symmetry

MDPI

Article

A Hybrid Imperialist Competitive Algorithm for the Distributed Unrelated Parallel Machines Scheduling Problem

Youlian Zheng [1], Yue Yuan [2], Qiaoxian Zheng [1,*] and Deming Lei [2,*]

[1] Faculty of Computer Science and Information Engineering, Hubei University, Wuhan 430061, China; zhengyl@hubu.edu.cn
[2] School of Automation, Wuhan University of Technology, Wuhan 430062, China; 9366@whut.edu.cn
* Correspondence: zqxlm1978@163.com (Q.Z.); deminglei11@163.com (D.L.)

Abstract: In this paper, the distributed unrelated parallel machines scheduling problem (DUPMSP) is studied and a hybrid imperialist competitive algorithm (HICA) is proposed to minimize total tardiness. All empires were categorized into three types: the strongest empire, the weakest empire, and other empires; the diversified assimilation was implemented by using different search operator in the different types of empires, and a novel imperialist competition was implemented among all empires except the strongest one. The knowledge-based local search was embedded. Extensive experiments were conducted to compare the HICA with other algorithms from the literature. The computational results demonstrated that new strategies were effective and the HICA is a promising approach to solving the DUPMSP.

Keywords: imperialist competitive algorithm; local search; distributed unrelated parallel machines scheduling problem

Citation: Zheng, Y.; Yuan, Y.; Zheng, Q.; Lei, D. A Hybrid Imperialist Competitive Algorithm for the Distributed Unrelated Parallel Machines Scheduling Problem. *Symmetry* **2022**, *14*, 204. https://doi.org/10.3390/sym14020204

Academic Editor: Mihai Postolache

Received: 17 December 2021
Accepted: 14 January 2022
Published: 21 January 2022

1. Introduction

In recent years, single-factory or centralized production has been continuously replaced by multi-factory production or distributed manufacturing with the further development of globalization. Distributed manufacturing enables manufacturers to be closer to their customers and suppliers, to produce and market their products more effectively, to respond to market changes more quickly and achieve better product quality, lower production cost, reduce management risk, etc. As an important part of manufacturing systems, scheduling is shifted from single-factory scheduling to distributed scheduling with the change of production mode. Distributed scheduling has been considered fully in the past decade [1–30].

As a common distributed scheduling problem, the distributed parallel machine scheduling problem (DPMSP) has attracted some attention since the works of Hooker [1]. Chen and Pundoor [2] analyzed the computational complexity of scheduling in supply chains and proposed some fast heuristics. Terrazas-Moreno and Grossmann [3] gave a hybrid bi-level and spatial Lagrangian decomposition method for the scheduling and planning problem in multiple factories with different sites. Behnamian and Fatemi Ghomi [4] applied a heuristic and a genetic algorithm (GA) with a new encoding scheme to minimize the makespan in heterogeneous factories. Behnamian [5] provided a decomposition-based hybrid variable neighborhood search/tabu search algorithm to deal with a DPMSP with the total cost of some factories and the profit of other factories. Behnamian and Fatemi Ghomi [6] developed a Monte-Carlo-based heuristic with seven local search algorithms to minimize the cost-related objective. Behnamian [7] designed a two-phase algorithm with particle swarm optimization for a DPMSP with parallel jobs.

Lei et al. [8] dealt with the distributed unrelated parallel machine scheduling problem (DUPMSP) in heterogeneous factories and proposed a novel imperialist competitive algorithm (ICA) with memory, new revolution, and imperialist competition to minimize

the makespan. Lei and Liu [9] studied the DUPMSP with preventive maintenance and makespan minimization and presented an artificial bee colony algorithm in which the wholes swarm is divided into one employed bee colony and three onlooker bee colonies, and the four colonies differed from each other in their search strategies. Lei et al. [10] considered a multi-objective DUPMSP and developed an improved ABC algorithm to minimize the makespan and total tardiness simultaneously, in which problem-related knowledge was proven and knowledge-based neighborhood search was proposed. Pan et al. [11] solved an energy-efficient DUPMSP by using a knowledge-based two-population optimization algorithm to minimize total energy consumption and total tardiness simultaneously, in which the nondominated sorting genetic algorithm-II and differential evolution performed cooperatively and two knowledge-based local search operators were proposed.

The ICA is constructed by simulating sociopolitical behaviors. It is made up of initialization, initial empires, assimilation, revolution, and imperialist competition. Unlike other meta-heuristics [31–33], the ICA has some characteristic features, which are a good neighborhood search ability, an effective global search property, and a good convergence rate [34].

As an effective method for solving production scheduling [35], the ICA has been extensively applied to many scheduling problems. Banisadr et al. [36] gave a hybrid ICA for single-machine scheduling. For assembly flow shop scheduling, Shokrollahpour et al. [37] developed a new ICA to optimize two conflicting objectives, and Seidgar et al. [38] proposed an ICA to solve a two-stage problem. With respect to the flexible job shop scheduling problem, Zandieh et al. [39] applied an improved ICA to solve the problem with maintenance. Karimi et al. [40] developed a hybrid algorithm with the ICA and simulated annealing for a problem with transportation times. Lei et al. [41] presented a two-phase algorithm based on the ICA and variable neighborhood search for a problem with an energy consumption threshold. Abedi et al. [42] gave a multi-objective ICA for identical parallel batch-processing machines' scheduling. Yazdani et al. [43] proposed a hybrid algorithm with the ICA and local search for a two-agent parallel machine scheduling problem. An improved ICA was proposed by Zhang et al. [44] to solve photolithography machines' scheduling. Li et al. [45] presented an ICA with empire cooperation for solving a fuzzy distributed assembly scheduling problem. Li et al. [46] proposed an ICA with feedback to solve an energy-efficient FJSP with transportation and a sequence-dependent setup time.

As a meta-heuristic with significant features, the ICA has great potential to generate high-quality solutions for the DPMSP; on the other hand, in the existing ICA [34–36], each empire has the same search operator in assimilation, all empires compete in imperialist competition, and the problem-related properties are not used fully. In general, assimilation and revolution should be performed based on the features of the empires. Generally, assimilation in the weakest empire should be intensified and implemented by moving colonies toward imperialists of other empires or other good solutions. It is useful to avoid prematurely excluding the strongest empire from imperialist competition, and the effective usage of problem-related properties or knowledge can improve the search efficiency. Thus, it is beneficial to investigate these improvements of the ICA to solve the DUPMSP efficiently.

In this study, the DUPMSP with total tardiness was considered and a hybrid imperialist competitive algorithm (HICA) is proposed. In the HICA, all empires are divided into three types: the strongest empire, the weakest empire, and other empires. Diversified assimilation was implemented, that is each kind of empire was provided a different assimilation from other kinds of empires. A novel imperialist competition was implemented among all empires except the first type of empire. Problem-related knowledge was included in the local search, and an effective way was applied to integrate knowledge with the ICA. Many computational experiments were conducted. The computational results validated the effectiveness of the new strategies of the HICA and its promising advantages in the DUPMSP.

The paper is organized as follows. The problem description is introduced in Section 2. The HICA for the DUPMSP is described in Section 3. Numerical test experiments are given in Section 4, and the conclusions and the future topics are provided in the final section.

2. Problem Description

The DUPMSP is described below. There exist n jobs $J_1, J_2, \cdots, J_n$ processed among F factories at different sites. Factory $f = 1, 2, \cdots, F$ possesses m_f unrelated parallel machines, which are $M_{s_f+1}, \cdots, M_{s_f+m_f}$ and available at all times, where $s_f = \sum_{l=1}^{f-1} m_l, f > 1, s_1 = 0$. The total number of machines in all factories is $W = \sum_{f=1}^{F} m_f$. Each job J_i is available at Time 0 and has a due date d_i and p_{ik}, which is the processing time of job J_i on machine M_k, $k = 1, 2, \cdots, W$.

There are some constraints on jobs and machines.

Each machine can deal with at most one job at a time.

No jobs can be processed on more than one machine at a time.

Operations are not allowed to be interrupted.

The goal of the problem is to minimize the total tardiness.

$$Tard_{tot} = \sum_{i=1}^{n} T_i \tag{1}$$

where C_i and T_i are the completion time and tardiness of job J_i, $T_i = \max\{C_i - d_i, 0\}$, and $Tard_{tot}$ indicates the total tardiness.

The DUPMSP is the extended version of the unrelated parallel machine scheduling problem (PMSP). The PMSP with total tardiness is NP-hard, so the DUPMSP with total tardiness is also NP-hard. When all parallel machines are identical and the objective is the makespan, any two machines are symmetrical to each other, for example for M_{s_f+1} and M_{s_f+2}, a job can be finished at the same time when it is processed on these two machines, respectively. In the DUPMSP, there is a certain amount of destruction of the above symmetry; however, the symmetry still exists.

For the DUPMSP with total tardiness, it consists of three sub-problems: factory assignment, machine assignment, and scheduling. Factory assignment is used to decide which jobs are allocated into each factory, and machine assignment is for selecting an appropriate machine for each job in a factory. Lei et al. [8] analyzed the strong coupled relation among sub-problems and pointed out that two assignment sub-problems can be integrated into an extended machine assignment, in which each job is allocated to one of W machines, not limited to a machine in a factory.

3. HICA for the DUPMSP

In ICA, a country represents a solution of the problem, and solutions in population P are categorized into two parts: imperialists and colonies, the former are some best solutions in P and the latter are all solutions of P except imperialists. The main steps of the ICA consist of the initial empires' construction, assimilation, revolution, and imperialist competition.

In the existing ICA [34,36,41], the improvements are made by using new strategies on initial empires, assimilation, revolution, and imperialist competition; however, assimilation and revolution are frequently executed in the same ways for all empires and seldom implemented in terms of the characteristics of empires; on the other hand, if the strongest empire has the biggest normalized total cost $\overline{TC}_k$, then this empire wins likely in the imperialist competition, and colonies can be easily reallocated into the empire; as a result, premature removal may occur. A novel algorithm named the HICA is designed based on the above analyses.

3.1. Initialization and Initial Empires

As stated above, the DUPMSP is composed of an extended machine assignment and scheduling. Lei et al. [10] proposed a two-string representation. For the DUPMSP,

its solution is represented as a machine assignment string $[M_{h_1}, M_{h_2}, \cdots, M_{h_n}]$ and a scheduling string $[q_1, q_2, \cdots, q_n]$. Machine M_{h_i} is allocated for job J_i, and q_l is a real number and corresponds to J_l. The decoding procedure [10] was directly used in this study.

Initial population P with N solutions is randomly generated, then all solutions are sorted according to the cost. N_{im} solutions with the smallest cost are chosen as imperialists, and the remaining countries are used as colonies, then the initial empires are constructed. Algorithm 1 shows the steps of the initial empires, where $Tard_{tot,i}$ is the total tardiness of the i-th imperialist, and F_k and NC_k are defined by:

$$F_k = \bar{c}_k \Big/ \sum_{l \in S_{im}} \bar{c}_l \tag{2}$$

$$NC_k = round(F_k \times N_{col}) \tag{3}$$

NC_k is the number of colonies possessed by imperialist k. S_{im} is the set of all imperialists. $N_{col} = N - N_{im}$ indicates the number of colonies. $round(x)$ denotes the nearest integer to x.

Algorithm 1 Initial empires.

1: Determine N_{im} imperialists from population P, and sort them in ascending order of total tardiness
2: Compute the normalized cost $\bar{c}_k = Tard_{tot,N_{im}-k+1}$ for imperialist $k = 1, 2, \cdots, N_{im}$
3: Calculate the power F_k and NC_k and randomly allocate NC_k colonies for each imperialist

In the HICA, the cost c_i of a solution x_i is defined as its objective value. When all imperialists are sorted, Imperialist 1 has the smallest total tardiness $Tard_{1,tot}$, Imperialist 2 possesses the second smallest total tardiness $Tard_{2,tot}$, and so on; obviously, $Tard_{1,tot} \leq Tard_{2,tot} \leq \cdots \leq Tard_{N_{im},tot}$.

For each initial empire k, its total cost TC_k is computed. Suppose that $TC_1 \leq TC_2 \leq \cdots \leq TC_{N_{im}}$, that is Empire 1 is the strongest one and empire N_{im} is the weakest one. All empires are divided into three types: the first type just has Empire 1, and third type only has empire N_{im}; the second type is made up of empires $2, \cdots, N_{im} - 1$.

$$TC_k = c_k + \zeta \sum_{\lambda \in Q_k} c_\lambda / NC_k \tag{4}$$

where Q_k is the set of colonies possessed by imperialist k and ζ is a real number and set to 0.1.

3.2. Diversified Assimilation and New Revolution

Assimilation and revolution are the main paths to produce new solutions. In general, the process of assimilation is identical for all empires. In the HICA, diversified assimilation is implemented, that is assimilation is performed differently in the different kinds of empires. Algorithm 2 shows the steps of diversified assimilation, where α is a real number, β is an integer, and the retained set Ω is used to store the solutions generated in Empire 1, that is a new solution is produced in Empire 1 and added into Ω directly, U_{A_i} and U_{R_i} are the probability of assimilation and revolution. In this study, we set $U_{A_2} = 0.8$, $U_{R_2} = 0.1$, $U_{R_3} = 0.2$, $\alpha = 0.1$, and $\beta = 5$ based on experiments.

Assimilation is executed by a global search between the colony and its learning object. In all empires except the worst one, the learning objective of the colony is its imperialist. For colony x_i and its learning object y, the global search is shown below. Two positions k_1, k_2 are stochastically decided, and machines between these positions on the first string of x_i are directly replaced with those on the first string of y; a new solution z is generated; if z has smaller $Tard_{tot}$ than or identical $Tard_{tot}$ with x_i, then replace x_i with z; otherwise, genes of z between two positions k_1 and k_2 of the scheduling string are displaced by those of y on the same positions; if the newly obtained solution z and x_i meets the above condition, then z substitutes for x_i.

Algorithm 2 Diversified assimilation and new revolution.

1: Add all solutions of Empire 1 into the retained set Ω
2: Choose colonies from Empire 1 by probability U_{A_1}, and execute assimilation between each chosen colony and its imperialist.
3: Select colonies of Empire 1 in terms of revolution probability U_{R_2}, and apply multiple neighborhood search to each chosen colony.
4: Choose the best $NC_1 + 1$ solutions from the retained set Ω; replace all solutions in Empire 1; decide on the new imperialist.
5: **for** $k = 2$ to $N_{im} - 1$ **do**
6: calculate $\tilde{c}_k = \sum_{x_i \in Q_k} (c_i - c_k)/NC_k$.
7: **if** $\tilde{c}_k < \alpha \times c_k$ **then**
8: implement assimilation with U_{A_1} and revolution with U_{R_1} as done in Empire 1.
9: **else**
10: execute assimilation with $1 - U_{A_1}$ and revolution with $1 - U_{R_1}$ as done in Empire 1.
11: **end if**
12: **end for**
13: For empire N_{im}, choose colonies with probability U_{A_2}; execute assimilation between each chosen colony and one of the best β solutions in Ω; perform revolution with U_{R_3} in the same way as for Empire 1.

In Algorithm 2, the assimilation probability or learning object differ from each other in different types of empires. For example, in the worst empire, its imperialist may have low quality and cannot guide the colony, so the colony learns from one best solution of P to obtain high-quality solutions.

Algorithm 2 also shows revolution in each type of empire. A multiple-neighborhood search is used in revolution. There are seven neighborhood structures: $\mathcal{N}_1, \mathcal{N}_2, \cdots, \mathcal{N}_7$.

$\mathcal{N}_1$ is described as follows. Move a job with the biggest tardiness from its current machine M_k to a randomly chosen machine M_w ($w \neq k$). $\mathcal{N}_2$ is performed for two randomly chosen machines M_{k_1} and M_{k_2}; shift the job with the biggest tardiness from M_{k_1} to M_{k_2}. $\mathcal{N}_3$ is also applied for two randomly chosen machines M_{k_1} and M_{k_2} by exchanging the job with the biggest tardiness on M_{k_1} and the job with the smallest tardiness on M_{k_2}.

$\mathcal{N}_4$ acts on the scheduling string by exchanging two randomly chosen genes q_{i_1} and q_{i_2}. $\mathcal{N}_5$ is used to produce a new solution by inserting a randomly chosen gene q_i into a new position $l, l \neq i$. $\mathcal{N}_6$ is shown as follows. Two positions k_1 and k_2 are randomly chosen, and genes between two positions are reversed.

$\mathcal{N}_7$ is depicted below. Begin with machine M_k with the biggest total tardiness of jobs; all jobs on M_k are in the ascending order of their due dates, then let the sequence of their q_l of these jobs be identical to the order of the due dates. For example, for jobs J_1, J_{17}, J_{11} on machine M_1, the ascending order of their due dates is J_{11}, J_{17}, J_1, and the sequence of their q_l becomes $0.33, 0.23, 0.32$, that is new q_1 is 0.33, new $q_{11} = 0.23$, and new $q_{17} = 0.32$.

The revolution of colony x_i is shown as follows. Let $g = 1, t = 1$; repeat the following steps until $t = R$: a new solution $z \in \mathcal{N}_g(x_i)$ is produced; if x_i can be replaced with z according to the condition in the assimilation, then z substitutes for x_i; otherwise, $g = g + 1$; let $g = 1$ if $g = 8$. $t = t + 1$.

When there are only two empires, colonies are chosen in terms of U_{A_1} for assimilation between the colony and its imperialist, and revolution is implemented using U_{R_1} and the above multiple-neighborhood search.

When assimilation and revolution are performed in all empires, the exchange step is executed. In each empire k, if the total tardiness of imperialist k is bigger than that of a colony, then imperialist k is exchanged with the colony.

Total cost TC_k is calculated for each empire k, and all empires are categorized into three types, as done in Section 3.1.

3.3. Imperialist Competition

In general, all empires compete with each other, and the weakest colony of the weakest empire is directly added into the winning empire; in this study, a new imperialist competition was applied and shown in Algorithm 3, where $\overline{TC}_k$ is defined by:

$$\overline{TC}_k = TC_{N_{im}-k+1} \tag{5}$$

When $N_{im} \leq 2$, all empires compete with each other, as done in the above procedure.

Algorithm 3 Imperialist competition.

1: Determine the normalized total cost $\overline{TC}_k$ for empire $k = 2, 3, \cdots, N_{im}$
2: Compute power POW_k for empire $k = 2, \cdots, N_{im}$ and construct vector $[POW_2 - r_2, \cdots, POW_{N_{im}} - r_{N_{im}}]$
3: Decide on an empire g with the biggest $POW_g - r_g$
4: Randomly choose a solution x from Empire 1; directly substitute for the weakest colony; add into empire g; then, execute multiple-neighborhood search on x, as done in revolution.

Unlike the existing imperialist competition, our competition process directly eliminates the possibility of including the weakest colony in empire g because the weakest colony is a worse solution and difficult to improve even if it is included in a strong empire.

3.4. Local Search

Problem-related knowledge was incorporated in the search procedure of the scheduling algorithm [10,11,47–49]. The inclusion of knowledge can avoid useless searching and make the algorithm search in the possible regions of the optimal solutions. There are some papers on knowledge-based scheduling [47–49], and the knowledge used is mainly about the makespan. In this study, the knowledge-based local search acts on the imperialist of each empire to intensify the local search ability of the HICA.

Theorem [10]: For two adjacent jobs J_i and J_j on a machine M_k, suppose t is the beginning time of J_i:

(1) If $T_i > 0$ and $T_j > 0$:

   a. If $t + p_{jk} - d_j \leq 0$ and $t + p_{ik} - d_j > 0$, then the sum of their tardiness will diminish after two jobs are exchanged;
   b. If $t + p_{jk} - d_j > 0$ and $p_{ik} - p_{jk} > 0$, then the sum of their tardiness will diminish after two jobs are exchanged;

(2) If $T_i \leq 0$, $T_j > 0$, and one of the following conditions is met:

   a. $t + p_{ik} + p_{jk} - d_i \leq 0$;
   b. $t + p_{ik} + p_{jk} - d_i > 0$, $t + p_{jk} - d_j \leq 0$, and $d_i > d_j$;
   c. $t + p_{ik} + p_{jk} - d_i > 0$, $t + p_{jk} - d_j > 0$, $d_i - t - p_{jk} > 0$; then the sum of their tardiness will diminish after two jobs are exchanged.

   T_i^* and T_j^* are the tardiness of J_i and J_j after the exchange.

The local search is shown below. For empire $k = 1, 2, \cdots, N_{im}$, the following steps are executed on the imperialist of the empire: on each machine M_l, $l = 1, 2, \cdots, W$, start with the first job on M_l; repeat the following steps until all adjacent jobs are checked: for Job J_i and its adjacent job J_j, exchange them if they meet the conditions in the theorem.

3.5. Algorithm Description

The HICA is shown in Algorithm 4, and its flow chart is described in Figure 1.

Algorithm 4 HICA.

1: Randomly produce an initial population P
2: **while** stopping condition is not met **do**
3: Construct initial empires
4: **if** $N_{im} > 2$ **then**
5: Sort empires, and categorize empires into three types
6: Execute Algorithm 2, and exchange for each empire
7: Implement the new imperialist competition for all empires except Empire 1
8: **else**
9: Perform Algorithm 2, and exchange
10: Execute imperialist competition
11: **end if**
12: Apply local search to the imperialist of each empire
13: **end while**

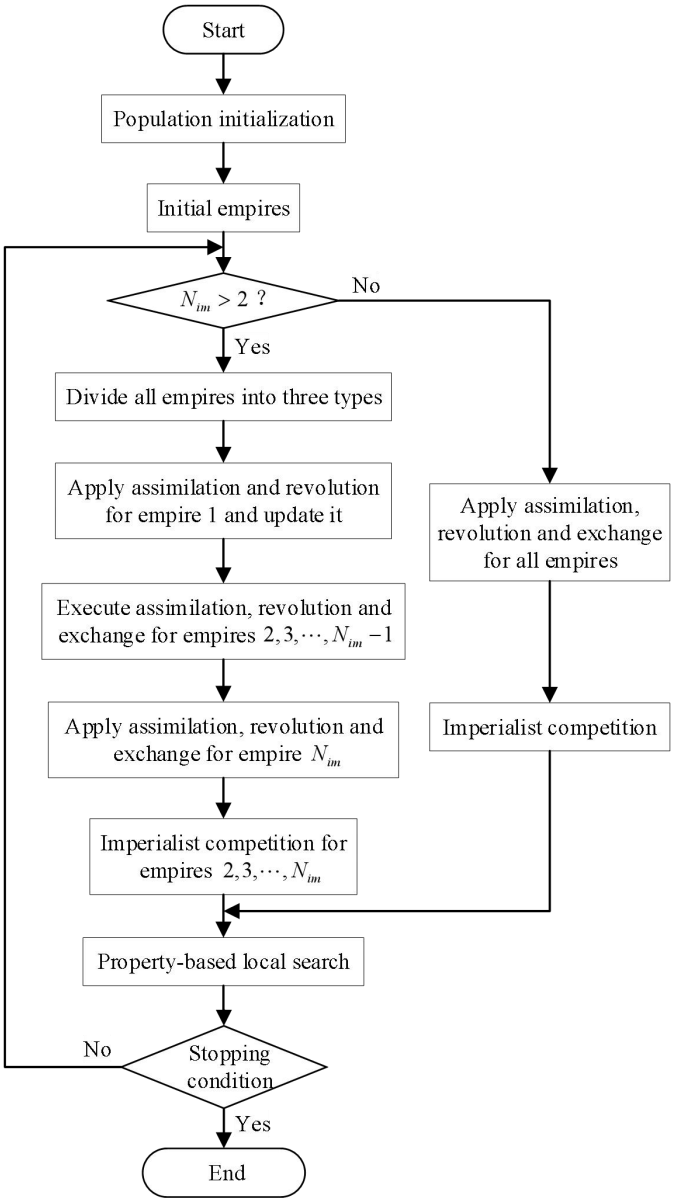

Figure 1. Flowchart of the HICA.

The HICA has the following time complexity $O(N \times R \times max\_g)$, where max_g indicates the repeated number of Lines 3–12 of Algorithm 4. The HICA has the following features: (1) Three types of empires are constructed, and diversified assimilation is per-

formed in these types of empires. (2) A new imperialist competition is implemented. Not all empires compete with each other, and the weakest colony is not included in the winning empire and replaced with a solution produced by the multiple-neighborhood search. (3) The knowledge-based local search is used to improve the imperialists of all empires for high search efficiency. The above features can provide a good balance between exploration and exploitation for the HICA and result in good performance.

4. Computational Experiments

Extensive experiments were conducted to test the performance of the HICA for the DUPMSP with total tardiness. All experiments were implemented by using Microsoft Visual C++ 2015 and run on a 4.0 G RAM 2.00 GHz CPU PC.

4.1. Test Instances, Metrics, and Comparative Algorithms

Ninety-six combinations were randomly generated, and their basic descriptions are given in Table 1. Five instances were stochastically produced for each combination, and four-hundred eighty instances were obtained. The due date of J_i is decided by:

$$d_i = \sum_{j=1}^{W} p_{ij} \times n \Big/ 2W^2 \quad (6)$$

Table 1. Information on combinations.

Instance	F	m_f	Instance	n
1–9	2	4 5	1,10,19,28,37,46,55,64,78,84,90,96	80
10–18	3	4 5 6	2,11,20,29,38,47,56,65	100
19–27	4	4 5 6 3	3,12,21,30,39,48,57,66	120
28–36	5	4 5 6 3 7	4,13,22,31,40,49,58,67	140
37–45	2	3 4	5,14,23,32,41,50,59,68	160
46–54	3	3 4 4	6,15,24,33,42,51,60,69	180
55–63	4	3 4 4 2	7,16,25,34,43,52,61,70	200
64–72	5	3 4 4 2 5	8,17,26,35,44,53,62,71	220
73–78	2	2 3	9,18,27,36,45,54,63,72	240
79–84	3	2 3 3	73,79,85,91	30
85–90	4	2 3 3 2	74,80,86,92	40
91–96	5	2 3 3 2 2	75,81,87,93	50
			76,82,88,94	60
$p_{ik} \in [30, 50]$			77,83,89,95	70

Behnamian and Fatemi Ghomi [4] proposed a GA for the DPMSP. Hulett et al. [50] presented a particle swarm optimization (PSO [50]) algorithm for a non-identical PMSP with total weighted tardiness. This GA can be directly applied to solve the DUPMSP. This PSO algorithm also can be used to solve our DUPMSP after the decoding part is replaced with the decoding procedure in the HICA, so they were chosen as comparative algorithm.

Each algorithm ran 20 times for each instance. For a combination, mn_i and mx_i were the best solution, the worst solution obtained in 20 runs for its instance $i = 1, 2, 3, 4, 5$, respectively. ag_i is the average makespan of 20 elite solutions for instance i. Three indices Min, Max, and Avg are defined by $Min = \sum_{i=1}^{5} mn_i/5$, $Max = \sum_{i=1}^{5} mx_i/5$, and $Avg = \sum_{i=1}^{5} ag_i/5$.

4.2. Parameter Settings

The HICA has the following parameters: N, N_{im}, max_it, U_{A_1}, U_{R_1}. The Taguchi method is often applied to decide parameter settings for optimization algorithm and was also adopted in this study. Table 2 describes the levels of all parameters.

Table 2. Parameters and their levels.

Parameter	Factor Level		
	1	**2**	**3**
N	80	100	120
N_{im}	5	6	7
max_it	80,000	100,000	120,000
U_{A_1}	0.6	0.7	0.8
U_{R_1}	0.2	0.3	0.4

The HICA ran 20 times for an instance of Problem Combination 14. the orthogonal array $L_{27}(3^5)$ was executed. The results of the *Min* and S/N ratio are shown in Figure 2. The S/N ratio is defined as $-10\log_{10}(0.1 \times Min^2)$, where *Min* denotes the best solution obtained in 20 runs. As shown in Figure 2, the best settings were $N = 100$, $N_{im} = 6$, $max\_it = 10^5$, $U_{A_1} = 0.7$, $U_{R_1} = 0.3$.

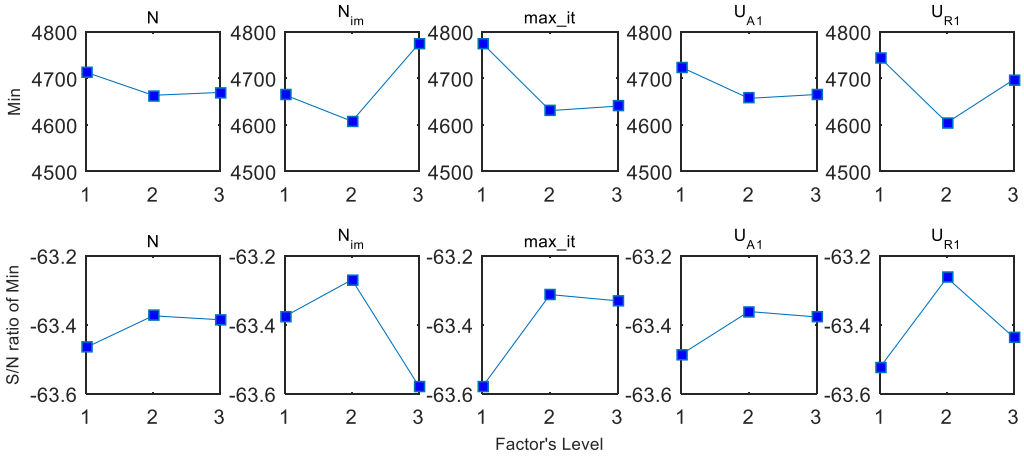

Figure 2. The mean *Min* and the mean S/N ratio of the *Min*.

4.3. Results and Discussion

Two variants named HICA1 and HICA2 are given. HICA1 was obtained after the knowledge-based local search was deleted from the HICA. The comparison between the HICA and HICA1 was to validate the impact of the local search on the performance of the HICA. In HICA2, when assimilation was performed, each colony moved toward its imperialist. The usage of HICA2 was to show if the diversified assimilation was useful to improve the performance of the HICA.

Tables 3–6 describe the computational results of the HICA, the two variants, and the two comparative algorithms, in which data highlighted in bold are results of HICA being better than any other algorithms. The Wilcoxon test was performed, and the corresponding results are shown in Table 7. The box plots of all algorithms are given in Figure 3. The convergence curves of all algorithms on Combinations 24 and 76 are listed in Figures 4 and 5.

Table 3. Computational results of all algorithms on *Min*.

Ins	HICA	HICA1	HICA2	PSO	GA	Ins	HICA	HICA1	HICA2	PSO	GA
1	**1947.9**	1951.2	1995.2	2636.9	3712.0	49	**4446.0**	4590.6	4621.0	6549.2	9700.8
2	**2899.3**	2939.5	2942.2	3709.9	6153.5	50	**5821.7**	5953.0	5914.4	8411.9	13,050
3	4056.7	3955.3	4143.9	5777.0	8730.9	51	**7214.5**	7490.5	7518.9	10,617	16,315
4	**5446.6**	5526.2	5449.8	8083.3	11,836	52	8863.0	9336.3	8862.4	13,163	19,163
5	**6797.9**	7193.9	6974.0	9818.1	14,518	53	**10,815**	11,061	10,863	16,159	24,050
6	**8620.1**	8621.3	8806.1	13,108	19,976	54	13,078	12,916	13,164	18,194	28,495
7	10,628	11,140	10,581	15,763	24,182	55	**1457.2**	1485.1	1563.7	2063.0	3294.1
8	**12,735**	13,340	13,484	20,572	28,028	56	2200.6	2175.5	2272.0	3131.2	4824.8
9	**15,003**	15,609	15,595	22,002	33,668	57	**2971.2**	3022.6	3118.6	4401.6	6599.4
10	1441.9	1389.3	1447.3	1753.6	2843.6	58	4036.7	4011.7	4127.6	6000.7	8789.3
11	1995.1	1961.7	2026.7	2584.6	3997.1	59	**5111.9**	5209.3	5269.2	7187.7	11,781
12	**2768.9**	2808.3	2930.4	3660.3	6098.7	60	6440.0	6590.3	6297.7	9362.4	14,327
13	**3637.5**	3676.8	3662.6	5195.9	7896.9	61	8063.4	7895.1	7825.1	11,786	17,114
14	4690.8	4651.6	4666.4	6505.5	10,745	62	9562.6	9868.5	9389.9	14,110	21,170
15	**5794.6**	5947.0	5905.0	8133.6	13,215	63	11,010	11,718	10,968	16,094	24,475
16	7187.8	7240.0	6912.8	10,217	16,849	64	**1218.7**	1244.6	1251.2	1649.0	2486.9
17	8560.6	8704.4	8446.9	12,470	19,056	65	1811.4	1799.0	1861.6	2415.0	3744.6
18	**10,193**	10,564	10,443	14,676	21,849	66	2377.2	2256.4	2414.3	3352.6	5008.1
19	**1258.3**	1271.7	1300.7	1649.0	2463.2	67	3201.1	3026.8	3163.6	4429.4	6823.8
20	1849.5	1761.0	1864.6	2415.0	3804.5	68	**3965.6**	4109.1	4126.6	5557.0	9059.4
21	2407.2	2349.0	2306.0	3352.6	4799.2	69	5229.4	5077.0	5242.3	6796.8	11,469
22	**3078.0**	3206.2	3152.1	4429.4	7010.2	70	**6045.5**	6140.0	6142.0	8721.3	13,997
23	**4053.5**	4078.9	4204.0	5557.0	8623.0	71	**7116.5**	7490.0	7446.1	10,603	16,531
24	5115.2	5206.7	5061.5	6796.8	11,079	72	8699.2	8938.0	8669.7	12,517	19,977
25	**6118.8**	6176.3	6192.2	8721.3	14,144	73	**610.00**	616.00	629.00	658.60	830.40
26	**7376.1**	7518.1	7381.8	10,603	16,652	74	959.40	942.00	982.60	1039.6	1426.4
27	8615.0	8787.0	8576.6	12,517	19,640	75	1368.0	1379.0	1363.0	1645.0	2367.0
28	1040.1	1042.5	1039.4	1203.8	2223.8	76	**1849.6**	1850.8	1918.8	2117.6	3408.6
29	1587.2	1555.8	1629.8	1867.5	3365.1	77	**2451.6**	2453.0	2506.2	3205.8	4639.0
30	**1903.2**	1907.3	1920.9	2539.1	4515.7	78	**2950.8**	3054.0	3165.6	3832.0	6242.6
31	2616.5	2471.7	2442.0	3382.9	5927.7	79	432.42	428.30	433.23	474.95	654.86
32	3230.0	3270.3	3160.0	4384.6	7567.1	80	673.75	669.31	661.44	761.63	1064.3
33	**4035.4**	4145.3	4083.9	5429.3	9204.3	81	934.42	926.27	958.02	1047.5	1662.9
34	**4795.0**	4944.6	4820.8	6626.0	10,775	82	1251.8	1212.1	1238.4	1610.2	2465.2
35	**5640.9**	5786.4	5716.0	7713.6	13,871	83	**1657.8**	1676.9	1687.0	2108.7	3216.3
36	6665.7	6771.7	6563.8	9498.1	14,968	84	**2060.9**	2085.8	2144.0	2635.5	4429.1
37	2314.2	2305.0	2279.1	2931.8	4815.8	85	**385.15**	393.15	398.00	399.70	618.50
38	**3336.9**	3445.0	3498.3	4413.2	7375.8	86	592.20	588.80	600.20	649.80	997.20
39	**4813.1**	4814.1	5000.3	6990.1	11,026	87	826.00	823.25	804.25	939.50	1438.5
40	**6533.6**	6674.3	6567.7	9515.3	14,633	88	1090.6	1074.1	1066.3	1421.1	2135.7
41	**8428.2**	8635.5	8512.7	12,280	17,861	89	1411.2	1460.7	1399.4	1866.9	2630.2
42	**10,476**	10,903	11,013	16,216	24,908	90	1792.4	1753.4	1807.6	2458.2	3392.8
43	**12,963**	13,606	13,150	20,764	29,993	91	**382.17**	385.97	385.96	402.63	623.90
44	**15,960**	16,431	16,175	26,088	36,909	92	521.75	528.17	517.67	571.06	872.31
45	**18,729**	20,151	19,521	29,486	40,083	93	744.85	709.73	740.42	863.57	1364.2
46	1725.9	1719.2	1684.9	2248.0	3454.0	94	1015.1	999.42	1003.1	1061.7	1937.4
47	2490.6	2471.6	2529.5	3218.7	5230.0	95	1276.4	1249.3	1252.8	1662.1	2434.6
48	**3384.2**	3486.1	3590.9	5029.9	7563.4	96	1523.2	1547.1	1518.3	1950.4	3048.9

Table 4. Computational results of all algorithms on *Max*.

Ins	HICA	HICA1	HICA2	PSO	GA	Ins	HICA	HICA1	HICA2	PSO	GA
1	2144.4	2231.1	2135.2	2741.2	4489.2	49	**4907.3**	5028.7	4960.0	7012.8	11,949
2	3195.6	3145.4	3135.5	4217.9	7010.6	50	**6135.9**	6388.7	6339.7	8585.7	15,309
3	4487.2	4591.4	4399.2	6018.2	10,320	51	**7709.8**	7954.7	8009.6	11,287	19,728
4	**5844.4**	6022.8	6114.9	8785.6	13,535	52	**9615.9**	9732.5	9676.8	13,249	24,731
5	**7485.3**	7592.8	7492.8	10,935	18,122	53	**11,497**	12,124	11,927	17,396	29,209
6	**9212.9**	9670.0	9238.6	13,475	24,590	54	14,461	14,114	13,900	18,332	33,131
7	**11,610**	11,699	11,660	16,978	30,082	55	**1623.1**	1664.8	1696.0	2171.4	3453.6
8	**13,679**	14,853	13,811	21,057	33,401	56	**2345.7**	2389.5	2418.5	3224.0	5472.3
9	**16,449**	17,289	16,906	24,130	41,408	57	**3278.5**	3282.8	3397.8	4425.6	7303.3
10	1592.0	1578.1	1606.2	1767.6	3259.5	58	4423.0	4603.7	4315.9	6170.1	10,419
11	2237.9	2183.7	2301.8	2768.1	4922.6	59	**5414.5**	5712.1	5606.8	7863.8	13,143
12	**3153.2**	3166.3	3176.6	3733.1	7027.6	60	**6699.3**	7011.9	7050.5	9603.0	16,804
13	**3990.5**	4126.5	3996.5	5370.4	9961.5	61	8487.0	8826.4	8475.1	12,117	21,839
14	5078.6	5056.4	5153.8	6908.9	12,328	62	10,173	10,622	10,127	15,035	25,253
15	6458.6	6469.6	6439.0	8504.0	16,348	63	**11,809**	12,656	11,883	16,723	27,980
16	**7709.6**	7829.9	7780.9	10,600	19,084	64	1409.3	1332.3	1390.3	1737.7	3044.7
17	9359.8	9616.1	9325.9	13,029	23,261	65	2016.0	1949.4	1967.3	2490.0	4267.4
18	11,386	11,342	10,994	15,013	25,834	66	2649.0	2596.2	2618.4	3378.6	6094.5
19	1412.8	1366.8	1427.7	1737.7	3059.2	67	3519.9	3425.8	3506.4	4585.6	8016.1
20	2006.5	1985.1	2002.2	2490.0	4815.9	68	4522.9	4468.4	4555.9	5931.4	9955.4
21	2650.0	2787.6	2582.0	3378.6	6454.4	69	**5451.7**	5628.2	5587.4	7235.0	13,841
22	3606.4	3463.7	3619.5	4585.6	7786.0	70	**6604.2**	6632.2	6637.1	8975.3	15,733
23	4584.1	4514.8	4564.1	5931.4	11,277	71	8074.4	8222.1	7910.2	10,623	20,741
24	**5422.7**	5559.9	5716.4	7235.0	13,419	72	9606.8	9802.5	9444.3	12,899	25,562
25	6795.2	6810.9	6718.2	8975.3	16,083	73	**646.60**	667.40	653.80	693.60	1025.0
26	**7933.9**	8438.3	8176.4	10,623	21,031	74	**999.40**	1046.0	1092.2	1138.2	1806.8
27	**9674.3**	9684.2	9715.6	12,899	23,756	75	1604.0	1479.0	1491.0	1721.0	2717.0
28	**1147.1**	1161.5	1256.3	1282.4	2521.3	76	**2028.6**	2078.6	2119.4	2291.0	4023.0
29	1777.0	1752.3	1788.1	1915.5	3895.7	77	2790.4	2734.4	2747.2	3320.6	5453.2
30	2131.1	2120.4	2113.6	2722.6	5342.7	78	**3349.8**	3379.4	3386.0	4188.2	7193.8
31	2841.2	2818.7	2764.8	3451.9	6667.5	79	491.83	458.91	527.95	500.66	814.91
32	3537.3	3529.6	3565.9	4495.0	9292.2	80	**732.81**	762.75	752.81	768.50	1400.9
33	**4375.8**	4602.7	4703.6	5640.1	11,199	81	1071.0	1006.1	1051.7	1234.7	1940.1
34	5473.2	5220.8	5325.4	6843.2	13,188	82	**1331.9**	1368.9	1373.8	1729.3	3107.8
35	6285.0	6339.3	6253.3	8071.2	16,488	83	1830.6	1788.1	1854.8	2365.8	3732.5
36	7337.0	7576.3	7223.6	9712.7	18,942	84	2240.3	2229.4	2497.9	2856.4	5045.9
37	2550.9	2516.7	2578.2	3180.0	5511.4	85	438.30	434.15	458.80	416.30	780.65
38	3800.3	3735.2	3850.5	4939.9	8454.0	86	**643.60**	688.80	690.20	682.80	1243.8
39	5187.8	5174.7	5408.9	8057.2	13,239	87	**915.00**	921.00	927.75	1025.3	1810.5
40	**6857.6**	7199.7	7158.4	10,683	17,267	88	1241.3	1173.5	1214.1	1518.3	2545.8
41	9538.5	9458.1	9844.5	12,765	21,890	89	1604.5	1608.9	1555.2	2022.1	3274.0
42	**11,108**	12,032	11,719	16,862	27,020	90	1980.0	1896.6	1980.2	2532.2	4188.0
43	**14,073**	14,808	14,859	21,386	33,387	91	**402.38**	410.88	404.50	448.21	808.35
44	**17,680**	19,137	18,337	26,708	43,549	92	550.11	548.50	562.47	595.03	1139.2
45	**20,789**	21,749	21,480	30,067	51,569	93	**789.12**	798.17	842.88	963.72	1619.8
46	1876.7	1863.6	1828.8	2327.6	3892.5	94	1090.8	1086.6	1101.0	1164.4	2220.4
47	**2679.1**	2761.5	2709.0	3659.1	6012.3	95	**1366.2**	1377.7	1393.5	1717.5	2843.5
48	3804.7	3762.8	3795.2	5147.7	8219.2	96	**1632.9**	1662.9	1655.8	2036.9	3787.4

Table 5. Computational results of all algorithms on *Avg*.

Ins	HICA	HICA1	HICA2	PSO	GA	Ins	HICA	HICA1	HICA2	PSO	GA
1	**2035.6**	2064.8	2057.8	2691.4	4243.9	49	**4698.8**	4831.4	4797.1	6704.5	10,896
2	**3002.2**	3053.5	3057.6	3890.5	6473.0	50	**6022.4**	6155.7	6097.4	8524.4	14,070
3	4223.8	4215.4	4250.3	5914.6	9605.7	51	**7510.6**	7698.6	7701.1	10,903	17,516
4	**5613.9**	5708.7	5718.7	8243.4	12,941	52	**9201.3**	9481.6	9272.1	13,195	22,054
5	**7156.8**	7366.3	7289.2	10,046	16,776	53	**11,048**	11,536	11,241	16,924	26,378
6	**8880.1**	9150.4	8999.2	13,352	21,420	54	**13,393**	13,665	13,412	18,283	30,927
7	**11,038**	11,448	11,147	16,221	26,089	55	**1561.5**	1590.0	1617.9	2073.9	3356.0
8	**13,313**	13,907	13,714	20,684	30,579	56	**2276.5**	2299.9	2329.6	3181.0	5105.2
9	**15,651**	16,800	16,290	22,970	36,951	57	**3164.9**	3172.7	3239.1	4406.0	6951.7
10	1522.5	1472.0	1530.0	1755.0	3043.6	58	**4175.3**	4227.6	4214.1	6030.8	9334.8
11	2111.4	2065.0	2166.2	2646.1	4490.8	59	**5297.2**	5433.7	5417.4	7461.0	12,347
12	**2972.9**	3032.0	3019.6	3684.7	6490.1	60	**6574.7**	6818.3	6715.3	9450.2	15,622
13	**3843.1**	3867.9	3878.2	5249.3	8493.0	61	8213.3	8320.4	8141.0	11,827	18,956
14	**4878.4**	4894.2	4949.5	6728.2	11,613	62	**9800.6**	10,126	9867.1	14,551	23,048
15	**6107.9**	6211.7	6179.9	8303.1	14,497	63	**11,411**	12,159	11,623	16,234	26,497
16	**7436.2**	7618.5	7506.0	10,418	18,180	64	1300.3	1288.8	1312.9	1676.6	2809.3
17	8964.1	9100.8	8747.5	12,677	21,782	65	1912.5	1860.1	1907.4	2432.6	3957.0
18	**10,604**	10,933	10,713	14,783	24,038	66	**2470.2**	2488.7	2524.0	3366.4	5469.4
19	1329.5	1313.1	1351.9	1676.6	2834.0	67	3388.8	3252.1	3304.7	4504.5	7534.8
20	1931.2	1899.8	1934.5	2432.6	4163.4	68	**4196.4**	4248.4	4299.0	5696.7	9494.5
21	2530.2	2535.8	2510.0	3366.4	5683.4	69	5319.4	5246.2	5348.8	6980.8	12,284
22	**3327.8**	3382.8	3399.0	4504.5	7498.1	70	**6295.2**	6453.8	6364.9	8801.1	14,707
23	4302.8	4266.8	4399.1	5696.7	9992.6	71	**7568.2**	7845.5	7618.2	10,605	18,240
24	**5301.4**	5392.4	5404.8	6980.8	12,282	72	9046.8	9268.3	8952.3	12,709	21,217
25	**6453.6**	6542.3	6464.6	8801.1	15,046	73	**629.66**	640.70	639.68	665.20	904.20
26	**7662.4**	8003.4	7791.1	10,605	18,636	74	**978.68**	999.32	1029.1	1078.7	1633.1
27	**9113.0**	9250.6	9155.8	12,709	21,583	75	1437.0	1429.2	1441.5	1704.5	2509.7
28	**1090.1**	1098.8	1108.7	1243.2	2407.5	76	**1946.0**	1963.7	1990.0	2176.8	3767.7
29	1683.8	1657.5	1731.3	1883.6	3552.8	77	**2554.4**	2555.0	2585.6	3256.1	4988.9
30	1987.7	1984.8	2006.9	2586.7	4821.6	78	**3170.3**	3175.8	3234.2	3957.7	6701.8
31	2721.0	2649.2	2641.3	3403.7	6351.5	79	447.43	444.50	459.59	483.07	732.87
32	**3320.7**	3392.2	3361.7	4440.3	8187.9	80	707.75	703.47	716.30	765.96	1164.5
33	**4250.9**	4305.8	4354.5	5497.2	10,175	81	990.89	956.92	996.13	1111.8	1808.7
34	5097.6	5090.9	5083.6	6765.3	11,815	82	**1284.1**	1290.1	1319.8	1647.0	2638.4
35	**5963.4**	6081.7	5996.0	7853.2	15,015	83	1755.9	1729.6	1753.0	2196.7	3525.7
36	6958.3	7132.0	6872.4	9619.8	17,167	84	**2146.8**	2160.8	2207.6	2782.2	4720.6
37	2398.8	2381.6	2430.1	3104.5	5079.1	85	416.04	416.14	413.36	403.54	699.71
38	3602.8	3588.7	3640.2	4562.0	7939.3	86	**631.92**	642.00	637.68	665.32	1125.4
39	**4982.1**	5027.9	5154.4	7369.9	11,807	87	873.58	872.88	881.6	972.38	1606.3
40	**6692.5**	6868.9	6792.8	9987.4	15,872	88	1171.3	1144.8	1168.1	1469.8	2278.1
41	8871.3	8922.3	8868.9	12,601	20,413	89	1518.7	1520.0	1512.4	1923.2	3007.6
42	**10,801**	11,431	11,361	16,432	25,801	90	1852.1	1831.6	1877.9	2477.5	3879.1
43	**13,501**	14,028	13,789	20,970	31,844	91	**390.75**	394.54	394.75	414.03	717.61
44	16,891	17,488	16,871	26,544	39,894	92	**538.26**	541.04	543.13	574.71	1029.5
45	**19,930**	20,912	20,373	29,599	45,940	93	768.41	759.94	782.24	879.57	1512.9
46	1773.2	1783.7	1772.7	2265.0	3703.7	94	1045.9	1050.0	1037.7	1126.4	1998.2
47	**2586.4**	2614.8	2608.8	3321.4	5623.1	95	1319.0	1302.5	1335.8	1669.5	2674.3
48	3649.5	3640.2	3689.0	5056.9	7902.6	96	1596.7	1604.3	1581.4	1973.4	3432.8

Table 6. Computational results of the five algorithms on metric *Std*.

Instance	HICA	HICA1	HICA2	PSO	GA	Instance	HICA	HICA1	HICA2	PSO	GA
1	64.592	81.659	61.212	**36.281**	218.84	49	123.48	**114.59**	123.40	185.23	580.19
2	80.080	62.621	**59.686**	159.12	246.46	50	98.966	122.97	138.13	**57.304**	661.26
3	124.15	151.02	79.150	**74.637**	498.95	51	**132.39**	141.64	143.98	257.70	886.46
4	**114.59**	171.28	176.50	253.56	484.62	52	237.57	139.02	247.10	**39.833**	1537.1
5	192.64	**123.85**	176.14	341.27	1023.7	53	**210.31**	304.32	308.27	327.58	1502.1
6	216.92	304.43	111.85	**103.13**	1263.3	54	403.67	322.32	220.00	**61.193**	1580.5
7	293.76	**166.53**	340.71	382.55	1584.7	55	55.726	64.038	40.212	**32.523**	58.745
8	236.31	418.52	**101.54**	176.97	1451.9	56	45.996	65.209	48.409	**32.398**	174.61
9	461.58	511.25	**385.46**	892.76	2096.4	57	92.341	69.442	83.776	**9.0250**	237.24
10	40.789	50.186	53.977	**4.2000**	132.89	58	124.74	166.33	66.110	**48.322**	477.68
11	81.350	69.732	77.438	**62.690**	248.87	59	**90.399**	155.12	125.42	225.83	418.85
12	97.549	117.37	73.253	**20.745**	253.90	60	86.427	135.58	224.19	**75.957**	856.96
13	102.54	125.53	89.390	**63.110**	598.38	61	113.29	286.39	192.64	**96.859**	1177.7
14	127.77	143.34	153.85	**112.01**	545.43	62	**186.12**	240.55	222.73	358.68	1133.1
15	175.88	168.69	180.15	**138.90**	971.14	63	245.48	244.08	254.60	**165.99**	1161.0
16	**134.55**	164.77	257.70	156.08	768.97	64	52.725	27.652	47.187	**25.169**	192.51
17	204.76	246.05	260.76	**184.94**	1049.3	65	58.271	48.583	32.906	**25.187**	158.90
18	317.30	203.60	167.80	**96.781**	1246.0	66	80.386	110.76	61.890	**12.278**	346.99
19	40.544	28.394	35.943	**25.169**	171.73	67	113.83	101.46	102.78	**37.022**	372.04
20	40.077	72.506	40.059	**25.187**	294.41	68	172.50	107.70	136.68	**98.365**	215.83
21	70.752	132.42	76.236	**12.278**	430.30	69	**67.291**	160.17	107.15	129.48	606.22
22	147.43	84.221	131.12	**37.022**	267.88	70	196.40	167.57	129.13	**93.038**	546.25
23	161.07	126.98	102.95	**98.365**	739.01	71	248.17	235.40	127.08	**5.8524**	1116.1
24	**108.45**	116.11	181.21	129.48	740.44	72	292.83	263.67	221.59	**98.095**	1515.6
25	214.52	197.63	179.32	**93.038**	643.46	73	12.360	15.480	**8.1507**	10.892	69.643
26	164.00	260.50	253.46	**5.8524**	1313.1	74	**16.101**	34.463	30.366	26.850	115.16
27	294.44	298.82	307.09	**98.095**	1335.2	75	63.887	25.463	37.025	**21.139**	105.22
28	33.956	31.167	57.815	**23.557**	107.58	76	50.010	76.592	57.765	**49.321**	208.33
29	61.444	55.095	50.979	**13.568**	171.47	77	97.412	78.212	66.040	**43.161**	226.59
30	68.121	68.678	**53.899**	54.716	284.16	78	126.09	93.546	**65.632**	114.13	292.21
31	77.914	104.96	89.017	**20.532**	224.80	79	17.243	10.743	25.834	**8.8486**	46.016
32	89.028	77.661	122.22	**41.395**	528.60	80	16.777	33.587	24.106	**3.1662**	89.043
33	107.50	129.55	168.55	**74.696**	596.14	81	37.285	**26.053**	28.279	60.188	75.397
34	200.77	89.597	165.51	**69.103**	606.58	82	**24.203**	49.112	35.666	46.434	183.81
35	184.90	161.39	170.09	**101.9**	743.40	83	63.174	**36.079**	48.430	79.620	138.93
36	207.98	250.55	209.06	**63.564**	1022.2	84	60.590	**40.744**	103.70	86.945	180.85
37	64.053	**58.114**	92.794	89.894	228.30	85	15.611	12.697	16.776	**5.0279**	54.738
38	136.51	**91.828**	113.95	170.77	386.66	86	14.511	29.229	24.374	**12.273**	64.962
39	126.86	**117.41**	120.42	379.27	585.97	87	34.738	**29.642**	36.062	30.091	115.75
40	**116.69**	155.74	185.42	319.56	866.03	88	45.453	**28.014**	42.436	28.557	115.37
41	316.88	240.44	366.72	**153.21**	1316.8	89	50.914	45.798	**41.662**	48.458	209.48
42	195.24	302.56	**174.97**	276.65	828.45	90	51.044	53.206	61.197	**20.645**	225.61
43	310.43	404.96	537.20	**184.91**	1011.0	91	6.0981	7.3517	**5.8396**	16.294	46.835
44	519.81	713.56	660.90	**252.45**	2129.6	92	9.1908	**6.0423**	12.694	7.2325	92.698
45	579.61	572.59	665.16	**181.97**	3264.7	93	**12.683**	29.535	32.935	29.320	75.813
46	41.188	48.871	44.995	**28.589**	135.69	94	**22.694**	29.216	27.806	41.225	81.823
47	70.778	103.05	**64.466**	145.58	259.37	95	31.513	37.417	40.741	**16.417**	129.29
48	121.14	81.980	66.900	**30.808**	249.62	96	32.529	37.904	46.590	**31.404**	237.22

Table 7. Results of the Wilcoxon test.

Wilcoxon-Test	*Min*	*Max*	*Avg*	*Std*
Wilcoxon test (H, H1)	0.000	0.001	0.000	0.262
Wilcoxon test (H, H2)	0.000	0.000	0.000	0.764
Wilcoxon test (H, P)	0.000	0.000	0.000	0.002
Wilcoxon test (H, G)	0.000	0.000	0.000	0.000

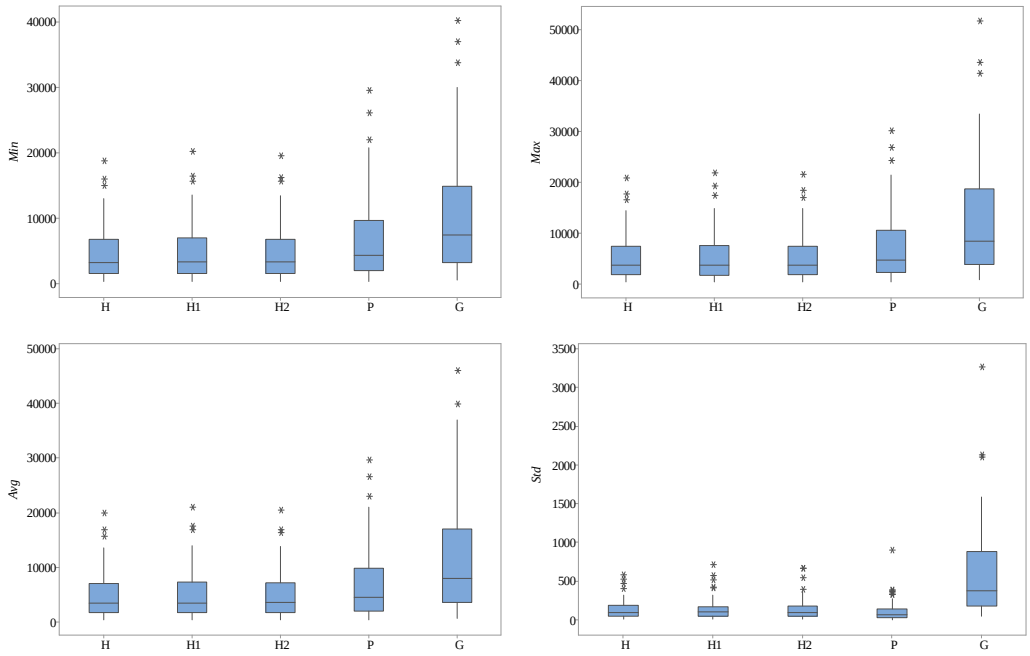

Figure 3. Box plot of the five algorithms.

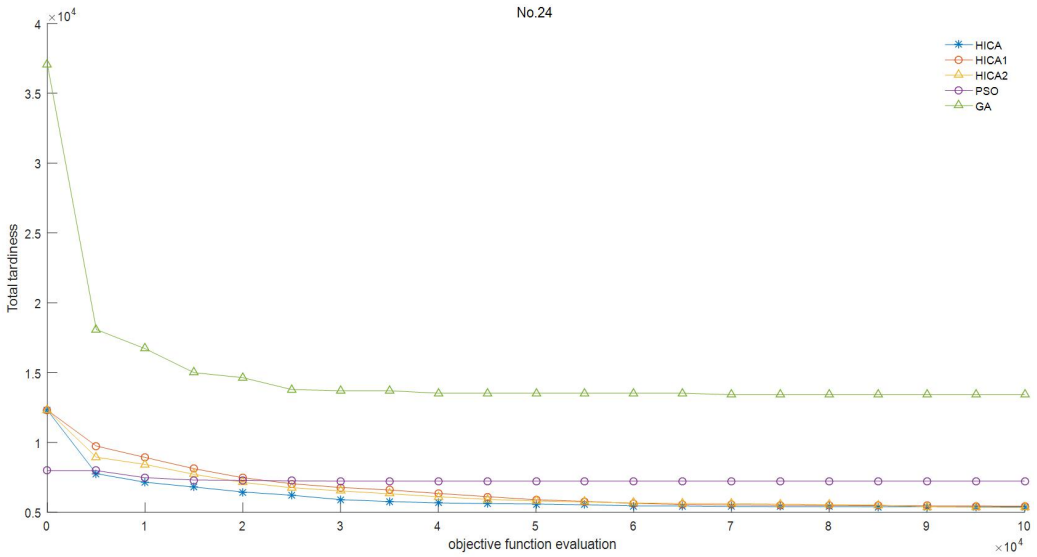

Figure 4. Convergence curves on an instance of Combination 24.

It can be found from Tables 3–6 that the HICA performed better than its two variants on the three metrics. The HICA produced better *Min* than HICA1 on 65 combinations; the *Max* of HICA1 was less than that of the HICA on 37 combinations; the HICA had a smaller *Avg* than HICA1 on 69 combinations. Obviously, the HICA had better convergence performance than HICA1, and the inclusion of the local search really improved the performance of the HICA. HICA2 could not generate better results than the HICA on most of the instances and was superior to the HICA on a limited number of instances. In the HICA, assimilation was implemented differently in different types of empires. The comparison between the HICA and HICA2 really proved the effectiveness of the diversified assimilation. Table 7 reveals that the HICA had a better *Min*, *Max*, and *Avg* than HICA1 and HICA2 in the statistical sense and produced a similar *Std* as its two variants; this conclusion also can be seen from Figures 3–5. Thus, it is necessary to add the local search and the diversified assimilation to the HICA.

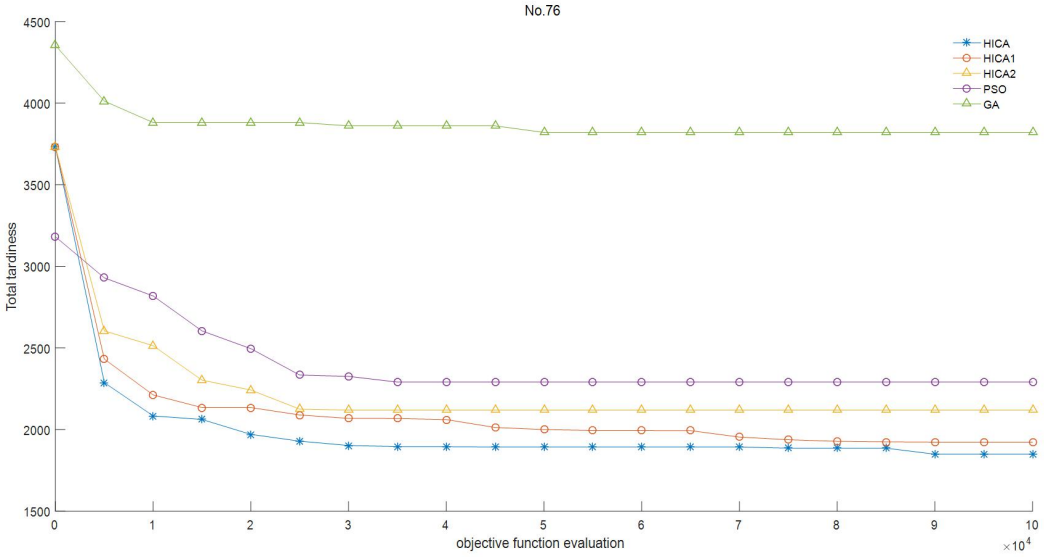

Figure 5. Convergence curves on an instance of Combination 76.

The parameters of the GA and the PSO algorithm were directly adopted from Behnamian and Fatemi Ghomi [4] and Hulett et al. [50], except the stopping condition. Three algorithms had the same termination condition: $max\_it = 10^5$. The computational results and times are shown in Tables 3–6 and 8, respectively.

Table 8. Computational times of the HICA and the two comparative algorithms.

Ins	Running Time (s)			Ins	Running Time (s)			Ins	Running Time (s)			Ins	Running Time (s)		
	HICA	PSO	GA		HICA	PSO	GA		HICA	PSO	GA		HICA	PSO	GA
1	4.08	7.04	47.3	25	13.3	46.8	146	49	8.34	15.9	61.1	73	1.29	1.57	6.98
2	5.22	10.1	58.0	26	14.9	51.2	162	50	9.71	19.8	69.8	74	1.89	2.48	9.05
3	6.61	12.7	70.8	27	16.5	49.5	179	51	11.7	22.6	79.5	75	2.34	2.91	11.2
4	8.26	14.8	82.7	28	5.32	20.9	80.6	52	13.4	26.1	92.9	76	3.06	3.56	14.1
5	10.3	18.1	98.4	29	6.53	29.3	90.9	53	15.5	28.8	108	77	3.60	4.35	17.3
6	12.0	19.8	111	30	7.84	35.9	113	54	17.4	36.0	117	78	4.36	5.26	20.6
7	14.2	25.5	126	31	9.28	38.6	131	55	4.10	11.5	31.6	79	1.39	2.17	7.79
8	15.9	26.4	140	32	11.0	50.1	147	56	5.37	13.5	39.1	80	1.68	2.89	9.40
9	18.2	30.7	148	33	12.4	56.1	166	57	6.47	15.0	45.4	81	2.17	3.54	12.7
10	4.27	12.4	61.0	34	13.8	57.8	179	58	8.20	18.1	55.3	82	2.87	4.76	14.8
11	5.44	13.8	74.3	35	15.8	69.4	203	59	9.78	21.3	68.7	83	3.32	5.47	17.2
12	6.70	19.9	86.5	36	17.5	72.6	226	60	11.4	26.1	74.9	84	3.87	6.20	20.0
13	8.28	22.0	105	37	4.13	5.88	33.9	61	13.1	29.0	84.8	85	1.49	2.30	7.15
14	9.81	28.2	121	38	5.39	7.87	48.1	62	14.5	32.8	95.4	86	1.75	3.26	9.48
15	11.2	28.5	138	39	6.96	10.3	56.4	63	17.0	39.8	105	87	2.36	4.18	11.6
16	13.2	32.6	155	40	8.85	12.8	65.4	64	4.47	16.1	49.5	88	2.71	5.43	13.7
17	15.0	40.7	173	41	10.8	15.0	72.9	65	5.67	17.1	56.1	89	3.30	7.80	16.3
18	16.9	42.5	186	42	12.6	17.7	80.9	66	6.92	21.0	70.6	90	3.79	9.15	18.7
19	4.45	15.1	56.0	43	15.0	20.6	90.1	67	8.31	25.5	82.3	91	1.49	2.91	6.67
20	5.78	18.7	71.5	44	16.8	24.3	99.1	68	10.3	32.6	94.3	92	1.93	3.95	8.95
21	7.08	21.0	85.1	45	20.0	26.9	107	69	11.4	34.3	99.1	93	2.38	5.40	10.8
22	8.33	26.7	103	46	4.00	9.93	34.4	70	13.1	44.4	124	94	2.96	6.13	13.4
23	9.95	29.7	117	47	5.29	10.4	42.4	71	14.8	49.0	140	95	3.43	7.93	15.7
24	11.3	39.8	133	48	6.43	13.1	51.3	72	16.4	48.1	144	96	3.91	9.12	18.3

As shown in Tables 3–7, the HICA generated a smaller *Min* than the two comparative algorithms on all instances, that is the HICA converged significantly better than the PSO algorithm and the GA. The *Max* and *Avg* of the HICA were less than those of the PSO algorithm on 95 of 96 combinations and better than those of the GA on all combinations.

The HICA obtained a smaller *Std* than the GA on most of the instances. The performance differences on the *Max*, *Min*, and *Avg* with respect to the HICA, PSO algorithm, and GA also can be found from Figures 3–5. Although HICA performed worse than the PSO algorithm on the *Std*, the HICA possessed better convergence, smaller average results, and a smaller *Max* than its two comparative algorithms.

In the HICA, The strongest empire was excluded from imperialist competition to avoid premature removal. The diversified assimilation and new revolution were implemented differently in the different types of empires, and the local search could effectively improve the search efficiency; on the contrary, the PSO algorithm and GA have a strong global search ability; however, their local search ability was not intensified, and this feature was the main reason for their low performance. Thus, the HICA can effectively solve the DUPMSP.

5. Conclusions

In this study, a new algorithm by hybridizing the ICA with the knowledge-based local search was proposed to solve the DUPMSP with total tardiness minimization, in which empires were divided into three types. To obtain high-quality solutions, the diversified assimilation and new revolution were designed; imperialist competition was newly implemented on $N_{im} - 1$ empires without the strongest one; the problem-related properties were proven; the knowledge-based local search was applied to improve the quality of the imperialists. Extensive experiments were conducted on 480 instances. The computational results demonstrated that the new strategies were effective and that the HICA had promising advantages in the considered DUPMSP.

The DPMSP is an important scheduling topic. We will focus on the DPMSP with various constraints such as additional resources and machine eligibility and try to solve the problem by using meta-heuristics with new optimization mechanisms such as reinforcement learning. Other distributed scheduling problems including distributed assembly hybrid flow shop scheduling are also our future topics. We will solve distributed scheduling problems with factory eligibility or energy-related constraints by using reinforcement-learning-based meta-heuristics.

Author Contributions: Methodolog, Y.Z.; computation experiments, Y.Z.; writing, Y.Z.; methodology, Y.Y.; computation experiments; Q.Z.; Writing, D.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Natural Science Foundation of China (61803149).

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

References

1. Hooker, J.N. A hybrid method for the planning and scheduling. *Constraints* **2005**, *10*, 385–401. [CrossRef]
2. Chen, Z.L.; Pundoor, G. Order assignment and scheduling in a supply chain. *Oper. Res.* **2006**, *54*, 555–572. [CrossRef]
3. Terrazas, M.; Grossmann, I.E. A multiscale decomposition method for the optimal planning and scheduling of multi-site continuous multiproduct plants. *Chem. Eng. Sci.* **2011**, *66*, 4307–4318. [CrossRef]
4. Behnamian, J.; Ghomi, S.F. The heterogeneous multi-factory production network scheduling with adaptive communication policy and parallel machine. *Inform. Sci.* **2013**, *219*, 181–196. [CrossRef]
5. Behnamian, J. Decomposition based hybrid VNS-TS algorithm for distributed parallel factories sceduling with virtual corporation. *Comput. Oper. Res.* **2014**, *52*, 181–191. [CrossRef]
6. Behnamian, J.; Ghomi, S.F. Minimizing cost-related objective in synchronous scheduling of parallel factories in the virtual production network. *Appl. Soft Comput.* **2015**, *29*, 221–232. [CrossRef]
7. Behnamian, J. Graph colouring-based algorithm to parallel jobs scheduling on parallel factories. *Int. J. Prod. Res.* **2016**, *29*, 622–635. [CrossRef]
8. Lei, D.M.; Yuan, Y.; Cai, J.C.; Bai, D.Y. An imperialist competitive algorithm with memory for distributed unrelated parallel machines scheduling. *Int. J. Prod. Res.* **2020**, *58*, 597–614. [CrossRef]
9. Lei, D.M.; Liu, M.Y. An artificial bee colony with division for distributed unrelated parallel machine scheduling with preventive maintenance. *Comput. Ind. Eng.* **2020**, *141*, 106320. [CrossRef]
10. Lei, D.M.; Yuan, Y.; Cai, J.C. An improved artificial bee colony for multi objective distributed unrelated parallel machine scheduling. *Int. J. Prod. Res.* **2020**, *59*, 5259–5271. [CrossRef]

11. Pan, Z.X.; Lei, D.M.; Wang, L. A knowledge-based two-population optimization algorithm for distributed energy-effificient parallel machines scheduling. *IEEE Trans. Cyber*. 2021, *in press*.
12. Zhao, F.; Zhao, L.; Wang, L.; Song, H. An ensemble discrete differential evolution for the distributed blocking flowshop scheduling with minimizing makespan criterion. *Exp. Syst. Appl*. **2020**, *160*, 113678. [CrossRef]
13. Shao, Z.; Pi, D.; Shao, W. Hybrid enhanced discrete fruit fly optimization algorithm for scheduling blocking flow-shop in distributed environment. *Exp. Syst. Appl*. **2020**, *145*, 113147. [CrossRef]
14. Chen, S.; Pan, Q.-K.; Gao, L.; Sang, H.-Y. A population-based iterated greedy algorithm to minimize total flowtime for the distributed blocking flowshop scheduling problem. *Eng. Appl. Artif. Intel*. **2021**, *104*, 104375. [CrossRef]
15. Ribas, I.; Companys, R.; Tort-Martorell, X. An iterated greedy algorithm for the parallel blocking flow shop scheduling problem and sequence-dependent setup times. *Exp. Syst. Appl*. **2021**, *184*, 115535. [CrossRef]
16. Li, Y.-Z.; Pan, Q.-K.; Li, J.-Q.; Gao, L.; Tasgetiren, M.F. An adaptive iterated greedy algorithm for distributed mixed no-idle permutation flowshop scheduling problems. *Swarm Evol. Comput*. **2021**, *63*, 100874. [CrossRef]
17. Lu, C.; Gao, L.; Gong, W.; Hu, C.; Yan, X.; Li, X. Sustainable scheduling of distributed permutation flow-shop with non-identical factory using a knowledge-based multi-objective memetic optimization algorithm. *Swarm Evol. Comput*. **2021**, *60*, 100803. [CrossRef]
18. Cai, J.C.; Zhou, R.; Lei, D.M. Dynamic shuffled frog-leaping algorithm for distributed hybrid flow shop scheduling with multiprocessor tasks. *Eng. Appl. Artif. Intel*. **2020**, *90*, 103540. [CrossRef]
19. Jiang, E.D.; Wang, L.; Wang, J.J. Decomposition-based multi-objective optimization for energy-aware distributed hybrid flow shop scheduling with multiprocessor tasks. *Tsinghua Sci. Technol*. **2021**, *26*, 646–663. [CrossRef]
20. Cai, J.C.; Lei, D.M.; Li, M. A shuffled frog-leaping algorithm with memeplex quality for bi-objective distributed scheduling in hybrid flow shop. *Int. J. Prod. Res.* **2021**, *59*, 5404–5421. [CrossRef]
21. Zheng, J.; Wang, L.; Wang, J.J. A cooperative coevolution algorithm for multi-objective fuzzy distributed hybrid flow shop. *Know-Based Syst*. **2020**, *194*, 105536. [CrossRef]
22. Wang, L.; Li, D.D. Fuzzy distributed hybrid flow shop scheduling problem with heterogeneous factory and unrelated parallel machine: A shuffled frog leaping algorithm with collaboration of multiple search strategies. *IEEE Access* **2020**, *8*, 214209–214223. [CrossRef]
23. Cai, J.C.; Lei, D.M. A cooperated shuffled frog-leaping algorithm for distributed energy-efficient hybrid flow shop scheduling with fuzzy processing time. *Complex Intel. Syst*. **2021**, *7*, 2235–2253. [CrossRef]
24. Lei, D.M.; Wang, T. Solving distributed two-stage hybrid flowshop scheduling using a shuffled frog-leaping algorithm with memeplex grouping. *Eng. Optim*. **2020**, *52*, 1461–1474. [CrossRef]
25. Cai, J.C.; Zhou, R.; Lei, D.M. Fuzzy distributed two-stage hybrid flow shop scheduling problem with setup time: Collaborative variable search. *J. Intel. Fuzzy Syst*. **2020**, *38*, 3189–3199. [CrossRef]
26. Shao, Z.S.; Shao, W.S.; Pi, D.C. Effective constructive heuristic and metaheuristic for the distributed assembly blocking flow-shop scheduling problem. *Appl. Intel*. **2020**, *50*, 4647–4649. [CrossRef]
27. Zhao, F.Q.; Zhao, J.L.; Wang, L.; Tang, J.X. An optimal block knowledge driven backtracking search algorithm for distributed assembly No-wait flow shop scheduling problem. *Appl. Soft Comput*. **2021**, *112*, 107750. [CrossRef]
28. Zhang, G.; Xing, K.; Cao, F. Scheduling distributed flowshops with flexible assembly and set-up time to minimize makespan. *Int. J. Prod. Res.* **2018**, *56*, 3226–3244. [CrossRef]
29. Lin, J.; Zhang, S. An effective hybrid biogeography-based optimization algorithm for the distributed assembly permutation flow-shop scheduling problem. *Comput. Ind. Eng*. **2016**, *97*, 128–136. [CrossRef]
30. Lin, J.; Wang, Z.J.; Li, X.D. A backtracking search hyper-heuristic for the distributed assembly flow-shop scheduling problem. *Swarm Evol. Comput*. **2017**, *36*, 124–135. [CrossRef]
31. Farshi, T.R. Battle royale optimization algorithm. *Neural Comput. Appl.* 2020, *in press*.
32. Seyyedabbasi, A.; Kiani, F. I-GWO and Ex-GWO: Improved algorithms of the grey wolf optimizer to solve global optimization problems. *Eng. Comput.* **2021**, *37*, 509–532. [CrossRef]
33. Nadimi-Shahraki, M.H.; Fatahi, A.; Zamani, H. Migration-based moth-flame optimization algorithm. *Processes* **2021**, *9*, 2276. [CrossRef]
34. Hosseini, S.; Khaled, A.A. A survey on the imperialist competitive algorithm metaheuristic: Implementation in engineering domain and directions for future research. *Appl. Soft Comput*. **2014**, *24*, 1078–1094. [CrossRef]
35. Lei, D.M.; Cai, J.C. Multi-population meta-heuristics for production scheduling: A survey. *Swarm Evol. Comput.* **2020**, *58*, 100739. [CrossRef]
36. Banisadr, A.H.; Zandieh, M.; Mahdavi, I. A hybrid imperialist competitive algorithm for single-machine scheduling problem with linear earliness and quadratic tardiness penalties. *Int. J. Adv. Manuf. Technol.* **2013**, *65*, 981–989. [CrossRef]
37. Shokrollahpour, E.; Zandieh, M.; Dorri, B. A novel imperialist competitive algorithm for bi-criteria scheduling of the assembly flow shop problem. *Int. J. Prod. Res.* **2011**, *49*, 3087–3103. [CrossRef]
38. Seidgar, H.; Kiani, M.; Abedi, M.; Fazlollahtabar, H. An efficient imperialist competitive algorithm for scheduling in the two-stage assembly flow shop problem. *Int. J. Prod. Res*. **2014**, *52*, 1240–1256. [CrossRef]
39. Zandieh, M.; Kahmati, A.R.; Rahmati, S.H.A. Flexible job shop scheduling under condition-based maintenance: Improved version of imperialist competitive algorithm. *Appl. Soft Comput.* **2017**, *58*, 449–464. [CrossRef]

40. Karimi, S.; Ardalan, Z.; Naderi, B.; Mohammadi, M. Scheduling flexible job-shops with transportation times: Mathematical models and a hybrid imperialist competitive algorithm. *Appl. Math. Model.* **2017**, *41*, 667–682. [CrossRef]
41. Lei, D.M.; Li, M.; Wang, L. A two-phase meta-heuristic for multi-objective flexible job shop scheduling problem with total energy consumption threshold. *IEEE Trans. Cyber.* **2019**, *49*, 1097–1109. [CrossRef]
42. Abedi, M.; Seidgar, H.; Fazlollahtabar, H.; Bijani, R. Bi-objective optimisation for scheduling the identical parallel batch-processing machines with arbitary job sizes, unequal job release times and capacity limits. *Int. J. Prod. Res.* **2015**, *53*, 1680–1711. [CrossRef]
43. Yazdani, M.; Khalili, S.M.; Jolai, F. A parallel machine scheduling problem with two-agent and tool change activities: An efficient hybrid metaheuristic algorithm. *Int. J. Comput. Int. Manuf.* **2016**, *29*, 1075–1088. [CrossRef]
44. Zhang, P.; Lv, Y.L.; Zhang, J. An improved imperialist competitive algorithm based photolithography machines scheduling. *Int. J. Prod. Res*. **2018**, *56*, 1017–1029. [CrossRef]
45. Li, M.; Su, B.; Lei, D.M. A Novel imperialist competitive algorithm for fuzzy distributed assembly flow shop Scheduling. *J. Intel. Fuzzy Syst.* **2021**, *40*, 4545–4561. [CrossRef]
46. Li, M.; Lei, D.M. An imperialist competitive algorithm with feedback for energy-efficient flexible job shop scheduling with transportation and sequence-dependent setup times. *Eng. Appl. Artif. Intel*. **2021**, *103*, 104307. [CrossRef]
47. Zheng, X.L.; Wang, L. A collaborative multiobjective fruit fly optimization algorithm foe the resource constrained unrelated parallel machine green scheduling problem. *IEEE Trans. Syst. Man, Cyber. Syst.* **2018**, *48*, 790–800. [CrossRef]
48. Wang, J.J.; Wang, L. A knowledge-Based cooperative algorithm for energy-efficient scheduling of distributed flow-shop. *IEEE Trans. Syst. Man Cyber. Syst.* **2020**, *50*, 1805–1819. [CrossRef]
49. Wang, L.; Zheng, X.L. A knowledge-guided multi-objective fruit fly optimization algorithm for the multi-skill resource constrained project scheduling problem. *Swarm Evol. Comput*. **2018**, *38*, 54–63. [CrossRef]
50. Hulett, M.; Damodaran, P.; Amouie, M. Scheduling non-identical parallel batch processing machines to minimize total weighted tardiness using particle swarm optimization. *Comput. Ind. Eng*. **2017**, *113*, 425–436. [CrossRef]