Disassembly sequence planning based on a modified grey wolf optimizer

Jin Xie

Huazhong University of Science and Technology

Xinyu Li

Huazhong University of Science and Technology

Liang Gao (✉ gaoliang@mail.hust.edu.cn)

Huazhong University of Science and Technology <https://orcid.org/0000-0002-1485-0722>

Research Article

Keywords: Disassembly sequence planning, end-of-life, grey wolf optimizer, real-world application

Posted Date: March 22nd, 2021

DOI: <https://doi.org/10.21203/rs.3.rs-336460/v1>

License: © ⓘ This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Version of Record: A version of this preprint was published at The International Journal of Advanced Manufacturing Technology on July 22nd, 2021. See the published version at <https://doi.org/10.1007/s00170-021-07696-x>.

Disassembly sequence planning based on a modified grey wolf optimizer

Jin Xie, Xinyu Li, Liang Gao*

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

**Corresponding author E-mail address: gaoliang@mail.hust.edu.cn*

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), for solving 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 one real-world application 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, which has garnered considerable attention in recent years owing to severe environmental problems. 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 the recovery efficiency. Lambert [2] has verified that DSP is an NP-hard problem. It is difficult to obtain the optimal disassembly sequence, even a feasible disassembly sequence considering various disassembly constraints. In recent years, DSP has always been a challenging research topic. Although the DSP problem needs to be solved urgently, most of the existing research focuses on assembly sequence planning [3-7]. The research on DSP is relatively few. Hsu [8] proposed a fuzzy knowledge-based DSP system for solving DSP. The system was derived from a fuzzy attributed Petri net. Yu et al. [9] analyzed the automobile parts, and drew a disassembly network graph to obtain optimum disassembly routes. The disassembly network graph was derived from complex AND/OR hierarchical relationships.

Kim and Lee [10] developed an integer programming model to represent disassembly sequences, and proposed a branch and bound method for obtaining the lower and upper bounds of the DSP problem. Tian et al. [11] designed an AND/OR graph for DSP considering the uncertainty of part quality and variable disassembly cost. Then, they proposed an intelligent algorithm combining artificial bee colony (ABC) with fuzzy simulation for solving the problem. Marconi et al. [12] used corrective factors in computing the effective disassembly time of the selective component. The factors were obtained by using data mining technology, derived from the observation of actual de-manufacturing activities. Compared with heuristic and data mining technologies, meta-heuristics are more commonly applied to deal with DSP. For example, Ilgin and Taşoğlu [13] proposed a genetic algorithm (GA) to determine the disassembly-to-order decision and the disassembly sequence simultaneously. Meng et al. [14] presented an improved co-evolutionary algorithm (ICA) for solving DSP. A search mechanism of endosymbiosis competition and localized interaction was introduced in their algorithm. Guo et al. [15] presented a dual-objective optimization model to represent the selective disassembly considering multi-resource constraints, and then developed a scatter search algorithm to solve the model. Additionally, many meta-heuristic algorithms, including GA [16-18], Particle Swarm Optimization (PSO) [19-21], Ant Colony Optimization (ACO) [22, 23], Teaching-Learning-Based Optimization (TLBO) [24, 25] and ABC [26-28], have also been developed to solve the DSP problem. However, the parameter settings in most of the above algorithms need constant tuning, such as acceleration constant and inertia weight in PSO, the probability of crossover and mutation in GA, pheromone evaporation rate in ACO and limited number of cycles in ABC. 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. 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 the stabilizer design of a wide-area power system [33], the path planning of an unmanned combat aerial vehicle [34], and PID controller design [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. Mirjalili et al. [29] have demonstrated that GWO is competitive to other state-of-the-art meta-heuristics such as GA, PSO, and HS.

However, GWO is designed for continuous optimization problems, so it cannot be directly applied to the DSP problem. This paper designs 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. Three new operators, i.e., feasible solution generator (FSG), neighborhood search operator (NSO), and guided search operator (GSO), are designed for the DSP problem. FSG is developed for generating feasible disassembly sequences. NSO and GSO are designed for updating disassembly sequences. Based on these three operators, MGWO can converge fast and search efficiently.

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

4 verifies the superiority of MGWO by utilizing two case studies and one real-world application. Section 5 discusses the effectiveness and superiority of MGWO. Finally, Section 6 gives the conclusion and future researches.

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. 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 [16, 20].

2.1 Geometric constraints of 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 $-x, -y, -z$ axes can be obtained by the inversion of IM_{10+} .

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

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

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.

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

2.2 Changes of disassembly directions and tools

In the process of disassembling EOL products, we found that changes of disassembly directions and tools increase extra expenses of time and cost, 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.

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

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

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

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

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

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.

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

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

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

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.

$$f_s(X) = \sum_{i=1}^n \sum_{j=i+1}^n s_{x_i x_j} \quad (10)$$

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

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

2.5 Fitness function

A disassembly sequence firstly needs to satisfy geometric constraints. According to literature [16], a linear weighted expression considering the above four indicators is designed to measure the disassembly quality for the feasible sequence.

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

Where w_d , w_t , w_m , 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.

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. Eq. 13 is expressed to the model of the encircling behavior.

$$D = |C \cdot X_p(t) - X(t)| \quad (13)$$

$$X(t+1) = X_p(t) - A \cdot D \quad (14)$$

where D denotes the distance between the prey and a wolf, X and X_p indicate the position vector of the wolf and the prey respectively, and t represents the current iteration. The vector A and C are formulated using Eqs. (15) and (16).

$$A = 2a \cdot r_1 - a \quad (15)$$

$$C = 2r_2 \quad (16)$$

where r_1 , 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 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 updating their position toward the prey. Eqs. (17-23) calculate the hunting practices of the wolf group [29].

$$D_\alpha = |C_1 \cdot X_\alpha - X| \quad (17)$$

$$D_\beta = |C_2 \cdot X_\beta - X| \quad (18)$$

$$D_\gamma = |C_3 \cdot X_\gamma - X| \quad (19)$$

$$X_1 = X_\alpha - A_1 \cdot D_\alpha \quad (20)$$

$$X_2 = X_\beta - A_2 \cdot D_\beta \quad (21)$$

$$X_3 = X_\gamma - A_3 \cdot D_\gamma \quad (22)$$

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

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 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.

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

Fig. 1. The pseudo-code of the MGWO algorithm

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). 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 next sections.

3.3 Feasible solution generator

The constraint of disassembly precedence should be satisfied in the disassembly process. In this paper, the constraint of 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.

FSG	
<hr/>	
Input: the interference matrix IM_{10+} of the EOL product	
Output: a feasible disassembly sequence	
1	Set the set C_A of components available for disassembly to $\emptyset$;
2	Set the map D_A for feasible disassembly directions of components in C_A to $\emptyset$;
3	For $i = 1$ to length of disassembly component sequence X do
4	For each component c in the EOL product do ;
5	If $c \in X$ then
6	Continue;
7	End if
8	If c can be disassembled by checking IM_{10+} then
9	Record the feasible disassembly direction set D_c of c ;
10	Push c back to C_A ;
11	Put D_c into D_A ;
12	End if
13	End for
14	Randomly choose a component c from C_A ;
15	Push the c back to X ;
16	Randomly choose a feasible disassembly direction k of c from D_A ;
17	Push the k back to disassembly direction sequence D ;
18	Set C_A to $\emptyset$;
19	Set D_A to $\emptyset$;
20	Set all the elements in row c or column c of IM_{10+} to 0;
21	End for

Fig. 2. The pseudo-code of FSG

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.

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	
1	Record the disassembly component sequence X and its corresponding disassembly direction sequence D ;
2	Randomly select a component c from X and record its disassembly direction k ;
3	Set the interference direction set I_c to $\emptyset$; // I_c includes the directions in which component c is interfaced by components preceding c
4	For $i =$ the serial number of c to 1 do
5	Record the interference direction set $\bar{D}_i$ of c interfered by component x_i through the matrix IM_{10+} ;
6	$I_c = I_c \cup \bar{D}_i$;
7	If the size of $I_c = 6$ then
8	Set the earliest insertion point a to i ;
9	Break;
10	End if
11	End for
12	For $i =$ the serial number of $c + 1$ to the length of X do
13	If c blocks the disassembly direction of x_i according to D then ;
14	Set the latest insertion point b to i ;
15	Break;
16	End if
17	End for
18	Randomly select an insertion point p_{insert} between a and b ;
19	Update X by inserting c into p_{insert} , and delete c at the original position;
20	For $i = 1$ to p_{insert} do
21	Set all the elements in row i or column i in IM_{10+} to 0;
22	End for
23	Record the feasible disassembly direction set D_c of c by checking IM_{10+} ;
24	If $p_{insert} =$ the serial number of c then
25	$D_c = D_c \setminus \{d\}$; // d is the original disassembly direction of c
26	End If
27	Randomly choose a feasible disassembly direction k from D_c ;
28	Update D by modifying the disassembly direction of c .

Fig. 3. The pseudo-code of NSO

Fig. 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.

Fig. 4. Schematic diagram of NSO

3.5 Guided search operator

α , β , and δ wolves execute GWO to guide the wolf group to hunting for the 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 the pseudo-code of GSO shown in Fig. 4, we can see the composition of the new X wolf is more from the wolves with the higher value of guiding factor.

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

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

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

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

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

Finally, the precedence preservative crossover (PPX) methodology [16] 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 that no new precedence relationships are introduced. The pseudo-code of GSO is shown in Fig. 4.

GSO	
Input: the disassembly sequences of α , β , δ , and X wolves	
Output: a new disassembly sequence	
1	Set the new disassembly component sequence X_{new} to $\emptyset$;
2	Set the new disassembly direction sequence D_{new} to $\emptyset$;
3	For $i = 1$ to length of disassembly component sequence X do
4	Randomly generate probability $p \in [0, 1]$;
5	If $p < p_s$ then
6	Push the leftmost component of X back to X_{new} ;
7	Push the leftmost direction of D back to D_{new} ;
8	Else if $p < p_s + p_\alpha$ then
9	Push the leftmost component of α back to X_{new} ;
10	Push the leftmost direction of D_α back to D_{new} ;
11	Else if $p < p_s + p_\alpha + p_\beta$ then
12	Push the leftmost component of β back to X_{new} ;
13	Push the leftmost direction of D_β back to D_{new} ;
14	Else
15	Push the leftmost component of δ back to X_{new} ;
16	Push the leftmost direction of D_δ back to D_{new} ;
17	End if
18	Remove the chosen component from X_i , α , β , δ ;
19	Remove the direction of the chosen component from D_i , D_α , D_β , D_δ ;
20	End for
21	Update X with X_{new} ;
22	Update D with D_{new} .

Fig. 4. The pseudo-code of GSO

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 $p_s, p_\alpha, p_\beta, p_\delta$. 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 Table1. 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 $D(+x, +x, +y, +y, -z)$, $\alpha(4, 3, 5, 1, 2)$ with $D_\alpha(+z, +x, +z, +y, -z)$, $\beta(4, 3, 5, 2, 1)$ with $D_\beta(+x, +x, +z, +y, -y)$ and $\delta(2, 3, 5, 1, 4)$ with $D_\delta(+y, +x, +z, +y, +z)$. We can observe that component 3 is disassembled before component 5 in all solutions. In addition, we suppose $p_s = 0.1, p_\alpha = 0.4, p_\beta = 0.3, p_\delta = 0.2$. It can be seen the precedence relationship between component 3 and component 5 is preserved in the new disassembly sequence.

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)	(<u>4</u> , 3, 5, 1, 2) (<u>+z</u> , +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)	(<u>3</u> , 5, 1, 2) (<u>+x</u> , +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)	(<u>2</u> , 5, 1) (<u>+y</u> , +z, +y)	(4, 3, 2) (+z, +x, +y)
4	0.7	(1, 5) (+y, -z)	(5, 1) (+z, +y)	(<u>5</u> , 1) (<u>+z</u> , -y)	(5, 1) (+z, +y)	(4, 3, 2, 5) (+z, +x, +y, +z)
5	0.4	(1) (+y)	(<u>1</u>) (<u>+y</u>)	(1) (-y)	(1) (+y)	(4, 3, 2, 5, 1) (+z, +x, +y, +z, +y)

Remarks: the underline presents the chosen component in each step.

4 Case studies and industrial application

This section aims to demonstrate the effectiveness of MGWO.

Firstly, in Section 4.1, a vise [6] with 11 components and a ball valve [7] 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 [36], SSO [19], STLBO [24], IHS [6] and ACO [22]. 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.

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. 5, and its corresponding interference and support matrices, material, and tool sets are offered in Fig. 6. MGWO is compared with its variations, including not using NSO (MGWO-1), not using GSO (MGWO-2), α , β , 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. 7.

Fig. 5 Assembly drawing of the vise

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

(a) Interference matrix

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

(b) Support matrix

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) Material set

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) Tool set

Fig. 6 Matrices, material, and tool sets of the vise**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

Fig. 7 Box plot of best fitness of vise under different strategies

It can be observed that MGWO, MGWO-1, MGWO-3 can obtain the optimal solution, but the mean of fitness values (3.8667) of MGWO is better (smaller) than that of other strategies. Fig. 7 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 falling into the local optimum. Fig. 8 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 effective, and the performance of MGWO is significantly better than its variants.

Fig. 8 Convergence curves 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

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 [7] with 17 components. The assembly view of the ball valve is shown in Fig. 9, and its corresponding interference and support matrices, material, and tool sets are offered in Fig. 10. 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. 11.

Fig. 9 Assembly drawing of ball valve

Fig. 11 Box plot of ball valve

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. Fig. 11 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%. Fig. 12 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.

Fig. 12 Convergence curves of ball valve

Table 5

Wilcoxon rank-sum test of the ball valve

Algorithms	Significance	<i>p</i> -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

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. 13.

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

Fig. 13 Box plot of the vise

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. Fig. 13 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. Fig. 14 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 significance between MGWO and other algorithms. It indicates that MGWO performs better than the other algorithms in solving the disassembly sequence of the vise.

Fig. 14 Convergence curves of the vise

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

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

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

Fig. 15 Box plot of ball valve

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. Fig. 15 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. Fig. 16 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 significant level of 0.05 to measure the significance between MGWO with other algorithms. It indicates that MGWO outperforms significantly the other algorithms in solving the disassembly sequence of the ball valve.

Fig. 16 Convergence curves of ball valve

Table 9

Wilcoxon rank-sum test of the ball valve

Algorithms	Significance	<i>p</i> -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

4.3 Industrial application

Used to verify the practicality of MGWO, azimuth thruster propeller [6], 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 its 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. 17. The simplification of the disassembling crank is shown in Fig. 18, and the matrices of the azimuth thruster propeller are presented in Fig. 19.

Fig. 17 Exploded view of azimuth thruster propeller

Fig. 18 Simplification of disassembling crank

072220000000000000	000000000000000000
507772222222222222	100000000000000000
05027220220220220	000000000000000000
05007773777777776	01100010010010010
05550110551554440	01110100100100100
00057010011444000	000000000000000000
00057703451444440	000001010000000000
00074470445444444	000000000000000000
00056000051044000	000000000000000000
00056000607044000	000000001010000000
00060000060004000	000000000000000000
00053000011054000	000000000000000000
00053000011307000	00000000000101000
00030000001030000	000000000000000000
00057000111044040	000000000000000000
00057110111154706	000000000000000101
00071111111115170	000000000000000000

(a) Interference matrix

(b) Support matrix

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) Material set

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) Tool set

Fig. 19 Matrices, material, and tool sets 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 statistic results of all algorithms are summarized in Table 10, and their corresponding box plots are presented in Fig. 20. The mean of fitness values (5.5669) of MGWO is better (smaller) 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 shows that MGWO is very suitable to solve the DSP problem.

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

Fig. 20 Box plot of azimuth thruster propeller

Fig. 21 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 accorded with the actual industrial disassembly sequence. Thus, the real-world application demonstrates the validity and practicality of the proposed method.

Fig. 21 Convergence curves of azimuth thruster propeller

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

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, Table 2 and Table 3 show that MGWO obtains the optimal solution with the highest probability in all strategies. For a vise case, 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 for the ball valve with the probability of 100%, but the probability of other strategies obtaining the optimal solution is less than 70%. Fig. 8 and Fig.12 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, Table 6 and Table 8 show that MGWO obtains the optimal solution with the highest probability in all algorithms. For 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 for the ball valve with the probability of 100%, but the probability

of other strategies obtaining the optimal solution is less than 20.00%. Fig. 14 and Fig. 16 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, a real-world industrial application proves that MGWO can obtain the disassembly sequence following practical production. Table 10 shows that MGWO receives the optimal solution with the highest probability in all algorithms. MGWO obtains the optimal solution with the probability of 76.67%, but the probability of other algorithms obtaining the optimal solution is less than 73.33%. Fig. 21 illustrates that MGWO converges faster than other algorithms.

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.

6 Conclusions and future researches

In this paper, a MGWO algorithm is proposed for solving the DSP problem. Three key operators, FSG, NSO, and GSO, are designed for the problem. Two engineering cases are employed to test the effectiveness of these operators. Then, they and one real-world application are employed 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. 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 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 design a new DSP model considering disassembly process constraints.

Authors' contributions 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. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture* 233(5): 1450-1471
2. Lambert AJ (2003) Disassembly sequencing: a survey. *International Journal of Production Research* 41(16): 3721-3759
3. Mishra A, Deb S (2019) Assembly sequence optimization using a flower pollination algorithm-based approach. *Journal of Intelligent Manufacturing* 30: 461-482
4. Jiménez P (2013) Survey on assembly sequencing: a combinatorial and geometrical perspective. *Journal of Intelligent Manufacturing* 24(2): 235-250
5. Tseng HE, Li RK (1999) A novel means of generating assembly sequences using the connector concept. *Journal of Intelligent Manufacturing* 10(5): 423-435
6. Li X, Qin K, Zeng B, Gao L, Su J (2016) Assembly sequence planning based on an improved harmony search algorithm. *The International Journal of Advanced Manufacturing Technology* 84(9-12): 2367-2380
7. Li X, Qin K, Zeng B, Gao L, Wang L (2017) A dynamic parameter controlled harmony search algorithm for assembly sequence planning. *The International Journal of Advanced Manufacturing Technology* 92(9-12): 3399-3411
8. Hsu HP (2016) A fuzzy knowledge-based disassembly process planning system based on fuzzy attributed and timed predicate/transition net. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 47(8): 1800-1813
9. Yu B, Wu E, Chen C, Yang Y, Yao B, Lin Q (2017) A general approach to optimize disassembly sequence planning based on disassembly network: A case study from automotive industry. *Advances in Production Engineering & Management* 12(4): 305-320
10. Kim HW, Lee DH (2017) An optimal algorithm for selective disassembly sequencing with sequence-dependent set-ups in parallel disassembly environment. *International Journal of Production Research* 55(24): 7317-7333
11. Tian G, Zhou M, Li P (2017) Disassembly sequence planning considering fuzzy component quality and varying operational cost. *IEEE Transactions on Automation Science and Engineering* 15: 748-760
12. Marconi M, Germani M, Mandolini M, Favi C (2019) Applying data mining technique to disassembly sequence planning: a method to assess effective disassembly time of industrial products. *International Journal of Production Research* 57(2): 599-623
13. Ilgin MA, Taşoğlu GT (2016) Simultaneous determination of disassembly sequence and disassembly-to-order decisions using simulation optimization. *Journal of Manufacturing Science and Engineering* 138(10): 101012

14. Meng K, Lou P, Peng X, Prybutok V (2016) An improved co-evolutionary algorithm for green manufacturing by integration of recovery option selection and disassembly planning for end-of-life products. *International Journal of Production Research* 54(18): 5567-5593
15. Guo X, Liu S, Zhou M, Tian G (2017) Dual-objective program and scatter search for the optimization of disassembly sequences subject to multiresource constraints. *IEEE Transactions on Automation Science and Engineering* 15(3): 1091-1103
16. Kongar E, Gupta SM (2006) Disassembly sequencing using genetic algorithm. *The International Journal of Advanced Manufacturing Technology* 30(5-6): 497-506
17. Tseng HE, Chang CC, Lee SC, Huang YM (2018) A Block-based genetic algorithm for disassembly sequence planning. *Expert Systems with Applications* 96: 492-505.
18. Feng Y, Gao Y, Tian G, Li Z, Hu H (2018) Flexible process planning and end-of-Life decision-making for product recovery optimization based on hybrid disassembly. *IEEE transactions on automation science and engineering* 16(1): 311-326
19. Yeh WC (2011) Optimization of the Disassembly Sequencing Problem on the Basis of Self-Adaptive Simplified Swarm Optimization. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans* 42(1): 250-261
20. 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. *International Journal of Advanced Manufacturing Technology* 57(9-12): 1183-1197
21. Kheder M, Triguí M, Aifaoui N (2017) A Disassembly Sequence Planning Approach Based on Particle Swarm Optimization. In *International Conference Design and Modeling of Mechanical Systems*, 1017-1025. Springer
22. Tseng HE, Chang CC, Lee SC, Huang YM (2019) Hybrid bidirectional ant colony optimization (hybrid BACO): An algorithm for disassembly sequence planning. *Engineering Applications of Artificial Intelligence* 83: 45-56
23. Feng Y, Zhou M, Tian G, Li Z, Zhang Z, Zhang Q, Tan J (2018) Target disassembly sequencing and scheme evaluation for CNC machine tools using improved multiobjective ant colony algorithm and fuzzy integral. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 49(12): 2438-2451
24. Xia K, Gao L, Li W, Chao KM (2014) Disassembly sequence planning using a simplified teaching-learning-based optimization algorithm. *Advanced Engineering Informatics* 28(4): 518-527
25. 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 Transactions on Automation Science and Engineering* 15(3): 1369-1385
26. Ren Y, Tian G, Zhao F, Yu D, Zhang C (2017) Selective cooperative disassembly planning based on multi-objective discrete artificial bee colony algorithm. *Engineering Applications of Artificial Intelligence* 64: 415-431
27. Tian G, Ren Y, Feng Y, Zhou M, Zhang H, Tan J (2018) Modeling and planning for dual-objective selective disassembly using AND/OR graph and discrete artificial bee colony. *IEEE Transactions on Industrial Informatics* 15(4): 2456-2468
28. Lu Q, Ren Y, Jin H, Meng L, Li L, Zhang C, Sutherland JW (2020) A hybrid metaheuristic algorithm for a profit-oriented and energy-efficient disassembly sequencing problem. *Robotics and Computer-Integrated Manufacturing* 61: 101828
29. Mirjalili S, Mirjalili SM, Lewis A (2014) Grey Wolf Optimizer. *Advances in Engineering Software*

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 Transactions on Sustainable 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. *International Journal of Electrical Power & Energy Systems* 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. *Engineering Applications of Artificial Intelligence* 57: 61-79
33. Shakarami MR, Davoudkhani IF (2016) Wide-area power system stabilizer design based on Grey Wolf Optimization algorithm considering the time delay. *Electric Power Systems Research* 133: 149-159
34. Zhang S, Zhou Y, Li Z, Pan W (2016) Grey wolf optimizer for unmanned combat aerial vehicle path planning. *Advances in Engineering Software* 99: 121-136
35. De M, Freire H, Pires ES (2016) Grey wolf optimization for PID controller design with prescribed robustness margins. *Soft Computing* 20(11): 4243-4255
36. Hui W, Dong X, Duan G (2008) A genetic algorithm for product disassembly sequence planning. *Neurocomputing* 71(13-15): 2720-2726

Figures

MGWO

Input: the interference matrix, support matrix, material set, tool set of the EOL product

Output: the optimal disassembly sequence

```
1  //Initialization;
2  For  $i = 1$  to population size do
3      Generate wolf  $X_i$  using FSG;
4      Calculate the fitness value of each wolf  $f(X_i)$ ;
5  End for
6  //Loop:
7  While (stop criterion is not satisfied & maximum iterations are not exceeded) do
8      // Neighborhood search:
9      Select two wolves with the two best fitness values as  $\alpha$  and  $\beta$ ;
10     Generate a new wolf as  $\delta$  using FSG and calculate its fitness value  $f(\delta)$ ;
11     Update  $\alpha$ ,  $\beta$ , and  $\delta$  wolves using NSO;
12     // Guided search:
13     For  $i = 1$  to population size do
14         Update wolf  $X_i$  using GSO;
15     End for
16 End while
```

Figure 1

The pseudo-code of the MGWO algorithm

FSG

Input: the interference matrix IM_{10+} of the EOL product

Output: a feasible disassembly sequence

```
1  Set the set  $C_A$  of components available for disassembly to  $\emptyset$ ;  
2  Set the map  $D_A$  for feasible disassembly directions of components in  $C_A$  to  $\emptyset$ ;  
3  For  $i = 1$  to length of disassembly component sequence  $X$  do  
4      For each component  $c$  in the EOL product do;  
5          If  $c \in X$  then  
6              Continue;  
7          End if  
8          If  $c$  can be disassembled by checking  $IM_{10+}$  then  
9              Record the feasible disassembly direction set  $D_c$  of  $c$ ;  
10             Push  $c$  back to  $C_A$ ;  
11             Put  $D_c$  into  $D_A$ ;  
12         End if  
13     End for  
14     Randomly choose a component  $c$  from  $C_A$ ;  
15     Push the  $c$  back to  $X$ ;  
16     Randomly choose a feasible disassembly direction  $k$  of  $c$  from  $D_A$ ;  
17     Push the  $k$  back to disassembly direction sequence  $D$ ;  
18     Set  $C_A$  to  $\emptyset$ ;  
19     Set  $D_A$  to  $\emptyset$ ;  
20     Set all the elements in row  $c$  or column  $c$  of  $IM_{10+}$  to 0;  
21 End for
```

Figure 2

The pseudo-code of FSG

Input: the interference matrix $IM_{(0+)}$ of the EOL product, the disassembly component sequence X , and its corresponding disassembly direction sequence D

Output: an improved disassembly sequence

```

1  Record the disassembly component sequence  $X$  and its corresponding disassembly direction
   sequence  $D$ ;
2  Randomly select a component  $c$  from  $X$  and record its disassembly direction  $k$ ;
3  Set the interference direction set  $I_c$  to  $\emptyset$ ; //  $I_c$  includes the directions in which component  $c$  is
   interfaced by components preceding  $c$ 
4  For  $i =$  the serial number of  $c$  to 1 do
5      Record the interference direction set  $\bar{D}_i$  of  $c$  interfered by component  $x_i$  through the matrix
        $IM_{(0+)}$ ;
6       $I_c = I_c \cup \bar{D}_i$ ;
7      If the size of  $I_c = 6$  then
8          Set the earliest insertion point  $a$  to  $i$ ;
9          Break;
10     End if
11 End for
12 For  $i =$  the serial number of  $c + 1$  to the length of  $X$  do
13     If  $c$  blocks the disassembly direction of  $x_i$  according to  $D$  then;
14         Set the latest insertion point  $b$  to  $i$ ;
15         Break;
16     End if
17 End for
18 Randomly select an insertion point  $p_{insert}$  between  $a$  and  $b$ ;
19 Update  $X$  by inserting  $c$  into  $p_{insert}$ , and delete  $c$  at the original position;
20 For  $i = 1$  to  $p_{insert}$  do
21     Set all the elements in row  $i$  or column  $i$  in  $IM_{(0+)}$  to 0;
22 End for
23 Record the feasible disassembly direction set  $D_c$  of  $c$  by checking  $IM_{(0+)}$ ;
24 If  $p_{insert} =$  the serial number of  $c$  then
25      $D_c = D_c \setminus \{d\}$ ; //  $d$  is the original disassembly direction of  $c$ 
26 End If
27 Randomly choose a feasible disassembly direction  $k$  from  $D_c$ ;
28 Update  $D$  by modifying the disassembly direction of  $c$ .
```

Figure 3

The pseudo-code of NSO

Original component sequence

3	5	4	2	1	6	8	7
+X	+Y	-X	-X	<u>+Z</u>	+Z	+X	+Y

Original direction sequence

New component sequence

3	5	1	4	2	6	8	7
+X	+Y	<u>+X</u>	-X	-X	+Z	+X	+Y

New direction sequence

Figure 4

Schematic diagram of NSO

GSO	
Input: the disassembly sequences of α , β , δ , and X wolves	
Output: a new disassembly sequence	
1	Set the new disassembly component sequence X_{new} to $\emptyset$;
2	Set the new disassembly direction sequence D_{new} to $\emptyset$;
3	For $i = 1$ to length of disassembly component sequence X do
4	Randomly generate probability $p \in [0, 1]$;
5	If $p < p_i$ then
6	Push the leftmost component of X back to X_{new} ;
7	Push the leftmost direction of D back to D_{new} ;
8	Else if $p < p_i + p_\alpha$ then
9	Push the leftmost component of α back to X_{new} ;
10	Push the leftmost direction of D_α back to D_{new} ;
11	Else if $p < p_i + p_\alpha + p_\beta$ then
12	Push the leftmost component of β back to X_{new} ;
13	Push the leftmost direction of D_β back to D_{new} ;
14	Else
15	Push the leftmost component of δ back to X_{new} ;
16	Push the leftmost direction of D_δ back to D_{new} ;
17	End if
18	Remove the chosen component from X_i , α , β , δ ;
19	Remove the direction of the chosen component from D_i , D_α , D_β , D_δ ;
20	End for
21	Update X with X_{new} ;
22	Update D with D_{new} .

Figure 5

The pseudo-code of GSO

Figure 6

Assembly drawing of the vise

Figure 7

Matrices, material, and tool sets of the vise

Figure 8

Box plot of best fitness of vise under different strategies

Figure 9

Convergence curves of vise under different strategies

Figure 10

Assembly drawing of ball valve

```

055550000000005577
70405000000001513
71005505001010100
75405454510101000
77550404015155512
20503000000010100
20457400000101000
20703200001000100
20477024010001000
20006000004207400
20002000000020700
20006000000045400
20002000000000500
74005000054540400
70005000005050000
60000000000000006
620000000000000070

```

```

01111000000001110
00000101000000001
00000010100000000
00000000000000000
00000000000000000
00110000000001100
00000000000000000
00000000000000000
00000000000000000
00000000000000000
00000000000000000
00000000000000000
00000000000000000
00000000000000000
00000000000000000
00000000010100000
00000000001010000
00000000000000000
00000000000000000

```

(a) Interference matrix

(b) Support matrix

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) Material set

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) Tool set

Figure 11

Matrices, material, and tool sets of ball valve

Figure 12

Box plot of ball valve

Figure 13

Convergence curves of ball valve

Figure 14

Box plot of the vise

Figure 15

Convergence curves of the vise

Figure 16

Box plot of ball valve

Figure 17

Convergence curves of ball valve

Figure 18

Exploded view of azimuth thruster propeller

Figure 19

Simplification of disassembling crank

[illegible]

Figure 20

Matrices, material, and tool sets of azimuth thruster propeller

Figure 21

Box plot of azimuth thruster propeller

Figure 22

Convergence curves of azimuth thruster propeller