TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 01/24 pp1249–1265
DOI: 10.26599/TST.2023.9010075
Volume 29, Number 5, October 2024

Multi-Objective Teaching-Learning-Based Optimizer for a Multi-Weeding Robot Task Assignment Problem

Nianbo Kang, Zhonghua Miao, Quan-Ke Pan*, Weimin Li, and M. Fatih Tasgetiren

Abstract: With the emergence of the artificial intelligence era, all kinds of robots are traditionally used in agricultural production. However, studies concerning the robot task assignment problem in the agriculture field, which is closely related to the cost and efficiency of a smart farm, are limited. Therefore, a Multi-Weeding Robot Task Assignment (MWRTA) problem is addressed in this paper to minimize the maximum completion time and residual herbicide. A mathematical model is set up, and a Multi-Objective Teaching-Learning-Based Optimization (MOTLBO) algorithm is presented to solve the problem. In the MOTLBO algorithm, a heuristic-based initialization comprising an improved Nawaz Enscore, and Ham (NEH) heuristic and maximum load-based heuristic is used to generate an initial population with a high level of quality and diversity. An effective teaching-learning-based optimization process is designed with a dynamic grouping mechanism and a redefined individual updating rule. A multi-neighborhood-based local search strategy is provided to balance the exploitation and exploration of the algorithm. Finally, a comprehensive experiment is conducted to compare the proposed algorithm with several state-of-the-art algorithms in the literature. Experimental results demonstrate the significant superiority of the proposed algorithm for solving the problem under consideration.

Key words: genetic algorithm; heuristic algorithm; Multi-Weeding Robot Task Assignment (MWRTA); teaching optimization algorithm

1 Introduction

With the advent of the new industrial revolution, future agricultural development[1, 2], where agricultural robots, including picking, weeding, and other robots, which are designed and developed for a certain task in the process of agricultural production, play an important role[3]. The cooperative operation of multiple agricultural robots has a remarkable application prospect[4]. Agricultural robots typically receive instructions to complete several tasks assigned by a central control system. Reasonable task allocation is related to the cost and efficiency of the entire system. This paper studies a Multi-Weeding Robot Task Assignment (MWRTA) problem to realize fixed target spraying and precise weed removal to save a considerable amount of labor and reduce unnecessary pesticide waste and environmental pollution.

The MWRTA problem can be attributed to a Multi-Robot Task Assignment (MRTA) problem that has been widely addressed by heuristics and meta-heuristics in recent years, because they can find optimal or near-optimal solutions in a reasonable computing time. The Teaching-Learning-Based

• Nianbo Kang, Zhonghua Miao, and Quan-Ke Pan are with School of Mechatronic Engineering and Automation, Shanghai University, Shanghai 200444, China. E-mail: knb1977903065@163.com; zhhmiao@shu.edu.cn; panquanke@shu.edu.cn.
• Weimin Li is with School of Computer Engineering and Science, Shanghai University, Shanghai 200444, China. E-mail: wmli@shu.edu.cn.
• M. Fatih Tasgetiren is with Industrial Engineering Department, Baskent University, Ankara 06790, Türkiye. E-mail: ftasgetiren@gmail.com.
* To whom correspondence should be addressed.
Manuscript received: 2023-05-07; revised: 2023-07-15; accepted: 2023-07-25

Optimization algorithm (TLBO) realizes optimization by establishing the "teaching" process of teachers and the mutual "learning" process of learners[5]. The TLBO is characterized by its simple structure with few parameters, strong convergence capability, and superior global search capability[6]. In addition, TLBO is proven to be effective in solving many combinatorial optimization problems[7–9]. Following the successful application of the TLBO algorithm, a Multi-Objective TLBO (MOTLBO) algorithm is presented to solve the MWRTA problem. The main contributions in this paper are presented as follows.

(1) This paper builds a mathematical model for the MWRTA problem and proposes an MOTLBO algorithm to solve the aforementioned problem.

(2) In MOTLBO, an Improved NEH (INEH, NEH is Nawaz, Enscore, and Ham heurisitic) heuristic and a maximum load-based heuristic are proposed to generate an initial population with a high level of quality and diversity.

(3) A grouping strategy and individual updating rule are designed to improve global exploitation and prevent falling into local optima in the "teaching" and "learning" phases.

(4) An efficient local search strategy that includes four kinds of neighborhood search operators is designed to improve the local exploration capability of MOTLBO.

This paper is structured as follows. Section 2 presents the literature review. Section 3 provides the mathematical model for the given problem. Section 4 comprehensively introduces the MOTLBO. Section 5 provides parameter calibration and a series of comparisons. Section 6 draws conclusions.

2 Literature Review

The current research on MRTA problems can be divided into two types: single-target and multi-target MRTA problems. For the single-target MRTA problem, Yuan et al.[10] established a task allocation model for multipicking robots to minimize overall picking time and designed a four-stage balanced heuristic auction algorithm. Lee et al.[11] proposed a framework for capability adjustment and load balancing in multirobot task allocation and successfully verified its effectiveness in cleaning tasks. Huang et al.[12] introduced a niche immune optimization algorithm based on SoftMax regression to solve the MRTA problem. Zhu et al.[13] proposed a method combining the standard particle swarm optimization algorithm with evolutionary game theory to address the MRTA problem. Chen and Sun[14] constructed a task allocation model for multi-heterogeneous mobile robots considering resource constraints, and proposed a leader-follower alliance method. Zhu and Yang[15] proposed a neural network task assignment method based on self-organizing mapping to solve the MRTA problem in dynamic environments. Guo and Liu[16] proposed a multirobot assignment algorithm based on an improved self-organizing mapping network to solve the problems of slow convergence speed and easy task conflict among the MRTA problems. Li and Yang[17] transformed several MRTA problems into multiple travel agent problems, and proposed a distributed cooperative task allocation strategy based on an improved gray wolf optimization algorithm. Nagarajan and Thondiyath[18] established a multi-robot task allocation model for minimizing turnaround time, and designed a task allocation method to control multiple robot pools using a set of static and mobile agents. Choudhury and Biswal[19] adopted a two-stage method to optimize task allocation among candidate robots, thus improving production efficiency and robot utilization. Chen et al.[20] proposed a marginal cost allocation heuristic and meta-heuristic improvement strategies based on large neighborhood searches. Chand and Carnegie[21] introduced a new multi-robot task allocation technique using fuzzy inference systems to reduce human user input. Wu et al.[22] established an MRTA model based on the SoftMan group, and introduced the resource requirement length algorithm and resource conformity degree algorithm to optimize the problem of robot resource contention.

For the multi-target MRTA problem, Zhou et al.[23] established a multi-warehouse handling robot task allocation model to distribute workload evenly and minimize transportation costs, and designed a balanced heuristic mechanism. Shen et al.[24] set up a robust optimization model with the number of vehicles and the cost of energy consumption as the targets, and proposed an improved adaptive large neighborhood search heuristic to solve the problem. Cui et al.[25] used a negotiation method based on game theory to study the MRTA problem, and designed a negotiation robot selection and negotiation set construction method based on a utility function, a negotiation mechanism is suitable for a distributed task allocation, and a negotiation strategy is based on game theory. Miyamoto and Inoue[26] established a model aiming at

Automated Guided Vehicle (AGV) driving distance and delivery delay time, and proposed a local/random method. Xu and Guo[27] set up a model with the maximum completion time, machine energy consumption, and the number of AGVs as targets, and proposed a hybrid evolutionary algorithm based on a segmented coding mode. Li et al.[28] set up a mathematical model and proposed an improved harmonic search algorithm with the objectives of AGV traveling distance, standard deviation of loading capacity, and standard deviation of the difference between the latest and expected delivery times. Eda et al.[29] established a model with AGV travel and balanced delivery times as targets, and proposed a Petri net decomposition method. Bai et al.[30] established a model aiming at the trajectory distance, trajectory time, trajectory threat, and trajectory coordination distance cost of UAVs, and used the NSGA-III algorithm to solve the problem.

A brief review reveals that many researchers have used heuristic and meta-heuristic algorithms, and achieved remarkable results. This finding shows that heuristic and meta-heuristic algorithms are feasible and effective for solving MRTA problems. However, studies in the agricultural field are limited. Therefore, different from the existing studies, this paper takes the weeding process in agricultural production as the background, and uses the TLBO algorithm to solve the problem raised.

3 Problem Modeling

3.1 Problem description

The MWRTA problem can be described as follows. As shown in Fig. 1, n task points in rectangles filled in green on the farmland must perform weeding tasks by spraying using two types of herbicides. The requirements for herbicide l (q_j^l) and weeding time (Δt_j) at each task point j can be detected by sensors. The travel distance between any two task points i and j, namely $d_{i,j}$, is also determined in advance. A set of m identical robots in the depot are given the task of performing the weeding tasks. Each robot starts from the depot, goes to the task points assigned to perform weeding tasks along the aisles, and then returns to the depot after all tasks assigned are executed. The spraying operation during a weeding task cannot be interrupted until the operation is completed. In other words, if the remaining load of a certain herbicide carried by a robot is less than the demand of the next task point, then the robot must return to the depot from the current task point to supplement the full load Q^l and then perform the next task. Without losing generality, this paper assumes that all task points must be assigned, and each task point can only be assigned to one robot. All robots operate normally, and accidents such as shutdowns, failures, and robot collisions do not occur. The speed of a robot is maintained as it moves forward and returns. The robot continues to drive at a constant speed in the process, and the speed remains known. The full load of a robot is greater than the demand of any task point. The problem lies in assigning task points to robots and sequencing them for each robot such that the completion time to finish all tasks and the total residual herbicide are minimized.

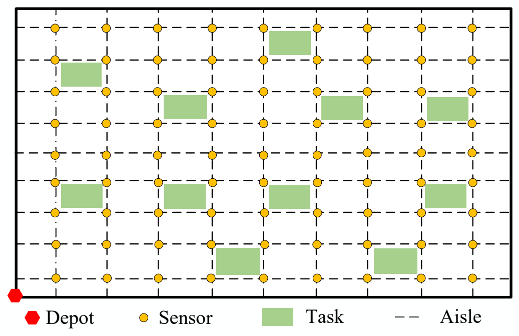


Fig. 1 Structure diagram of a smart farmland.

3.2 Mathematical model

A mathematical model is set up in this section as follows:

Objective:

$$\min C = \max_{i \in \mathrm{N} \setminus \{0\}} \left\{ \left(\mathrm{DT}_i + \frac{d_{i,0}}{v} \right) \times x_{i,0} \right\} \tag{1}$$

$$\min U = \sum_{l \in L} \sum_{i=1}^{n} \{ \mathrm{DQ}_i^l \times x_{i,0} \} \tag{2}$$

s.t.,

$$\sum_{i=0}^{n} x_{i,j} = 1, \ \forall j \in N \setminus \{0\} \tag{3}$$

$$\sum_{j=0}^{n} x_{i,j} = 1, \ \forall i \in N \setminus \{0\} \tag{4}$$

$$\sum_{j=1}^{n} x_{0,j} = m \tag{5}$$

$$\sum_{i=1}^{n} x_{i,0} = m \tag{6}$$

$$\sum_{(i,j)\in S\times S}^{n} x_{i,j} \leqslant |S|-1,\ \forall S \subseteq N\setminus\{0\},\ 1\leqslant |S| \leqslant n \tag{7}$$

$$\mathrm{DQ}_0^l = Q^l,\ \forall l \in L \tag{8}$$

$$\mathrm{DT}_0 = 0 \tag{9}$$

$$\mathrm{AQ}_j^l = \sum_{i=0}^{n} x_{i,j} \times \left(\mathrm{sgn}\left(\min_{l\in L}\left\{\mathrm{DQ}_i^l - q_j^l\right\}\right)\times \mathrm{DQ}_i^l + \overline{\mathrm{sgn}}\left(\min_{l\in L}\left\{\mathrm{DQ}_i^l - q_j^l\right\}\right)\times Q^l\right),\ \forall j \in N\setminus\{0\},\ \forall l \in L \tag{10}$$

$$\mathrm{AT}_j = \sum_{i=0}^{n} x_{i,j} \times \left(\mathrm{sgn}\left(\min_{l\in L}\left\{\mathrm{DQ}_i - q_j^l\right\}\right)\times \frac{d_{i,j}}{v} + \overline{\mathrm{sgn}}\left(\min_{l\in L}\left\{\mathrm{DQ}_i^l - q_j^l\right\}\right)\times \frac{d_{i,0}+d_{0,j}}{v}\right),\ \forall j \in N\setminus\{0\} \tag{11}$$

$$\mathrm{DQ}_j^l = \mathrm{AQ}_j^l - q_j^l,\ \forall j \in N\setminus\{0\},\ \forall l \in L \tag{12}$$

$$\mathrm{DT}_j = \mathrm{AT}_j + \Delta t_j,\ \forall j \in N\setminus\{0\} \tag{13}$$

$$x_{i,j} \in \{0, 1\},\ \forall i \in N,\ \forall j \in N \tag{14}$$

Formulas (1) and (2) are the two objective functions, where Formula (1) represents the total time cost of the robot, which is the last to return to the warehouse. Formula (2) represents the sum of the remaining herbicides after all robots complete their tasks. Equations (3) and (4) ensure that exactly one robot goes to and departs from each task point. Equations (5) and (6) guarantee that all the m robots leave and return to the depot after completing the assigned tasks. Formula (7) eliminates subtours. Equalion (8) indicates that each herbicide load of a robot is full when it departs from the depot. Equalion (9) implies that each robot departs from the depot at time 0. In Equations (10) and (11), $\mathrm{sgn}(\,)$ is a function that returns 1 if its input argument $\min_{l\in L}\left\{\mathrm{DQ}_i - q_j^l\right\}$ is larger than or equal to 0; otherwise, it returns 0. $\overline{\mathrm{sgn}}(\,)$ is another function that returns 1 if its input argument $\min_{l\in L}\left\{\mathrm{DQ}_i - q_j^l\right\}$ is less than 0; otherwise, it returns 0. Equations (10) and (11) calculate the arrival time and remaining herbicides for a robot at task point j and ensure that the remaining herbicides meet the demand of task point j. Equations (12) and (13) calculate the departure time and residual herbicide after finishing task point j. Formula (14) states the domains of the decision variables.

The relevant parameters and their definitions are shown in Table 1.

3.3 Example instance

Consider an example with $n=9$, $m=3$, $v=1$ m/s and $Q^l = 20$ dL, $l \in \{1, 2\}$. The travel distance between any two task points is shown in Table 2. The information for each task point is shown in Table 3.

Suppose a solution is obtained with the decision variables $x_{0,2}$, $x_{2,1}$, $x_{1,0}$, $x_{0,4}$, $x_{4,8}$, $x_{8,6}$, $x_{6,0}$, $x_{0,7}$, $x_{7,5}$, $x_{5,9}$, $x_{9,3}$, and $x_{3,0}$ equal to 1, and the others equal to 0, which form the following three roads each for a robot: $0\to2\to1\to0$, $0\to4\to8\to6\to0$, and $0\to7\to5\to9\to3\to0$. For the first, second, and third roads, the residual herbicide and completion time after finishing the last task are equal to 13 dL and 163 s, 7 dL and 163 s, and 14 dL and 342 s, respectively. Notably, in the third road, when the robot finishes its work at task point 5, the remaining herbicide 1 is less than the demand of task point 9. Therefore, the robot must return to the depot to replenish its full load before proceeding to task point 9. Overall, the final values of the objective functions are $U = 34$ dL and $C = 342$ s.

4 Proposed MOTLBO Algorithm

An MOTLBO algorithm is proposed in this section to

Table 1 Parameter list.

Parameter	Definition
i, j	Task point index
$x_{i,j}$	If task point i is an immediate predecessor of task point j, then $x_{i,j}=1$; else $x_{i,j}=0$
n	Number of task points
m	Number of robots
l	Herbicide index
Δt_i (s)	Weeding time required by task point i
$d_{i,j}$ (m)	Travel distance from task points i to j
$t_{i,j}$ (s)	Travel time from task points i to j
AT_j (s)	Arrival time of a robot at task point j
DT_j (s)	Departure time of a robot from task point j
Q^l (dL)	Full load of herbicide l
q_j^l (dL)	Requirement of herbicide l at task point j
AQ_j^l (dL)	Remaining herbicide l when a robot arrives at task point j
DQ_j^l (dL)	Remaining herbicide l when a robot departs from task point j
C (s)	Completion time to finish all tasks
U (dL)	Total residual herbicide after finishing all tasks
v (m/s)	Speed of robot
$L=\{1, 2\}$	Herbicide set
$N=\{0, 1, 2, \ldots, n\}$	Task point set, where 0 is the depot

Table 2 Distance between task points.

(m)

Task point	Task point									
	0	1	2	3	4	5	6	7	8	9
0	0	10	40	47	15	24	48	19	47	19
1	10	0	32	39	17	26	40	21	39	21
2	40	32	0	13	39	30	18	41	25	47
3	47	39	13	0	40	31	11	42	18	48
4	15	17	39	40	0	11	35	14	34	14
5	24	26	30	31	11	0	26	23	25	23
6	48	40	18	11	35	26	0	37	17	43
7	19	21	41	42	14	23	37	0	30	12
8	47	39	25	18	34	25	17	30	0	36
9	19	21	47	48	14	23	43	12	36	0

Table 3 Task information.

Resource	Task point								
	1	2	3	4	5	6	7	8	9
Herbicide 1 (dL)	10	7	6	9	9	2	9	8	5
Herbicide 2 (dL)	9	1	6	1	7	10	3	3	9
Weeding time (s)	57	24	36	30	48	36	36	33	42

solve the MWRTA problem. First, the MOTLBO framework is introduced. The solution representation, heuristic-based initialization, TLBO process, local search, and population updating method are then comprehensively discussed.

4.1 Framework of MOTLBO

The basic idea of the classical TLBO algorithm is derived from the influence of teachers' work on students; that is, the teaching level of teachers will affect the academic performance of students. The process of TLBO is divided into teaching and learning phases. In the teaching phase, students learn from teachers; in the learning phase, students learn from each other. A student group is a population in the TLBO algorithm. The learning score of students corresponds to the fitness value, and the best solution in the population is selected as the teacher. The pseudocode of the proposed MOTLBO algorithm is shown in Algorithm 1.

Algorithm 1 MOTLBO algorithm

```
1 Set algorithmic parameters;
2 Generate an initial population $P$ by the proposed heuristics
  and size of $P$ is set to NP;
3 Set EP = $\varnothing$; // EP is a set that stores nondominated solutions
4 while a termination condition is not met do
5     Identify teacher individual $\{\pi^{\text{Gbest}}, \rho^{\text{Gbest}}\}$;
6     Divide population $P$ into groups with $\Omega$ individuals;
7     for each group do
8         Identify group leader $\{\pi_i^{\text{Gbest}}, \rho_i^{\text{Gbest}}\}$;
9         Calculate group average $\{\pi_i^{\text{mean}}, \rho_i^{\text{mean}}\}$;
10        Execute the teaching and learning operations;
11    end
12    Obtain a new population $P'$;
13    Execute local search to all nondominant solutions in $P'$ to
      obtain $P''$;
14    $P :=$ best NP solutions from $P \bigcup P' \bigcup P''$;
15    Update EP by all nondominated solutions from $P' \bigcup P''$;
16 end
```

4.2 Solution representation

A direct solution representation is used to facilitate the implementation of the algorithm, where a solution is expressed by a task point sequence $\pi = (\pi_1, \pi_2, \ldots, \pi_n)$ and a split point string $\rho = (\rho_1, \rho_2, \ldots, \rho_{m-1})$; that is, $\{(\pi_1, \pi_2, \ldots, \pi_n), (\rho_1, \rho_2, \ldots, \rho_{m-1})\}$, where $\pi_i \in N \setminus \{0\}$ $(i = 1, 2, \ldots, n)$ and ρ_k $(k = 1, 2, \ldots, m-1)$ is a split point. In the task point sequence π, each task point appears exactly once. Meanwhile, in the split point string ρ, $\rho_k < \rho_{k+1}$, $k = 1, 2, \ldots, m-1$. The task point sequence obtained for Robot 1 is $\Pi_1 = (\pi_1, \pi_2, \ldots, \pi_{\rho_1})$, the task point sequence for Robot 2 is $\Pi_2 = (\pi_{\rho_1+1}, \pi_{\rho_1+2}, \ldots, \pi_{\rho_2})$, and so on. The solution of the example instance in Section 3.3 can be represented by {(2, 1, 4, 8, 6, 7, 5, 9, 3), (2, 5)}, where the task point sequence $\pi = (2, 1, 4, 8, 6, 7, 5, 9, 3)$ and the split point string $\rho = (2, 5)$.

4.3 Heuristic-based initialization

An initial population with quality and diversity can continuously facilitate the quick convergence of the algorithm and obtain a good result. An INEH heuristic and a maximum load-based heuristic are proposed to generate initial solutions to obtain a promising initial population.

The NEH heuristic is widely used in the flow shop scheduling problem[31–34]. In the heuristic, a seed sequence of jobs is first generated in accordance with the largest processing time rule. From front to back, the first job in the seed sequence is then taken out and forms a partial solution. Next, the second job is taken out, and a testing process where each possible slot of the partial solution is tested to facilitate job insertion is performed. All the partial solutions obtained are evaluated, and the job is finally inserted into the slot that leads to the best object value. The above process is repeated until all jobs are considered successively and a complete solution is generated.

The NEH heuristic was originally designed for a single-objective problem. That is, only a single objective is evaluated in the testing process. An index function that combines the two objectives considered as follows is defined to solve the multi-objective problem by the NEH heuristic,

$$f = \lambda C + (1-\lambda)U \tag{15}$$

where f represents an index function, and λ is a weight parameter that is used to balance the two objectives considered.

If the index function is used to determine suitable slots for all the n task points, then only one complete solution with a high-quality f value for a given weight λ can be obtained. This condition might miss some nondominated solutions that are beneficial to future exploitation. Therefore, during the testing process of the last task point, the index function was not used again to address this problem. Instead, all the solutions generated are compared by the dominance relationship. All the nondominated solutions enter the initial population.

Weight λ plays an important role in the final nondominated solutions. Hence, weight λ is changed from 0.0 to 1.0 with a step of 0.1 to generate an initial population with a high level of quality and diversity, and the nondominated solutions generated under each λ value enter the initial population. This problem generates m sequences for each robot. All the m sequences are finally combined to form an entire solution that is represented by a task point sequence and a split point string. The pseudocode of the INEH heuristic is given in Algorithm 2.

Algorithm 2 INEH heuristic

```
1 Set $P = \varnothing$; //The initial population is empty
2 Set $\pi = \{1, 2, \ldots, n\}$;
3 Set $\Gamma = \{0.0, 0.1, \ldots, 1.0\}$;
4 for $\Lambda = 1 : |\Gamma|$ do
5     for $k = 1 : m$ do
6         $\Pi_k = \varnothing$; // Task point sequence of robot $k$ is empty
7     end
8     for $j = 1 : n-1$ do
9         Test to insert task point $\pi_j$ into each slot of $\Pi_k$, $k = \{1, 2, \ldots, m\}$;
10        Calculate index function $f$ with $\Gamma_\Lambda$ for each partial solution generated;
11        Insert task point $\pi_j$ into the slot leading to the lowest $f$ value;
12    end
13    Insert task point $\pi_n$ into each slot of $\Pi_k$, $k = \{1, 2, \ldots, m\}$;
14    Calculate objective values $C$ and $U$ for each solution generated;
15    Get nondominant solutions among the solutions;
16    $P = P \bigcup$ {nondominated solutions};
17 end
  Output: $P$
```

After the INEH heuristic, the maximum load-based heuristic is used to yield the remaining solutions if the initial population is not full. In the maximum load-based heuristic, a sequence π of the n task points is first randomly generated, and the robot task point sequence $\Pi_k = \varnothing$, $k = \{1, 2, \ldots, m\}$, is set. Then, let $k = 1$, and a solution is constructed by scanning the unassigned task points in sequence π from front to back. If the demand of the current task point for each herbicide does not exceed the remaining load of robot k, then the current task point is appended to the robot sequence Π_k; otherwise, let robot k return to the depot, and robot $k+1$ is considered. After all the m robots are considered, if unassigned task points still exist, then the above process will be repeated by assuming that all robots start from the depot with the maximum load again. The maximum load-based heuristic is repeated until the initial population is full. The sequences of all task points are randomly generated. Therefore, each herbicide is maximized, and the maximum load-based heuristic can also obtain a population of solutions with a high level of quality and diversity. The pseudocode of the maximum load-based heuristic is shown in Algorithm 3.

4.4 TLBO process

4.4.1 Grouping mechanism

In the TLBO algorithm, a teacher teaches students the knowledge to help them improve their performance, thereby enhancing the knowledge level of the entire class[35]. The update of the students is based on the average knowledge level of the entire class and their teacher. This method helps students quickly approach the knowledge level of their teacher, and the initiative of students is minimally utilized. Thus, the population or class easily loses its diversity, and the algorithm directly falls into a local optimum. Inspired by the group teaching method that is becoming increasingly

Algorithm 3 Maximum load-based heuristic

```
1 Generate a random sequence $\pi = (\pi_1, \pi_2, \dots, \pi_n)$;
2 Set $\Pi_k = \varnothing$, $k = 1, 2, \dots, m$;
3 Set $i = 1$;
4 while $i \leqslant n$ do
5     $Q_k^1 = Q^1$ and $Q_k^2 = Q^2$;
6     $k = 1$;
7     while $k \leqslant m$ do
8         if $Q_k^1 - q_{\pi_i}^1 \geqslant 0$ and $Q_k^2 - q_{\pi_i}^2 \geqslant 0$ then
9             $\Pi_k = \Pi_k \cup \pi_i$;
10            for $l = 1 : 2$ do
11                $Q_k^l = Q_k^l - q_{\pi_i}^l$;
12            end
13            $i = i + 1$;
14        else
15            $k = k + 1$;
16        end
17    end
18 end
   Output: Solution formed by $\Pi_k$, $k = 1, 2, \dots, m$
```

popular in seminar classes, a dynamic grouping mechanism where the entire population is partitioned into several groups is introduced to address the problem. In each group, the students learn not only from the teacher but also from the group leader and group average. After this iteration, the students in all the groups are combined into a whole population, and then the population is divided again.

4.4.2 Selection for teacher, group leader, and group average

The teacher (denoted by $\{\pi^{\text{Gbest}}, \rho^{\text{Gbest}}\}$), group leader (denoted by $\{\pi^{\text{Lbest}}, \rho^{\text{Lbest}}\}$), and group average are determined in this section. Following the original TLBO, the first nondominant solution in the population and the group after the nondominated sorting are set as the teacher and the group leader, respectively. The group average is a virtual solution generated by all the solutions in the group. Let $\{\pi^{\text{mean}}, \rho^{\text{mean}}\}$ denote the group average, where $\pi^{\text{mean}} = \left\{\pi_1^{\text{mean}}, \pi_2^{\text{mean}}, \dots, \pi_n^{\text{mean}}\right\}$ and $\rho^{\text{mean}} = \left\{\rho_1^{\text{mean}}, \rho_2^{\text{mean}}, \dots, \rho_{m-1}^{\text{mean}}\right\}$. Let $\{\pi^{\omega}, \rho^{\omega}\}$ represent individual ω in the group, where $\pi^{\omega} = \left\{\pi_1^{\omega}, \pi_2^{\omega}, \dots, \pi_n^{\omega}\right\}$, $\rho^{\omega} = \left\{\rho_1^{\omega}, \rho_2^{\omega}, \dots, \rho_{m-1}^{\omega}\right\}$, and $\omega = 1, 2, \dots, \Omega$, and Ω is the group size. The group average is then generated by

$$\pi_i^{\text{mean}} = \left\lceil \sum_{\omega=1}^{\Omega} \frac{\pi_i^{\omega}}{\Omega} \right\rceil, \; i = 1, 2, \dots, n \tag{16}$$

$$\rho_i^{\text{mean}} = \left\lceil \sum_{\omega=1}^{\Omega} \frac{\rho_i^{\omega}}{\Omega} \right\rceil, \; i = 1, 2, \dots, m-1 \tag{17}$$

where $\lceil\,\rceil$ is a function that rounds its elements to the nearest integers toward infinity.

The above formulations can generate feasible split point string ρ_i^{mean} because $\rho_i^{\omega} < \rho_{i+1}^{\omega}$, $i = 1, 2, \dots, m-2$ for each solution ω. However, the task point sequence π_i^{mean} is different; in this sequence, some task points may repeat numerous times, and some task points may disappear. Thus, a repair operator is presented. The repair operator scans the task point sequence π_i^{mean} from front to back and marks the task points that appear more than once. The last repetition of each marked task point is maintained, and the other repetitions are replaced by zeros. Then, from front to back, all zeros in the task point sequence π_i^{mean} are replaced one by one by the lost task points in the increasing order of their indexes. An example of the repair operator is given in Fig. 2.

The task point sequence $\pi_i^{\text{mean}} = \{1, 2, 2, 6, 3, 3\}$ shown in Fig. 2 is an unfeasible solution. A total of 6 task points are available. Task points 2 and 3 appear twice. Task points 1 and 6 appear once. Task points 4 and 5 do not appear. Therefore, the repair operator is performed on these task points. First, task points 2 and 3 are marked. Then, the first "2" and "3" are replaced with zeros. Finally, a feasible $\pi_i^{\text{mean}} = \{1, 4, 2, 6, 5, 3\}$ can be obtained by replacing the first and second 0 with task points 4 and 5, respectively.

4.4.3 "Teaching" phase

In a group, a student can learn from the teacher $\{\pi^{\text{Gbest}}, \rho^{\text{Gbest}}\}$, the group leader $\{\pi^{\text{Lbest}}, \rho^{\text{Lbest}}\}$, and the group's average $\{\pi^{\text{mean}}, \rho^{\text{mean}}\}$. Four different learning operations, $\{\pi^1, \rho^1\} \otimes \{\pi^{\text{mean}}, \rho^{\text{mean}}\}$, $\{\pi^2, \rho^2\} \otimes \{\pi^{\text{Gbest}}, \rho^{\text{Gbest}}\}$, $\{\pi^3, \rho^3\} \otimes \{\pi^{\text{Lbest}}, \rho^{\text{Lbest}}\}$, and $\{\pi^4, \rho^4\} \otimes \{\pi^{\text{mean}}, \rho^{\text{mean}}\}$, are designed to generate new solutions, where $\{\pi^{\omega}, \rho^{\omega}\}$, $\omega = 1, 2, 3, 4$, is a solution in the group (that is, the group size is fixed at $\Omega = 4$), and the symbol "$\otimes$" denotes a learning operator. Notably, the

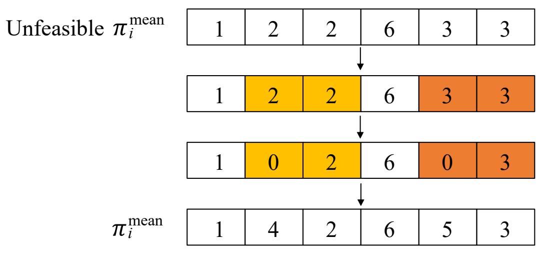


Fig. 2 Example of the repair operator.

solutions in the group are already sorted by the nondominated sorting method. Thus, $\{\pi^{\text{Lbest}}, \rho^{\text{Lbest}}\} = \{\pi^1, \rho^1\}$ is obtained.

The original TLBO algorithm is designed for continuous problems. However, the problem is discrete; thus, the learning operator in the original TLBO cannot be used. Inspired by the crossover operators in evolutionary algorithms, the learning operator is defined as a crossover; that is, the solutions before and after $\otimes$ are crossed with a crossover probability P_c to generate two children. Hence, a random number is generated in the range of [0, 1]. If the random number is less than the crossover probability P_c, then the crossover operator is performed; otherwise, the original solutions are maintained. In the crossover operator, the superior styles representing the superior genes of the two solutions are preserved to the greatest extent possible. As shown in Fig. 3, two crossover points are randomly generated, namely the starting and ending points, for the task point sequences π^1 and π^{mean}, respectively. The genes between the two crossover points from π^1 are copied to the front position of Child 1, and then the missing genes in Child 1 are inserted sequentially from π^{mean}. Another child is obtained similarly. Starting and ending points are also randomly selected to generate a split point string. The gene regions between the two points from the two split point strings are then swapped to produce offspring. If the split point string ρ of a child is infeasible or $\rho_k \geqslant \rho_{k+1}$, $k \in \{1, 2, \ldots, m-2\}$, then a split point string is randomly selected from its parent as the split point string of the child.

After the crossover operator is completed, a mutation operator is performed on all the children with a mutation probability P_m. For simplicity, this study uses swap mutation, where two task points in the task sequence are randomly chosen and their positions are exchanged.

After executing the crossover and mutation

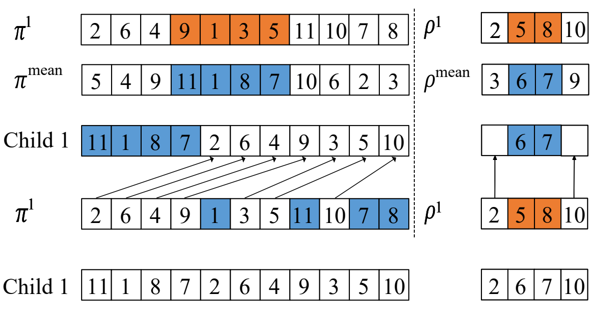


Fig. 3　Example of the crossover operator.

operators, the new solutions are merged with the original solutions, and then they are sorted using the nondominated sorting method. The group is refilled by the best four solutions, denoted by $\{\pi'^1, \rho'^1\}$, $\{\pi'^2, \rho'^2\}$, $\{\pi'^3, \rho'^3\}$, and $\{\pi'^4, \rho'^4\}$, and the other solutions are discarded.

4.4.4　"Learning" phase

In the "learning" phase of the original TLBO, the students perform updating operations according to the differences between the current and randomly selected solutions. The solutions in the group survive the competition in the teaching phase; thus, they should carry the good information implied by different superior styles. Learning from each other is also beneficial. Therefore, the learning operations $\{\pi'^1, \rho'^1\}\otimes\{\pi'^2, \rho'^2\}$ and $\{\pi'^3, \rho'^3\}\otimes\{\pi'^4, \rho'^4\}$ are performed with a crossover probability P_c to generate four new solutions. Afterward, the swap mutation is performed on each new solution with a mutation probability P_m. The solutions generated are then merged with their parent solutions, and the solutions merged using the nondominated sorting method are sorted. The group is finally refilled with the four best solutions.

4.5　Local search method

A high-quality solution is usually obtained by an effective neighborhood search operator, and multiple neighborhood search operators with different interference strategies often have a high probability of obtaining high-quality solutions[36, 37]. Therefore, a local search based on four neighborhood operators is designed. The four neighborhood search operators are as follows.

(1) Task exchange within a robot

A robot task sequence Π_k, $k \in \{1, 2, \ldots, m\}$, is randomly chosen from the current solution π. Two task points, namely $\Pi_{k,i}$ and $\Pi_{k,j}$ are then randomly selected in the robot task sequence Π_k, and their positions are exchanged, resulting in a new solution π'. For simplicity, the operator is denoted as NS_1.

(2) Task exchange between two robots

Two robot task sequences, namely Π_k and Π'_k, $k, k' \in \{1, 2, \ldots, m\}$, are randomly chosen from the current solution π. A task point $\Pi_{k,i}$ from the robot task sequence Π_k and a task point $\Pi_{k',j}$ from the robot task sequence Π'_k are then randomly selected, and their positions are exchanged, resulting in a new solution π'. The operator is denoted as NS_2.

(3) Task shift within a robot

A robot task sequence Π_k, $k \in \{1, 2, \ldots, m\}$, is randomly chosen from the current solution π. A task point $\Pi_{k,i}$, a position i' in the robot task sequence Π_k $(i \neq i')$ are also randomly selected. Task point $\Pi_{k,i}$ is shifted to position i', resulting in a new solution π'. The operator is denoted as NS_3.

(4) Task shift between two robots

Two robot task sequences, namely Π_k and Π'_k, k, $k' \in \{1, 2, \ldots, m\}$, are randomly chosen from the current solution π. A task point $\Pi_{k,i}$ from the robot task sequence Π_k and a position i' in the robot task sequence Π'_k are also randomly selected. Task point $\Pi_{k,i}$ is taken out from the robot task sequences Π_k and then inserted into the position i' in the robot task sequence Π'_k, resulting in a new solution π'. The operator is denoted as NS_4.

Only some solutions generated by the TLBO process use the local search method to balance the local and global exploitations of the proposed algorithm effectively. Instead, the local search method is conducted for the nondominated solutions in the new population P'. The nondominated solutions survive in evolution and carry better information than the others in the population. This method is expected to obtain numerous high-quality solutions in their neighborhoods. Specifically, in the local search method, the four neighborhood search operators are performed in order for each nondominated solution in the population P'. For each of the four neighborhood search operators, Ψ times are conducted to the current solution, where Ψ is an algorithmic parameter that is calibrated in Section 5. If a solution that dominates the current solution is obtained in the process, then the procedure will be stopped; otherwise, the next neighborhood search operator is performed. All the solutions generated in the process are put in a solution set P''. The nondominated solutions in the set P'' are selected as the output of the local search method. The pseudocode of the local search operator is shown in Algorithm 4.

In the local search algorithm, the method shown in Section 3 can be used to calculate the completion time and residual herbicide for each robot to evaluate the generated solution, and the maximum completion time and overall residual herbicide can then be obtained. This method will lead to a slow local search when the parameter Ψ is large. Close observation reveals that the proposed neighborhood search operators only slightly change the current solution. In other words, the solution generated by a neighborhood search operator is highly similar to the current solution. Therefore, this similarity can be maximized to save computational time. For example, NS_1 and NS_3 only change a task point sequence of a robot, but the task point sequences of the other robots remain unchanged. Thus, the completion time and residual herbicide of an unchanged robot need not be calculated again when evaluating the newly generated solution. In this way, the computational time of the local search decreases heavily.

Algorithm 4 Local search algorithm

```
1 $A = \{A_1, A_2, \ldots, A_{|A|}\}$; //Set of all nondominant solutions in $P'$
2 Set $P'' = \varnothing$; //Set an empty set
3 for $I = 1 : |A|$ do
4     Put $A_I$ in $P''$;
5     for $J = 1 : 4$ do
6         Execute operator $\mathrm{NS}_J$ to $A_I$ for $\Psi$ times;
7         Put all solutions generated in $P''$;
8         if there is a solution generated better than $A_I$ then
9             break;
10        end
11    end
12 end
  Output: all nondominated solutions in $P''$
```

4.6 Population updating

First, the original population P and the population P'' generated in the TLBO process, and the solutions generated by the local search are combined to form a temporary population. The fast nondominated sorting algorithm is then applied to the solutions in the temporary population. The best NP different individuals from the temporary population are selected as the population P for the next iteration. Therefore, the population contains NP different high-quality individuals, which will steadily evolve into a promising region. Notably, in the MOTLBO algorithm, an external population EP is also maintained to store the identified nondominated solutions, which are updated by the nondominated solutions in the temporary population in each iteration.

5 Experimental Comparison and Analysis

5.1 Setup of experiment

A total of 120 examples are generated for experimental analysis according to a real farm to evaluate the performance of the MOTLBO algorithm. These examples are divided into two sets: test and calibration sets. The test and calibration sets contain 96 and 24 examples, respectively. The test set is used to assess the performance of all comparison algorithms, and the calibration set is used to calibrate the parameters of all comparison algorithms. The map size is set to a 50×50 raster map. The number of task points is set as $n=\{30, 40, 50, 60, 70, 80\}$, and the number of agricultural robots $m=\{3, 4, 5, 6\}$. The herbicide requirement of each task point is randomly distributed in the range of [0, 10].

The Nondominated Sorting Genetic Algorithm II (NSGA-II)[38], Multi-Objective Evolutionary Algorithm based on Decomposition (MOEA/D)[39], Multi-Objective Greedy Algorithm (MOGA)[40], and Multi-Objective Particle Swarm Optimization algorithm (MOPSO)[41], which are the most advanced algorithms and have outstanding performance in relevant research, are selected as comparison algorithms. All comparison algorithms are encoded and decoded using the solution expression method proposed in this paper for fairness. The termination conditions of all algorithms comply with the requirements of this paper. All algorithms share the common codes of the proposed algorithm, such as the calculation of the target value and the judgment of the constraint conditions. The parameters of all algorithms are recalibrated in accordance with the problems.

All algorithms are compiled using the MATLAB 2016b platform. The operating system is Microsoft Windows 10. The PC environment is an Intel (R) Xeon (R) Silver 4210R CPU, 2.40 GHz, and 128 GB RAM. All comparison algorithms adopt the same termination condition, that is, the running time of the algorithm $t_{\max}=n\times m$ seconds.

5.2 Performance indicators

Two performance indexes are used to evaluate the performance of all comparison algorithms comprehensively. These indexes are the Inverse Generation Distance (IGD)[42, 43] and Hypervolume (HV)[44, 45]. These indexes were chosen because they are comprehensive performance indexes considering the convergence and distribution of solution sets.

HV refers to the volume of the region in the target space enclosed by the nondominated solution set and the reference point obtained after running the algorithm. A large HV value leads to the superior comprehensive performance of the algorithm[46, 47]. The reference point in this paper is set to (1, 1), and the nondominant solution set obtained by an algorithm is denoted as EP. The HV value of EP can be obtained by the following formula:

$$\mathrm{HV}\,(\mathrm{EP})=\delta\left(\bigcup_{i=x}^{|\mathrm{EP}|} v_x\right) \tag{18}$$

where δ represents the Lebesgue measure used to calculate the volume, $|\mathrm{EP}|$ represents the number of nondominated solutions in the set EP, and v_x represents the super volume formed by the reference point and the x-th solution in the set.

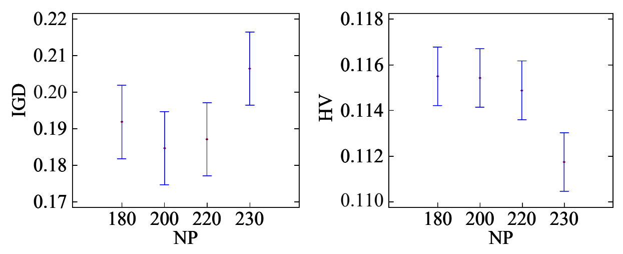
IGD calculates the average distance between the individuals in the Pareto optimal solution set EP^* and the nondominant solution set EP obtained by an algorithm. An IGD value leads to the superior comprehensive performance of the algorithm[48]. The IGD value can be obtained using the following formula:

$$\mathrm{IGD}(\mathrm{EP}, \mathrm{EP}^*)=\frac{\sum\limits_{x\in \mathrm{EP}^*} d\,(x, \mathrm{EP})}{|\mathrm{EP}^*|} \tag{19}$$

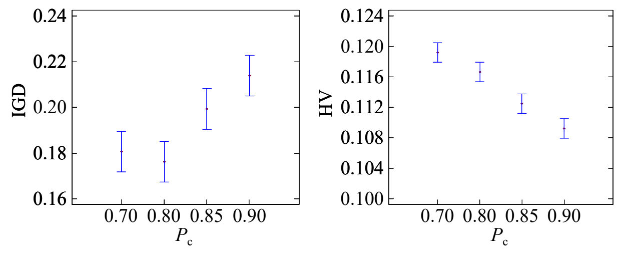
where $d\,(x, \mathrm{EP})$ represents the Euclidean distance between point x in the solution set EP^* and point y in the solution set EP.

5.3 Algorithm calibration

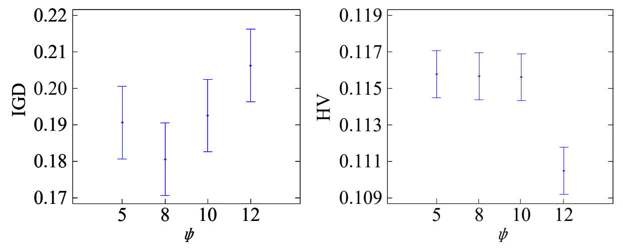
The parameters of the MOTLBO algorithm and comparison algorithms are calibrated. The MOTLBO algorithm has four parameters: population size (NP), local search operator execution times $\varPsi$, crossover probability ($P_{\rm c}$), and mutation probability ($P_{\rm m}$). After the primary experiments, four levels are set for NP (180, 200, 220, and 230), $\varPsi$ (5, 8, 10, and 12), and $P_{\rm c}$ (0.7, 0.8, 0.85, and 0.9). Changing $P_{\rm m}$ in the range of [0, 0.5] does not lead to significant results. Thus, $P_{\rm m}=0.2$ is set to save the running time of the calibration procedure. The above parameter levels produce a total of $4\times 4\times 4=64$ parameter combinations. The MOTLBO algorithm is performed with each parameter combination, and three independent repetitions are conducted on each of the 24 calibration instances, leading to $64\times 3\times 24=4608$ results in total. The mean plots with a 95% confidence interval considering IGD and HV for the three parameters are shown in Fig. 4.



(a) NP calibration result



(b) P_c calibrationresult



(c) Ψ calibration result

Fig. 4 Means plots of the parameters in MOTLBO.

Figure 4 shows that levels 180, 200, and 220 of parameter NP generate significantly better results than level 230, considering IGD and HV. However, the confidence intervals for NP = 180, NP = 200, and NP = 220 overlap, indicating that NP does not lead to significantly different results at the three levels. NP = 200 produces slightly better results than the others; thus, NP = 200 is chosen. Similarly, 0.8 and 8 are selected for P_c and Ψ, respectively. Therefore, the final parameter combination for the MOTLBO algorithm is NP = 200, $P_c = 0.8$, and $\Psi = 8$. In addition, the parameters of the comparison algorithms are calibrated similarly. The calibrated parameters for the comparison algorithms are shown in Table 4.

Table 4 Calibration result of contrast algorithms.

Algorithm	Parameter
MOEA/D	NP = 220, $P_c = 0.8$, $P_m = 0.2$
NSGA-II	NP = 200, $P_c = 0.9$, $P_m = 0.2$
MOPSO	NP = 30, $r_1 = 0.8$, $r_2 = 0.7$
MOGA	NP = 50, $d = 5$

Note: r_1 and r_2 denote individual and group learning factors, respectively; d denotes deconstruction scale.

5.4 Evaluation of the components of MOTLBO

Four algorithms, namely MOTLBO$_1$, MOTLBO$_2$, MOTLBO$_3$, and MOTLBO$_4$, are designed. MOTLBO$_1$ is the same as MOTLBO, except that INEH is not used in the initialization. MOTLBO$_2$ is also similar to MOTLBO, except that heuristic-based initialization is not used to generate initial solutions. MOTLBO$_2$ randomly generates the initial solutions. MOTLBO$_3$ is MOTLBO without the local search method, and the MOTLBO$_4$ algorithm is MOTLBO without a grouping strategy. The five aforementioned algorithms are run to solve the examples from the test set. Each example was run independently five times. Figure 5 shows the mean plots considering IGD and HV obtained under 95% confidence intervals.

Figure 5 shows that MOTLBO performs slightly better than its variants regardless of HV or IGD, which indicates that the proposed components in MOTLBO are effective.

5.5 Comparison with existing algorithms

MOTLBO is compared with NSGA-II, MOEA/D, MOGA, and MOPSO. Five replications are conducted for all the algorithms for each of the test instances. The average IGD and HV values on the instances with the same n and m obtained by each algorithm are given in Table 5.

As shown in Table 5, considering the IGD metric, MOTLBO generates an overall average IGD value equal to 0.1412. This value is slightly better than that yielded by MOEA/D (0.1830) and those obtained by NSGA-II (0.3654), MOGA (0.6475), and MOPSO (0.7743). Out of the 24 instance sizes, MOTLBO generates the 17 best IGD results, while MOEA/D achieves the 7 best IGD results. None of the best IGD values are generated by NSGA-II, MOGA, or MOPSO. Considering the HV metric, MOTLBO generates an overall average HV value equal to 0.1399, which is

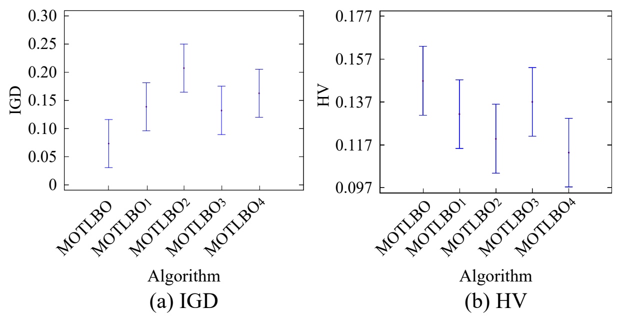


Fig. 5 Mean plots of different MOTLBO variants.

Table 5 Average IGD and HV values.

Example's scale	IGD					HV				
$n\times m$	MOTLBO	NSGA-II	MOEA/D	MOGA	MOPSO	MOTLBO	NSGA-II	MOEA/D	MOGA	MOPSO
30×3	0.1029	0.1311	0.1482	0.4023	0.4130	0.1248	0.1067	0.0807	0.0004	0.0010
30×4	0.1486	0.3185	0.2064	0.6844	0.5590	0.0928	0.0254	0.0600	0	0
30×5	0.1208	0.2681	0.1703	0.7561	0.6292	0.0581	0.0310	0.0435	0	0
30×6	0.7210	1.1950	1.3850	5.4032	5.1316	0.0320	0.0173	0.0161	0	0
Mean	0.2733	0.4782	0.4775	1.8115	1.6832	0.0769	0.0451	0.0501	0.0001	0.0003
40×3	0.0929	0.1461	0.0807	0.1635	0.2175	0.2605	0.1606	0.2429	0.1337	0.0944
40×4	0.0895	0.1762	0.0822	0.3709	0.4375	0.1601	0.0794	0.1469	0.0044	0.0010
40×5	0.0979	0.1835	0.1064	0.4613	0.4443	0.1809	0.0983	0.1587	0.0039	0.0012
40×6	0.1108	0.2577	0.1113	0.5621	0.5419	0.1175	0.0258	0.1130	0	0.0018
Mean	0.0978	0.1909	0.0952	0.3895	0.4103	0.1798	0.0910	0.1654	0.0355	0.0246
50×3	0.1485	0.3651	0.1531	0.3456	0.6110	0.1065	0.0138	0.0893	0.0173	0
50×4	0.1303	0.2936	0.0939	0.3343	0.5134	0.1075	0.0264	0.1067	0.0108	0
50×5	0.1004	0.3019	0.0829	0.5325	0.6398	0.1587	0.0326	0.1498	0	0
50×6	0.1050	0.3425	0.1483	0.5217	0.5048	0.1263	0.0088	0.0742	0	0
Mean	0.1211	0.3258	0.1196	0.4335	0.5673	0.1248	0.0204	0.1050	0.0070	0
60×3	0.1704	0.6338	0.2613	0.6157	1.1265	0.1110	0.0053	0.0308	0	0
60×4	0.2893	1.0345	0.1833	0.9512	1.7107	0.0573	0	0.0425	0	0
60×5	0.0733	0.1519	0.0761	0.2209	0.2745	0.1916	0.0867	0.1778	0.0329	0.0101
60×6	0.0586	0.2231	0.1382	0.5464	0.5480	0.2058	0.0256	0.0758	0	0
Mean	0.1479	0.5108	0.1647	0.5836	0.9149	0.1414	0.0294	0.0817	0.0082	0.0025
70×3	0.0830	0.2711	0.0844	0.1351	0.4384	0.1429	0.0201	0.1241	0.0777	0
70×4	0.1007	0.3448	0.1414	0.3089	0.4757	0.1478	0.0089	0.0865	0.0126	0
70×5	0.0617	0.1793	0.0686	0.2113	0.2638	0.2032	0.0517	0.1555	0.0172	0.0040
70×6	0.0886	0.1428	0.0713	0.2077	0.2265	0.2513	0.1272	0.2217	0.0424	0.0362
Mean	0.0835	0.2345	0.0914	0.2158	0.3511	0.1863	0.0520	0.1470	0.0375	0.0101
80×3	0.1458	0.5948	0.1951	0.4104	0.9825	0.1198	0.0052	0.0695	0.0183	0
80×4	0.0872	0.4090	0.1452	0.4053	0.6721	0.1512	0.0062	0.0804	0.0018	0
80×5	0.1803	0.5990	0.1775	0.6983	0.9051	0.0988	0.0002	0.0548	0	0
80×6	0.0820	0.2054	0.0793	0.2895	0.3150	0.1505	0.0278	0.1253	0.0018	0.0003
Mean	0.1238	0.4521	0.1493	0.4509	0.7187	0.1301	0.0099	0.0825	0.0055	0.0001
Overall	0.1412	0.3654	0.1830	0.6475	0.7743	0.1399	0.0413	0.1053	0.0156	0.0063

Note: Overall is the mean of all examples in this column.

slightly better than those yielded by MOEA/D (0.1053), NSGA-II (0.0413), MOGA (0.0156), and MOPSO (0.0063). MOTLBO generates the best HV values for all 24 instance sizes, and none of the best HV values are obtained by the other comparison algorithms. Therefore, compared with NSGA-II, MOEA/D, MOGA, and MOPSO, MOTLBO shows significant advantages in solving examples of various scales.

Figure 6 shows the means plots considering the IGD and HV values obtained by the five competitive algorithms under 95% confidence intervals. Regardless of the IGD or HV index, the figure reveals that the confidence intervals of MOTLBO and NSGA-II, MOGA, and MOPSO do not overlap, indicating the presence of significant differences between MOTLBO and NSGA-II, MOGA, and MOPSO. The confidence intervals of MOTLBO and MOEA/D overlap considering the IGD index, but their confidence intervals at the HV index do not completely overlap. Thus, MOTLBO produces superior results. Therefore, MOTLBO is statistically superior to the four other comparison algorithms.

The superiority of MOTLBO to other algorithms lies

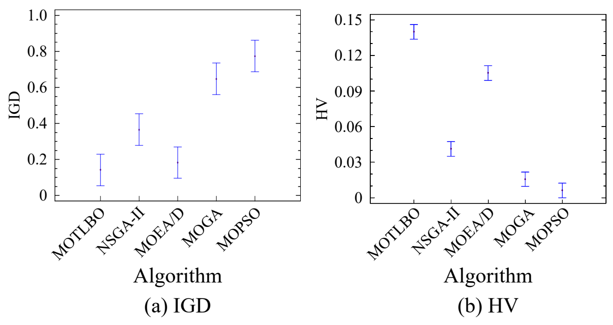


Fig. 6 Mean plots of all comparison algorithms.

in the design of its various parts. First, INEH and the maximum load-based heuristic enable MOTLBO to obtain relatively excellent task point sequences and split points from the initialization stage. Second, the introduction of the group mechanism and the local search strategy, which includes four kinds of neighborhood search operators, substantially expand the search capability of MOTLBO.

Figure 7 shows the interaction graph of the comparison algorithms with the number of task points (n) and the comparison algorithms with the number of robots (m) under the IGD and HV indexes, respectively. At the IGD index, the performance of MOTLBO is similar to MOEA/D. However, with the increase in the number of task points and the number of robots, MOTLBO has additional advantages. Moreover, compared with the other algorithms, MOTLBO obtains smooth IGD values, thereby demonstrating its stable performance. At the HV index, the difference between all comparison algorithms is significant, and the performance of MOTLBO is the best among all the comparison algorithms regardless of the instance sizes. This finding is mainly attributed to the desirable neighborhood for MOTLBO due to the grouping mechanism and local search strategy. The boxplots of different robot scales are shown in Fig. 8. The boxplots directly reflect the stability of the algorithms[49–51]. As shown in Fig. 8, the length of the box for MOTLBO is shorter than that of the other algorithms. Therefore, the results of MOTLBO are highly concentrated. The Friedman test is then utilized to conduct a statistical comparison[52]. The results are shown in Tables 6 and 7 and Fig. 9, where CN is the number of cases, CD is critical difference, and α is significance level. IGD is used to evaluate all the compared algorithms. The solid and dotted lines in Fig. 9 are the critical difference at 95% and 90% confidence intervals, respectively. As illustrated in

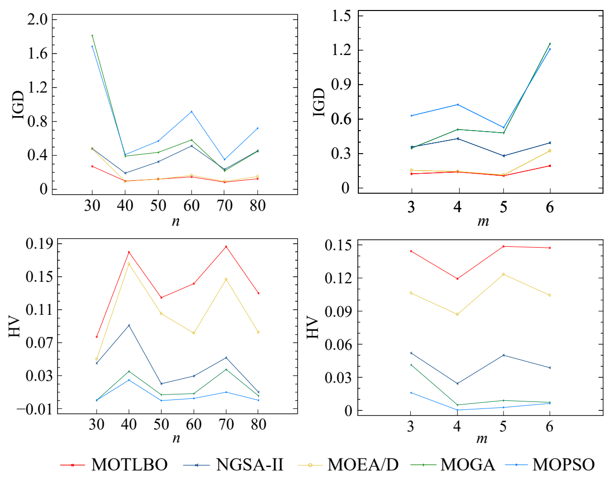


Fig. 7 Interaction graph between comparison algorithms and example size.

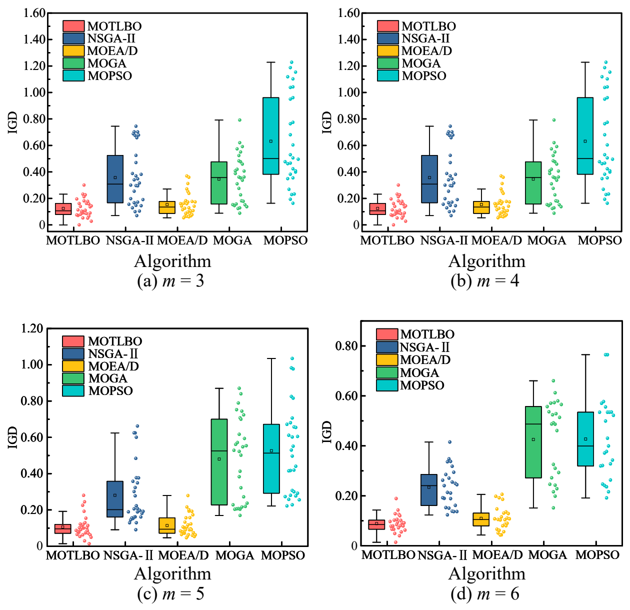


Fig. 8 Boxplots of IGD for all algorithms.

Fig. 9, MOTLBO has the best ranking among the five algorithms.

The above experiments reveal that the results of MOTLBO are significantly better than those of the comparison algorithms. Specifically, MOPSO and MOGA have poor solving accuracy due to their poor local searching capabilities. However, MOTLBO has a grouping mechanism and local search strategy, including four kinds of neighborhood structure, leading to a highly effective algorithm. NSGA-II and MOEA/D combine the decomposition method with neighborhood search for continuous iterative optimization. The two algorithms mainly focus on improving the local search capability. Thus, both algorithms are suitable for

Table 6 Average rank calculation results achieved by Friedman test at different robots (IGD).

Algorithm	Number of robots (m)			
	3	4	5	6
MOTLBO	1.47	1.43	1.47	1.36
NSGA-II	3.37	3.33	3.00	3.00
MOEA/D	1.70	1.57	1.53	1.64
MOGA	3.50	3.83	4.27	4.44
MOPSO	4.97	4.83	4.73	4.56

Table 7 Critieal difference and *p*-value calculation results achieved by Friedman test at different robots.

Indicator	Number of robots (m)			
	3	4	5	6
CN	30	30	30	30
p-value	1.24×10^{-20}	1.31×10^{-21}	1.01×10^{-22}	1.05×10^{-18}
CD ($\alpha = 0.05$)	1.020	1.020	1.020	1.020
CD ($\alpha = 0.10$)	0.915	0.915	0.915	0.915

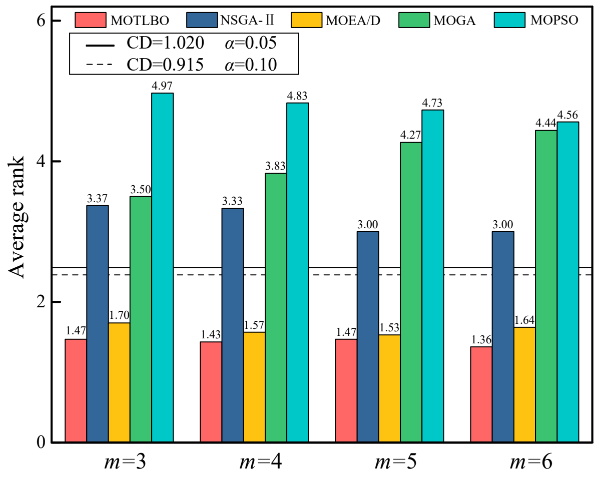


Fig. 9 Friedman test at different metrics.

solving small-scale benchmark instances. However, as the problem size increases, they do not perform as well as MOTLBO due to a lack of superior global search.

Figure 10 illustrates the Pareto frontier diagrams obtained by all the competitive algorithms for six typical examples. The figure reveals that the Pareto frontiers generated by MOTLBO are substantially close to the lower left corner of the figure. This closeness indicates that most of the solutions generated by MOTLBO dominate those generated by the other comparison algorithms.

6 Conclusion and Future Research

Multirobot task allocation has become a popular research topic in the past few years. However, its application in the field of agricultural robots is limited. The intelligent weeding robot is taken in this study as the application object, and an MWRTA problem is presented. A multi-objective mixed integer programming model, which took the maximum completion time and herbicide residual as the optimization objectives, was established. Combined with the structural and objective characteristics of MWRTA, an MOTLBO algorithm is proposed. Some advanced technologies, including an improved NEH and a maximum load-based heuristic, a grouping mechanism, a crossover-and-mutation-based individual updating rule, and a multi-neighborhood-based local search method, are introduced. MOTLBO is calibrated using the design of experiments and analyses of variance. Afterward, MOTLBO is compared against the state-of-the-art algorithms, including the NSGA-II, MOEA/D, MOGA, and MOPSO algorithms from the recent literature. Considering the IGD and HV performance indexes for solving MWRTA, the experimental results showed that MOTLBO generated significantly better results than its competitors.

Future research will continue to focus on the other MWRTA problems and attempt to find problem-

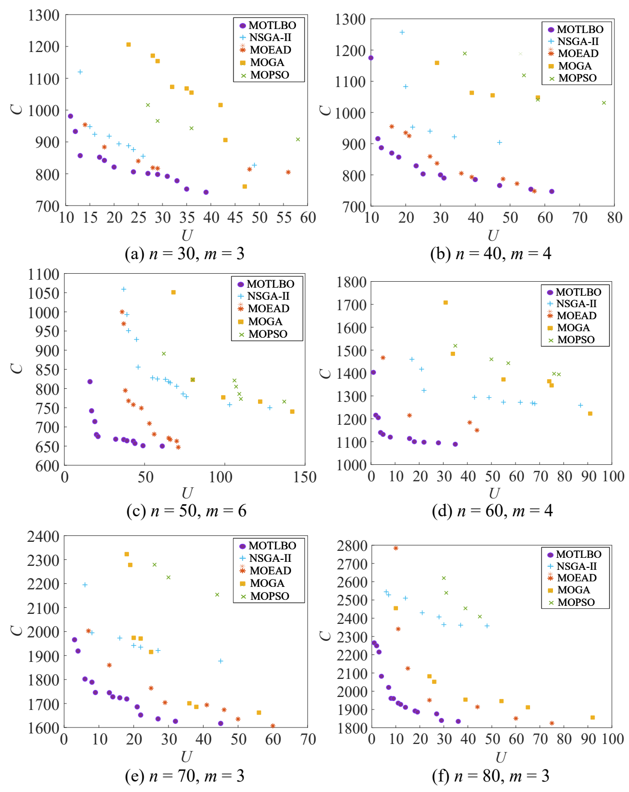


Fig. 10 Pareto frontier.

specific characteristics for these problems. MOTLBO will also be employed to solve other existing task assignments in the agriculture production process.

Acknowledgment

This research was partially supported by the National Natural Science Foundation of China (Nos. 62273221 and 61973203), the Program of Shanghai Academic/Technology Research Leader (No. 21XD1401000), and the Shanghai Key Laboratory of Power Station Automation Technology.

References

[1] A. Walter, R. Finger, R. Huber, and N. Buchmann, Smart farming is key to developing sustainable agriculture, *Proc. Natl. Acad. Sci. U. S. A.*, vol. 114, no. 24, pp. 6148–6150, 2017.

[2] I. Charania and X. Li, Smart farming: Agriculture's shift from a labor intensive to technology native industry, *Internet Things*, vol. 9, p. 100142, 2020.

[3] V. O. Abegunde and A. Obi, The role and perspective of climate smart agriculture in Africa: A scientific review, *Sustainability*, vol. 14, no. 4, p. 2317, 2022.

[4] X. Chen, P. Zhang, G. Du, and F. Li, A distributed method for dynamic multi-robot task allocation problems with critical time constraints, *Robot. Auton. Syst.*, vol. 118, pp. 31–46, 2019.

[5] X. Ji, H. Ye, J. Zhou, Y. Yin, and X. Shen, An improved teaching-learning-based optimization algorithm and its application to a combinatorial optimization problem in foundry industry, *Appl. Soft Comput.*, vol. 57, pp. 504–516, 2017.

[6] R. V. Rao and V. Patel, An improved teaching-learning-based optimization algorithm for solving unconstrained optimization problems, *Sci. Iran.*, vol. 20, no. 3, pp. 710–720, 2013.

[7] A. Baykasoğlu, A. Hamzadayi, and S. Y. Köse, Testing the performance of teaching-learning based optimization (TLBO) algorithm on combinatorial problems: Flow shop and job shop scheduling cases, *Inf. Sci.*, vol. 276, pp. 204–218, 2014.

[8] Y. Xu, L. Wang, S. Y. Wang, and M. Liu, An effective teaching–learning-based optimization algorithm for the flexible job-shop scheduling problem with fuzzy processing time, *Neurocomputing*, vol. 148, pp. 260–268, 2015.

[9] L. R. Rodrigues and J. P. P. Gomes, TLBO with variable weights applied to shop scheduling problems, *CAAI Trans. Intell. Technol.*, vol. 4, no. 3, pp. 148–158, 2019.

[10] R. Yuan, J. Li, X. Wang, and L. He, Multirobot task allocation in e-commerce robotic mobile fulfillment systems, *Math. Probl. Eng.*, vol. 2021, p. 6308950, 2021.

[11] D. H. Lee, S. A. Zaheer, J. H. Han, J. H. Kim, and E. Matson, Competency adjustment and workload balancing framework in multirobot task allocation, *Int. J. Adv. Rob. Syst.*, vol. 15, no. 6, p. 172988141881296, 2018.

[12] L. Huang, Y. Ding, M. Zhou, Y. Jin, and K. Hao, Multiple-solution optimization strategy for multirobot task allocation, *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 50, no. 11, pp. 4283–4294, 2020.

[13] Z. Zhu, B. Tang, and J. Yuan, Multirobot task allocation based on an improved particle swarm optimization approach, *Int. J. Adv. Rob. Syst.*, vol. 14, no. 3, p. 172988141771031, 2017.

[14] J. Chen and D. Sun, Resource constrained multirobot task allocation based on leader-follower coalition methodology, *Int. J. Robot. Res.*, vol. 30, no. 12, pp. 1423–1434, 2011.

[15] A. Zhu and S. X. Yang, A neural network approach to dynamic task assignment of multirobots, *IEEE Trans. Neural Netw.*, vol. 17, no. 5, pp. 1278–1287, 2006.

[16] Z. Guo and D. Liu, Multi-robot task assignment algorithm based on improved self-organizing mapping network, *J. Phys.: Conf. Ser.*, vol. 1550, no. 3, p. 032062, 2020.

[17] J. Li and F. Yang, Task assignment strategy for multi-robot based on improved Grey Wolf Optimizer, *J. Ambient Intell. Humaniz. Comput.*, vol. 11, no. 12, pp. 6319–6335, 2020.

[18] T. Nagarajan and A. Thondiyath, Heuristic based task allocation algorithm for multiple robots using agents, *Procedia Eng.*, vol. 64, pp. 844–853, 2013.

[19] B. B. Choudhury and B. B. Biswal, Alternative methods for multi-robot task allocation, *J. Adv. Manuf. Syst.*, vol. 8, no. 2, pp. 163–176, 2009.

[20] Z. Chen, J. Alonso-Mora, X. Bai, D. D. Harabor, and P. J. Stuckey, Integrated task assignment and path planning for capacitated multi-agent pickup and delivery, *IEEE Robot. Autom. Lett.*, vol. 6, no. 3, pp. 5816–5823, 2021.

[21] P. Chand and D. A. Carnegie, Development of a reduced human user input task allocation method for multiple robots, *Robot. Auton. Syst.*, vol. 60, no. 10, pp. 1231–1244, 2012.

[22] D. Wu, G. Zeng, D. He, Z. Qian, and Q. Zhang, Task coordination organization model and the task allocation algorithm for resource contention of the syncretic system, *Tsinghua Science and Technology*, vol. 21, no. 4, pp. 459–470, 2016.

[23] L. Zhou, Y. Shi, J. Wang, and P. Yang, A balanced heuristic mechanism for multirobot task allocation of intelligent warehouses, *Math. Probl. Eng.*, vol. 2014, p. 380480, 2014.

[24] Y. Shen, L. Yu, and J. Li, Robust electric vehicle routing problem with time windows under demand uncertainty and weight-related energy consumption, *Complex Syst. Model. Simul.*, vol. 2, no. 1, pp. 18–34, 2022.

[25] R. Cui, J. Guo, and B. Gao, Game theory-based negotiation for multiple robots task allocation, *Robotica*, vol. 31, no. 6, pp. 923–934, 2013.

[26] T. Miyamoto and K. Inoue, Local and random searches for dispatch and conflict-free routing problem of capacitated AGV systems, *Comput. Ind. Eng.*, vol. 91, pp. 1–9, 2016.

[27] W. Xu and S. Guo, A multi-objective and multi-dimensional optimization scheduling method using a hybrid evolutionary algorithms with a sectional encoding

mode, *Sustainability*, vol. 11, no. 5, p. 1329, 2019.

[28] G. Li, X. Li, L. Gao, and B. Zeng, Tasks assigning and sequencing of multiple AGVs based on an improved harmony search algorithm, *J. Ambient Intell. Humaniz. Comput.*, vol. 10, no. 11, pp. 4533–4546, 2019.

[29] S. Eda, T. Nishi, T. Mariyama, S. Kataoka, K. Shoda, and K. Matsumura, Petri net decomposition approach for Bi-objective routing for AGV systems minimizing total traveling time and equalizing delivery time, *J. Adv. Mech. Des. Syst. Manuf.*, vol. 6, no. 5, pp. 672–686, 2012.

[30] H. Bai, T. Fan, Y. Niu, and Z. Cui, Multi-UAV cooperative trajectory planning based on many-objective evolutionary algorithm, *Complex Syst. Model. Simul.*, vol. 2, no. 2, pp. 130–141, 2022.

[31] J. Y. Mao, Q. K. Pan, Z. H. Miao, and L. Gao, An effective multi-start iterated greedy algorithm to minimize makespan for the distributed permutation flowshop scheduling problem with preventive maintenance, *Expert Syst. Appl.*, vol. 169, p. 114495, 2021.

[32] P. J. Kalczynski and J. Kamburowski, On the NEH heuristic for minimizing the makespan in permutation flow shops, *Omega*, vol. 35, no. 1, pp. 53–60, 2007.

[33] W. Liu, Y. Jin, and M. Price, A new improved NEH heuristic for permutation flowshop scheduling problems, *Int. J. Prod. Econ.*, vol. 193, pp. 21–30, 2017.

[34] X. Dong, H. Huang, and P. Chen, A more effective constructive algorithm for permutation flowshop problem. E. Corchado, H. Yin, V. Botti, and C. Fyfe, eds. *Intelligent Data Engineering and Automated Learning – IDEAL 2006*. Berlin, Germany: Springer, 2006. pp. 25–32,

[35] R. V. Rao and V. Patel, Multi-objective optimization of two stage thermoelectric cooler using a modified teaching-learning-based optimization algorithm, *Eng. Appl. Artif. Intell.*, vol. 26, no. 1, pp. 430–445, 2013.

[36] J. P. Huang, Q. K. Pan, Z. H. Miao, and L. Gao, Effective constructive heuristics and discrete bee colony optimization for distributed flowshop with setup times, *Eng. Appl. Artif. Intell.*, vol. 97, p. 104016, 2021.

[37] 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, p. 100804, 2021.

[38] Z. Xu, A. Elomri, S. Pokharel, and F. Mutlu, A model for capacitated green vehicle routing problem with the time-varying vehicle speed and soft time windows, *Comput. Ind. Eng.*, vol. 137, p. 106011, 2019.

[39] J. Q. Li, X. R. Tao, B. X. Jia, Y. Y. Han, C. Liu, P. Duan, Z. X. Zheng, and H. Y. Sang, Efficient multi-objective algorithm for the lot-streaming hybrid flowshop with variable sub-lots, *Swarm Evol. Comput.*, vol. 52, p. 100600, 2020.

[40] W. Q. Zou, Q. K. Pan, L. Wang, Z. H. Miao, and C. Peng, Efficient multiobjective optimization for an AGV energy-efficient scheduling problem with release time, *Knowl. Based Syst.*, vol. 242, p. 108334, 2022.

[41] X. Huang, Z. Guan, and L. Yang, An effective hybrid algorithm for multi-objective flexible job-shop scheduling problem, *Adv. Mech. Eng.*, vol. 10, no. 9, p. 168781401880144, 2018.

[42] P. A. N. Bosman and D. Thierens, The balance between proximity and diversity in multiobjective evolutionary algorithms, *IEEE Trans. Evol. Comput.*, vol. 7, no. 2, pp. 174–188, 2003.

[43] F. Gu, H. Liu, and H. Liu, A coevolutionary algorithm for many-objective optimization problems with independent and harmonious objectives, *Complex Syst. Model. Simul.*, vol. 3, no. 1, pp. 59–70, 2023.

[44] E. Zitzler and L. Thiele, Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach, *IEEE Trans. Evol. Comput.*, vol. 3, no. 4, pp. 257–271, 1999.

[45] R. Li, W. Gong, L. Wang, C. Lu, and X. Zhuang, Surprisingly popular-based adaptive memetic algorithm for energy-efficient distributed flexible job shop scheduling, *IEEE Trans. Cybern.*, vol. 53, no. 12, pp. 8013–8023, 2023.

[46] X. He, Q. K. Pan, L. Gao, L. Wang, and P. N. Suganthan, A greedy cooperative co-evolutionary algorithm with problem-specific knowledge for multiobjective flowshop group scheduling problems, *IEEE Trans. Evol. Comput.*, vol. 27, no. 3, pp. 430–444, 2023.

[47] A. M. Luvembe, W. Li, S. Li, F. Liu, and G. Xu, Dual emotion based fake news detection: A deep attention-weight update approach, *Inf. Process. Manag.*, vol. 60, no. 4, p. 103354, 2023.

[48] W. Li, C. Guo, Z. Deng, F. Liu, J. Wang, R. Guo, C. Wang, and Q. Jin, Coevolution modeling of group behavior and opinion based on public opinion perception, *Knowl. Based Syst.*, vol. 270, p. 110547, 2023.

[49] E. Jiang, L. Wang, and J. Wang, Decomposition-based multi-objective optimization for energy-aware distributed hybrid flow shop scheduling with multiprocessor tasks, *Tsinghua Science and Technology*, vol. 26, no. 5, pp. 646–663, 2021.

[50] Y. Zhang, Z. Jing, and Y. Zhang, MR-IDPSO: A novel algorithm for large-scale dynamic service composition, *Tsinghua Science and Technology*, vol. 20, no. 6, pp. 602–612, 2015.

[51] H. Zhang, J. Xie, J. Ge, J. Shi, and Z. Zhang, Hybrid particle swarm optimization algorithm based on entropy theory for solving DAR scheduling problem, *Tsinghua Science and Technology*, vol. 24, no. 3, pp. 282–290, 2019.

[52] X. Han, Y. Han, Q. Chen, J. Li, H. Sang, Y. Liu, Q. Pan, and Y. Nojima, Distributed flow shop scheduling with sequence-dependent setup times using an improved iterated greedy algorithm, *Complex Syst. Model. Simul.*, vol. 1, no. 3, pp. 198–217, 2021.

Nianbo Kang received the BEng degree from Lanzhou Jiao Tong University, China in 2020. He is currently a master student at School of Mechatronic Engineering and Automation, Shanghai University, China. His research interests include intelligent optimization, task allocation, path planning, and scheduling algorithms.

Zhonghua Miao received the PhD degree in mechatronic engineering from Shanghai Jiao Tong University, China in 2010. He is currently a full professor and doctoral advisor at School of Mechatronic Engineering and Automation, Shanghai University, China. His research interests include intelligent robotics, swarm robots, and agricultural machinery equipment.

Weimin Li received the BEng and MEng degrees from Shandong University of Science and Technology, China in 1996 and 2004, respectively, and the PhD degree from Donghua University, China in 2008. He was a research fellow at Japan Society for the Promotion of Science with Department of Human Informatics and Cognitive Sciences, Waseda University, Japan, from November 2012 to January 2013. He was a visiting scholar at Department of Computer Science, University of California, Santa Barbara, USA, supported by the China Scholarship Council from December 2015 to December 2016. He is currently a professor at School of Computer Engineering and Science, Shanghai University. He is involved in extensive research in computer science, service computing, and database technology. His current research interests include social computing, bioinformatics, group behavior modeling and simulating, and service recommendation.

Quan-Ke Pan received the BEng and PhD degrees from Nanjing University of Aeronautics and Astronautics, Nanjing, China in 1993 and 2003, respectively. From 2003 to 2011, he was at School of Computer Science Department, Liaocheng University, China, where he became a full professor in 2006. From 2011 to 2014, he worked at State Key Laboratory of Synthetical Automation for Process Industries, Northeastern University, Shenyang, China. From 2014 to 2015, he was at State Key Laboratory of Digital Manufacturing and Equipment Technology, Huazhong University of Science & Technology, China. He has worked at School of Mechatronic Engineering and Automation, Shanghai University since 2015. His current research interests include intelligent optimization and scheduling algorithms.

M. Fatih Tasgetiren received the BEng and MEng degrees from Istanbul Technical University (ITU), Türkiye, and the PhD degree in production and operations management from Istanbul University, Türkiye. His research focus is on modeling, analysis, and optimization of complex systems through the use of computational intelligence methods. He works on the design and development of modern meta-heuristic algorithms to solve discrete/combinatorial/binary, as well as real-parameter unconstrained/constrained optimization problems. His main research interest has been on sequencing and scheduling problems. Furthermore, he recently works on the development of energy-efficient production scheduling systems, as well as architectural design optimization problems. His current google academic citations are 10 033 with an h-index of 45.