symmetry

MDPI

Article

An Artificial Bee Colony with Adaptive Competition for the Unrelated Parallel Machine Scheduling Problem with Additional Resources and Maintenance

Mingbo Li, Huan Xiong and Deming Lei *

School of Automation, Wuhan University of Technology, Wuhan 430062, China; feng10111217@whut.edu.cn (M.L.); bearh@whut.edu.cn (H.X.)
* Correspondence: deminglei11@whut.edu.cn or deminglei11@163.com

Abstract: The unrelated parallel machine scheduling problem (UPMSP) is a typical production scheduling problem with certain symmetries on machines. Additional resources and preventive maintenance (PM) extensively exist on parallel machines; however, UPMSP with additional resources and PM has been scarcely investigated. Adaptive competition is also seldom implemented in the artificial bee colony algorithm for production scheduling. In this study, UPMSP with additional resources and PM is investigated, which has certain symmetries with machines. An artificial bee colony with adaptive competition (ABC-AC) is proposed to minimize the makespan. Two employed bee swarms are constructed and evaluated. In the employed bee phase, adaptive competition is used to dynamically decide two cases. The first is the shifting of search resources from the employed bee swarm with a lower evolution quality to another one, and the second is the migration of solutions from the employed bee swarm with a higher evolution quality to another one. An adaptive onlooker bee phase and a new scout phase are given. Extensive experiments are conducted on 300 instances. The computational results demonstrate that the new strategies of ABC-AC are effective, and ABC-AC provides promising results for the considered UPMSP.

Keywords: scheduling; unrelated parallel machine; artificial bee colony; additional resource; competition

Citation: Li, M.; Xiong, H.; Lei, D. An Artificial Bee Colony with Adaptive Competition for the Unrelated Parallel Machine Scheduling Problem with Additional Resources and Maintenance. *Symmetry* **2022**, *14*, 1380. https://doi.org/10.3390/sym14071380

Academic Editor: Basil Papadopoulos

Received: 19 May 2022
Accepted: 23 June 2022
Published: 5 July 2022

1. Introduction

In the past decades, UPMSP has been extensively considered, and in most of works on UPMSP, the machine is the only considered resource; however, additional resources often exist in many real-world parallel machine manufacturing process. Additional resources include automated guided vehicles, machine operators, tools, pallets, dies and industrial robots, and the total number of the used additional resources on each machine cannot exceed a given threshold at any time. Unrelated parallel machine scheduling problem with additional resources (UPMSPR) has become a significant area of scheduling research, and many results have been obtained [1–16].

There are two types of UPMSPR. The first is UPMSPR with one additional resource [2–8]. Zheng and Wang [3] proposed a two-stage adaptive fruit fly optimization algorithm (TAFOA) with a heuristic and knowledge-guided search for the problem with renewable resource. Fanjul-Peyro et al. [4] presented two integer linear programming models and three math-euristics. Fleszar and Hindi [5] presented an efficient mixed-integer linear programming (MILP) model and a constraint programming model.

Zheng and Wang [6] developed a collaborative multi-objective fruit fly optimization algorithm for UPMSPR with carbon emission minimization. Villa et al. [7] gave several heuristics based on resource constraint and assignment rule. Vallada et al. [8] applied an enriched scatter search and an enriched iterated greedy. The second is UPMSPR with multiple additional resources [9–16].

UPMSP with some types of dies as second resources [9] and UPMSP with auxiliary resources in a photolithography workshop are solved by heuristic and memetic algorithms. Afzalirad and Shafipour [11] presented an integer mathematical programming model and two genetic algorithms (GA) for UPMSPR with eligibility restrictions. Al-Harkan and Qamhan [12] developed a two-stage hybrid meta-heuristic for solving UPMSPR with non-zero arbitrary release dates and sequence-dependent setup times (SDST). UPMSPR with more than two conditions and constraints have also been studied [13–16], including processing resources, setup resources and shared resources [13], setup times and additional limited resources in setup [14], SDST, precedence relation, machine eligibility and release dates [15] and release dates, SDST [16]. These problems are handled by using a three-phase algorithm [13], heuristics and GRASP algorithm [14] and a modified harmony search algorithm [16].

Generally, preventive maintenance (PM) is an effective way to prevent potential failures and serious accidents in parallel machines. Regarding UPMSP with PM, Yang et al. [17] found that UPMSP with aging effects, PM and total machine load minimization can be solved by polynomial algorithm. Tavana et al. [18] proposed a three-stage maintenance scheduling model for UPMSP with aging effect and multi-maintenance activities. Wang and Liu [19] proposed an improved non-dominated sorting genetic algorithm-II for UPMSP with multi-resource PM planning.

Several performance criteria, different maintenance systems and a new method are presented for UPMSP with deteriorating maintenance [20]. UPMSP with PM and SDST is often considered by using a meta-heuristic with a multi-start strategy [21], a novel imperialist competitive algorithm with an estimation of distribution algorithm and an artificial bee colony (ABC) with swarm division and swarm updating with optimization data [22]. Pang et al. [23] proposed a feature-extraction-based iterated algorithm for UPMSP with release times and maintenance activities. Lei and Yi [24] presented a differentiated shuffled frog-leaping algorithm for solving UPMSP with deteriorating PM and SDST.

As stated above, additional resources and PM are frequently considered constraints in many parallel machine processes, and some results have been obtained on UPMSP with these two constraints. Although additional resources and PM often appear simultaneously in parallel machine shops, optimization results of UPMSP including these can effectively reflect real-life situations, and the obtained schedule has high application value, UPMSP with these two constraints is seldom studied; thus, it is necessary to solve UPMSP with additional resources and PM.

On the other hand, UPMSPR and UPMSP with PM are solved with polynomial time algorithms, heuristics and meta-heuristics; moreover, meta-heuristics have been successfully applied to solve the above two UPMSP [25,26]. As typical stochastic optimization methods [27], meta-heuristics are not applied to solve UPMSP with the above two constraints, and some meta-heuristics, such as ABC, are seldom used to deal with UPMSPR and UPMSP with PM, let alone UPMSPR with PM; thus, meta-heuristics, including ABC should be studied well to obtain competitive approaches to UPMSPR with PM.

ABC is a meta-heuristic inspired by the intelligent foraging behavior of honeybee swarms [28]. It has features including simplicity and the ease of implementation. In the past decade, as a main approach to optimization problems [29,30], ABC has been successfully applied to solve various production scheduling problems [31–41]. ABC also has been successfully applied to solve UPMSP [34,35,37,41,42], and some ABC algorithms were proposed, including hybrid ABC with iterated greedy and simulated annealing-based acceptance rule [35].

ABC with a new neighborhood approach [37], hybrid ABC-tabu search [41] and improved ABC with problem-related knowledge and knowledge-based neighborhood search [42]. UPMSPR with PM is an extended UPMSP with additional resources and PM. It has some similar characteristics to UPMSP by ABC [34,35,37,41,42]. For example, they have the same sub-problems. The previous works on ABC for UPMSP and the relations between

UPMSP and UPMSPR with PM that ABC is a potential good optimization algorithm to solve UPMSPR with PM; therefore, ABC is used.

When two bee swarms or multiple bee swarms are used, adaptive competition as an effective way is implemented among bee swarms in previous ABC [43,44]. Chu et al. [44] presented an adaptive competition by evaluation of swarm and solutions of migration from the inferior swarm to superior one. Wang et al. [43] proposed an adaptive competition in the onlooker bee phase, in which two employed bee swarms are given a selection probability. Adaptive competitions can effectively use search advantages of the winning bee swarms and intensify the search efficiency of ABC; however, adaptive competition is seldom investigated in ABC, and the corresponding implementations are also limited.

In this study, UPMSPR with PM and makespan minimization is considered. An effective way is provided to implement adaptive competition, and a novel artificial bee colony with adaptive competition (ABC-AC) is proposed. Two employed bee swarms are constructed and compared according to evolution quality. In the employed bee phase, adaptive competition is fulfilled to dynamically select the following two cases. The first is the shifting of search resources from the employed bee swarm with lower evolution quality to another one, and the second is the migration of solutions from the employed bee swarm with higher evolution quality to another one. An adaptive onlooker bee phase is implemented, and a new scout phase is given. A number of experiments are conducted. The computational results demonstrate that new strategies of ABC-AC are effective and that ABC-AC is a competitive algorithm for solving the considered UPMSPR.

The rest of the paper is arranged as follows. Section 2 describes the considered UPMSPR with PM. ABC is introduced in Section 3. Section 4 shows the detailed steps of ABC-AC for UPMSPR with PM. Section 5 presents the computational results and analyses. The conclusions and future topics are reported in the final section.

2. Problem Description

UPMSPR with PM is described as follows. There are n jobs $J_1, J_2, \cdots, J_n$ and m unrelated parallel machines $M_1, M_2, \cdots, M_m$. Each job can be processed on any one of m machines. p_{ki} indicates processing time of job J_i on machine M_k. An additional renewable resource is considered. Job J_i processed on M_k needs r_{ki} units of the additional resource. At most, R_{max} units of the additional resource can be used at any time.

To keep the manufacturing system at the desired level of operation, PM is considered. A time interval exists between two consecutive PM and jobs are processed in the interval. u_k is the length of the interval on M_k, w_k indicates the duration of PM on M_k and the beginning time of the g-th PM is $g \times u_k$.

There have following constraints on jobs and machines.

All jobs and machines are available at time zero.

Each job can be processed on only one machine at a time.

Each machine handles at most one job at a time.

Preemption is not allowed.

The problem is composed of the scheduling sub-problem and machine assignment sub-problem. The goal of the problem is to minimize the makespan.

$$\min C_{max} = \max\{C_j | j = 1, 2, \cdots, n\} \tag{1}$$

where C_{max} indicates the maximum completion time of all jobs.

For UPMSP with the objective of makespan, on each machine, there exists a job-related symmetry, that is, two adjacent jobs are exchanged, and the objective is not changed. When additional resources are considered, there is a certain amount of destruction on the above symmetry; however, the symmetry still exists.

An example with eight jobs and two machines is given with $R_{max} = 10$. Its schedule is shown in Figure 1, in which numbers in each box are job and how many units of the addi-

tional resource are used. For example, 8(6) on machine M_2, 8 means job J_8, and 6 indicates r_{28} of 6. $p_{ki} = \begin{pmatrix} 5 & 3 & 5 & 4 & 4 & 4 & 2 & 5 \\ 4 & 2 & 6 & 7 & 8 & 7 & 5 & 3 \end{pmatrix}$; $r_{ki} = \begin{pmatrix} 5 & 3 & 5 & 2 & 5 & 5 & 4 & 1 \\ 6 & 3 & 7 & 6 & 2 & 7 & 2 & 6 \end{pmatrix}$.

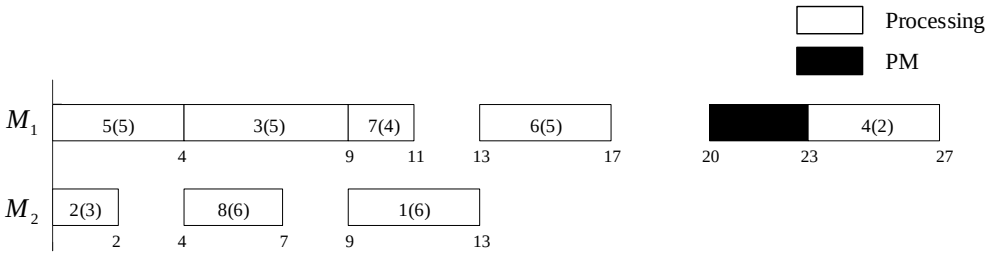

Figure 1. A schedule of the example.

3. Introduction to ABC

In ABC, there are three types of artificial bees. The first is employed bee, who searches for the food source. The second is the onlooker bee, who is in the hive to choose a food source. The third is the scout, who does random searches for a new food source. A solution of the problem is depicted as the position of a food source, the nectar amount of which is the fitness of the solution.

In the search process, the initial population P with N solutions is first produced, and then three phases—bee phase, onlooker bee phase and scout phase—are performed repeatedly before the stopping condition is met.

In the employed bee phase, a new solution y_i is produced for each $x_i \in P$.

$$y_i = x_i + \phi(x_i - x_k) \tag{2}$$

where $\phi \in [-1, 1]$ is a real random number, and $x_k \in P$ is a randomly selected solution, $i \neq k$.

Greedy selection is applied between x_i and y_i: if $fit(y_i) > fit(x_i)$, then y_i substitutes for x_i, where $fit(x_i)$ denotes the fitness of x_i.

In the onlooker bee phase, each onlooker bee chooses a food source by roulette selection based on the probability defined by

$$prob_i = fit(x_i) \Big/ \sum_{l=1}^{N} fit(x_l) \tag{3}$$

where $prob_i$ indicates the probability of solution x_i.

Once an onlooker bee selects a food solution x_i, a new solution y_i is obtained by Equation (2) and the above greedy selection is applied to decide if x_i can be replaced with y_i.

In the above two phases, a $trial_i$ is computed for each x_i. Initially, $trial_i = 0$ for all solutions in P. If the newly obtained y_i cannot update x_i, then $trial_i = trial_i + 1$; otherwise, $trial_i = 0$.

In scout phase, if $trial_i$ of a food source exceeds a threshold $Limit$, the corresponding employed bee will turn into a scout, which randomly produces a food source to substitute for the old one.

4. ABC-AC for UPMSPR with PM

Competition is often performed among populations or swarms in the following way. After populations or swarms are evaluated, the winning population or swarm are determined, and then solutions of other population or swarm are migrated to the winning one. In this study, a new way is applied to execute adaptive competition, in which solution migration or search resource shifting between two employed bee swarms are dynamically

decided according to competition results, adaptive onlooker bee phase and a scout phase are also newly implemented. The detailed descriptions are shown below.

4.1. Solution Representation

In this study, a new solution representation is presented. For UPMSPR with n jobs, m machines, R_{max} units of the additional resource and PM, its solution is represented as a machine assignment string $[M_{h_1}, M_{h_2}, \cdots, M_{h_n}]$ and a scheduling string $[\theta_1, \theta_2, \cdots, \theta_n]$, where M_{h_i} is the assigned machine for job J_i and θ_i is real number.

The decoding procedure is shown below.

(1) All assigned jobs on each machine $M_k, k = 1, 2, \cdots, M_m$ are decided in terms of machine assignment string;

(2) For each machine $M_k, k = 1, 2, \cdots, m$, (1) a permutation of all jobs on M_k is gotten by sorting these jobs in the ascending order of θ_i, (2) for the permutation, start with first job, for each job J_i, first decide each idle period of M_k, if J_i can be inserted into some idle periods when processing time and resource constraint are met, then choose an idle period with the smallest beginning time and insert J_i into the chosen period in terms of processing time and resource constraint; otherwise, J_i is processed after the current last processed job of M_k; if the completion time of J_i exceeds $g \times u_k$, then PM is first done, and then J_i is processed.

All constraints of UPMSPR with PM are directly handled in the decoding procedure, and the obtained schedule is always feasible. A solution of the example in Section 2 is $[2, 2, 1, 1, 1, 1, 1, 2]$ and $[0.22, 0.72, 0.11, 0.84, 0.03, 0.35, 0.52, 0.17]$, and the corresponding schedule is depicted in Figure 1. As shown in Figure 1, J_5 is assigned on M_1, $s_5 = 0$ and $C_5 = 4$; moreover, all constraints are met, and thus a feasible schedule is obtained.

A solution with m job sequences [3] and a representation method with three strings [11] are used to denote solutions of UPMSPR; however, strings or job sequences in these methods are dependent each other. In this study, machine assignment string and scheduling string are independent, and additional resources are effectively handled in the decoding procedure.

The initial population with N solutions is produced as follows. A heuristic is presented for an initial solution. Each job J_i is first assigned on a machine M_k with the smallest p_{ik} and allocated on a M_k with the smallest r_{ki} when $p_{i1} = p_{i2} = \cdots = p_{im}$, then scheduling string is randomly produced. The remaining $N - 1$ initial solutions are randomly generated.

After initial population P is produced, all solutions in P are sorted in the ascending order of C_{max}, suppose that $C_{max}^{x_1} \leq C_{max}^{x_2} \leq \cdots C_{max}^{x_N}$; then, x_1 is added into EB_1, x_2 is included into EB_2, x_3 is assigned into EB_1, x_4 becomes a member of EB_2 and so on; finally, two employed bee swarms EB_1, EB_2 are obtained, where $C_{max}^{x_i}$ indicates makespan of solution x_i.

4.2. Employed Bee Phase with Adaptive Competition

Adaptive competition is performed between EB_1 and EB_2 based on their evolution quality, which is defined below.

$$Evq_{EB_j}^{gen} = \sum_{x_i \in EB_j} \lambda_i^{gen} / N \tag{4}$$

where $Evq_{EB_j}^{gen}$ is the evolution quality of EB_j on generation gen, if x_i is replaced with z in the employed bee phase on generation gen, λ_i^{gen} is 1; otherwise 0.

When EB_1 and EB_2 are compared, if $Evq_{EB_1}^{gen} > Evq_{EB_2}^{gen}$, then EB_1 obtains extra R searches from EB_2, that is, EB_1 is given $N/2 + R$ searches and EB_2 has $N/2 - R$ searches. One search means that, for a solution x_i, global search is first done and then a multiple neighborhood search of x_i is executed.

For solution $x_i \in EB_j$, global search is shown as follows. Randomly choose a solution $y \in EB_j$, execute two-point crossover between x_i, y on machine assignment string, if the

obtained solution z is better than x_i, then update Θ with x_i and replace x_i with z; else perform two-point crossover between x_i, y on scheduling string, if the produced solution z is better than x_i, then update Θ with x_i and replace x_i with z.

Θ denotes a set of historical optimization data and is updated as follows. If $|\Theta| < |\Theta|_{max}$, then solution z is directly included into Θ; otherwise, if z is better than the worst member of Θ, then the worst member is replaced with z, where $|\Theta|_{max}$ indicates maximum size of Θ. We set $|\Theta|_{max}$ to be 50 by experiments.

Neighborhood structures $\mathcal{N}_1 - \mathcal{N}_5$ are used. $\mathcal{N}_1$ is depicted as follows. A randomly chosen job from a machine with the longest completion time is moved to a machine with smallest completion time. $\mathcal{N}_2$ generates new solutions by deciding a randomly selected job on a machine M_k with the longest completion time and a randomly chosen job on machine $M_l, l \neq k$ and swapping them. $\mathcal{N}_3$ is shown below. Randomly decide two machines $M_k, M_l, l \neq k$ and swap a randomly selected job J_i on M_k and a randomly chosen job J_j on M_l. In $\mathcal{N}_1, \mathcal{N}_2, \mathcal{N}_3$, only machine assignment string is changed.

$\mathcal{N}_4$ is applied to obtain new solutions by randomly deciding a machine M_k and two jobs on M_k and exchanging them. When $\mathcal{N}_5$ is done, a M_k, J_i, J_j on M_k are randomly determined, ten θ_i is inserted on the position $j-1$ of scheduling string, if $j = 1$, θ_i is inserted on position j. Multiple neighborhood search of x_i is shown below. Let $g = 1$, repeat the following steps until $g = 6$: produce a solution $z \in \mathcal{N}_g(x_i)$, if $C^z_{max} < C^{x_i}_{max}$, then x_i is replaced with z and $g = 6$; otherwise, $g = g + 1$.

com_j is defined. Initial $com_j = 0$, $j = 1, 2$. If EB_j obtains extra seach times from EB_{3-j}, then $com_j = com_j + 1$ and $com_{3-j} = 0$. To avoid excessive competition, solution migration is executed if one of com_1 and com_2 exceeds or is equal to Q. If $com_j \geq Q$, then R solutions with smallest makespan are chosen from EB_j and substitutes for the worst R solutions of EB_{3-j}, and reduced variable neighborhood search (RVNS) acts on each of R newly added solutions of EB_{3-j}, where Q is integer.

It can be found that when $com_j = Q$, com_{3-j} must be 0, and thus only one of com_1 and com_2 exceeds or is equal to Q. RVNS is performed for solution x: let $w = 1$, $g = 1$, repeat the following steps until $w > T$: produce a new solution $z \in \mathcal{N}_g(x)$, if z is better than x, then update Θ with y and replace y with z, and $g = 1$; otherwise, $g = g + 1$, let $g = 1$ if $g = 6$, where T is integer.

Employed bee phase is composed of two cases. If $com_1 < Q$ and $com_2 < Q$, then the first case is executed; otherwise, the second case is performed. The first case is executed as follows.

(1) Compute $Evq^{gen}_{EB_1}$ and $Evq^{gen}_{EB_2}$.
(2) If $Evq^{gen}_{EB_1} = Evq^{gen}_{EB_2}$, then $com_1 = com_2 = 0$, execute sequentially one search for each solution in EB_1 and EB_2.
(3) If $Evq^{gen}_{EB_1} > Evq^{gen}_{EB_2}$, then $com_1 = com_1 + 1$, $com_2 = 0$, sort all solutions of EB_1 and EB_2, respectively, in the ascending order of makespan, let W be the set of R solutions with smallest makespan from EB_1, execute one search for each solution in EB_1 and one search for each $x \in W$, sequentially, perform one search for each of the first $N/2 - R$ solutions of EB_2.
(4) If $Evq^{gen}_{EB_2} > Evq^{gen}_{EB_1}$, then $com_2 = com_2 + 1$, $com_1 = 0$, EB_2 obtains extra R searches and EB_1 just has $N/2 - R$ searches as done in (3).

The second case is described below. If $com_j \geq Q$, R solutions with best makespan are chosen from EB_j, for each chosen solution $x \in EB_j$, if it is better than the worst solution of EB_{3-j}, then the worst solution y of EB_{3-j} is replaced with x and RVNS acts on y.

When the second case is executed, solution migration is applied, RVNS only acts on the transferred solutions from EB_j with $com_j \geq Q$ and searches of the first case are not done, as a result, evolution quality of EB_{3-J} can be improved and EB_{3-j} can win in the next competition with EB_j.

EB_1 and EB_2 compete according to evolution quality, excessive competition is considered and the worse employed bee swarm is improved, as a result, EB_1 and EB_2 can compete extensively.

4.3. Adaptive Onlooker Bee Phase and New Scout Phase

Adaptive onlooker bee phase is shown as follows.

(1) Compute $\bar{C}^1_{max} = \sum_{x_i \in EB_1} 2 \times C^{x_i}_{max}/N$ and $\bar{C}^2_{max} = \sum_{x_i \in EB_2} 2 \times C^{x_i}_{max}/N$
(2) If random number $rand < \bar{C}^1_{max}/(\bar{C}^2_{max}+\bar{C}^1_{max})$, then EB_1 is chosen; otherwise, EB_2 is selected
(3) For each onlooker bee $l = 1, 2, \cdots, N$, select a x from the chosen empoyed bee swarm by roulette selection in Section 3, execute one search for the solution x.

where $rand$ follows uniform distribution on [0, 1], $fit(x_i)$ is equal to $1/C^{x_i}_{max}$.

In the onlooker bee phase, a employed bee swarm is selected adaptively. If $\bar{C}^1_{max} < \bar{C}^2_{max}$, then the probability of EB_1 is less than that of EB_2 and EB_2 has higher possibility than EB_1 in step (2), that is, EB_2 with lower solution quality is given higher selection possiblity, as a result, EB_2 may possess more searches, its solutions can be improved, and EB_2 can win in the next competition.

A new scout phase is described below.

(1) Sort all solutions of P in the ascending order of $C^{x_i}_{max}$
(2) For each solution $x_i \in P$ with $trial_i \geq Limit$,

If $i \leq \gamma \times N$, then for each solution $y \in \Theta$, compute its probability pr_y; then, select a solution $x \in \Omega$ by roulette selection based on pr; produce $y_g \in \mathcal{N}_g(x)$, $g = 1, 2, 3, 4, 5$ sequentially, the best y_g directly substitutes for x_i.

Otherwise, a set Φ with solutions $x_1, x_2, \cdots, x_{\gamma \times N}$ is constructed, and a solution $x \in Phi$ is selected using the same way in the first case; we generate y_1, y_2, y_3, y_4, y_5, and the best them becomes new x_i.

$$pr_y = \left(C^y_{max}\right)^{-1} \Big/ \sum_{x \in \Theta} \left(C^x_{max}\right)^{-1} \tag{5}$$

4.4. Algorithm Description

The detailed steps of ABC-AC are shown as follows.

(1) Randomly produce initial population P with N solutions and divide the whole population into EB_1 and EB_2, $gen = 1$.
(2) Execute employed bee phase with adaptive competition.
(3) Perform onlooker bee phase.
(4) Execute scout phase.
(5) $gen = gen + 1$. If the stopping condition is not met, go to step (2); otherwise, stop the search.

Unlike the previous ABCs [31–43,45,46], ABC-AC has the following features. (1) Two employed bee swarms EB_1, EB_2 are constructed, and the employed bee phase consists of two cases, searches shifting and solution migration between EB_1 and EB_2. An adaptive competition is performed between EB_1 and EB_2 to dynamically select one of two cases on each generation. (2) In the onlooker bee phase, one employed bee swarm is first chosen adaptively, and then all onlooker bees select food sources from the selected employed bee swarm.

A new scout phase is implemented based on the solution quality. Competitive and adaptive onlooker bee phase that lead to EB_j have more chance to improve performance when $com_j \geq Q$ and $\bar{C}^j_{max} > \bar{C}^{3-j}_{max}$, as a result, EB_j can win extra R search in the next employed bee phase, EB_1 and EB_2 can compete well and the possibility of falling local optimal can reduce notably; therefore, the search efficiency can be improved.

5. Computational Experiments

Many experiments are conducted to test the performance of ABC-AC for UPMSPR with PM. All experiments were implemented using Microsoft Visual C++ 2019 and run on 8.0G RAM 2.30 GHz CPU PC.

5.1. Test Instances and Comparative Algorithms

A total of 300 instances were used [4], which can be obtained directly from http://soa.iti.es (accessed on 19 May 2022). $R_{max} = 5m$, w_k is an integer selected from the same interval as p_{ki}, $u_k = round(w_k + 3.5 \times \max_{i=1,2,\cdots,n}\{p_{ki}\})$. $round(x)$ denotes an integer being closet to x. Five ways were used to produce processing time, and two ways were applied to generate additional resource; thus, 10 combinations of processing time and additional resources were used. No is defined as a combination of the a-th way of processing time and the b-th way of additional resource. $b = 1$ for $No \leq 5$ and $b = 2$ for $No > 5$, and thus the instance is depicted as $n \times m \times No$.

TAFOA [3] and a multi-pass heuristic (MPH) [7] are chosen as comparative algorithm because they can be directly used to solve UPMSPR with PM. Salehi Mir and Rezaeian [47] presented a hybrid particle swarm optimization and genetic algorithm (HPSOGA), which can be directly applied to our UPMSPR after the decoding procedure of ABC-AC is adopted; therefore, we selected it as a comparative algorithm.

ABC is constructed to show the effect of new strategies of ABC-AC, which are adaptive competition, adaptive onlooker bee phase and new scout phase. In ABC, only one employed bee swarm and no adaptive competition is used in the employed bee phase, each onlooker bee selects a food source according to probability $prob_i$ and a randomly chosen neighborhood structure is performed on the selected food source. The scout phase of Section 3 is directly adopted.

5.2. Parameter Settings

ABC-AC has the following parameters: N, R, T, Q, γ, $Limit$ and stopping condition. We found that ABC-AC can converge well with $0.3n$ seconds of CPU time, and $0.3n$ seconds CPU time also can be used as a stopping condition of comparative algorithms. Thus, $0.3n$ seconds of CPU time is used as the stopping condition.

Then, we apply the Taguchi method [48] to decide the settings for other parameters. We select instance $50 \times 10 \times 1$. Table 1 gives the levels of each parameter. The orthogonal array $L_{27}(3^6)$ is tested. ABC-AC with each combination run 10 times independently for the chosen instance.

Table 1. Parameters and their levels.

Parameters	Factor Level		
	1	**2**	**3**
R	8	10	12
Q	3	4	5
T	8	10	12
γ	0.2	0.3	0.4
N	90	100	110
$Limit$	8	10	12

Figure 2 shows the results of MIN and S/N ratio, in which the S/N ratio is defined as $-10 \times \log_{10}(MIN^2)$ and MIN is the best solution found in 10 runs. As shown in Figure 2, it can be found that ABC-AC with following combination $N = 100$, $Q = 4$, $R = 10$, $T = 10$, $\gamma = 0.3$ and $Limit = 10$ can obtain better results than ABC-AC with other combinations; thus, the above combination is adopted.

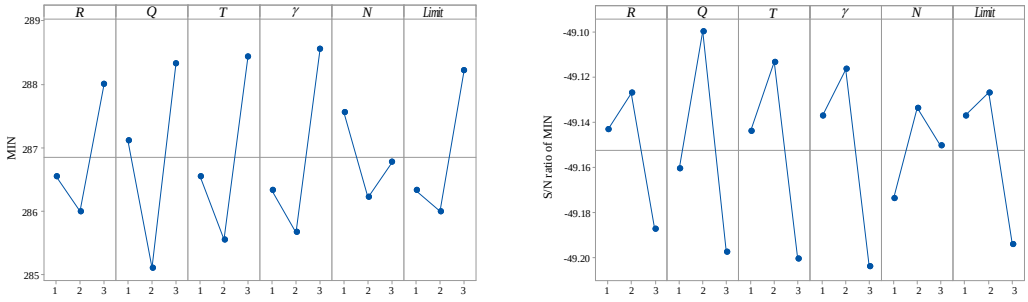

Figure 2. The mean MIN and the mean S/N ratio of MIN.

ABC has following parameters: $N = 100$ and $Limit = 8$.

Parameter settings of three comparative algorithms are directly selected from references [3,7,47] except that the stopping condition. We also test these settings for each comparative algorithm by Taguchi method. The experimental results show that those settings of each comparative algorithm are still effective and comparative algorithms with those settings can produce better results than MPH, HPSOGA and TAFOA with other settings.

5.3. Results and Discussion

ABC-AC is compared with ABC and three comparative algorithms. Each of five algorithms randomly runs 10 times on each instance. The corresponding results of all algorithms are shown in Tables 2–7 and A1–A3, where AVG is the average value of 10 elite solutions obtained in 10 runs, and MAX is the worst of 10 elite solutions in 10 runs. Tables A1–A3 are listed in Appendix A. Figure 3 presents a mean plot with a 95% confidence interval.

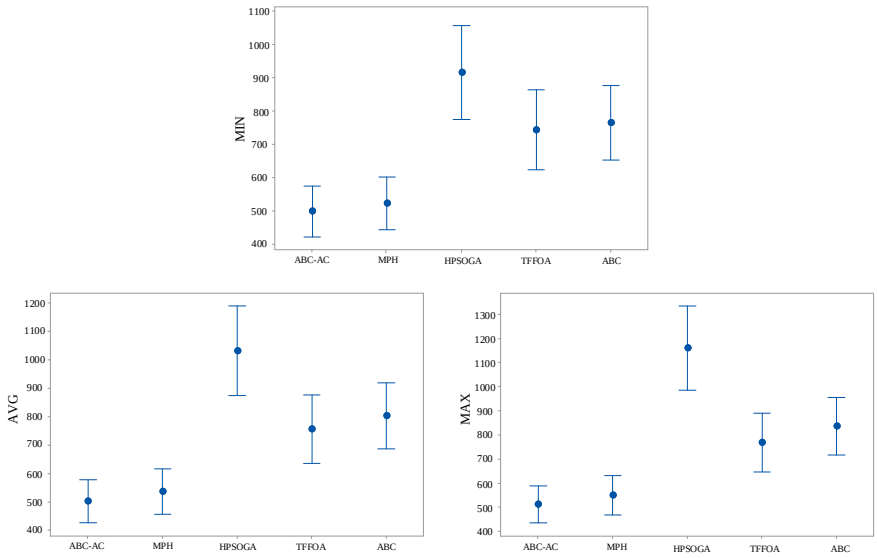

Figure 3. Mean plot with 95% confidence interval.

As shown in Tables 2–4, ABC-AC produces better MIN than or identical MIN with ABC on all instances; moreover, MIN of ABC is worse than that of ABC-AC by at least 20 on more than 210 instances. ABC-AC converges better than ABC. This conclusion also can be obtained from Figure 3. It also can be found from Tables 5–7 and A1–A3 and Figure 3 that ABC-AC performs significantly than ABC on AVG and MAX. ABC-AC produces smaller AVG and MAX than ABC on all instances and AVG and MAX of ABC-AC are notably less than those of ABC on all instances with $n \geq 30$. ABC-AC significantly outperforms ABC on convergence, average result and stability; thus, it can be concluded that the usage

of new strategies, such as adaptive competition has a positive impact on the performance of ABC-AC.

Table 2. The results of five algorithms on $MIN(1)$.

Instance	ABC-AC	MPH	HPSOGA	TAFOA	ABC	Instance	ABC-AC	MPH	HPSOGA	TAFOA	ABC
8 × 2 × 1	326.0	326.0	337.3	326.0	326.0	12 × 6 × 1	145.0	155.5	176.8	163.3	156.2
8 × 2 × 2	269.0	269.0	270.6	311.6	269.0	12 × 6 × 2	146.5	156.0	167.2	162.0	155.4
8 × 2 × 3	197.0	204.6	199.2	197.0	197.2	12 × 6 × 3	62.0	62.0	71.5	65.3	72.0
8 × 2 × 4	241.0	248.7	265.3	256.1	245.7	12 × 6 × 4	63.0	64.8	74.3	75.1	67.6
8 × 2 × 5	194.0	194.0	208.0	194.0	194.0	12 × 6 × 5	49.0	49.5	78.8	82.0	59.4
8 × 2 × 6	139.0	139.0	141.5	139.0	139.0	12 × 6 × 6	57.0	57.7	81.4	64.0	62.4
8 × 2 × 7	204.0	204.4	233.4	224.0	204.0	12 × 6 × 7	57.0	63.4	101.4	111.0	73.1
8 × 2 × 8	171.0	171.0	179.7	171.0	171.0	12 × 6 × 8	63.3	70.6	90.4	77.0	78.4
8 × 2 × 9	544.0	544.0	550.3	544.0	544.0	12 × 6 × 9	250.0	286.7	357.8	480.0	273.4
8 × 2 × 10	577.0	577.0	596.8	631.2	590.8	12 × 6 × 10	263.0	297.0	372.4	318.3	281.6
8 × 4 × 1	124.0	131.0	136.1	137.0	127.1	16 × 2 × 1	565.4	645.0	640.8	653.4	609.5
8 × 4 × 2	119.0	120.2	131.4	126.0	122.1	16 × 2 × 2	600.0	655.0	675.6	688.4	635.7
8 × 4 × 3	174.0	181.7	175.6	184.7	174.3	16 × 2 × 3	460.0	529.9	572.9	528.0	534.5
8 × 4 × 4	123.0	124.0	139.3	157.4	124.4	16 × 2 × 4	642.0	803.6	876.6	791.3	768.9
8 × 4 × 5	80.0	89.7	84.1	95.9	80.0	16 × 2 × 5	332.0	348.0	411.8	341.0	343.8
8 × 4 × 6	66.0	76.6	85.2	78.0	72.4	16 × 2 × 6	264.0	284.0	340.8	284.6	284.6
8 × 4 × 7	91.0	97.6	103.5	97.3	91.6	16 × 2 × 7	363.0	391.9	436.1	379.5	385.3
8 × 4 × 8	78.0	88.7	95.7	89.0	87.0	16 × 2 × 8	358.0	382.0	444.2	385.7	394.6
8 × 4 × 9	279.0	306.1	304.3	312.9	281.5	16 × 2 × 9	1165.0	1392.0	1459.5	1314.1	1345.9
8 × 4 × 10	275.0	305.1	348.6	342.0	286.8	16 × 2 × 10	1219.0	1409.8	1643.9	1680.5	1403.5
8 × 6 × 1	103.0	105.0	105.5	107.5	103.0	16 × 4 × 1	280.6	311.0	409.9	340.0	336.2
8 × 6 × 2	106.0	106.0	108.8	111.2	106.0	16 × 4 × 2	213.0	225.3	247.4	237.9	232.8
8 × 6 × 3	46.0	46.0	50.9	48.4	75.8	16 × 4 × 3	181.0	189.2	212.9	352.5	196.0
8 × 6 × 4	71.0	73.3	80.7	77.8	73.3	16 × 4 × 4	140.0	153.9	159.4	150.2	158.2
8 × 6 × 5	36.0	36.0	51.9	56.0	37.1	16 × 4 × 5	80.0	83.7	127.0	114.0	104.7
8 × 6 × 6	58.0	58.0	58.9	58.0	58.0	16 × 4 × 6	90.0	101.8	137.0	102.0	129.8
8 × 6 × 7	42.0	42.0	59.3	68.0	44.8	16 × 4 × 7	112.0	113.0	144.3	158.0	132.6
8 × 6 × 8	62.0	62.0	64.0	62.0	62.0	16 × 4 × 8	121.0	128.0	181.4	121.0	159.3
8 × 6 × 9	214.0	219.0	227.8	254.0	214.5	16 × 4 × 9	488.0	504.7	560.0	711.0	535.0
8 × 6 × 10	218.0	225.3	242.1	225.5	222.2	16 × 4 × 10	508.0	551.5	602.6	543.3	569.2
12 × 2 × 1	326.0	326.0	345.1	326.0	326.2	16 × 6 × 1	143.8	153.2	183.0	165.2	163.1
12 × 2 × 2	514.6	639.9	666.6	633.6	595.5	16 × 6 × 2	150.9	159.0	178.6	219.0	165.3
12 × 2 × 3	355.3	400.7	429.5	393.4	385.1	16 × 6 × 3	123.0	126.0	153.5	143.5	137.4
12 × 2 × 4	292.0	332.0	347.4	319.8	317.4	16 × 6 × 4	115.0	124.0	137.6	277.7	122.3
12 × 2 × 5	171.0	173.4	178.0	172.0	173.1	16 × 6 × 5	67.0	74.5	118.5	75.7	95.8
12 × 2 × 6	261.0	275.2	278.7	261.0	266.8	16 × 6 × 6	62.0	69.7	110.4	75.0	94.8
12 × 2 × 7	205.0	205.0	206.3	205.0	205.2	16 × 6 × 7	91.4	91.6	135.3	110.0	117.8
12 × 2 × 8	296.0	317.7	329.3	300.0	307.6	16 × 6 × 8	85.0	92.5	129.4	95.0	112.6
12 × 2 × 9	774.0	774.0	1024.6	774.0	794.3	16 × 6 × 9	356.0	379.4	426.6	368.5	387.2
12 × 2 × 10	990.9	1076.0	1088.1	1129.0	1053.0	16 × 6 × 10	360.0	381.5	454.3	380.1	398.2
12 × 4 × 1	231.0	241.4	269.9	282.4	257.6	20 × 2 × 1	561.9	699.0	660.2	613.1	638.8
12 × 4 × 2	199.2	213.8	216.0	242.0	213.4	20 × 2 × 2	621.9	644.7	702.6	700.1	658.6
12 × 4 × 3	214.0	245.5	280.8	281.7	255.5	20 × 2 × 3	724.8	855.4	1099.1	940.7	970.7
12 × 4 × 4	229.0	242.2	249.7	260.0	234.6	20 × 2 × 4	592.4	708.3	809.0	731.7	722.1
12 × 4 × 5	93.0	94.4	130.9	94.0	102.3	20 × 2 × 5	409.9	482.0	524.3	435.4	484.3
12 × 4 × 6	70.0	75.5	105.0	90.9	99.9	20 × 2 × 6	338.0	422.0	456.7	424.0	398.8
12 × 4 × 7	110.8	111.0	134.2	113.3	130.4	20 × 2 × 7	502.2	568.0	603.5	516.6	550.5
12 × 4 × 8	98.0	105.2	128.9	110.0	118.1	20 × 2 × 8	397.0	549.0	582.9	533.0	550.5
12 × 4 × 9	403.0	420.9	472.5	437.1	429.0	20 × 2 × 9	1468.0	1735.0	1975.9	1628.5	1731.7
12 × 4 × 10	388.0	435.0	473.0	434.1	428.7	20 × 2 × 10	1507.0	1600.8	1959.1	1676.8	1841.6

Table 3. The results of five algorithms on $MIN(2)$.

Instance	ABC-AC	MPH	HPSOGA	TAFOA	ABC	Instance	ABC-AC	MPH	HPSOGA	TAFOA	ABC
20 × 4 × 1	353	357	442	424	409	30 × 2 × 1	1292	1624	1518	1617	1570
20 × 4 × 2	295	305	315	336	311	30 × 2 × 2	915	934	967	908	973
20 × 4 × 3	164	168	166	168	169	30 × 2 × 3	358	374	372	373	371
20 × 4 × 4	170	169	170	215	177	30 × 2 × 4	992	1133	1088	1296	1120
20 × 4 × 5	113	115	146	117	128	30 × 2 × 5	670	789	779	799	802
20 × 4 × 6	111	111	161	143	129	30 × 2 × 6	697	790	834	924	801
20 × 4 × 7	153	156	195	161	181	30 × 2 × 7	754	884	925	914	904
20 × 4 × 8	149	151	183	184	183	30 × 2 × 8	759	871	899	835	895
20 × 4 × 9	628	648	677	640	658	30 × 2 × 9	2303	2838	2942	2789	2877
20 × 4 × 10	629	715	681	645	683	30 × 2 × 10	2515	2796	2934	2787	2934
20 × 6 × 1	210	217	243	248	240	30 × 4 × 1	405	429	573	524	557
20 × 6 × 2	199	212	231	298	215	30 × 4 × 2	378	388	429	427	425
20 × 6 × 3	180	180	190	202	187	30 × 4 × 3	356	368	405	382	394
20 × 6 × 4	58	64	71	64	73	30 × 4 × 4	381	395	415	458	415
20 × 6 × 5	79	78	141	88	110	30 × 4 × 5	156	156	302	190	230
20 × 6 × 6	57	57	121	61	101	30 × 4 × 6	130	132	226	135	219
20 × 6 × 7	107	114	151	114	134	30 × 4 × 7	218	214	320	254	281
20 × 6 × 8	84	84	134	91	106	30 × 4 × 8	179	187	331	192	269
20 × 6 × 9	445	463	536	468	474	30 × 4 × 9	1140	1200	1359	1189	1262
20 × 6 × 10	435	450	524	449	478	30 × 4 × 10	1080	1094	1261	1117	1188
25 × 2 × 1	770	910	880	855	869	30 × 6 × 1	311	319	387	406	381
25 × 2 × 2	772	788	804	773	797	30 × 6 × 2	268	275	333	285	304
25 × 2 × 3	1499	1707	1866	1929	1810	30 × 6 × 3	224	232	279	528	243
25 × 2 × 4	281	296	315	281	311	30 × 6 × 4	266	286	349	816	292
25 × 2 × 5	632	690	803	674	723	30 × 6 × 5	75	75	161	105	130
25 × 2 × 6	473	505	547	495	541	30 × 6 × 6	82	89	204	129	165
25 × 2 × 7	711	917	793	750	811	30 × 6 × 7	114	114	214	142	165
25 × 2 × 8	559	582	620	565	612	30 × 6 × 8	124	130	233	172	210
25 × 2 × 9	2150	2395	2336	2157	2417	30 × 6 × 9	581	591	727	603	668
25 × 2 × 10	1973	2107	2120	2090	2100	30 × 6 × 10	599	633	785	723	723
25 × 4 × 1	337	344	464	489	423	50 × 10 × 1	281	281	382	371	365
25 × 4 × 2	413	436	461	579	452	50 × 10 × 2	275	288	354	394	321
25 × 4 × 3	309	315	375	348	357	50 × 10 × 3	289	290	440	905	369
25 × 4 × 4	159	167	184	174	184	50 × 10 × 4	262	253	349	421	313
25 × 4 × 5	132	132	236	141	199	50 × 10 × 5	64	65	291	120	219
25 × 4 × 6	160	178	276	171	230	50 × 10 × 6	76	90	253	131	221
25 × 4 × 7	175	175	242	190	224	50 × 10 × 7	107	102	334	173	253
25 × 4 × 8	199	234	305	224	303	50 × 10 × 8	125	146	321	174	249
25 × 4 × 9	758	763	1101	759	1046	50 × 10 × 9	573	604	1056	1008	782
25 × 4 × 10	815	884	1187	861	1180	50 × 10 × 10	604	650	1196	682	750
25 × 6 × 1	231	225	272	302	252	50 × 20 × 1	171	190	281	399	239
25 × 6 × 2	210	220	240	256	225	50 × 20 × 2	148	155	248	278	183
25 × 6 × 3	154	161	179	356	169	50 × 20 × 3	116	106	206	788	141
25 × 6 × 4	91	99	131	132	116	50 × 20 × 4	93	92	205	344	138
25 × 6 × 5	76	76	153	92	127	50 × 20 × 5	24	29	183	34	140
25 × 6 × 6	72	76	165	78	139	50 × 20 × 6	24	24	168	38	121
25 × 6 × 7	104	104	185	116	147	50 × 20 × 7	50	49	196	62	155
25 × 6 × 8	107	111	181	117	137	50 × 20 × 8	44	48	174	71	140
25 × 6 × 9	541	542	622	555	558	50 × 20 × 9	320	324	623	421	426
25 × 6 × 10	550	554	646	567	607	50 × 20 × 10	310	319	607	434	422

Table 4. The results of five algorithms on $MIN(3)$.

Instance	ABC-AC	MPH	HPSOGA	TAFOA	ABC	Instance	ABC-AC	MPH	HPSOGA	TAFOA	ABC
50 × 30 × ×1	107	115	189	196	147	250 × 20 × 1	775	730	1377	1313	1049
50 × 30 × 2	109	119	178	199	136	250 × 20 × 2	744	766	1336	1312	1074
50 × 30 × 3	87	65	170	232	101	250 × 20 × 3	814	771	1319	6901	1371
50 × 30 × 4	89	81	149	362	102	250 × 20 × 4	784	865	1798	2719	1445
50 × 30 × 5	16	15	136	16	92	250 × 20 × 5	76	76	1110	96	865
50 × 30 × 6	16	15	143	27	95	250 × 20 × 6	83	88	1145	109	827
50 × 30 × 7	38	35	161	50	108	250 × 20 × 7	194	193	1239	265	991
50 × 30 × 8	32	35	159	51	103	250 × 20 × 8	202	208	1270	227	845
50 × 30 × 9	214	219	474	412	315	250 × 20 × 9	1584	1562	4205	2386	2794
50 × 30 × 10	211	215	445	409	305	250 × 20 × 10	1607	1630	4038	2119	2733
150 × 10 × 1	1011	909	1478	1076	1229	250 × 30 × 1	519	491	1025	1032	768
150 × 10 × 2	872	875	1264	1286	1135	250 × 30 × 2	501	495	1095	877	753
150 × 10 × 3	1068	1087	1589	3740	1222	250 × 30 × 3	331	448	1157	922	879
150 × 10 × 4	753	784	1258	1550	1374	250 × 30 × 4	584	557	1449	3193	1017
150 × 10 × 5	150	160	994	247	923	250 × 30 × 5	45	49	961	67	658
150 × 10 × 6	167	175	925	226	822	250 × 30 × 6	48	58	850	78	620
150 × 10 × 7	279	282	1024	405	1024	250 × 30 × 7	127	128	1017	160	681
150 × 10 × 8	306	320	1086	442	936	250 × 30 × 8	128	138	995	182	657
150 × 10 × 9	2056	2072	3506	2972	3067	250 × 30 × 9	1157	1133	2837	1390	2044
150 × 10 × 10	2075	2155	3293	2996	2855	250 × 30 × 10	1160	1162	2902	1600	2011
150 × 20 × 1	454	424	793	670	678	350 × 10 × 1	2310	2308	3440	2684	2934
150 × 20 × 2	423	419	840	758	648	350 × 10 × 2	2226	2260	3428	2793	2704
150 × 20 × 3	469	424	929	2555	758	350 × 10 × 3	2169	2028	3341	4768	3550
150 × 20 × 4	517	487	970	2609	668	350 × 10 × 4	1712	1628	2496	5547	2630
150 × 20 × 5	58	56	665	83	580	350 × 10 × 5	378	382	2639	513	2377
150 × 20 × 6	53	55	642	71	515	350 × 10 × 6	353	378	2394	389	2174
150 × 20 × 7	125	126	725	175	612	350 × 10 × 7	713	716	2900	872	2487
150 × 20 × 8	125	129	675	147	528	350 × 10 × 8	716	783	3037	803	2267
150 × 20 × 9	851	868	2318	1229	1582	350 × 10 × 9	4809	4836	8305	6081	7293
150 × 20 × 10	854	1070	2123	1194	1511	350 × 10 × 10	5056	5145	9601	5786	7088
150 × 30 × 1	282	286	628	436	428	350 × 20 × 1	1116	1077	2062	1811	1497
150 × 30 × 2	274	276	572	555	352	350 × 20 × 2	1033	1077	1974	1745	1413
150 × 30 × 3	216	233	635	1333	429	350 × 20 × 3	699	696	1504	1678	1479
150 × 30 × 4	205	208	775	1076	569	350 × 20 × 4	1087	1174	2812	4002	2195
150 × 30 × 5	27	31	509	47	337	350 × 20 × 5	107	113	1618	133	1139
150 × 30 × 6	30	31	480	43	305	350 × 20 × 6	106	113	1501	136	1142
150 × 30 × 7	80	85	559	123	358	350 × 20 × 7	268	264	1889	301	1329
150 × 30 × 8	79	90	523	110	354	350 × 20 × 8	268	290	1757	340	1256
150 × 30 × 9	604	607	2011	1128	1212	350 × 20 × 9	2318	2316	5990	3088	3889
150 × 30 × 10	604	620	1506	1194	1178	350 × 20 × 10	2328	2383	5594	3168	3843
250 × 10 × 1	1545	1506	2442	1706	1949	350 × 30 × 1	729	727	1537	1392	1087
250 × 10 × 2	1521	1540	2098	1861	1810	350 × 30 × 2	756	707	1421	1685	966
250 × 10 × 3	1572	1571	2380	3869	2527	350 × 30 × 3	624	656	1555	2169	1464
250 × 10 × 4	1484	1479	2050	4081	1853	350 × 30 × 4	424	510	1558	1046	1282
250 × 10 × 5	259	264	1764	298	1624	350 × 30 × 5	61	60	1291	73	851
250 × 10 × 6	299	323	1813	359	1483	350 × 30 × 6	54	56	1213	76	880
250 × 10 × 7	513	521	2069	580	1761	350 × 30 × 7	172	165	1243	218	948
250 × 10 × 8	559	577	1921	637	1622	350 × 30 × 8	172	180	1303	220	944
250 × 10 × 9	3372	3440	6531	3951	5062	350 × 30 × 9	1480	1472	4560	2143	2833
250 × 10 × 10	3534	3778	5980	3995	5155	350 × 30 × 10	1482	1527	4410	2394	2771

Table 5. The results of five algorithms on $AVG(1)$.

Instance	ABC-AC	MPH	HPSOGA	TAFOA	ABC	Instance	ABC-AC	MPH	HPSOGA	TAFOA	ABC
$8 \times 2 \times 1$	326.0	326.0	337.3	326.0	326.0	$12 \times 6 \times 1$	145.0	155.5	176.8	163.3	156.2
$8 \times 2 \times 2$	269.0	269.0	270.6	311.6	269.0	$12 \times 6 \times 2$	146.5	156.0	167.2	162.0	155.4
$8 \times 2 \times 3$	197.0	204.6	199.2	197.0	197.2	$12 \times 6 \times 3$	62.0	62.0	71.5	65.3	72.0
$8 \times 2 \times 4$	241.0	248.7	265.3	256.1	245.7	$12 \times 6 \times 4$	63.0	64.8	74.3	75.1	67.6
$8 \times 2 \times 5$	194.0	194.0	208.0	194.0	194.0	$12 \times 6 \times 5$	49.0	49.5	78.8	82.0	59.4
$8 \times 2 \times 6$	139.0	139.0	141.5	139.0	139.0	$12 \times 6 \times 6$	57.0	57.7	81.4	64.0	62.4
$8 \times 2 \times 7$	204.0	204.4	233.4	224.0	204.0	$12 \times 6 \times 7$	57.0	63.4	101.4	111.0	73.1
$8 \times 2 \times 8$	171.0	171.0	179.7	171.0	171.0	$12 \times 6 \times 8$	63.3	70.6	90.4	77.0	78.4
$8 \times 2 \times 9$	544.0	544.0	550.3	544.0	544.0	$12 \times 6 \times 9$	250.0	286.7	357.8	480.0	273.4
$8 \times 2 \times 10$	577.0	577.0	596.8	631.2	590.8	$12 \times 6 \times 10$	263.0	297.0	372.4	318.3	281.6
$8 \times 4 \times 1$	124.0	131.0	136.1	137.0	127.1	$16 \times 2 \times 1$	565.4	645.0	640.8	653.4	609.5
$8 \times 4 \times 2$	119.0	120.2	131.4	126.0	122.1	$16 \times 2 \times 2$	600.0	655.0	675.6	688.4	635.7
$8 \times 4 \times 3$	174.0	181.7	175.6	184.7	174.3	$16 \times 2 \times 3$	460.0	529.9	572.9	528.0	534.5
$8 \times 4 \times 4$	123.0	124.0	139.3	157.4	124.4	$16 \times 2 \times 4$	642.0	803.6	876.6	791.3	768.9
$8 \times 4 \times 5$	80.0	89.7	84.1	95.9	80.0	$16 \times 2 \times 5$	332.0	348.0	411.8	341.0	343.8
$8 \times 4 \times 6$	66.0	76.6	85.2	78.0	72.4	$16 \times 2 \times 6$	264.0	284.0	340.8	284.6	284.6
$8 \times 4 \times 7$	91.0	97.6	103.5	97.3	91.6	$16 \times 2 \times 7$	363.0	391.9	436.1	379.5	385.3
$8 \times 4 \times 8$	78.0	88.7	95.7	89.0	87.0	$16 \times 2 \times 8$	358.0	382.0	444.2	385.7	394.6
$8 \times 4 \times 9$	279.0	306.1	304.3	312.9	281.5	$16 \times 2 \times 9$	1165.0	1392.0	1459.5	1314.1	1345.9
$8 \times 4 \times 10$	275.0	305.1	348.6	342.0	286.8	$16 \times 2 \times 10$	1219.0	1409.8	1643.9	1680.5	1403.5
$8 \times 6 \times 1$	103.0	105.0	105.5	107.5	103.0	$16 \times 4 \times 1$	280.6	311.0	409.9	340.0	336.2
$8 \times 6 \times 2$	106.0	106.0	108.8	111.2	106.0	$16 \times 4 \times 2$	213.0	225.3	247.4	237.9	232.8
$8 \times 6 \times 3$	46.0	46.0	50.9	48.4	75.8	$16 \times 4 \times 3$	181.0	189.2	212.9	352.5	196.0
$8 \times 6 \times 4$	71.0	73.3	80.7	77.8	73.3	$16 \times 4 \times 4$	140.0	153.9	159.4	150.2	158.2
$8 \times 6 \times 5$	36.0	36.0	51.9	56.0	37.1	$16 \times 4 \times 5$	80.0	83.7	127.0	114.0	104.7
$8 \times 6 \times 6$	58.0	58.0	58.9	58.0	58.0	$16 \times 4 \times 6$	90.0	101.8	137.0	102.0	129.8
$8 \times 6 \times 7$	42.0	42.0	59.3	68.0	44.8	$16 \times 4 \times 7$	112.0	113.0	144.3	158.0	132.6
$8 \times 6 \times 8$	62.0	62.0	64.0	62.0	62.0	$16 \times 4 \times 8$	121.0	128.0	181.4	121.0	159.3
$8 \times 6 \times 9$	214.0	219.0	227.8	254.0	214.5	$16 \times 4 \times 9$	488.0	504.7	560.0	711.0	535.0
$8 \times 6 \times 10$	218.0	225.3	242.1	225.5	222.2	$16 \times 4 \times 10$	508.0	551.5	602.6	543.3	569.2
$12 \times 2 \times 1$	326.0	326.0	345.1	326.0	326.2	$16 \times 6 \times 1$	143.8	153.2	183.0	165.2	163.1
$12 \times 2 \times 2$	514.6	639.9	666.6	633.6	595.5	$16 \times 6 \times 2$	150.9	159.0	178.6	219.0	165.3
$12 \times 2 \times 3$	355.3	400.7	429.5	393.4	385.1	$16 \times 6 \times 3$	123.0	126.0	153.5	143.5	137.4
$12 \times 2 \times 4$	292.0	332.0	347.4	319.8	317.4	$16 \times 6 \times 4$	115.0	124.0	137.6	277.7	122.3
$12 \times 2 \times 5$	171.0	173.4	178.0	172.0	173.1	$16 \times 6 \times 5$	67.0	74.5	118.5	75.7	95.8
$12 \times 2 \times 6$	261.0	275.2	278.7	261.0	266.8	$16 \times 6 \times 6$	62.0	69.7	110.4	75.0	94.8
$12 \times 2 \times 7$	205.0	205.0	206.3	205.0	205.2	$16 \times 6 \times 7$	91.4	91.6	135.3	110.0	117.8
$12 \times 2 \times 8$	296.0	317.7	329.3	300.0	307.6	$16 \times 6 \times 8$	85.0	92.5	129.4	95.0	112.6
$12 \times 2 \times 9$	774.0	774.0	1024.6	774.0	794.3	$16 \times 6 \times 9$	356.0	379.4	426.6	368.5	387.2
$12 \times 2 \times 10$	990.9	1076.0	1088.1	1129.0	1053.0	$16 \times 6 \times 10$	360.0	381.5	454.3	380.1	398.2
$12 \times 4 \times 1$	231.0	241.4	269.9	282.4	257.6	$20 \times 2 \times 1$	561.9	699.0	660.2	613.1	638.8
$12 \times 4 \times 2$	199.2	213.8	216.0	242.0	213.4	$20 \times 2 \times 2$	621.9	644.7	702.6	700.1	658.6
$12 \times 4 \times 3$	214.0	245.5	280.8	281.7	255.5	$20 \times 2 \times 3$	724.8	855.4	1099.1	940.7	970.7
$12 \times 4 \times 4$	229.0	242.2	249.7	260.0	234.6	$20 \times 2 \times 4$	592.4	708.3	809.0	731.7	722.1
$12 \times 4 \times 5$	93.0	94.4	130.9	94.0	102.3	$20 \times 2 \times 5$	409.9	482.0	524.3	435.4	484.3
$12 \times 4 \times 6$	70.0	75.5	105.0	90.9	99.9	$20 \times 2 \times 6$	338.0	422.0	456.7	424.0	398.8
$12 \times 4 \times 7$	110.8	111.0	134.2	113.3	130.4	$20 \times 2 \times 7$	502.2	568.0	603.5	516.6	550.5
$12 \times 4 \times 8$	98.0	105.2	128.9	110.0	118.1	$20 \times 2 \times 8$	397.0	549.0	582.9	533.0	550.5
$12 \times 4 \times 9$	403.0	420.9	472.5	437.1	429.0	$20 \times 2 \times 9$	1468.0	1735.0	1975.9	1628.5	1731.7
$12 \times 4 \times 10$	388.0	435.0	473.0	434.1	428.7	$20 \times 2 \times 10$	1507.0	1600.8	1959.1	1676.8	1841.6

Table 6. The results of five algorithms on $AVG(2)$.

Instance	ABC-AC	MPH	HPSOGA	TAFOA	ABC	Instance	ABC-AC	MPH	HPSOGA	TAFOA	ABC
20 × 4 × 1	353.8	374.9	554.8	433.2	435.7	30 × 2 × 1	1298.8	1624.0	1659.5	1650.5	1615.1
20 × 4 × 2	296.3	308.7	348.2	346.4	325.2	30 × 2 × 2	915.0	937.1	1067.7	916.8	997.3
20 × 4 × 3	164.0	171.4	176.9	175.1	174.4	30 × 2 × 3	367.3	374.0	391.3	390.3	405.3
20 × 4 × 4	170.0	172.7	177.1	221.9	180.6	30 × 2 × 4	992.0	1153.3	1255.8	1330.4	1156.2
20 × 4 × 5	116.9	124.6	180.5	128.0	158.5	30 × 2 × 5	670.0	789.0	877.9	810.3	847.6
20 × 4 × 6	111.0	117.8	178.9	143.0	153.2	30 × 2 × 6	697.0	811.6	969.4	924.0	887.6
20 × 4 × 7	153.0	159.1	228.3	171.9	200.3	30 × 2 × 7	754.7	884.0	974.8	914.0	957.2
20 × 4 × 8	149.0	158.8	202.7	184.0	200.2	30 × 2 × 8	775.4	913.6	1000.6	852.9	938.0
20 × 4 × 9	628.6	658.8	777.7	655.5	699.5	30 × 2 × 9	2303.0	2862.8	3076.4	2841.0	3012.1
20 × 4 × 10	629.0	726.1	801.2	668.1	704.2	30 × 2 × 10	2523.1	2902.2	3087.3	2888.4	2950.2
20 × 6 × 1	210.0	218.6	268.2	256.6	249.3	30 × 4 × 1	405.0	452.9	631.3	525.9	567.9
20 × 6 × 2	199.4	221.2	248.6	298.0	221.7	30 × 4 × 2	385.6	395.7	516.9	428.6	440.0
20 × 6 × 3	180.0	183.4	219.3	272.5	193.2	30 × 4 × 3	356.1	368.8	428.1	397.1	414.1
20 × 6 × 4	58.1	68.2	82.0	67.6	77.3	30 × 4 × 4	381.1	403.1	468.8	513.4	419.2
20 × 6 × 5	79.0	82.6	153.0	94.5	129.4	30 × 4 × 5	156.0	169.0	360.1	190.0	265.7
20 × 6 × 6	57.4	59.0	132.3	61.5	113.7	30 × 4 × 6	130.0	143.2	266.6	144.7	256.7
20 × 6 × 7	107.0	114.0	178.2	125.6	151.5	30 × 4 × 7	218.3	219.2	454.3	254.0	320.4
20 × 6 × 8	84.0	86.8	154.0	96.6	129.5	30 × 4 × 8	179.0	195.0	364.0	197.4	306.9
20 × 6 × 9	445.0	471.5	601.9	495.7	497.0	30 × 4 × 9	1140.0	1207.9	1446.2	1208.2	1316.2
20 × 6 × 10	435.0	463.6	570.1	473.0	501.9	30 × 4 × 10	1080.0	1125.4	1353.6	1118.8	1247.7
25 × 2 × 1	770.0	924.1	972.1	873.3	885.4	30 × 6 × 1	311.8	325.9	458.6	426.6	394.5
25 × 2 × 2	772.0	795.7	861.4	783.3	809.7	30 × 6 × 2	268.0	282.7	375.1	299.4	310.5
25 × 2 × 3	1504.6	1711.5	2076.6	1999.9	1862.9	30 × 6 × 3	224.4	234.7	316.2	549.4	260.2
25 × 2 × 4	281.0	304.8	337.4	293.8	320.5	30 × 6 × 4	267.2	293.8	393.1	835.9	319.4
25 × 2 × 5	632.0	712.7	853.3	698.0	788.2	30 × 6 × 5	75.0	77.8	194.3	105.0	151.0
25 × 2 × 6	473.2	509.6	591.9	513.4	572.1	30 × 6 × 6	82.0	94.0	237.6	129.0	198.2
25 × 2 × 7	711.2	918.4	954.1	757.9	914.7	30 × 6 × 7	114.0	118.2	238.5	142.0	184.1
25 × 2 × 8	559.0	596.8	655.2	574.3	637.0	30 × 6 × 8	125.8	132.6	273.5	172.0	233.6
25 × 2 × 9	2150.0	2395.0	2591.4	2211.4	2493.2	30 × 6 × 9	581.0	610.0	814.7	610.7	682.3
25 × 2 × 10	1988.0	2203.7	2463.1	2121.4	2173.2	30 × 6 × 10	605.3	637.5	1059.2	723.0	767.8
25 × 4 × 1	337.4	347.8	557.1	507.3	469.8	50 × 10 × 1	286.0	292.7	435.2	371.0	372.8
25 × 4 × 2	415.0	441.0	596.0	638.7	464.0	50 × 10 × 2	276.6	295.4	400.4	394.0	328.5
25 × 4 × 3	309.0	327.4	415.8	370.7	368.4	50 × 10 × 3	298.4	314.2	542.5	919.5	384.0
25 × 4 × 4	162.0	168.0	202.3	187.5	188.4	50 × 10 × 4	262.0	256.9	453.4	426.6	338.1
25 × 4 × 5	132.5	137.6	260.4	141.0	231.2	50 × 10 × 5	68.2	69.2	341.4	120.0	252.2
25 × 4 × 6	160.0	182.8	319.6	173.8	254.6	50 × 10 × 6	79.2	95.8	305.8	131.0	233.8
25 × 4 × 7	175.0	180.6	304.6	190.0	262.8	50 × 10 × 7	109.7	107.5	379.3	173.0	280.0
25 × 4 × 8	199.0	238.8	355.5	224.0	312.1	50 × 10 × 8	126.4	155.6	335.2	174.0	263.0
25 × 4 × 9	758.2	767.3	1171.9	783.6	1068.8	50 × 10 × 9	574.3	620.1	1244.5	1008.0	855.2
25 × 4 × 10	817.2	950.6	1227.1	941.8	1202.2	50 × 10 × 10	612.8	678.3	1251.7	716.0	787.2
25 × 6 × 1	231.0	236.9	311.5	304.4	268.8	50 × 20 × 1	176.9	191.5	316.6	399.0	249.7
25 × 6 × 2	210.5	226.4	274.3	256.0	238.1	50 × 20 × 2	148.0	167.2	260.7	278.0	191.0
25 × 6 × 3	155.3	165.5	203.2	359.0	173.7	50 × 20 × 3	117.0	114.0	252.6	797.2	153.4
25 × 6 × 4	91.0	105.0	154.3	136.8	125.5	50 × 20 × 4	94.0	94.9	246.1	356.2	152.9
25 × 6 × 5	76.0	77.6	183.9	95.5	152.3	50 × 20 × 5	25.4	31.3	196.2	35.2	146.8
25 × 6 × 6	74.8	76.0	189.0	80.0	161.1	50 × 20 × 6	24.0	30.0	190.2	38.0	132.7
25 × 6 × 7	104.0	106.1	216.3	128.8	181.4	50 × 20 × 7	50.0	51.2	223.3	73.2	163.4
25 × 6 × 8	107.0	115.2	206.7	117.0	178.1	50 × 20 × 8	45.2	55.7	194.3	71.0	148.9
25 × 6 × 9	541.0	547.6	696.8	569.0	605.6	50 × 20 × 9	320.3	325.4	665.3	437.3	454.9
25 × 6 × 10	550.1	562.2	727.5	574.2	622.7	50 × 20 × 10	310.0	320.8	664.4	434.0	437.6

Table 7. The results of five algorithms on $AVG(3)$.

Instance	ABC-AC	MPH	HPSOGA	TAFOA	ABC	Instance	ABC-AC	MPH	HPSOGA	TAFOA	ABC
50 × 30 × 1	107.9	123.8	209.9	196.0	154.8	250 × 20 × 1	822.3	746.9	1506.0	1314.7	1095.7
50 × 30 × 2	109.0	121.1	215.7	199.0	143.5	250 × 20 × 2	755.3	791.0	1570.0	1313.3	1099.1
50 × 30 × 3	87.1	79.1	193.5	232.0	107.6	250 × 20 × 3	841.6	832.9	1889.1	6921.4	1453.1
50 × 30 × 4	89.6	93.7	191.5	369.3	106.3	250 × 20 × 4	807.2	953.5	2217.2	2752.7	1575.7
50 × 30 × 5	16.0	15.6	150.1	16.0	104.9	250 × 20 × 5	76.0	84.1	1186.7	102.8	929.9
50 × 30 × 6	16.0	16.3	153.3	27.0	102.4	250 × 20 × 6	83.3	94.8	1218.7	109.0	870.3
50 × 30 × 7	38.0	36.4	175.0	50.0	116.1	250 × 20 × 7	195.3	201.4	1406.2	268.5	1033.3
50 × 30 × 8	37.0	37.9	174.1	51.4	113.7	250 × 20 × 8	205.4	211.7	1398.1	229.8	941.8
50 × 30 × 9	214.0	232.5	529.0	412.0	331.6	250 × 20 × 9	1600.0	1573.0	4514.0	2386.1	2891.3
50 × 30 × 10	211.0	224.2	520.6	409.3	320.8	250 × 20 × 10	1625.7	1637.8	4504.4	2119.0	2865.2
150 × 10 × 1	1029.4	968.1	1588.4	1094.0	1271.9	250 × 30 × 1	559.1	503.3	1202.9	1037.4	793.3
150 × 10 × 2	873.8	888.6	1445.4	1286.2	1157.7	250 × 30 × 2	517.1	546.1	1188.9	892.2	783.1
150 × 10 × 3	1084.0	1114.6	1770.8	3766.9	1364.4	250 × 30 × 3	348.0	541.2	1308.9	926.3	942.4
150 × 10 × 4	784.7	804.5	1463.5	1594.7	1459.0	250 × 30 × 4	613.6	582.2	1643.0	3212.1	1121.3
150 × 10 × 5	150.1	166.2	1130.8	247.0	984.3	250 × 30 × 5	49.1	50.7	1022.8	67.0	683.8
150 × 10 × 6	168.5	182.8	1046.3	226.0	862.1	250 × 30 × 6	48.4	58.2	945.2	78.0	638.3
150 × 10 × 7	279.3	291.3	1250.6	405.0	1067.9	250 × 30 × 7	127.1	135.8	1107.6	163.5	701.9
150 × 10 × 8	306.1	328.8	1208.1	442.0	976.1	250 × 30 × 8	130.3	145.4	1078.0	186.1	688.1
150 × 10 × 9	2058.8	2099.6	3963.7	2972.0	3111.2	250 × 30 × 9	1160.7	1149.7	3317.7	1390.0	2124.2
150 × 10 × 10	2078.1	2191.3	3830.2	2996.0	2973.6	250 × 30 × 10	1162.5	1169.2	3174.7	1600.0	2073.9
150 × 20 × 1	476.0	439.7	877.9	670.0	698.6	350 × 10 × 1	2327.7	2342.9	3687.0	2733.3	2976.1
150 × 20 × 2	424.6	426.4	933.2	758.3	674.1	350 × 10 × 2	2268.0	2296.1	3678.2	2822.4	2760.1
150 × 20 × 3	474.4	438.7	1175.7	2575.6	838.4	350 × 10 × 3	2184.6	2225.2	3923.0	4839.8	3920.0
150 × 20 × 4	523.8	500.4	1064.2	2663.3	727.0	350 × 10 × 4	1726.8	1695.6	2862.2	5564.9	3155.9
150 × 20 × 5	60.7	69.4	738.1	85.8	606.5	350 × 10 × 5	378.8	388.6	3045.1	513.0	2408.2
150 × 20 × 6	53.2	58.4	708.6	71.0	536.8	350 × 10 × 6	356.8	386.2	2824.5	389.0	2248.9
150 × 20 × 7	125.8	136.9	794.7	177.9	629.6	350 × 10 × 7	714.3	738.1	3262.9	872.4	2519.9
150 × 20 × 8	125.0	132.3	778.1	147.0	578.3	350 × 10 × 8	719.2	786.1	3185.2	803.0	2365.6
150 × 20 × 9	904.7	930.9	2671.1	1250.5	1681.1	350 × 10 × 9	4822.6	4855.4	9756.6	6081.0	7402.6
150 × 20 × 10	986.7	1085.4	2427.6	1194.8	1587.4	350 × 10 × 10	5062.8	5197.6	10406.8	5786.0	7222.3
150 × 30 × 1	283.5	294.3	700.8	473.9	452.9	350 × 20 × 1	1139.2	1096.9	2260.5	1818.9	1559.4
150 × 30 × 2	276.9	278.9	612.4	555.0	380.1	350 × 20 × 2	1044.1	1080.9	2120.1	1745.4	1451.8
150 × 30 × 3	222.4	273.0	730.3	1343.3	506.1	350 × 20 × 3	710.5	929.9	2016.8	1691.6	1628.5
150 × 30 × 4	211.4	265.6	849.4	1091.6	600.9	350 × 20 × 4	1129.2	1366.6	3171.1	4036.0	2286.3
150 × 30 × 5	31.0	31.6	556.8	47.0	357.4	350 × 20 × 5	108.9	120.5	1739.6	133.0	1210.7
150 × 30 × 6	31.4	33.4	554.5	43.0	319.9	350 × 20 × 6	106.0	118.4	1726.5	136.0	1169.4
150 × 30 × 7	80.0	88.7	611.3	123.0	411.7	350 × 20 × 7	268.5	275.7	2037.2	301.2	1387.3
150 × 30 × 8	79.2	91.3	596.3	110.0	378.7	350 × 20 × 8	270.6	295.6	1996.1	340.0	1328.7
150 × 30 × 9	605.3	616.3	2086.9	1128.0	1275.8	350 × 20 × 9	2318.8	2326.3	6539.2	3088.0	4018.8
150 × 30 × 10	605.3	621.2	1879.4	1194.0	1223.7	350 × 20 × 10	2337.9	2420.9	6302.0	3179.3	3965.8
250 × 10 × 1	1550.6	1545.7	2511.7	1710.1	2010.7	350 × 30 × 1	764.8	754.3	1708.3	1405.5	1140.0
250 × 10 × 2	1522.5	1565.3	2423.9	1888.5	1887.3	350 × 30 × 2	769.0	723.8	1606.0	1687.2	1030.9
250 × 10 × 3	1590.8	1634.2	2837.1	3904.2	2670.9	350 × 30 × 3	640.5	859.8	1998.4	2260.8	1547.6
250 × 10 × 4	1497.0	1512.6	2311.7	4101.8	1966.7	350 × 30 × 4	453.3	614.2	1936.0	1046.0	1401.4
250 × 10 × 5	262.1	274.7	2009.8	300.6	1664.3	350 × 30 × 5	62.0	64.2	1433.7	73.0	919.3
250 × 10 × 6	299.2	330.3	1959.6	359.0	1535.9	350 × 30 × 6	54.1	59.8	1330.2	76.0	915.5
250 × 10 × 7	518.3	530.3	2282.6	581.2	1801.9	350 × 30 × 7	175.1	174.0	1541.1	218.0	1014.5
250 × 10 × 8	565.4	594.0	2098.5	637.0	1702.7	350 × 30 × 8	172.0	184.9	1479.1	220.0	985.5
250 × 10 × 9	3376.1	3461.3	7143.9	3951.0	5229.1	350 × 30 × 9	1495.4	1482.6	4918.3	2143.0	2971.9
250 × 10 × 10	3540.2	3829.8	6826.0	3995.0	5247.4	350 × 30 × 10	1488.5	1547.8	5068.4	2394.0	2912.4

It can be found from Tables 2–4 that ABC-AC converges better than its comparative algorithms. ABC-AC produces smaller than or identical MIN with three comparative

algorithms on 258 of 300 instances; moreover, MIN of ABC-AC is less than that of TAFOA by at least 20 on 241 instances, smaller than that of MPH by at least 20 on 66 instances and better than that of HPSOGA by at least 20 on more than 250 instances. ABC-AC has better convergence than MPH, HSPOGA and TAFOA. This conclusion can also be drawn from Figure 3.

As shown in Tables 5–7, ABC-AC obtains smaller AVG than or the same AVG as MPH, TAFOA and HPSOGA on 278 instances; moreover, AVG of ABC-AC is better than that of its all comparative algorithms by at least 10 on more than 160 instances. ABC-AC possesses better average performance than its three comparative algorithms. Figure 3 also depicts the average performance differences between ABC-AC and each comparative algorithm.

It also can be seen from Tables A1–A3 that MAX of ABC-AC exceeds that of three comparative algorithms on only 18 instances. ABC-AC has smaller MAX than comparative algorithms by at least 10 on 253 instances. Figure 3 also shows that ABC-AC possesses better stability than the comparative algorithms.

The good performance of ABC-AC results from its adaptive competition, adaptive onlooker bee phase and new scout phase. Adaptive competition and adaptive onlooker bee phase can effectively lead to the extensive competition between two employed bee swarms, which can be evolved fully. A new scout phase can effectively extend the exploitation ability of ABC-AC. These features can result in a low possibility of a falling local optimum and a good balance between exploration and exploitation; thus, ABC-AC is a competitive algorithm for UPMSPR with PM.

6. Conclusions and Future Research

The consideration of additional resources and PM leads to a high application value of the results of UPMSPR with PM, and competition between swarms is an effective path to intensify the performance of ABC. In this study, new adaptive competition was implemented, and ABC-AC was proposed to solve UPMSPR with PM. Two employed bee swarms were evaluated and compete with each other in an adaptive way to dynamically select one from two cases employed in the bee phase.

The first is the shifting of search resources from the employed bee swarm with lower evolution quality to another one, and the second is the migration of solutions from the employed bee swarm with higher evolution quality to another one. An adaptive onlooker bee phase was implemented, and a new scout phase was given. A number of experiments were conducted on 300 instances. The computational results demonstrate that the new strategies of ABC-AC are effective, and ABC-AC has promising advantages in solving the considered UPMSPR.

Additional resources (learning effect, deteriorating jobs, etc.) often exist in an unrelated parallel machine manufacturing process; UPMSPR with these constraints is a future research topic. We will attempt to solve the above problems using meta-heuristics, such as the imperialist competitive algorithm. Competition or cooperation among populations are effective paths to improve the performance of meta-heuristics with multiple populations, and we will apply new methods to implement competition or cooperation to obtain new meta-heuristics with a high search efficiency. The hybrid flow shop scheduling problem with additional resources is also a future topic of ours.

Author Contributions: Methodology, M.L.; software, H.X. and D.L.; data curation, H.X. and D.L.; computation experiments; writing—review and editing, H.X. and 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 (61573264).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

Appendix A

Table A1. The results of five algorithms on $MAX(1)$.

Instance	ABC-AC	MPH	HPSOGA	TAFOA	ABC	Instance	ABC-AC	MPH	HPSOGA	TAFOA	ABC
$8 \times 2 \times 1$	326	326	357	326	326	$12 \times 6 \times 1$	145	161	240	174	161
$8 \times 2 \times 2$	269	269	277	326	269	$12 \times 6 \times 2$	151	156	188	162	160
$8 \times 2 \times 3$	197	216	209	197	198	$12 \times 6 \times 3$	62	62	80	70	72
$8 \times 2 \times 4$	241	252	413	283	251	$12 \times 6 \times 4$	63	65	115	87	72
$8 \times 2 \times 5$	194	194	244	194	194	$12 \times 6 \times 5$	49	50	92	82	76
$8 \times 2 \times 6$	139	139	164	139	139	$12 \times 6 \times 6$	57	60	105	64	71
$8 \times 2 \times 7$	204	205	259	224	204	$12 \times 6 \times 7$	57	67	116	111	87
$8 \times 2 \times 8$	171	171	212	171	171	$12 \times 6 \times 8$	64	82	102	77	85
$8 \times 2 \times 9$	544	544	607	544	544	$12 \times 6 \times 9$	250	308	403	480	284
$8 \times 2 \times 10$	577	577	708	634	634	$12 \times 6 \times 10$	263	323	439	322	305
$8 \times 4 \times 1$	124	131	176	137	131	$16 \times 2 \times 1$	573	645	719	664	623
$8 \times 4 \times 2$	119	125	164	126	126	$16 \times 2 \times 2$	600	655	763	690	654
$8 \times 4 \times 3$	174	192	183	190	177	$16 \times 2 \times 3$	522	534	613	544	557
$8 \times 4 \times 4$	123	124	152	171	131	$16 \times 2 \times 4$	642	813	1107	803	804
$8 \times 4 \times 5$	80	96	97	122	80	$16 \times 2 \times 5$	332	348	608	341	361
$8 \times 4 \times 6$	66	78	103	78	78	$16 \times 2 \times 6$	264	284	424	307	300
$8 \times 4 \times 7$	91	100	116	100	96	$16 \times 2 \times 7$	363	394	548	384	406
$8 \times 4 \times 8$	78	89	112	89	90	$16 \times 2 \times 8$	358	386	505	388	418
$8 \times 4 \times 9$	279	326	364	357	292	$16 \times 2 \times 9$	1165	1392	1631	1337	1416
$8 \times 4 \times 10$	275	317	382	342	317	$16 \times 2 \times 10$	1219	1416	1785	1718	1449
$8 \times 6 \times 1$	103	109	118	115	103	$16 \times 4 \times 1$	283	325	545	340	355
$8 \times 6 \times 2$	106	106	134	114	106	$16 \times 4 \times 2$	213	229	329	255	240
$8 \times 6 \times 3$	46	46	57	53	77	$16 \times 4 \times 3$	181	194	263	383	208
$8 \times 6 \times 4$	71	75	90	87	78	$16 \times 4 \times 4$	140	159	170	163	165
$8 \times 6 \times 5$	36	36	68	56	44	$16 \times 4 \times 5$	80	90	171	114	117
$8 \times 6 \times 6$	58	58	67	58	58	$16 \times 4 \times 6$	90	102	175	102	154
$8 \times 6 \times 7$	42	42	68	68	49	$16 \times 4 \times 7$	112	117	157	158	160
$8 \times 6 \times 8$	62	62	77	62	62	$16 \times 4 \times 8$	121	128	238	121	175
$8 \times 6 \times 9$	214	227	255	254	217	$16 \times 4 \times 9$	488	532	626	711	560
$8 \times 6 \times 10$	218	226	301	230	225	$16 \times 4 \times 10$	508	558	705	566	603
$12 \times 2 \times 1$	326	326	453	326	328	$16 \times 6 \times 1$	146	159	209	167	172
$12 \times 2 \times 2$	593	684	775	663	605	$16 \times 6 \times 2$	153	159	221	238	171
$12 \times 2 \times 3$	380	415	527	399	396	$16 \times 6 \times 3$	123	126	202	162	146
$12 \times 2 \times 4$	292	332	421	329	324	$16 \times 6 \times 4$	115	126	154	296	124
$12 \times 2 \times 5$	171	175	195	176	175	$16 \times 6 \times 5$	67	75	145	82	106
$12 \times 2 \times 6$	261	280	335	261	280	$16 \times 6 \times 6$	62	75	146	75	108
$12 \times 2 \times 7$	205	205	208	205	206	$16 \times 6 \times 7$	95	95	181	110	127
$12 \times 2 \times 8$	296	318	436	300	320	$16 \times 6 \times 8$	85	100	151	95	122
$12 \times 2 \times 9$	774	774	1164	774	811	$16 \times 6 \times 9$	356	389	492	389	402
$12 \times 2 \times 10$	1035	1094	1195	1129	1091	$16 \times 6 \times 10$	360	389	535	413	413
$12 \times 4 \times 1$	235	253	353	285	272	$20 \times 2 \times 1$	563	699	725	621	654
$12 \times 4 \times 2$	204	218	242	242	221	$20 \times 2 \times 2$	622	674	799	701	669
$12 \times 4 \times 3$	214	253	416	305	269	$20 \times 2 \times 3$	728	860	1291	1001	1004
$12 \times 4 \times 4$	229	244	301	269	240	$20 \times 2 \times 4$	596	710	946	753	753
$12 \times 4 \times 5$	93	96	164	94	117	$20 \times 2 \times 5$	448	482	568	448	515
$12 \times 4 \times 6$	70	84	128	91	113	$20 \times 2 \times 6$	338	422	559	424	422
$12 \times 4 \times 7$	112	115	160	125	138	$20 \times 2 \times 7$	504	568	644	528	579
$12 \times 4 \times 8$	98	107	151	134	131	$20 \times 2 \times 8$	397	549	659	533	568
$12 \times 4 \times 9$	403	438	546	439	451	$20 \times 2 \times 9$	1468	1735	2174	1701	1880
$12 \times 4 \times 10$	388	444	562	444	446	$20 \times 2 \times 10$	1507	1610	2126	1926	1956

Table A2. The results of five algorithms on $MAX(2)$.

Instance	ABC-AC	MPH	HPSOGA	TAFOA	ABC	Instance	ABC-AC	MPH	HPSOGA	TAFOA	ABC
20 × 4 × 1	355	383	671	470	463	30 × 2 × 1	1303	1624	1733	1660	1650
20 × 4 × 2	304	312	391	371	338	30 × 2 × 2	915	965	1146	932	1017
20 × 4 × 3	164	175	189	183	179	30 × 2 × 3	371	374	429	410	416
20 × 4 × 4	170	178	189	233	184	30 × 2 × 4	992	1164	1380	1375	1199
20 × 4 × 5	126	130	234	152	181	30 × 2 × 5	670	789	1050	814	894
20 × 4 × 6	111	130	215	143	176	30 × 2 × 6	697	820	1076	924	945
20 × 4 × 7	153	165	258	183	218	30 × 2 × 7	759	884	1095	914	1012
20 × 4 × 8	149	179	235	184	213	30 × 2 × 8	784	939	1142	889	959
20 × 4 × 9	631	677	1116	691	732	30 × 2 × 9	2303	2869	3206	2901	3090
20 × 4 × 10	629	729	1096	681	728	30 × 2 × 10	2544	2937	3450	2932	2994
20 × 6 × 1	210	223	314	264	260	30 × 4 × 1	405	459	702	543	581
20 × 6 × 2	203	225	267	298	229	30 × 4 × 2	389	408	651	443	461
20 × 6 × 3	180	188	286	316	198	30 × 4 × 3	357	376	449	419	423
20 × 6 × 4	59	71	133	71	84	30 × 4 × 4	382	413	604	547	426
20 × 6 × 5	79	94	190	102	144	30 × 4 × 5	156	176	539	190	314
20 × 6 × 6	59	62	155	65	121	30 × 4 × 6	130	150	305	156	286
20 × 6 × 7	107	114	231	140	163	30 × 4 × 7	219	221	556	254	346
20 × 6 × 8	84	92	183	106	148	30 × 4 × 8	179	209	399	204	344
20 × 6 × 9	445	479	696	534	521	30 × 4 × 9	1140	1219	1599	1233	1360
20 × 6 × 10	435	473	634	497	522	30 × 4 × 10	1080	1138	1436	1119	1306
25 × 2 × 1	770	934	1120	919	914	30 × 6 × 1	314	331	560	455	406
25 × 2 × 2	772	798	895	802	825	30 × 6 × 2	268	290	460	316	318
25 × 2 × 3	1520	1752	2222	2215	1953	30 × 6 × 3	226	243	374	568	271
25 × 2 × 4	281	312	369	299	332	30 × 6 × 4	277	302	455	857	336
25 × 2 × 5	632	748	1069	728	851	30 × 6 × 5	75	81	245	105	173
25 × 2 × 6	474	530	689	524	596	30 × 6 × 6	82	101	282	129	219
25 × 2 × 7	713	919	1100	765	970	30 × 6 × 7	114	123	313	142	201
25 × 2 × 8	559	611	698	594	664	30 × 6 × 8	127	136	309	172	253
25 × 2 × 9	2150	2395	3010	2269	2721	30 × 6 × 9	581	632	1041	642	700
25 × 2 × 10	2018	2222	2739	2191	2234	30 × 6 × 10	608	642	1279	723	812
25 × 4 × 1	339	356	608	517	497	50 × 10 × 1	299	306	502	371	381
25 × 4 × 2	422	447	709	648	483	50 × 10 × 2	277	303	434	394	337
25 × 4 × 3	309	350	478	403	392	50 × 10 × 3	326	335	613	932	395
25 × 4 × 4	171	174	215	189	193	50 × 10 × 4	262	263	608	432	364
25 × 4 × 5	133	145	297	141	257	50 × 10 × 5	70	72	433	120	275
25 × 4 × 6	160	190	367	182	278	50 × 10 × 6	81	103	339	131	257
25 × 4 × 7	175	182	359	190	288	50 × 10 × 7	110	110	461	173	295
25 × 4 × 8	199	245	471	224	324	50 × 10 × 8	133	164	360	174	281
25 × 4 × 9	759	776	1219	791	1102	50 × 10 × 9	581	634	1375	1008	1057
25 × 4 × 10	826	995	1322	995	1220	50 × 10 × 10	645	704	1395	775	837
25 × 6 × 1	231	248	346	310	279	50 × 20 × 1	185	195	379	399	256
25 × 6 × 2	212	233	323	256	245	50 × 20 × 2	148	190	290	278	198
25 × 6 × 3	159	171	269	368	177	50 × 20 × 3	119	119	365	809	163
25 × 6 × 4	91	106	183	142	133	50 × 20 × 4	102	102	284	376	161
25 × 6 × 5	76	79	230	109	163	50 × 20 × 5	29	34	212	38	158
25 × 6 × 6	76	76	216	85	176	50 × 20 × 6	24	35	213	38	140
25 × 6 × 7	104	113	264	145	201	50 × 20 × 7	50	57	236	93	180
25 × 6 × 8	107	124	220	117	203	50 × 20 × 8	48	59	223	71	157
25 × 6 × 9	541	557	750	605	635	50 × 20 × 9	323	332	717	519	477
25 × 6 × 10	551	567	791	575	638	50 × 20 × 10	310	326	741	434	445

Table A3. The results of five algorithms on $MAX(3)$.

Instance	ABC-AC	MPH	HPSOGA	TAFOA	ABC	Instance	ABC-AC	MPH	HPSOGA	TAFOA	ABC
50 × 30 × 1	116	136	226	196	159	250 × 20 × 1	944	767	1649	1317	1153
50 × 30 × 2	109	125	244	199	150	250 × 20 × 2	836	833	1713	1315	1124
50 × 30 × 3	88	91	231	232	122	250 × 20 × 3	884	851	2229	6939	1570
50 × 30 × 4	92	100	260	378	114	250 × 20 × 4	842	1047	2608	2765	1622
50 × 30 × 5	16	16	173	16	112	250 × 20 × 5	76	90	1359	109	962
50 × 30 × 6	16	17	166	27	113	250 × 20 × 6	86	107	1316	109	903
50 × 30 × 7	38	38	198	50	129	250 × 20 × 7	201	212	1611	272	1083
50 × 30 × 8	39	43	187	55	127	250 × 20 × 8	209	218	1534	239	988
50 × 30 × 9	214	249	588	412	341	250 × 20 × 9	1658	1585	4817	2387	2975
50 × 30 × 10	211	238	568	412	334	250 × 20 × 10	1668	1658	4816	2119	2965
150 × 10 × 1	1051	990	1685	1107	1291	250 × 30 × 1	677	512	1286	1055	821
150 × 10 × 2	874	917	1704	1287	1180	250 × 30 × 2	590	567	1251	905	820
150 × 10 × 3	1133	1134	1969	3785	1487	250 × 30 × 3	391	617	1572	935	1018
150 × 10 × 4	803	823	1836	1636	1633	250 × 30 × 4	656	601	1833	3219	1201
150 × 10 × 5	151	172	1277	247	1023	250 × 30 × 5	52	54	1090	67	703
150 × 10 × 6	175	193	1133	226	897	250 × 30 × 6	51	59	1057	78	661
150 × 10 × 7	281	302	1470	405	1094	250 × 30 × 7	128	140	1229	171	724
150 × 10 × 8	307	339	1350	442	1006	250 × 30 × 8	134	151	1218	198	710
150 × 10 × 9	2063	2142	4258	2972	3159	250 × 30 × 9	1176	1174	3794	1390	2218
150 × 10 × 10	2099	2234	4193	2996	3029	250 × 30 × 10	1172	1218	3688	1600	2137
150 × 20 × 1	523	449	1010	670	716	350 × 10 × 1	2362	2370	4096	2762	2998
150 × 20 × 2	432	441	1118	759	691	350 × 10 × 2	2404	2302	3882	2830	2808
150 × 20 × 3	497	466	1385	2595	926	350 × 10 × 3	2238	2506	4255	4872	4437
150 × 20 × 4	536	537	1191	2718	793	350 × 10 × 4	1786	1776	3913	5611	3539
150 × 20 × 5	62	81	789	95	634	350 × 10 × 5	382	397	3341	513	2457
150 × 20 × 6	54	60	780	71	570	350 × 10 × 6	365	392	3290	389	2345
150 × 20 × 7	133	146	869	183	647	350 × 10 × 7	723	769	3518	874	2603
150 × 20 × 8	125	139	827	147	602	350 × 10 × 8	725	794	3343	803	2407
150 × 20 × 9	1010	1014	3040	1278	1756	350 × 10 × 9	4852	4887	10646	6081	7602
150 × 20 × 10	1040	1101	2838	1202	1639	350 × 10 × 10	5105	5266	11632	5786	7454
150 × 30 × 1	294	318	752	598	471	350 × 20 × 1	1226	1127	2474	1834	1608
150 × 30 × 2	287	285	657	555	395	350 × 20 × 2	1074	1083	2361	1746	1505
150 × 30 × 3	251	300	844	1363	561	350 × 20 × 3	736	1255	2260	1702	1703
150 × 30 × 4	239	289	993	1101	656	350 × 20 × 4	1182	1416	3686	4108	2351
150 × 30 × 5	32	32	608	47	403	350 × 20 × 5	113	128	1937	133	1260
150 × 30 × 6	32	38	595	43	342	350 × 20 × 6	106	131	1861	136	1228
150 × 30 × 7	80	95	645	123	481	350 × 20 × 7	273	283	2247	302	1438
150 × 30 × 8	81	97	658	110	411	350 × 20 × 8	272	298	2187	340	1396
150 × 30 × 9	609	625	2204	1128	1334	350 × 20 × 9	2324	2340	7364	3088	4097
150 × 30 × 10	612	627	2117	1194	1263	350 × 20 × 10	2368	2427	6953	3269	4040
250 × 10 × 1	1563	1568	2621	1720	2040	350 × 30 × 1	847	774	1830	1421	1185
250 × 10 × 2	1536	1610	2792	1904	1944	350 × 30 × 2	806	754	1808	1692	1112
250 × 10 × 3	1635	1750	3381	3991	2794	350 × 30 × 3	709	951	2218	2304	1599
250 × 10 × 4	1542	1547	2665	4133	2073	350 × 30 × 4	531	700	2357	1046	1552
250 × 10 × 5	268	287	2186	305	1711	350 × 30 × 5	66	68	1530	73	973
250 × 10 × 6	301	338	2142	359	1579	350 × 30 × 6	55	63	1475	76	955
250 × 10 × 7	530	546	2501	589	1861	350 × 30 × 7	182	189	1746	218	1058
250 × 10 × 8	577	607	2289	637	1792	350 × 30 × 8	172	193	1642	220	1040
250 × 10 × 9	3401	3502	7838	3951	5341	350 × 30 × 9	1604	1500	5237	2143	3028
250 × 10 × 10	3577	3897	7558	3995	5329	350 × 30 × 10	1521	1589	5762	2394	2994

References

1. Edis, E.B.; Oguz, C.; Ozkarahan, I. Parallel machine scheduling with additional resources: Notation, classification, models and solution methods. *Eur. J. Oper. Res.* **2013**, *230*, 449–463. [CrossRef]
2. Ventura, J.A.; Kim, D. Parallel machine scheduling with earliness-tardiness penalties and additional resource constraints. *Comput. Oper. Res.* **2003**, *30*, 1945–1958. [CrossRef]
3. Zheng, X.L.; Wang, L. A two-stage adaptive fruit fly optimization algorithm for unrelated parallel machine scheduling problem with additional resource constraints. *Expert Syst. Appl.* **2016**, *65*, 28–39. [CrossRef]
4. Fanjul-Peyro, L.; Perea, F.; Ruiz, R. Models and matheuristics for the unrelated parallel machine scheduling problem with additional resources. *Eur. J. Oper. Res.* **2017**, *260*, 482–493. [CrossRef]
5. Fleszar, K.; Hindi, K.S. Algorithms for the unrelated parallel machine scheduling problem with a resource constraint. *Eur. J. Oper. Res.* **2018**, *271*, 839–848. [CrossRef]
6. Zheng, X.L.; Wang, L. A collaborative multiobjective fruit fly optimization algorithm for the resource constrained unrelated parallel machine green scheduling problem. *IEEE Trans. Syst. Man Cybern. Syst.* **2018**, *48*, 790–800. [CrossRef]
7. Villa, F.; Vallada, E.; Fanjul-Peyro, L. Heuristic algorithms for the unrelated parallel machine scheduling problem with one scarce additional resource. *Expert Syst. Appl.* **2018**, *93*, 28–38. [CrossRef]
8. Vallada, E.; Villa, F.; Fanjul-Peyro, L. Enriched metaheuristics for the resource unrelated parallel machine scheduling problem. *Comput. Oper. Res.* **2019**, *111*, 415–424. [CrossRef]
9. Chen, J.F. Unrelated parallel machine scheduling with second resource constraints. *Int. J. Adv. Manuf. Technol.* **2005**, *26*, 285–292. [CrossRef]
10. Bitar, A.; Dauzère-Pérès, S.; Yugma, C.; Roussel, R. A memetic algorithm to solve an unrelated parallel machine scheduling problem with auxiliary resources in semiconductor manufacturing. *J. Sched.* **2016**, *19*, 367–376. [CrossRef]
11. Afzalirad, M.; Shafipour, M. Design of an efficient genetic algorithm for resource-constrained unrelated parallel machine scheduling with machine eligibility restrictions. *J. Intell. Manuf.* **2018**, *29*, 423–437. [CrossRef]
12. Al-Harkan, I.M.; Qamhan, A.A. Optimize unrelated parallel machine scheduling problems with multiple limited additional resources, sequence-dependent setup times and release date constraints. *IEEE Access* **2019**, *7*, 171533–171547. [CrossRef]
13. Fanjul-Peyro, L. Models and an exact method for the unrelated parallel machine scheduling problem with setups and resources. *Expert Syst. Appl.* **2020**, *5*, 100022. [CrossRef]
14. Yepes-Borrero, J.C.; Villa, F.; Perea, F.; Caballero-Villalobos, J.P. GRASP algorithm for the unrelated parallel machine scheduling problem with setup times and additional resources. *Expert Syst. Appl.* **2020**, *141*, 112959. [CrossRef]
15. Pinar, Y.; Seyda, T.Y. Constraint programming approach for multi-resource-constrained unrelated parallel machine scheduling problem with sequence-dependent setup times. *Int. J. Prod. Res.* **2021**, *98*, 40–52.
16. Al-Harkan, I.M.; Qamhan, A.A.; Badwelan, A.; Alsamhan, A.; Hidri, L. Modified harmony search algorithm for resource-constrained parallel machine scheduling problem with release dates and sequence-dependent setup times. *Processes* **2021**, *9*, 654. [CrossRef]
17. Yang, D.L.; Cheng, T.C.E.; Yang, S.J.; Hsu, C.J. Unrelated parallel-machine scheduling with aging effects and multi-maintenance activities. *Comput. Oper. Res.* **2012**, *39*, 1458–1464. [CrossRef]
18. Tavana, M.; Zarook, Y.; Santos-Arteaga, F.J. An integrated three-stage maintenance scheduling model for unrelated parallel machines with aging effect and multi-maintenance activities. *Comput. Ind. Eng.* **2015**, *83*, 226–236. [CrossRef]
19. Wang, S.J.; Liu, M. Multi-objective optimization of parallel machine scheduling integrated with multi-resources preventive maintenance planning. *J. Manuf. Syst.* **2015**, *37*, 182–192. [CrossRef]
20. Gara-Ali, A.; Finke, G.; Espinouse, M.L. Parallel-machine scheduling with maintenance: Praising the assignment problem. *Eur. J. Oper. Res.* **2016**, *252*, 90–97. [CrossRef]
21. Avalos-Rosales, O.; Angel-Bello, F.; Alvarez, A.; Cardona-Valds, Y. Including preventive maintenance activities in an unrelated parallel machine environment with dependent setup times. *Comput. Ind. Eng.* **2018**, *123*, 364–377. [CrossRef]
22. 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]
23. Pang, J.H.; Tsai, Y.C.; Chou, F.D. Feature-extraction-based iterated algorithm to solve the unrelated parallel machine problem with periodic maintenance activities. *IEEE Access* **2021**, *9*, 139089–139108. [CrossRef]
24. Lei, D.M.; Yi, T. A novel shuffled frog-leaping algorithm for unrelated parallel machine scheduling with deteriorating maintenance and setup Time. *Symmetry* **2021**, *13*, 1574. [CrossRef]
25. Doush, I.A.; Al-betar, M.A.; Mohammed, A.A.; Santos, E.; Hammouri, A.I. Flow shop scheduling with blocking using modified harmony search algorithm with neighboring heuristics methods. *Appl. Soft Comput.* **2019**, *85*, 105861.
26. Xue, Y.; Wang, T.; Pang, W.; Liu, A.X. Self-adaptive parameter and strategy based particle swarm optimization for large-scale feature selection problems with multiple classifier. *Appl. Soft Comput.* **2020**, *88*, 106031. [CrossRef]
27. Kizieiewicz, B.; Salabun, W. A new approach to identifying a multi-criteria decision model based on stochastic optimization techniques. *Symmetry* **2020**, *12*, 1551. [CrossRef]
28. Karaboga, D. *An Idea Based on Honeybee Swarm for Numerical Optimization*; Technical Report TR06; Erciyes University, Engineering Faculty, Computer Engineering Department: Kayseri, Turkey, 2005.

29. Lei, D.M.; Cai, J.C. Multi-population meta-heuristics for production scheduling: A survey. *Swarm Evol. Comput.* **2020**, *58*, 100739. [CrossRef]
30. Bolaji, A.L.; Khader, A.T.; Al-betar, M.A.; Mohammed, A.A. Artificial bee colony algorithm, its variants and applications: A survey. *J. Theor. Appl. Inf. Technol.* **2013**, *47*, 434–459.
31. Li, J.Q.; Pan, Q.K.; Gao, K.Z. Pareto-based discrete artificial bee colony algorithm for multi-objective flexible job shop scheduling problem. *Int. J. Adv. Manuf. Technol.* **2011**, *55*, 1159–1169. [CrossRef]
32. Pan, Q.K.; Tasgetiren, M.F.; Suganthan, P.N.; Chua, T.J. A discrete artificial bee colony algorithm for the lot-streaming flow shop scheduling problem. *Inf. Sci.* **2011**, *181*, 2455–2468. [CrossRef]
33. Wang, L.; Zhou, G.; Xu, Y.; Liu, M. An enhanced Pareto-based artificial bee colony algorithm for the multi-objective flexible job-shop scheduling. *Int. J. Adv. Manuf. Technol.* **2011**, *60*, 1111–1123. [CrossRef]
34. Ying, K.C.; Lin, S.W. Unrelated parallel machine scheduling with sequence- and machine-dependent setup times and due date constraints. *Int. J. Innov. Comput. Inf. Control* **2012**, *8*, 3279–3297.
35. Lin, S.W.; Ying, K.C. ABC-based manufacturing scheduling for unrelated parallel machines with machine-dependent and job sequence-dependent setup times. *Comput. Oper. Res.* **2014**, *51*, 172–181. [CrossRef]
36. Han, Y.Y.; Gong, D.W.; Sun, X.Y. A discrete artificial bee colony algorithm incorporating differential evolution for the flow-shop scheduling problem with blocking. *Eng. Optim.* **2015**, *47*, 927–946. [CrossRef]
37. Caniyilmaz, E.; Benli, B.; Ilkay, M.S. An artificial bee colony algorithm approach for unrelated parallel machine scheduling with processing set restrictions, job sequence-dependent setup times, and due date. *Int. J. Adv. Manuf. Technol.* **2015**, *77*, 2105–2115. [CrossRef]
38. Asadzadeh, L. A parallel artificial bee colony algorithm for the job shop scheduling problem with a dynamic migration strategy. *Comput. Ind. Eng.* **2016**, *102*, 359–367. [CrossRef]
39. Zhang, R.; Chang, P.C.; Song, S.J.; Wu, C. A multi-objective artificial bee colony algorithm for parallel batch-processing machine scheduling in fabric dyeing processes. *Knowl.-Based Syst.* **2017**, *116*, 114–129. [CrossRef]
40. Gong, D.W.; Han, Y.Y.; Sun, J.Y. A novel hybrid multi-objective artificial bee colony algorithm for blocking lot-streaming flow shop scheduling problems. *Knowl.-Based Syst.* **2018**, *148*, 115–130. [CrossRef]
41. Lu, S.J.; Liu, X.B.; Pei, J.; Thai, M.T.; Pardalos, P.M. A hybrid ABC-TS algorithm for the unrelated parallel-batching machines scheduling problem with deteriorating jobs and maintenance activity. *Appl. Soft Comput.* **2018**, *66*, 168–182. [CrossRef]
42. 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.* **2021**, *59*, 5259–5271. [CrossRef]
43. Wang, J.; Lei, D.M.; Cai, J.C. An adaptive artificial bee colony with reinforcement learning for distributed three-stage assembly scheduling with maintenance. *Appl. Soft Comput.* **2022**, *117*, 108371. [CrossRef]
44. Chu, X.H.; Cai, F.L.; Gao, D.; Li, L.; Cui, J.S. An artificial bee colony algorithm with adaptive heterogeneous competition for global optimization problems. *Appl. Soft Comput.* **2020**, *93*, 106391. [CrossRef]
45. Li, J.Q.; Han, Y.Y. A hybrid multi-objective artificial bee colony algorithm for flexible task scheduling problems in cloud computing system. *Clust. Comput.* **2020**, *23*, 2483–2499. [CrossRef]
46. Meng, T.; Pan, Q.K. A distributed heterogeneous permutation flowshop scheduling problem with lot-streaming and carryover sequence-dependent setup time. *Swarm Evol. Comput.* **2021**, *60*, 100804. [CrossRef]
47. Salehi Mir, M.S.; Rezaeian, J. A robust hybrid approach based on particle swarm optimization and genetic algorithm to minimize the total machine load on unrelated parallel machines. *Appl. Soft Comput.* **2016**, *41*, 488–504. [CrossRef]
48. Taguchi, G. *Introduction to Quality Engineering*; Asian Productivity Organization: Tokyo, Japan, 1986.