RESEARCH ARTICLE

Multi-objective robust optimization over time for dynamic disassembly sequence planning

Xin Zhang[a], Yilin Fang[a], Quan Liu[a] and Danial Yazdani[b]

[a]School of Information Engineering, Wuhan University of Technology, Wuhan, China;
[b]Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, China

ARTICLE HISTORY
Compiled August 31, 2022

ABSTRACT
Disassembly sequence planning aims to optimize disassembly sequences of end-of-life (EOL) products in order to minimize the cost and environmental pollutant emission. Various unpredictable factors in the disassembly environment (e.g., EOL product status and capabilities of operators) lead to significant uncertainty making the optimal disassembly sequence change over time. Considering existing multiple objectives and dynamic environment, this problem is indeed dynamic multi-objective optimization. As deploying a new solution (i.e., disassembly sequence) is costly in this problem, it is undesirable to change the deployed solution after each environmental change. In this paper, we first propose a model for disassembly sequence planning problem in which several factors including the environmental changes, deployed solution switching cost, constraints, and multiple objectives are taken into account. To solve this problem where frequently changing the deployed solution must be avoided, we propose a new multi-objective robust optimization over time (ROOT) framework to find robust solutions based on two new robustness definitions: average performance and stability. The proposed framework benefits from a novel accurate online predictor and the knee-oriented dominance which is applied to select the naturally preferred tradeoff solution to meet the application requirements of ROOT. Computational experiments demonstrate the effectiveness of the proposed ROOT framework.

KEYWORDS
Robust optimization over time; dynamic multi-objective optimization ; disassembly sequence planning; knee-oriented dominance.

1. Introduction

Disassembly plays a key role in reusing and remanufacturing of end-of-life (EOL) products (1), since it enables the recovery of embodied value left within disposed of EOL products, and can be profitable and less harmful to the environment. A well-designed disassembly sequence helps in improving disassembly efficiency and reducing disassembly costs, which is called “disassembly sequence planning” (DSP) (2).

The type of disassembly planning methods can be summarized into two main categories: complete disassembly and selective disassembly for EOL product components, and sequential disassembly and parallel disassembly for the disassembly process. Com-

CONTACT Yilin Fang. Email: fangspirit@whut.edu.cn

plete disassembly aimes to disassemble and recycle all components contained in the EOL product (3; 4). Unlike complete disassembly, selective disassembly is carried out for specific components of the EOL product (e.g., high-margin components, high-hazard components, components in need of repair, etc.), or disassembly is stopped when a specific disassembly target is reached (5; 6). Sequential disassembly performs the tasks of disassembling components sequentially according to the structural characteristics and disassembly priorities of the EOL product (7; 8); parallel disassembly improves disassembly efficiency by performing disassembly tasks without mutual constraints concurrently (9; 10). In general, the disassembly mode depends on the EOL product and the actual disassembly requirements. A key element of DSP is a suitable mathematical representation that can accurately describe the disassembly process. Task precedence diagrams (TPDs) including AND/OR graphs (11), transformed AND/OR graphs (TAOG) (12), and Petri nets can accurately describe the product structure changes and precedence relationships of disassembly tasks during disassembly. However, there are many uncertainties in the actual disassembly process. EOL products that experience long-term use can suffer from significant uncertainty, which is the main source of uncertainty in the disassembly process. Many works have been carried out to deal with EOL products uncertainty. Gungor and Gupta (13) describe the uncertainty related difficulties in a DSP. They present a methodology for DSP for products with defective parts in product recovery. Tian *et al.* (14) present an energy management method for a product disassembly process integrating the maximum entropy principle and stochastic simulation. Gao *et al.* (15) introduce a disassembly planning and demanufacturing scheduling approach to an integrated flexible demanufacturing system. Gupta (16) and Brennan *et al.* (17) propose a DSP method which deals with defective parts. Tian *et al.* (18; 19; 20) form a chance constrained programming approach to determine the optimal product disassembly sequence. They analyze the main uncertainty factors for a disassembly process, e.g., disassembly time and tools (21; 22). Tian *et al.* (23) determine the optimal disassembly sequence by taking multiple uncertainty variables and mixed uncertainty features into account, where disassembled component quality is described as a fuzzy member function and disassembly cost is a randomly generated variable with uniform distribution. Laili *et al.* (24) proposed an interference probability matrix of product to indicate uncertainty in the interference and established a multi-threshold planning scheme to generate the optimal disassembly sequence plans.

The disassembly process with uncertainty of EOL products making the optimal disassembly sequence dynamically change over time, thus constituting a dynamic multi-objective optimization problem (DMOP). Most existing works mainly focuse on converting the uncertainty into certainty in various ways and then searching for the optimal disassembly sequence in the current environment (25), which is the idea of tracking the moving optimum (TMO). However, TMO results in frequently changing the deployed solution to response to the high uncertainty of the disassembly environment, which leads to additional costs of resources and is difficult to apply (26).

To address issues caused by frequently changing the deployed solution, robust optimization over time (ROOT) was proposed by Yu *et al.* (27). The main idea behind ROOT is to find solutions for deployment that remain acceptable after several upcoming environmental changes. Jin *et al.* (28) present a generic framework for finding robust solutions. This method consists of a population-based optimization component, a database, a fitness approximator, and a fitness predictor. Fu *et al.* (29; 30) propose two robustness definitions and metrics, namely survival time and average fitness. In the average fitness based robustness, the main goal is to find solutions for deployment

Table 1. The sets and parameters used in dDSP

Notion	Definition	Notion	Definition
A	Set of artificial nodes (components) that is indexed by a	$P(A_a)$	Set of immediate predecessor of ath artificial node
B	Set of normal nodes (tasks) that is indexed by b	$P(B_b)$	Set of immediate predecessor of bth normal node
I	Set of specific nodes (operations) that is indexed by i	L	Set of tools that is indexed by i
R	Set of robots that is indexed by r	$S(A_a)$	Set of immediate successor of ath artificial node
D	Set of directions that is indexed by d	$S(B_b)$	Set of immediate successor of bth normal node
$P\left(I_i^b\right)$	Set of immediate predecessor of ith specific node in bth normal node	$S\left(I_i^b\right)$	Set of immediate successor of ith specific node in bth normal node
T	Total number of EOL products to be disassembled	J	Set of subassemblies that is indexed by j
G	Set of variable ranges of subaseemblies state	W_i	Set of ranked positional weights of selected operations i
$\theta^{(t)}$	The independent disassembly environment constituted by the tth EOL product with an uncertain state	$c_j\left(\theta^{(t)}\right)$	The state weight of the jth subaseembly in tth environment
tw	The user-specified using time window of the solution	ef	The maximun number of iteration
SC	The switching cost of changing the new solution to replace the deployed solution	RS	The stability of the solution within the time window
DT	Disassembly tree from the corresponding ETAOG of a model	**OS**	Operation set vector
OR	Operation-robot assignment vector		

whose average fitness value over a predefined time window (i.e., a predefined number of environments) remain high. On the other hand, in the survival time based robustness, the goal is to maximize the average number of environments in which the deployed solution remain acceptable. Guo *et al.* (31) propose a two-layer multiobjective method to find robust solutions that can maximize both survival time and average fitness. Huang *et al.* (32; 33) take switching cost and survival time as optimization objectives, which aim to maximize the survival time and minimize the costs of switching deployed solutions, simultaneously. Euclidean distance between the deployed solution and a candidate solution used as the switching cost. Yazdani *et al.* (34) propose a multipopulation method which learns the problem space characteristics to estimate the future robustness of solutions instead of using predictors. Chen *et al.* (35; 36) proposed a robustness definitions of multi-objective for finding robust Pareto-optimal solutions over time (RPOOT), which is mainly oriented to DMOPs where fitness changes frequently but not drastically.

Most existing works on ROOT (27; 28; 29; 30; 31; 32; 33; 34) focus on solving artificial benchmark problems with continuous search space, in particular the moving peaks benchmark (MPB) (37). And, there are few researches and certain challenges on applying ROOT to practical problems, and the reasons may be as follows: Firstly, MPB is a single-objective optimization problem with continuous search space, while many practical problems are DMOPs. Secondly, in DMOPs often only a Pareto-optimal front (PoF) consisting of mutually non-dominated solutions can be obtained. How to select a solution from the PoF for application is a challenge to extend ROOT for DMOPs.

According to our investigations on DSP, ROOT based on average fitness is a suitable for practical disassembly since using a fixed time window makes industrial applications

Table 2. Decision variables of dDSP

Notion	Definition
ST_i	Starting time of ith operation
$X_{i,r}$	1,if ith operation is assigned to the rth robot;0,otherwise
Y_i	1,if ith operation is selected to perform;0,otherwise
Z_b	1,if bth task is selected to perform;0,otherwise

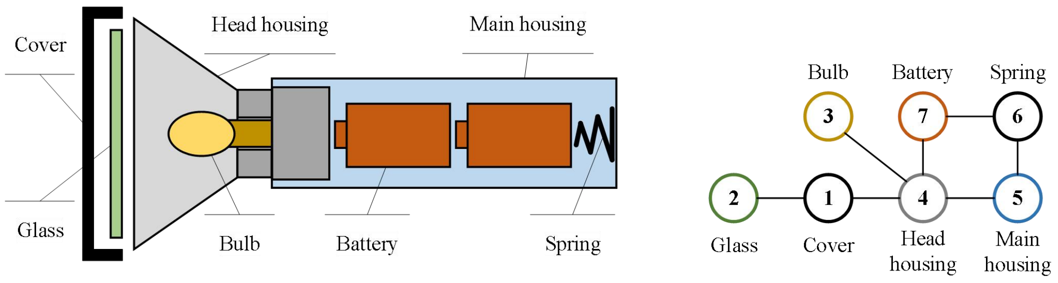


Figure 1. The structure of flashlight

more convenient and concise. In addition, ROOT based on average fitness can reduce the inconvenience of TMO while maintaining outstanding average fitness, which can reduce disassembly consumption and maximize disassembly profit.

The knee points (38) are solutions on the Pareto optimal front (PoF) and are the naturally preferred solutions that can maximize the benefits since they can significantly increase the fitness of other objectives by slightly sacrificing some objectives. The characteristics of knee points enable them to often obtain a large hypervolume (39), and have been applied to effectively solve DMOPs (40). Yu *et al.* (41) proposed a knee-oriented dominance relationship to accurately locate the knee solutions and eliminate the influence of interference solutions. Based on our investigations, finding knee points based on the knee-oriented dominance relationship for ROOT can save computational resources by minimizing the range of selected solutions while maximizing the performance of the solution.

In this paper, we propose a new model for dynamic disassembly sequence planning (dDSP) problem which considers the uncertainty of EOL products and operators. The performance of the disassembly sequence changes dynamically in the disassembly environment with uncertain EOL product status. To avoid frequent changes of deployed solution and switching cost, we propose a new multi-objective ROOT (mROOT) framework to find solutions for deployment that maintain outstanding average performance and stability within fixed time windows. mROOT consists of a new online predictor, new definitions of robustness, and knee-oriented-dominated sorting genetic algorithm (KSGA). We empirically evaluate our framework on different problem instances and the experimental results demonstrate the effectiveness of the proposed framework. In short, this article has the following major contributions:

(1) This is the first attempt to extend DSP to DMOP by presenting a novel dDSP model which uses a new TPD formulation of the dynamics of disassembly process

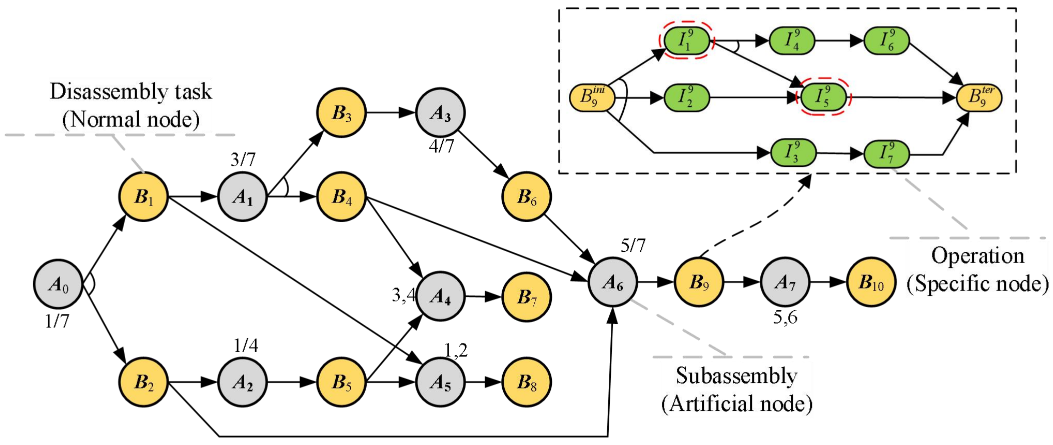


Figure 2. ETAOG of the flashlight

caused by the uncertainty of the product state.

(2) To avoid the extensive cost burden of TMO, we propose mROOT based on new robustness definitions for dDSP to find solutions that maintain outstanding average performance and stability within fixed time windows.

(3) KSGA based on the knee-oriented dominance relationship is proposed to accurately locate the global knee points for mROOT.

(4) A new online predictor is proposed to improve the prediction accuracy for mROOT.

The rest of this paper is organized as follows: Section 2 describes the dDSP model. Section 3 proposes a new mROOT framework. Section 4 describes the performance indicators, experimental setting, and methods to be compared. Section 5 reports the experimental results, analyses, and comparisons. Finally, Section 6 concludes this work and discusses some future research directions.

2. Dynamic disassembly sequence planning

Before we describe the proposed dDSP model, to improve the readability, the sets, parameters, and variables used in the model are listed in Tables 1 and 2, respectively.

2.1. *Description of dynamic disassembly environment*

In this study, we construct a new dDSP problem by considering the uncertainty of the subassemblies in EOL products. Due to the independence and randomness of each EOL product state, we investigate dDSP by decomposing the entire dynamic environment into a set of successive stationary environments as follows:

$$\left\{f(\vec{x},\vec{\theta}^{(k)})\right\}_{k=1}^{T}=\left\{f(\vec{x},\vec{\theta}^{(1)}),f(\vec{x},\vec{\theta}^{(2)}),\ldots,f(\vec{x},\vec{\theta}^{(T)})\right\} \tag{1}$$

where environment $\theta^{(t)}$ corresponds to the independent disassembly environment constituted by the tth EOL product with an uncertain state. Note that all EOL products go through a complete disassembly in turn resulting in the environmental changes

Table 3. Range of variations in subassembly state

J	G_E	G_M	G_P
1	$[0, 0.3)$	$[0.3, 0.6]$	$(0.6, 1]$
2	$[0, 0.5)$	$[0.5, 0.7]$	$(0.7, 1]$
3	$[0, 0.4)$	$[0.4, 0.6]$	$(0.6, 1]$
4	$[0, 0.3)$	$[0.3, 0.7]$	$(0.7, 1]$
5	$[0, 0.3)$	$[0.3, 0.6]$	$(0.6, 1]$
6	$[0, 0.4)$	$[0.4, 0.8]$	$(0.8, 1]$
7	$[0, 0.2)$	$[0.2, 0.6]$	$(0.6, 1]$

occurring in discrete time steps.

In the process of long-term use, the state of each subassembly in EOL products tends to be different due to product structure and user usage habits, which leads to significant uncertainty. Fig. 1 shows the structure of flashlight (42). In Fig. 1, the bulb of the flashlight may be fragile due to frequent use, while the internal spring is often in solid condition due to good protection. Tian *et al.* (23) proposed fuzzy variables to represent different quality levels of disassembled components. We introduce this concept in dDSP and define the state of the subassembly is classified into three categories: excellent, medium, and poor. As shown in Table 3, the range of variations G is set for each subassembly to indicate its possible state tendency according to its structural characteristics. E, M, and P indicate the excellent, medium, and poor states respectively. The random variable with uniform distribution $c_j\left(\theta^{(t)}\right) \in [0,1]$ indicates the state tendency of the jth subassembly of tth EOL product. For example, $c_j\left(\theta^{(t)}\right) \in G_E$ means the state of jth subassembly is excellent of tth EOL product. Naturally, the larger the range of variations G, the more likely the subassembly is to be in this state. The size of G is set according to the structure of EOL products and the characteristics of user habits.

To represent the task precedence relationship in dDSP, TAOG (12) is employed to model all possible alternatives for the disassembly process and precedence relationships among subassemblies and tasks. Moreover, we refine tasks in TAOG into multiple subassembly-specific operations and call it elaborated TAOG (ETAOG), which means that the state of the subassembly will directly affect the efficiency of the operation. The ETAOG of the flashlight is presented in Fig. 2. In Fig. 2, subassemblies, disassembly tasks, and operations are modelled by artificial nodes labelled A_a, normal nodes labelled B_b, and specific nodes labeled I_i^b, respectively. The ETAOG of the flashlight includes eight artificial nodes and ten normal nodes, and each normal node contains multiple operation schemes. Each subassembly is represented by an artificial node labelled A_a. Set A contain the indices of all possible subassemblies that can be generated by the tasks from B. The set of parts in a subassembly is denoted by a number sequence below the corresponding artificial node. For simplicity, subassemblies with only one part are not shown. For instance, artificial node A_2 represents the subassembly '1/4' which consists of components 1 through 4 (the cover, glass, bulb and head housing in Fig. 1).

Arcs in the ETAOG are of two types: AND-type and OR-type. While the AND-type arc imposes a mandatory precedence relation case, the OR-type arc allows alternative selection of the following routes. An indicator of OR-type arc is shown by a small curve. An artificial node can be preceded or succeeded by one (AND-type) or more

Table 4. Example of operations information of robot 1

I	B	J	L	D	$dt_{i,r}\,(E/M/P)$	$c_{i,r}\,(E/M/P)$
1	9	5	T1	+X	2.22/3.32/3.88	3.13/4.13/3.64
2	9	6	T4	+X	-	-
3	9	7	T3	-X	6.44/8.33/9.76	2.78/3.96/3.33
4	9	6	T1	+Y	1.33/3.34/2.22	6.68/4.48/6.77
5	9	7	T3	-X	3.12/2.98/5.45	3.88/4.54/2.48
6	9	5	T2	+Y	-	-
7	9	6	T3	+Y	1.82/2.23/3.42	2.82/4.42/5.42

(OR-type) normal nodes, but only one predecessor or successor will be executed. As shown in Fig. 4, a disassembly tree (DT) is generated by selecting nodes from the ETAOG according to the AND/OR-type relationship to form a disassembly sequence. In ETAOG, not only sequential tasks but also parallel tasks, which take part in a subassembly into two, each of which has at least two parts, can be considered. Each task refine tasks in TAOG into multiple subassembly-specific operations to complete. For example, normal node B_9 represents the task of removing component 7 (battery) from the subassembly A_6 with components 5, 6, and 7 (main housing, spring and battery), and task 9 is subdivided into many step of operations to complete. As shown in Table 4, each operation contains an independent direction and tool processes only one subassembly. For instance, I_1^9 is the operation to hold the component 5 (Main housing) with the T1 tool in the "+X" direction, while the I_5^9 operation is to take out the component 7 (battery) with the T3 tool along the "-X" direction. Task 9 is completed by executing I_1^9 and I_5^9 sequentially.

Each step of the operation needs to be assigned to a robot for execution, and due to the specificity of different robot configurations, different robots have different tendencies and efficiencies when handling the same operation. For example, for the same operation of pulling subassembly 5 in the +X direction with tool T1, robot 1 is good at handling subassembly 5 in poor condition, while robot 2 is better at handling subassembly 5 in medium and excellent condition. Each robot has different tools that limit the operations it can perform. Fang *et al.* (43) set different disassembly time and cost for the disassembly task according to the inclination of different robots. Tian *et al.* (44) and Hu *et al.* (45) define the directions and tools included in the disassembly operation, respectively. On this basis, we add disturbance factors to represent the impact of product state. As shown in Table 4, the disassembly time and cost for each robot to process operations are set to fixed values that vary with the state of the subassembly according to the operating characteristics and robot configurations. Robot 1 is not suitable for performing operations (e.g., I_2^9 and I_6^9) with certain tools (e.g., T2 and T4) due to constructional characteristics that may be specific to the output power or flexibility of the robot. The uncertainty of the EOL product state makes the same disassembly sequence behave differently when disassembling products in different states, thus constituting a dynamic disassembly environment.

Based on the above description of the dynamic disassembly environment, the following assumptions are considered for dDSP:

(1) EOL products are the same type and different products have different states of subassemblies which are known at the time of disassembly.
(2) EOL product requires multiple tasks for disassembly, each task contains multiple

operations, and each operation faces one subassembly and is affected by its state.

(3) Robots in dDSP are different and have different tools and capabilities for subassembly in different states.
(4) Disassembly operation has requirements for specific tools, so operation is not applicable to all robots.
(5) Each operation can be assigned to exactly one robot at the same time, and robot has an independent working area so physical collisions would not happen.
(6) Disassembly time and cost for a disassembly operation depends on the capabilities of assigned robot and the state of the subassembly, it is deterministic, constant, and discrete.
(7) Disassembly operations cannot be split and the precedence relationship in ETAOG should be met while assigning the operations.
(8) Each disassembly takes apart the product or subassembly into two new subassemblies and each EOL product undergoes complete disassembly.
(9) Setup times between disassembly tasks are ignored and no work-in-process inventory buffer is considered.

2.2. *Mathematical model of dDSP*

2.2.1. Objective functions

Minimizing disassembly time and cost are the most important disassembly objectives, which reflect the efficiency and reliability of the disassembly sequence. In this study, we use the minimizations of disassembly time and cost as the optimization goals. Note that the fitness values are affected by the environment and capability of the assigned robot to operate. Therefore, the disassembly time and cost that changes dynamically under the influence of the environment and capability of the assigned robot to operate are formulated as follows:

$$\text{Minimize} \quad f_1\left(\vec{x}, \theta^{(t)}\right) = \max_{r\in R} \sum_{i\in I} X_{i,r} \times dt_{i,r}\left(\theta^{(t)}\right), \tag{2}$$

$$\text{Minimize} \quad f_2\left(\vec{x}, \theta^{(t)}\right) = \sum_{i\in I}\sum_{r\in R} X_{i,r} \times c_{i,r}\left(\theta^{(t)}\right). \tag{3}$$

where $dt_{ir}\left(\theta^{(t)}\right)$ and $c_{ir}\left(\theta^{(t)}\right)$ are the working time and energy consumption, when rth robot processes ith operation in $\theta^{(t)}$ environment, respectively. As shown in Table 4, where the state of the EOL product component is the direct factor affecting dt_{ir} and dt_{ir} in $\theta^{(t)}$ environment. Since the disassembly is done in parallel, the total disassembly time is determined by the robot who has spent the longest time to carry out its operations. Besides, the disassembly cost is considered as the sum of the consumptions incurred by all robots performing all assigned operations. Disassembly time and disassembly cost are two conflicting objectives. Minimizing disassembly time requires the robot to complete operations more quickly, which inevitably requires the robot to sacrifice some costs (e.g., more energy consumption and tool consumption)

Table 5. Constraints of dDSP

Index	Formula
C1	$0 \leq \sum_{r \in R} X_{i,r} \leq 1, \forall i \in I$
C2	$\sum_{B_b \in S(A_a)} Z_b = 1, a = 0, a \in A$
C3	$\sum_{B_b \in S(A_a)} Z_b = \sum_{B_b \in P(A_n)} Z_b, \forall a \neq 0, a \in A$
C4	$\sum_{k \in S(I_i)} Y_k = \sum_{j \in P(I_i)} Y_j = 1, \forall i \in I$
C5	$\sum_{p \in P(I_i)} Y_p \geq Y_i \geq \sum_{s \in S(I_i)} Y_s \geq 0, \forall i \in I$
C6	$\sum_{p \in P(A_a)} Z_p \geq \sum_{s \in S(A_a)} Z_s \geq 0, \forall a \in A$
C7	$ST_s > ST_i > ST_p \geq 0, \forall i \in I, p \in P(I_i), s \in S(I_i)$

to increase efficiency, resulting in more disassembly cost. Therefore, our goal is to find a balance between two conflicting disassembly objectives in a dynamic disassembly environment. Note that the objective functions (2) and (3) need to be transformed by (5) and (6) when mROOT is applied to find robust disassembly sequence in time domain for dDSP.

2.2.2. *Switching cost*

One main purpose of using ROOT is to reduce the cost burden of frequently changing the deployed solution. Therefore, it is necessary to consider the switching cost of the deployed solution. Huang *et al.* (33) measured switching cost by easily calculating the Euclidean distance between the deployed solution and a candidate solution. However, dDSP is a combinatorial optimization problem which make it difficult to measure the switching cost. Considering the characteristics of dDSP, the following formula is proposed for calculating the switching cost in our model:

$$SC = \sum_{r \in R} dif\left(OR\left(\overrightarrow{x}^*, r\right), OR\left(\overrightarrow{x}, r\right)\right). \tag{4}$$

where SC is the switching cost, OR is the operation-robot assignment vector representing the operations assigned to each robot by the solution, $\overrightarrow{x}^*$ is the current solution, and $\overrightarrow{x}$ is the candidate solution. SC in dDSP reflects the consumption of switching solutions by measuring the number of different operations performed by each robot in the new and current solution. When the more identical operations are assigned to each robot by two disassembly sequences, the more parts of the original setup are preserved when switching solutions, and the less realistic resources are consumed to redeploy solutions.

2.2.3. *Constraints*

A feasible disassembly sequence in dDSP must satisfy the constraints formulate in Table 5. Constraint C1 guarantees that each operation is assigned to only one robot and can be performed at most once. C3 ensures that the generation of DT does not violate the precedence constraint of tasks. Constraints C2 and C4 ensure that each DT can only generate one disassembly sequence to completely disassemble the product. Constraint C5 and C6 ensure that the assignment of all selected tasks and operations does not violate the precedence constraints of these tasks and operations. Constraint C7 guarantees that the start time of operations is subject to the precedence constraints of operations.

3. Proposed mROOT framework

In this section, we describe the proposed mROOT for tracking dDSP. Unlike the majority of the existing works in the ROOT literature that focused on solving artificial benchmark problems with continuous search space, the proposed mROOT is designed to tackle a real-world combinatorial optimization problem. mROOT consists of several components including a new predictor, definitions of robustness for multi-objective, KSGA based on the knee-oriented dominance relationship (41), genetic operator, and solution encoding/decoding that will be described in the following.

Algorithm 1: mROOT

```
while ($t < T$) do
    if environmental has changed then
        $t = t + 1$;
        if The usage time of the current solution is within the time window then
            Continue use the current solution in new environment;
        else
            $P_t(0)$ ← Initialize population by the encoding and decoding rule in Section 3.5;
            $g = 0$;
            while ($g < ef$) do
                Calculate the performance of $P_t(g)$ in the historical environment and update datebase;
                $P_t(g+1)$=KSGA($P_t(g)$);
                $g = g + 1$;
            end
            Train the predictor using the updated datebase;
            Select the next solution for deployment with the smallest $RS$ from PoF;
        end
    end
end
```

3.1. *Overall framework*

The high-level procedure of mROOT is presented in Algorithm 1. The proposed approach is executed after each environmental change until all EOL products have been disassembled, where t is the current disassembled products and T is the total number of EOL products. Before a new EOL product is disassembled, a quality check is performed on each of its components. If the component status of the current EOL product is significantly different from the previous EOL product, the disassembly environment is considered to have changed. Similar to the ROOT based on average fitness proposed by Fu *et al.* (29), if the use time of the current deployed solution is within the time window specified by the user, there is no need to find another solution for deployment.

Once current time window ends, the algorithm searches for a new solution for deployment for the next time window. In other words, a disassembly solution needs to be re-optimized and switched after a period of time (the time window specified by the user) depending on the actual disassembly environment. The procedure of finding a new solution starts with generating a population $P_t(0)$ for initialization according to the rules of solution encoding and decoding proposed in Section 3.5 (line 7). Then, the population is iteratively evolved through the $KSGA$ proposed in Section 3.3, where ef is the maximum number of iteration (lines 9-13). To estimate the future performance of a candidate solution, online predictors are used which are described in Section 3.4. After obtaining a PoF that outperforms in the time window, a solution is selected from PoF for deployment using the proposed decision making process in Section 3.2 (line 15).

3.2. *Definition of multi-objective robustness*

Note that mROOT is an algorithmic framework with generality. Therefore, the objective functions (2) and (3) in Section 3.2 need to be transformed with by the proposed robustness definition when mROOT is applied to dDSP (lines 3-5 in Algorithm 2).

3.2.1. Multi-objective average fitness

The multi-objective average fitness represents the average performance of the deployed solution in a user-specified time window. For a solution $\vec{x}$ at the tth environment, the corresponding robustness is defined as:

$$mF^A\left(\vec{x}_i, \theta^{(t)}\right) = \left(f_1^{ave}\left(\vec{x}_i, \theta^{(t)}\right), \cdots, f_m^{ave}\left(\vec{x}_i, \theta^{(t)}\right)\right), \tag{5}$$

$$f_m^{\text{ave}}\left(\vec{x}_i, \theta^{(t)}\right) = \frac{1}{tw}\left(f_m\left(\vec{x}_i, \theta^{(t)}\right) + \sum_{l=1}^{tw-1} \hat{f}_m\left(\vec{x}_i, \theta^{(t+l)}\right)\right). \tag{6}$$

where $\theta^{(t)}$ reprensents the environmental state, which corresponds to the tth EOL product in dDSP, tw is the user-specified time window, $\hat{f}_m\left(\vec{x}_i, \theta^{(t+l)}\right)$ is the approximated fitness of the mth objective in the t+lth environment estimated by a predictor, and the multi-objective average fitness $mF^A\left(\vec{x}_i, \theta^{(t)}\right)$ measures the average performance of solution $\vec{x}_i$ in each objective during the time window.

3.2.2. Multi-objective stability

The multi-objective stability represents the fluctuation of the fitness of the deployed solution within the time window, which is defined as follows:

$$RS\left(\vec{x}_i, \theta^{(t)}\right) = \frac{1}{tw}\sum_{l=0}^{tw}\left(\left\|mF^A\left(\vec{x}_i, \theta^{(t)}\right) - \hat{F}\left(\vec{x}_i, \theta^{(t+l)}\right)\right\|\right). \tag{7}$$

where RS responds to the stability of the solution's fitness within the time window. Smaller values of RS indicate smaller fitness fluctuation of the solution over a time window and higher stability.

Note that both average fitness and stability depend on the estimated fitness in the future environment provided by a predictor. Consequently, the performance of robustness is highly dependent on the accuracy of the predictor.

Algorithm 2: *Procedure of KSGA*

```
   Input: $P_t(g)$, output population size: $n$
   Output: $P_t(g+1)$
1  $Q_t(g) \leftarrow$ crossover and mutation $P_t(g)$;
2  $R_t(g) \leftarrow P_t(g) \cup Q_t(g)$;
3  for all individuals $\overrightarrow{x_i}$ in $R_t(g)$ do
4      Calculate $mF^A\left(\vec{x}_i, \theta^{(t)}\right)$, $RS\left(\vec{x}_i, \theta^{(t)}\right)$ by (5), (7);
5  end
6  KD Sorting($R_t(g)$)={$L_1, L_2, ...$};
7  for $L_t \in \{L_1, L_2, ...\}$ do
8      if $|P_t(g)| + |L_t| \leq n$ then
9          $P_t(g+1) = P_t(g) \cup L_t$;
10     else
11         $S_t \leftarrow$ Select $(n - |P_t(g)|)$ individuals from $L_t$ in ascending order of their
            stability $RS$;
12         $P_t(g+1) = P_t(g) \cup S_t$;
13     end
14 end
```

3.3. *Procedure of KSGA*

According to nondominated sorting-based genetic algorithm (NSGA-II) (46), KSGA replaces the original dominance relationship and the crowding distance with the knee-oriented dominance relationship and stability RS, respectively. The knee-oriented dominance relationship is proposed in (41) to accurately locate the knee solutions and eliminate the influence of interference solution. Since mROOT needs to select the solution with the most outstanding average performance in the time window to apply. Different from Yu *et al.* local search of multi-knee region in order to preserve diversity, we directly apply the knee-oriented dominance relationship to find the global knee points to save computational resources and narrow the selection of solutions. Given two solutions A and B, and the dominance indicator μ of knee-oriented dominance relationship as follows:

$$\mu(A,B)=\left\langle \overrightarrow{N_{id}A}, \overrightarrow{AB} \right\rangle - \tau \cdot \left(\max_{i=1,\dots,m} \{\varphi_i(A)\} + \min_{i=1,\dots,m} \{\varphi_i(A)\} \right), \tag{8}$$

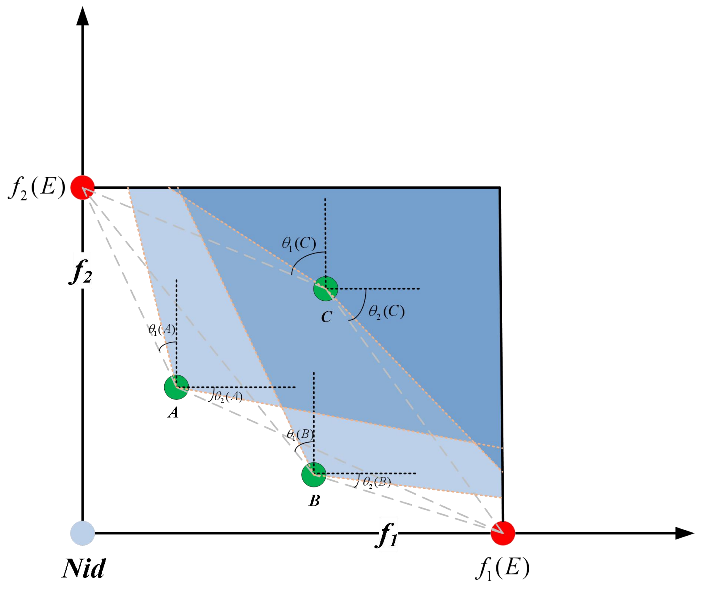


Figure 3. Example of knee-oriented regions

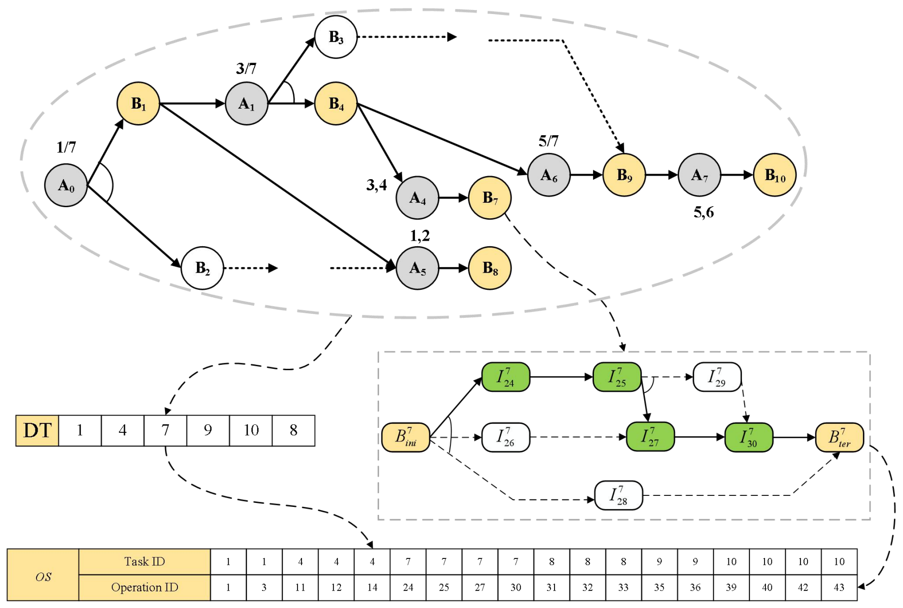
where

$$\varphi_i(A) = \arctan\left(\frac{\sqrt{\sum_{j=1, j\neq i}^{m}\left(f_j^{ave}(A) - f_j^{ave}\left(N_{id}\right)\right)^2}}{\left|f_j^{ave}(A) - \max f_i(E) - \varepsilon\right|}\right). \tag{9}$$

where $\varphi_i(A)$ is an acute angle determined by the ith objective value of A, $\max f_i(E)$ is the extreme value of the ith objective within the population, N_{id} is the ideal point, ε is a small positive constant to ensure the denominator is not equal to zero, $\tau \in [0,1]^2$ is a parameter controlling the size of the knee region to be achieved, and $f_j^{ave}(A)$ is the average fitness of jth objective defined in (6).

Note that $\mu(A,B) < 0$ means solution A knee-oriented dominates B. If both $\mu(A,B)$ and $\mu(B,A)$ are positive, it means they are non-knee-oriented dominated by each other. The shaded area in Fig. 3 represents the respective dominated region of A, B, and C. As can be seen in this figure, solutions A and B are non-knee-oriented dominated from each other, while C is knee-oriented dominated by both A and B. Moreover, the closer a solution from the ideal point is, the larger its dominated region becomes, the more likely it is a knee point, and the less likely such a solution will be dominated by other solutions.

The procedure of *KSGA* is shown in Algorithm 2 which has four steps. First, a new offspring population $Q_t(g)$ is generated by crossover and mutation of $P_t(g)$ based on the rule that will be described in Section 3.6, and $R_t(g)$ population was obtained by merging $P_t(g)$ and $Q_t(g)$ (lines 1-2). Second, the individuals in $R_t(g)$ population are sorted according to the knee-oriented dominance relationship (KD Sorting) and $R_t(g)$ is divided into a number of fronts, a front number is assigned to each individual (line 6). Third, the solutions are selected front by front according to their front number in ascending order (lines 7-9). The selection continues until it starts to select solutions from the critical front denoted by L_t. The critical front is defined by Yu *et al.*(41), as



DT	1	4	7	9	10	8

OS	Task ID	1	1	4	4	4	7	7	7	7	8	8	8	9	9	10	10	10	10
	Operation ID	1	3	11	12	14	24	25	27	30	31	32	33	35	36	39	40	42	43

Figure 4. Example of the DT and OS generation procedure

the last front from which only part of its solutions will be selected because it contains more individuals than the remaining space of the population. Finally, individuals were selected from L_t in ascending order of stability RS to complete the new population (lines 11-12).

KSGA expands the dominant region of the solution to accurately locate global knee points and retains individuals with stable performance within a fixed window, providing ideal naturally preferred tradeoff solutions to meet the application requirements of mROOT.

3.4. *Online predictor*

The existing prediction-based works (29; 30; 31; 32; 33; 35; 36) use Autoregressive (AR) or moving average (MA) model as predictor, which is suitable for problems with continuous search space. However, dDSP is a combinatorial optimization problem whose discrete search space will lead to sharp changes in fitness, which makes prediction through time series forecasting methods very error-prone.

Therefore, we propose an online predictive model based on XGBoost (47) for learning problem space characteristics as the environment changes, in order to find a certain rule by learning the massive performance of the different solutions in the historical environments. Since the information of historical environments is known, we can obtain historical dataset by calculating the performance of all solutions in the population in the historical environment and storing them in the database. Update the database and prediction model after each robust solution search (lines 10, 14 in Algorithm 1). Robust solutions found by ROOT methods can generally be used over a longer period of time, so the update of database and prediction model will not be frequent.

Note that this method may not be able to accurately predict the fitness of the solution, but it can approximately approach a trend of change. Indeed, there are many other machine learning models that can learn and represent this kind of trend more accurately. However, the main reason behind choosing XGBoost is that its training

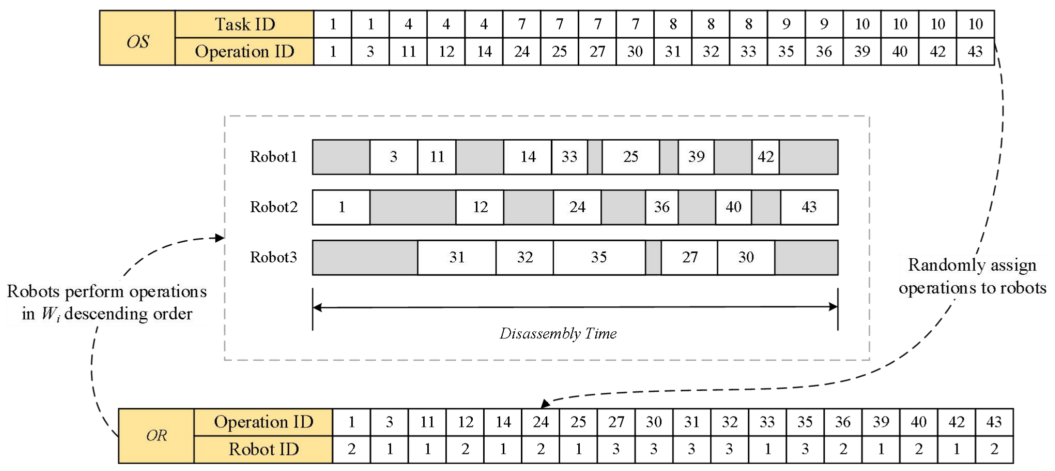

OS
Task ID 1 1 4 4 4 7 7 7 7 8 8 8 9 9 10 10 10 10
Operation ID 1 3 11 12 14 24 25 27 30 31 32 33 35 36 39 40 42 43
Robot1 3 11 14 33 25 39 42
Robot2 1 12 24 36 40 43
Robot3 31 32 35 27 30
Disassembly Time
Robots perform operations in W_i descending order
Randomly assign operations to robots
OR
Operation ID 1 3 11 12 14 24 25 27 30 31 32 33 35 36 39 40 42 43
Robot ID 2 1 1 2 1 2 1 3 3 3 3 1 3 2 1 2 1 2

Figure 5. Example of solution encoding and decoding for dDSP

process is fast and the model is computationally light, which make it suitable to be applied in online training and prediction in our method.

3.5. *Solution encoding and decoding*

In this study, two integer vectors are used for the encoding: operation set vector **OS** and operation-robot assignment vector **OR**. **OS** is generated on the basis of a feasible DT from the corresponding ETAOG of a model. Fig. 4 illustrates the DT and **OS** generation procedure on a sample ETAOG. The DT generation procedure starts with randomly selecting a normal node (task) from the successors of the first artificial node (subassembly) in ETAOG, which is B_1 in Fig. 4. Then, the immediate successor artificial nodes of the selected normal node are determined, which is A_1 and A_5. For each artificial node, a new normal node is selected randomly from its successors, which are B_4 and B_8. This procedure is repeated until no successor normal nodes are selected and the output represents a DT. As can be seen in Fig. 4, tasks 1, 4, 7, 8, 9, and 10 are randomly chosen to compose a DT. After determining the DT, the operation in each normal node is randomly selected according to the predecessor and successor relationship of the operation. For example, in Fig. 4, operations 24, 25, 27, and 30 are randomly chosen to compose the operation set for task 7, as shown in Fig. 4. Similarly, other tasks randomly select the operations it contains and output represents an operation set

After obtaining the **OS**, each operation is randomly assigned to a robot that satisfies the operating conditions. In this study, it is necessary that the robot has the tools required for the operation. In Fig. 5, all operations are randomly assigned to robots with operating conditions to form an **OR**. Each robot performs operations in ranked positional weights W_i (48) ascending order, which is the sum of the processing times of the operation i and of all its successors. In other words, if more operations need to wait for operation i to complete before they can be executed in order to satisfy constraints, then the priority of operation i is increased. For example, Fig. 5 and Fig. 4 show that all operations must wait for operation 1 to complete. In such a circumstance, operation 1 is prioritized and executed first among the operations assigned to robot 2. In addition, operation 12 is the successor of operation 11 so it must wait for the completion of operation 11 before it can be executed. Operation 31, on the other hand,

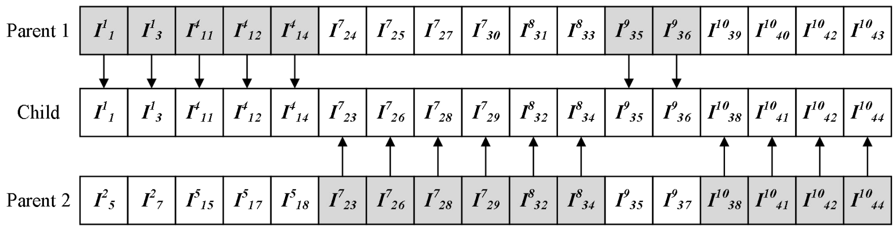


Figure 6. Example of crossover operation

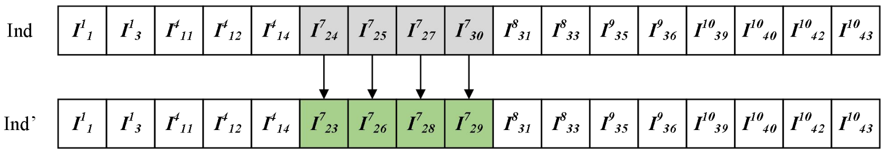


Figure 7. Example of mutation operation

is the successor of operation 3 but has no predecessor or successor relationship with operation 12. Consequently, operation 31 can be executed in parallel with operation 12 after waiting for the completion of operation 3.

3.6. *Genetic operators*

The genetic operators consist of crossover and mutation on **OS**. After completing the crossover and mutation operation, **OR** will be redistributed under the condition that constraints are satisfied. In dDSP, each task is composed of multiple operation steps. In order to meet the priority relationship between tasks, crossover and mutation operations mainly swap or mutate the operation sequences in that task as a unit.

Since the disassembly tasks contained in the two parents may be different, the method of crossover is position-based crossover combined with uniform crossover (49). Based on the method of position-based crossover, the crossover operator selects the task sequence of parent 1 as the genetic template, and tasks unique to parent 1 are directly inherited by the offspring. For tasks identical in parents 1 and 2 (each containing different sequences of operations)are inherited randomly to the offspring based on the uniform crossover approach. As shown in Fig. 6, tasks 1 and 4 are task sequences unique to parent 1. Therefore, the operations I_1^1, I_3^1, I_{11}^4,I_{12}^4, and I_{14}^4 contained in tasks 1 and 4 are copied to the offspring, while for tasks 7, 8, 9, and 10 common to parents 1 and 2 are randomly selected for copying. Similarly, the mutation operation is a randomly selected task from an individual that causes a sequence of operations to be reselected to achieve mutation. As shown in Fig. 7, task 7 is randomly selected and regenerated with a new sequence of operations.

4. Experimental design

4.1. *Performance indicators*

To assess the performance of algorithms, we introduce five performance indicators: $\overline{f_1}$, $\overline{f_2}$, $\overline{RHV}$, $\overline{RS}$, $\overline{SC}$ and $\overline{CT}$ which denote the average disassembly time, average disassembly cost, average fitness, average stability, average switching cost and average cost time of deployed solutions, respectively. Note that $\overline{RHV}$ is based on hypervolume

Table 6. Information of test problem

Instance	Prodcut	Components	Tasks	Operations
P1	Handlight	7	11	50
P2	Ball-point pen	10	20	77
P3	Automatic pencil	11	37	132

Table 7. The parameter levels

Level	Population	Iteration	Mutation
1	100	30	0.5
2	80	20	0.33
3	50	10	0.25

value (50) to represent the fitness of multi-objective. Suppose the total number of environments is T and the number of switching solutions is N, the definition of three indicators are as follows:

$$\overline{RHV} = \frac{1}{T} \sum_{i=0}^{N-1} \sum_{l=1}^{tw} \text{hypervolume} \left(F \left(\vec{x_i}, \theta^{(i*tw+l)} \right), W \right), \tag{10}$$

$$\overline{RS} = \frac{1}{N} \sum_{i=1}^{N} RS(\vec{x_i}), \tag{11}$$

$$\overline{SC} = \frac{1}{T} \sum_{i=1}^{N} SC(\vec{x_i}). \tag{12}$$

Where $F\left(\vec{x_i}, \theta^{(i*tw+l)}\right)$ are the fitness of ith solution in $(i * tw + l)$th environment. $\overline{RHV}$ is the average of the hypervolume enclosed between the robust solutions and a given reference point W in all environments, which represents the average fitness of the multi-objective robust solutions. $\overline{RS}$ represents the average variation of the robust solution within the time window, and a smaller $\overline{RS}$ means that the multi-objective robust solutions are less volatiled and more stable within the time window. The definition of $\overline{SC}$ is similar to Huang *et al.* (33), representing the average switching cost of the deployed solution.

4.2. *Test problem instances*

To evaluate the performance of the algorithm under different scales of decision space, we use three problem instances with three types of products (42; 51; 52) with different disassembly difficultices, P1, P2, P3. The detailed information of the three problem instances are shown in table 6. We consider three types of robots in our experiments. Each robot has different types of tools and unique disassembly time and cost for each

Table 8. Orthogonal table based on taguchi method

NO	Population	Iteration	Mutation
1	1	1	1
2	1	2	3
3	1	3	2
4	2	1	3
5	2	2	2
6	2	3	1
7	3	1	2
8	3	2	1
9	3	3	3

operation. According to the random number method introduced in Section 2.1, we generate 100 EOL products with different states of subassemblies. Each each EOL product corresponds to an independent disassembly environment. The uncertain state of EOL products and robots with different capabilities for each operation create an uncertain and dynamic disassembly environment for dDSP.

4.3. *Tuning and setting the parameter of algorithms*

Many standard parameters, such as population size, iteration number, and mutation probability, are involved in the the evolutionary search part of the algorithm framework of mROOT. Therefore, it is necessary to tune and set the standard parameters to achieve the best running state of algorithms. Since the Taguchi method is one of the most efficient methods to find the optimal solution with a few experiments (53), we plan to adopt this method to tune the three mutual parameters of the algorithm. We use three representative multi-objective optimization algorithms (MOEAs) to compare: NSGA-II (46), multi-objective evolutionary algorithm based on decomposition (MOEA/D) (54), and Indicator-Based evolutionary algorithm (IBEA) (55). Parameter Tuning experiments are performed using different MOEAs for evolutionary search in the same disassembly environment for each problem instance and comparing inverted generational distance (IGD) (56) results. IGD can evaluate the convergence performance and distribution performance of the algorithm.

The suitable values for the population size, iteration number, and mutation probability are given in Table 7, and the orthogonal table based on the Taguchi method is shown in Table 8. The numbers 1-9 in the first column of Table 8 represent the different experimental settings. Results of IGD obtained by multiple experiments are recorded in Table 9. The specific calculation method of signal-to-noise ratio S/N in the Taguchi method can be found in Lotfi et al. (57).

As shown in Table 9, the optimal values of different algorithms are distinct. However, with the parameter settings of Experiment 2, the performance of each algorithm is well balanced and excellent. Therefore, we refer to the parameter settings of Experiment 2 set the population size to 100, the mutation probability to 0.25, and the iterations number to 20.

Table 9. IGD results of tuning the parameter of algorithms

Instance	NO	NSGA-II		MOEA/D		IBEA	
		IGD	S/N	IGD	S/N	IGD	S/N
P1	1	0.0039	3.74	0.71	-10.15	0.26	-1.45
	2	**0.0038**	**4.03**	0.019	1.97	0.019	2.14
	3	0.050	0.42	0.14	-1.14	0.025	1.88
	4	0.029	1.13	0.28	-2.11	0.38	-3.88
	5	0.013	2.23	0.38	-3.31	0.26	-1.86
	6	0.083	-0.34	0.26	-1.97	0.13	-1.23
	7	0.0039	3.68	0.38	-3.25	0.39	-3.34
	8	0.025	1.89	0.025	1.79	**0.017**	**1.12**
	9	0.064	0.23	0.32	-3.84	0.31	-3.28
P2	1	1.05	-34.12	**1.52**	**-41.65**	1.52	-41.03
	2	**0.96**	**-21.23**	1.73	-45.34	**1.03**	**-22.03**
	3	1.52	-44.12	1.72	-44.88	1.62	-43.12
	4	1.34	-27.73	1.52	-42.01	1.52	-43.39
	5	0.98	-22.45	1.54	-42.32	1.58	-44.12
	6	1.78	-46.42	1.93	-47.87	1.62	-43.02
	7	1.52	-43.77	1.76	-44.91	1.52	-43.34
	8	1.74	-45.54	1.73	-43.97	1.73	-45.43
	9	1.57	-43.23	1.80	-46.67	2.09	-50.23
P3	1	0.32	-3.03	1.82	-47.02	0.36	-3.13
	2	**0.075**	**0.03**	1.36	-27.91	0.41	-4.03
	3	0.48	-4.12	0.54	-6.51	0.71	-10.04
	4	0.30	-2.98	2.57	-52.12	**0.22**	**-1.79**
	5	0.21	-1.67	0.73	-11.03	0.33	-3.67
	6	0.18	-1.77	0.61	-9.67	0.54	-5.63
	7	0.76	-11.02	**0.43**	**-3.98**	0.71	-10.87
	8	0.48	-4.34	0.62	-9.98	2.03	-50.23
	9	0.77	-11.42	2.31	-51.47	1.73	-45.33

4.4. *Comparison methods*

The TMO and RPOOT approaches are compared with the proposed method in three problem instances. The main idea of TMO is to track the moving optimum after each environmental change and choose the best found solution for deployment in each environment. Thus, we use mROOT with tw=1 to represent the TMO method. To the best of our knowledge, the RPOOT proposed by Chen *et al.* (35; 36) has made exploration in the field of multi-objective ROOT for DMOP. RPOOT uses MOEA/D as MOEA and MA model as a predictor to find robust Pareto-optimal solutions for DMOPs. Chen *et al.* (36) proposed two RPOOT models based on robustness and average fitness, respectively. According to the dDSP characteristics, the RPOOT based

Table 10. Average results (and standard deviation in parenthesis) obtained by mROOT with different values of τ for KSGA on P1. The highlighted entries are significantly better using Wilcoxon rank-sum test with $\alpha = 0.05$.

τ	$\overline{RHV}_{\text{mean}}$	$\overline{SC}_{\text{mean}}$	$\overline{RS}_{\text{mean}}$
0.15	14.95 (12.65)	5.35	**0.46 (4.46E-04)**
0.3	17.72 (13.72)	5.04 (0.09)	0.63 (5.09E-04)
0.5	**19.01 (17.23)**	**4.21 (0.08)**	0.79 (0.0024)

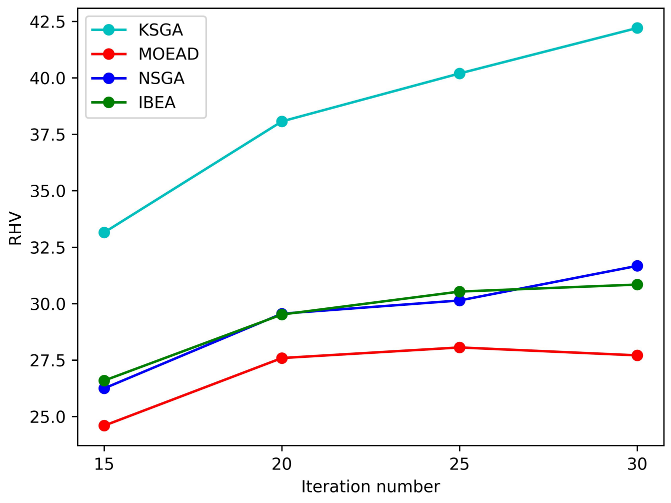


Figure 8. Average fitness of MOEAs on P3 with different number of iterations

on average fitness and choose the most stable solution from the PoF is applicable for this problem.

We use three representative multi-objective optimization algorithms (MOEAs) to in the evolutionary search part of the algorithm framework of mROOT: NSGA-II (46) that represents the MOEA based on Pareto dominance, the multi-objective evolutionary algorithm based on decomposition (MOEA/D) (54) which represents the MOEA based on decomposition, and Indicator-Based evolutionary algorithm (IBEA) (55) that represents the MOEA based on evaluation indicators.

5. Experimental analysis

The experiments in this section are carried out in three parts. In the first part, we investigate the effectiveness of using knee-oriented dominance relationship by comparing KSGA with three representative MOEAs in the framework of mROOT: NSGA-II (46) that represents the MOEA based on Pareto dominance, MOEA/D (54) which represents the MOEA based on decomposition, and IBEA (55) that represents the MOEA based on evaluation indicators. In the second part, we examine the effectiveness of the proposed online predictor by comparing the performance of the algorithm with MA model, AR model (most commonly used in the ROOT literature), and ours. Finally, we compare the proposed method with comparison algorithms.

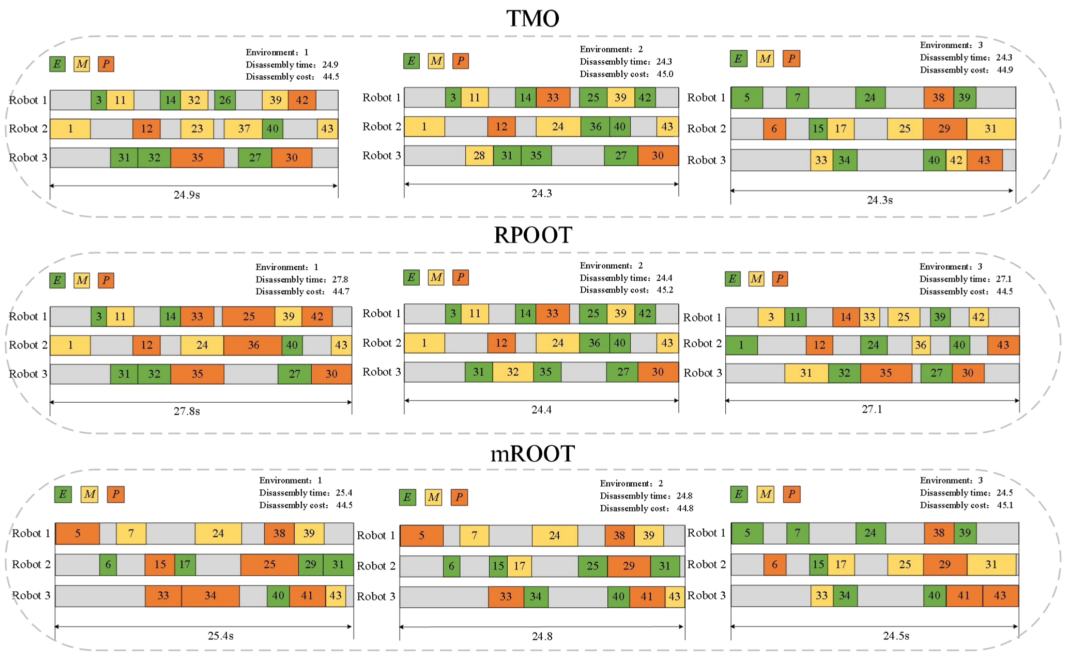


Figure 9. Disassembly sequences obtained by 3 different methods under 3 different disassembly environments

All experiments are performed on a computer with an Intel Core i7-9700 3.00GHz processor with 16 GB of RAM. The statistical results provided in this section are based on 30 independent runs in 100 environments. For statistical analysis, we use comparison tests using Wilcoxon rank-sum test with $\alpha = 0.05$ and the best results are highlighted in each table. If there are more than one highlighted results, it means they are not significantly different.

5.1. *Verification of disassembly sequence*

Before conducting comparative experiments, it is necessary to verify the correctness of the disassembly sequences obtained by the proposed dynamic disassembly sequence planning. We use the flashlight in Fig. 1 as an example to verify the validity of the disassembly sequences obtained by different methods. Fig. 2 shows the ETAOG of the flashlight, and Table 6 shows that the flashlight contains 7 components, 11 disassembly tasks, and a total of 50 operation steps.

Fig. 9 shows the performance of disassembly sequences obtained by 3 different methods under 3 different disassembly environments. Each of the small rectangles in the figure represents the disassembly operation performed by the robot. The small green, yellow, and red rectangles represent the operations performed on components with excellent, medium, and poor status, respectively. TMO tracks the optimal disassembly sequence in each environment, so its disassembly sequence changes with the environment. In contrast, RPOOT and mROOT use the same disassembly sequence in all three environments (within the time window), avoiding the cost incurred by re-optimization and switching solutions. The disassembly time and disassembly cost of the disassembly sequences obtained by RPOOT and mROOT change dynamically with the environment due to the different states of the EOL product components. TMO obtains the best fitness on all problem instances since it deploys the best found solution in each environment and does not keep any deployed solution for more than one environ-

Table 11. Average results (and standard deviation in parenthesis) obtained by mROOT with different MOEAs on different problem instances with *tw*=3 and AR model

Problem	MOEAs	$\overline{f_1}_{mean}$	$\overline{f_2}_{mean}$	$\overline{RHV}_{mean}$	$\overline{SC}_{mean}$	$\overline{RS}_{mean}$	$\overline{CT}_{mean}$
P1	KSGA	**24.97** **(0.88)**	44.24 (1.12)	**19.23** **(7.25)**	**4.48** **(0.10)**	**0.85** **(0.0012)**	**152.74** **(2.34)**
	NSGA-II	26.91 (1.21)	**43.35** **(1.10)**	10.86 (15.58)	5.76 (0.29)	0.88 (0.0016)	153.34 (2.98)
	MOEA/D	26.98 (1.24)	43.42 (1.08)	11.72 (7.13)	5.28 (0.20)	**0.84** **(0.0042)**	192.45 (3.24)
	IBEA	26.92 (1.13)	43.37 (1.11)	11.93 (10.28)	5.34 (0.19)	0.87 (0.0023)	229.98 (4.16)
P2	KSGA	**26.73** **(1.34)**	48.29 (2.13)	**31.61** **(11.21)**	**5.94** **(0.24)**	1.40 (0.0084)	**284.94** **(2.88)**
	NSGA-II	29.42 (2.31)	**47.04** **(2.01)**	21.25 (8.34)	7.16 (0.45)	**1.25** **(0.0012)**	**284.19** **(2.91)**
	MOEA/D	29.67 (2.39)	47.44 (2.89)	21.58 (10.23)	7.49 (0.16)	1.29 (0.0089)	296.23 (3.56)
	IBEA	29.24 (2.21)	47.39 (2.77)	22.35 (11.28)	7.03 (0.12)	1.30 (0.0089)	382.28 (4.78)
P3	KSGA	**38.72** **(2.01)**	55.56 (3.43)	**42.21** **(10.88)**	**9.34** **(0.55)**	1.77 (0.023)	598.19 (5.89)
	NSGA-II	41.92 (4.14)	**54.03** **(3.13)**	31.67 (6.31)	9.99 (0.38)	1.59 (0.014)	**578.99** **(5.38)**
	MOEA/D	42.53 (4.54)	54.36 (3.46)	27.71 (9.68)	10.83 (0.51)	**1.57** **(0.011)**	604.60 (6.01)
	IBEA	41.11 (4.46)	54.32 (3.88)	32.82 (7.54)	10.72 (0.19)	1.78 (0.024)	747.82 (8.01)

Table 12. Average results (and standard deviation in parenthesis) obtained by mROOT with different predictors on different time windows on P1

Predictor	*tw*	$\overline{f_1}_{mean}$	$\overline{f_2}_{mean}$	$\overline{RHV}_{mean}$	$\overline{SC}_{mean}$	$\overline{RS}_{mean}$	$\overline{CT}_{mean}$
XGBoost	3	**24.76** **(0.81)**	**44.21** **(1.01)**	**20.09** **(16.69)**	5.42 (0.13)	**0.77** **(0.0013)**	201.72 (5.01)
AR		24.97 (0.88)	44.24 (1.12)	19.15 (19.74)	**4.92** **(0.28)**	0.93 (0.0050)	**152.74** **(2.34)**
MA		24.88 (0.88)	44.28 (1.18)	19.25 (18.44)	5.24 (0.38)	0.86 (0.0034)	**152.77** **(2.39)**
XGBoost	4	**24.89** **(0.78)**	**44.26** **(0.94)**	**19.21** **(17.30)**	4.53 (0.11)	**0.83** **(0.0017)**	174.39 (4.13)
AR		25.01 (0.83)	44.31 (0.98)	18.94 (17.91)	**4.43** **(0.12)**	0.87 (0.0017)	**115.89** **(1.88)**
MA		25.13 (0.85)	44.43 (1.01)	18.09 (17.28)	4.55 (0.11)	**0.84** **(0.0037)**	116.13 (1.96)
XGBoost	5	**24.93** **(0.74)**	**44.25** **(0.91)**	**19.13** **(16.30)**	**3.35** **(0.10)**	**0.94** **(0.0015)**	138.34 (3.98)
AR		25.02 (0.81)	44.30 (0.94)	18.91 (11.93)	3.60 (0.10)	1.02 (0.0025)	**90.88** **(0.89)**
MA		25.05 (0.83)	44.38 (0.98)	18.69 (18.06)	3.55 (0.10)	0.97 (0.0014)	**90.96** **(0.92)**

ment. TMO optimizes and deploys the best found solution in each environment and thus has the best disassembly time and disassembly cost, while inevitably incurring switching costs. mROOT obtains disassembly sequences with less disassembly time and cost compared to RPOOT, and is more robust with less fluctuations within the time window.

5.2. *Analyzing the effectiveness of KSGA*

ROOT needs to select a solution with the best average fitness performance to be deployed at the beginning of each in a time window. However, choosing the appropriate solution to apply for MOPs with no explicit preferences is a challenge for decision makers. To address this issue, we proposed KSGA based on the knee-oriented dominance relationship (41) to find the global natural preference solution, narrow down the selection of solutions, and improve the average performance of solutions. In (8) the value of τ directly determines the size of the knee region which affects the dominance range of the knee points that is the decisive parameter affecting the knee-oriented dominance relationship.

Table 10 shows the effect of different values of τ on the performance of mROOT on P1. The solution search for the knee-oriented dominance relationship with $\tau = 0.5$ has the best average fitness and switching cost. This means that by increasing τ, the dominance range of the solution increases. As a result, the global knee points can be located with higher accuracy. However, the stability of the solution decreases as τ increases. Since mROOT selects the most stable solution from the PoF for application, this phenomenon represents that the expanded dominance range is at the expense of solution diversity. The small dominance range of $\tau = 0.15$ leads to difficulties in mutual dominance between many knee points which results in preserving the diversity of solutions. As τ increases, the expanded dominance region is able to more accurately locate the global knee points that maximizes the benefits by sacrificing a certain degree of solutions diversity. Considering the aforementioned observations and discussion, the value of τ for KSGA in the rest of experiments is set to 0.5.

Table 11 shows the results obtained by mROOT with different MOEAs on different problem instances with $tw = 3$ and AR model. KSGA finds solutions that perform significantly better on average fitness and lower switching costs in comparison to other MOEAs, which is more consistent with the main idea of mROOT to find solutions that perform well within a fixed time window. The performance of the two objectives shows that KSGA obtains a significant improvement on f_1 by sacrificing the partial fitness of the solution on f_2, which is the advantage of the knee points and thus leads to the best $\overline{RHV}$ for KSGA. KSGA gives up a certain degree of population diversity by finding global knee points, which narrows down the selection of solutions and leads to smaller PoF than other MOEAs. As a result, smaller PoF and solution selection strategy of mROOT (line 15 in Algorithm 1) makes KGSA perform mediocrely on $\overline{RS}$ and $\overline{CT}$. However, the diversity of populations increases the uncertainty in the selected solutions of other MOEAS, leading to unstable and unsatisfactory performance of all indicators under the influence of AR model prediction errors. Besides, reported results indicate large standard deviation for $\overline{RHV}$. The main reason is the average of hypervolume in all environments and the environmental changes cause the hypervolume change drastically from one environment to another, resulting in a large standard deviation. Considering that mROOT needs to select the best-performing solution for deployment, it is reasonable to sacrifice a certain degree of diversity to improve the performance of

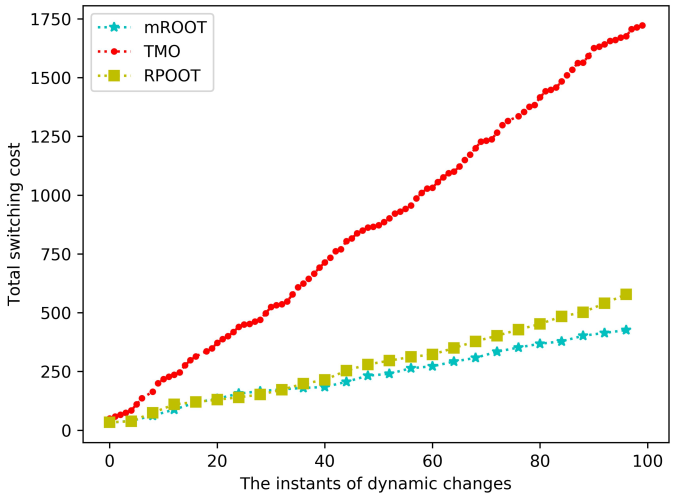


Figure 10. The switching process of the compared methods on P1

the solution using KSGA.

Fig. 8 shows the average fitness of MOEAs in P3 with different numbers of iterations. By increasing the number of iterations, all MOEAs can search for solutions with better average fitness. As can be seen in this figure, KSGA outperforms other MOEAs in terms of average fitness because it focuses on searching global knee points. In addition, by increasing the number of iterations, KSGA maintains its superiority.

5.3. *Analyzing the effectiveness of online predictor*

Table 12 shows the results obtained by mROOT with different prediction components with different time windows in P1. The average fitness and stability of the solution obtained by the mROOT equipped with the proposed online predictor are the best for all time windows since this predictor improves the accuracy of the prediction. However, XGBoost requires real-time training and invocation of the model for prediction, causing it to be more time-consuming than the time-series prediction approach. The performance of online predictors on switching cost is mediocre, which imply that the average fitness and stability of the solutions may conflict with the switching cost. A potential reason is that the structure of the solution that maintains good performance and stability in the new environment is more different from the old one due to the severe change in the environment, and thus a certain switching cost is inevitable to ensure the performance of the solution. However, the overall performance of the proposed online predictor is better than the AR and MA models. The experimental results show that by increasing tw, the differences between the performance of predictors decrease. The reason may be that the definition of average fitness reduces the interference caused by prediction error through averaging. The larger the time window tw, the smaller the difference between the predicted average fitness and the actual average fitness. Moreover, increasing the time window will reduce the switching times of the solution, thereby reducing the switching cost. However, it is difficult to find solutions with good average performance and low volatility within a long time window. As a result, the average fitness and stability of the solution decrease by the increasing time window.

Table 13. Average results (and standard deviation in parenthesis) obtained by different methods on different problem instances and different time windows

Problem	Method	tw	$\overline{f_1}_{mean}$	$\overline{f_2}_{mean}$	$\overline{RHV}_{mean}$	$\overline{SC}_{mean}$	$\overline{RS}_{mean}$	$\overline{CT}_{mean}$
P1	TMO	1	24.52 (0.77)	44.31 (0.81)	21.08 (12.01)	15.06 (0.58)	-	446.83 (8.89)
	RPOOT	3	26.81 (2.13)	**43.50 (1.98)**	12.10 (11.82)	7.43 (0.353)	**0.75 (0.0015)**	211.34 (3.78)
	mROOT	3	**24.76 (0.81)**	44.21 (1.01)	**20.09 (16.69)**	**5.42 (0.13)**	0.77 (0.0013)	**201.72 (5.01)**
	RPOOT	4	27.80 (2.47)	**43.43 (1.69)**	12.09 (21.88)	5.61 (0.34)	0.85 (0.0042)	179.04 (2.98)
	mROOT	4	**24.89 (0.78)**	44.26 (0.94)	**19.21 (17.30)**	**4.53 (0.11)**	**0.83 (0.0017)**	**174.39 (4.13)**
	RPOOT	5	26.80 (2.01)	**43.65 (1.34)**	11.69 (17.78)	4.58 (0.34)	0.93 (0.0012)	**124.45 (2.00)**
	mROOT	5	**24.93 (0.74)**	44.25 (0.91)	**19.13 (16.30)**	**3.35 (0.10)**	**0.94 (0.0015)**	138.34 (3.98)
P2	TMO	1	26.01 (0.91)	48.59 (0.98)	33.76 (10.35)	20.21 (0.51)	-	837.83 (11.34)
	RPOOT	3	29.62 (2.41)	**47.57 (3.01)**	21.58 (15.40)	9.88 (0.27)	1.15 (0.013)	298.34 (5.78)
	mROOT	3	**26.33 (2.18)**	48.24 (2.93)	**32.58 (13.56)**	**8.04 (0.09)**	**0.89 (0.002)**	**284.94 (5.01)**
	RPOOT	4	29.97 (2.47)	**47.68 (2.69)**	19.09 (14.81)	7.29 (0.31)	1.33 (0.0071)	**226.54 (4.18)**
	mROOT	4	**26.48 (2.01)**	48.38 (2.94)	**30.88 (9.34)**	**5.87 (0.08)**	1.40 (0.0084)	241.39 (4.33)
	RPOOT	5	30.66 (2.61)	**47.36 (2.34)**	18.35 (16.20)	5.98 (0.14)	1.45 (0.013)	**195.13 (3.89)**
	mROOT	5	**26.61 (1.94)**	49.04 (2.78)	**28.45 (14.23)**	**5.13 (0.05)**	**1.18 (0.0032)**	212.98 (3.71)
P3	TMO	1	37.71 (1.03)	56.05 (1.14)	43.66 (14.44)	32.41 (2.26)	-	1739.52 (18.39)
	RPOOT	3	41.86 (4.13)	**54.61 (4.48)**	29.09 (15.12)	15.26 (0.31)	1.35 (0.012)	**570.79 (10.78)**
	mROOT	3	**38.17 (3.15)**	55.49 (3.98)	**42.26 (11.77)**	**12.45 (0.68)**	1.32 (0.007)	597.75 (11.01)
	RPOOT	4	41.93 (3.97)	**54.62 (4.37)**	26.46 (15.75)	10.68 (0.53)	**1.57 (0.088)**	**418.29 (9.18)**
	mROOT	4	**38.30 (3.08)**	55.79 (3.94)	**41.04 (11.86)**	**9.37 (0.52)**	1.68 (0.067)	438.94 (9.81)
	RPOOT	5	41.98 (3.71)	**54.82 (3.92)**	27.56 (16.21)	8.94 (0.14)	1.73 (0.0069)	**332.92 (7.10)**
	mROOT	5	**38.52 (2.74)**	55.63 (3.91)	**40.38 (12.71)**	**7.45 (0.37)**	**1.55 (0.0068)**	351.39 (8.98)

5.4. *Comparison with other methods*

Table 13 shows the results obtained by different methods on different problem instances and different time windows. TMO obtains the best fitness on all problem instances since it deploys the best found solution in each environment and does not keep any deployed solution for more than one environment. Consequently, frequently switching the deployed solution leads to a significant switching cost and searching time. RPOOT and mROOT reduce the switching cost by finding solutions with good average fitness over multiple environments. The more accurate predictor of mROOT makes up for some of the disadvantages of KSGA on $\overline{RS}$, so there is no clear winner for mROOT and RPOT on $\overline{RS}$. Moreover, since mROOT benefits from a better online predictor and search for natural preference solutions through KSGA, it significantly outperforms RPOOT in average fitness and switching cost. mROOT takes advantage of the knee points to obtain a better $\overline{RHV}$ by sacrificing a small portion of the f_2 fitness to significantly improve the performance of f_1. Compared with MOEA/D, the advantage of KSGA in search time makes up for the time-consuming characteristics of the online predictor in mROOT, so RPOOT and mROOT are comparable in time consumption. As the number of operations in different problems instances increases, the switching cost also increases gradually. Our method reduces the switching cost by sacrificing a small amount of fitness to avoid the inconvenience of frequent search.

Fig. 10 shows the switching cost over time obtained by the compared methods on P1. Due to frequently switching solutions, TMO suffers from huge switching costs over the disassembly process. While mROOT benefits from KSGA and online predictor, it can obtain the minimum switching cost. In this work, the value of the switching cost only indicates the consumption incurred by the switching solution. In real-life applications, the computational resources consumed by frequent optimization of TMO in each environment is also significant.

6. Conclusion

To model the uncertain DSP more accurately, we construct a mathematical model of dDSP by considering the uncertainty of the state of EOL products and operators. In dDSP, the performance of the disassembly sequence is subject to environmental influences. mROOT consists of several components including a new predictor, definitions of robustness for multi-objective, KSGA based on the knee-oriented dominance relationship (41) is proposed to find a solution for the dDSP within a fixed time window. We tested mROOT on different problem instances and provided performance analysis based on them. The experimental results demonstrate the effectiveness of the new predictor and KSGA. mROOT reduces the switching cost to avoid the inconvenience of TMO and outperforms the state-of-the-art ROOT method in the multi-objective domain.

To simplify the problem, this study sets the operation time and cost of the robot to handle products with different states as fixed numbers. In practice, uncertain product states tend to make the operation time and consumption vary to some extent. In the future, the use of interval numbers to represent operation time and consumption can better describe the uncertainty in actual disassembly. Furthermore, much valid information from the historical environment can be considered for reuse to guide the judgment of mROOT on the future environment through transfer learning.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This work was supported by in part the National Natural Science Foundation of China under Grant 52075402 and Grant 52005376.

References

[1] V. R. Guide, Production planning and control for remanufacturing: industry practice and research needs, Journal of Operations Management 18 (4) (2000) 467–483. doi:10.1016/S0272-6963(00)00034-6.

[2] A. J. D. Lambert, Disassembly sequencing: A survey, International Journal of Production Research 41 (16) (2003) 3721–3759. doi:10.1080/0020754031000120078.

[3] W.-C. Yeh, 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) (2011) 250–261. doi:10.1109/TSMCA.2011.2157135.

[4] W.-C. Yeh, Simplified swarm optimization in disassembly sequencing problems with learning effects, Computers & Operations Research 39 (9) (2012) 2168–2177. doi:10.1016/j.cor.2011.10.027.

[5] Y. Luo, Q. Peng, Disassembly sequence planning for product maintenance, in: International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Vol. 45042, American Society of Mechanical Engineers, 2012, pp. 601–609. doi:https://doi.org/10.1115/DETC2012-70430.

[6] Y. Luo, Q. Peng, P. Gu, Integrated multi-layer representation and ant colony search for product selective disassembly planning, Computers in Industry 75 (2016) 13–26. doi:10.1016/j.compind.2015.10.011.

[7] K. Xia, L. Gao, W. Li, K.-M. Chao, Disassembly sequence planning using a simplified teaching-learning-based optimization algorithm, in: Sustainable Manufacturing and Remanufacturing Management, Springer, 2019, pp. 319–343. doi:10.1016/j.aei.2014.07.006.

[8] K. Xia, L. Gao, L. Wang, W. Li, K.-M. Chao, A simplified teaching-learning-based optimization algorithm for disassembly sequence planning, in: 2013 IEEE 10th International Conference on e-Business Engineering, IEEE, 2013, pp. 393–398. doi:10.1109/ICEBE.2013.60.

[9] X. F. Zhang, G. Yu, Z. Y. Hu, C. H. Pei, G. Q. Ma, Parallel disassembly sequence planning for complex products based on fuzzy-rough sets, The International Journal of Advanced Manufacturing Technology 72 (1) (2014) 231–239. doi:10.1007/s00170-014-5655-4.

[10] X. F. Zhang, S. Y. Zhang, Product cooperative disassembly sequence planning based on branch-and-bound algorithm, The International Journal of Advanced Manufacturing Technology 51 (9) (2010) 1139–1147. doi:10.1007/s00170-010-2682-7.

[11] L. H. De Mello, A. C. Sanderson, And/or graph representation of assembly plans, IEEE Transactions on robotics and automation 6 (2) (1990) 188–199. doi:10.1109/70.54734.

[12] A. Koc, I. Sabuncuoglu, E. Erel, Two exact formulations for disassembly line balancing problems with task precedence diagram construction using an AND/OR graph, IIE Transactions 41 (10) (2009) 866–881. doi:10.1080/07408170802510390.

[13] A. Gungor, S. M. Gupta, Disassembly sequence planning for products with defective

parts in product recovery, Computers & Industrial Engineering 35 (1-2) (1998) 161–164. doi:10.1016/S0360-8352(98)00047-3.

[14] G. Tian, Y. Liu, H. Ke, J. Chu, Energy evaluation method and its optimization models for process planning with stochastic characteristics: A case study in disassembly decision-making, Computers & Industrial Engineering 63 (3) (2012) 553–563. doi:10.1016/j.cie.2011.08.011.

[15] Meimei Gao, MengChu Zhou, R. Caudill, Integration of disassembly leveling and bin assignment for demanufacturing automation, IEEE Trans. Robot. Automat. 18 (6) (2002) 867–874. doi:10.1109/TRA.2002.805650.

[16] M. A. Ilgin, S. M. Gupta, Environmentally conscious manufacturing and product recovery (ECMPRO): A review of the state of the art, Journal of Environmental Management 91 (3) (2010) 563–591. doi:10.1016/j.jenvman.2009.09.037.

[17] L. Brennan, S. M. Gupta, K. N. Taleb, Operations Planning Issues in an Assembly/Disassembly Environment, Int Jrnl of Op & Prod Mnagemnt 14 (9) (1994) 57–67. doi:10.1108/01443579410066767.

[18] G. Tian, M. Zhou, J. Chu, Y. Liu, Probability Evaluation Models of Product Disassembly Cost Subject to Random Removal Time and Different Removal Labor Cost, IEEE Trans. Automat. Sci. Eng. 9 (2) (2012) 288–295. doi:10.1109/TASE.2011.2176489.

[19] G. Tian, Y. Liu, Q. Tian, J. Chu, Evaluation model and algorithm of product disassembly process with stochastic feature, Clean Techn Environ Policy 14 (2) (2012) 345–356. doi:10.1007/s10098-011-0406-9.

[20] G. Tian, M. Zhou, J. Chu, A Chance Constrained Programming Approach to Determine the Optimal Disassembly Sequence, IEEE Trans. Automat. Sci. Eng. 10 (4) (2013) 1004–1013. doi:10.1109/TASE.2013.2249663.

[21] G. Tian, J. Chu, T. Qiang, Influence factor analysis and prediction models for component removal time in manufacturing, Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture 227 (10) (2013) 1533–1540. doi:10.1177/0954405413489292.

[22] G. Tian, J. Chu, H. Hu, H. Li, Technology innovation system and its integrated structure for automotive components remanufacturing industry development in China, Journal of Cleaner Production 85 (2014) 419–432. doi:10.1016/j.jclepro.2014.09.020.

[23] G. Tian, M. Zhou, P. Li, Disassembly Sequence Planning Considering Fuzzy Component Quality and Varying Operational Cost, IEEE Trans. Automat. Sci. Eng. 15 (2) (2018) 748–760. doi:10.1109/TASE.2017.2690802.

[24] Y. Laili, F. Ye, Y. Wang, L. Zhang, Interference probability matrix for disassembly sequence planning under uncertain interference, Journal of Manufacturing Systems 60 (2021) 214–225. doi:10.1016/j.jmsy.2021.05.014.

[25] Z. Zhou, J. Liu, P. D. Truong, W. Xu, R. F. Javier, C. Ji, L. Quan, Disassembly sequence planning: Recent developments and future trends, Proceedings of the Institution of Mechanical Engineers Part B Journal of Engineering Manufacture 233 (2018) 095440541878997. doi:10.1177/0954405418789975.

[26] D. Yazdani, J. Branke, M. N. Omidvar, T. T. Nguyen, X. Yao, Changing or keeping solutions in dynamic optimization problems with switching costs, in: Proceedings of the Genetic and Evolutionary Computation Conference, 2018, pp. 1095–1102. doi:doi.org/10.1145/3205455.3205484.

[27] X. Yu, Y. Jin, K. Tang, X. Yao, Robust optimization over time — a new perspective on dynamic optimization problems, in: IEEE Congress on Evolutionary Computation, 2010, pp. 1–6. doi:10.1109/CEC.2010.5586024.

[28] Y. Jin, K. Tang, X. Yu, B. Sendhoff, X. Yao, A framework for finding robust optimal solutions over time, Memetic Comp. 5 (1) (2013) 3–18. doi:10.1007/s12293-012-0090-2.

[29] H. Fu, B. Sendhoff, K. Tang, X. Yao, Finding robust solutions to dynamic optimization problems, in: Applications of Evolutionary Computation, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 616–625. doi:10.1007/978-3-642-37192-9_62.

[30] Fu, Haobo and Sendhoff, Bernhard and Tang, Ke and Yao, Xin, Robust optimization over

time: Problem difficulties and benchmark problems, IEEE Transactions on Evolutionary Computation 19 (5) (2015) 731–745. doi:10.1109/TEVC.2014.2377125.

[31] Y.-n. Guo, M. Chen, H. Fu, Y. Liu, Find robust solutions over time by two-layer multi-objective optimization method, in: 2014 IEEE Congress on Evolutionary Computation (CEC), IEEE, Beijing, China, 2014, pp. 1528–1535. doi:10.1109/CEC.2014.6900241.

[32] Y. Huang, Y. Ding, K. Hao, Y. Jin, A multi-objective approach to robust optimization over time considering switching cost, Information Sciences 394-395 (2017) 183–197. doi:10.1016/j.ins.2017.02.029.

[33] Y. Huang, Y. Jin, K. Hao, Decision-making and multi-objectivization for cost sensitive robust optimization over time, Knowledge-Based Systems 199 (2020) 105857. doi:10.1016/j.knosys.2020.105857.

[34] D. Yazdani, T. T. Nguyen, J. Branke, Robust Optimization Over Time by Learning Problem Space Characteristics, IEEE Trans. Evol. Computat. 23 (1) (2019) 143–155. doi:10.1109/TEVC.2018.2843566.

[35] M. Chen, Y. Guo, H. Liu, C. Wang, The Evolutionary Algorithm to Find Robust Pareto-Optimal Solutions over Time, Mathematical Problems in Engineering 2015 (2015) 1–18. doi:10.1155/2015/814210.

[36] M. R. Chen, Y. N. Guo, D. W. Gong, Z. Yang, A Novel Dynamic Multi-objective Robust Evolutionary Optimization Method, Zidonghua Xuebao/acta Automatica Sinica 43 (11) (2017) 2014–2032. doi:10.16383/j.aas.2017.c160300.

[37] D. Yazdani, R. Cheng, D. Yazdani, J. Branke, Y. Jin, X. Yao, A survey of evolutionary continuous dynamic optimization over two decades—part b, IEEE Transactions on Evolutionary Computation 25 (4) (2021) 630–650. doi:10.1109/TEVC.2021.3060012.

[38] K. Deb, S. Gupta, Understanding knee points in bicriteria problems and their implications as preferred solution principles, Engineering optimization 43 (11) (2011) 1175–1204. doi:10.1080/0305215X.2010.548863.

[39] X. Zhang, Y. Tian, Y. Jin, A knee point-driven evolutionary algorithm for many-objective optimization, IEEE Transactions on Evolutionary Computation 19 (6) (2014) 761–776. doi:10.1109/tevc.2014.2378512.

[40] J. Zou, Q. Li, S. Yang, H. Bai, J. Zheng, A prediction strategy based on center points and knee points for evolutionary dynamic multi-objective optimization, Applied soft computing 61 (2017) 806–818. doi:10.1016/j.asoc.2017.08.004.

[41] G. Yu, Y. Jin, M. Olhofer, A multiobjective evolutionary algorithm for finding knee regions using two localized dominance relationships, IEEE Transactions on Evolutionary Computation 25 (1) (2021) 145–158. doi:10.1109/TEVC.2020.3008877.

[42] Y. Tang, M. Zhou, E. Zussman, R. Caudill, Disassembly modeling, planning, and application, Journal of Manufacturing Systems 21 (3) (2002) 200–217. doi:10.1016/S0278-6125(02)80162-5.

[43] Y. Fang, H. Xu, Q. Liu, D. T. Pham, Evolutionary optimization using epsilon method for resource-constrained multi-robotic disassembly line balancing, Journal of Manufacturing Systems 56 (2020) 392–413. doi:10.1016/j.jmsy.2020.06.006.

[44] G. Tian, M. Zhou, J. Chu, Y. Liu, Probability evaluation models of product disassembly cost subject to random removal time and different removal labor cost, IEEE Transactions on Automation Science and Engineering 9 (2) (2012) 288–295. doi:10.1109/TASE.2011.2176489.

[45] B. Hu, Y. Feng, H. Zheng, J. Tan, Sequence planning for selective disassembly aiming at reducing energy consumption using a constraints relation graph and improved ant colony optimization algorithm, Energies 11 (8) (2018) 2106. doi:doi.org/10.3390/en11082106.

[46] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Trans. Evol. Computat. 6 (2) (2002) 182–197. doi:10.1109/4235.996017.

[47] T. Chen, C. Guestrin, XGBoost: A Scalable Tree Boosting System, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, San Francisco California USA, 2016, pp. 785–794. doi:10.1145/2939672.2939785.

[48] Y. Fang, Q. Liu, M. Li, Y. Laili, D. T. Pham, Evolutionary many-objective optimization for mixed-model disassembly line balancing with multi-robotic workstations, European Journal of Operational Research 276 (1) (2019) 160–174. doi:10.1016/j.ejor.2018.12.035.

[49] G. Pavai, T. Geetha, A survey on crossover operators, ACM Computing Surveys (CSUR) 49 (4) (2016) 1–43.

[50] E. Zitzler, L. Thiele, Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach, IEEE transactions on Evolutionary Computation 3 (4) (1999) 257–271. doi:10.1109/4235.797969.

[51] Y.-S. Ma, H.-B. Jun, H.-W. Kim, D.-H. Lee, Disassembly process planning algorithms for end-of-life product recovery and environmentally conscious disposal, International Journal of Production Research 49 (23) (2011) 7007–7027. doi:10.1080/00207543.2010.495089.

[52] A. Lambert, Optimizing disassembly processes subjected to sequence-dependent cost, Computers & Operations Research 34 (2) (2007) 536–551. doi:10.1016/j.cor.2005.03.012.

[53] R. Lotfi, Y. Z. Mehrjerdi, N. Mardani, A multi-objective and multi-product advertising billboard location model with attraction factor mathematical modeling and solutions, International Journal of Applied Logistics 7 (1) (2017) 64–87. doi:10.4018/IJAL.2017010104.

[54] Q. Zhang, H. Li, Moea/d: A multiobjective evolutionary algorithm based on decomposition, IEEE Transactions on evolutionary computation 11 (6) (2007) 712–731. doi:10.1109/TEVC.2007.892759.

[55] E. Zitzler, S. Künzli, Indicator-based selection in multiobjective search, in: International conference on parallel problem solving from nature, Springer, 2004, pp. 832–842. doi:10.1007/978-3-540-30217-9_84.

[56] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, V. Fonseca, Performance assessment of multiobjective optimizers: an analysis and review, IEEE Transactions on Evolutionary Computation 7 (2) (2003) 117–132. doi:10.1109/TEVC.2003.810758.

[57] R. Lotfi, Y. Z. Mehrjerdi, N. Mardani, A multi-objective and multi-product advertising billboard location model with attraction factor mathematical modeling and solutions, International Journal of Applied Logistics 7 (1) (2017) 64–87. doi:10.4018/IJAL.2017010104.