applied sciences

MDPI

Article

Hybrid Flow Shop Scheduling Problems Using Improved Fireworks Algorithm for Permutation

Xuelian Pang [1,2], Haoran Xue [3], Ming-Lang Tseng [4,5,6,*], Ming K. Lim [7] and Kaihua Liu [1]

1 School of Microelectronics, Tianjin University, Tianjin 300072, China; xpang@tju.edu.cn (X.P.); liukaihua@tju.edu.cn (K.L.)
2 Tianjin Electronic Information College, Tianjin 300350, China
3 School of Artificial intelligence, Hebei University of Technology, Tianjin 300401, China; alanfev0101@gmail.com
4 Institute of Innovation and Circular Economy, Asia University, Taichung 41354, Taiwan
5 Department of Medical Research, China Medical University Hospital, China Medical University, Taichung 40402, Taiwan
6 Faculty of Economics and Management, Universiti Kebangsaan Malaysia, Bangi 43600, Malaysia
7 Supply Chain Sustainability and Strategy within the Centre for Business in Society, Coventry University, Coventry CV1 5FB, UK; ac2912@coventry.ac.uk
* Correspondence: tsengminglang@gmail.com or tsengminglang@asia.edu.tw

Received: 4 January 2020; Accepted: 6 February 2020; Published: 10 February 2020

Abstract: Prior studies are lacking which address permutation flow shop scheduling problems and hybrid flow shop scheduling problems together to help firms find the optimized scheduling strategy. The permutation flow shop scheduling problem and hybrid flow shop scheduling problems are important production scheduling types, which widely exist in industrial production fields. This study aimed to acquire the best scheduling strategy for making production plans. An improved fireworks algorithm is proposed to minimize the makespan in the proposed strategies. The proposed improved fireworks algorithm is compared with the fireworks algorithm, and the improvement strategies include the following: (1) A nonlinear radius is introduced and the minimum explosion amplitude is checked to avoid the waste of optimal fireworks; (2) The original Gaussian mutation operator is replaced by a hybrid operator that combines Cauchy and Gaussian mutation to improve the search ability; and (3) An elite group selection strategy is adopted to reduce the computing costs. Two instances from the permutation flow shop scheduling problem and hybrid flow shop scheduling problems were used to evaluate the improved fireworks algorithm's performance, and the computational results demonstrate the improved fireworks algorithm's superiority.

Keywords: permutation flow shop scheduling problem; hybrid flow shop scheduling problem; improved fireworks algorithm; scheduling; makespan

1. Introduction

Effectively solving scheduling problems is a topic in the optimization field [1]. There are scheduling problems in all walks of life, such as medical resource allocation, power system scheduling, wireless network optimization, and electric vehicle scheduling [2–6]. Production scheduling is a common scheduling type and an important part of industry optimization scheduling. A good scheduling strategy contributes to production efficiency and enables the entire production process to be executed successfully. Production scheduling has been widely studied to consider the increasing scale of production and intense market competition [7].

In the process of production scheduling, the producers arrange a set of available production resources to complete the production plan. During the process, the jobs are assigned to the corresponding

machines. The job sequence and operation time on each machine are determined to achieve one or more performance indexes [8]. Flow shop scheduling is a common type of production scheduling that has been adopted in many large-scale industrial productions. The permutation flow shop scheduling problems (PFSP) is a type of typical combinatorial optimization problem that exists widely in the field of industrial automation [9]. The permutation flow shop is regarded as a simplified model of numerous actual flow production lines, and it is suitable for mass production. A hybrid flow shop consists of multiple parallel machines at each stage, where the processing situation is more complicated than the permutation flow shop [10]. As the extension of the PFSP, the hybrid flow shop scheduling problem (HFSP) can flexibly adapt to the actual production conditions [11]. The solution methods for these two types of problems help production firms find the optimal scheduling strategy. In view of this, two kinds of problems were studied.

The flow shop scheduling problem (FSP) was developed for focusing on mathematical programming, multi-objective optimization, etc. [12]. These methods are often used to find the optimal solution, while it is insufficient to handle large-scale issues [13]. It is shown that the FSPs belong to a class of NP-hard problems when the scale is more than three and the optimization algorithms with polynomial time have not yet been found [14]. For these typical scheduling problems, it is difficult to obtain satisfactory results by the traditional optimization methods. Finding efficient methods for solving FSPs has always been an important topic in academia and industry. The intelligent algorithm obtains the optimal schedule in a short time by searching the neighborhood of the solution and improving the current solution continuously. At present, the trend of study is to solve FSPs with intelligent algorithms. The fireworks algorithm (FWA) is an intelligent algorithm inspired by the rule of fireworks explosion and has shown good performance in solving complex optimization problems [15]. Through the explosion and mutation of fireworks, the FWA has strong global search ability, but also has disadvantages such as premature convergence and cumbersome calculation. In this study, an improved fireworks algorithm (IFWA) is put forward for enhancing the performance of FWA and solving the two scheduling problems.

The improvement involves three processes, including explosion, mutation, and selection, compared with the original algorithm. The results confirm the performance of the improved algorithm by solving two instances from the FSP. The rest of the study is structured as follows: In Section 2, the relevant approaches for FSP and improvement strategies for the FWA are introduced. Section 3 introduces the FWA and IFWA. In Section 4, the FSP and HFSP are described in detail. In Section 5, two instances are given to test the IFWA's performance in solving the FSP and the results are discussed. Finally, the conclusions and future studies are discussed.

2. Literature Review

Methods for solving FSPs are divided into three categories: exact algorithms, construction algorithms, and intelligent algorithms. Exact algorithms accurately obtain the optimal schedule, while they are greatly affected by the scale of the problem and the amount of calculation. Exact algorithms mainly include integer programming, the branch and bound method, and other methods. Wang et al. [16] used the branch and bound algorithm to study a class of two-stage HFSPs below 20 jobs. However, the applicability of the method is limited as the number of jobs increases. Xu et al. [17] proposed different mixed integer programming equations to solve an FSP with up to 100 jobs. In a sense, these methods belong to the enumeration method and unnecessary calculations are excluded based on the operation results. However, the computational cost of these algorithms increases exponentially with the increase of jobs. In general, exact algorithms are suitable for small-scale scheduling problems, while they are insufficient to solve large-scale scheduling problems. Heuristic algorithms for obtaining a feasible or the suboptimal solution have been rapidly developed, considering the huge cost of getting an optimal solution. A constructive heuristic algorithm can construct the solution of the problem through certain rules and quickly obtain the solution of the problem.

However, the construction rules are usually complicated, and the solution obtained is not satisfactory. Construction algorithms mainly include the Palmer method, the Nawaz–Enscore–Ham method, the Gupta method, etc. [18–20]. The Johnson rule obtained good results when dealing with an FSP involving two machines [21]. It laid a good foundation for solving FSPs with multiple machines by using heuristic rules later. The Campbell–Dudek–Smith rule is an extended form of the Johnson rule, which is often used to solve scheduling problems involving multiple machines [22]. The Nawaz–Enscore–Ham method is regarded to be the most effective construction algorithm. Kalczynski and Kamburowsk [23] proved that other heuristic algorithms do not perform as well as the Nawaz–Enscore–Ham algorithm in dealing with PFSPs. Semančo and Modrák [24] proposed a new constructive heuristic algorithm named the modified Johnson's algorithm, which applied the Johnson rule and a pair-splitting strategy. The performance of the modified Johnson's algorithm was better than those of three other classic heuristics (Palmer, Campbell–Dudek–Smith, and Gupta), and its computational cost is made much lower by verifying the benchmark function.

A main method of solving FSPs is the use of intelligent optimization algorithms, which can obtain optimal solutions with a high probability in a reasonable time. Several intelligent algorithms, such as the evolutionary algorithm [25], particle swarm optimization (PSO), grey wolf optimizer, firefly algorithm, and ant colony optimization algorithm have been extensively used [26–29]. Dasgupta and Das [30] introduced a heuristic rule to convert the continuous position values into discrete arrangement values in the optimization problem to solve the FSP based on the Cuckoo algorithm. The application of the algorithm can effectively reduce the completion time and average processing time. Marichelvam et al. [31] introduced the scheduling rules and heuristic methods in the monkey search algorithm. The method has been proven suitable to FSP compared with other approaches. Tang et al. [32] adopted the Hill function to redefine the inertia coefficient of the velocity update equation in the PSO, which can avoid the premature convergence. The application of the method can effectively improve energy efficiency in HFSPs. Lin et al. [33] put forward a new crossover and mutation strategy on the traditional backtracking search algorithm, which can improve the diversity of solutions and improve the ability to solve discrete problems. The method can accurately obtain the best completion time of each case by solving the benchmark problem from the flow shop scheduling field. Komaki and Kayvanfar [34] adjusted the lower limit of the workload of each stage and the scheduling rules. Then, the grey wolf optimizer was used to find the optimal job sequence. Jiang and wang [35] proposed a multi-objective evolutionary algorithm to optimize the makespan and energy consumption in PFSPs, and obtained good optimization results. Abdel-Basset et al. [36] enhanced the local search capabilities of the traditional whale optimization algorithm (WOA) and combined it with Nawaz–Enscore–Ham to further enhance the ability of WOA. The hybrid algorithm showed efficient performance in solving the PFSP. In general, the performance of intelligent optimization algorithms has an important impact on the final solution in the optimization process.

The FWA is an intelligent algorithm inspired by the regularity of the explosion of fireworks [37]. Each firework is considered as the solution of the feasible domain and the optimal solution is found through the explosion and mutation process. The exchange of the information and the distribution of the resources between each spark and other sparks in whole population can ensure the algorithm takes into account both global and local search. FWA has good performance for mathematical optimization. This method has been widely used in the photovoltaic energy field, in big data optimization, and in network community testing [38–40].

In recent decades, many improvement methods have been developed to better enhance the performance of the FWA. Ye et al. [41] proposed a new FWA called the firework algorithm local search method and chaotic mutation, and logistic chaotic mutation was introduced to replace Gaussian mutation to maintain the diversity of the population. Yu et al. [42] adopted a differential mutation strategy during the search process. After the mutation and the crossover operation, a selection operator is employed to filter the subgroups. The hybrid algorithm improved the search ability of fireworks particles and the performance was verified by the functions from the evolutionary computation in

2014. In view of the shortcomings of the FWA (e.g., lower convergence speed and high computational cost), Jadoun et al. [43] introduced three operators in the process of fireworks explosion. The mapping range, particle dimension, and search mode of the fireworks were modified to improve the global and local optimization ability. A novel guiding spark combined with the smallest explosion amplitude and adaptive amplitude factor was proposed by Arsic et al. [44]. With the implementation of the algorithm, the explosion amplitude changes constantly. Zheng et al. [45] proposed another adaptive amplitude factor, and the mapping method after the position of the sparks exceeded the feasible region was improved so that the sparks can evenly distribute in the feasible region. In summary, these improvement strategies are beneficial, but they do not emphasize the degree of the mutation or optimize the selection strategy. In this study, three improvements are introduced: a Cauchy–Gauss mutation operator and an elite group selection strategy are used, and the overall performance of FWA is effectively enhanced by introducing a nonlinear radius.

3. Fireworks Algorithm

This section introduces traditional fireworks algorithms and proposes three improvement measures. These improvement measures include: (1) nonlinear radius and radius check; (2) a hybrid mutation operator; (3) an elite group selection strategy.

3.1. The Introduction of Traditional FWA

A flow diagram of the traditional FWA is shown in Figure 1. The specific steps are shown below:

1. N solutions are chosen as the initial fireworks, which are generated randomly in the feasible domain.
2. Evaluate the fitness value of the fireworks. Explosive fireworks and Gaussian fireworks are obtained from the initial fireworks by the explosion and Gaussian operations.
3. Test the termination condition. N sparks would be selected in the all sparks as the initial fireworks in the next iteration if the termination condition is not satisfied.

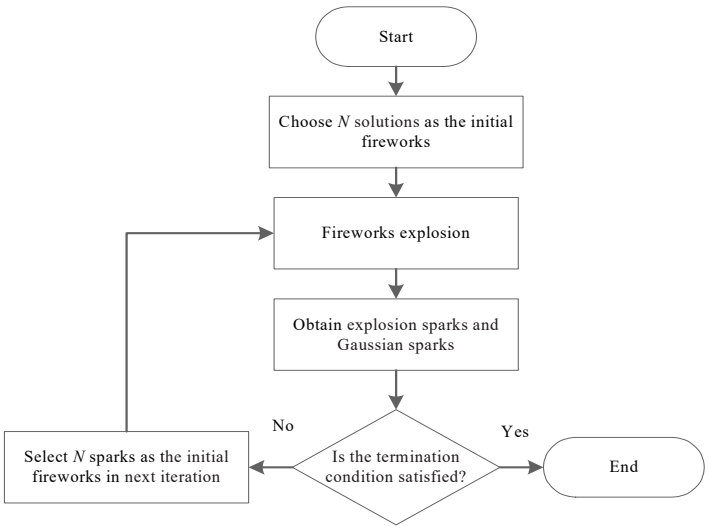

Figure 1. Flowchart of the traditional fireworks algorithm (FWA).

In FWA, each spark is a feasible solution and the process of explosion is regarded as the process of searching for the optimal solution around the neighborhood [46]. The fireworks with worse fitness value have better global search ability, and the fireworks with better fitness value have better local search ability [47]. The explosion operator, the mutation operator, and the selection strategy directly determine the efficiency of this algorithm and whether it can find the global optimum.

(1) Explosion operator

The amplitude of fireworks explosion depends on the fitness value. The better fireworks are much closer to the best solution than the worse fireworks. The better fireworks should generate more explosion sparks in a smaller range. The explosion amplitude is computed in Equation (1):

$$A_i = A \times \frac{f(x_i) - y_{\min} + \varepsilon}{\sum_{i=1}^{N} (f(x_i) - y_{\min}) + \varepsilon} \tag{1}$$

A represents the maximum range of the fireworks explosion, x_i denotes the ith initial firework, $f(x_i)$ represents the fitness value of x_i, $y_{\min} = \min(f(x_i))$ $(i = 1, 2, \cdots, N)$, and ε is the smallest constant to avoid zero-division error.

The number of the explosion sparks also depends on its fitness value. A better value will generate more sparks, while the worse fireworks with worse fitness generate fewer sparks. The number of the explosion sparks of different fireworks is presented as follows:

$$S_i = M \times \frac{y_{\max} - f(x_i) + \varepsilon}{\sum_{i=1}^{N} (y_{\max} - f(x_i)) + \varepsilon} \tag{2}$$

where M is a constant value to restrict the quantity of the explosion sparks, $y_{\max} = \max(f(x_i))$ $(i = 1, 2, \cdots, N)$.

When too many sparks are generated in a small amplitude, the sparks inevitably overlap. Similarly, when the number of fireworks generated is too small, it is not conducive to global search, and the quantity of the explosion sparks is bounded to avoid the situations in Equation (3):

$$S_i = \begin{cases} \text{round}(a \times M), S_i < aM \\ \text{round}(b \times M), S_i > bM, a < b < 1 \\ \text{round}(S_i), \text{otherwise} \end{cases} \tag{3}$$

where a and b are two constants which confine the range of the amount of sparks, and "round" indicates the rounding function.

(2) Mutation operator

The Gaussian operator is used to increase the diversity of the sparks. The process of generating Gaussian sparks is as follows: the specified initial sparks are selected randomly. Moreover, the k dimension of the selected spark is used in the mutation operation, as shown in Equation (4):

$$\hat{x}_{ik} = x_{ik} \times e \tag{4}$$

where $e \sim N(1, 1)$, $\hat{x}_{ik}$ is the k-dimension of the Gaussian spark, x_{ik} is the k-dimension of the explosion spark x_i.

In the process of explosion and mutation, it is inevitable that some of the sparks will transcend the range of the feasible region. When a spark exceeds the boundary at dimension k, the k-dimension of the spark is mapped to a new position by Equation (5):

$$x_{ik} = x_{LB,k} + |\hat{x}_{ik}| \% (x_{UB,k} - x_{LB,k}) \tag{5}$$

where $x_{UB,k}$ and $x_{LB,k}$ are the upper and lower boundaries in the kth dimension, and % is the surplus function.

(3) Selection strategy

Assuming that the number of the fireworks in the current iteration is G and the number of initial fireworks is N, the spark with the best fitness would be selected as one of the initial fireworks in the

next iteration. The remaining $N-1$ sparks would be selected among the $G-1$ sparks. The selection probability of the candidate spark is shown as follows:

$$p(x_i) = \frac{R(x_i)}{\sum\limits_{x_j \in k} R(x_j)} \tag{6}$$

where $R(x_i) = \sum\limits_{x_j \in k} d(x_i - x_j) = \sum\limits_{x_j \in k} \|x_i - x_j\|$.

According to the selection strategy, if the number of fireworks around an individual is smaller, the probability of selecting that individual as the initial firework in the next explosion is greater.

3.2. The Improved Fireworks Algorithm

(1) When calculating the explosion amplitude, the radius of the fireworks with the optimal value is approximately zero, and the explosion fireworks generated by the optimal firework are the greatest in number, which undoubtedly causes a waste of resources. It is necessary to limit the range of fireworks. An exponential decreasing radius is introduced, and the minimal firework radius is replaced when the minimal radius is less than the exponential decreasing radius. The decreasing radius is shown in Equation (7):

$$A_i = \begin{cases} A_i & if\ A_i > A_{\min} \\ A_{\min} = A_{final} \times \left(\frac{A_{inital}}{A_{final}}\right)^{(1-\frac{t}{T})} & otherwise \end{cases} \tag{7}$$

where T and t are the maximal and current number of iterations respectively, and $A_{initial}$ and A_{final} are two parameters to limit the range of $A_{\min}$.

(2) Considering that the Gaussian operator is not sufficient for global search, the Cauchy mutation operator is added in this study. Figure 2 shows the Gaussian and Cauchy distributions. It is evident that the Cauchy distribution drops gently from the peak to the both sides, and a point after Cauchy mutation will be less affected by the local extremum point. In addition, the Cauchy distribution peak is relatively small and the ability for local search is poor, while the opposite is true for the Gaussian distribution.

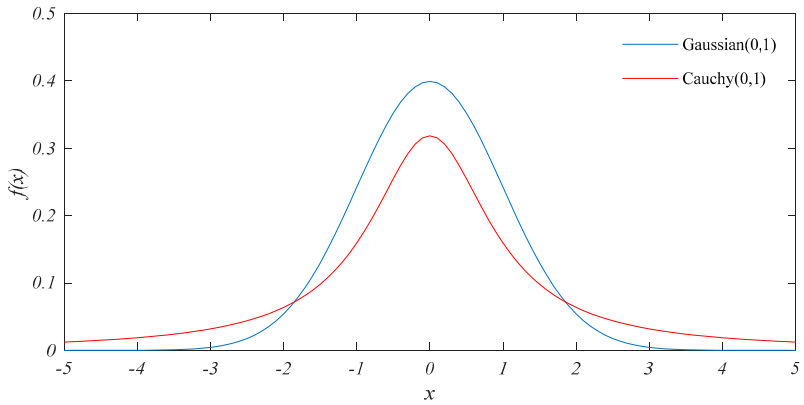

Figure 2. Gaussian and Cauchy distributions.

A Cauchy–Gauss mutation operator is proposed to adjust the role of the two operators dynamically. In the early stage, the mutation operator is mainly used for global search. The mutation ability around the extreme points is strengthened in the later stage. The hybrid mutation operator is shown as follows:

$$\hat{x}_{ik} = x_{ik} \times \left(\frac{t}{T} \cdot Gaussian(1,1) + (1 - \frac{t}{T}) \cdot Cauchy(1,1)\right) \tag{8}$$

(3) In the traditional FWA, the calculation of the Euclidean distance between sparks inevitably reduces the running speed of algorithm when the dimension or the number of sparks is large. This

study proposes an elite group selection strategy. All fireworks generated in the current iteration are sorted from good to bad according to the fitness value, and the first N fireworks are chosen as the initial fireworks in the next generation.

The pseudo-code of IFWA is described as follows (Algorithm 1):

Algorithm 1. The pseudo-code of IFWA.

```
Initialize parameters and the location of the fireworks.
Evaluate the fitness value of the fireworks.
t = 0;
while (t < iterations)
  Generate explosion fireworks
  Calculate A_min by Formula (7).
  for each firework x_i (i = 1, 2, ... , N) do
      Calculate the A_i by Formula (1).
      if A_i < A_min
        A_i = A_min
      end
      Calculate S_i by Formulae (2) and (3).
      for j = 1: S_i
        q = round(d×rand(0,1))      (d is the dimension of sparks).
        Calculate the displacement h = A_i × rand(−1, 1)
        for p = 1:q
          Randomly select kth dimension of x_i (Not repeating).
          x_ik = x_ik + h
          if x_ik is out of bounds then
            Map sparks to a new location by Formula (5)
          end if
        end for
      end for
  end for
  Generate mutation fireworks
  for i = 1:mutation_num (mutation_num is the number of mutation fireworks).
    Randomly select x_i.
    x̂_i = x_i
    q = round(d×rand(0,1)).
    for p = 1:q
      Randomly select the kth dimension of x̂_ik (Not repeating).
      Generate the mutation sparks by Formula (8).
      if x̂_ik out of bounds then
        Map sparks to a new location by Formula (5).
      end if
    end for
  end for
  Select fireworks in the next iteration
  Select N sparks as the initial fireworks in the next iteration by the new selection strategy.
  Update the optimal value.
  t = t + 1
end while
return optimization results
```

4. Problem Description

This section discusses the PFSP and HFSP. The corresponding mathematical models, objective functions, and decoding method are given in detail.

4.1. Permutation Flow Shop Scheduling Problems

The PFSP can generally be described as m jobs to be processed on n machines, where all jobs must be processed through n stages in the same sequence [48]. The makespan is the completion time of the last job m on the last machine n. The layout of PFSP is shown in Figure 3. The assumptions of the PFSP are as follows:

1. Each machine can process only one job at a time;
2. Each job is processed on one machine at a time;
3. The processing sequence of all jobs on different machines is same, which is unknown;
4. The processing sequence of each job on all machines is same, which is known;
5. The processing times of each job on each machine are given;
6. Pre-emption is not allowed.

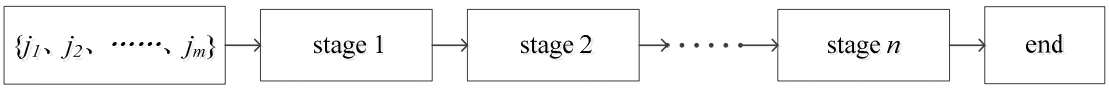

Figure 3. The layout of permutation flow shop scheduling.

In Figure 3, each stage consists of only one machine. $j_1, j_2, \ldots, j_m$ is the processing order of the jobs and j_1 is the first job to be processed. All of jobs are completed when the j_m is completed on stage n. The completion time of job j_i on machine k is noted by $c(j_i, k)$. The objective of the scheduling is to obtain minimal $c(j_m, n)$. The problem is described as:

$$c(j_1, 1) = T_{j_1,1} \tag{9}$$

$$c(j_1, k) = c(j_1, k-1) + T_{j_1,k}, k = 2, \cdots, n \tag{10}$$

$$c(j_i, 1) = c(j_{i-1}, 1) + T_{j_i,1}, i = 2, \cdots, m \tag{11}$$

$$c(j_i, k) = \max\{c(j_{i-1}, k), c(j_i, k-1)\} + T_{j_i,k}, i = 2, \cdots, m; k = 2, \cdots, n \tag{12}$$

where $T_{j_i,k}$ is the processing time of job j_i on machine k. The objective function is described as follows:

$$F = \min c(j_m, n) \tag{13}$$

PFSP is a non-numerical optimization problem in discrete space. This study is necessary to encode the sparks in continuous space to solve the problem. By the integer coding based on the sequence of the jobs, the position of the individual fireworks is encoded as the machining sequence of the jobs.

The PFSP with seven jobs is taken to explain the mapping rule from continuous space to discrete space. If the randomly generated fireworks are sorted by size and the position sequence is (3, 5, 4, 6, 1, 7, 2), the processing order of the jobs is $\{j_3 \rightarrow j_5 \rightarrow j_4 \rightarrow j_6 \rightarrow j_1 \rightarrow j_7 \rightarrow j_2\}$. In this way, the sparks in continuous space are uniquely mapped to a solution in discrete space.

4.2. Hybrid Flow Shop Scheduling Problems

In the HFSP, the m jobs are processed through n stages ($n \geq 1$) and there is at least one stage that has more than one parallel machine [49]. The minimum makespan is chosen as the optimization objective of the HFSP. The assumptions of HFSP are as follows [50]:

1. Each machine can process only one job at a time;

2. Each job is processed on one machine at a time;
3. Each job is processed at each stage in the same order;
4. The parallel machines in each stage are independent;
5. The processing time of the jobs is related to machines and process, and is known;
6. Pre-emption, lot streaming, and backlogging are not allowed [51].

The layout of the HFSP is presented in Figure 4. There are N_j parallel machines at stage j. All jobs are completed when the process stage n of the last job is completed.

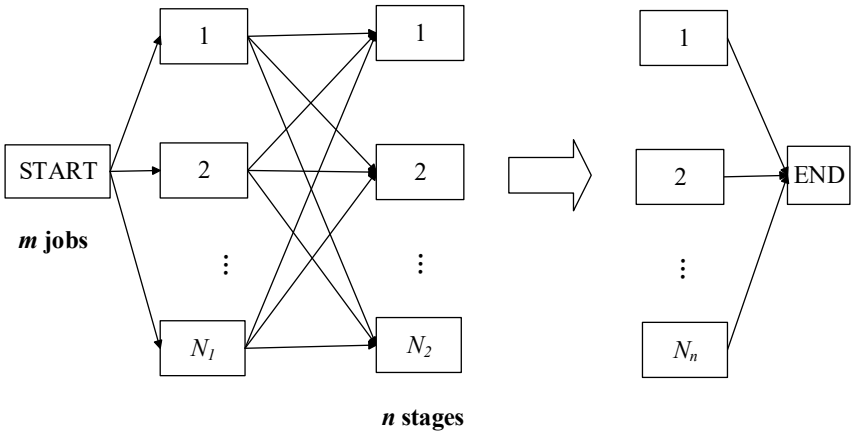

Figure 4. The layout of the hybrid flow shop scheduling problem (HFSP).

The mathematical model of the HFSP is shown from Equations (14) to (16) [52]. The optimization goal is shown in Equation (17).

$$\begin{gathered} C_{ijk} \le S_{i(j+1)r} \quad i = 1,2,\cdots m; \\ j = 1,2,\cdots n;\ k = 1,2,\cdots N_j;\ r = 1,2,\cdots ,N_{j+1} \end{gathered} \tag{14}$$

$$\begin{gathered} C_{h_{jk}} \le S_{(h+1)_{jk}} \quad i = 1,2,\cdots m; \\ j = 1,2,\cdots n; h = 1,2,\cdots N_{jk} - 1; \end{gathered} \tag{15}$$

$$\begin{gathered} C_{ijk} = S_{ijk} + T_{ijk} \quad i = 1,2,\cdots m; \\ j = 1,2,\cdots n; k = 1,2,\cdots N_j; \end{gathered} \tag{16}$$

$$F = \min C_{\max} = \min\{\max\{C_{ink}\}\}, i = 1,2,\cdots m;\ k = 1,2,\cdots N_n; \tag{17}$$

where N_{jk} represents the number of jobs processed on machine k at stage j. h_{jk} indicates the hth job processed on machine k at stage j. T_{ijk} denotes the processing time when job i is processed on machine k at stage j. S_{ijk} and C_{ijk} represent the start and end time of job i on machine k at stage j. $S_{h_{jk}}$ and $C_{h_{jk}}$ represent the start and end time of the hth job on machine k at stage j.

The solution is decoded by scheduling matrix. The scheduling matrix A is composed of m rows and n columns. Moreover, each element in matrix A is a random number limited by the number of parallel machines. The matrix A is depicted as follows.

$$A_{m\times n} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \tag{18}$$

where a_{ij} is a random value between the interval $(1, N_j + 1)$. It shows that the ith job would be processed on machine int(a_{ij}) at stage j, (int(a_{ij}) means the integer components of a_{ij}). Furthermore, the priority

of the jobs is determined by the size of the a_{ij}. An example to determine the process priority of the jobs by matrix $A_{3\times3}$ is shown as follows:

$$A_{3\times3} = \begin{bmatrix} 1.3 & 2.5 & 3.3 \\ 2.5 & 1.8 & 2.8 \\ 2.2 & 1.1 & 1.2 \end{bmatrix}$$

Matrix A shows that three jobs are processed through three process stages and the number of machines in each stage is two, two, and three. According to the stated rules, the first column of matrix A shows that three jobs are processed at stage one. The first job is processed on the first machine at stage one. The second job and the third job are processed on the second machine at stage one. The job with small a_{ij} would be processed preferentially, so the third job would be processed first. From the second column of matrix A, the first job is processed on the second machine at stage two. The second job and the third job are processed on the first machine at stage two, but the third job would be processed first. The processing sequence of the jobs in process stage three are determined similarly.

5. Computational Experiments

Two instances from PFSP and HFSP are selected as experimental objects. The specific description of these examples is shown in detail. The experimental environment was MATLAB 2017a under Windows 7. The frequency of the CPU of the experimental computer was 2.3 GHz, and the memory size was 8 GB. In addition, the original fireworks algorithm, PSO, and the whale optimization algorithm were used to test the performance of IFWA and experimental results are presented as follows.

5.1. Permutation Flow Shop Scheduling Problem

A set of plastic toys consists of seven small toys. Each small toy needs to be processed through seven steps, including melting, shaping, grinding, surface decoration, dyeing, testing, and packaging. The seven jobs are processed by seven machines respectively and the processing time of each small toy is listed in Table 1.

Table 1. Processing time in the permutation flow shop problem (PFSP).

Job	Stages						
	Melting (s)	Shaping (s)	Grinding (s)	Surface Decoration (s)	Dyeing (s)	Testing (s)	Packaging (s)
1	692	310	832	630	258	147	255
2	581	582	14	214	147	753	806
3	475	475	785	578	852	2	699
4	23	196	696	214	586	356	877
5	158	325	530	785	325	565	412
6	796	874	214	236	896	898	302
7	542	205	578	963	325	800	120

This is a classic PFSP with seven jobs and seven machines. The four algorithms ran 20 times independently by taking the makespan as the objective function. The number of iterations was set to 200, and the search range was between 0 and 7. The specific parameters are provided in Table 2. Among them, the specific parameters were taken from the default values of the original algorithm.

The statistical results are given in Table 3. The minimum completion time obtained by the exhaustive method was 6590 s. It is clear that the four algorithms could find the minimum completion time of the problem. The IFWA had the highest success rate, and achieved a significant improvement compared with the other algorithms. The improved algorithm had better stability through the analysis of the average value and standard deviation. Compared to the PSO and WOA, the FWA and the IFWA took a long time because of their structure. The improved algorithm adopts a new selection strategy to avoid calculating the Euclidean distance between particles, which greatly reduces the computational

cost. Compared to the original version, the computation time for IFWA was reduced by nearly 37%. The proposed algorithm had a good optimization performance for solving the PFSP.

Table 2. Parameter settings in PFSP. IFWA: improved fireworks algorithm; PSO: particle swarm optimization; WOA: whale optimization algorithm. lb: lower bound; ub: upper bound.

Algorithm	Parameters	Value
IFWA and FWA	Number of fireworks N	5
	Constant a	0.04
	Constant b	0.8
	A	7
	M	50
	$A_{initial}$	(ub-lb) × 0.02
	A_{final}	(ub-lb) × 0.001
PSO	C_1, C_2	1.49445
	Population size	40
	Range of particle velocity	[–2,2]
WOA	Population size	40
	b	1
	p	0.5

Table 3. Results of the optimizations in PFSP.

Algorithm	Theoretical Optimal Value (s)	Success Rate	Optimal Value (s)	Average (s)	Worst (s)	STD	Time (s)
IFWA	6590	90%	6590	6595.3	6643	16.31	2.14
FWA		75%	6590	6603.2	6643	23.54	3.40
PSO		30%	6590	6665	7016	119.5	0.34
WOA		55%	6590	6614	6643	27.05	0.26

5.2. The Hybrid Flow Shop Scheduling Problem

The HFSP is more complex than the PFSP. There are $P = \prod_{i=1}^{n} n_i^m$ kinds of processing sequences when m jobs are processed through n stages, where n_i is the number of the machines at stage i.

Seven jobs are waiting to be processed at the processing plant, and each job goes through lathing, planning, and grinding processing steps. There are three lathes, two planers, and three grinders in three respective stages. The specific processing times are illustrated in Table 4.

Table 4. The processing times in hybrid flow shop scheduling.

Job	Stages							
	Lathe (s)			Planer (s)		Grinder (s)		
	1	2	3	1	2	1	2	3
1	24	75	51	84	62	39	54	11
2	3	75	17	86	73	99	31	70
3	66	17	71	52	23	33	16	18
4	16	12	91	48	2	14	15	80
5	80	17	22	89	14	38	14	51
6	41	63	87	7	77	56	71	55
7	33	84	21	51	97	63	46	21

The decoding method of the HFSP is described by a schedule matrix. Each column of the matrix represents a production stage, which is different from the PFSP. The scheduling of the different process stages can only be solved separately and then combined into a solution of the HFSP.

The HFSP contains only seven jobs and three stages, but there are $3^7 \times 2^7 \times 3^7 \approx 6 \times 10^8$ different processing sequences. The complexity of HFSP is very high. The four algorithms were also used to solve HFSP. The amount of iterations was set to 200. Given that the number of machines in each stage is different, the upper and lower boundaries were as follows: lathe: (1, 4), planer: (1, 3), grinder: (1, 4). The specific parameter settings are listed in Table 5.

Table 5. Parameter settings in the HFSP.

Algorithm	Parameters	Value
IFWA and FWA	Number of fireworks N	5
	M	50
	A	4
	Constant a	0.04
	Constant b	0.8
	$A_{initial}$	(ub - lb) × 0.02
	A_{final}	(ub - lb) × 0.001
PSO	C_1, C_2	1.49445
	Population size	40
	Range of particle velocity	[−1,1]
WOA	Population size	40
	b	1
	p	0.5

Figure 5 presents the boxplot for the four algorithms when solving this instance from HFSP. It is evident that the obtained optimization results had greater volatility because of the complexity of the HFSP. It is clear that the overall distribution of the results obtained by the IFWA was the most concentrated and the overall makespan was smaller than that of the other three algorithms. Figure 6 shows the optimal curve of four algorithms. It is evident that the IFWA could find the optimal makespan.

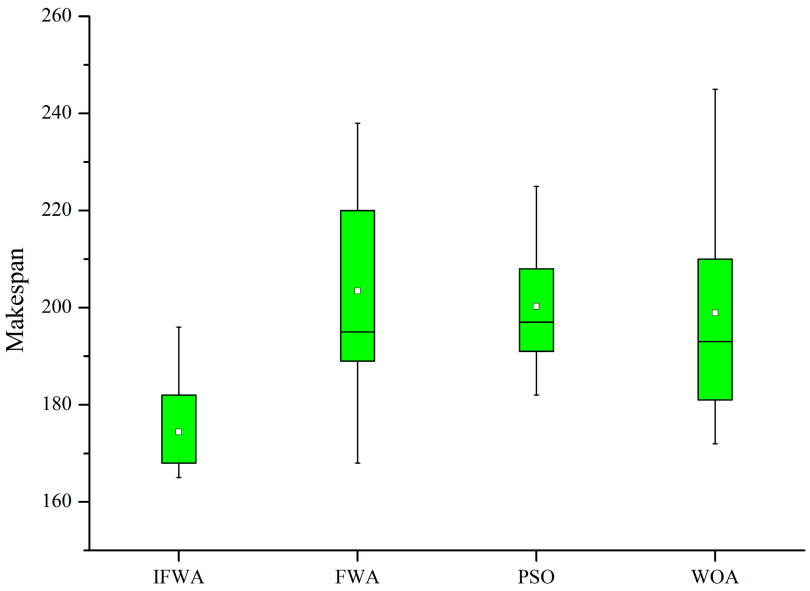

Figure 5. Boxplot for the instance of the HFSP.

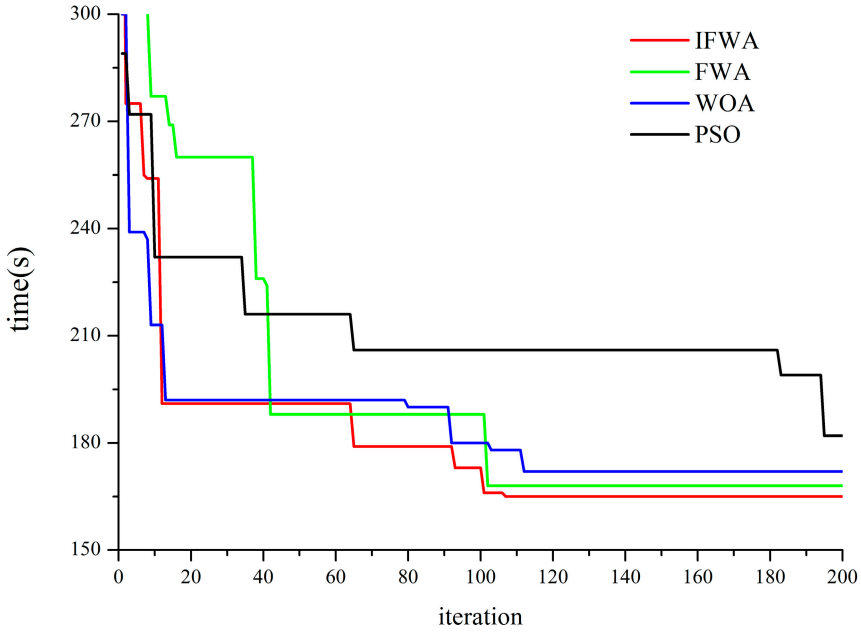

Figure 6. The optimal curves of the four algorithms.

The specific statistical results are compared in Table 6. Based on the accuracy of the optimization, the IFWA could find the minimum makespan. Compared with FWA, PSO, and WOA, the maximum completion time calculated by IFWA was reduced by 3.5%, 9.3%, and 4.1%, respectively, which shows its excellent optimization performance. From the perspective of standard deviation and average value, the improved algorithm had good performance. The corresponding evaluation indicators were much smaller than the other three algorithms, which means the IFWA could better find the optimal solution in each calculation. From the perspective of computing time, FWA and IFWA took a long time to optimize. Although the new selection strategy can save half the running time in IFWA, there was still a big gap compared with PSO and WOA. In general, the IFWA significantly improved the optimization accuracy and stability compared with the other algorithms. The calculation time with IFWA was much shorter than that of the FWA. The results prove that the IFWA has an excellent ability to deal with the HFSP.

Table 6. Comparisons of the optimizations in HFSP.

Algorithm	Best (s)	Average (s)	Worst (s)	STD (s)	Time (s)
IFWA	165	174	196	9.31	6.24
FWA	171	208	233	18.51	12.85
PSO	182	200	225	11.84	1.78
WOA	172	198	245	21.22	1.99

6. Conclusions

An improved fireworks algorithm was proposed to minimize the makespan in PFSPs and HFSPs. Three improved strategies are introduced in the new algorithm. Firstly, a non-linear decreasing radius is introduced to replace the minimum fireworks radius selectively in order to avoid the situation where the search radius of the optimal fireworks is zero. Secondly, a Cauchy and Gaussian mutation operator is introduced to replace the original Gaussian mutation operator, which enhances the search ability. Finally, an elite group selection strategy is adopted to reduce the system load. In order to verify the ability to solve the FSP, the IFWA was used to solve two instances (PFSP and HFSP).

When solving the PFSP, the IFWA had a 90% probability of obtaining the minimum completion time, which was 15% higher than the original version and far superior to the results of PSO and WOA.

When solving the HFSP by IFWA, the optimal value and stability of the results were very superior to the other algorithms. Compared with FWA, PSO, and WOA, the makespan calculated by IFWA was reduced by 3.5%, 9.3%, and 4.1%, respectively, and the standard deviation was the smallest, indicating its excellent performance. Compared with FWA, the computation time of IFWA was reduced by nearly half, which undoubtedly indicates the improved efficiency of the algorithm. The results prove the feasibility and effectiveness of IFWA in dealing with the PFSP and HFSP.

Compared with the original version, the IFWA significantly improved the optimization precision, stability, and speed, and could be adopted for NP-hard combination optimization problems. The IFWA is used as a reliable meta-heuristic algorithm to guide flow shop scheduling.

Modern enterprise competition depends not only on advanced production technology and equipment, but also on advanced management technology. In the field of flow shops, the method proposed in this paper can provide a valuable suggestion for solving the FSP, thereby helping enterprises to find the best scheduling strategy and improve their management level. Because actual flow shop scheduling has large scale and complicated calculation, the proposed method is of great significance both in terms of academic research and practical industrial production. At the same time, this study also has limitations. Firstly, only two instances were used to validate the performance of IFWA, which is insufficient. More examples need to be introduced in future study. Secondly, compared with other classical algorithms, the calculation time of the proposed method is still longer, which limits further applications. This is determined by the design structure of FWA itself. The algorithm structure should be further studied and the structure of other excellent algorithms should also be explored. Future study will also apply FWA to other types of scheduling.

Author Contributions: Investigation, X.P. and H.X.; Methodology, K.L.; Writing—original draft, X.P.; Writing—review & editing, M.-L.T. and M.K.L. All authors have read and agreed to the published version of the manuscript.

Funding: This study was partially funded by the project of ministry of science and technology, grant number MOST 108-2221-E-468 -004 -MY2.

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

References

1. Bibiks, K.; Hu, Y.F.; Li, J.P.; Pillai, P.; Smith, A. Improved discrete cuckoo search for the resource-constrained project scheduling problem. *Appl. Soft Comput.* **2018**, *69*, 493–503. [CrossRef]
2. Chen, Y.; Wang, X.; Cai, L. On Achieving Fair and Throughput-Optimal Scheduling for TCP Flows in Wireless Networks. *IEEE Trans. Wirel. Commun.* **2016**, *15*, 7996–8008. [CrossRef]
3. Huang, W.T.; Chen, P.S.; Liu, J.J.; Chen, Y.R.; Chen, Y.H. Dynamic configuration scheduling problem for stochastic medical resources. *J. Biomed. Inform.* **2018**, *80*, 96–105. [CrossRef] [PubMed]
4. Crow, M.L. Electric Vehicle Scheduling Considering Co-optimized Customer and System Objectives. *IEEE Trans. Sustain. Energy* **2018**, *9*, 410–419. [CrossRef]
5. Reddy, S.S. Optimal scheduling of thermal-wind-solar power system with storage. *Renew. Energy* **2017**, *101*, 1357–1368. [CrossRef]
6. Vagropoulos, S.I.; Balaskas, G.A.; Bakirtzis, A.G. An Investigation of Plug-In Electric Vehicle Charging Impact on Power Systems Scheduling and Energy Costs. *IEEE Trans. Power Syst.* **2017**, *32*, 1902–1912. [CrossRef]
7. Wang, Z.J.; Hu, H.; Gong, J. Modeling Worker Competence to Advance Precast Production Scheduling Optimization. *J. Constr. Eng. Manag.* **2018**, *144*, 04018098. [CrossRef]
8. Branke, J.; Nguyen, S.; Pickardt, C.W.; Zhang, M.J. Automated Design of Production Scheduling Heuristics: A Review. *IEEE Trans. Evolut. Comput.* **2016**, *20*, 110–124. [CrossRef]
9. Ribas, I.; Leisten, R.; Framinan, J.M. Review and classification of hybrid flow shop scheduling problems from a production system and a solutions procedure perspective. *Comput. Oper. Res.* **2010**, *37*, 1439–1454. [CrossRef]
10. Lei, D.M.; Guo, X.P. Hybrid flow shop scheduling with not-all-machines options via local search with controlled deterioration. *Comput. Oper. Res.* **2016**, *65*, 76–82. [CrossRef]

11. Lei, D.M.; Zheng, Y.L. Hybrid flow shop scheduling with assembly operations and key objectives: A novel neighborhood search. *Appl. Soft Comput.* **2017**, *61*, 122–128. [CrossRef]
12. Liu, W.B.; Jin, Y.; Price, M. A new improved NHE heuristic for permutation flowshop scheduling problems. *Int. J. Prod. Econ.* **2017**, *193*, 21–30. [CrossRef]
13. Che, A.; Kats, V.; Levner, E. An efficient bicriteria algorithm for stable robotic flow shop scheduling. *Eur. J. Oper. Res.* **2017**, *260*, 964–971. [CrossRef]
14. Liu, Y.F.; Liu, S.Y. A hybrid discrete artificial bee colony algorithm for permutation flowshop scheduling problem. *Appl. Soft Comput.* **2013**, *13*, 1459–1463. [CrossRef]
15. Huang, X.B.; Li, H.B.; Zhu, Y.C. Short-term ice accretion forecasting model for transmission lines with modified time-series analysis by fireworks algorithm. *IET Gener. Transm. Distrib.* **2018**, *12*, 1074–1080. [CrossRef]
16. Wang, S.J.; Liu, M.; Chu, C.B. A branch-and-bound algorithm for two-stage no-wait hybrid flow-shop scheduling. *Int. J. Prod. Res.* **2015**, *53*, 1143–1167. [CrossRef]
17. Xu, Z.; Xu, D.; He, J.; Wang, Q.; Liu, A.; Xiao, J. Mixed Integer Programming Formulations for Two-Machine Flow Shop Scheduling with an Availability Constraint. *Arab. J. Sci. Eng.* **2018**, *43*, 777–788. [CrossRef]
18. Palmer, D.S. Sequencing Jobs Through a Multi-Stage Process in the Minimum Total Time—A Quick Method of Obtaining a Near Optimum. *J. Oper. Res. Soc.* **1965**, *16*, 101–107. [CrossRef]
19. 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]
20. Gupta, J.N.D. A Functional Heuristic Algorithm for the Flowshop Scheduling Problem. *J. Oper. Res. Soc.* **1971**, *22*, 39–47. [CrossRef]
21. Johnson, S.M. Optimal two- and three-stage production schedules with setup times included. *Nav. Res. Logist. Q.* **1954**, *1*, 61–68. [CrossRef]
22. Campbell, H.G.; Dudek, R.A.; Smith, M.L. A heuristic algorithm for the n job, m machine sequencing problem. *Manag. Sci.* **1970**, *16*, 630. [CrossRef]
23. Kalczynski, P.J.; Kamburowski, J. On the NEH heuristic for minimizing the makespan in permutation flow shops. *Omega Int. J. Manag. Sci.* **2007**, *35*, 53–60. [CrossRef]
24. Semanco, P.; Modrak, V. A Comparison of Constructive Heuristics with the Objective of Minimizing Makespan in the Flow-Shop Scheduling Problem. *Acta Polytech. Hung.* **2012**, *9*, 177–190.
25. Zhang, H.; Zhou, A.M.; Song, S.M.; Zhang, Q.F.; Gao, X.Z.; Zhang, J. A Self-Organizing Multiobjective Evolutionary Algorithm. *IEEE Trans. Evolut. Comput.* **2016**, *20*, 792–806. [CrossRef]
26. Engin, O.; Guclu, A. A new hybrid ant colony optimization algorithm for solving the no-wait flow shop scheduling problems. *Appl. Soft Comput.* **2018**, *72*, 166–176. [CrossRef]
27. Fathi, M.; Rodriguez, V.; Fontes, D.B.M.M.; Alvarez, M.J. A modified particle swarm optimisation algorithm to solve the part feeding problem at assembly lines. *Int. J. Prod. Res.* **2016**, *54*, 878–893. [CrossRef]
28. Precup, R.E.; David, R.C.; Petriu, E.M. Grey Wolf Optimizer Algorithm-Based Tuning of Fuzzy Control Systems With Reduced Parametric Sensitivity. *IEEE Trans. Ind. Electron.* **2017**, *64*, 527–534. [CrossRef]
29. Marichelvam, M.K.; Prabaharan, T.; Yang, X.S. A Discrete Firefly Algorithm for the Multi-Objective Hybrid Flowshop Scheduling Problems. *IEEE Trans. Evolut. Comput.* **2014**, *18*, 301–305. [CrossRef]
30. Dasgupta, P.; Das, S. A Discrete Inter-Species Cuckoo Search for flowshop scheduling problems. *Comput. Oper. Res.* **2015**, *60*, 111–120. [CrossRef]
31. Marichelvam, M.K.; Tosun, O.; Geetha, M. Hybrid monkey search algorithm for flow shop scheduling problem under makespan and total flow time. *Appl. Soft Comput.* **2017**, *55*, 82–92. [CrossRef]
32. Tang, D.B.; Dai, M.; Salido, M.A.; Giret, A. Energy-efficient dynamic scheduling for a flexible flow shop using an improved particle swarm optimization. *Comput. Ind.* **2016**, *81*, 82–95. [CrossRef]
33. Lin, Q.; Gao, L.; Li, X.Y.; Zhang, C.J. A hybrid backtracking search algorithm for permutation flow-shop scheduling problem. *Comput. Ind. Eng.* **2015**, *85*, 437–446. [CrossRef]
34. Komaki, G.M.; Kayvanfar, V. Grey Wolf Optimizer algorithm for the two-stage assembly flow shop scheduling problem with release time. *J. Comput. Sci.* **2015**, *8*, 109–120. [CrossRef]
35. 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.* **2019**, *57*, 1756–1771. [CrossRef]

36. Abdel-Basset, M.; Manogaran, G.; El-Shahat, D.; Mirjalili, S. A hybrid whale optimization algorithm based on local search strategy for the permutation flow shop scheduling problem. *Future Gener. Comput. Syst.* **2018**, *85*, 129–145. [CrossRef]
37. Tan, Y.; Zhu, Y. Fireworks Algorithm for Optimization. In Proceedings of the Advances in Swarm Intelligence, Berlin/Heidelberg, Germany, 12–15 June 2010; pp. 355–364.
38. Babu, T.S.; Ram, J.P.; Sangeetha, K.; Laudani, A.; Rajasekar, N. Parameter extraction of two diode solar PV model using Fireworks algorithm. *Sol. Energy* **2016**, *140*, 265–276. [CrossRef]
39. Guendouz, M.; Amine, A.; Hamou, R.M. A discrete modified fireworks algorithm for community detection in complex networks. *Appl. Intell.* **2017**, *46*, 373–385. [CrossRef]
40. El Majdouli, M.A.; Rbouh, I.; Bougrine, S.; El Benani, B.; El Imrani, A.A. Fireworks algorithm framework for Big Data optimization. *Memet. Comput.* **2016**, *8*, 333–347. [CrossRef]
41. Ye, S.G.; Ma, H.P.; Xu, S.; Yang, W.Q.; Fei, M.R. An effective fireworks algorithm for warehouse-scheduling problem. *Trans. Inst. Meas. Control* **2017**, *39*, 75–85. [CrossRef]
42. Yu, C.; Kelley, L.; Zheng, S.Q.; Tan, Y. Fireworks Algorithm with Differential Mutation for Solving the CEC 2014 Competition Problems. In Proceedings of the 2014 IEEE Congress on Evolutionary Computation (CEC), Beijing, China, 6–11 July 2014; pp. 3238–3245.
43. Jadoun, V.K.; Pandey, V.C.; Gupta, N.; Niazi, K.R.; Swarnkar, A. Integration of renewable energy sources in dynamic economic load dispatch problem using an improved fireworks algorithm. *IET Renew. Power Gener.* **2018**, *12*, 1004–1011. [CrossRef]
44. Arsic, A.; Tuba, M.; Jordanski, M. Fireworks algorithm applied to wireless sensor networks localization problem. In Proceedings of the 2016 IEEE Congress on Evolutionary Computation, Vancouver, BC, Canada, 24–29 July 2016; pp. 4038–4044.
45. Zheng, S.; Janecek, A.; Tan, Y. Enhanced Fireworks Algorithm. In Proceedings of the 2013 IEEE Congress on Evolutionary Computation, Cancun, Mexico, 20–23 June 2013; pp. 2069–2077.
46. Zhang, B.; Zheng, Y.J.; Zhang, M.X.; Chen, S.Y. Fireworks Algorithm with Enhanced Fireworks Interaction. *IEEE ACM Trans. Comput. Biol. Bioinform.* **2017**, *14*, 42–55. [CrossRef]
47. Xue, J.J.; Wang, Y.; Li, H.; Meng, X.F.; Xiao, J.Y. Advanced Fireworks Algorithm and Its Application Research in PID Parameters Tuning. *Math. Probl. Eng.* **2016**, *2016*, 2534632. [CrossRef]
48. Tsai, J.T.; Yang, C.I.; Chou, J.H. Hybrid sliding level Taguchi-based particle swarm optimization for flowshop scheduling problems. *Appl. Soft Comput.* **2014**, *15*, 177–192. [CrossRef]
49. Meng, L.L.; Zhang, C.Y.; Shao, X.Y.; Ren, Y.P.; Ren, C.L. Mathematical modelling and optimisation of energy-conscious hybrid flow shop scheduling problem with unrelated parallel machines. *Int. J. Prod. Res.* **2019**, *57*, 1119–1145. [CrossRef]
50. Perez, R.; Jons, S.; Hernandez, A. Solution of a flexible jobshop scheduling problem using an Estimation of Distribution Algorithm. *Rev. Iberoam. Autom. Inform.* **2015**, *12*, 49–57.
51. Yu, C.L.; Semeraro, Q.; Matta, A. A genetic algorithm for the hybrid flow shop scheduling with unrelated machines and machine eligibility. *Comput. Oper. Res.* **2018**, *100*, 211–229. [CrossRef]
52. Sun, Z.W.; Gu, X.S. A novel hybrid estimation of distribution algorithm for solving hybrid flowshop scheduling problem with unrelated parallel machine. *J. Cent. South Univ.* **2017**, *24*, 1779–1788. [CrossRef]