. Let $C_{max}(s,\ \xi)$ denote the maximum completion time of schedule s under an uncertain parameter $\xi \epsilon U$. We considered two uncertain parameters, namely, the uncertain processing time and the setup time, and we assumed that their probability density functions can be obtained from historical data. By combining these data with the a priori structural features of uncertainties, we constructed a mathematical model for AJSSP. Some notations used for formulating the mathematical model are listed in Table 1.

Table 1. Parameters and variables for modeling.

	Parameters		**Variables**
I	Index set of the operations	C_{max}	Maximum completion time
M	Index set of the machines	ST_i	Start time of operation $i,\ i \in I$
I_m	Set of operations that are processed in machine $m,\ m \in M$	X_{ij}	Binary decision variable, $i, j \in I$
CA	Set of pairs $(i,\ j)$ that the predecessor $i$of, $i, j \in I$		
PT_i	Nominal processing time of operation $i,\ i \in I$		
SU_i	Nominal processing time of operation $i,\ i \in I$		
$\widetilde{PT}_i$	Uncertain processing time of operation $i,\ i \in I$		
$\widetilde{SU}_i$	Uncertain setup time of operation $i,\ i \in I$		
K	Big value		

The model is presented as follows:

$$\text{minimize } C_{max}(s,\ \xi) \tag{1}$$

subject to

$$C_{max}(s,\ \xi) \geq ST_i + \widetilde{SU}_i + \widetilde{PT}_i \qquad \forall i \in I \tag{2}$$

$$ST_j - ST_i \geq \widetilde{SU}_i + \widetilde{PT}_i \qquad \forall (i,\ j) \in \mathbf{CA} \tag{3}$$

$$ST_i - ST_j + KX_{ij} \geq \widetilde{SU}_j + \widetilde{PT}_j \qquad \forall i,\ j \in I_m, \forall m \tag{4}$$

$$ST_j - ST_i + K(1 - X_{ij}) \geq \widetilde{SU}_i + \widetilde{PT}_i \qquad \forall i,\ j \in I_m, \forall m \tag{5}$$

$$ST_i \geq 0 \qquad \forall i \in I \tag{6}$$

$$X_{ij} \in \{0, 1\} \qquad \forall i,\ j \in I_m, \forall m \tag{7}$$

The binary decision variable X_{ij} is 1 if operation i is scheduled before j, and 0 otherwise. C_i, which is the completion time of operation i, is $ST_i + \widetilde{SU}_i + \widetilde{PT}_i$, the right-hand side of Equation (2). Since we are interested in minimizing the makespan, i.e., the completion time of the final operation, the objective function is $\max_{i \in I} C_i$, which is equivalent to Equations (1) and (2). Equation (3) guarantee that each operation can only start after the related processing operations and subassembly operations are finished. Equations (4) and (5) ensure that every single machine can only process one operation simultaneously.

Obviously, the above mathematical model is a kind of semi-infinite problem and cannot be solved directly, since it has an infinite number of constraints due to the uncertain set U to which ξ belongs. Once the interval times of the schedule are determined, this model can be solved by linear programming methods, such as robust linear programming. In this research, a novel data-driven method, which combines metaheuristic algorithm and statis-

tical analysis technology, is used to solve the above AJSSP under uncertainty. The details of this method are introduced in the next section.

3. Data-Driven Robust Scheduling Method

The proposed data-driven robust scheduling method includes two parts: an uncertain parameter analysis based on kernel density estimation technology and an interval scheduling optimization based on a particle swarm optimization algorithm. First, we estimate the probability density function of uncertain parameters based on the kernel density estimation method, and we adjust the range of the interval number through the confidence level. Second, we design an interval scheduling method that takes the interval number of uncertain parameters as input and optimizes the interval makespan through the PSO algorithm.

3.1. Kernel-Based Estimation of Uncertain Parameters

In the actual production process, uncertain parameters are usually regarded as bounded uncertainties, and managers often define the bounds of uncertain parameters based on experience. In the absence of domain-specific knowledge, the analysis of historical data provides a practical way to characterize uncertain intervals. For example, the upper and lower bounds of the interval can be specified as the maximum and minimum of data samples. However, this method may be too conservative because the worst-case scenarios at the boundary are very unlikely to occur in practice.

In this paper, we decompose the uncertain parameters $\widetilde{SU}_i$ and $\widetilde{PT}_i$ in the following forms:

$$\widetilde{SU}_i = SU_i + \xi_{1,i}\hat{SU}_i \tag{8}$$

$$\widetilde{SU}_i = SU_i + \xi_{1,i}\hat{PT}_i \tag{9}$$

where SU_i and PT represent the nominal value of the uncertain parameter; $\hat{SU}_i$ and $\hat{PT}_i$ denote the magnitude of the random perturbation, which can be approximated by the bounds of the samples; ξ_i is a normalized random variable, which controls the range of uncertainties. By specifying the interval of ξ_i, inherent uncertain parameters can be transformed to bounded uncertainties that are easy to model mathematically.

In recent years, some researchers have proposed to adopt statistical analysis and machine learning methods to learn the distributional information from the data and then establish a data-driven uncertainty set to reduce conservatism. As a well-known nonparametric estimation approach, kernel density estimation (KDE) has proven to be an effective tool for modeling probability density distribution [29]. Assume that we have the data of uncertain parameters drawn i.i.d. according to distribution $\mathbb{P}$. In this paper, we propose to use the confidence region of the probability density function to quantify the uncertain interval by KDE.

Given historical data of uncertain parameters, KDE learns the probability density function of the samples according to

$$p(x) = \frac{1}{nh}\sum_{i=1}^{n}\varphi\left(\frac{x - x_i}{h}\right) \tag{10}$$

where $p(x)$ denotes the estimated density function of uncertain parameter x; h is a positive parameter representing the bandwidth; n is the number of samples; $\varphi(\cdot)$ is a kernel function, which satisfies

$$\varphi(u) \geq 0 \tag{11}$$

$$\int \varphi(u)du = 1 \tag{12}$$

In our research, the Gaussian function commonly used in engineering problems is introduced as the kernel function, and the value of bandwidth h is determined through pilot experiments.

Based on the probability density functions of uncertainties, the confidence interval of the uncertain values can be calculated through interval estimation. To reduce the conservatism of the uncertain interval, we introduce an adjustable parameter σ to control the number of uncertain values that lie within the uncertain interval. Taking the uncertain parameter $\widetilde{SU}$ as an example, the corresponding σ can be described as

$$\sigma_i = \frac{\hat{SU}_i^{1-\eta}}{\hat{SU}_i^{max}} \tag{13}$$

where $\hat{SU}_i^{1-\eta}$ represents the value of $\hat{SU}_i$ under the confidence level $1-\eta$, which can be calculated by the quartile function of $\widetilde{SU}_i$ [30], and $\hat{SU}_i^{max}$ denotes the maximum range of uncertainty. When the confidence level $1-\eta$ equals 1, all of the sampled points are contained in the uncertain interval and $\hat{SU}_i^{1-\eta} = \hat{SU}_i^{max}$. Conversely, as the value of $1-\eta$ falls to 0, fewer samples are considered (see Figure 2). Thus, the region of $\widetilde{SU}_i$ determined by σ_i can be described as:

$$\widetilde{SU}_i = SU_i + \xi_{1,i}\hat{SU}_i^{max} \quad \xi_{1i} \in [-\sigma_i, \sigma_i] \tag{14}$$

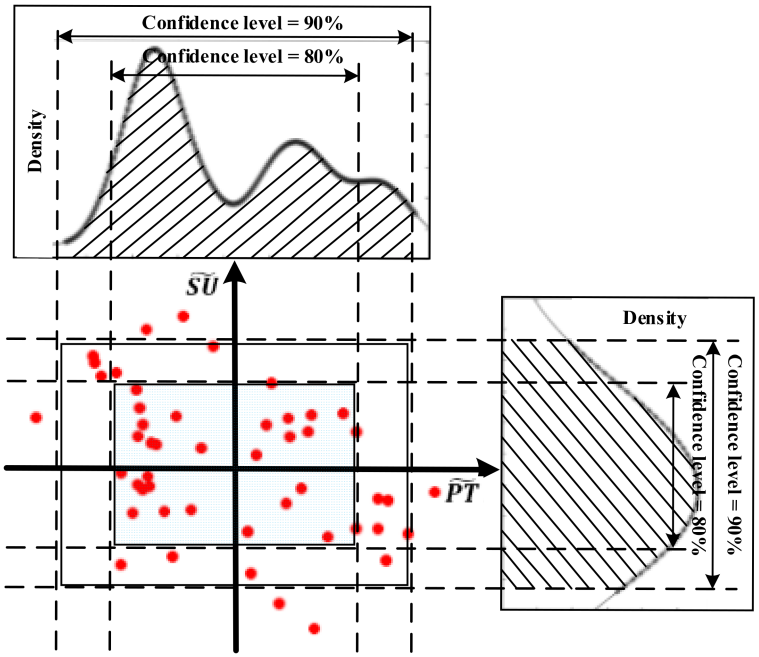

Figure 2. Data-driven uncertain interval of uncertain parameters.

In the following optimization algorithm, the above data-driven uncertain interval is used to assist the encoding and decoding procedure.

3.2. Data-Driven PSO Algorithm with Uncertain Parameters

Over the past two decades, a large number of metaheuristic algorithms have been developed to solve production scheduling problems with complicated constraints. In this paper, we address the AJSSP with uncertain parameters, where the objectives cannot be evaluated in the form of traditional deterministic fitness functions. To solve this problem, we propose using a data-driven interval schedule to approximate objective functions under the effect of uncertainties. On this basis, this paper proposes a data-driven particle swarm optimization (DPSO) algorithm framework (see Figure 3). In DPSO, we improved the original PSO algorithm by adding the Metropolis operator and genetic operator.

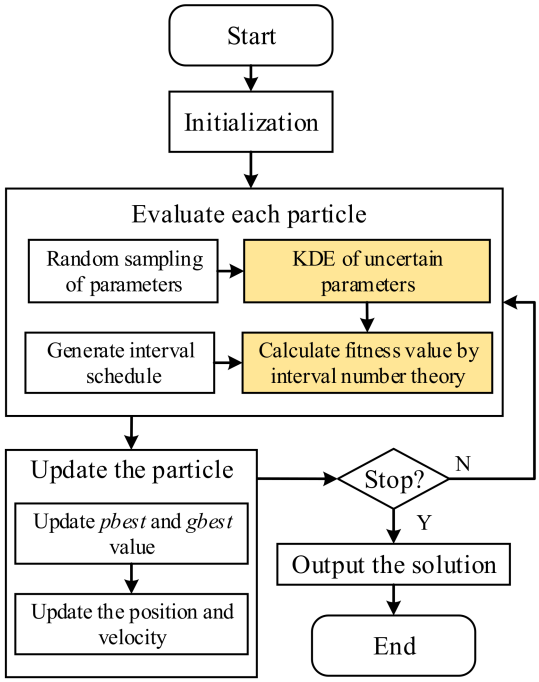

Figure 3. Diagram of the proposed DPSO algorithm framework.

3.2.1. Encoding Scheme of AJSSP

In the design of heuristic algorithms, the term *chromosome* represents the encoding of a solution. A reasonable chromosome representation is the primary task for the successful application of heuristic algorithms to solve actual production scheduling problems. In order to make the PSO algorithm applicable to the discrete AJSSP, we propose a discrete chromosome coding and initialization method based on the adjacency matrix to represent sequential constraints of the product.

To make the description about the adjacency matrix easier to follow, we assume that one product (i.e., product A) will be scheduled, and its assembly structure is shown in Figure 1. First, we establish the corresponding adjacency matrix $A = (a_{ij})_{n\times n}$ of product A to represent the sequential constraints (see Figure 4). The row and column numbers of the matrix A represent the numbers of all parts/subassemblies to be scheduled, and A_0 is a virtual node that represents the end of assembly. In the adjacency matrix A, if the value of element a_{ij} is not equal to 0, it means that A_i is a lower level part of A_j, i.e., A_j can only start after all operations of A_i are completed. The value of a_{ij} represents the number of operations of part A_i that have not yet been processed. In the initialization and repair process of DPSO, one can continuously search and update the adjacency matrix A to avoid generating an infeasible schedule.

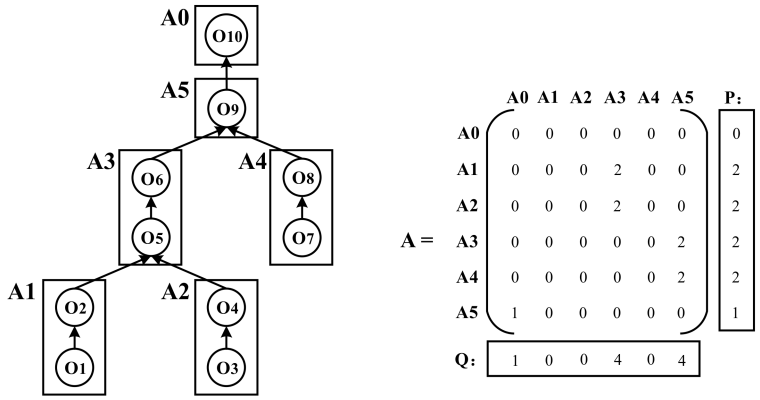

Figure 4. Adjacency matrix of product A.

A discrete chromosome representation decodes a schedule of AJSSP with a positive integer string O, which defines the permutation of jobs, i.e., parts and subassemblies. The O-string consists of a sequence of job numbers. The number of job i occurs n_i times in the O-string, where n_i is the number of operations included in job i. Therefore, the lengths of the O-string is equals to the sum of all operations of all jobs to be scheduled. Table 2 shows a feasible chromosome of product A. In the decoding process, the O-string is converted to a sequence of operations, and then each operation is assigned to the prespecified machine.

Table 2. A feasible chromosome for product A.

O-string	4	1	2	2	1	3	4	3	5
Operation	O_7	O_1	O_3	O_4	O_2	O_5	O_8	O_6	O_9
Machine	M1	M2	M3	M2	M1	M2	M2	M1	M3

3.2.2. Decoding Scheme Based on Interval Schedule

By using the KDE-based estimation for historical data, the processing time and setup time of each operation can be represented as interval number $\widetilde{PT}_i = \left[\underline{PT_i}, \overline{PT_i}\right]$ and $\widetilde{SU}_i = \left[\underline{SU_i}, \overline{SU_i}\right]$. The left and right bounds of the interval number can be calculated by Equation (13) and adjusted by setting different confidence levels. After that, we can construct an interval schedule based on interval theory to obtain the interval makespan.

Unlike the scheme of generating a deterministic schedule, interval theory redefines basic operation rules of interval numbers. In this study, we focus on the addition and comparison rules involved in the production scheduling process. Assuming that A and B are two interval numbers, A^L/B^L and A^R/B^R represent the left and right limits, respectively. The addition and comparison rules of A and B are defined as follows [31]:

$$A + B = \left[A^L + B^L,\ A^R + B^R\right] \tag{15}$$

$$P(A \le B) = \begin{cases} 0 & A^L \ge B^R \\ 0.5 * \frac{B^R - A^L}{A^R - A^L} * \frac{B^R - A^L}{B^R - B^L} & B^L \le A^L \le B^R \le A^R \\ \frac{B^L - A^L}{A^R - A^L} + 0.5 * \frac{B^R - B^L}{A^R - A^L} & A^L \le B^L \le B^R \le A^R \\ \frac{B^L - A^L}{A^R - A^L} + \frac{A^R - B^L}{A^R - A^L} * \frac{B^R - A^R}{B^R - B^L} + 0.5 * \frac{A^R - B^L}{A^R - A^L} * \frac{A^R - B^L}{B^R - B^L} & A^L \le B^L \le A^R \le B^R \\ \frac{B^R - A^R}{B^R - B^L} + 0.5 * \frac{A^R - A^L}{B^R - B^L} & B^L \le A^L \le A^R \le B^R \\ 1 & A^R \le B^L \end{cases} \tag{16}$$

In the decoding process, Equation (15) issed to calculate the completion time of an operation, and Equation (16) is used to compare the release time of different machines.

Based on an O-string, a corresponding interval schedule can be obtained through the above operation rules. By comparing the interval numbers for C_i of different operations, the interval makespan C_{max} can be finally obtained.

3.2.3. Particle Update Method

In this study, we design a discrete PSO algorithm to realize the discretization of the position update process through operations such as insertion and exchange. In addition, the Metropolis criterion in the simulated annealing algorithm is used to set the acceptance probability of the location update.

To ensure the diversity of the initial population, we use a random generation method to generate initial particles. In the PSO algorithm, the particles are updated according to the interaction of particle velocity, the best-found location, and the global best-found position. In each iteration step, the position and the velocity of the i-th particle are updated in the following formulas:

$$X_i = \begin{cases} g(w \otimes h(X_i),\ pB_i), & if\ RAND < c \\ g(w \otimes h(X_i),\ \mathrm{g}B_i), & else \end{cases} \tag{17}$$

where pB_i is the best-found location by particle i up to this step; $\mathrm{g}B_i$ is the global best-found position among all particles up to this step; w is the inertia weight, which is used to control whether to execute $h(\cdot)$; $RAND$ represents a random number between [0,1], and c is a preset positive constant. The inertia weight w is expressed in the form of probability, and $w \otimes h(X_i)$ means that the algorithm executes $h(X_i)$ with the probability of w. The design of the functions $h(X)$ and $g(X, Y)$ refers to the commonly used genetic operations. Specifically, the function $h(X)$ exchanges the values of two random positions in the particle X, and the function $g(X, Y)$ is defined as the crossover operation in genetic algorithm.

In addition, we introduce the Metropolis criterion to set the particle update probability in the following formulas:

$$\triangle_i = f\left(X_i^{'}\right) - f(\ pB_i) \tag{18}$$

where $X_i^{'}$ represents the updated new position of particle i, and $f(X_i)$ is the fitness function value corresponding to particle i at position X_i. If $\triangle_i < 0$, update the new position of particle i to $X_i^{'}$. Otherwise, update the position of i with the probability $exp(-\triangle_i / f(\ pB_i))$.

In order to ensure the feasibility of the generated solutions, DPSO uses the adjacency matrix to identify whether the new solution is feasible after each operator is executed. If the solution is infeasible, the algorithm re-executes the operator until a feasible solution is obtained.

4. Experimental Study

To test the performance of the proposed data-driven robust scheduling method, a series of calculation experiments was performed on 2 AJSSP benchmarks and 8 cases originating from an aerospace structure parts workshop in Shanghai. Among them, Cases 1 and 2 were derived from two AJSSP examples used in Shi et al. [6]. We modified the above two cases by introducing random processing time and setup time. The features of these cases are shown in Table 3.

All the experiments were performed in Visual Studio 2017 on a personal computer with Intel Core I7-6700 2.6 GHz CPU and 4 GB RAM.

Table 3. Characteristics of test instances.

Instance	No. of Operations	No. of Machines	No. of Assembly Levels
1	18	4	3
2	33	4	3
3	86	5	4
4	92	5	5
5	125	6	5
6	148	8	5
7	164	8	5
8	170	9	5
9	176	9	6
10	185	10	6

4.1. Setting Algorithm Parameters

As it is generally known, the performance of metaheuristic algorithms is greatly affected by the values of the algorithm parameters. To investigate the statistically significance of the parameter values of the algorithm, we performed an analysis of variance (ANOVA) on the obtained experimental results. In this study, 4 key parameters were considered, and each parameter contained 3 alternative value levels. Specifically, the population size included 3 levels, namely 100, 150, and 200; the inertia weight w included 3 levels, namely 0.7, 0.8, and 0.9; the constant c includes three levels, namely 0.3, 0.5, and 0.7; the function g in Equation (17) can choose three crossover operation modes, namely SPX, PMX, and OX [32]. The result of the ANOVA is shown in Table 4.

Table 4. ANOVA results for the obtained results of RPD.

	Sum of Squares (SS)	df	Mean Square (MS)	F	Sig.
A. population size	0.049	2	0.025	4.235	0.019
B. inertia weight w	0.059	2	0.030	5.111	0.009
C. constant c	0.053	2	0.027	4.577	0.014
D. function g	0.041	2	0.020	3.510	0.036

In Table 4, *SS* means the sum of squares for the factor, which reflects the degree of variation between sample means. The abbreviation df means the degree of freedom of the factor. Here, df = k − 1, and k is the number of value levels of the factor. *MS* means the mean square between groups for the factor, and MS = SS/df. The label "F" is the test statistic in the ANOVA, which is equal to the ratio of the mean square between groups and the mean square within groups. The greater the value of F, the greater the difference between the groups, or in other words, the more significant the influence of this factor on the results.

According to Table 4, since the p-value (sig.) is less than the significance level (0.05), the above four parameters all have significant impact on the experimental results. Taking the relative percent difference (RPD) of the makespan as the metric, Figure 5 shows the corresponding means plot and the least significant difference (LSD) intervals of inertia weight w at the 95% confidence level for the RPD metric. When the inertia weight w is set to 0.8, the value of the RPD metric is the smallest, and thus its value is set to 0.8. Similarly, the value of population size is set to 200, while the value of constant c is set to 0.5, and the function g is set to be the SPX mode.

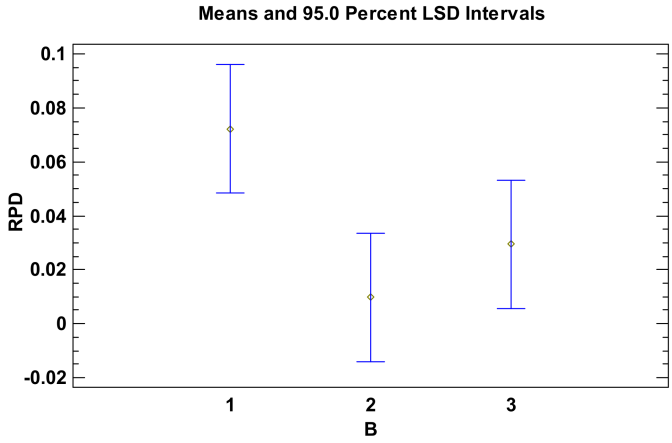

Figure 5. Means plot at the 95% confidence level LSD intervals for RPD metric.

4.2. Robustness of Interval Schedules under Different Confidence Levels

In the proposed method, the confidence level of uncertain parameters is an important tool to control the robustness of the schedule. Based on Equation (15), different confidence levels correspond to different bounds of intervals, which make the schedule flexible to fluctuations of production parameters. Taking Case 1 as an example, the product structure and the original deterministic processing time PT_i are detailed in Shi et al. [6]. In this study, the uncertain processing times $\widetilde{PT}_i$ were assumed to follow Gaussian distribution with the mean of PT_i and the standard deviation $\sigma = 0.2 * p_0$. Here, p_0 represents the standard processing time of the operation. We obtained the optimized solution (see Figure 6) of Case 1 in the deterministic scenario and in the uncertain scenario with a 80% confidence level. As can be seen from Figure 6b, each operation in the interval schedule was represented by two intervals. Taking operation 4 as an example, the lower line O4_S indicated the interval start time of operation 4, and the upper line O4_E indicated the interval completion time, which was obtained by adding the interval processing time to the interval start time.

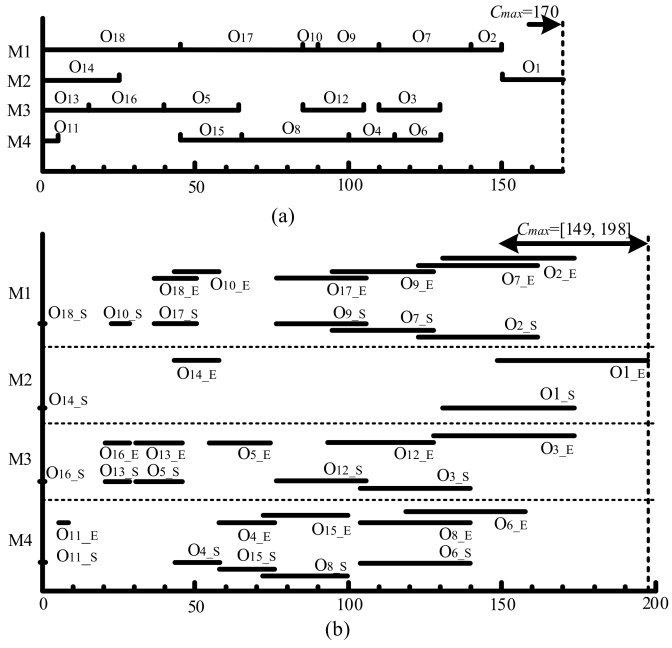

Figure 6. Gantt chart of the optimal solution for Case 1. (**a**) Optimal solution on deterministic scenario; (**b**) Optimal solution on uncertain scenario with 80% confidence level.

Compared with the schedule for deterministic scenario, although the makespan of interval schedule increased, it is obvious that each operation reserves appropriate time redundancy based on its historical data, which is more suitable for the actual production process. From another perspective, the result also proves that the uncertainty of the processing time may cause the maximum 16.5% fluctuation in the makespan of Case 1, which can provide a reference for the delivery date of the product. In addition, the proposed method can help managers flexibly adjust the conservativeness and robustness of the schedules by setting different confidence levels. We tested the performance of interval schedules under different confidence levels and reported the results of Cases 1–5 in Figure 7.

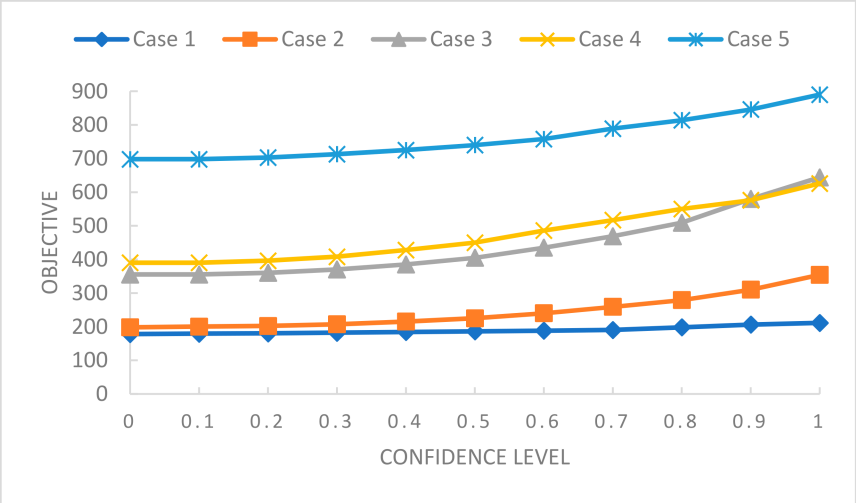

Figure 7. Objective values of different confidence levels.

In Figure 7, when the confidence level was equal to 0, all parameters became deterministic values, and the interval scheduling at this time was equivalent to deterministic scheduling. With the continuous improvement of the confidence level, the interval of uncertain parameters covers more uncertain scenarios, and the value of the objective also increases. Taking Case 1 as an example, the objective value of confidence level 0 was 178, and the value of confidence level 1 was 211. This meant that the uncertainty of the processing time and setup time may cause a performance deviation of 18.5% in the makespan. In addition, for Cases 3–10, the average fluctuation of the objective value caused by different confidence levels of the processing time and setup time is 15–20%, which is basically consistent with the actual situation of the workshop.

4.3. Performance Comparison of Different Optimization Algorithms

To verify the performance of the proposed algorithm, the DPSO algorithm was compared with two classic heuristic algorithms, namely the simulated annealing algorithm (SA) and the genetic algorithm (GA) [7]. GA is a typical population-based heuristic algorithm, while SA belongs to another type of heuristic algorithm based on a single individual. We chose these two algorithms for comparison because they are the prototypes of many evolutionary algorithms of the same type, and they have shown excellent performance in many workshop scheduling problems. In addition, we also added the original PSO algorithm for comparison. The GA, SA, and PSO algorithms used the same interval-based decoding scheme as that of DPSO, and the algorithm parameters were also optimized by the ANOVA. To make a fair comparison, all algorithms used the interval numbers of uncertainty parameters under the 80% confidence interval as input, and the cut-off time of the algorithms was set to 120 s. The above three algorithms were applied on each instance with 10 independent runs. In order to comprehensively compare the stability and superiority of different algorithms, we recorded and reported the average value (denoted as Avg) and the best value (denoted as Best) of the makespan on each instance. In addition,

the Avg/Best makespan of each algorithm was compared with the Avg/Best makespan of all comparison algorithms (denoted as $Avg^*/Best^*$) from the relative percentage error (RPE). The RPE can be denoted as follows:

$$\text{RPE} = \frac{Best^* - Best}{Best^*} \times 100\% \tag{19}$$

The computational results in Table 5 show that the proposed DPSO algorithm is significantly better than the SA and GA. The pair of numbers under each case denote the scale of the case. For example, (92 × 5) means that this case contains 92 processes that need to be processed on 5 machines. For Cases 1 and 2, due to the small scale of these problems, the above three algorithms can stably obtain the same optimal solution. However, as the scale of the problem continues to increase, GA and DPSO are superior to SA in terms of optimization capability and algorithm stability. Specifically, for Cases 3 and 5, GA and DPSO can obtain the same optimal solution and perform better than SA on the same CPU time. However, even for these two cases, the average values of DPSO are 3.9% and 1.3% smaller than those of GA, which show that the stability of the proposed algorithm is better.

Table 5. Comparisons of SA-MAE and the four existing algorithms for the AJSSP on different scenarios.

Instance	Algorithm	*Avg*	RPE_A (%)	*Best*	RPE_B (%)
Case 1 (18 × 4)	SA	**[149, 198]**	0	**[149, 198]**	0
	GA	**[149, 198]**	0	**[149, 198]**	0
	PSO	**[149, 198]**	0	**[149, 198]**	0
	DPSO	**[149, 198]**	0	**[149, 198]**	0
Case 2 (33 × 4)	SA	**[216, 279]**	0	**[216, 279]**	0
	GA	**[216, 279]**	0	**[216, 279]**	0
	PSO	**[216, 279]**	0	**[216, 279]**	0
	DPSO	**[216, 279]**	0	**[216, 279]**	0
Case 3 (86 × 5)	SA	[501.5, 553.9]	−4.7	[462, 515]	−1.2
	GA	[490.3, 549.5]	−3.9	**[452, 509]**	0
	PSO	[479.5, 541.2]	−2.3	**[452, 509]**	0
	DPSO	**[471.6, 528.8]**	0	**[452, 509]**	0
Case 4 (92 × 5)	SA	[539.4, 599.6]	−5.3	[499, 563]	−2.3
	GA	[518.7, 583.5]	−2.5	[494, 555]	−0.9
	PSO	[512.4, 573.4]	−0.7	**[487, 550]**	0
	DPSO	**[506.3, 569.4]**	0	**[487, 550]**	0
Case 5 (125 × 6)	SA	[795.8, 874.2]	−4.0	[757, 827]	−1.6
	GA	[775.4, 851.2]	−1.3	**[739, 814]**	0
	PSO	[772.3, 847.5]	−0.9	**[739, 814]**	0
	DPSO	**[762.2, 840.3]**	0	**[739, 814]**	0
Case 6 (148 × 8)	SA	[721.6, 788.8]	−8.3	[670, 738]	−5.6
	GA	[692.6, 764.3]	−4.9	[647, 719]	−2.8
	PSO	[673.2, 749.4]	−2.9	[641, 711]	−1.7
	DPSO	**[658.4, 728.6]**	0	**[627, 699]**	0
Case 7 (164 × 8)	SA	[833.5, 907.1]	−8.5	[783, 861]	−5.8
	GA	[807.3, 889.3]	−6.4	[767, 849]	−4.3
	PSO	[786.3, 869.8]	−4.1	[745, 827]	−1.6
	DPSO	**[756.8, 835.7]**	0	**[734, 814]**	0
Case 8 (170 × 9)	SA	[891.9, 984.8]	−14.1	[818, 903]	−7.8
	GA	[820.4, 909.4]	−5.3	[784, 868]	−3.6
	PSO	[811.5, 896.2]	−3.8	[773, 859]	−2.5
	DPSO	**[779.6, 863.4]**	0	**[752, 838]**	0
Case 9 (176 × 9)	SA	[871.5, 970.0]	−9.3	[816, 904]	−5.9
	GA	[814.2, 912.4]	−2.8	[797, 886]	−3.7
	PSO	[812.6, 907.6]	−2.3	[788, 880]	−3.1
	DPSO	**[794.7, 887.2]**	0	**[763, 854]**	0
Case 10 (185 × 10)	SA	[1069.2, 1177.9]	−11.4	[985, 1094]	−6.4
	GA	[998.2, 1116.3]	−5.9	[972, 1076]	−4.7
	PSO	[988.5, 1091.2]	−3.2	[961, 1068]	−3.9
	DPSO	**[953.7, 1057.4]**	0	**[924, 1028]**	0

For more complex cases (Cases 6–10), compared with the SA and GA, DPSO obtains all the optimal solutions among the three algorithms. In terms of optimization capability, the best values of optimal solutions obtained by DPSO are 3.8% and 6.3% better than GA

and SA, respectively. In terms of algorithm stability, the objectives of DPSO are on average 5.1% and 10.3% better than GA and SA, respectively. Considering the similar structure of the population-based optimization algorithms, the computational results of GA and DPSO show that the proposed encoding and initialization method based on the adjacency matrix are effective. The discretized particle update method is also more advantageous than conventional genetic operations.

In addition, we compared the performance of the original PSO algorithm and the improved DPSO algorithm. For Cases 1–5, although the PSO also obtained the same best values of optimal solutions, the algorithm is not stable enough in 10 independent runs. For Cases 6–10, DPSO is obviously better than PSO in all respects. Specifically, the best values and average value of optimal solutions are increased by 2.6% and 3.3%, respectively. This result also proves the effectiveness of the proposed particle update method. In summary, the proposed DPSO algorithm is competitive in performance and stability, especially on larger-scale problems.

5. Conclusions

Assembly job shop is a common production organization mode in discrete manufacturing, and its production scheduling is more complicated than a single processing workshop. This paper proposes a data-driven robust scheduling method for the AJSSP with uncertain production parameters, which can generate adjustable robust schedule. The improvements of the proposed method consist of three points: (1) using the kernel density estimation method to obtain the probability density function of uncertain parameters, and introducing the confidence level to adjust the bounds of uncertain parameters; (2) establishing a decoding scheme based on interval scheduling to realize the combination of interval numbers of uncertain parameters and schedule; (3) designing a discrete PSO algorithm to deal with the assembly constraints and optimize solutions. The experimental results show that the proposed method can effectively improve the performance and robustness of the schedule.

The improvement of the digital level of manufacturing enterprises is the key to applying new data analysis methods to solve traditional scheduling problems. Taking into account the complexity of the actual production environment, this study only considers the uncertainty of the processing time and setup time in the production process. Future research work can be divided into two aspects: (1) considering more types of uncertain parameters according to the actual production environment; and (2) incorporating more prior information and characteristics of uncertain parameters into the scheduling method to further improve the effectiveness and robustness of the schedule.

Author Contributions: Conceptualization, P.Z. (Peng Zheng) and J.Z.; methodology, P.Z. (Peng Zheng) and P.Z. (Peng Zhang); software, M.W.; validation, M.W. and P.Z. (Peng Zheng); formal analysis, P.Z. (Peng Zheng); investigation, M.W.; resources, P.Z. (Peng Zhang); data curation, M.W.; writing—original draft preparation, P.Z. (Peng Zheng); writing—review and editing, P.Z. (Peng Zheng), P.Z. (Peng Zhang) and J.Z.; visualization, P.Z. (Peng Zheng); supervision, J.Z.; project administration, P.Z. (Peng Zhang); funding acquisition, P.Z. (Peng Zhang). All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Program of the National Natural Science Foundation of China under Grant No. 52005099, National Key R&D Program of China under Grant No. 2019YFB1706300, and the Fundamental Research Funds for the Central Universities.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Zou, P.; Rajora, M.; Liang, S.Y. Multimodal Optimization of Permutation Flow-Shop Scheduling Problems Using A Clustering-Genetic-Algorithm-Based Approach. *Appl. Sci.* **2021**, *11*, 3388. [CrossRef]
2. Ning, C.; You, F. A data-driven multistage adaptive robust optimization framework for planning and scheduling under uncertainty. *Aiche J.* **2017**, *63*, 4343–4369. [CrossRef]
3. Li, W.; He, L.; Cao, Y. Many-Objective Evolutionary Algorithm With Reference Point-Based Fuzzy Correlation Entropy for Energy-Efficient Job Shop Scheduling With Limited Workers. *IEEE Trans. Cybern.* **2021**. [CrossRef]
4. Bertsimas, D.; Gupta, V.; Kallus, N. Data-driven robust optimization. *Math. Program.* **2018**, *167*, 235–292. [CrossRef]
5. Pereira, M.T.; Santoro, M.C. An integrative heuristic method for detailed operations scheduling in assembly job shop systems. *Int. J. Prod. Res.* **2011**, *49*, 6089–6105. [CrossRef]
6. Zheng, P.; Wang, J.; Zhang, J.; Yang, C.; Jin, Y. An Adaptive CGAN/IRF-Based Rescheduling Strategy for Aircraft Parts Remanufacturing System under Dynamic Environment. *Robot. Comput. Integr. Manuf.* **2019**, *58*, 230–238. [CrossRef]
7. Shi, F.; Zhao, S.; Meng, Y. Hybrid algorithm based on improved extended shifting bottleneck procedure and GA for assembly job shop scheduling problem. *Int. J. Prod. Res.* **2019**, *58*, 2604–2625. [CrossRef]
8. Mehta, S.V.; Uzsoy, R.M. Predictable scheduling of a job shop subject to breakdowns. *IEEE Trans. Robot. Autom.* **1998**, *14*, 365–378. [CrossRef]
9. O'Donovan, R.; Uzsoy, R.; McKay, K.N. Predictable scheduling of a single machine with breakdowns and sensitive jobs. *Int. J. Prod. Res.* **1999**, *37*, 4217–4233. [CrossRef]
10. Al-Hinai, N.; El Mekkawy, T.Y. Robust and stable flexible job shop scheduling with random machine breakdowns using a hybrid genetic algorithm. *Int. J. Prod. Econ.* **2011**, *132*, 279–291. [CrossRef]
11. Hu, Z.; Hu, G. A multi-stage stochastic programming for lot-sizing and scheduling under demand uncertainty. *Comput. Ind. Eng.* **2018**, *119*, 157–166. [CrossRef]
12. Jia, Z.; Leung, Y.T. Ant colony optimization algorithm for scheduling jobs with fuzzy processing time on parallel batch machines with different capacities. *Appl. Soft Comput.* **2019**, *75*, 548–561. [CrossRef]
13. Wang, Z.; Pang, C.K.; Ng, T.S. Robust scheduling optimization for flexible manufacturing systems with replenishment under uncertain machine failure disruptions. *Control. Eng. Pract.* **2019**, *92*, 104094. [CrossRef]
14. Janak, S.L.; Floudas, C.A. Advances in robust optimization approaches for scheduling under uncertainty. *Comput. Aided Chem. Eng.* **2005**, *20*, 1051–1056.
15. Stastny, J.; Skorpil, V.; Balogh, Z. Job Shop Scheduling Problem Optimization by Means of Graph-Based Algorithm. *Appl. Sci.* **2021**, *4*, 1921. [CrossRef]
16. Cao, Y.; Zhang, H.; Li, W. Comprehensive Learning Particle Swarm Optimization Algorithm with Local Search for Multimodal Functions. *IEEE Trans. Evol. Comput.* **2019**, *23*, 718–731. [CrossRef]
17. Gao, K.Z.; Cao, Z.; Zhang, L. A Review on Swarm Intelligence and Evolutionary Algorithms for Solving Flexible Job Shop Scheduling Problems. *IEEE/CAA J. Autom. Sin.* **2019**, *4*, 904–916. [CrossRef]
18. Han, J.; Liu, Y.; Luo, L. Integrated production planning and scheduling under uncertainty: A fuzzy bi-level decision-making approach. *Knowl. Based Syst.* **2020**, *201*, 106056. [CrossRef]
19. Jamrus, T.; Chien, C.F.; Gen, M. Hybrid Particle Swarm Optimization Combined With Genetic Operators for Flexible Job-Shop Scheduling Under Uncertain Processing Time for Semiconductor Manufacturing. *IEEE Trans. Semicond. Manuf.* **2018**, *31*, 32–41. [CrossRef]
20. Panadero, J.; Villarinho, P.A.; Pessoa, L.S. A Simheuristic Algorithm for the Stochastic Permutation Flow-Shop Problem with Delivery Dates and Cumulative Payoffs. *Int. Trans. Oper. Res.* **2020**, *28*, 716–737.
21. Feng, X.; Zheng, F.; Xu, Y. Robust scheduling of a two-stage hybrid flow shop with uncertain interval processing times. *Int. J. Prod. Res.* **2016**, *54*, 3706–3717. [CrossRef]
22. Wang, B.; Wang, X.; Lan, F. A hybrid local-search algorithm for robust job-shop scheduling under scenarios. *Appl. Soft Comput.* **2018**, *62*, 259–271. [CrossRef]
23. Lu, C.C.; Lin, S.W.; Ying, K.C. Minimizing worst-case regret of makespan on a single machine with uncertain processing and setup times. *Appl. Soft Comput.* **2014**, *23*, 144–151. [CrossRef]
24. Na, H.; Park, J. Multi-level job scheduling in a flexible job shop environment. *Int. J. Prod. Res.* **2014**, *52*, 3877–3887. [CrossRef]
25. Zhang, S.; Li, X.; Zhang, B. Multi-objective optimisation in flexible assembly job shop scheduling using a distributed ant colony system. *Eur. J. Oper. Res.* **2020**, *283*, 441–460. [CrossRef]
26. Zheng, P.; Zhang, P.; Wang, J. A data-driven robust optimization method for the assembly job-shop scheduling problem under uncertainty. *Int. J. Comput. Integr. Manuf.* **2020**, 1–16. [CrossRef]
27. Zhang, M.; Tao, F.; Nee, A. Digital Twin Enhanced Dynamic Job-Shop Scheduling. *J. Manuf. Syst.* **2020**, *58*, 146–156. [CrossRef]
28. Abumaizar, R.J.; Svestka, J.A. Rescheduling job shops under random disruptions. *Int. J. Prod. Res.* **1997**, *35*, 2065–2082. [CrossRef]
29. Sheather, S.J.; Jones, M.C. A reliable data-based bandwidth selection method for kernel density estimation. *J. R. Stat. Soc.* **1991**, *53*, 683–690. [CrossRef]
30. Zhang, Y.; Feng, Y.; Rong, G. New Robust Optimization Approach Induced by Flexible Uncertainty Set: Optimization under Continuous Uncertainty. *Ind. Eng. Chem. Res.* **2017**, *56*, 270–287. [CrossRef]

31. Jiang, C.; Han, X.; Liu, G.R. A nonlinear interval number programming method for uncertain optimization problems. *Eur. J. Oper. Res.* **2008**, *188*, 1–13. [CrossRef]
32. Juwairiah, J.; Pratama, D.; Rustamaji, H.C. Genetic Algorithm for Optimizing Traveling Salesman Problems with Time Windows (TSP-TW). *Int. J. Artif. Intell. Robot.* **2019**, *1*, 1. [CrossRef]