Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2024.054473

ARTICLE

Dynamical Artificial Bee Colony for Energy-Efficient Unrelated Parallel Machine Scheduling with Additional Resources and Maintenance

Yizhuo Zhu[1], **Shaosi He**[2] and **Deming Lei**[2,*]

[1]School of Science, Hong Kong University of Science and Technology, Hong Kong, 999077, China

[2]School of Automation, Wuhan University of Technology, Wuhan, 430070, China

*Corresponding Author: Deming Lei. Email: deminglei11@163.om

Received: 29 May 2024 Accepted: 23 August 2024 Published: 15 October 2024

ABSTRACT

Unrelated parallel machine scheduling problem (UPMSP) is a typical scheduling one and UPMSP with various real-life constraints such as additional resources has been widely studied; however, UPMSP with additional resources, maintenance, and energy-related objectives is seldom investigated. The Artificial Bee Colony (ABC) algorithm has been successfully applied to various production scheduling problems and demonstrates potential search advantages in solving UPMSP with additional resources, among other factors. In this study, an energy-efficient UPMSP with additional resources and maintenance is considered. A dynamical artificial bee colony (DABC) algorithm is presented to minimize makespan and total energy consumption simultaneously. Three heuristics are applied to produce the initial population. Employed bee swarm and onlooker bee swarm are constructed. Computing resources are shifted from the dominated solutions to non-dominated solutions in each swarm when the given condition is met. Dynamical employed bee phase is implemented by computing resource shifting and solution migration. Computing resource shifting and feedback are used to construct dynamical onlooker bee phase. Computational experiments are conducted on 300 instances from the literature and three comparative algorithms and ABC are compared after parameter settings of all algorithms are given. The computational results demonstrate that the new strategies of DABC are effective and that DABC has promising advantages in solving the considered UPMSP.

KEYWORDS

Artificial bee colony; parallel machine scheduling; energy; additional resource

1 Introduction

Scheduling problems and algorithms have been extensively utilized in manufacturing and service industries to enhance production efficiency. As a typical scheduling problem, parallel machine scheduling problem (PMSP) extensively exists in many processes of manufacturing and service including production lines, hospital management systems, computer systems and shipping docks [1,2]. In unrelated parallel machine scheduling (UPMSP), the processing time of a job depends on its assigned machine. UPMSP with various conditions and constraints such as additional resources, maintenance and energy have been well studied and a number of results were obtained in the past decade [3–5].

There are many works on unrelated parallel machine scheduling problems with additional resources (UPMSPR). Ventura et al. [6] proved that the problem with one single type of additional resources is equivalent to the asymmetric assignment problem. Zheng et al. [7] reported a two-stage adaptive fruit fly optimization algorithm (FOA) with a heuristic and knowledge-guided search. Fanjul-Peyro et al. [8] presented two integer linear programming models and three matheuristics. Fleszar et al. [9] gave an efficient mixed-integer linear programming (MILP) model for a lower bound. Zheng et al. [10] proposed a collaborative multi-objective FOA to minimize carbon emissions. Villa et al. [11] developed several heuristics based on resource constraints and assignment rules. Afzalirad et al. [12] presented an integer mathematical programming model and two genetic algorithms for the problem with eligibility restrictions. Vallada et al. [13] applied an enriched scatter search and an enriched iterated greedy with a best-known heuristic and a repair mechanism.

UPMSPR with at least two real-life constraints is also studied, which are non-zero arbitrary release dates and sequence-dependent setup times (SDST) [14], processing resources, setup resources and shared resources [15], and additional resources in processing and setup [16]. Pinar et al. [17] proposed three heuristics and greedy randomized adaptive search procedures for UPMSP with setup times, and additional limited resources in setup.

Preventive maintenance (PM) is often applied to prevent potential failures and serious accidents in parallel machines and UPMSP with PM is frequently addressed. Some real-life constraints such as aging effects [18], multi-resources PM planning [19], deteriorating [20] and SDST [21] are included into UPMSP with PM. Various meta-heuristics including genetic algorithm [20], novel imperialist competitive algorithm (NICA) with an estimation of distribution algorithm [22], a differentiated shuffled frog-leaping algorithm [23], iterated algorithm [24], artificial bee colony (ABC [25]) and adaptive ABC [26].

The increasing environmental and energy pressures result in the increasing attention to energy saving or energy efficiency in manufacturing industries. In recent years, UPMSP with energy has received some attention. Che et al. [27] presented an improved continuous-time MILP model and a two-stage heuristic for UPMSP under time-of-use (TOU) electricity price. Cota et al. [28] proposed a MILP model and a novel math-heuristic algorithm for UPMSP with makespan and total consumption of electricity. Abikarram et al. [29] developed a mathematical optimization model and some analyses for UPMSP with energy cost. Zhang et al. [30] provided a new heuristic evolutionary algorithm to solve UPMSP with tool changes, makespan and total energy consumption. Wang et al. [31] applied a modified artificial immune algorithm to deal with UPMSP with energy, auxiliary resource shared among machines. For UPMSP with TOU electricity tariffs, Saberi-Aliabad et al. [32] presented a MILP model and a number of dominance rules and valid inequalities and Pei et al. [33] proposed an approximate algorithm after the problem is transformed into single machine problems with TOU electricity price. Zhang et al. [34] developed a combinatorial evolutionary algorithm (CEA) for UPMSP with setup times, limited worker resources and learning effect.

As stated above, UPMSPR, UPMSP with PM and UPMSP with energy have attracted attention and have been addressed using metaheuristics like ABC, NICA and FOA etc.; moreover, UPMSP with at least two real-life constraints is often studied [14–17,22–24]; however, UPMSP with additional resources, maintenance and energy is hardly investigated. In many unrelated parallel machine production processes, additional resources and maintenance often exist simultaneously and energy efficiency is important for production with the increasing pressures of environmental protection and energy price. The consideration of these things can result in a high application value of the obtained schedule, so it is essential to solve energy-efficient UPMSP with additional resources and PM.

It also can be found that ABC is an effective method to solve UPMSPR and UPMSP with PM. As a meta-heuristic inspired by the intelligent foraging behavior of honeybee swarm, ABC has some features such as simplicity and ease of implementation, and it has been successfully applied to deal with various production scheduling problems [35–39] and notable advantages of ABC in solving UPMSP [36–40] are proved by computational results. The energy-efficient UPMSP with additional resources and PM is an extended version of the UPMSP. It is still composed of the same sub-problems as UPMSP [36–40]. ABC has some particular features. It also has successfully applied to hand various UPMSP. There are close relations between UPMSP and its extended version. These three things reveal that ABC has potential optimization advantages in solving energy-efficient UPMSP with additional resources and PM, which is why ABC is chosen.

In this study, energy consumption, additional resources and PM are integrated into UPMSP and an effective way is provided for the problem by adding some new dynamical optimization mechanisms into ABC. The main contributions are summarized as follows. (1) Energy-efficient UPMSP with PM and additional resources is considered. (2) The dynamical artificial bee colony (DABC) is presented to minimize makespan and total energy consumption. Three heuristics are used in the initialization. Employed bee swarm and onlooker bee swarm are constructed and computing resources are shifted from the dominated solutions to non-dominated solutions in each swarm when the given condition is met. The dynamical employed bee phase is implemented by computing resource shifting and solution migration. The Dynamical onlooker bee phase involves computing resource shifting and feedback. This phase is applied to dynamically select search operators based on global and neighborhood searches. (3) Many experiments are conducted. The computational results demonstrate that new strategies of DABC are effective and that DABC has promising advantages in solving the considered UPMSP.

The remainder of the paper is organized as follows. Problem description is given in Section 2. Section 3 shows DABC for the considered problem. Section 4 gives numerical experiments on DABC and Section 5 shows the conclusions and some topics of future research are provided.

2 Problem Description

Energy-efficient UPMSP with additional resources and PM is composed of 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} is processing time of job J_i on machine M_k. An additional renewable resource is needed for each job. For job J_i processed on M_k, it needs r_{ki} units of the additional resource. At most $R_{\max}$ units of additional resources can be used at any time.

PM is considered. There is a time interval between two consecutive PMs, during which jobs are processed. For M_k, u_k is the length of the interval, w_k denotes the duration of PM, and the start time of the $g - th$ PM is $g \times u_k$

Machine M_k has three modes: processing mode, idle mode and PM mode. e_k, ie_k and pe_k indicate the energy consumption per unit time when M_k is in processing mode, idle mode and PM mode, respectively.

The mathematical mode of the problem is shown below:

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

$$TEC = \sum_{k=1}^{m}\sum_{i=1}^{n}\int_0^{C_{\max}} w_{ikt} \times e_k dt + \sum_{k=1}^{m}(ie_k \times ip_k + pe_k \times tp_k) \tag{2}$$

$$s.t. \quad \sum_{k=1}^{m}\sum_{l=1}^{n} x_{ikl} = 1 \qquad \forall i \tag{3}$$

$$\sum_{i=1}^{n} x_{ikl} \le 1 \qquad \forall k, l \tag{4}$$

$$b_{k,1} = 0 \ \forall k \tag{5}$$

$$z_{klg} \times \left(b_{k,l+1} - b_{k,l} - \sum_{l=1}^{n} p_{ik} \times x_{ikl}\right) = 0 \ \forall g \tag{6}$$

$$\left(1 - z_{klg}\right) \times \left(b_{k,l+1} - gu_k\right) = 0 \ \forall g \tag{7}$$

$$\sum_{k=1}^{m}\sum_{i=1}^{n}\sum_{l} r_{ki} x_{ikl} w_{ikt} \le R_{\max} \ \forall t \tag{8}$$

$$\overline{b_i} = \sum_{k=1}^{m}\sum_{l=1}^{n} b_{kl} \times x_{ikl} \quad \forall i \tag{9}$$

$$C_i = \overline{b_i} + \sum_{k=1}^{m}\sum_{l=1}^{n} p_{ik} \times x_{ikl} \quad \forall i \tag{10}$$

$$x_{ikl} \in \{0, 1\} \quad \forall k, l \tag{11}$$

where C_i indicates completion time of job J_i and $C_{\max}$ is maximum completion time of all jobs, w_{ikt} is 1 if job J_i is processed on M_k at time t and 0 otherwise. x_{ikl} is 1 if J_i is processed on position l on M_k and 0 otherwise. z_{klg} is 1 if $b_{k,l} + \sum_{l=1}^{n} p_{ik} \times x_{ikl} \le gu_k$ and 0 otherwise, $b_{k,l}$ is beginning time of job on position l of machine M_k, ip_k, tp_k are the total idle time and total maintenance duration, respectively. TEC denotes total energy consumption.

Eqs. (1) and (2) are about objectives. Constraint (3) indicates that job J_i is just needed to be assign to one machine. Constraint (4) denotes that at most one job is assigned to one position of one machine. Constraints ((6), (7)) are about PM. Constraint (8) is related one on additional resource. The last two constraints are about beginning time and completion time of job J_i.

For energy-efficient UPMSP with $C_{\max}$ and TEC, $z \succ x$ means that z dominates x and defined below:

$C^z_{\max} \le C^x_{\max}$, $TEC^z \le TEC^x$, at least one of $C^z_{\max} < C^x_{\max}$, $TEC^z < TEC^x$ exists. When $z \succ x$, $x \succ z$ are not met, z, x are non-dominated each other. $C^x_{\max}$ and TEC^x are makespan and total energy consumption of x.

An illustrative example with 2 machines and 8 jobs is given, the matrix of processing time and matrix of additional resource are provided in Eqs. (12) and (13), $e_1 = 2$, $e_2 = 3$, $ie_k = 1$, $pe_k = 5$, $u_k = 24$, $w_k = 3$.

$$(p_{ki})_{m\times n} = \begin{pmatrix} 5 & 6 & 6 & 5 & 2 & 4 & 4 & 6 \\ 3 & 3 & 4 & 4 & 4 & 5 & 3 & 3 \end{pmatrix} \tag{12}$$

$$(r_{ki})_{m\times n} = \begin{pmatrix} 5 & 7 & 7 & 3 & 3 & 7 & 6 & 5 \\ 3 & 4 & 5 & 8 & 4 & 3 & 3 & 2 \end{pmatrix} \tag{13}$$

Fig. 1 shows a schedule, in which 6 (7) as an example indicates job J_6 with r_{16} of 7.

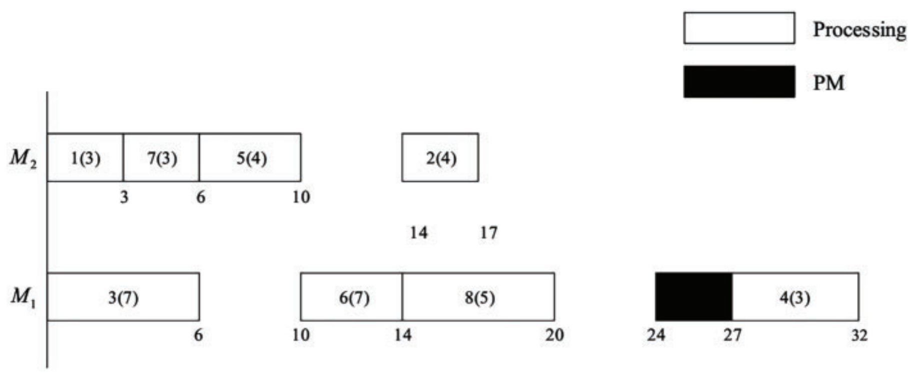

Figure 1: A schedule of example

3 DABC for Energy-Efficient UPMSP with Additional Resource and PM

Dynamical optimization mechanisms such as feedback and competition have been successfully used in ABC to adjust dynamically search operators or search behaviors [41–45]. The search advantages of dynamical mechanisms are tested and proved. In this study, dynamical optimization mechanism is implemented by computing resource shifting, solution migration and feedback.

3.1 Initialization

Lei et al. [26] proposed a two-string representation. For energy-efficient UPMSP with n jobs, m machines, R_{max} units of additional resource and PM, its solution consists of a machine assignment string $\left[M_{h_1}, M_{h_2}, \cdots, M_{h_n}\right]$ 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 described as follows:

(1) Obtain job permutation $[\pi_1, \pi_2, \cdots, \pi_n]$ by sorting all jobs in ascending order of θ_i.

(2) Start with π_1, for each job π_i, assign it to its machine $M_{h_{\pi_i}}$ according to the first string, decide if job π_i can be inserted into idle period and deal with PM as done in paper [16].

For the example in Section 2, a solution consists of $[M_2, M_2, M_1, M_1, M_2, M_1, M_2, M_1]$ and $[0.1, 0.3, 0.7, 0.57, 0.62, 0.23, 0.85, 0.41$, the obtained job permutation is $[1, 3, 7, 5, 6, 2, 8, 4]$, when J_6 is allocated on M_1, if no additional resource constraint is considered, J_6 can be processed between [6,10], however, the sum of the additional resource is 11, the additional resource constraint is violated, so J_6 is processed on [10,14]. For J_4, if it is processed directly after J_8, $C_4 = 25 > u_1$, so PM is first executed and then J_4 is processed. The obtained schedule is shown in Fig. 1.

β initial solutions are produced by heuristics. Heuristic 1 is used to produce solution x_1 and described as follows. For each J_i, $\min\{p_{ki}, 1 \le k \le m\}$ is decided, a machine $M_{h_{\pi_i}}$ with $p_{h_i i} = \min\{p_{ki}\}$ and the smallest $p_{h_i i} \times e_{h_i}$ are chosen; then a scheduling string is randomly generated. Heuristic 2 is used for solution x_2 and shown below. For each job J_i, compute $\min\{p_{ki} \times e_k, 1 \le k \le m\}$ and then select $M_{h_{\pi_i}}$ with the smallest $p_{h_i i}$ and $p_{h_i i} \times e_{h_i} = \min\{p_{ki} \times e_k\}$. The scheduling string of x_2 is also stochastically generated.

Heuristic 3 is used for each of $\beta - 2$ solutions: randomly produce a scheduling string, for each job J_i, if *rand* < 0.5, then decide a machine $M_{h_{\pi_i}}$ as done in heuristic 1 for each J_i; else determine a machine $M_{h_{\pi_i}}$ as done in heuristic 2 for each J_i. Where *rand* is random number following uniform distribution on [0, 1].

$N - \beta$ initial solutions are stochastically gotten. Employed bee swarm EB consists of randomly chosen $N/2$ solutions from P and onlooker bee swarm OB is composed of the remained $N/2$ solutions.

3.2 Dynamically Employed Bee Phase

Six neighborhood structures are used. $\mathcal{N}_1$ is used to move a randomly chosen job J_i on a machine with the biggest completion time to a machine M_k with the smallest $p_{ki} \times e_k$. $\mathcal{N}_2$ is similar with $\mathcal{N}_1$, J_i is moved to M_k with the smallest p_{ki} in $\mathcal{N}_2$. $\mathcal{N}_3$ is shown below. Decide $\max\left\{p_{h_i i} \times e_{h_i}, 1 \le i \le n\right\}$ and a job J_j with $p_{h_j j} \times e_{h_j} = \max\left\{p_{h_i i} \times e_{h_i}\right\}$ and move J_j to a machine M_k with $p_{kj} \times e_k = \min\left\{p_{lj} \times e_l, 1 \le l \le m\right\}$. $\mathcal{N}_4$ is adopted to exchange a randomly selected job on a machine M_k with the biggest completion time and a randomly chosen job on a stochastically decided $M_l, l \neq k$. $\mathcal{N}_5$ is shown below. Randomly choose M_k and swap two randomly selected jobs J_i, J_j on M_k, that is, θ_i, θ_j are exchanged. $\mathcal{N}_6$ is described as follows. Randomly decide a machine M_k and two randomly selected jobs J_i, J_j on M_k, then insert θ_i into position j on scheduling string.

Algorithm 1 describes the detailed steps of dynamical employed bee phase, where cn_i and $rank_{x_i}$ indicate the number of searches on a generation and *rank* of x_i decided by non-dominated sorting [46], $\mathcal{N}_g(x)$ is the set of neighborhood solutions of x produced by $\mathcal{N}_g$. The set Ω is used to store historical optimization data. When Ω is updated with x, x is added into Ω and all solutions of Ω are compared and all dominated ones are removed.

Algorithm 1: Dynamical employed bee phase

for each $x_i \in EB$ **do**
  for $g = 1$ to cn_i **do**
    execute global search between x_i and $y \in EB$
    perform neighborhood search NS_1 on x_i
  end for
  let $cn_i = 1$ if $cn_i = 0$ or $cn_i > 1$
end for
apply non-dominated sorting on all solutions of EB
for each $x_i \in EB$ **do**
  if $It \le trail_i, rank_{x_i} > 1$ **then**
    randomly choose solution $x_j \in EB$ with $rank_{x_j} = 1$, $cn_j = cn_j + 1, cn_i = 0$
  end if
  if $trial_i \ge 2 \times It$ and $rank_{x_i} > 1$ **then**
   replace x_i with a randomly produced solution and let $trial_i = 0$
  end if
end for
if each x_i with $rank_{x_i} = 1$ meets $It \le trail_i$ **then**
  decide a solution x_j with the smallest $trial_j$, execute NS_2 on x_j, implement solution migration from OB to EB
end if

Global search between x_i, y is shown below. Execute two-point crossover on machine assignment strings of x_i and a randomly chosen $y \in EB$, and obtain a new one z, if $z \succ x_i$ or z, x_i are non-dominated each other, then let $trail_i = 0$, x_i is used to renew Ω and z substitutes for x_i; otherwise, $trail_i = trail_i + 1$, perform two-point crossover on scheduling strings of x_i and a randomly selected $y \in EB$, obtain a new one z and update x_i, $trail_i$ and Ω according to the above condition.

Neighborhood search NS_1 on x_i is described as follows. Randomly decide $\mathcal{N}_g$, produce z $\in \mathcal{N}_g(x_i)$, update x_i, $trail_i$ and Ω in terms of conditions of global search.

Solution migration is described below. Define $\Delta = \left\{x_i \in EB \,\middle|\, rank_{x_i} = 1, It \leq trail_i\right\}$, then perform non-dominated sorting on OB, sort all solutions of OB in the ascending order of $rank$, for some solutions with same $rank$, sort them in the ascending order of $trail_i$, select the first $|\Delta|$ solutions, for each chosen solution x_i, a multiple neighborhood search is applied and let $trail_i = 0$.

For solution x_i, multiple neighborhood search is executed below. Let $g = 1$, repeat the following steps until $g = 7$: produce a solution z $\in \mathcal{N}_g(x_i)$, if $z \succ x_i$ or z, x_i are non-dominated each other, then z substitutes for x_i and let $g = 7$; otherwise $g = g + 1$.

Neighborhood search NS_2 on x_i is shown as follows. (1) Select the machine M_k with the biggest completion time and randomly choose a job J_i assigned to M_k, Then, repeat the following steps: insert J_i into each possible position on M_k and obtain a solution z until $z \succ x_i$. (2) Determine a machine M_k with the biggest energy consumption and randomly select a job J_i on M_k, repeat the following steps: move J_i to $M_l, l \neq k$ and obtain a solution z until $z \succ x_i$. In NS_2, if $z \succ x_i$ is not met, then $trail_i = trail_i + 1$; else $trail_i = 0$.

In dynamical employed bee phase, some dominated solutions with $rank_{x_i} > 1$ have $cn_i = 0$, and their computing resources are reallocated to non-dominated solutions with $rank_{x_i} = 1$. As a result, cn_i for some solutions exceed 1 and cn_i of other solutions are 0. This indicates that the search times for solutions are dynamically adjusted based on solution quality. Additionally, solution migration is triggered when all non-dominated solutions satisfy $It \leq trail_i$. In this case, some best solutions of OB are moved to EB and solutions of EB are dynamically adjusted when the given condition is met, Therefore, dynamic adjustment is applied in both scenarios.

3.3 Dynamical Onlooker Bee Phase

Four search operators $SO_1 - SO_4$ are given. SO_1 is described below. For a solution 0, select a $\mathcal{N}_g$ according to an adaptive process and produce a solution z $\in \mathcal{N}_g(x_i)$, if $z \succ x_i$, then x_i is used to update Ω and z substitutes for x_i; if z, x_i are non-dominated each other, then z is applied to renew Ω; if $x_i \succ z$, then randomly select $y \in EB$, multiple neighborhood search acts on y and x_i is replaced with y.

Adaptive process is depicted below. Choose a neighborhood structure by roulette selection based on Pse_g; if $rand > Q$, then randomly choose a neighborhood structure; suppose $\mathcal{N}_a$ is chosen, produce a new solution z $\in \mathcal{N}_a(x_i)$, if $z \succ x_i$, then $count_a = count_a + 2$; if z, x_i are non-dominated each other, then $count_a = count_a + 1$, where Q is threshold.

SO_2 is shown as follows. For a solution $x_i \in OB$, let $\alpha = 0$, execute variable neighborhood descent (VND) shown in Algorithm 2, if $\alpha = 0$, then perform multiple neighborhood search on x_i.

SO_3 is done in the following way. For a solution $x_i \in OB$, randomly choose $y \in EB$, perform global search between x_i, y as done in Lines 3–7 of Algorithm 1; then execute multiple neighborhood search on x_i. SO_4 has the same steps as SO_3; however, $y \in \Omega$ in SO_4.

Algorithm 2: VND

let $g = 1$
for $l = 1$ to R **do**
 produce a solution z $\in \mathcal{N}_g(x_i)$
 if $z \succ x_i$ or z, x_i are non-dominated each other **then**
  $\alpha = \alpha + 1$, update with Ω with x_i and replace x_i with z, $trail_i = 0, g = 1$
 else
  $trail_i = trail_i + 1, g = g + 1$
 end if
 end for

In $SO_1 - SO_4$, when multiple neighborhood search acts on x_i, for each z, if x_i cannot be replaced with z, then $trail_i = trail_i + 1$; otherwise, $trail_i = 0$.

Algorithm 3: Dynamical onlooker bee phase on $gen > 2$

1: perform non-dominated sorting on OB
2: compute Evo_{OB}^{gen-1}
3: **if** each $x_i \in OB$ with $rank_{x_i} = 1$ meets $It \leq trail_i$ **then**
4: randomly choose one of SO_3 and SO_4 and randomly select a y
5: **for** each solution $x_l \in OB$ **do**
6: **if** $rank_{x_l} > 1$ and $x_l \succ y$ **then**
7: stochastically a solution $x_j \in OB$ with $rank_{x_j} = 1$ and perform the chosen operator on $x_j \in OB$
8: **else**
9: execute the chosen operator on $x_l \in OB$
10: **end if**
11: **end for**
12: **else**
13: decide a search operator by feedback for each $x_i \in OB$
14: **end if**

In SO_1, an adaptive process is adopted to select neighborhood structure adaptively, SO_2 is an adaptive combination of VND and multiple neighborhood search, SO_3, SO_3 are combination of global search and multiple neighborhood search.

In onlooker bee phase, for each $\mathcal{N}_g$, set initial $count_g = 1$ and define selection probability Pse_g.

$$Pse_g = count_g / \sum_{l=1}^{6} count_l \tag{14}$$

Algorithm 3 describes dynamical onlooker bee phase on generation gen, where if SO_3 is chosen in Line 4, then $y \in EB$ is randomly decided; if SO_4 is selected, then $y \in \Omega$ is chosen randomly, in Lines 7, 9, when the chosen operator is executed, the decided y in Line 4 is directly used, Evo_{OB}^{gen} denotes the evolution quality.

$$Evo_{OB}^{gen} = \sum_{x_i \in OB} new_{x_i}^{gen} \tag{15}$$

where $new_{x_i}^{gen}$ is defined below. For x_i when an operator SO_l acts on x_i on generation, gen if new solution $z \succ x_i$, then $new_{x_i}^{gen} = new_{x_i}^{gen} + 2$; if z, x_i are non-dominated each other, $new_{x_i}^{gen} = new_{x_i}^{gen} + 1$.

Feedback is dynamical process used in control. In this study, feedback is applied to decide one of $SO_1 - SO_4$ dynamically, for each $x_i \in OB$, on generation *gen*, if $Evo_{OB}^{gen-1} < Evo_{OB}^{gen-2}$, then random select one operator of SO_1, SO_2 and perform the chosen operator on $x_i \in OB$; otherwise, execute the chosen operator on generation $gen - 1$ on $x_i \in OB$.

In dynamical onlooker bee phase, for each x_l, if $rank_{x_l} > 1$ and $x_l \succ y$, then computing resource of x_l is shifted to non-dominated solution $x_j \in OB$, feedback is used to dynamical decide search operator by selecting a new one if $Evo_{OB}^{gen-1} < Evo_{OB}^{gen-2}$ or using search operator of generation $gen - 1$, that is, search operator on generation *gen* is decided or affected by evolution on the previous two generations, obviously, computing resource and search operator are dynamically adjusted.

Algorithm 4: DABC

```
produce initial population P ∪ Ω  using heuristics and random way
decide EB, OB, gen = 1
while  stopping condition is not met do
  execute dynamical employed bee phase
  if gen ≤ 2, then
    for  each solution x_i ∈ OB do
          execute the randomly chosen operator from SO_1, SO_2
     end for
  else
     perform Algorithm 3
  end if
 apply scout phase
  gen = gen + 1
end while
output the non-dominated solutions in P ∪ Ω
```

In Algorithm 1, the search operator is combination of global search and neighborhood search NS_1, in Algorithm 3, SO_3, SO_4 are composed of global search and multiple neighborhood search, SO_1, SO_2 are neighborhood search-based operator; moreover, these operators are dynamically selected by feedback, these operators can be useful to make good balance between exploration and exploitation.

3.4 Algorithm Description

The detailed steps of DABC are shown in Algorithm 4.

Scout phase is described as follows. For each solution $x_i \in P$, if $trail_i \geq Limit$, then x_i is used to update Ω and then replaced with a randomly produced solution and $trail_i = 0$.

Unlike the previous ABC [36–40], DABC has the following new features. (1) The initial population is produced by three heuristics. (2) Dynamical employed bee phase is implemented by using computing resource shifting and solution migration. (3) Four search operators are used and dynamical onlooker bee phase is performed by applying computing resource shifting and feedback. The above dynamical optimization mechanisms such as solution migration and feedback can decide the number of searches and adjust solutions of swarms and search operator dynamically, as a result, search efficiency can be improved. On the other hand, many new things are required to be implemented when DABC is used, this may be a disadvantage of DABC.

4 Computational Experiments

Extensive experiments are conducted to test the performance of DABC for energy-efficient UPMSP with additional resource and PM. All experiments are implemented by using Microsoft Visual C++ 2019 and run on 8.0 G Random Access Memory 2.30 GHz Central Processing Unit Personal Computer.

4.1 Test Instances, Metrics and Comparative Algorithms

Fanjul-Peyro et al. [8] provided 300 instances, which can be divided into 30 types and the size of each type is depicted as $n \times m$, $n \in \{8, 12, 16, 20, 25, 30, 50, 150, 250, 350\}$ and $m \in \{2, 4, 6\}$. For each type $n \times m$, five ways are used for generating p_{ki} and two ways are applied for r_{ki}, 10 instances $n \times m \times 1, \cdots, n \times m \times 10$ are generated. Fanjul-Peyro et al. [8] described seven ways for p_{ki}, r_{ki} and the related data can be obtained directly from http://soa.iti.es/problem-instances (accessed on 24 May 2024). $R_{\max} = 5m$. We generate PM data as follows, w_k is integer selected from the same interval as p_{ki}, $u_k = round\left(w_k + 3.5 \times \max_{i=1,2,\cdots,n}\{p_{ki}\}\right)$. Where $round(x)$ is an integer being closet to x.

Metric $\mathcal{C}$ [47] is used to compare the approximate Pareto optimal set respectively obtained by algorithms.

$$\mathcal{C}(L, B) = \frac{|\{b \in B\colon \exists h \in L, h \succ b\}|}{|B|} \tag{16}$$

Metric ρ is the ratio of $|\{x \in \Omega_l \,|x \in \Omega^*\}|$ to $|\Omega^*|$ [48], where Ω_l is non-dominated set of Algorithm l, the reference set Ω^* consists of the non-dominated solutions in the union of non-dominated sets of all algorithms.

Metric DI_R [49] is used to measure the convergence performance by computing the distance of the non-dominated set Ω_l relative to a reference set Ω^*.

$$DI_R(\Omega_l) = \frac{1}{|\Omega^*|}\sum_{y\in\Omega^*}\min\left\{\sigma_{xy}|x \in \Omega_l\right\}0 \tag{17}$$

where σ_{xy} is the distance between a solution x and a reference solution y in the normalized objective space.

Lei et al. [23] proposed NICA for multi-objective UPMSP with PM. Shahidi-Zadeh et al. [3] presented a multi-objective harmony search (MOHS) for UPMSP. Zhang et al. [34] developed CEA for energy-efficient UPMSP with makespan and total energy consumption. These algorithms can be used to solve energy-efficient UPMSP with additional resource and PM after related steps on additional resource and PM are added into decoding procedure; moreover, they have promising advantages in solving UPMSP, so they are chosen as comparative algorithms.

ABC is used to show the effect of new strategies of DABC. ABC is constructed as follows: in employed bee phase, Lines 1–10 with $cn_i = 1$ for each $x_i \in P$ of Algorithm 1 are executed; in onlooker bee phase, a solution $x_i \in P$ is selected by binary tournament and the above Lines 1–10 are executed, scout phase of DABC is adopted in ABC.

4.2 Parameter Settings

DABC has following parameters: N, It, β, R, Q, $Limit$ and stopping condition. Stopping condition is first decided independently as done in [18], we found by experiments that DABC converges well when

$0.3n$ s CPU time reaches. We also obtained that when $0.3n$ s CPU time is applied, all comparative algorithms also converge fully, so stopping condition is set as $0.3n$ s CPU time for all algorithms.

An empirical method was used to determine the settings for other parameters by using the instance $50 \times 20 \times 5$. Table 1 shows the levels of each parameter. The orthogonal array $L_{27}\left(3^6\right)$ is tested. DABC with each combination runs 10 times on the chosen instance.

Table 1: Parameters and their levels

Parameters	Factor level		
	1	2	3
β	5	10	15
N	80	100	120
It	3	5	7
R	8	10	12
Q	0.25	0.3	0.35
Limit	8	10	12

Fig. 2 shows the results of ρ and S/N ratio, which is defined as $-10 \times \log_{10}\left(\rho^2\right)$. It can be found from Fig. 2 that DABC with following combination $N = 100, It = 5, \beta = 10, R = 10, Q = 0.3, Limit = 10$ produces better results than DABC with other combinations, moreover, we tested the above combination on all instances, the results reveal that the above combination is still effective, so the above parameter settings are adopted.

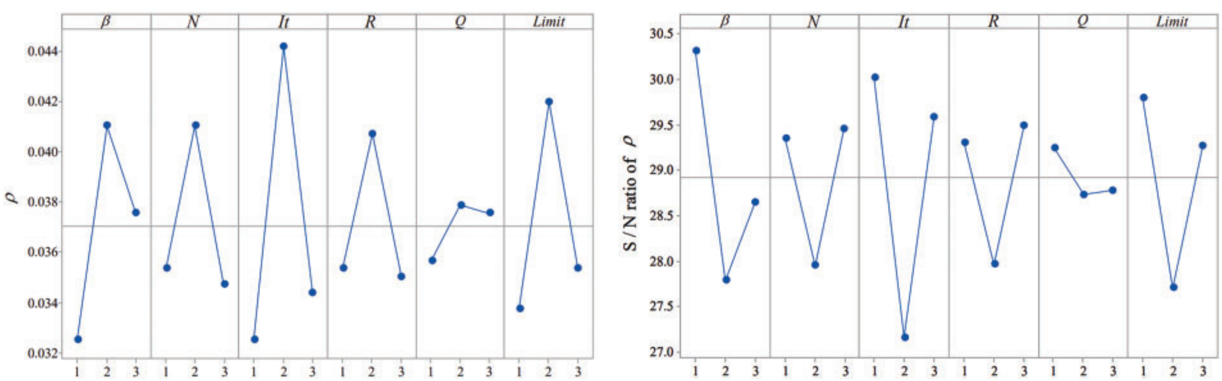

Figure 2: Main effect plot for mean and *S*/*N* ratio

ABC has $N = 100$, $Limit = 10$ and the above stopping condition.

Parameter settings of three comparative algorithms are directly selected from References [3,23,34] except that the stopping condition. To compare fairly, all algorithms should be stopped under the same condition, so MOHS, CEA and NICA are given the same stopping condition as DABC. We conducted experiments on other parameters of comparative algorithms, the experimental results show that each comparative algorithm with parameter settings from [3,23,34] can produce better results than the same algorithm with other parameter settings, so the original parameter settings are still used.

4.3 Results and Discussions

DABC, its three comparative algorithms and ABC are compared. Each algorithm randomly runs 10 times for each instance. Tables 2–9 describe the corresponding results of five algorithms. D, A, N, M, C denote DABC, ABC, NICA, MOHS and CEA. Fig. 3 shows the distribution of non-dominated solutions obtained by all algorithms.

Table 2: Results of all algorithms on metric $\mathcal{C}$

Type	$\mathcal{C}(D, C)$	$\mathcal{C}(C, D)$	$\mathcal{C}(D, N)$	$\mathcal{C}(N, D)$	$\mathcal{C}(D, M)$	$\mathcal{C}(M, D)$	$\mathcal{C}(D, A)$	$\mathcal{C}(A, D)$
8 × 2	0.000	0.000	0.000	0.000	0.0	0.000	1.000	0.000
	0.000	0.000	0.000	0.000	0.625	0.000	0.800	0.000
	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	10	10	10	10	10	10	10	10
8 × 4	0.000	0.000	0.769	0.000	1.000	0.000	1.000	0.000
	0.000	0.000	0.091	0.000	1.000	0.000	1.000	0.000
	0.308	0.250	0.000	0.000	0.000	0.000	0.667	0.000
	10	5	10	4	10	3	10	0
8 × 6	0.000	0.000	0.000	0.000	1.000	0.000	1.000	0.000
	0.000	0.000	0.333	0.000	1.000	0.000	1.000	0.000
	0.810	0.231	0.900	0.357	1.000	0.000	0.667	0.000
	10	5	10	3	10	0	10	0
12 × 2	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000
	0.400	0.000	0.000	0.000	0.750	0.000	0.600	0.000
	0.500	0.333	0.333	0.333	1.000	0.100	0.778	0.182
	10	6	10	7	10	2	10	1
12 × 4	0.429	0.000	0.333	0.000	1.000	0.000	1.000	0.000
	0.600	0.067	0.267	0.250	1.000	0.000	1.000	0.000
	0.471	0.235	0.500	0.500	1.000	0.000	0.500	0.273
	10	1	10	1	10	0	10	0
12 × 6	0.600	0.000	0.875	0.000	1.000	0.000	1.000	0.000
	0.722	0.111	0.917	0.077	1.000	0.000	1.000	0.000
	0.556	0.444	0.500	0.333	0.818	0.161	0.667	0.065
	10	1	10	0	10	0	10	0
16 × 2	0.909	0.000	1.000	0.000	1.000	0.000	1.000	0.000
	0.900	0.182	0.222	0.222	1.000	0.000	1.000	0.000
	0.800	0.600	0.500	0.875	0.500	0.000	0.778	0.333
	10	0	9	2	10	0	10	0
16 × 4	1.000	0.000	1.000	0.000	1.000	0.000	1.000	0.000
	0.625	0.143	0.000	0.000	1.000	0.000	1.000	0.000
	0.600	0.400	0.580	0.538	0.750	0.000	0.500	0.000
	10	1	10	1	10	0	10	0

Table 3: Results of all algorithms on metric $\mathcal{C}$

Type	$\mathcal{C}(D, C)$	$\mathcal{C}(C, D)$	$\mathcal{C}(D, N)$	$\mathcal{C}(N, D)$	$\mathcal{C}(D, M)$	$\mathcal{C}(M, D)$	$\mathcal{C}(D, A)$	$\mathcal{C}(A, D)$
16 × 6	1.000	0.000	1.000	0.000	1.000	0.000	1.000	0.000
	0.684	0.188	0.571	0.000	1.000	0.000	1.000	0.000
	0.444	0.400	0.438	0.412	0.917	0.118	0.667	0.176
	10	0	10	1	10	0	10	1
20 × 2	1.000	0.000	0.857	0.000	1.000	0.000	1.000	0.000
	0.750	0.000	0.700	0.200	1.000	0.000	1.000	0.000
	0.500	0.400	0.500	0.286	0.857	0.100	0.500	0.500
	10	0	10	1	10	0	10	1
20 × 4	0.857	0.000	0.857	0.000	1.000	0.000	1.000	0.000
	0.600	0.000	0.500	0.250	1.000	0.000	1.000	0.000
	0.500	0.500	0.800	0.375	0.500	0.000	0.750	0.000
	10	2	10	0	10	0	10	0
20 × 6	1.000	0.000	1.000	0.000	1.000	0.000	1.000	0.000
	0.474	0.200	0.000	0.000	1.000	0.000	1.000	0.000
	0.800	0.333	0.312	0.308	0.833	0.000	0.250	0.167
	10	1	10	1	10	0	10	0
25 × 2	1.000	0.000	0.333	0.000	1.000	0.000	1.000	0.000
	0.818	0.143	0.556	0.244	1.000	0.000	0.667	0.000
	0.500	0.250	0.500	0.500	0.857	0.000	0.357	0.286
	10	0	10	1	10	0	10	0
25 × 4	1.000	0.000	1.000	0.000	1.000	0.000	1.000	0.000
	0.800	0.000	0.500	0.000	1.000	0.000	1.000	0.000
	0.692	0.273	0.500	0.500	0.000	0.000	0.231	0.000
	10	0	10	1	10	1	10	0
25 × 6	1.000	0.000	1.000	0.000	1.000	0.000	1.000	0.000
	0.833	0.200	0.667	0.000	1.000	0.000	1.000	0.000
	0.500	0.500	0.400	0.333	0.923	0.000	0.429	0.238
	10	1	10	0	10	0	10	0
30 × 2	1.000	0.000	1.000	0.000	1.000	0.000	1.000	0.000
	0.688	0.105	0.500	0.000	1.000	0.000	1.000	0.000
	0.850	0.670	0.100	0.500	0.857	0.105	0.538	0.533
	10	1	9	1	10	0	10	1

Table 4: Results of all algorithms on metric $\mathcal{C}$

Type	$\mathcal{C}(D,C)$	$\mathcal{C}(C,D)$	$\mathcal{C}(D,N)$	$\mathcal{C}(N,D)$	$\mathcal{C}(D,M)$	$\mathcal{C}(M,D)$	$\mathcal{C}(D,A)$	$\mathcal{C}(A,D)$
30 × 4	1.000	0.000	1.000	0.000	1.000	0.000	1.000	0.000
	1.000	0.000	1.000	0.000	1.000	0.000	1.000	0.000
	0.571	0.286	0.750	0.333	1.000	0.000	0.333	0.250
	10	0	10	0	10	0	10	0
30 × 6	1.000	0.000	1.000	0.000	1.000	0.000	1.000	0.000
	0.667	0.000	0.667	0.000	1.000	0.000	1.000	0.000
	0.571	0.429	0.538	0.286	1.000	0.000	1.000	0.000
	10	0	10	0	10	0	10	0
50 × 10	1.000	0.000	1.000	0.000	1.000	0.000	1.000	0.000
	0.667	0.000	0.667	0.000	1.000	0.000	1.000	0.000
	0.600	0.333	0.333	0.333	1.000	0.000	0.462	0.333
	10	0	10	1	10	0	10	0
50 × 20	1.000	0.000	1.000	0.000	1.000	0.000	1.000	0.000
	0.500	0.000	0.500	0.000	1.000	0.000	1.000	0.000
	0.625	0.273	0.000	0.500	0.818	0.000	0.500	0.500
	10	0	9	1	10	0	10	1
50 × 30	1.000	0.000	1.000	0.000	1.000	0.000	1.000	0.000
	0.000	0.000	1.000	0.000	1.000	0.000	1.000	0.000
	0.611	0.400	0.462	0.294	1.000	0.000	0.500	0.375
	10	2	10	1	10	0	10	0
150 × 10	1.000	0.000	1.000	0.000	1.000	0.000	1.000	0.000
	0.400	0.000	0.333	0.000	1.000	0.000	1.000	0.000
	0.000	0.167	0.000	0.750	0.500	0.000	0.000	0.091
	9	1	7	3	10	0	9	1
150 × 20	1.000	0.000	1.000	0.000	1.000	0.000	1.000	0.000
	1.000	0.000	1.000	0.000	1.000	0.000	1.000	0.000
	0.333	0.421	0.000	0.000	1.000	0.000	0.000	1.000
	9	1	10	1	10	0	9	1
150 × 30	1.000	0.000	1.000	0.000	10.000	0.000	1.000	0.000
	0.500	0.000	0.750	0.000	10.000	0.000	1.000	0.000
	0.167	0.462	0.000	0.000	0.000	0.000	0.000	0.105
	9	3	10	2	10	1	9	1

Table 5: Results of all algorithms on metric $\mathcal{C}$

Type	$\mathcal{C}(D, C)$	$\mathcal{C}(C, D)$	$\mathcal{C}(D, N)$	$\mathcal{C}(N, D)$	$\mathcal{C}(D, M)$	$\mathcal{C}(M, D)$	$\mathcal{C}(D, A)$	$\mathcal{C}(A, D)$
250 × 10	1.000	0.000	1.000	0.000	1.000	0.000	1.000	0.000
	0.571	0.000	0.600	0.000	1.000	0.000	1.000	0.000
	0.400	0.556	0.000	0.000	0.000	0.000	0.000	0.400
	9	2	10	1	10	1	9	2
250 × 20	1.000	0.000	1.000	0.000	1.000	0.000	1.000	0.000
	0.833	0.000	0.667	0.000	1.000	0.000	1.000	0.000
	0.429	0.667	0.200	0.556	0.333	0.000	0.000	0.028
	8	3	7	4	10	0	9	2
250 × 30	1.000	0.000	1.000	0.000	1.000	0.000	1.000	0.000
	0.500	0.000	1.000	0.000	1.000	0.000	1.000	0.000
	0.222	0.467	0.200	0.333	1.000	0.000	0.615	0.786
	7	3	9	1	10	0	9	1
350 × 10	1.000	0.000	1.000	0.000	1.000	0.000	1.000	0.000
	0.750	0.000	0.778	0.111	1.000	0.000	1.000	0.000
	0.250	0.750	0.000	0.333	1.000	0.000	0.000	0.000
	8	2	9	1	10	0	10	2
350 × 20	1.000	0.000	1.000	0.000	1.000	0.000	1.000	0.000
	0.750	0.000	0.333	0.000	1.000	0.000	1.000	0.000
	0.125	0.625	0.182	0.500	1.000	0.000	0.000	0.000
	6	4	5	5	10	0	10	2
350 × 30	1.000	0.000	1.000	0.000	1.000	0.000	1.000	0.000
	0.000	0.000	0.500	0.000	1.000	0.000	1.000	0.000
	0.500	0.833	0.286	0.333	1.000	0.000	0.571	0.400
	6	5	6	5	10	0	10	0

Table 6: Results of all algorithms on metric ρ

Type	DABC	CEA	NICA	MOHS	ABC	Instance	DABC	CEA	NICA	MOHS	ABC
8 × 2	0.286	0.286	0.286	0.250	0.200	16 × 6	0.722	0.409	0.333	0.045	0.105
	0.250	0.250	0.250	0.200	0.091		0.500	0.273	0.158	0.000	0.000
	0.200	0.200	0.200	0.091	0.000		0.409	0.000	0.045	0.000	0.000
	10,10	10	10	6	2		10,10	1	0	0	0
8 × 4	0.550	0.391	0.333	0.182	0.062	20 × 2	1.000	0.417	0.375	0.000	0.148
	0.348	0.318	0.312	0.000	0.000		0.571	0.286	0.167	0.000	0.000
	0.273	0.273	0.100	0.000	0.000		0.417	0.000	0.000	0.000	0.000
	10,10	6	3	0	0		10,10	1	0	0	0

(Continued)

Table 6 (continued)

Type	DABC	CEA	NICA	MOHS	ABC	Instance	DABC	CEA	NICA	MOHS	ABC
8 × 6	0.750	0.476	0.333	0.000	0.048	20 × 4	0.692	0.400	0.286	0.000	0.400
	0.375	0.364	0.250	0.000	0.000		0.615	0.222	0.154	0.000	0.000
	0.333	0.200	0.000	0.000	0.000		0.381	0.000	0.000	0.000	0.000
	10,10	5	1	0	0		10,10	0	0	0	1
12 × 2	0.500	0.393	0.333	0.250	0.214	20 × 6	1.000	0.400	0.333	0.000	0.097
	0.333	0.250	0.250	0.000	0.111		0.556	0.258	0.182	0.000	0.000
	0.214	0.214	0.059	0.000	0.000		0.444	0.000	0.000	0.000	0.000
	10,10	6	5	2	1		10,10	0	0	0	0
12 × 4	0.647	0.375	0.273	0.000	0.083	25 × 2	1.000	0.333	0.300	0.000	0.286
	0.529	0.267	0.133	0.000	0.000		0.556	0.312	0.167	0.000	0.000
	0.364	0.206	0.071	0.000	0.000		0.333	0.000	0.000	0.000	0.000
	10,10	1	0	0	0		10,10	1	0	0	0
12 × 6	0.686	0.385	0.250	0.029	0.000	25 × 4	1.000	0.400	0.400	0.000	0.200
	0.556	0.333	0.111	0.000	0.000		0.700	0.222	0.200	0.000	0.000
	0.385	0.250	0.000	0.000	0.000		0.400	0.000	0.000	0.000	0.000
	10,10	1	0	0	0		10,10	0	0	0	0
16 × 2	0.800	0.500	0.500	0.111	0.143	25 × 6	0.800	0.400	0.333	0.000	0.250
	0.556	0.231	0.214	0.000	0.000		0.650	0.091	0.067	0.000	0.000
	0.333	0.000	0.000	0.000	0.000		0.360	0.000	0.000	0.000	0.000
	10,10	1	0	0	0		10,10	0	0	0	0
16 × 4	1.000	0.364	0.333	0.000	0.111	30 × 2	1.000	0.444	0.333	0.000	0.250
	1.000	0.364	0.333	0.000	0.111		1.000	0.444	0.333	0.000	0.250
	0.333	0.000	0.000	0.000	0.000		0.417	0.000	0.000	0.000	0.000
	10,10	1	0	0	0		10,10	1	0	0	0

Table 7: Results of all algorithms on metric ρ

Type	DABC	CEA	NICA	MOHS	ABC	Instance	DABC	CEA	NICA	MOHS	ABC
30 × 4	1.000	0.333	0.444	0.000	0.167	150 × 30	1.000	0.273	0.364	0.000	0.250
	0.556	0.111	0.222	0.000	0.000		0.577	0.154	0.250	0.000	0.000
	0.444	0.000	0.000	0.000	0.000		0.250	0.000	0.000	0.000	0.000
	10,10	0	1	0	0		10,10	1	2	0	1
30 × 6	0.800	0.500	0.500	0.000	0.000	250 × 10	1.000	0.500	0.357	0.000	0.100
	0.500	0.333	0.200	0.000	0.000		0.500	0.000	0.154	0.000	0.000
	0.400	0.000	0.000	0.000	0.000		0.214	0.000	0.000	0.000	0.000
	10,10	1	2	0	0		10,8	3	1	0	0
50 × 10	0.714	0.500	0.400	0.000	0.286	250 × 20	1.000	0.667	0.333	0.000	0.333
	0.667	0.167	0.286	0.000	0.000		0.600	0.171	0.059	0.000	0.000
	0.286	0.000	0.000	0.000	0.000		0.188	0.000	0.000	0.000	0.000
	10,10	1	0	1	1		10,8	2	2	0	2

(Continued)

Table 7 (continued)

Type	DABC	CEA	NICA	MOHS	ABC	Instance	DABC	CEA	NICA	MOHS	ABC
50 × 20	1.000	0.486	0.500	0.000	0.400	250 × 30	1.000	0.407	0.375	0.000	0.154
	0.500	0.192	0.243	0.000	0.000		0.722	0.125	0.077	0.000	0.000
	0.250	0.000	0.000	0.000	0.000		0.296	0.000	0.000	0.000	0.000
	10,8	3	4	0	0		10,9	1	1	0	0
50 × 30	1.000	0.553	0.333	0.000	0.250	350 × 10	1.000	0.750	0.750	0.000	0.083
	0.583	0.400	0.053	0.000	0.000		0.556	0.111	0.108	0.000	0.000
	0.083	0.000	0.000	0.000	0.000		0.000	0.000	0.000	0.000	0.000
	9,7	5	0	0	1		10,6	4	3	0	0
150 × 10	0.750	0.600	0.400	0.000	0.148	350 × 20	1.000	0.463	0.300	0.000	0.146
	0.625	0.250	0.175	0.000	0.000		0.615	0.308	0.077	0.000	0.000
	0.200	0.000	0.000	0.000	0.000		0.000	0.000	0.000	0.000	0.000
	10,9	1	2	0	0		8,7	3	2	0	2
150	1.000	0.346	0.500	0.000	0.038	350 × 30	1.000	0.667	0.600	0.000	0.385
×	1.000	0.000	0.000	0.000	0.000		0.600	0.250	0.115	0.000	0.000
20	0.250	0.000	0.000	0.000	0.000		0.077	0.000	0.000	0.000	0.000
	10,9	1	1	0	0		9,7	3	2	0	2

Table 8: Results of all algorithms on metric DI_R

Type	DABC	CEA	NICA	MOHS	ABC	Instance	DABC	CEA	NICA	MOHS	ABC
8 × 2	0.000	0.000	0.000	0.000	0.000	16 × 6	0.692	1.814	4.669	13.144	16.648
	0.000	0.000	0.000	0.000	3.846		1.152	3.538	5.147	26.361	40.427
	0.000	0.000	0.000	42.857	100.000		4.444	16.637	10.880	69.538	67.904
	10,10	10	10	6	2		10,10	0	0	0	0
8 × 4	0.000	0.000	0.000	3.766	6.654	20 × 2	0.000	0.755	1.203	3.408	1.811
	0.000	0.167	0.572	9.411	9.191		0.474	7.935	5.036	18.071	9.128
	0.321	0.999	1.417	30.664	34.327		2.297	50.000	43.180	94.993	100.000
	10,10	4	2	0	0		10,10	0	0	0	0
8 × 6	0.000	0.000	0.000	11.142	14.455	20 × 4	0.900	3.285	2.992	19.106	13.910
	0.000	0.000	3.951	22.031	28.493		3.135	3.586	9.642	39.259	31.781
	0.198	2.619	8.589	92.708	84.691		6.950	51.568	15.732	60.100	44.216
	10,10	5	1	0	0		10,10	0	0	0	0
12 × 2	0.000	0.000	0.000	0.000	0.000	20 × 6	0.000	3.731	4.697	16.191	13.669
	0.000	0.000	0.000	8.796	4.135		2.149	5.570	10.885	36.125	38.886
	0.690	4.688	1.517	73.949	100.000		4.688	17.075	24.271	97.793	69.746
	10,10	5	5	2	1		10,10	0	0	0	0
12 × 4	0.000	0.096	0.489	10.903	9.402	25 × 2	0.000	3.665	2.754	12.520	3.398
	0.688	1.850	4.780	22.040	22.271		1.127	6.534	7.667	45.544	13.626
	1.641	3.701	11.318	56.530	37.893		5.561	100.000	100.000	100.000	100.000
	10,10	1	0	0	0		10,10	0	0	0	0

(Continued)

Table 8 (continued)

Type	DABC	CEA	NICA	MOHS	ABC	Instance	DABC	CEA	NICA	MOHS	ABC
12 × 6	0.000	0.000	1.737	9.992	18.410	25 × 4	0.000	6.657	1.215	32.039	25.712
	0.787	3.535	9.426	22.824	34.006		2.768	20.957	15.231	58.581	55.415
	3.119	9.156	19.697	60.749	84.707		7.135	94.664	93.887	100.000	100.000
	10,10	1	0	0	0		10,10	0	0	0	0
16 × 2	0.228	0.429	0.657	6.583	2.310	25 × 6	0.000	2.743	3.121	14.836	13.859
	2.990	5.416	3.720	13.469	11.005		1.605	9.131	10.827	76.045	75.346
	29.537	41.479	41.776	55.219	42.501		4.835	18.484	55.152	98.989	88.533
	10,9	1	0	0	0		10,10	0	0	0	0
16 × 4	0.000	0.000	2.784	29.208	20.496	30 × 2	0.000	1.537	3.318	10.886	2.907
	1.086	5.178	6.996	39.978	41.635		0.777	21.111	23.611	34.565	12.180
	2.976	19.988	13.023	94.391	61.735		5.557	62.865	50.000	96.825	72.055
	10,10	1	0	0	0		10,10	0	0	0	0

Table 9: Results of all algorithms on metric DI_R

Type	DABC	CEA	NICA	MOHS	ABC	Instance	DABC	CEA	NICA	MOHS	ABC
30 × 4	0.000	4.078	9.050	20.873	12.064	150 × 30	0.000	3.147	1.818	33.880	22.208
	2.778	22.912	29.358	40.732	55.556		0.371	21.324	10.409	73.530	52.304
	7.843	100.000	47.900	91.081	100.000		14.916	68.540	55.243	97.673	100.000
	10,10	0	0	0	0		10,9	1	0	0	0
30 × 6	0.855	3.125	2.595	28.402	21.317	250 × 10	0.000	3.631	4.444	38.044	30.896
	3.852	12.425	15.976	83.571	61.155		1.498	12.070	10.196	86.830	90.751
	8.897	58.935	30.844	97.913	83.508		38.541	89.377	98.177	99.780	99.237
	10,10	0	0	0	0		10,7	3	3	0	0
50 × 10	0.000	6.373	1.899	40.233	26.576	250 × 20	0.000	4.809	5.845	48.209	41.680
	3.630	18.084	16.317	70.903	53.464		0.524	23.026	14.444	89.536	93.734
	24.515	31.599	47.880	98.895	99.434		33.232	95.257	97.190	97.930	99.700
	10,10	0	0	0	0		10,9	1	1	0	0
50 × 20	0.000	4.856	3.696	22.677	30.675	250 × 30	0.000	0.000	0.000	52.845	27.228
	2.206	8.892	13.516	61.503	55.940		0.233	22.886	18.901	84.399	77.308
	6.178	98.308	92.308	100.000	100.000		38.460	68.485	63.845	98.585	100.000
	10,10	0	0	0	0		10,7	3	3	0	0
50 × 30	0.000	0.000	5.420	22.854	25.063	350 × 10	0.000	3.469	4.549	43.642	35.814
	1.145	15.143	15.545	66.273	80.504		1.049	11.522	8.607	86.296	95.444
	25.644	100.000	78.571	98.730	100.000		58.968	73.077	96.254	99.267	99.037
	10,9	1	1	0	0		10,6	4	3	0	0
150 × 10	0.786	6.443	4.808	38.625	47.939	350 × 20	0.000	0.000	3.398	60.513	44.489
	5.818	18.276	8.589	70.293	77.728		2.099	19.703	38.876	83.983	95.917
	37.183	47.245	33.333	93.330	99.688		100.000	92.726	59.979	99.409	99.494
	10,9	1	1	0	0		9,8	2	2	1	1

(Continued)

Table 9 (continued)

Type	DABC	CEA	NICA	MOHS	ABC	Instance	DABC	CEA	NICA	MOHS	ABC
150 × 20	0.000	4.203	3.865	37.709	54.733	350 × 30	0.000	5.980	11.858	37.035	33.895
	0.000	23.529	33.226	77.475	73.128		2.273	14.596	16.250	85.323	82.288
	5.104	97.705	100.000	100.000	98.114		28.742	84.354	85.286	99.717	98.070
	10,10	0	0	0	0		10,9	1	0	0	0

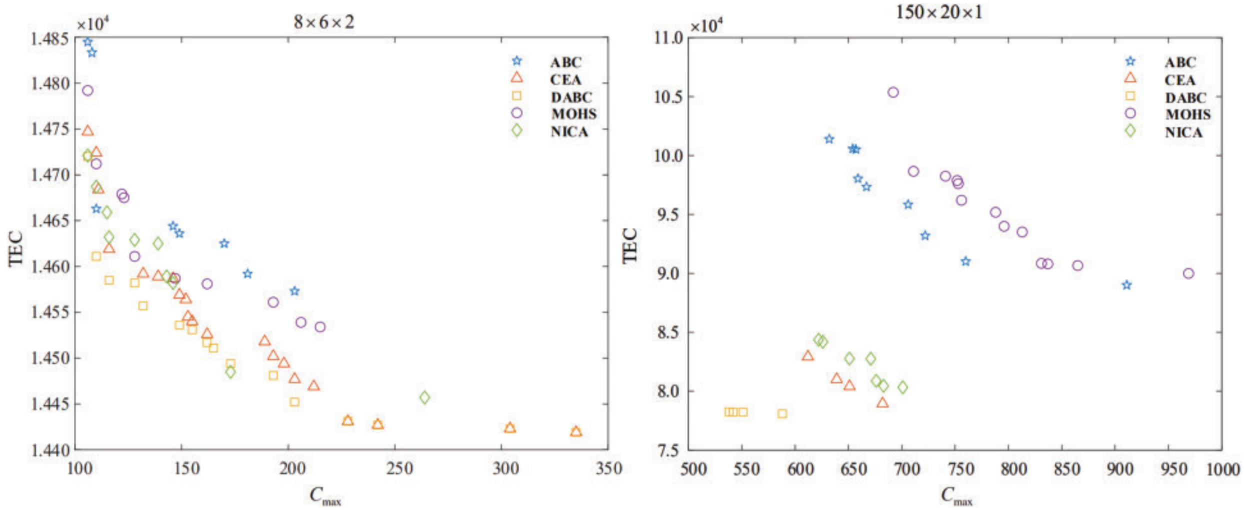

Figure 3: Distribution of non-dominated solutions of five algorithms

An effective way is applied to show results of five algorithms on 300 instances. In Tables 2–5, for each type $n \times m$, four groups of data are given, $\mathcal{C}(C, D)$ and $\mathcal{C}(D, C)$ are computed for each instances, 10 $\mathcal{C}(C, D)$ are sorted in the ascending order, the first group is the smallest $\mathcal{C}(C, D)$ and its corresponding $\mathcal{C}(D, C)$, the second is the fifth $\mathcal{C}(C, D)$ and its $\mathcal{C}(D, C)$, the third is the tenth $\mathcal{C}(C, D)$ and its corresponding $\mathcal{C}(D, C)$, let $\alpha_1 = \alpha_2 = 0$, for $\mathcal{C}(C, D)$, $\mathcal{C}(D, C)$ of each instance of $n \times m$, if $\mathcal{C}(C, D) < \mathcal{C}(D, C)$, then $\alpha_1 = \alpha_1 + 1$; if $\mathcal{C}(C, D) > \mathcal{C}(D, C)$, then $\alpha_2 = \alpha_2 + 1$; if $\mathcal{C}(C, D) = \mathcal{C}(D, C)$, then $\alpha_1 = \alpha_1 + 1$, $\alpha_2 = \alpha_2 + 1$, the fourth group consists of α_1, α_2.

For type 16 × 6, 10 pairs of $\mathcal{C}(C, D)$, $\mathcal{C}(D, C)$ are listed below. (0.4, 0.6), (0.143, 0.824), (0.188, 0.684), (0.4, 0.444), (0, 0.125), (0, 0.857), (0.2, 0.333), (0, 1), (0.286, 0.7), (0.25, 0.273), obviously, $\alpha_1 = 10$, $\alpha_2 = 0$, which means that $\mathcal{C}(C, D)$ is less than $\mathcal{C}(D, C)$ on 10 instances.

The same way is used to decide four group for $\mathcal{C}(D, N)$, $\mathcal{C}(N, D)$ and other columns, α_i is defined for the $i - th$ column.

In Tables 6, 7, for each type $n \times m$, 10 results are obtained and sorted in the descending order for each algorithm, the first group of data is the smallest value, the second group is the fifth value and the third group is the worst value, for each instance, a best value between DABC, ABC is decided, if ρ of DABC is equal to the best value, $\alpha_1 = \alpha_1 + 1$, if ρ of DABC is better than that of ABC, then $\alpha_2 = \alpha_2 + 1$, the first group is composed of α_1, α_2 for DABC, ABC. The way of α_1 is used to decide $\alpha_3, \alpha_4, \alpha_5, \alpha_6$ for

CEA, NICA, MOHS, ABC. Four groups of data for each type are decided for Tables 8, 9 in the same way of Tables 6, 7, 10, DI_R are sorted in the ascending order.

Table 10: Results to Wilcoxon-test

Wilcoxon-test	$\mathcal{C}$	DI_R	ρ
Wilcoxon-test (DABC, CEA)	0.000	0.000	0.000
Wilcoxon-test (DABC, NICA)	0.000	0.000	0.000
Wilcoxon-test (DABC, MOHS)	0.000	0.000	0.000
Wilcoxon-test (DABC, ABC)	0.000	0.000	0.000

Table 10 gives the results of pair-sample Wilcoxon-test, in which Wilcoxon-test (A, B) means a test conducted to judge whether Algorithm A gives a better sample mean than B and data on columns 2–4 are *p*-value. A significance level is 0.05. There is significant difference between A and B in the statistical sense if the *p*-value is less than 0.05.

As shown in Tables 2–5, DABC obtains the smaller value of $\mathcal{C}(A, D)$ and $\mathcal{C}(D, A)$ on 294 instances, ABC has the smaller value of $\mathcal{C}(A, D)$ and $\mathcal{C}(D, A)$ on 20 instances, and DABC generates smaller $\mathcal{C}(A, D)$ than $\mathcal{C}(D, A)$ on 280 instances; moreover, $\mathcal{C}(D, A)$ is equal to 1 on at least 138 instances, that is all solu-tions of ABC are dominated by non-dominated solutions of DABC on these instances. DABC converge significantly better than ABC.

Tables 6, 7 show that ρ of DABC outperforms ABC on more than 280 instances, while ρ of ABC is 0 on more than 177 instances, meaning ABC fails to contribute any members for the set Ω^*. Tables 8, 9 show that DABC obtains smaller DI_R than ABC on most of instances. Table 10 and Fig. 3 also reveal that performance differences between DABC and ABC are significant, obviously, the new strategies have positive impact on the performance of DABC, so new strategies are effective.

Tables 2–5 show that DABC produces smaller $\mathcal{C}(C, D)$ than $\mathcal{C}(D, C)$ on 241 instances and obtains $\mathcal{C}(D, C)$ of 1 on at least 31 instances. As shown in Tables 6 and 7, DABC outperforms CEA on 236 instances, with ρ greater than 0.6 on at least 71 instances, that is, members of reference set Ω^* are mainly produced by DABC. DABC also performs better than CEA on metric DI_R because DABC gets better DI_R than CEA on 260 instances. The above analyses reveal that DABC provides better results than CEA. Table 10 shows that the performance different between DABC and CEA are significant in the statistical sense. It can be found from Fig. 3 that the obtained non-dominated solutions can dominate most of solutions of other algorithms, thus, DABC performs better than CEA.

As listed in Tables 2–5, DABC has smaller $\mathcal{C}(N, D)$ than $\mathcal{C}(D, N)$ on more than 80% instances, DABC gets bigger ρ than NICA on more than 250 instances, and obtains better DI_R than NICA on 270 instances. There are notable performance differences between DABC and NICA; moreover, these differences also can be found in Table 10 and Fig. 3. On the other hand, DABC performs better than MOHS. $\mathcal{C}(D, M)$ is 1 on more than 190 instances and $\mathcal{C}(M, D)$ is 0 on 280 instances, that is, non-dominated solutions of DABC do not dominate by any solutions of MOHS. The notable convergence differences also can be seen from Fig. 3. ρ of MOHS is 0 on 276 instances and MOHS cannot provide any members of Ω^*. Tables 8, 9 show the performance differences between DABC and MOHS on metric DI_R. The statistical results in Table 10 also reveals that the performance differences between DABC, MOHS are significant.

The above analyses reveal that DABC performs better than MOHS, NICA and CEA. In DABC, three dynamical adjustment strategies are implemented, which are computing resource shifting, feedback and solution migration. Computing resource shifting can lead to extensive usage of non-dominated solutions, solution migration can increase the diversity of employed bee swarms and feedback based on four operators can result in the dynamical adjustment of the search operators according to search behavior. These strategies can effectively extend exploration ability, keep a high diversity of population and lead to a low possibility of falling local optima, thus, DABC is a promising method for energy-efficient UPMSP with additional resources and PM.

5 Conclusions

Additional resources, maintenance and energy are often considered in UPMSP; however, the existing researches seldom deal with these three things together in UPMSP. In this study, energy-efficient UPMSP with additional resources and PM is addressed, and a new algorithm called DABC is proposed to minimize makespan and total energy consumption. In DABC, some dynamical optimization mechanisms are implemented. The dynamic employed bee phase involves computing resource shifting and solution migration. The dynamical onlooker bee phase is applied by computing resource shifting and feedback. Extensive experiments are conducted on 300 instances. The computational results show that the new strategies such as the dynamical employed bee phase are effective and DABC can provide better results than its comparative algorithms.

UPMSP with several real-life conditions and constraints has attracted some attention. We will focus on UPMSP by involving additional resources, machine eligibility, and SDST, addressing these problems through meta-heuristics combined with new optimization mechanisms such as reinforcement learning and competition among sub-populations. We also handle distributed hybrid flow shop scheduling problems with some practical constraints in the near future. Additionally, distributed assembly scheduling problems involving transportation will be among our future research topics.

Acknowledgement: The authors would like to thank the editors and reviewers for their valuable work, as well as the supervisor and family for their valuable support during the research process.

Funding Statement: This research was funded by the National Natural Science Foundation of China (grant number 61573264).

Author Contributions: The authors confirm contribution to the paper as follows: study conception and design: Deming Lei, Shaosi He; data collection: Yizhuo Zhu; analysis and interpretation of results: Deming Lei, Shaosi He, Yizhuo Zhu; draft manuscript preparation: Deming Lei, Yizhuo Zhu, Shaosi He. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: Data supporting this study are described in the first paragraph of Section 4.1.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] T. C. E. Cheng and C. C. S. Sin, "A state-of-the-art review of parallel-machine scheduling research," *Eur. J. Oper. Res.*, vol. 47, no. 3, pp. 271–292, 1990. doi: 10.1016/0377-2217(90)90215-W.

[2] E. Mokotoff, "Parallel machine scheduling problems: A survey," *Asia-Pacific. J. Oper. Res.*, vol. 18, pp. 193–242, 2011.

[3] B. Shahidi-Zadeh, R. Tavakkoli-Moghaddam, A. Taheri-Moghadam, and I. Rastgar, "Solving a bi-objective unrelated parallel batch processing machines scheduling problem: A comparison study," *Comput. Oper. Res.*, vol. 88, no. 6, pp. 71–90, 2017. doi: 10.1016/j.cor.2017.06.019.

[4] C. Wang, X. Li, and Y. Gao, "A novel collaborative evolutionary algorithm with two-population for multi-objective flexible job shop scheduling," *Comput. Model. Eng. Sci.*, vol. 137, no. 2, pp. 1849–1870, 2023. doi: 10.32604/cmes.2023.028098.

[5] L. Wang and Y. Qi, "Scheduling an energy-aware parallel machine system with deteriorating and learning effects considering multiple optimization objectives and stochastic processing time," *Comput. Model. Eng. Sci.*, vol. 135, no. 1, pp. 325–339, 2023. doi: 10.32604/cmes.2022.019730.

[6] J. A. Ventura and D. Kim, "Parallel machine scheduling with earliness-tardiness penalties and additional resource constraints," *Comput. Oper. Res.*, vol. 30, no. 13, pp. 1945–1958, 2003. doi: 10.1016/S0305-0548(02)00118-1.

[7] X. L. Zheng and L. Wang, "A two-stage adaptive fruit fly optimization algorithm for unrelated parallel machine scheduling problem with additional resource constraints," *Expert. Syst. Appl.*, vol. 65, no. 9–12, pp. 28–39, 2016. doi: 10.1016/j.eswa.2016.08.039.

[8] L. Fanjul-Peyro, F. Perea, and R. Ruiz, "Models and matheuristics for the unrelated parallel machine scheduling problem with additional resources," *Eur. J. Oper. Res.*, vol. 260, no. 2, pp. 482–493, 2017. doi: 10.1016/j.ejor.2017.01.002.

[9] K. Fleszar and K. S. Hindi, "Algorithms for the unrelated parallel machine scheduling problem with a resource constraint," *Eur. J. Oper. Res.*, vol. 271, no. 3, pp. 839–848, 2018. doi: 10.1016/j.ejor.2018.05.056.

[10] X. L. Zheng and L. Wang, "A collaborative multiobjective fruit fly optimization algorithm for the resource constrained unrelated parallel machine green scheduling problem," *IEEE Trans. Syst. Man Cyber. Syst.*, vol. 48, no. 5, pp. 790–800, 2018. doi: 10.1109/TSMC.2016.2616347.

[11] F. Villa, E. Vallada, and L. Fanjul-Peyro, "Heuristic algorithms for the unrelated parallel machine scheduling problem with one scarce additional resource," *Expert. Syst. Appl.*, vol. 93, pp. 28–38, 2018. doi: 10.1016/j.eswa.2017.09.054.

[12] M. Afzalirad and M. Shafipour, "Design of an efficient genetic algorithm for resource-constrained unrelated parallel machine scheduling with machine eligibility restrictions," *J. Intell. Manuf.*, vol. 29, no. 2, pp. 423–437, 2018. doi: 10.1007/s10845-015-1117-6.

[13] E. Vallada, F. Villa, and L. Fanjul-Peyro, "Enriched metaheuristics for the resource unrelated parallel machine scheduling problem," *Comput. Oper. Res.*, vol. 111, pp. 415–424, 2019.

[14] I. M. Al-Harkan and A. A. Qamhan, "Optimize unrelated parallel machine scheduling problems with multiple limited additional resources, sequence-dependent setup times and release date constraints," *IEEE Access*, vol. 7, pp. 171533–171547, 2019. doi: 10.1109/ACCESS.2019.2955975.

[15] L. Fanjul-Peyro, "Models and an exact method for the unrelated parallel machine scheduling problem with setups and resources," *Expert Syst. Appl. X*, vol. 5, 2020, Art. no. 100022.

[16] A. Lopez-Esteve, F. Perea, and J. C. Yepes-Borrero, "GRASP algorithm for the unrelated parallel machines scheduling problem with additional resources during processing and setups," *Int. J. Prod. Res.*, vol. 61, no. 17, pp. 6013–6029, 2023. doi: 10.1080/00207543.2022.2121869.

[17] Y. Pinar and T. Y. Seyda, "Constraint programming approach for multiresource-constrained unrelated parallel machine scheduling problem with sequence-dependent setup times," *Int. J. Prod. Res.*, vol. 60, no. 7, pp. 2212–2229, 2022. doi: 10.1080/00207543.2021.1885068.

[18] D. L. Yang, T. C. E. Cheng, S. J. Yang, and C. J. Hsu, "Unrelated parallel machine scheduling with aging effects and multi-maintenance activities," *Comput. Oper. Res.*, vol. 39, no. 7, pp. 1458–1464, 2012. doi: 10.1016/j.cor.2011.08.017.

[19] S. J. Wang and M. Liu, "Multi-objective optimization of parallel machine scheduling integrated with multi-resources preventive maintenance planning," *J. Manuf. Syst.*, vol. 37, no. 7, pp. 182–192, 2015. doi: 10.1016/j.jmsy.2015.07.002.

[20] A. Gara-Ali, G. Finke, and G. Espinouse, "Parallel-machine scheduling with maintenance: Praising the assignment problem," *Eur. J. Oper. Res.*, vol. 252, no. 1, pp. 90–97, 2016. doi: 10.1016/j.ejor.2015.12.047.

[21] O. Avalos-Rosales, F. Angel-Bello, A. lvarez, and Y. Cardona-Valds, "Including preventive maintenance activities in an unrelated parallel machine environment with dependent setup times," *Comput. Ind. Eng.*, vol. 123, pp. 364–377, 2018. doi: 10.1016/j.cie.2018.07.006.

[22] M. Wang and G. H. Pan, "A novel imperialist competitive algorithm with multi-elite individuals guidance for multi-object unrelated parallel machine scheduling problem," *IEEE Access*, vol. 7, pp. 121223–121235, 2019. doi: 10.1109/ACCESS.2019.2937747.

[23] D. M. Lei and T. Yi, "A novel shuffled frog-leaping algorithm for unrelated parallel machine scheduling with deteriorating maintenance and setup time," *Symmetry*, vol. 13, no. 9, 2021, Art. no. 1574. doi: 10.3390/sym13091574.

[24] J. H. Pang, Y. C. Tsai, and F. D. Chou, "Feature-extraction-based iterated algorithm to solve the unrelated parallel machine problem with periodic maintenance activities," *IEEE Access*, vol. 9, pp. 139089–139108, 2021. doi: 10.1109/ACCESS.2021.3118986.

[25] D. M. Lei and M. Y. Liu, "An artificial bee colony with division for distributed unrelated parallel machine scheduling with preventive maintenance," *Comput. Ind. Eng.*, vol. 141, no. 6, 2020, Art. no. 106320. doi: 10.1016/j.cie.2020.106320.

[26] D. M. Lei and S. S. He, "An adaptive artificial bee colony for unrelated parallel machine scheduling with additional resource and maintenance," *Expert. Syst. Appl.*, vol. 205, no. 2, 2022, Art. no. 117577. doi: 10.1016/j.eswa.2022.117577.

[27] A. Che, S. B. H. Zhang, and X. Q. Wu, "Energy-conscious unrelated parallel machine scheduling under time-of-use electricity tariffs," *J. Clean. Prod.*, vol. 156, no. 2, pp. 688–697, 2017. doi: 10.1016/j.jclepro.2017.04.018.

[28] L. P. Cota, V. N. Coelho, F. G. Guimaraes, and M. J. F. Souza, "Bi-criteria formulation for green scheduling with unrelated parallel machines with sequence-dependent setup times," *Int. Trans. Oper. Res.*, vol. 28, no. 2, pp. 996–1017, 2021. doi: 10.1111/itor.12566.

[29] J. B. Abikarram, K. McConky, and R. Proano, "Energy cost minimization for unrelated parallel machine scheduling under real time and demand charge pricing," *J. Clean. Prod.*, vol. 208, no. 1, pp. 232–242, 2019. doi: 10.1016/j.jclepro.2018.10.048.

[30] L. K. Zhang, Q. W. Deng, G. L. Gong, and W. W. Han, "A new unrelated parallel machine scheduling problem with tool changes to minimise the total energy consumption," *Int. J. Prod. Res.*, vol. 58, no. 22, pp. 6826–6845, 2020. doi: 10.1080/00207543.2019.1685708.

[31] Z. Wang and T. Y. Liu, "A novel multi-objective scheduling method for energy based unrelated parallel machines with auxiliary resource constraints," *IEEE Access*, vol. 7, pp. 168688–168699, 2019. doi: 10.1109/ACCESS.2019.2954601.

[32] H. Saberi-Aliabad, M. Reisi-Nafchi, and G. Moslehi, "Energy-efficient scheduling in an unrelated parallel-machine environment under time-of-use electricity tariffs," *J. Clean. Prod.*, vol. 249, no. 2, 2020, Art. no. 119393. doi: 10.1016/j.jclepro.2019.119393.

[33] Z. Pei, M. Z. Wan, Z. Z. Jiang, Z. T. Wang, and X. Dai, "An approximation algorithm for unrelated parallel machine scheduling under TOU electricity tariffs," *IEEE Trans. Auto. Sci. Eng.*, vol. 18, no. 2, pp. 743–756, 2020. doi: 10.1109/TASE.2020.2995078.

[34] L. K. Zhang, Q. W. Deng, R. H. Lin, G. L. Gong, and W. W. Han, "A combinatorial evolutionary algorithm for unrelated parallel machine scheduling problem with sequence and machine-dependent setup times, limited worker resources and learning effect," *Expert. Syst. Appl.*, vol. 175, 2021, Art. no. 114843. doi: 10.1016/j.eswa.2021.114843.

[35] J. Q. Li, Q. K. Pan, and K. Z. Gao, "Pareto-based discrete artificial bee colony algorithm for multi-objective flexible job shop scheduling problem," *Int. J. Adv. Manuf. Techno.*, vol. 55, no. 9-12, pp. 1159–1169, 2011. doi: 10.1007/s00170-010-3140-2.
[36] K. C. Ying and S. W. Lin, "Unrelated parallel machine scheduling with sequence and machine-dependent setup times and due date constraints," *Int. J. Innov. Comput.*, vol. 8, pp. 3279–3297, 2012.
[37] K. C. Ying and S. W. Lin, "ABC-based manufacturing scheduling for unrelated parallel machines with machine-dependent and job sequence-dependent setup times," *Comput. Oper. Res.*, vol. 51, no. 5, pp. 172–181, 2014. doi: 10.1016/j.cor.2014.05.013.
[38] E. Caniyilmaz, B. Benli, and M. S. Ilkay, "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.*, vol. 77, no. 9–12, pp. 2105–2115, 2015. doi: 10.1007/s00170-014-6614-9.
[39] S. J. Lu, X. B. Liu, J. Pei, M. T. Thai, and P. M. Pardalos, "A hybrid ABC-TS algorithm for the unrelated parallel-batching machines scheduling problem with deteriorating jobs and maintenance activity," *Appl. Soft Comput.*, vol. 66, no. 2, pp. 168–182, 2018. doi: 10.1016/j.asoc.2018.02.018.
[40] D. M. Lei, Y. Yuan, and J. C. Cai, "An improved artificial bee colony for multi-objective distributed unrelated parallel machine scheduling," *Int. J. Prod. Res.*, vol. 59, no. 17, pp. 5259–5271, 2020. doi: 10.1080/00207543.2020.1775911.
[41] J. Q. Li and Y. Q. Han, "A hybrid multi-objective artificial bee colony algorithm for flexible task scheduling problems in cloud computing system," *Cluster Comput.*, vol. 23, no. 4, pp. 2483–2499, 2020. doi: 10.1007/s10586-019-03022-z.
[42] T. Meng and Q. K. Pan, "A distributed heterogeneous permutation flowshop scheduling problem with lot-streaming and carryover sequence-dependent setup time," *Swarm Evol. Comput.*, vol. 60, no. 9, 2021, Art. no. 100804. doi: 10.1016/j.swevo.2020.100804.
[43] D. W. Gong, Y. Y. Han, and J. Y. Sun, "A novel hybrid multi-objective artificial bee colony algorithm for blocking lot-streaming flow shop scheduling problems," *Knowl-Based Syst.*, vol. 148, pp. 115–130, 2018.
[44] J. Wang, D. M. Lei, and J. C. Cai, "An adaptive artificial bee colony with reinforcement learning for distributed three-stage assembly scheduling with maintenance," *Appl. Soft Comput.*, vol. 117, no. 2, 2022, Art. no. 108371. doi: 10.1016/j.asoc.2021.108371.
[45] J. Wang, H. T. Tang, and D. M. Lei, "A feedback-based artificial bee colony algorithm for energy-efficient flexible flow shop scheduling problem with batch processing machines," *Appl. Soft Comput.*, vol. 153, no. 1, 2024, Art. no. 111254. doi: 10.1016/j.asoc.2024.111254.
[46] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evolu. Comput.*, vol. 6, no. 2, pp. 182–197, 2002. doi: 10.1109/4235.996017.
[47] E. Zitzler and L. Thiele, "Multi-objective evolutionary algorithms: A comparative case study and the strength Pareto approach," *IEEE Trans. Evolu. Comput.*, vol. 3, no. 4, pp. 257–271, 1999. doi: 10.1109/4235.797969.
[48] D. M. Lei, "Pareto archive particle swarm optimization for multi-objective fuzzy job shop scheduling problems," *Int. J. Adv. Manuf. Tech.*, vol. 37, no. 1–2, pp. 157–165, 2008. doi: 10.1007/s00170-007-0945-8.
[49] J. D. Knowles and D. W. Corne, "On metrics for comparing nondominated sets," in *Proc. ICAIS*, New York, NY, USA, 2002, pp. 711–716.