The International Journal of Advanced Manufacturing Technology (2021) 116:3731–3750
https://doi.org/10.1007/s00170-021-07696-x

ORIGINAL ARTICLE

Disassembly sequence planning based on a modified grey wolf optimizer

Jin Xie[1] · Xinyu Li[1] · Liang Gao[1]

Received: 17 March 2021 / Accepted: 11 July 2021 / Published online: 22 July 2021

Abstract

Disassembly sequence planning (DSP) can effectively increase the disassembly efficiency, shorten the disassembly cycle, reduce disassembly costs, and reduce environmental hazards of end-of-life (EOL) products, playing an important role in manufacturing industries. Thus, it is urgent to propose an approach to solve the DSP problem. DSP is a famous NP-hard combinatorial optimization problem. As the size of components increases, exact algorithms can hardly obtain the optimal disassembly sequence. Therefore, we propose a promising intelligence algorithm, modified grey wolf optimizer (MGWO), to solve the DSP problem. MGWO inherits the main idea of the hierarchy and hunting mechanism of the original grey wolf optimizer (GWO). Three new operators are designed in MGWO to ensure the feasibility of solutions under the complex constraint of disassembly precedence. The feasible solution generator (FSG) is designed to obtain feasible disassembly sequences, the neighborhood search operator (NSO) is developed to make wolves (solutions) self-evolving, and the guided search operator (GSO) is used to make the wolf group guided by three leaders of wolves. Two engineering cases are applied to validate the effectiveness of the proposed operators. Then, they and two real-world applications are used to compare the MGWO with other reported methods. The results demonstrate that MGWO can solve the DSP problem effectively.

Keywords Disassembly sequence planning · End-of-life · Grey wolf optimizer · Real-world application

1 Introduction

With the renewal of modern technology and the shortening of product life, an enormous amount of end-of-life (EOL) products has been generated. Minimizing the adverse environmental effect of EOL products and increasing their recovery rate become a global trend. Disassembly, a systematic method to divide products into components and subassemblies, is the first and often the most challenging process of remanufacturing EOL products [1]. A good disassembly sequence can effectively improve the disassembly efficiency, shorten the disassembly cycle, and reduce the disassembly cost. Hence, the key of the disassembly process is to obtain the optimal disassembly sequence.

Disassembly sequence planning (DSP) studies the disassembly sequence of components for a given EOL product to minimize disassembly time and cost, maintain stability during the disassembly process, and improve recovery efficiency. It has been proved to be an NP-hard problem [2]. The common methods for solving DSP include modeling, mathematical programming, and intelligent optimization algorithms. The modeling methods contain AND/OR graphs, Petri nets, and matrix-based models. Lambert [3] used an AND/OR graph to represent the disassembly process and designed a binary integer linear programming approach based on the AND/OR graph to solve DSP. Cappelli et al. [4] proposed an AND/OR graph of mechanical assemblies to represent the solution space of disassembly sequences. Rai et al. [5] combined a Petri net with heuristic search procedures to solve DSP. The Petri net guides the disassembly process through a token game. Kuo [6] proposed a Petri net–based analysis approach to solve DSP. The Petri net determined the optimal tradeoff between the cost and the environmental effectiveness of the disassembly processes. Li et al. [7] designed an interference matrix to represent the geometric constraints between components. The feasible disassembly sequence was generated based on the

✉ Liang Gao
gaoliang@mail.hust.edu.cn

[1] State Key Laboratory of Digital Manufacturing Equipment and Technology, Huazhong University of Science and Technology, Wuhan 430074, China

interference matrix. In addition to these modeling methods, many linear programming methods are also used to solve DSP. Zhu et al. [8] introduced a machine-readable disassembly information model and proposed a linear programming-based optimization method to solve the method. Ma et al. [9] designed an integer programming method to solve DSP in a parallel disassembly environment to maximize profit. Behdad et al. [10] developed a stochastic mixed-integer nonlinear programming method to solve DSP considering uncertain outcomes, such as time, cost, and the probability of causing damage. Kim and Lee [11] developed an integer programming model to represent disassembly sequences and proposed a branch and bound method for obtaining the lower and upper bounds of DSP. The limitation of modeling and mathematical programming is that they can only solve small-scale DSP problems. With the expansion of the scale of problems, the computational time of modeling and mathematical programming increases exponentially. The high computational time is unacceptable. Hence, the intelligent optimization algorithm is the most widely used method for solving DSP. Wang et al. [12] proposed a genetic algorithm (GA) to solve DSP. Their crossover and mutation operators can effectively help the algorithm to find the optimal disassembly sequence. Go et al. [13] designed an optimization model for DSP and presented a GA to solve it. Yeh [14] proposed a simplified swarm optimization (SSO) algorithm for solving DSP. His algorithm combined the precedence preservative operator, feasible solution generator, repetitive pairwise exchange, and self-adaptive parameter control procedures. Then, Yeh [15] modified the SSO algorithm by improving its update mechanism and revising its self-adaptive parameter control procedure to make it better solve DSP. Percoco and Diella [16] used an artificial bee colony (ABC) algorithm for DSP considering the disassembly cost and the environmental impact. Tian et al. [17] presented an ABC algorithm to solve DSP considering the stochastic cost and time of product disassembly. Additionally, many intelligent optimization algorithms, including GA [18–20], particle swarm optimization (PSO) [21, 22], ant colony optimization (ACO) [23, 24], teaching-learning-based optimization (TLBO) [25, 26], and Bees algorithm (BA) [27, 28], have also been designed for solving DSP. The limitation of intelligent optimization algorithms is that their accuracy is not exact. The intelligent optimization algorithms can only obtain an approximate optimal solution, but cannot guarantee to achieve an exact solution. Therefore, how to design an intelligent optimization algorithm to balance solution quality and computational efficiency has become the focus of our research. Besides, the parameter settings in most of the above algorithms need constant tunings, such as acceleration constant and inertia weight in PSO, the probability of crossover and mutation in GA, and pheromone evaporation rate in ACO. The characteristic makes it difficult for above algorithms to adapt to a variety of EOL products. Based on this, we design an algorithm with few parameters, which are adaptive and robust for various situations. It is an obvious advantage of our algorithm compared with previously reported methods.

Grey wolf optimizer (GWO) is a novel intelligence optimization method designed by Mirjalili et al. [29], inspired by mimicking the hunting mechanism and the social hierarchy of the wolf group. Due to its high convergence speed and simple implementation, GWO has been applied in many industrial fields. For example, Mohanty et al. [30] used GWO to implement a maximum power point tracking design for maximizing power extraction in a photovoltaic system. Jayakumar et al. [31] proposed GWO to allocate generation and heat outputs to the committed units in a power system operation. Their proposed method maximized power output while minimizing cost. Lu et al. [32] applied a multi-objective GWO to the welding shop scheduling. Besides, GWO was also used in engineering design [33], path planning [34], and power scheduling [35], etc. Unlike many meta-heuristic algorithms, GWO does not require many parameters to be tuned. Hence, it is very convenient to use GWO to solve optimization problems.

However, GWO is designed for continuous optimization problems, so it cannot be directly applied to the DSP problem. This paper develops a modified grey wolf optimizer (MGWO) to solve the DSP problem. The algorithm inherits the significant thought of the social hierarchy and hunting mechanism of the original GWO. The novelty of our GWO is as follows: (1) Three new operators, i.e., feasible solution generator (FSG), neighborhood search operator (NSO), and guided search operator (GSO), are designed for DSP. FSG is developed for generating feasible disassembly sequences. NSO and GSO are designed to update disassembly sequences and make the wolf group evolve in a good direction. (2) In the original GWO, the δ wolf is the third-best wolf. However, through experimental analysis, we found that if δ wolf is set as a new wolf generated by FSG, the performance of GWO is better for solving DSP. (3) In addition, in the original GWO, the first three wolves are preserved to the next generation without being guided by other wolves, but in our GWO, α, β, and δ wolves would exchange their prey information with each other.

The remaining sections of this paper are organized as follows. Section 2 gives the disassembly model for DSP. Section 3 describes the MGWO for obtaining the

Fig. 1 The pseudo-code of the MGWO algorithm

```
MGWO
Input: the interference matrix, support matrix, material set, tool set of the EOL product
Output: the optimal disassembly sequence
//Initialization;
    For i = 1 to population size do
    Generate wolf      X_i using FSG;
    Calculate the      fitness value of each wolf f(X_i);
    End for
//Loop    :
    While (stop criterion is not satisfied & maximum iterations are not exceeded) do
    // Neighborhood search      :
    Select two wolves with the      two best fitness values as α and β;
    Generate a new wolf as      δ using FSG and calculate its fitness value f(δ);
    Update      α, β, and δ wolves using NSO;
    // Guided search      :
        For i = 1 to population size do
      Update wolf      X_i using GSO;
        End for
    End while
```

optimal disassembly sequence. Section 4 verifies the superiority of MGWO by utilizing two case studies and two real-world applications. Section 5 discusses the effectiveness and superiority of MGWO. Finally, Section 6 provides the conclusion and future researches.

Fig. 2 The pseudo-code of FSG

```
FSG
Input: the interference matrix IM_10+ of the EOL product
Output: a feasible disassembly sequence
Set the set C_A of components available for disassembly to Ø;
Set the map D_A for feasible disassembly directions of components in C_A to Ø;
For i = 1 to the length of disassembly component sequence X do
    For each component c in the EOL product do;
        If c ∈ X then
            Continue;
        End if
        If c can be disassembled by checking IM_10+ then
            Record the feasible disassembly direction set D_c of c;
            Push c back to C_A;
            Put D_c into D_A;
        End if
    End for
    Randomly choose a component c from C_A;
    Push the c back to X;
    Randomly choose a feasible disassembly direction k of c from D_A;
    Push the k back to disassembly direction sequence D;
    Set C_A to Ø;
    Set D_A to Ø;
    Set all the elements in row c or column c of IM_10+ to 0;
End for
```

Fig. 3 The pseudo-code of NSO

NSO

Input: the interference matrix IM_{10+} of the EOL product, the disassembly component sequence X, and its corresponding disassembly direction sequence D
Output: an improved disassembly sequence

```
Record the disassembly component sequence X and its corresponding disassembly direction sequence D;
Randomly select a component c from X and record its disassembly direction k;
Set the interference direction set I_c to Ø;  // I_c includes the directions in which component c is interfaced by components preceding c
For i = the serial number of c to 1 do
Record the interference direction set D̄_i of c interfered by component x_i through the matrix IM_10+;
    I_c = I_c ∪ D̄_i;
    If the size of I_c = 6 then
  Set the earliest insertion point a to i;
  Break;
    End if
End for
For i = the serial number of c + 1 to the length of X do
    If c blocks the disassembly direction of x_i according to D then;
   Set the latest insertion point b to i;
   Break;
    End if
End for
Randomly select an insertion point p_insert between a and b;
Update X by inserting c into p_insert, and delete c at the original position;
For i = 1 to p_insert do
 Set all the elements in row i or column i in IM_10+ to 0;
End for
Record the feasible disassembly direction set D_c of c by checking IM_10+;
If p_insert = the serial number of c then
    D_c = D_c \ {d};  // d is the original disassembly direction of c
End If
Randomly choose a feasible disassembly direction k from D_c;
Update D by modifying the disassembly direction of c.
```

2 Disassembly modeling

A proper disassembly sequence is vital for high recovery efficiency and short recovery time as a starting point for the engineer considering recycling the EOL products. Therefore, the first step of the DSP problem is modeling. This section introduces a disassembly model considering some important disassembly factors, including the geometric constraints, the changes of disassembly directions, the changes of disassembly tools, and the stability of disassembly [18, 21].

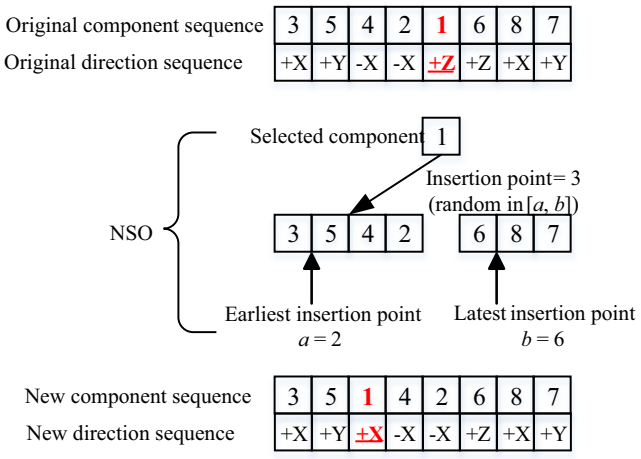

Fig. 4 Schematic diagram of NSO

2.1 Geometric constraints of the disassembly process

In the space rectangular coordinate system, the disassembly directions of components are generally divided into 6 directions $d_k = \{-x, -y, -z, +x, +y, +z\}$. In this paper, a decimal interference matrix IM_{10+} is defined to denote the interference relationship between components along $+x$-, $+y$-, and $+z$-axes. The interference relationship between components along $-x$-, $-y$-, and $-z$-axes can be obtained by the inversion of IM_{10+}.

$$IM_{10+} = (I_{ij})_{n\times n} \tag{1}$$

$$IM_{10-} = (IM_{10+})^T \tag{2}$$

where I_{ij} denotes the interference relationship between component i and component j, when component i is disassembled in the positive direction of each axis.

Fig. 5 The pseudo-code of GSO

GSO

Input: the disassembly sequences of α, β, δ, and X wolves

Output: a new disassembly sequence

Set the new disassembly component sequence X_{new} to Ø;
Set the new disassembly direction sequence D_{new} to Ø;
For i = 1 to the length of disassembly component sequence X **do**
    Randomly generate probability $p \in [0, 1]$;
    If $p < p_s$ **then**
        Push the leftmost component of X back to X_{new};
        Push the leftmost direction of D back to D_{new};
    Else if $p < p_s + p_\alpha$ **then**
        Push the leftmost component of α back to X_{new};
        Push the leftmost direction of D_α back to D_{new};
    Else if $p < p_s + p_\alpha + p_\beta$ **then**
        Push the leftmost component of β back to X_{new};
        Push the leftmost direction of D_β back to D_{new};
    Else
        Push the leftmost component of δ back to X_{new};
        Push the leftmost direction of D_δ back to D_{new};
    End if
    Remove the chosen component from X_i, α, β, δ;
    Remove the direction of the chosen component from D_i, D_α, D_β, D_δ;
End for
Update X with X_{new};
Update D with D_{new}.

$$I_{ij} = \begin{cases} 0, & \text{if component } i \text{ isn't interfered by component } j \\ 1, & \text{if component } i \text{ is interfered by component } j \text{ in the direction } +z \\ 2, & \text{if component } i \text{ is interfered by component } j \text{ in the direction } +y \\ 3, & \text{if component } i \text{ is interfered by component } j \text{ in the directions } +y, \quad +z \\ 4, & \text{if component } i \text{ is interfered by component } j \text{ in the direction } +x \\ 5, & \text{if component } i \text{ is interfered by component } j \text{ in the directions } +x, \quad +z \\ 6, & \text{if component } i \text{ is interfered by component } j \text{ in the directions } +x, \quad +y \\ 7, & \text{if component } i \text{ is interfered by component } j \text{ in the direction } +x, \quad +y, \quad +z \end{cases} \tag{3}$$

Table 1 An example of executing GSO to obtain a new disassembly sequence

Step	p	X, D	α, D_α	β, D_β	δ, D_δ	X_{new}, D_{new}
1	0.2	(2, 3, 1, 4, 5) (+x, +x, +y, +y, −z)	(4, 3, 5, 1, 2) (+z, +x, +z, +y, −z)	(4, 3, 5, 2, 1) (+x, +x, +z, +y, −y)	(2, 3, 5, 1, 4) (+y, +x, +z, +y, +z)	(4) (+z)
2	0.3	(2, 3, 1, 5) (+x, +x, +y, −z)	(3, 5, 1, 2) (+x, +z, +y, −z)	(3, 5, 2, 1) (+x, +z, +y, −y)	(2, 3, 5, 1) (+y, +x, +z, +y)	(4, 3) (+z, +x)
3	0.9	(2, 1, 5) (+x, +y, −z)	(5, 1, 2) (+z, +y, −z)	(5, 2, 1) (+z, +y, −y)	(2, 5, 1) (+y, +z, +y)	(4, 3, 2) (+z, +x, +y)
4	0.7	(1, 5) (+y, −z)	(5, 1) (+z, +y)	(5, 1) (+z, −y)	(5, 1) (+z, +y)	(4, 3, 2, 5) (+z, +x, +y, +z)
5	0.4	(1) (+y)	(1) (+y)	(1) (−y)	(1) (+y)	(4, 3, 2, 5, 1) (+z, +x, +y, +z, +y)

Remarks: the underline presents the chosen component in each step

2.2 Changes of disassembly directions and tools

In disassembling EOL products, we found that changes in disassembly directions and tools increase extra expenses of time and cost and reduce the disassembly efficiency and flexibility. Hence, few changes of disassembly directions and tools benefit disassembly.

This paper defines $f_d(x)$ to denote the fitness evaluation for changes of disassembly directions and $f_t(x)$ to represent the fitness evaluation for changes of disassembly tools.

$$f_d(X) = \sum_{i=1}^{n-1} d(x_i) \tag{4}$$

$$f_t(X) = \sum_{i=1}^{n-1} t(x_i) \tag{5}$$

where $X = \{x_1, x_2, \ldots, x_n\}$ denotes the disassembly component sequence, $d(x_i)$ and $t(x_i)$ can be calculated as follows:

$$d(x_i) = \begin{cases} 0, \text{if the directions between } x_i \text{ and } x_{i+1} \text{ are not changed} \\ 1, \text{if the directions between } x_i \text{ and } x_{i+1} \text{ are changed by } 90^o \\ 2, \text{if the directions between } x_i \text{ and } x_{i+1} \text{ are changed by } 180^o \end{cases} \tag{6}$$

$$t(x_i) = \begin{cases} 0, \text{if the tools between } x_i \text{ and } x_{i+1} \text{ are not changed} \\ 1, \text{if the tools between } x_i \text{ and } x_{i+1} \text{ are changed} \end{cases} \tag{7}$$

2.3 Changes of component materials

To facilitate the subsequent recycling of EOL products, we want to disassemble the components with the same materials in priority. Therefore, it is necessary to minimize the material changes of components during a disassembly process.

In this paper, $f_m(x)$ represents the fitness evaluation for material changes of the disassembly sequence.

$$f_m(X) = \sum_{i=1}^{n-1} m(x_i) \tag{8}$$

Table 2 Comparison results of the vise with different strategies

Algorithms	MGWO	MGWO-1	MGWO-2	MGWO-3	MGWO-4
Best fitness	**3.7500**	**3.7500**	4.0000	**3.7500**	4.2500
Minimum	**3.7500**	**3.7500**	4.0000	**3.7500**	4.2500
Maximum	**4.2500**	**4.2500**	4.5000	4.5000	5.0000
Median	**3.7500**	4.0000	4.2500	4.0000	4.6250
Mean	**3.8667**	4.0167	4.2167	4.0583	4.6250
Average computation time (s)	7.05	7.73	8.85	9.13	**6.48**

The optimal value is indicated in bold

where $m(x_i)$ can be calculated as follows:

$$m(x_i) = \begin{cases} 0, \text{if the materials between } x_i \text{ and } x_{i+1} \text{ are not changed} \\ 1, \text{if the materials between } x_i \text{ and } x_{i+1} \text{ are changed} \end{cases} \tag{9}$$

2.4 Stability of the disassembly process

The stability of disassembly sequences is used to evaluate the stability between connected components during the disassembly process. An unstable situation may cause damage to recyclable components, reduce disassembly efficiency, and increase costs. For this reason, a stability matrix $S = (s_{ij})_{n\times n}$ is defined to represent the stability between components. If component i supports component j, $s_{ij} = 1$; otherwise, $s_{ij} = 0$.

In this paper, $f_s(x)$ is defined to represent the fitness evaluation for component stability of the disassembly process.

$$f_s(X) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} s_{x_i x_j}$$

where $s_{x_i x_j}$ is computed as follows:

$$s_{x_i x_j} = \begin{cases} 1, x_i \text{ supports } x_j \\ 0, x_i \text{ does not support } x_j \end{cases} \tag{11}$$

2.5 Fitness function

A disassembly sequence firstly needs to satisfy geometric constraints. Then, according to the literature [18], a linear weighted expression considering the above four indicators is designed to measure the disassembly quality for the feasible sequence.

$$f(X) = w_d f_d(X) + w_t f_t(X) + w_m f_m(X) + w_s f_s(X) \tag{12}$$

where w_d, w_t, w_m, and w_s are the weight factors of $f_d(X)$, $f_t(X)$, $f_m(X)$, $f_s(X)$, respectively. In subsequent experiments, all weight factors are set to 0.25.

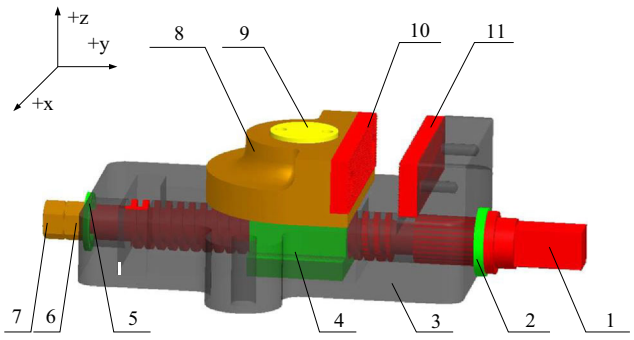

Fig. 6 Assembly drawing of the vise

$$\begin{bmatrix} 0 & 5 & 5 & 5 & 5 & 5 & 1 & 1 & 1 & 1 \\ 7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 7 & 2 & 0 & 6 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 7 & 2 & 7 & 0 & 0 & 0 & 0 & 7 & 7 & 3 & 2 \\ 7 & 2 & 2 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 7 & 0 & 2 & 2 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 7 & 0 & 2 & 2 & 2 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 6 & 0 & 0 & 0 & 0 & 7 & 3 & 2 \\ 0 & 0 & 2 & 6 & 0 & 0 & 0 & 6 & 0 & 2 & 2 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

(a)

$$\begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

(b)

Component	1	2	3	4	5	6	7	8	9	10	11
Material	A	B	C	B	B	D	D	D	E	A	A

(c)

Component	1	2	3	4	5	6	7	8	9	10	11
Tool	A	B	C	C	B	D	D	C	E	C	C

(d)

Fig. 7 Matrices, material, and tool sets of the vise. **a** Interference matrix. **b** Support matrix. **c** Material set. **d** Tool set

3 Proposed MGWO for DSP

This section introduces the MGWO for the DSP problem in detail. The original GWO is described briefly, firstly. Afterward, the framework of MGWO is proposed. Finally, three critical operators are designed in MGWO for solving the combinatorial optimization problem.

3.1 Brief introduction of GWO

GWO is a new meta-heuristic algorithm characterized by presenting the social hierarchy and hunting mechanism of the wolf group. All wolves are divided into four hierarchies based on their fitness values. The best wolf is represented as α, the second wolf is denoted as β, and the third-best wolf is expressed as δ. The remaining wolves are described as ω. During the search process, α, β, and δ wolves guide ω wolves.

The prey is firstly encircled by grey wolves when hunting. Equation 13 is expressed to the model of the encircling behavior.

$$D = \left| C \cdot X_{\mathrm{p}}(t) - X(t) \right| \tag{13}$$

$$X(t+1) = X_{\mathrm{p}}(t) - A \cdot D \tag{14}$$

where D denotes the distance between the prey and a wolf, $\boldsymbol{X}$ and $\boldsymbol{X}_{\mathbf{p}}$ indicate the position vector of the wolf and the prey, respectively, and t represents the current iteration. Finally, the vector $\boldsymbol{A}$ and $\boldsymbol{C}$ are formulated using Eqs. (15) and (16).

$$A = 2a \cdot r_1 - a \tag{15}$$

$$C = 2r_2 \tag{16}$$

where $\boldsymbol{r_1}$ and $\boldsymbol{r_2}$ are random vectors between 0 and 1, and a decreases linearly from 2 to 0 throughout the iterations. α, β, and δ wolves guide the wolf group to hunt for the prey. Nevertheless, the optimum (prey) position cannot be determined in an unknown and abstract solution space. Aiming at mathematically describing the hunting practices of wolves, the

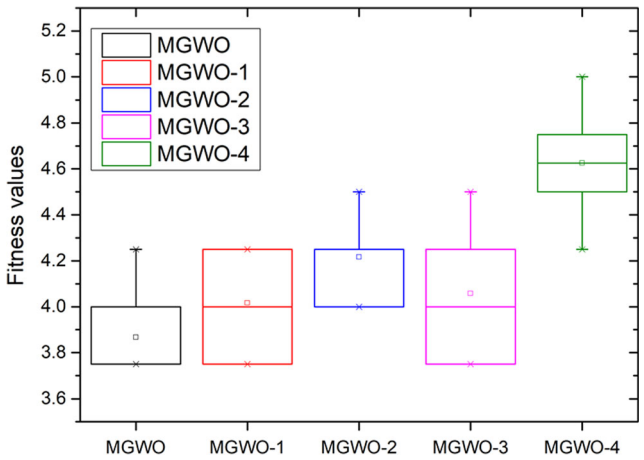

Fig. 8 Box plot of best fitness of vise under different strategies

Table 3 Wilcoxon rank-sum test of the vise with different strategies

Strategies	Significance	*p*-value
MGWO-1	1	4.4058E−03
MGWO-2	1	2.0515E−08
MGWO-3	1	1.6708E−03
MGWO-4	1	1.1214E−11

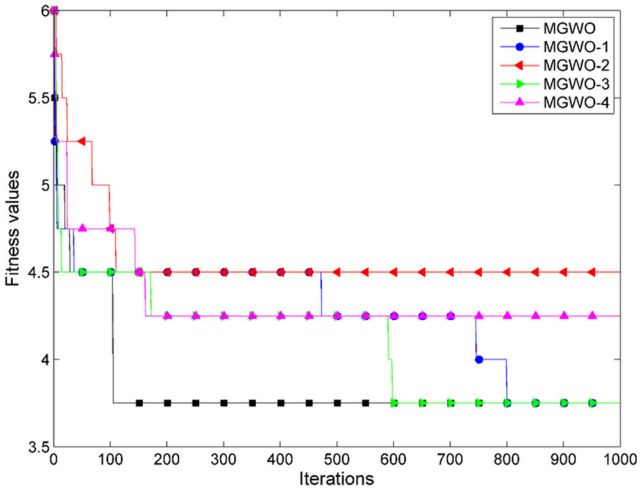

Fig. 9 Convergence curves of vise under different strategies

algorithm assumes that α, β, and δ wolves are closer to the prey than ω wolves. Hence, the three best wolves are saved and oblige the other wolves to update their position toward the prey. Equations (17–23) calculate the hunting practices of the wolf group [29].

$$D_\alpha = |C_1 \cdot X_\alpha - X| \tag{17}$$

$$D_\beta = |C_2 \cdot X_\beta - X| \tag{18}$$

$$D_\gamma = |C_3 \cdot X_\gamma - X| \tag{19}$$

$$X_1 = X_\alpha - A_1 \cdot D_\alpha \tag{20}$$

$$X_2 = X_\beta - A_2 \cdot D_\beta \tag{21}$$

$$X_3 = X_\gamma - A_3 \cdot D_\gamma \tag{22}$$

$$X(t+1) = \frac{X_1 + X_2 + X_3}{3} \tag{23}$$

In summary, wolves are first divided into four kinds of hierarchies in GWO. The first three wolves lead the remaining wolves to search for the prey. According to Eq. (14), when $|A| < 1$, wolves will move between themselves and the prey. Otherwise, they attack the prey. Since the actual prey position is unknown, Eqs. (20–22) assumes the position of the prey is

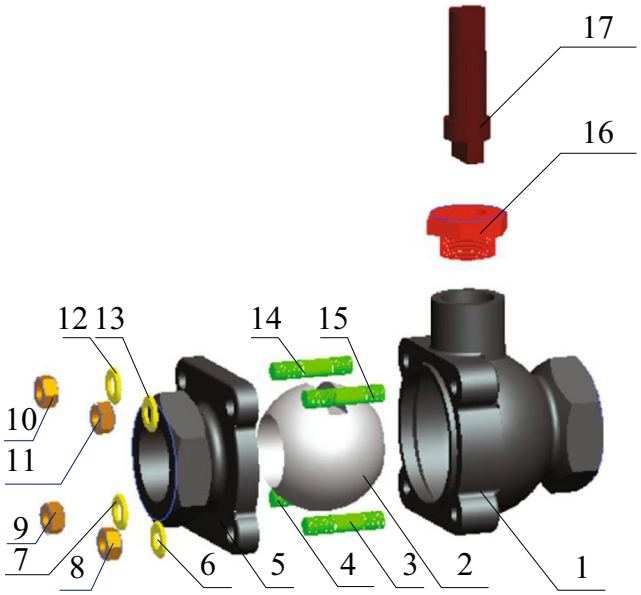

Fig. 10 Assembly drawing of ball valve

the same as that of α, β, and δ wolves. Finally, the optimal solution is obtained when the stop criterion is satisfied.

3.2 Framework of the proposed MGWO

The proposed MGWO inherits the main idea of the social hierarchy and hunting mechanism from the original GWO. Feasible solution generator (FSG), neighborhood search operator (NSO), and guided search operator (GSO) are designed in the algorithm. In the original GWO, the δ wolf is the third-best wolf. However, in the proposed MGWO, the δ wolf is a new wolf generated by FSG. In addition, in the original GWO, the first three wolves are preserved to the next generation without being guided by other wolves, but in the MGWO, α, β, and δ wolves would be guided by each other. The reason for the modification is explained in Section 4.1. The computation results verify that the modified algorithm outperforms the original algorithm in solving the DSP problem. The pseudo-code of MGWO is shown in Fig. 1.

Table 4 Comparison results of the ball valve

Algorithms	MGWO	MGWO-1	MGWO-2	MGWO-3	MGWO-4
Best fitness	**3.7500**	**3.7500**	**3.7500**	**3.7500**	**3.7500**
Minimum	**3.7500**	**3.7500**	**3.7500**	**3.7500**	**3.7500**
Maximum	**3.7500**	4.5000	4.2500	4.5000	5.0000
Median	**3.7500**	**3.7500**	4.0000	4.0000	4.0000
Mean	**3.7500**	3.8583	4.0417	3.9917	4.1417
Average computation time (s)	7.21	6.94	8.73	10.57	**6.26**

The optimal value is indicated in bold

$$\begin{bmatrix} 0&5&5&5&5&0&0&0&0&0&0&0&0&5&5&7&7\\ 7&0&4&0&5&0&0&0&0&0&0&0&0&1&5&1&3\\ 7&1&0&0&5&5&0&5&0&0&1&0&1&0&1&0&0\\ 7&5&4&0&5&4&5&4&5&1&0&1&0&1&0&0&0\\ 7&7&5&5&0&4&0&4&0&1&5&1&5&5&5&1&2\\ 2&0&5&0&3&0&0&0&0&0&0&0&1&0&1&0&0\\ 2&0&4&5&7&4&0&0&0&0&0&1&0&1&0&0&0\\ 2&0&7&0&3&2&0&0&0&0&1&0&0&0&1&0&0\\ 2&0&4&7&7&0&2&4&0&1&0&0&0&1&0&0&0\\ 2&0&0&0&6&0&0&0&0&0&4&2&0&7&4&0&0\\ 2&0&0&0&2&0&0&0&0&0&0&0&2&0&7&0&0\\ 2&0&0&0&6&0&0&0&0&0&0&0&4&5&4&0&0\\ 2&0&0&0&2&0&0&0&0&0&0&0&0&0&5&0&0\\ 7&4&0&0&5&0&0&0&0&5&4&5&4&0&4&0&0\\ 7&0&0&0&5&0&0&0&0&0&5&0&5&0&0&0&0\\ 6&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&6\\ 6&2&0&0&0&0&0&0&0&0&0&0&0&0&0&7&0 \end{bmatrix}$$

(a)

$$\begin{bmatrix} 0&1&1&1&0&0&0&0&0&0&0&0&0&1&1&1&0\\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1\\ 0&0&0&0&1&1&0&1&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&1&0&1&0&1&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&1&0&0&0&0&1&0&1&0&0&0&0&0\\ 0&0&0&0&1&0&0&0&0&0&1&0&1&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \end{bmatrix}$$

(b)

Component	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Material	A	B	C	C	D	E	E	D	D	D	D	E	E	C	C	F	G

(c)

Component	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Tool	A	B	C	C	D	E	E	F	F	F	F	E	E	C	C	G	H

(d)

Fig. 11 Matrices, material, and tool sets of ball valve. **a** Interference matrix. **b** Support matrix. **c** Material set. **d** Tool set

In the NS phase, α and β wolves with the best fitness values are identified first in the population, and the δ wolf is generated by FSG. Then, a neighborhood search is performed on each of the three wolves (disassembly sequences). First, a component in the disassembly sequence is randomly selected, and then its earliest insertion point a and latest insertion point b in the original disassembly sequence are determined. Finally, the component is inserted into a random position between [a, b] of the original disassembly sequence, and then the new disassembly sequence is generated.

In the GS phase, α, β, and δ wolves guide each wolf in the population to search the prey. The self-updating factor and the guided factors are calculated according to the fitness values of the chosen wolf and these three wolves. These self-adaptive factors determine the probability that the chosen wolf will be guided by the other three wolves.

A detailed description of FSG, NSO, and GSO will be introduced in the following sections.

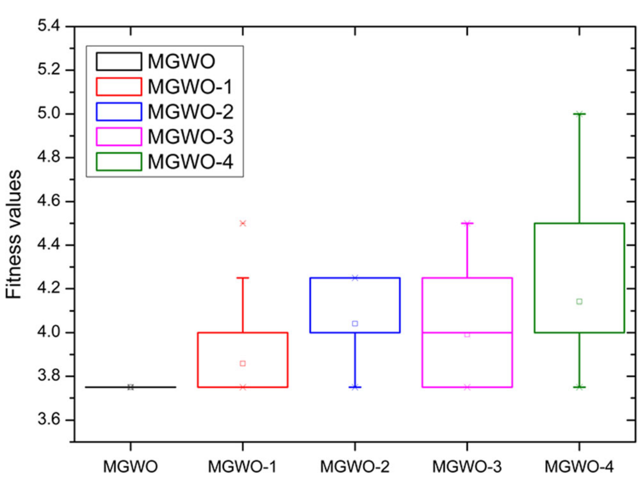

Fig. 12 Box plot of ball valve

3.3 Feasible solution generator

The constraint of disassembly precedence should be satisfied in the disassembly process. In our research, the constraint of

Table 5 Wilcoxon rank-sum test of the ball valve

Algorithms	Significance	p-value
MGWO-1	1	1.3506E−03
MGWO-2	1	9.7929E−11
MGWO-3	1	2.0576E−06
MGWO-4	1	1.7293E−09

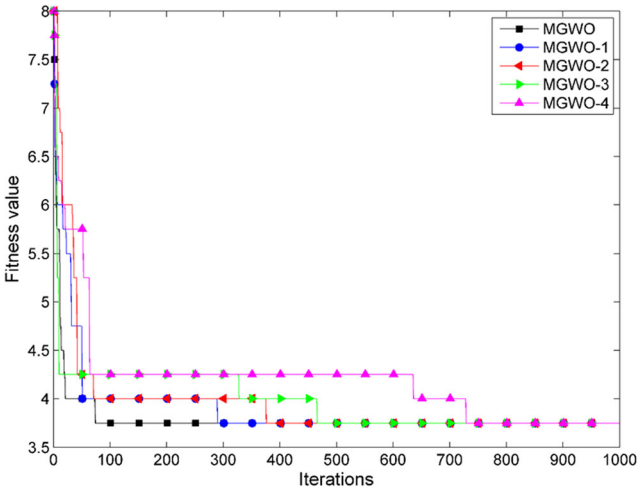

Fig. 13 Convergence curves of ball valve

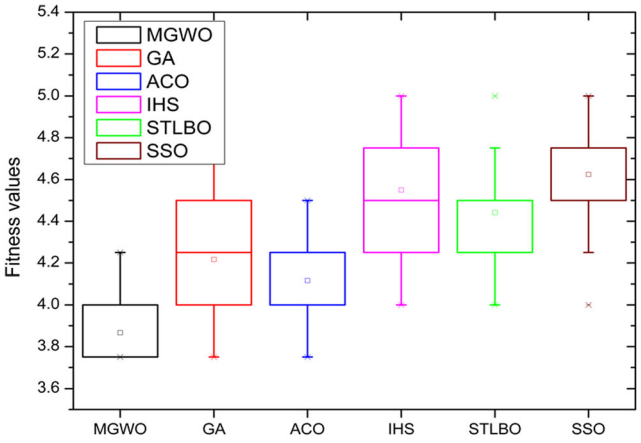

Fig. 14 Box plot of the vise

disassembly precedence is represented by the decimal interference matrix IM_{10+}, introduced in Section 2.1. A feasible disassembly sequence is generated by FSG. The pseudo-code of FSG is shown in Fig. 2.

3.4 Neighborhood search operator

NSO is used for neighborhood search on α, β, and δ wolves, making them update themselves. For each of these three disassembly sequences, a component is randomly selected to change its position or disassembly direction, making the sequence variation. If the fitness value of the disassembly sequence is improved after using NSO, the original sequence will be replaced by the new sequence. In the NS phase, the most critical thing is to find the earliest insertion point a and the latest insertion point b of the selected component. In this paper, the decimal interference matrix IM_{10+} is used to find a and b. The pseudo-code of NSO is shown in Fig. 3.

Figure 4 provides a schematic diagram of NSO. Firstly, selecting component 1 to change its position and disassembly direction in the original component sequence X_{original} (3, 5, 4, 2, 1, 6, 8, 7). Secondly, obtaining the earliest insertion point a and the latest insertion point b through the interference matrix IM_{10+}. Thirdly, randomly selecting an insertion point in $[a, b]$ and inserting component 1 into the position. Finally, a new component sequence X_{new} (3, 5, 1, 4, 2, 6, 8, 7) and its corresponding direction sequence D_{new} ($+x$, $+y$, $+x$, $-x$, $-x$, $+z$, $+x$, $+y$) are generated.

3.5 Guided search operator

α, β, and δ wolves execute GWO to guide the wolf group to hunt prey. X represents a guided wolf in the population. The three wolves try to improve the fitness value of X wolf by sharing their positions to lead it to move toward them.

The new solution can be generated according to α, β, and δ wolves, and X wolves using GSO. Firstly, the lowest bound of the fitness value f_{lb} is given according to the best disassembly sequence in the population. Then, four self-adaptive parameters, named self-updating factor p_s, α-guiding factor p_α, β-guiding factor p_β, and δ-guiding factor p_δ, are used for modifying X solution. Each parameter can be calculated according to the fitness value of X wolf f_X, the fitness value of α wolf f_α, the fitness value of β wolf f_β, the fitness value of δ wolf f_δ and the lowest bound of the fitness value f_{lb}. If the fitness value of a solution is lower, its guiding factor is higher. According to

Table 6 Comparison results of the vise

Algorithms	MGWO	GA	ACO	IHS	STLBO	SSO
Best fitness	**3.7500**	**3.7500**	**3.7500**	4.0000	4.0000	4.0000
Minimum	**3.7500**	**3.7500**	**3.7500**	4.0000	4.0000	4.0000
Maximum	**4.2500**	4.7500	4.5000	5.0000	5.0000	5.0000
Median	**3.7500**	4.2500	4.2500	4.2500	4.2500	4.2500
Mean	**3.8667**	4.2167	4.1167	4.5500	4.5500	4.4417
Average computation time (s)	**7.05**	7.15	7.69	7.77	9.55	8.86

The optimal value is indicated in bold

Table 7 Wilcoxon rank-sum test of the vise

Algorithms	Significance	*p*-value
GA	1	3.0421E−06
ACO	1	3.5517E−05
HS	1	6.4773E−11
STLBO	1	1.0286E−10
SSO	1	2.1573E−11

the pseudo-code of GSO shown in Fig. 5, we can see the composition of the new X wolf is more from the wolves with the higher value of guiding factor.

$$g(X) = \frac{f_X}{f_X - f_{\text{lb}}} \tag{24}$$

$$p_s = \frac{g(f_s)}{g(f_s) + g(f_\alpha) + g(f_\beta) + g(f_\delta)} \tag{25}$$

$$p_\alpha = \frac{g(f_\alpha)}{g(f_s) + g(f_\alpha) + g(f_\beta) + g(f_\delta)} \tag{26}$$

$$p_\beta = \frac{g(f_\beta)}{g(f_s) + g(f_\alpha) + g(f_\beta) + g(f_\delta)} \tag{27}$$

$$p_\delta = \frac{g(f_\delta)}{g(f_s) + g(f_\alpha) + g(f_\beta) + g(f_\delta)} \tag{28}$$

Finally, the precedence preservative crossover (PPX) methodology [18] is used to ensure the precedence relationship between components. The PPX operator passes on the precedence relationship based on two disassembly sequences to a new sequence while ensuring no new precedence relationships are introduced. The pseudo-code of GSO is shown in Fig. 5.

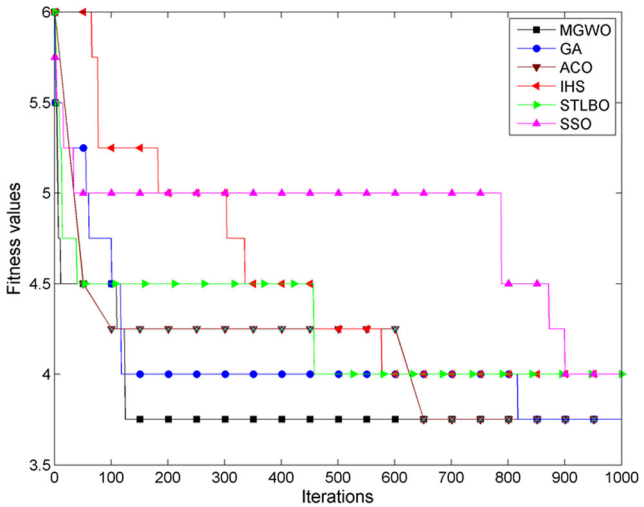

Fig. 15 Convergence curves of the vise

In the GSO procedure, the new disassembly component sequence X_{new} and its corresponding disassembly direction sequence D_{new} are generated by choosing and setting the components one by one from X, α, β, and δ. Firstly, the probability p is randomly generated in the range of [0, 1], compared with p_s, p_α, p_β, p_δ. If p is less than p_s, the leftmost component in X is pushed back to X_{new}; if p is less than $p_s + p_\alpha$, the leftmost component in α is pushed back to X_{new}; if p is less than $p_s + p_\alpha + p_\beta$, the leftmost component in β is pushed back to X_{new}; otherwise, the leftmost component in δ is pushed back to X_{new}. Then, the chosen component is removed from the four wolves. Meanwhile, the disassembly direction of the selected component is added to D_{new}. Finally, a new solution is generated by adding components one by one, as above. The precedence relationship between components can be preserved by this method.

An example of executing GSO to obtain a new disassembly sequence is shown in Table 1. A disassembly component sequence X_{new} (4, 3, 2, 5, 1) with its disassembly direction sequence D_{new} (+z, +x, +y, +z, +y) is generated by X (2 3 1 4 5) with D (+x, +x, +y, +y, −z), α (4, 3, 5, 1, 2) with D_α (+z, +x, +z, +y, −z), β (4, 3, 5, 2, 1) with D_β (+x, +x, +z, +y, −y), and δ (2, 3, 5, 1, 4) with D_δ (+y, +x, +z, +y, +z). We can observe that component 3 is disassembled before component 5 in all solutions. In addition, we suppose p_s = 0.1, p_α = 0.4, p_β= 0.3, p_δ = 0.2. It can be seen the precedence relationship between component 3 and component 5 is preserved in the new disassembly sequence.

4 Case studies and industrial application

This section aims to demonstrate the effectiveness of MGWO.

Firstly, in Section 4.1, a vise [36] with 11 components and a ball valve [37] with 17 components are used to verify the effectiveness of the modified strategies in MGWO.

Secondly, in Section 4.2, two case studies are implemented to present the superiority of MGWO compared with five algorithms: GA [12], SSO [14], STLBO [25], IHS [36], and ACO [23]. All of these methods have been proven to be effective and efficient in solving the sequence planning problem. To ensure the fairness of comparison, we set the population size of all algorithms to 20 and the maximum number of termination iterations to 1000. Other parameter settings can be referred to in the corresponding literatures.

Finally, a real-world engineering case is used to verify the practicality of the MGWO in Section 4.3.

All algorithms are coded in Matlab 2017a and carried out on an Intel Core 3.3-GHz PC with 4-GB memory.

Table 8 Comparison results of the ball valve

Algorithms	MGWO	GA	ACO	IHS	STLBO	SSO
Best fitness	**3.7500**	**3.7500**	**3.7500**	**3.7500**	**3.7500**	4.0000
Minimum	**3.7500**	**3.7500**	**3.7500**	**3.7500**	**3.7500**	4.0000
Maximum	**3.7500**	5.0000	4.2500	5.0000	5.0000	5.0000
Median	**3.7500**	4.2500	4.0000	4.2500	4.2500	4.2500
Mean	**3.7500**	4.1667	4.0500	4.2000	4.2750	4.3833
Average computation time (s)	**7.21**	7.57	7.84	8.35	11.84	8.43

The optimal value is indicated in bold

4.1 Effectiveness of proposed strategies in MGWO

4.1.1 Case 1: vise

In the subsection, a vise is used to test the effectiveness of the proposed strategies. The assembly view of the vise is shown in Fig. 6, and its corresponding interference and support matrices, material, and tool sets are offered in Fig. 7. MGWO is compared with its variations, including not using NSO (MGWO-1), not using GSO (MGWO-2), α, β, and δ wolves do not participate in the GS phase (MGWO-3), and δ wolf is the third-best wolf, not a newly generated one (MGWO-4). We set the population size to 20 and the maximum number of termination iterations to 1000. After 30 independent runs, the comparison results of different strategies are shown in Table 2. The average computation time is the average time taken by various strategies to obtain the optimal solution. The box plot of best fitness values obtained by different strategies in 30 runs is shown in Fig. 8.

It can be observed that MGWO, MGWO-1, and MGWO-3 can obtain the optimal solution, but the mean of fitness values (3.8667) of MGWO is better (lower) than that of other strategies. Figure 8 presents the box plot of fitness values of the vise in 30 independent runs, showing that the solutions obtained by MGWO are more concentrated than its variants. MGWO can obtain the optimal value (3.7500) with the probability of 63.33%. MGWO-1 and MGWO-3 can obtain the optimal value with the probability of 30.00%. Additionally, MGWO-2 and MGWO-4 cannot obtain the optimal value. Compared with MGWO-1, MGWO makes α, β, and δ wolves have more opportunities to get the optimal solution through self-updating using NSO. Compared with MGWO-2, MGWO allows each wolf to be guided by the best wolves instead of inefficiently searching for the prey by itself. Compared with MGWO-3, in MGWO, α, β, and δ wolves would exchange their prey information, enabling them to move together toward the prey. Compared with MGWO-4, δ wolf is a new one in MGWO, increasing the diversity of the population and avoids the algorithm from falling into the local optimum. Figure 9 presents the convergence curves of MGWO compared with four other strategies with the optimal solution in 30 independent runs. It reveals that MGWO has the highest probability of obtaining the optimal solution and converges faster to the optimal solution than other strategies. In Table 3, the Wilcoxon rank-sum test with a significant level of 0.05 is used to measure the significance between MGWO with other strategies. Significance value = 1 indicates the result obtained by MGWO is significantly different from the other strategies used in comparison; significance value = 0 represents the result obtained by MGWO and is not substantially different from the other strategies used in the comparison. Obviously, our proposed modified strategies are

Table 9 Wilcoxon rank-sum test of the ball valve

Algorithms	Significance	*p*-value
GA	1	5.1021E−10
ACO	1	1.3274E−09
HS	1	2.7086E−12
STLBO	1	1.5967E−10
SSO	1	8.7051E−13

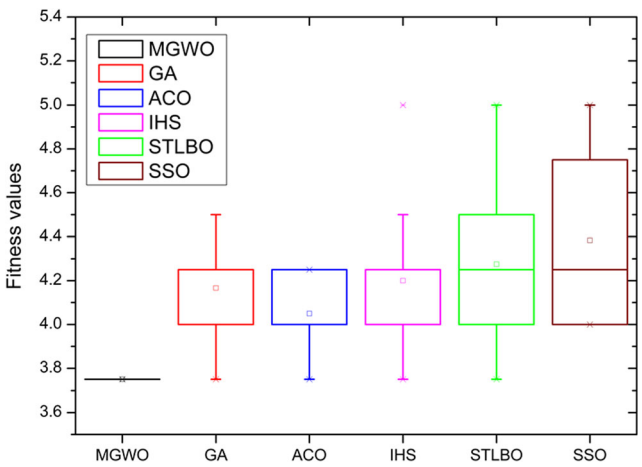

Fig. 16 Box plot of ball valve

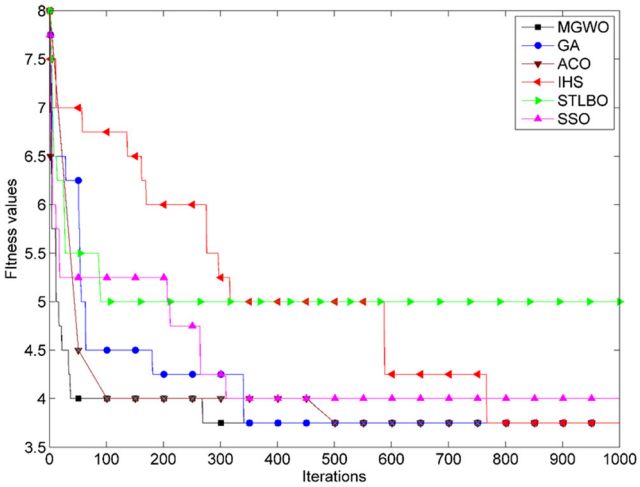

Fig. 17 Convergence curves of ball valve

effective, and the performance of MGWO is significantly better than its variants.

4.1.2 Case 2: ball valve

In this case, to further measure the significance of our proposed strategies, we compare MGWO with MGWO-1, MGWO-2, MGWO-3, and MGWO-4 for another case, the ball valve [37] with 17 components. The assembly view of the ball valve is shown in Fig. 10, and its corresponding interference and support matrices, material, and tool sets are offered in Fig. 11. We set the population size to 20 and the maximum number of termination iterations to 1000. After 30 independent runs, the comparison results of different strategies are shown in Table 4. The average computation time is the average time taken by various strategies to obtain the optimal solution. The box plot of best fitness values obtained by different strategies in 30 runs is shown in Fig. 12.

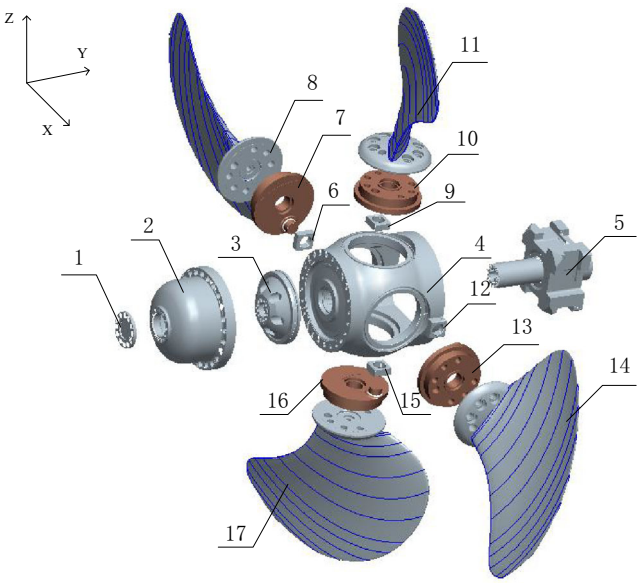

Fig. 18 Exploded view of azimuth thruster propeller

Table 4 shows that all strategies can obtain the optimal solution, but the mean of fitness values (3.7500) of MGWO is better than that of other strategies. Figure 12 presents the box plot of fitness values of the ball valve in 30 independent runs, showing that the solutions obtained by MGWO are more concentrated than those of other algorithms. MGWO, MGWO-1, MGWO-2, MGWO-3, and MGWO-4 can reach the optimal value (3.7500) with the probability of 100%, 70%, 13.33%, 43.33%, and 20%. Figure 13 shows the convergence curves of MGWO compared with other strategies with the optimal solution in 30 independent runs. It reveals that compared with MGWO-1, MGWO-2, MGWO-3, and MGWO-4, MGWO has a higher probability and faster convergence to obtain the optimal solution. Table 5 uses the Wilcoxon rank-sum test with a significant level of 0.05 to measure the significance between MGWO with other strategies. It shows that MGWO outperforms other strategies in solving the disassembly sequence of the ball valve.

4.2 Case studies and comparisons with other algorithms

4.2.1 Case 1: vise

In this subsection, the vise introduced in Section 4.1.1 is used to test the performance of MGWO. Five classic algorithms, GA, ACO, SSO, STLBO, and IHS, are used to compare with MGWO. Like Section 4.1, we set the population size to 20 and the maximum number of termination iterations to 1000. After 30 independent runs, Table 6 presents the comparison results of the six algorithms. The average computation time is the average time taken by algorithms to get the optimal disassembly sequence. The box plot of the distribution of best fitness in 30 runs is shown in Fig. 14.

We can observe from Table 6 that only MGWO, GA, and ACO can obtain the optimal solution, and the mean of fitness values (3.8667) of MGWO is better than that of other algorithms. Figure 14 also presents that the solutions obtained by MGWO are better than other algorithms. MGWO, GA, and ACO can receive the optimal solution (3.7500) with the probability of 63.33%, 13.33%, and 16.67%. Additionally, other algorithms cannot get the optimal solution. The optimal disassembly component sequence is (7, 6, 9, 1, 11, 10, 8, 3, 4, 2, 5), and its corresponding disassembly direction sequence is $(-y, -y, +z, +y, +z, +z, +z, +z, +z, +z, +z)$. Figure 15 presents the convergence curves of all algorithms with the optimal solution in 30 independent runs. It reveals that MGWO has the highest probability of obtaining the optimal solution and converges to the optimal solution with the least number of iterations in all algorithms. In Table 7, the Wilcoxon rank-sum test with the significant level of 0.05 is used to test the

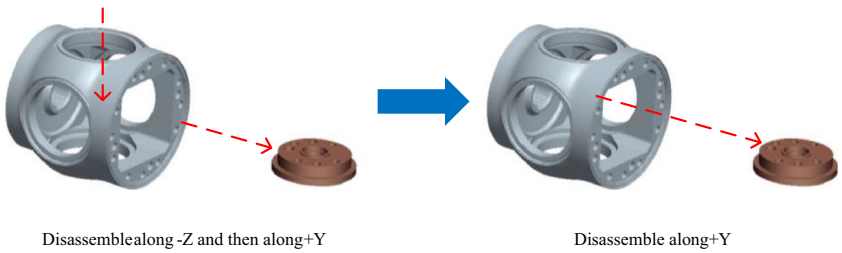

Fig. 19 Simplification of disassembling crank

significance between MGWO and other algorithms. It indicates that MGWO performs better than the other algorithms in solving the disassembly sequence of the vise.

4.2.2 Case 2: ball valve

To further measure the effectiveness of MGWO, we compare the algorithm with GA, ACO, SSO, STLBO, and IHS for the ball valve introduced in Section 4.1.2. We set the population size to 20 and the maximum number of termination iterations to 1000. After 30 independent runs, the experimental results of all algorithms are presented in Table 8. The average computation time is the average time taken by algorithms to obtain the optimal solution.

Table 8 shows that all algorithms can obtain the optimal solution except SSO, and the mean of fitness values (3.7500) of MGWO is better than that of other algorithms. Figure 16 presents the box plot of fitness values of the ball valve in 30 independent runs, showing that the solutions obtained by MGWO are more concentrated than other algorithms. MGWO, GA, ACO, IHS, and STLBO can achieve the optimal value (3.7500) with the probability of 100%, 16.67%, 20.00%, 3.33%, and 13.33%, respectively. In addition, SSO cannot reach optimal value. The optimal disassembly component sequence is (16, 17, 11, 10, 8, 9, 12, 6, 13, 7, 5, 3, 4, 15, 14, 2, 1), and its corresponding disassembly direction sequence is ($+z$, $+z$, $-y$, $-y$, $-y$, $-y$, $-y$, $-y$, $-y$, $-y$, $-y$, $-y$, $-y$, $-y$, $-y$, $-y$). Figure 17 shows the convergence curves of all algorithms with the optimal solution in 30 independent runs. It reveals that MGWO has the highest probability and the fastest convergence speed to obtain the optimal solution in all algorithms. Table 9 uses the Wilcoxon rank-sum test with a

$$\begin{bmatrix}
0&7&2&2&2&0&0&0&0&0&0&0&0&0&0&0&0\\
5&0&7&7&7&2&2&2&2&2&2&2&2&2&2&2&2\\
0&5&0&2&7&2&2&0&2&0&2&2&0&2&2&0&2\\
0&5&0&0&7&7&7&3&7&7&7&7&7&7&7&7&6\\
0&5&5&5&0&1&1&0&5&5&1&5&5&4&4&4&0\\
0&0&0&5&7&0&1&0&0&1&1&4&4&4&0&0&0\\
0&0&0&5&7&7&0&3&4&5&1&4&4&4&4&4&0\\
0&0&0&7&4&4&7&0&4&4&5&4&4&4&4&4&4\\
0&0&0&5&6&0&0&0&0&5&1&0&4&4&0&0&0\\
0&0&0&5&6&0&0&0&6&0&7&0&4&4&0&0&0\\
0&0&0&6&0&0&0&0&0&6&0&0&0&4&0&0&0\\
0&0&0&5&3&0&0&0&0&1&1&0&5&4&0&0&0\\
0&0&0&5&3&0&0&0&0&1&1&3&0&7&0&0&0\\
0&0&0&3&0&0&0&0&0&0&1&0&3&0&0&0&0\\
0&0&0&5&7&0&0&0&1&1&1&0&4&4&0&4&0\\
0&0&0&5&7&1&1&0&1&1&1&1&5&4&7&0&6\\
0&0&0&7&1&1&1&1&1&1&1&1&1&5&1&7&0
\end{bmatrix}$$

(a)

$$\begin{bmatrix}
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&1&1&0&0&0&1&0&0&1&0&0&1&0&0&1&0\\
0&1&1&1&0&1&0&0&1&0&0&1&0&0&1&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&1&0&1&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&1&0&1&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&0&1&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0
\end{bmatrix}$$

(b)

Component	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Material	A	A	A	A	A	B	C	D	B	C	D	B	C	D	B	C	D

(c)

Component	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Tool	A	B	C	B	B	A	C	D	A	C	D	A	C	D	A	C	D

(d)

Fig. 20 Matrices, material, and tool sets of azimuth thruster propeller. **a** Interference matrix. **b** Support matrix. **c** Material set. **d** Tool set

Table 10 Comparison results of the azimuth thruster propeller

Algorithms	MGWO	GA	ACO	IHS	STLBO	SSO
Best fitness	**5.5000**	**5.5000**	**5.5000**	**5.5000**	**5.5000**	5.7500
Minimum	**5.5000**	**5.5000**	**5.5000**	**5.5000**	**5.5000**	5.7500
Maximum	**6.0000**	6.5000	6.2500	6.5000	6.2500	6.2500
Median	**5.5000**	5.7500	5.7500	5.7500	5.5000	5.7500
Mean	**5.5669**	5.8667	5.7083	5.8333	5.6000	5.8083
Average computation time (s)	**9.05**	9.86	10.67	11.45	12.06	10.19

significant level of 0.05 to measure the significance between MGWO with other algorithms. It indicates that MGWO significantly outperforms the other algorithms in solving the disassembly sequence of the ball valve.

4.3 Industrial application

4.3.1 Application 1: azimuth thruster propeller

Used to verify the practicality of MGWO, azimuth thruster propeller [36], a vital product of a Chinese boat company, is employed in this section. Azimuth thruster is the critical equipment of the marine dynamic positioning system. Propeller is used to generate thrust for the azimuth thruster. Many components of the azimuth thruster propeller can be recycled and reused at the end of life. A good disassembly sequence can dramatically improve the recovery efficiency of these components, so the DSP of the azimuth thruster propeller has significant research value. The azimuth thruster propeller has been modified for simplification to satisfy the requirement of disassembly modeling. The exploded view of the azimuth thruster propeller is shown in Fig. 18. The simplification of the disassembling crank is shown in Fig. 19, and the matrices of the azimuth thruster propeller are presented in Fig. 20.

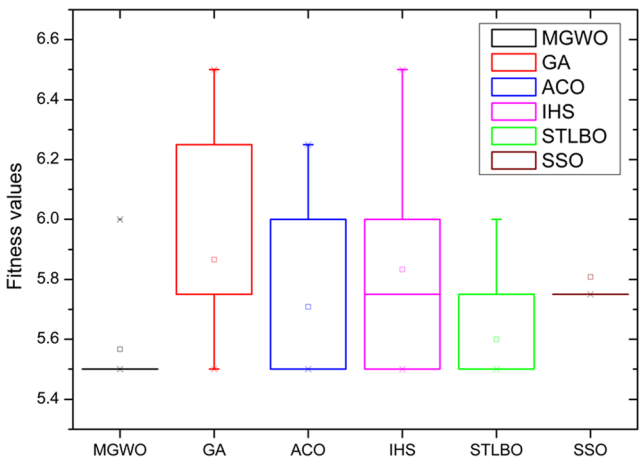

Fig. 21 Box plot of azimuth thruster propeller

The proposed MGWO is applied to the actual case, and GA, ACO, SSO, STLBO, and IHS are used to compare with MGWO. We set the population size to 20 and the maximum number of termination iterations to 1000. After 30 independent runs, the statistical results of all algorithms are summarized in Table 10, and their corresponding box plots are presented in Fig. 21. The mean of fitness values (5.5669) of MGWO is better (lower) than that of other algorithms. MGWO, GA, ACO, IHS, and STLBO can obtain the optimal solution with the probability of 76.67%, 20.00%, 50.00%, 30.00%, and 73.33%, respectively. In addition, SSO cannot obtain the optimal solution. It can be observed that MGWO performs better than the compared algorithms in terms of the probability of obtaining the optimal solution and the concentration of solutions in this case. The results show that MGWO is very suitable to solve the DSP problem.

Figure 22 shows the convergence curves of all algorithms with the optimal solution in 30 independent runs. It demonstrates that MGWO has the highest probability and the fastest convergence speed to obtain the optimal solution in all algorithms. In Table 11, the Wilcoxon rank-sum test with a significant level of 0.05 is used to measure the significance between MGWO with other algorithms. It shows that MGWO performs significantly better than other algorithms except for STLBO. MGWO can reach the optimal value of 5.5000 with the corresponding disassembly component sequence (1, 2, 3, 8, 17, 14, 11, 5, 12, 9, 6, 15, 13, 16, 7, 10, 4) and the disassembly direction sequence ($-y$, $-y$, $-y$, $-x$, $-z$, $+x$, $+z$, $+y$, $+y$, $+y$, $+y$, $+y$, $+y$, $+y$, $+y$, $+y$, $+y$), which is

Table 11 Wilcoxon rank-sum test of the azimuth thruster propeller

Algorithms	Significance	*p*-value
GA	1	6.2885E−06
ACO	1	1.4992E−02
HS	1	6.4012E−05
STLBO	0	6.7427E−01
SSO	1	9.9657E−09

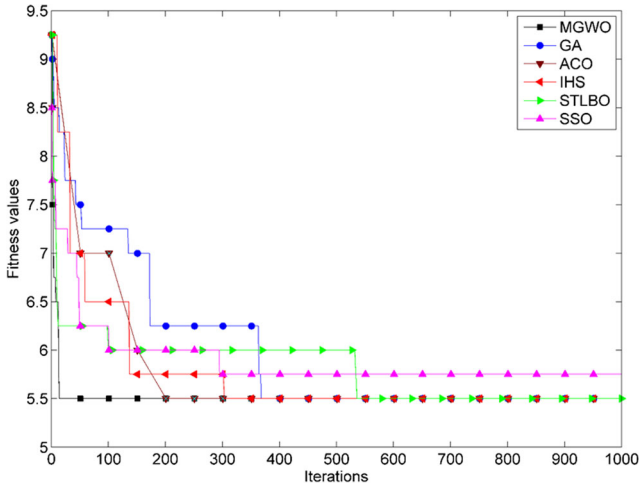

Fig. 22 Convergence curves of azimuth thruster propeller

accorded with the actual industrial disassembly sequence. Thus, the real-world application demonstrates the validity and practicality of the proposed method.

4.3.2 Application 2: robot arm

In this section, we use a robot arm [38] to validate the practicability of MGWO further. The robot arm consists of 30 functional components. It is more complex than the azimuth thruster propeller. The assembly of the robot arm is shown in Fig. 23, and its corresponding interference and support matrices, material, and tool sets are provided in Fig. 24.

We compare MGWO with GA, ACO, SSO, STLBO, and HIS on the robot arm. The population size is set to 20, and the maximum number of termination iterations is set to 1000. Table 12 gives the statistical results of all algorithms, and Fig. 25 provides their corresponding box plots. The mean of fitness values (9.6250) of MGWO is better than that of ACO, IHS, STLBO, and SSO and is slightly worse than that of GA. The optimal value (8.7500) can only be obtained by MGWO, and cannot be found by the other algorithms. Besides, the computation time of MGWO is shorter than that of all other algorithms. The experimental results demonstrate that MGWO outperforms the other algorithms in both solution quality and computational efficiency.

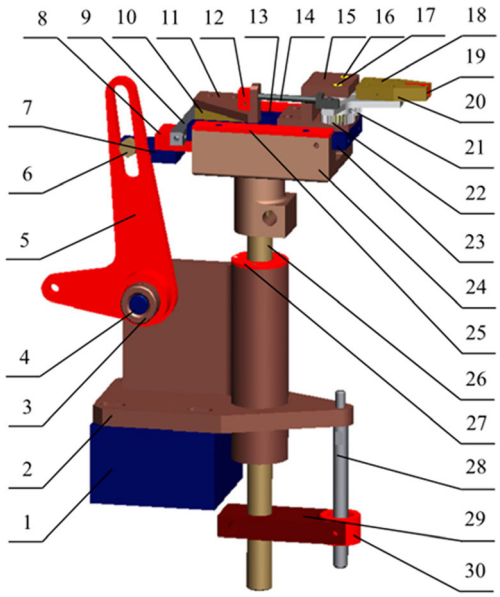

Fig. 23 Assembly drawing of the robot arm

Figure 26 shows the convergence curves of all algorithms with the optimal solution in 30 independent runs. It presents that only MGWO can converge to the optimal solution. In Table 13, the Wilcoxon rank-sum test with a significant level of 0.05 is used to measure the significance between MGWO with other algorithms. It indicates that MGWO significantly outperforms other algorithms except for GA. MGWO can obtain the optimal disassembly component sequence (26, 30, 29, 28, 14, 3, 27, 5, 25, 19, 13, 12, 11, 9, 7, 4, 6, 20, 18, 10, 17, 16, 22, 21, 15, 2, 23, 1, 8, 24) and its corresponding disassembly direction sequence ($-z$, $-z$, $-z$, $-z$, $+y$, $+x$, $+x$, $+x$, $+x$, $+z$, $+z$, $-x$, $-x$, $-x$, $-x$, $-x$, $-x$, $+z$, $+z$, $+z$, $+z$, $+z$,$+y$,$+y$, $+y$, $+y$, $+y$, $+y$, $+z$). The two sequences are accorded with the actual industrial disassembly sequence. Hence, the industrial application of the robot arm verifies the practicality of our proposed method.

5 Discussion

The above experimental results demonstrate that MGWO outperforms the other compared algorithms. This is detailed below. In general, MGWO outperforms the other algorithms in both convergence speed and solution quality.

In Section 4.1, Tables 2 and 3 show that MGWO obtains the optimal solution with the highest probability in all strategies. For the vise, MGWO receives the optimal solution with the probability of 63.33%, but the probability of other strategies obtaining the optimal solution is less than 30%. MGWO obtains the optimal solution with the probability of 100% for the ball valve, but the probability of other strategies obtaining the optimal solution is less than 70%. Figures 9 and 13 illustrate that the convergence speed of MGWO is faster than that of other strategies. These experimental results suggest that the effectiveness of NSO and GSO operators.

In Section 4.2, Tables 6 and 8 show that MGWO obtains the optimal solution with the highest probability in all algorithms. For the vise, MGWO obtains the optimal solution with the probability of 63.33%, but the probability of other algorithms obtaining the optimal solution is less than 16.67%. MGWO obtains the optimal solution with the probability of 100% for the ball

$$\begin{bmatrix}
0&1&1&1&1&1&1&1&1&1&1&0&1&0&0&0&0&0&0&0&0&0&1&1&1&2&0&2&0&0\\
0&0&5&3&5&1&1&1&1&1&1&1&1&1&1&1&1&0&0&0&1&1&1&1&1&6&1&6&0&0\\
0&2&0&3&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&3&3&0&3&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&2&0&0&0&0\\
0&2&4&3&0&3&3&3&2&2&2&0&0&0&2&0&0&2&2&2&2&2&2&2&0&2&2&0&0&0\\
0&0&0&0&3&0&3&2&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&2&0&0&0&0&0&0\\
0&0&0&0&4&7&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&2&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&1&1&1&0&0&0&1&0&0&0&0&0&0&0&1&4&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&2&0&0&0&0&0&0&0&0&0&0&0&0&2&2&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&1&0&0&0&2&0&0&0&0&0&2&2&6&4&4&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&1&0&7&2&2&2&0&0&0&2&2&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&2&0&0&7&0&0&0&0&0&0&0&2&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&4&0&0&0&0&4&4&4&0&0&0&4&4&0&0&4&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&5&5&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&4&0&7&7&2&2&2&3&3&0&0&4&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&4&6&0&4&2&0&0&6&4&0&0&4&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&4&6&0&0&0&0&2&0&6&0&0&4&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&6&4&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&4&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&4&1&6&4&3&2&0&0&6&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&5&1&0&6&0&0&3&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&4&1&0&1&0&1&1&1&0&0&0&1&1&0&4&1&0&0&0&0&0\\
0&0&0&0&0&0&0&5&0&5&1&1&1&1&1&1&1&0&0&0&1&1&5&0&1&6&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&1&1&0&1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\
0&6&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&7&0&0&6&2&6&2\\
0&0&0&0&0&0&0&1&0&1&0&0&0&0&1&0&0&0&0&0&0&0&1&1&0&6&0&0&0&0\\
0&7&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&1&1&1&1&0&0&0&0&4&6\\
1&1&0&0&0&0&0&1&0&1&0&0&0&0&1&1&1&0&0&0&1&1&1&1&0&6&1&6&0&6\\
0&1&0&0&0&0&0&1&0&0&0&0&0&0&1&1&1&0&0&0&1&1&1&1&0&0&0&6&4&0
\end{bmatrix}$$

(a)

$$\begin{bmatrix}
0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&0&0\\
0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&1&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&1&1&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&1&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&1&0&1&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0
\end{bmatrix}$$

(b)

Component	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
Material	A	B	B	A	C	D	A	C	E	D	B	C	C	E	B
	F	F	D	C	D	G	G	A	H	C	D	C	E	I	C

(c)

Component	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
Material	B	B	E	C	A	C	A	B	A	A	A	A	A	C	A
	C	C	A	A	A	A	A	B	B	A	D	E	C	B	E

(d)

Fig. 24 Matrices, material, and tool sets of the robot arm. **a** Interference matrix. **b** Support matrix. **c** Material set. **d** Tool set

Table 12 Comparison results of the azimuth thruster propeller

Algorithms	MGWO	GA	ACO	IHS	STLBO	SSO
Best fitness	**8.7500**	9.0000	9.0000	9.0000	9.2500	9.2500
Minimum	**8.7500**	9.0000	9.0000	9.0000	9.2500	9.2500
Maximum	11.2500	**10.0000**	11.5000	12.5000	12.7500	12.2500
Median	**9.5000**	**9.5000**	10.3750	10.7500	11.0000	11.0000
Mean	9.6250	**9.5333**	10.2667	10.7500	10.9750	11.0167
Average computation time (s)	**14.72**	16.29	17.07	17.62	19.26	15.67

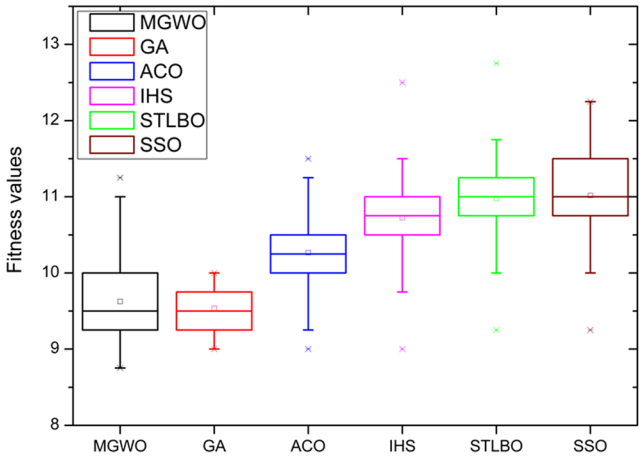

Fig. 25 Box plot of the robot arm

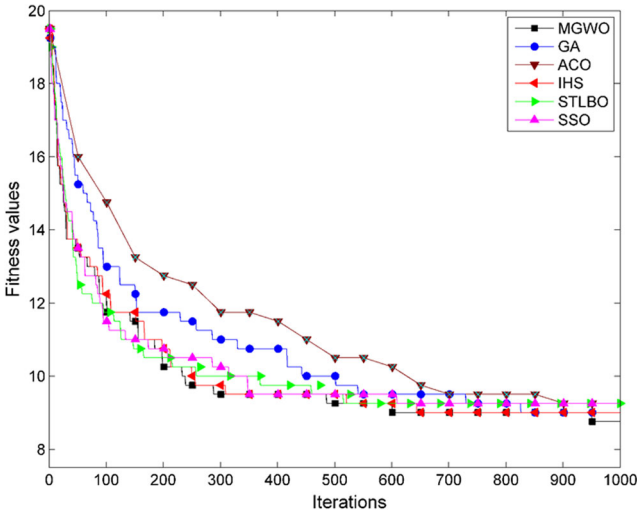

Fig. 26 Convergence curves of the robot arm

valve, but the probability of other strategies obtaining the optimal solution is less than 20.00%. Figures 15 and 17 illustrate that MGWO converges faster than other algorithms. The computation results verify that MGWO is more appropriate for solving the DSP problem than other compared algorithms.

In Section 4.3, two real-world industrial applications prove that MGWO can obtain the disassembly sequence following practical production. MGWO receives the optimal solution with the probability of 76.67% for the azimuth thruster propeller, but the probability of other algorithms obtaining the optimal solution is less than 73.33%. Only MGWO can obtain the optimal solution for the robot arm, and other comparison algorithms cannot find it. Figure 22 illustrates that MGWO converges faster than other algorithms. Figure 26 shows that only MGWO can converge to the optimal solution.

The outstanding performance of MGWO for the DSP problem mostly depends on NSO and GSO operators. The two operators can well balance local search and global search. Additionally, the hierarchy and hunting mechanism allows the population to evolve in a good direction without falling into the local optimum. Another important outcome is that MGWO can be well applied to engineering cases, proving our proposed algorithm can effectively guide the disassembly practice.

Table 13 Wilcoxon rank-sum test of the robot arm

Algorithms	Significance	*p*-value
GA	0	9.2250E−01
ACO	1	4.0862E−04
HS	1	2.2653E−06
STLBO	1	1.2842E−07
SSO	1	3.1218E−08

6 Conclusions and future researches

In this paper, an MGWO algorithm is proposed for solving the DSP problem. Three key operators, FSG, NSO, and GSO, are designed for the problem. First, two engineering cases are employed to test the effectiveness of these operators. Then, they and two real-world applications are used to test the validity of MGWO. The experimental results verify that MGWO can solve the DSP problem effectively. The advantages of MGWO are as follows:

- Three key operators are designed in this paper: FSG, NSO, and GSO. FSG generates the feasible disassembly sequence; NSO and GSO balance both the local search and global search.
- The implementation of MGWO is simple. Moreover, all parameters of MGWO are self-adapted, so its performance is stable.
- Due to the hierarchy and hunting mechanism of the wolf group, the convergence speed of MGWO is very fast.

Although MGWO shows a good performance for solving the DSP problem, the algorithm still has some limitations. One is that only three leading wolves execute the neighborhood search operator, and the remaining wolves obey the three wolves and lack self-exploration. Another limitation is that our mathematical model considers only one constraint of component interferences, but other constraints also need to be introduced into the model. Concerning future work, the first research direction is to design a novel self-search mechanism for remaining wolves to enhance their self-explore ability. The

second promising research direction is to develop a new DSP model considering disassembly process constraints.

Author contribution All the authors designed research, performed research, analyzed data, and wrote the paper.

Funding This work was supported by the National Natural Science Foundation of China under Grant 51825502 and Grant 51721092, by the Natural Science Foundation of Hubei Province under Grant 2018CFA078, and by the Program for HUST Academic Frontier Youth Team under Grant 2017QYTD04.

Data availability All data generated or analyzed during this study are included in this paper.

Declarations

Ethical approval Not applicable.

Consent to participate Not applicable.

Consent for publication Not applicable.

Competing interests The authors declare no competing interests.

References

1. Zhou Z, Liu J, Pham DT, Xu W, Ramirez FJ, Ji C, Liu Q (2019) Disassembly sequence planning: recent developments and future trends. Proc Inst Mech Eng B J Eng Manuf 233(5):1450–1471
2. Lambert AJ (2003) Disassembly sequencing: a survey. Int J Prod Res 41(16):3721–3759
3. Lambert AJ (2007) Optimizing disassembly processes subjected to sequence-dependent cost. Comput Oper Res 34(2):536–551
4. Cappelli F, Delogu M, Pierini M, Schiavone F (2007) Design for disassembly: a methodology for identifying the optimal disassembly sequence. J Eng Des 18(6):563–575
5. Rai R, Rai V, Tiwari M, Allada V (2002) Disassembly sequence generation: a Petri net based heuristic approach. Int J Prod Res 40(13):3183–3198
6. Kuo TC (2013) Waste electronics and electrical equipment disassembly and recycling using Petri net analysis: considering the economic value and environmental impacts. Comput Ind Eng 65(1): 54–64
7. Li HJ, Jiang J, Wang YF (2013) Disassembly sequence planning based on extended interference matrix and genetic algorithm. Comput Eng Des 34(3):1064–1068
8. Zhu B, Sarigecili MI, Roy U (2013) Disassembly information model incorporating dynamic capabilities for disassembly sequence generation. Robot Comput Integr Manuf 29(5):396–409
9. Ma YS, Jun HB, Kim HW, Lee DH (2011) Disassembly process planning algorithms for end-of-life product recovery and environmentally conscious disposal. Int J Prod Res 49(23):7007–7027
10. Behdad S, Berg LP, Thurston D, Vance J (2014) Leveraging virtual reality experiences with mixed-integer nonlinear programming visualization of disassembly sequence planning under uncertainty. J Mech Des 136(4):MD-12-1247
11. Kim HW, Lee DH (2017) An optimal algorithm for selective disassembly sequencing with sequence-dependent set-ups in parallel disassembly environment. Int J Prod Res 55(24):7317–7333
12. Hui W, Dong X, Duan G (2008) A genetic algorithm for product disassembly sequence planning. Neurocomputing 71(13-15):2720–2726
13. Go TF, Wahab DA, Rahman MA, Ramli R, Hussain A (2012) Genetically optimised disassembly sequence for automotive component reuse. Expert Syst Appl 39(5):5409–5417
14. Yeh WC (2011) Optimization of the disassembly sequencing problem on the basis of self-adaptive simplified swarm optimization. IEEE Trans Syst Man Cybern Syst Hum 42(1):250–261
15. Yeh WC (2012) Simplified swarm optimization in disassembly sequencing problems with learning effects. Comput Oper Res 39(9):2168–2177
16. Percoco G, Diella M (2013) Preliminary evaluation of artificial bee colony algorithm when applied to multi objective partial disassembly planning. Res J Appl Sci 6(17):3234–3243
17. Tian G, Zhou M, Li P (2017) Disassembly sequence planning considering fuzzy component quality and varying operational cost. IEEE Trans Autom Sci Eng 15:748–760
18. Kongar E, Gupta SM (2006) Disassembly sequencing using genetic algorithm. Int J Adv Manuf Technol 30(5-6):497–506
19. Tseng HE, Chang CC, Lee SC, Huang YM (2018) A block-based genetic algorithm for disassembly sequence planning. Expert Syst Appl 96:492–505
20. Li B, Li C, Cui X, Lai X, Ren J, He Q (2020) A disassembly sequence planning method with team-based genetic algorithm for equipment maintenance in hydropower station. IEEE Access 8: 47538–47555
21. Tseng YJ, Yu FY, Huang FY (2011) A green assembly sequence planning model with a closed-loop assembly and disassembly sequence planning using a particle swarm optimization method. Int J Adv Manuf Technol 57(9-12):1183–1197
22. Gulivindala AK, Bahubalendruni MR, Varupala SP, Ravi C (2021) Exponential moving average modelled particle swarm optimization algorithm for efficient disassembly sequence planning towards practical feasibility. Int J Performability Eng 17(3):289
23. Tseng HE, Chang CC, Lee SC, Huang YM (2019) Hybrid bidirectional ant colony optimization (hybrid BACO): an algorithm for disassembly sequence planning. Eng Appl Artif Intell 83:45–56
24. Xing Y, Wu D, Qu L (2021) Parallel disassembly sequence planning using improved ant colony algorithm. Int J Adv Manuf Technol 113(7):2327–2342
25. Xia K, Gao L, Li W, Chao KM (2014) Disassembly sequence planning using a simplified teaching–learning-based optimization algorithm. Adv Eng Inform 28(4):518–527
26. Gunji AB, Deepak B, Bahubalendruni CR, Biswal DBB (2018) An optimal robotic assembly sequence planning by assembly subsets detection method using teaching learning-based optimization algorithm. IEEE Trans Autom Sci Eng 15(3):1369–1385
27. Liu J, Zhou Z, Pham DT, Xu W, Ji C, Liu Q (2020) Collaborative optimization of robotic disassembly sequence planning and robotic disassembly line balancing problem using improved discrete Bees algorithm in remanufacturing. Robot Comput Integr Manuf 61: 101829
28. Xu W, Tang Q, Liu J, Liu Z, Zhou Z, Pham DT (2020) Disassembly sequence planning using discrete Bees algorithm for human-robot collaboration in remanufacturing. Robot Comput Integr Manuf 62: 101860
29. Mirjalili S, Mirjalili SM, Lewis A (2014) Grey wolf optimizer. Adv Eng Softw 69(3):46–61
30. Mohanty S, Subudhi B, Ray PK (2015) A new MPPT design using grey wolf optimization technique for photovoltaic system under partial shading conditions. IEEE Trans Sustain Energy 7(1):181–188
31. Jayakumar N, Subramanian S, Ganesan S, Elanchezhian EB (2016) Grey wolf optimization for combined heat and power dispatch with cogeneration systems. Int J Electr Power Energy Syst 74:252–264

32. Lu C, Gao L, Li X, Xiao S (2017) A hybrid multi-objective grey wolf optimizer for dynamic scheduling in a real-world welding industry. Eng Appl Artif Intell 57:61–79
33. Gupta S, Deep K, Moayedi H, Foong LK, Assad A (2020) Sine cosine grey wolf optimizer to solve engineering design problems. Eng Comput. https://doi.org/10.1007/s00366-020-00996-y
34. Zhang S, Zhou Y, Li Z, Pan W (2016) Grey wolf optimizer for unmanned combat aerial vehicle path planning. Adv Eng Softw 99:121–136
35. Makhadmeh SN, Khader AT, Al-Betar MA, Naim S, Abasi AK, Alyasseri ZAA (2021) A novel hybrid grey wolf optimizer with min-conflict algorithm for power scheduling problem in a smart home. Swarm Evol Comput 60:100793
36. Li X, Qin K, Zeng B, Gao L, Su J (2016) Assembly sequence planning based on an improved harmony search algorithm. Int J Adv Manuf Technol 84(9-12):2367–2380
37. Li X, Qin K, Zeng B, Gao L, Wang L (2017) A dynamic parameter controlled harmony search algorithm for assembly sequence planning. Int J Adv Manuf Technol 92(9-12):3399–3411
38. Li M, Zhang Y, Zeng B, Zhou H, Liu J (2016) The modified firefly algorithm considering fireflies' visual range and its application in assembly sequences planning. Int J Adv Manuf Technol 82(5-8): 1381–1403