# An improved multi-objective discrete bees algorithm for robotic disassembly line balancing problem in remanufacturing

Liu, Jiayi; Zhou, Zude; Pham, Duc Truong; Xu, Wenjun; Yan, Junwei; Liu, Aiming; Ji, Chunqian; Liu, Quan

*DOI:*
10.1007/s00170-018-2183-7

# An improved multi-objective discrete Bees algorithm for robotic disassembly line balancing problem in remanufacturing

Jiayi Liu[1,2], Zude Zhou[1,2], Duc Truong Pham[3], Wenjun Xu[1,4,2], Junwei Yan[1,4,2,*], Aiming Liu[1,2], Chunqian Ji[3], Quan Liu[1,2]

[1]*School of Information Engineering, Wuhan University of Technology, Wuhan 430070, China*

[2]*Hubei Key Laboratory of Broadband Wireless Communication and Sensor Networks (Wuhan University of Technology), Wuhan 430070, China*

~~[2]Key Laboratory of Fiber Optic Sensing Technology and Information Processing (Wuhan University of Technology), Ministry of Education, Wuhan 430070, China~~

~~[3]Department~~ [3]*Department of Mechanical Engineering, University of Birmingham, Birmingham B15 2TT, UK*

~~[4]Hubei Key Laboratory of Broadband Wireless Communication and Sensor Networks (Wuhan University of Technology), Wuhan 430070, China~~

Email: jyliu@whut.edu.cn; zudezhou@whut.edu.cn; d.t.pham@bham.ac.uk; xuwenjun@whut.edu.cn; junweiyan@whut.edu.cn; liuaiming@cbmi.com.cn; c.ji@bham.ac.uk; quanliu@whut.edu.cn

Formatted: Font: Not Italic

**Abstract** Remanufacturing is an effective way to realize the reutilization of resources. Disassembly, as an essential step of remanufacturing, is usually finished by manual work which is low efficiency and high labor cost. Robotic disassembly provides an alternative way to reduce labor intensity and disassembly cost. Disassembly line is an efficient method to deal with end-of-life products on a large scale. To balance the workload of robotic workstations is the main objective of robotic disassembly line balancing problem. In this paper, an improved multi-objective discrete Bees algorithm is proposed to solve robotic disassembly line balancing problem. The feasible disassembly sequence is obtained by space interference matrix method. It is used to generate robotic disassembly line solution by robotic workstation assignment method. After that, the multi-objective robotic disassembly line balancing problem is proposed. With the help of efficient non-dominated Pareto sorting method, improved multi-objective discrete Bees algorithm is proposed to find Pareto optimal solutions. Based on a gear pump and a camera, the performance of improved multi-objective discrete Bees algorithm is analyzed under different parameters and compared with the other optimization algorithms. In addition, Pareto fronts of robotic disassembly line balancing problem are also compared with those of the other two cases. The result shows the proposed method can find better quality of solutions using comparable running time compared with the other optimization algorithms.

**Keywords** Remanufacturing • Robotic disassembly line balancing problem • Improved multi-objective discrete bees algorithm • Robotic disassembly • Optimization problems

## 1. Introduction

Traditional manufacturing industry has the disadvantages of high environmental pollution and low resource utilizations. Under this condition, sustainable manufacturing [1] and remanufacturing [2, 3] are the future trends of manufacturing industry. In China, according to statistics, there are approximately 5 million cars, 20 million mobile phones and 500 million tons of solid waste to be scrapped every year [4].

Products are usually incinerated or buried after exceeding the service life. This always leads to resource waste and environmental pollution. Remanufacturing, which aims to recover the size and performance of used products [5], closes the materials loop and forms a closed-loop manufacturing system. According to statistics, the energy and resource consumption of remanufacturing a product are respectively a half and seven tenths of manufacturing a new one [6]. Thus, remanufacturing provides an economically and environmentally sound way to realize sustainable developments of manufacturing industry [7, 8]. In all steps of remanufacturing, disassembly is an inevitable process to handle end-of-life products (EoLP) [9-11]. Due to the uncertainty of EoLP's quality, traditional disassembly process is always finished by manual works [12]. To reduce labor intensity, much attention has been paid to robotic disassembly [13, 14]. The cognitive robotics used for robotic disassembly was proposed to handle uncertainties in the dynamic disassembly process [12]. Afterwards, the basic behavior control [8] and learning/revision strategies [15] of cognitive robotics were proposed to disassemble the liquid-crystal display (LCD) screens.

Successful production planning helps to improve capacity utilization [16, 17]. In the production planning process, a well-designed assembly/disassembly line [18] helps to improve production efficiency. The assembly/disassembly line balancing problem is a non-negligible problem for the assembly/disassembly line. The key problem of line balancing problem is to assign manufacturing tasks to a set of workstations in a balanced way [10]. Disassembly line consists of several disassembly workstations which are assigned different disassembly tasks [19]. The core objective of disassembly line balancing problem (DLBP) is to balance the workload of workstations. To find optimal disassembly line balancing solution of an EoLP with $n$ parts, it requires investigating all the permutations of disassembly sequence ($n!$ possible solutions). Searching this solution space is bounded by exponential time. The decision version of DLBP has been proven to be NP-complete problem and its optimization version is NP-hard problem [20]. Many researchers have studied DLBP. To disassemble smart phones, a linear physical programming method was proposed to balance the mixed-model disassembly line [21]. Based on a cell phone which contains 25 parts, Ding et al. [22] used multi-objective ant colony algorithm to simultaneously minimize the number of workstation, the demand rating and the measure of balance. After that, to optimize the balancing index and the design cost of disassembly line, genetic algorithm was used to find the optimal solutions of stochastic DLBP [23]. The reinforcement learning algorithm was also used to solve multi-objective DLBP within reasonable computation time [24]. Bentaha et al. [25] used known probability distributions to describe the disassembly task time and solved DLBP by two-stage stochastic linear mixed integer program and sample average approximation method. To minimize the number of workstations, Hezer et al. [26] used the shortest route model to solve parallel DLBP. Considering the uncertainties in disassembly process, hybrid discrete artificial bee colony algorithm was proposed to solve fuzzy DLBP [27]. To find optimal disassembly line balancing solutions within reasonable computation time, Mete et al. [28] used beam search algorithm to minimize the number of workstations. Recently, sequence-dependent DLBP (SDDLBP) is gradually paid more attention. In SDDLBP, if task A interacts with another task B (such as physical obstruction *etc.*), additional disassembly time should be considered. Kalayci et al. [29] used ant colony optimization to minimize the number of workstation, the demand rating, hazardous rating and balancing index of disassembly line. Particle swarm optimization algorithm with neighborhood-based mutation operator was also used to find multi-objective optimal solutions of SDDLBP [30]. Improved artificial bee colony algorithm was used to find optimal disassembly line balancing solutions with minimum environmental impact and maximum profit [31]. After that, tabu search [32], hybrid genetic [33] and artificial bee colony [34] algorithms were used to solve SDDLBP.

The robotic disassembly line is different from manual disassembly line. For manual disassembly line, the working efficiency of each workstation is influenced by the workload while for robotic disassembly line, it is not. In addition, in manual disassembly line, current researches always ignore the moving time between different parts in each workstation. When robotic disassembly line balancing problem (RDLBP) is considered, the moving path of industrial robot's end-effector should be considered to avoid obstacles caused by the contour of EoLP. However, it is always ignored in the existing researches. Until now, there is no research studies RDLBP. But there are similar researches in robotic disassembly sequence planning. Based on the personal computer, Alshibli et al. [35] used tabu search and genetic algorithm to find optimal disassembly sequence with minimum disassembly time. ElSayed et al. [36] used genetic algorithm to solve robotic disassembly sequencing problem and an intelligent disassembly cell was also proposed [37]. Based on the personal computer, an online genetic algorithm was used to minimize the total disassembly time [38]. Generally, the moving time of industrial robot's end-effector between different parts should be a part of total disassembly time for each workstation. However, in the aforementioned methods, the moving time is calculated by Euclidean distance between different parts and the moving speed of industrial robot's end-effector. It ignores obstacle caused by the contour of EoLP. When robotic disassembly is considered, the obstacle-avoiding path of industrial robot's end-effector should be considered.

Many meta-heuristic algorithms are used to solve DLBP such as genetic algorithm [20], ant colony optimization [39], *etc.* It is the first application of BA to solve RDLBP. BA is derived from the foraging behavior of bees and has been successfully applied in many fields. The Pareto-based multi-objective Bees algorithm (MOBA) was used to solve welded beam design problem [40]. To maximize the slackness index and the efficiency of the assembly line, BA was used to solve two-sided assembly line balancing problem under constrained fuzzy environment [41]. Ercin et al. [42] used multi-objective BA and artificial Bees colony algorithm to solve proportional-integral-derivative (PID) tuning problems. To minimize the total lead-time and the total cost, BA was used to find optimal configuration of supply chain networks [43]. Lu et al. [44] used BA to solve quality-of-experience (QoE) based spectrum allocation optimization problem of cognitive radio networks. Based on motorcycle assembly process, Xu et al. [45] used multi-objective BA to solve correlation-aware service aggregation optimization problem. The performance of BA was compared with multi-objective genetic algorithm and multi-objective particle swarm optimization algorithm.

DLBP is multi-objective optimization problem which can be solved by lexicographical optimization methods, objective weighting optimization methods and Pareto based optimization methods [46]. Comparing with the former two methods, Pareto based multi-objective optimization method treats all the objectives equally and provides several non-dominated solutions for the decision-makers to choose. However, recently, most of optimization algorithms used for solving DLBP is lexicographic-based multi-objective optimization algorithms [32-34] and few studies have reported Pareto-based multi-objective optimization algorithms to solve DLBP.

The major contribution of this paper is to solve robotic disassembly line balancing problem which has not been studied yet. Until now, no research considers the moving time between different parts as a part of total working time of robotic workstation, which can be calculated by the obstacle-avoiding path and moving speed of industrial robot's end-effector. In this paper, the significance and related works are discussed in section 1. The notations used in this paper are summarized in Section 2. In Section 3, disassembly model is built by space interference matrix method (SIMM) and robotic disassembly line solution is obtained by robotic workstation assignment method. In Section 4, multi-objective robotic disassembly line balancing problem is described. Then, improved multi-objective discrete Bees algorithm

(IMODBA) are introduced in Section 5. In Section 6, case studies based on a gear pump and a camera are used to verify the effectiveness of proposed method and conclusions are drawn in Section 7.

## 2. Notation

| | |
|---|---|
| $A$ | Allocation matrix used for robotic workstation assignment method |
| $bt$ | Basic disassembly time |
| $bdn$ | Number of solutions which dominate the provided solution |
| $CT$ | Cycle time of robotic disassembly line |
| $D_{set}$ | A set of solutions which is dominated by the provided solution |
| $d_i$ | The Euclidean distance between $i^{th}$ member of obtained non-dominated solutions and its nearest member of Pareto optimal solutions |
| $dt$ | The direction-change time |
| $FSeq$ | Feasible disassembly sequences generated by SIMM |
| $Fr$ | Set of Pareto rankings |
| $Fr_i$ | Pareto ranking $i$ |
| $h_{pi}$ | Demand; quantity of part $p_i$ requested |
| $iter$ | Iteration number |
| $m$ | Number of robotic workstation |
| $md$ | Length of moving path between different disassembly parts |
| $ms$ | Moving speed of industrial robot's end-effector |
| $mt$ | Moving time between different disassembly parts |
| $n$ | Number of parts in an EoLP |
| $nd$ | Number of non-dominated solutions |
| $nes$ | Number of elite sites |
| $ns$ | Number of selected sites |
| $P$ | Population set |
| $P^*$ | Sorted population set |
| $P_{Fr}$ | Pareto sorted population set |
| $rnes$ | Number of follower bees around elite sites |
| $rns$ | Number of follower bees around non-elite selected sites |
| $scoutn$ | Number of scout bee |
| $Seq$ | Disassembly sequences |
| $S_d$ | Space interference matrix ($d$ = x+, x-, y+, y-, z+, z-) |
| $S_{x,y,z}$ | Integrated space interference matrix |
| $t_{i,total}$ | Total working time of $i^{th}$ robotic workstation |
| $t_{pi}$ | Basic disassembly time for part $p_i$ |
| $tta$, $tt$ | Tool-change time |

## 3. Disassembly model for EoLP

The disassembly model is always built by graph-based method [47, 48], Petri Net method [49] and matrix-based method [50, 51], *etc*. For robotic disassembly, feasible disassembly direction of each part should be provided for industrial robots. In this paper, SIMM [50, 51] which consists of space interference matrix and interference matrix analysis is used to generate feasible disassembly sequence and

disassembly direction.

3.1 Space interference matrix

Firstly, six interference matrices along six directions (x+, x-, y+, y-, z+ and z-) are described by Equation (1).

In the matrix $S_d$, element $s_{i,j}$ means whether component $C_i$ can be removed along $d$ direction ($d$ = x+, x-, y+, y-, z+ or z-) when component $C_j$ exists. If component $C_i$ can be removed along $d$ direction when component $C_j$ exists, element $s_{i,j}$ in the matrix $S_d$ is 0. Otherwise, it is 1. As shown in Figure 1, six interference matrices of this example are described by Equation (2). For instance, in the matrix $S_{z+}$, element $s_{AG}$ is 0, because bolt A can be removed along z+ direction by unscrewing operation although component G has contact relationships with bolt A. In the matrix $S_{z-}$, element $s_{GA}$ is 1, it is because component G cannot be removed along z- direction when bolt A exists.

$$S_d = \begin{array}{c} \\ C_1 \\ C_2 \\ \dots \\ C_n \end{array} \begin{array}{c} \begin{array}{cccc} C_1 & C_2 & \dots & C_n \end{array} \\ \begin{bmatrix} s_{1,1} & s_{1,2} & \dots & s_{1,n} \\ s_{2,1} & s_{2,2} & \dots & s_{2,n} \\ \dots & \dots & \dots & \dots \\ s_{n,1} & s_{n,2} & \dots & s_{n,n} \end{bmatrix} \end{array} \quad d = \begin{array}{l} x+ \text{ or } x- \\ \text{or } y+ \text{ or } y- \\ \text{or } z+ \text{ or } z- \end{array} \tag{1}$$

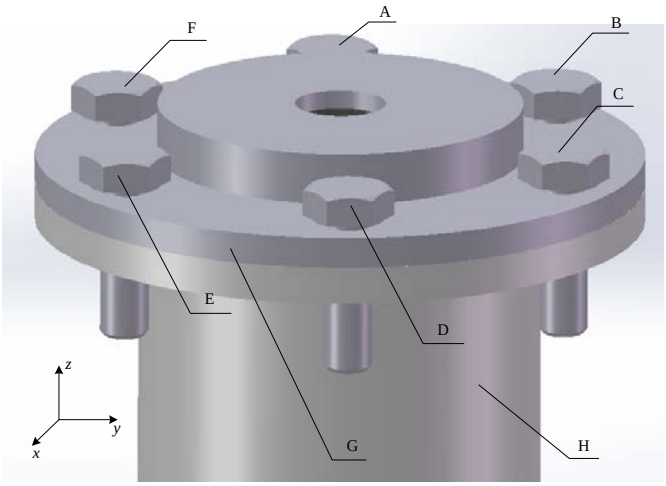



**Fig. 1** An example of SIMM

$$S_{x+} = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \end{array} \begin{array}{c} \begin{array}{cccccccc} A & B & C & D & E & F & G & H \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \end{bmatrix} \end{array} \quad S_{x-} = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \end{array} \begin{array}{c} \begin{array}{cccccccc} A & B & C & D & E & F & G & H \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \end{bmatrix} \end{array}$$

$$
S_{y+} = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \end{array} \begin{array}{c} \begin{array}{cccccccc} A & B & C & D & E & F & G & H \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \end{bmatrix} \end{array} \quad S_{y-} = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \end{array} \begin{array}{c} \begin{array}{cccccccc} A & B & C & D & E & F & G & H \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \end{bmatrix} \end{array} \tag{2}
$$

$$
S_{z+} = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \end{array} \begin{array}{c} \begin{array}{cccccccc} A & B & C & D & E & F & G & H \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \end{bmatrix} \end{array} \quad S_{z-} = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \end{array} \begin{array}{c} \begin{array}{cccccccc} A & B & C & D & E & F & G & H \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \end{bmatrix} \end{array}
$$

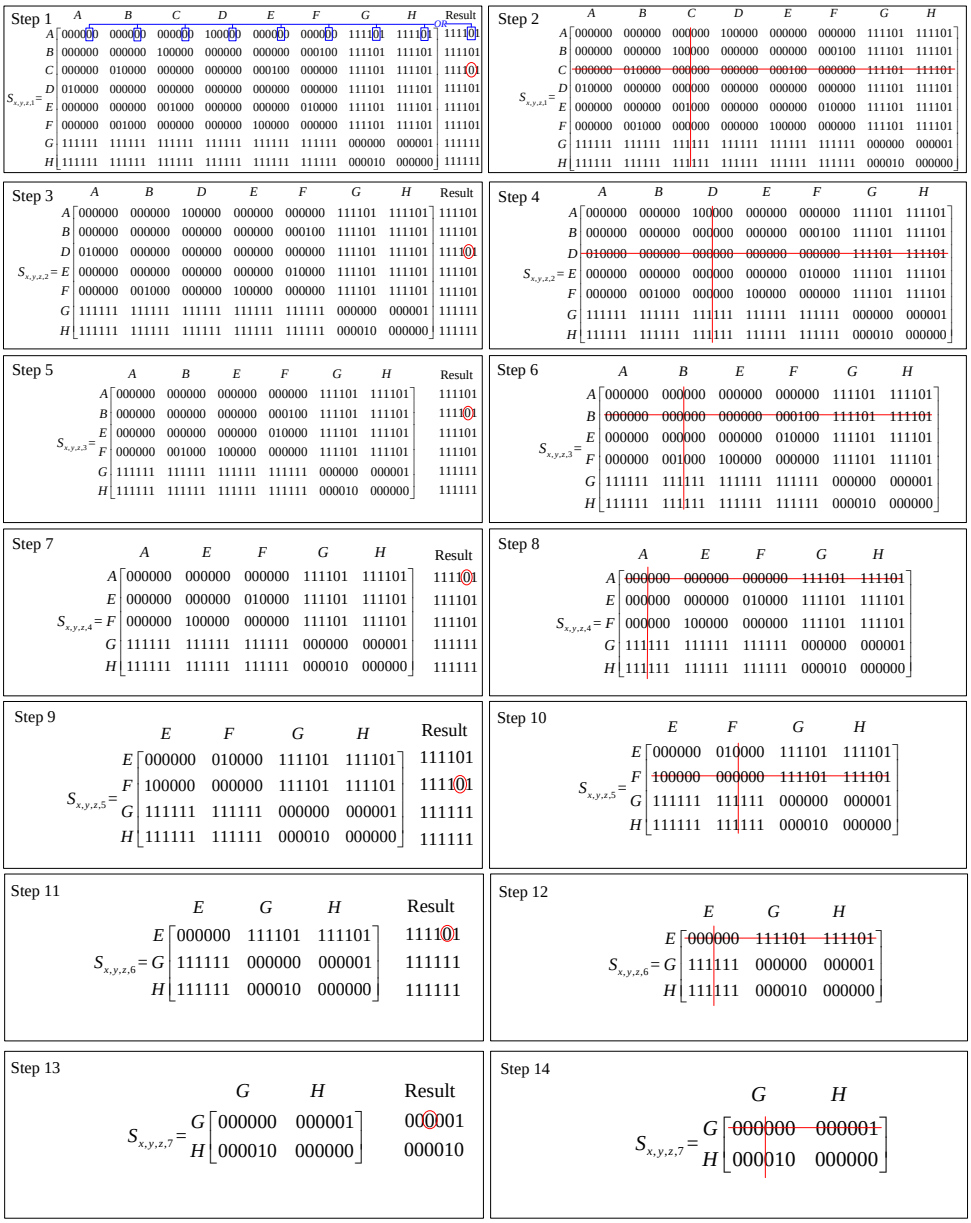



**Fig. 2** A feasible disassembly sequence obtained by SIMM

### 3.2 Interference matrix analysis

Feasible disassembly sequence and disassembly direction are generated by space interference matrices and interference matrix analysis method. Based on the example used in Section 3.1, six space interference matrices are integrated to be matrix $S_{x,y,z,1}$ as shown in Figure 2. The column *Result* is obtained by step1 in Figure 2. The first element of column *Result* is 111101, the fifth bit '0' is obtained by 'OR' operator acted on the fifth bit of all the elements in the first row. Besides, bit '0' means corresponding part can be removed from given direction (the first bit for x+ direction, the second bit for

x- direction, the third bit for y+ direction, *etc*.). Otherwise, it cannot. For instance, the first element of column *Result* is 111101, the fifth bit is '0' while the other bits are '1'. It means bolt A can only be removed along z+ direction. Thus, in step 1, it is obvious that all the bolts A, B, C, D, E and F can be removed along z+ direction. If bolt C has been removed along z+ direction (the red circle in step 1), the corresponding row and column are deleted as shown in step 2. Under this condition, a new matrix $S_{x,y,z,2}$ is obtained as shown in step 3. It is also obvious that all the components A, B, D, E and F can be removed along z+ direction. If bolt D has been removed along z+ direction, the corresponding row and column are deleted as shown in step 4. This procedure continues in the same manner as shown in steps 5 ~ 12. After bolts A ~ F have been removed, matrix $S_{x,y,z,7}$ is obtained in step 13. From the column of *Result* in step 13, it is obvious component G can be removed along x+, x-, y+, y- or z+ direction and component H can be removed along x+, x-, y+, y- or z- direction. If the component G has been removed along y+ direction, component H can be removed along any direction of x+, x-, y+, y-, z+ or z- (direction z+ is chosen here). Thus, feasible disassembly sequence (C/D/B/A/F/E/G/H) is obtained and the corresponding disassembly direction is z+/z+/z+/z+/z+/z+/y+/z+.

## 4. Multi-objective robotic disassembly line balancing problem

Robotic disassembly line is the flow-oriented production system which consists of several robotic workstations. The disassembly tasks are consecutively launched down the line and EoLP is sequentially moved from one workstation to another. For each robotic workstation, disassembly operations are repeatedly performed within cycle time of robotic disassembly line. The decision problem of optimally assigning the disassembly tasks to different robotic workstations in a balanced way is defined as RDLBP.

### 4.1 Assumptions

In this paper, the following assumptions are made. Only single type of EoLP is disassembled on the robotic disassembly line. The structure and geometric information of EoLP are provided in advance so that industrial robot's end-effector can move along the predefined path. Each disassembly task is assigned to only one robotic workstation. The basic disassembly time, tool-change time and direction-change time are deterministic and constant [33]. All the EoLPs are used for complete disassembly. The total working time of robotic workstation should not exceed cycle time (CT) of robotic disassembly line.

### 4.2 Multi-objective formulation

The traditional DLBP only considers basic disassembly time of parts, tool-change time and direction-change time between different parts. The moving time between different parts is always ignored. When RDLBP is considered, the traditional method for solving DLBP is not applicable. The differences between DLBP and RDLBP are described in Figure 3.

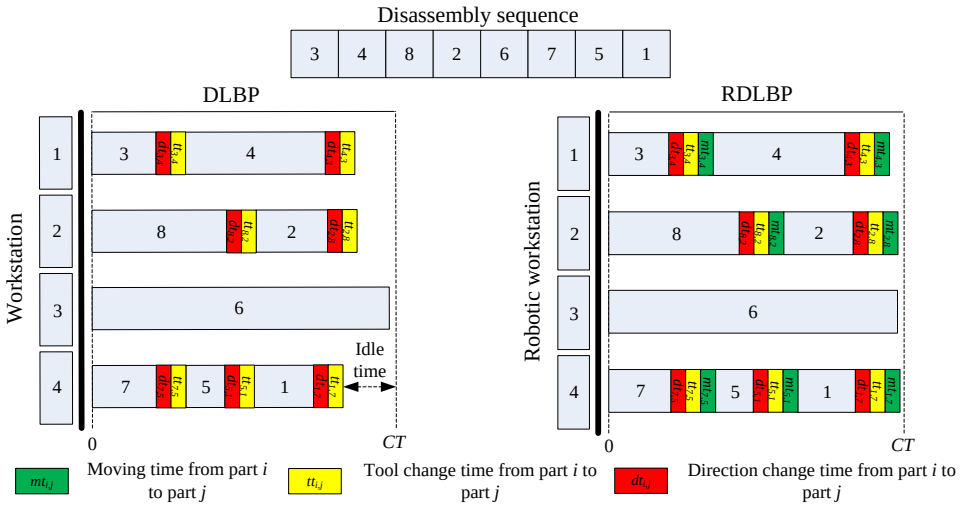



**Fig. 3** The differences between DLBP and RDLBP

In this paper, the multi-objectives of RDLBP are to simultaneously minimize the number of robotic workstations ($f_1$), balance the workload of robotic workstations ($f_2$) and disassemble high demand parts as early as possible ($f_3$) as shown in Equations (3) ~ (5). Minimizing the number of robotic workstation helps to reduce the cost for enterprises. The second objective is used to ensure that idle time of robotic workstations is similar. It helps to level the workload of robotic workstations. Lower $f_2$ value means more similar idle time of robotic workstations. For the third objective, it means high demand parts should be disassembled as early as possible to reduce damage of high demand parts in the disassembly process. In this paper, lower $f_3$ value is more desirable [34].

$$\min f_1 = m \tag{3}$$

$$\min f_2 = \sum_{i=1}^{m} (CT - t_{i,total})^2 \tag{4}$$

$$\min f_3 = \sum_{i=1}^{n} i * h_{p_i} \tag{5}$$

Subject to:

$$Seq = (p_1, p_2, ..., p_n) \in FSeq \tag{6}$$

$$\left\lceil \frac{\sum_{i=1}^{n} t_{p_i}}{CT} \right\rceil \le m \le n \tag{7}$$

$$t_{i,total} \le CT, \quad i \in Z, \quad \forall i \in [0, m] \tag{8}$$

component $p_j$ is only assigned to $RobWork_i$, $\forall i \in [0, m]$, $\forall j \in [0, n]$ (9)

In Equations (3) ~ (9), $m$, $n$, $t_{i,total}$ and $h_{pj}$ respectively represent the number of robotic workstation, the number of parts in EoLP, the total working time of $i^{th}$ robotic workstation and the demand value of part $p_j$. $CT$ is the cycle time of robotic disassembly line which is predefined in this paper. $FSeq$ means the feasible disassembly sequences generated by SIMM. Equation (6) ensures that the disassembly sequence is feasible. Equation (7) guarantees the number of workstation is limited within permitted values. Equation (8) ensures the total working time of any robotic workstation should not exceed cycle time of robotic disassembly line and Equation (9) means any disassembly task should be assigned to only one robotic workstation.

For the moving time between different parts, strictly, it is influenced by not only the position of EoLP in the robot workspace, but also the type of industrial robot and the complexity of EoLP. In this paper, for simplicity, the moving time ($mt_{i,j}$) is calculated by the length of moving path between part $i$ and part $j$ ($md_{i,j}$) and the moving speed ($ms$) of industrial robot's end-effector which is assumed to be constant. Besides, safe distance between the contour of EoLP and moving path of industrial robot's end-effector is also considered to protect industrial robot's end-effector from collision as shown in Figure 4. For the disassembly sequence '3-4-8-2-6-7-5-1', to calculate the moving time between different parts, the moving distance matrix ($MD$) is described by Equation (10). The moving distance between part 3 and part 4 ($md_{3,4}$) and the moving distance between part 3 and part 8 ($md_{3,8}$) are manually calculated by Equation (11). Hence, the corresponding moving time ($mt_{3,4}$ and $mt_{3,8}$) is calculated by the moving distance ($md_{3,4}$ and $md_{3,8}$) and the moving speed ($ms$).

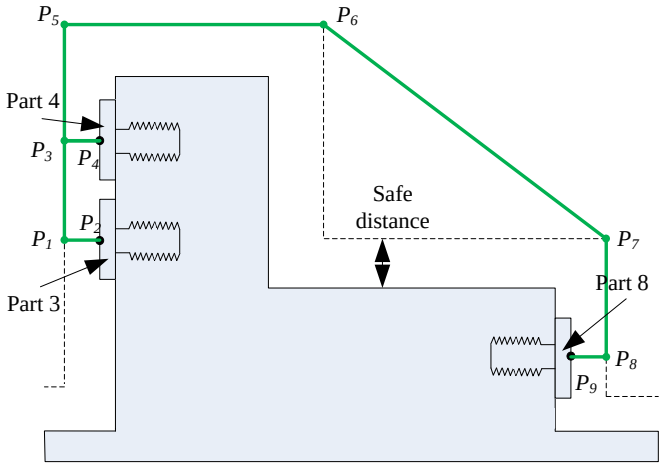



**Fig. 4** The moving path between different parts

$$MD = \begin{bmatrix} 0 & md_{1,2} & md_{1,3} & md_{1,4} & md_{1,5} & md_{1,6} & md_{1,7} & md_{1,8} \\ md_{2,1} & 0 & md_{2,3} & md_{2,4} & md_{2,5} & md_{2,6} & md_{2,7} & md_{2,8} \\ md_{3,1} & md_{3,2} & 0 & md_{3,4} & md_{3,5} & md_{3,6} & md_{3,7} & md_{3,8} \\ md_{4,1} & md_{4,2} & md_{4,3} & 0 & md_{4,5} & md_{4,6} & md_{4,7} & md_{4,8} \\ md_{5,1} & md_{5,2} & md_{5,3} & md_{5,4} & 0 & md_{5,6} & md_{5,7} & md_{5,8} \\ md_{6,1} & md_{6,2} & md_{6,3} & md_{6,4} & md_{6,5} & 0 & md_{6,7} & md_{6,8} \\ md_{7,1} & md_{7,2} & md_{7,3} & md_{7,4} & md_{7,5} & md_{7,6} & 0 & md_{7,8} \\ md_{8,1} & md_{8,2} & md_{8,3} & md_{8,4} & md_{8,5} & md_{8,6} & md_{8,7} & 0 \end{bmatrix} \tag{10}$$

$$md_{3,4} = P_1P_2 + P_1P_3 + P_3P_4$$
$$md_{3,8} = P_1P_2 + P_1P_5 + P_5P_6 + P_6P_7 + P_7P_8 + P_8P_9 \quad (11)$$

Different disassembly operations need different disassembly tools as shown in Figure 5. To finish different disassembly operations, industrial robot needs additional disassembly time to change disassembly tools. This additional time is described by Equation (12) ('*Sp*', '*Sc*', '*Gr*', '*Pl*', '*Ha*' and '*EC*' respectively mean spanners, screwdrivers, grippers, pliers, hammers and electrical cutting). After the disassembly operations have been determined, different disassembly tools should also be considered to handle different sizes of components. For example, to disassemble different sizes of bolts (M1, M2, M3, M4, *etc.*), different types of spanners should be considered. This also causes additional disassembly time for industrial robot to change disassembly tools as shown in Equation (13).

Similarly, for each robotic workstation, the direction-change time between different parts is a part of the total working time. It is described by Equation (14).

$$TT = \begin{array}{c} \\ Sp \\ Sc \\ Gr \\ Pl \\ Ha \\ EC \end{array} \begin{array}{c} \begin{array}{cccccc} Sp & Sc & Gr & Pl & Ha & EC \end{array} \\ \begin{bmatrix} TT_1 & tt_{1,2} & tt_{1,3} & tt_{1,4} & tt_{1,5} & tt_{1,6} \\ tt_{2,1} & TT_2 & tt_{2,3} & tt_{2,4} & tt_{2,5} & tt_{2,6} \\ tt_{3,1} & tt_{3,2} & TT_3 & tt_{3,4} & tt_{3,5} & tt_{3,6} \\ tt_{4,1} & tt_{4,2} & tt_{4,3} & TT_4 & tt_{4,5} & tt_{4,6} \\ tt_{5,1} & tt_{5,2} & tt_{5,3} & tt_{5,4} & TT_5 & tt_{5,6} \\ tt_{6,1} & tt_{6,2} & tt_{6,3} & tt_{6,4} & tt_{6,5} & TT_6 \end{bmatrix} \end{array} \quad (12)$$

$$TT_1 = \begin{array}{c} \\ M1 \\ M2 \\ M3 \\ \dots \\ Mn \end{array} \begin{array}{c} \begin{array}{ccccc} M1 & M2 & M3 & \dots & Mn \end{array} \\ \begin{bmatrix} 0 & tta_{1,2} & tta_{1,3} & \dots & tta_{1,n} \\ tta_{2,1} & 0 & tta_{2,3} & \dots & tta_{2,n} \\ tta_{3,1} & tta_{3,2} & 0 & \dots & tta_{3,n} \\ \dots & \dots & \dots & \dots & \dots \\ tta_{n,1} & tta_{n,2} & tta_{n,3} & \dots & 0 \end{bmatrix} \end{array} \quad (13)$$

$$dt(p_i, p_{i+1}) = \begin{cases} 0 & \text{direction is not changed} \\ 1 & \text{direction is changed by } 90^\circ \\ 2 & \text{direction is changed by } 180^\circ \end{cases} \quad (14)$$

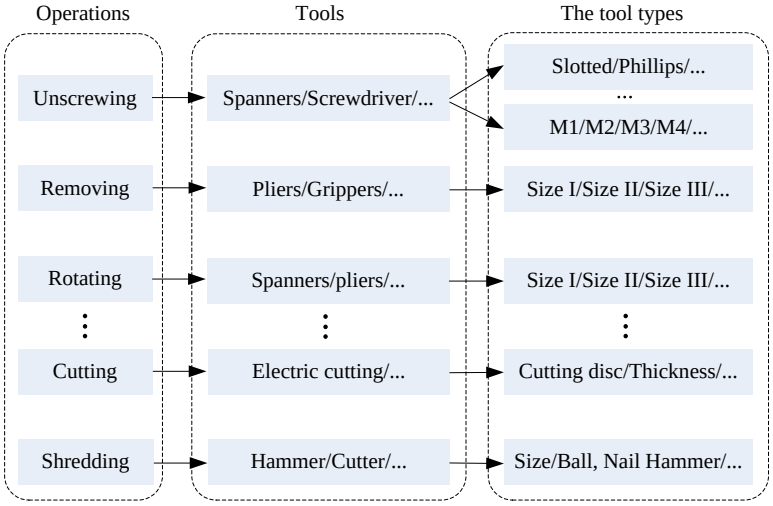



**Fig. 5** Disassembly operations and corresponding disassembly tools

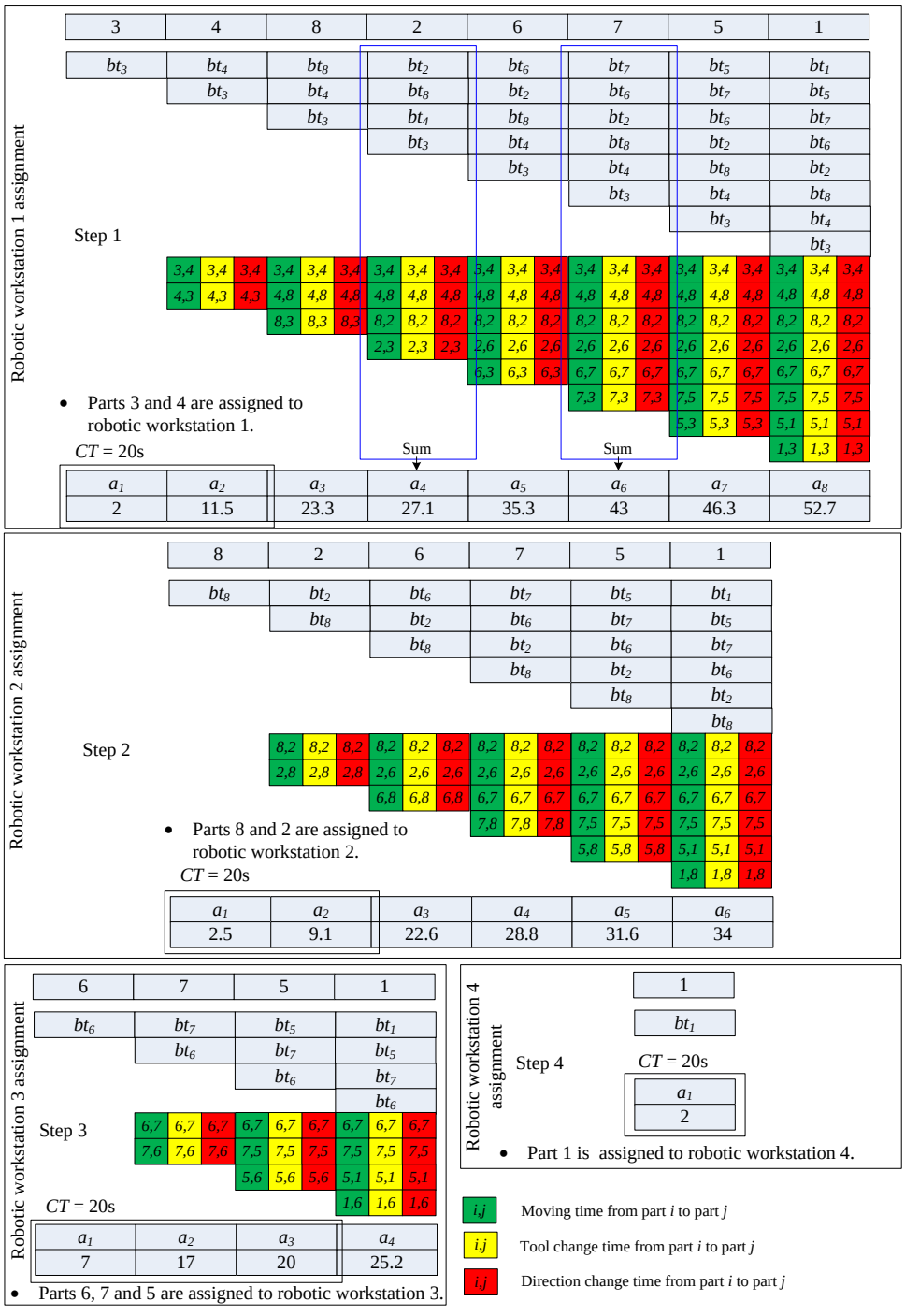



**Fig. 6** The robotic workstation assignment method for RDLBP

**Table 1** An example used to calculate the multi-objective values

| Disassembly sequence | 3 | 4 | 8 | 2 | 6 | 7 | 5 | 1 |
|---|---|---|---|---|---|---|---|---|

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Basic disassembly time | 2 | 2.5 | 2.5 | 2 | 7 | 2 | 1.5 | 2 |
| Disassembly direction | x+ | y+ | y- | y- | x+ | z+ | x+ | x+ |
| Disassembly tool | *Sp*1 | *Sp*2 | *Gr*1 | *Gr*1 | *Sp*1 | *Sp*2 | *Sp*2 | *Gr*1 |
| $h_{pi}$ | 3 | 3 | 2 | 1 | 4 | 3 | 3 | 1 |
| Moving speed | 10 *cm/s* | | | | | | | |

$$TT = \begin{array}{c} \\ Sp1 \\ Sp2 \\ Gr1 \\ Gr2 \end{array} \begin{array}{c} \begin{array}{cccc} Sp1 & Sp2 & Gr1 & Gr2 \end{array} \\ \begin{bmatrix} 0 & 1 & 2 & 2 \\ 1 & 0 & 2 & 2 \\ 2 & 2 & 0 & 1 \\ 2 & 2 & 1 & 0 \end{bmatrix} \end{array} \tag{15}$$

$$MD = \begin{bmatrix} 0 & 14 & 21 & 18 & 12 & 15 & 17 & 22 \\ 14 & 0 & 25 & 20 & 15 & 13 & 19 & 23 \\ 21 & 25 & 0 & 15 & 19 & 24 & 21 & 30 \\ 18 & 20 & 15 & 0 & 20 & 30 & 35 & 28 \\ 12 & 15 & 19 & 20 & 0 & 25 & 10 & 10 \\ 15 & 13 & 24 & 30 & 25 & 0 & 20 & 15 \\ 17 & 19 & 21 & 35 & 10 & 20 & 0 & 17 \\ 22 & 23 & 30 & 28 & 10 & 15 & 17 & 0 \end{bmatrix} \tag{16}$$

An example is used here to calculate multi-objective values of a robotic disassembly line solution. For the disassembly sequence '3-4-8-2-6-7-5-1' as shown in table 1, the cycle time of robotic disassembly line is 20 s, the direction-change time, tool-change time and length of moving path between different parts are respectively defined by Equation (14), Equation (15) (*Sp1*, *Sp2*, *Gr1*, *Gr2* respectively mean spanner-I, spanner-II, gripper-I and gripper-II) and Equation (16). Firstly, it needs to generate robotic disassembly line solutions based on feasible disassembly sequence. From Figure 6, disassembly tasks are assigned to robotic workstations with the help of an allocation matrix $A = [a_i]$. For example, as shown in step 1 of Figure 6, $a_4$ and $a_6$ are respectively calculated by Equation (17) and Equation (18). After the allocation matrix $A$ is obtained in step 1, it is obvious that $a_2$ is less than the cycle time (20s) and $a_3$ is greater than the cycle time. Because the total working time of robotic workstation must not exceed the cycle time, only $a_1$ and $a_2$ should be selected. It means parts 3 and 4 should be assigned to the robotic workstation 1. After that, parts 3 and 4 are deleted in the disassembly sequence '3-4-8-2-6-7-5-1'. In step 2, the allocation matrix $A$ = [2.5, 9.1, 22.6, 28.8, 31.6, 34] is obtained in the same manner and only the former two elements are less than the cycle time. It means parts 8 and 2 should be assigned to robotic workstation 2. Similarly, in steps 3 and 4, parts 6, 7 and 5 are assigned to robotic workstation 3 and part 1 is assigned to robotic workstation 4. Thus, the robotic disassembly line solution is obtained as shown in Figure 7. The multi-objective values ($f_1$, $f_2$ and $f_3$) are calculated by Equations (19) ~ (21).

$$\begin{aligned} a_4 &= bt_3 + bt_4 + bt_8 + bt_2 + mt_{3,4} + tt_{3,4} + dt_{3,4} + mt_{4,8} + tt_{4,8} + dt_{4,8} + \\ &\quad mt_{8,2} + tt_{8,2} + dt_{8,2} + mt_{2,3} + tt_{2,3} + dt_{2,3} \\ &= 2 + 2.5 + 2.5 + 2 + 15/10 + 1 + 1 + 28/10 + 2 + 2 + 23/10 + 0 + 0 + \\ &\quad 25/10 + 2 + 1 = 27.1 \end{aligned} \tag{17}$$

$$\begin{aligned} a_6 &= bt_3 + bt_4 + bt_8 + bt_2 + bt_6 + bt_7 + mt_{3,4} + tt_{3,4} + dt_{3,4} + mt_{4,8} + tt_{4,8} + dt_{4,8} + \\ &\quad mt_{8,2} + tt_{8,2} + dt_{8,2} + mt_{2,6} + tt_{2,6} + dt_{2,6} + mt_{6,7} + tt_{6,7} + dt_{6,7} + mt_{7,3} + tt_{7,3} + dt_{7,3} \\ &= 2 + 2.5 + 2.5 + 2 + 7 + 2 + 15/10 + 1 + 1 + 28/10 + 2 + 2 + 23/10 + 0 + 0 + \\ &\quad 13/10 + 2 + 1 + 20/10 + 1 + 1 + 21/10 + 1 + 1 = 43 \end{aligned} \tag{18}$$

$$f_1 = 4 \tag{19}$$

$$f_2 = \sum_{i=1}^{m} (CT - t_{i,total})^2 = (20-11.5)^2 + (20-9.1)^2 + (20-20)^2 + (20-2)^2 = 515.06 \quad (20)$$

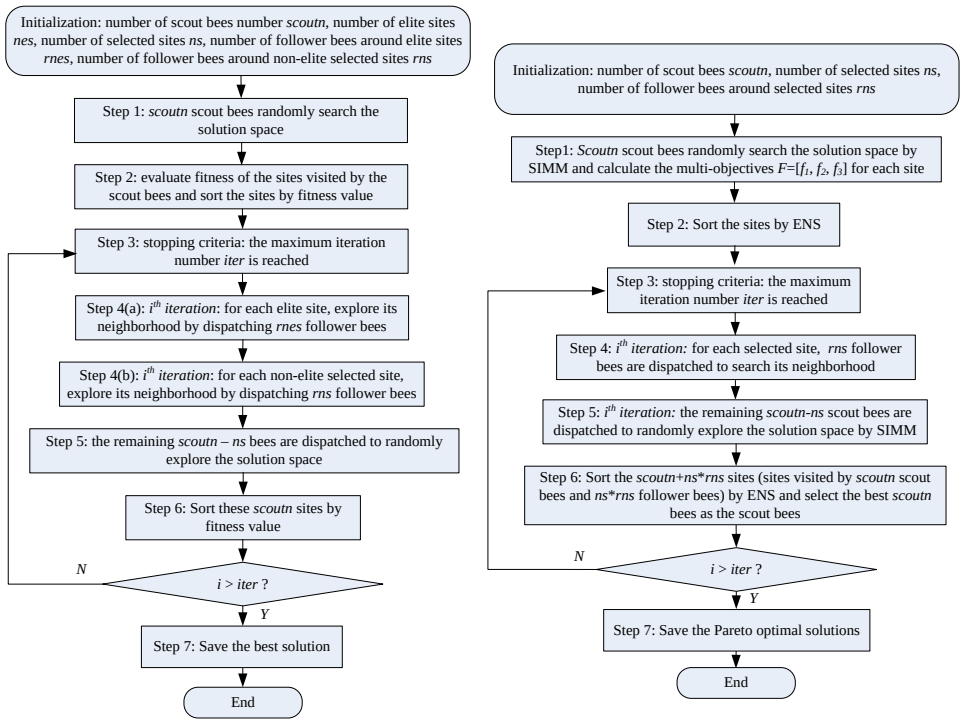
$$f_3 = \sum_{i=1}^{n} i * h_{p_i} = 1\times3 + 2\times3 + 3\times2 + 4\times1 + 5\times4 + 6\times3 + 7\times3 + 8\times1 = 86 \quad (21)$$

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Disassembly sequence | 3 | 4 | 8 | 2 | 6 | 7 | 5 | 1 |
| Robotic workstation assignments | 1 | | 2 | | 3 | | | 4 |
| Working time of robotic workstations | 11.5 | | 9.1 | | 20 | | | 2 |
| $h_{pi}$ | 3 | 3 | 2 | 1 | 4 | 3 | 3 | 1 |

**Fig. 7** The robotic disassembly line solution

## 5. Bees algorithm

### 5.1 The basic BA




(a) Flowchart of basic BA (b) Flowchart of IMODBA

**Fig. 8** Flowchart of basic BA and IMODBA

BA is inspired by the foraging behavior of honey bees [52]. For the basic BA, several parameters should be initialized, namely: number of scout bees (*scoutn*), number of elite sites (*nes*), number of selected sites(*ns*), number of follower bees around elite sites (*rnes*), number of follower bees (*rns*) around non-elite selected sites (*ns* - *nes*) and the stopping criteria. As shown in Figure 8(a), *scoutn* scout bees randomly search the solution space (step 1), fitness of the sites visited by scout bees is evaluated and these sites are sorted by fitness value in step 2. After that, the stopping criteria (whether the maximum iteration is reached) is determined in step 3. The best *nes* sites and *ns* sites visited by scout bees are

respectively selected as ‘elite sites’ and ‘selected sites’. For each elite site (*nes*), *rnes* follower bees are dispatched to search its neighborhood. The scout bee will be replaced by the follower bee only if the quality of the site visited by this follower bee is better than the scout bee (step 4(a)). For each non-elite selected site (*ns* - *nes*), *rns* follower bees are dispatched to search its neighborhood. The scout bee will also be replaced by the follower bee only if the quality of the site visited by this follower bee is better than the scout bee (step 4(b)). The remaining *scoutn* – *ns* bees are dispatched to randomly explore the solution space (step 5). Then, *scoutn* scout bees are sorted by fitness value of the sites and the next iteration starts (step 6). The iteration process continues until the maximum iteration is reached. Finally, the best solution is saved in step 7.

### 5.2 IMODBA

The flowchart of IMODBA is described in Figure 8(b). The number of scout bees *scoutn*, number of selected sites *ns* and number of follower bees around selected sites *rns* are initialized. The disassembly sequence and disassembly direction of scout bees are generated by SIMM and the multi-objective values of the sites visited by scout bees are calculated by the method mentioned in Section 4 (step 1). Then, in step 2, these sites are sorted by efficient non-dominated Pareto sorting method (ENS) [53]. The sites which have the same front rankings are sorted by crowding-distance. When the maximum iteration *iter* is reached, this procedure stops (step 3). After that, each site visited by scout bee is assigned front number $Fr = [Fr_1, Fr_2, Fr_3, \dots, Fr_n]$ and crowding distance. The best *ns* sites are selected as the selected sites. For each selected site, *rns* follower bees are dispatched to search the neighborhood (step 4). The remaining *scout-ns* scout bees randomly search the solution space by SIMM (step 5). After that, *scoutn+ns*rns* sites visited by scout bees and follower bees are sorted and the best *scoutn* bees are selected as the scout bees for the next iteration (step 6). The iteration process continues until the maximum iteration is reached. Finally, the Pareto optimal solutions of RDLBP are obtained (step 7).

#### 5.2.1 Representation of Bees

A bee is represented in Figure 9. The feasible disassembly sequence and disassembly direction of bees are generated by SIMM used in Section 3. The robotic workstation array is obtained by the robotic workstation assignment method. After that, multi-objective values of the sites visited by scout bees are calculated by the method mentioned in Section 4.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Disassembly sequence | 3 | 4 | 8 | 2 | 6 | 7 | 5 | 1 |
| Disassembly direction | x+ | y+ | y- | y- | x+ | z+ | x+ | x+ |
| Robotic workstations | 1 | 1 | 2 | 2 | 3 | 3 | 3 | 4 |
| Disassembly tool | *Sp*1 | *Sp*2 | *Gr*1 | *Gr*1 | *Sp*1 | *Sp*2 | *Sp*2 | *Gr*1 |
| $h_{pi}$ | 3 | 3 | 2 | 1 | 4 | 3 | 3 | 1 |
| Multi-objective values | 4 | 515.06 | 86 | | | | | |

**Fig. 9** The representation of bees

5.2.2 Initialization of IMODBA

In the initialization step of IMODBA, number of scout bees *scoutn*, number of selected sites *ns* and number of follower bees around selected sites *rns* are initialized and *scoutn* scout bees are dispatched to randomly search the solution space.

5.2.3 Pareto optimal solution

In this paper, the goal of IMODBA is to minimize three objectives mentioned in Section 4 as shown by Equation (22).

$$min\ F(Seq) = [f_1(Seq), f_2(Seq), f_3(Seq)] \quad Seq \in FSeq \tag{22}$$

Suppose both $Seq_1$ and $Seq_2$ are feasible disassembly sequences, $Seq_2$ is dominated by $Seq_1$ if and only if

$$(f_i(Seq_1) \le f_i(Seq_2)) \;\&\&\; (f_j(Seq_1) < f_j(Seq_2)) \quad \forall i = 1,\ 2,\ 3 \quad \exists j = 1,\ 2,\ 3 \tag{23}$$

If a solution is not dominated by any other solutions, it is Pareto optimal solution. The Pareto optimal set is a solution set which consists of all the Pareto optimal solutions, corresponding function value is the Pareto optimal front.

5.2.4 Pareto sorting

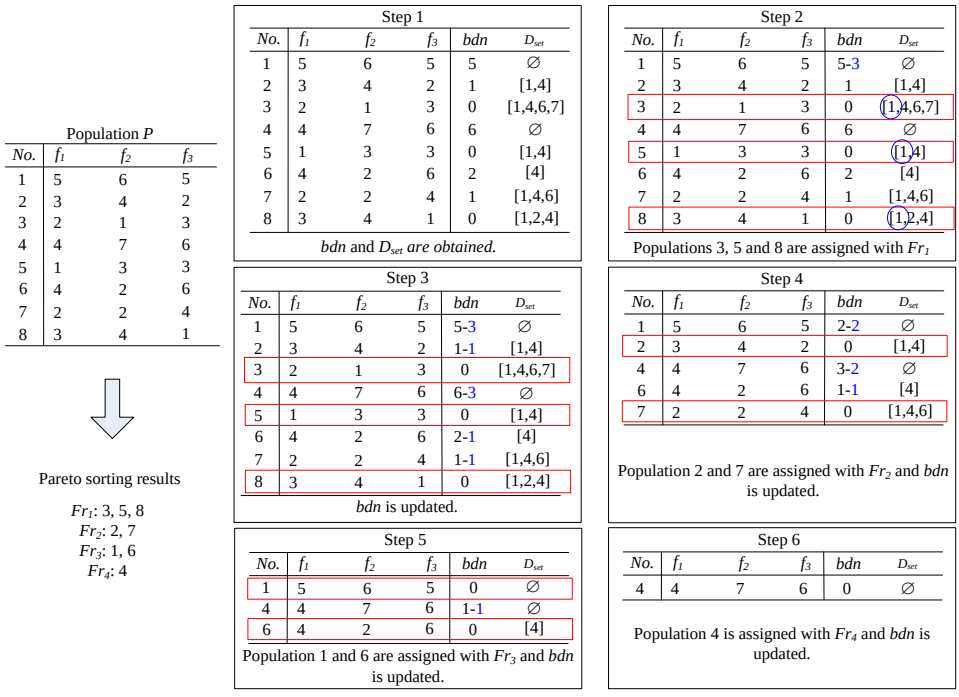



**Fig. 10** An example of traditional Pareto sorting method

The goal of Pareto sorting is to assign front rankings to all the solutions based on their dominance

relationships. Most of the Pareto-based multi-objective optimization methods use the fast non-dominated sorting method [54]. An example is used to describe the process of this method in Figure 10. For any solution in the population *P*, *bdn* represents the number of solutions that dominate this solution while $D_{set}$ is a set of solutions which is dominated by this solution. For example, in step 1 (Figure 10), solution 2 is only dominated by solution 8, thus, *bdn* of solution 2 is 1. Solutions 1 and 4 are dominated by solution 2, thus, $D_{set}$ of solution 2 is [1, 4]. From step 2, it is obvious that solutions 3, 5 and 8 are the non-dominated solutions (*bdn* is 0) and they are assigned front ranking $Fr_1$. After that, solution 3, 5 and 8 are deleted. In step 2, $D_{set}$ of solutions 3, 5 and 8 are respectively [1, 4, 6, 7], [1 4] and [1, 2, 4]. It means solution 1 is dominated by solutions 3, 5 and 8 (3 solutions). Thus, in step 3, after solutions 3, 5 and 8 are deleted, *bdn* of solution 1 should minus 3. It is the same with solutions 2, 4, 6 and 7 in step 3. It is obvious the solutions 2 and 7 are the non-dominated solutions in step 4 and they are assigned front ranking $Fr_2$. After that, solutions 2 and 7 are deleted and *bdn* of the other solutions are updated (the blue font in step 4). This procedure continues in the same manner as shown in steps 5 and 6. Finally, all the solutions are assigned front rankings as shown in Figure 10.

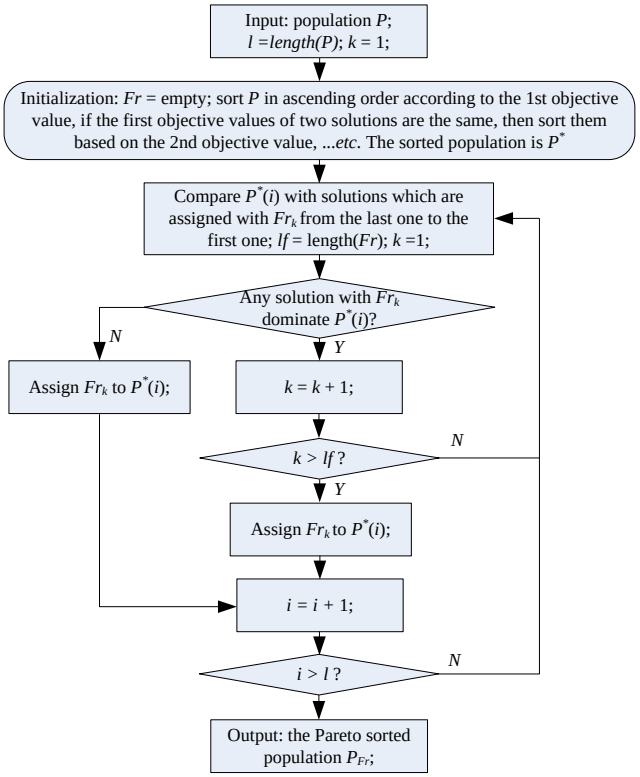



**Fig. 11** The workflow of ENS

However, the traditional Pareto sorting method is poor efficiency [55]. ENS is proposed to improve the efficiency of Pareto sorting method without changing the sorting results [53]. As shown in Figure 11, population *P* is sorted in ascending order according to the first objective value. If the first objective values of two solutions are the same, then the solutions are sorted in ascending order according to the second objective value. If the first and second objective values are still the same, these solutions are sorted in ascending order according to the third objective value. After that, the sorted population $P^*$ is obtained. It is obvious that solution $P^*(1)$ (solution 5) should be assigned front ranking $Fr_1$, because no solution in

population $P^*$ dominates it. For solution $P^*(i)$ ($i$ = 2, 3…), it should be compared with the solutions which have been assigned $Fr_k$ ($k$ = 1) from the last solution to the first solution. If no solution which has been

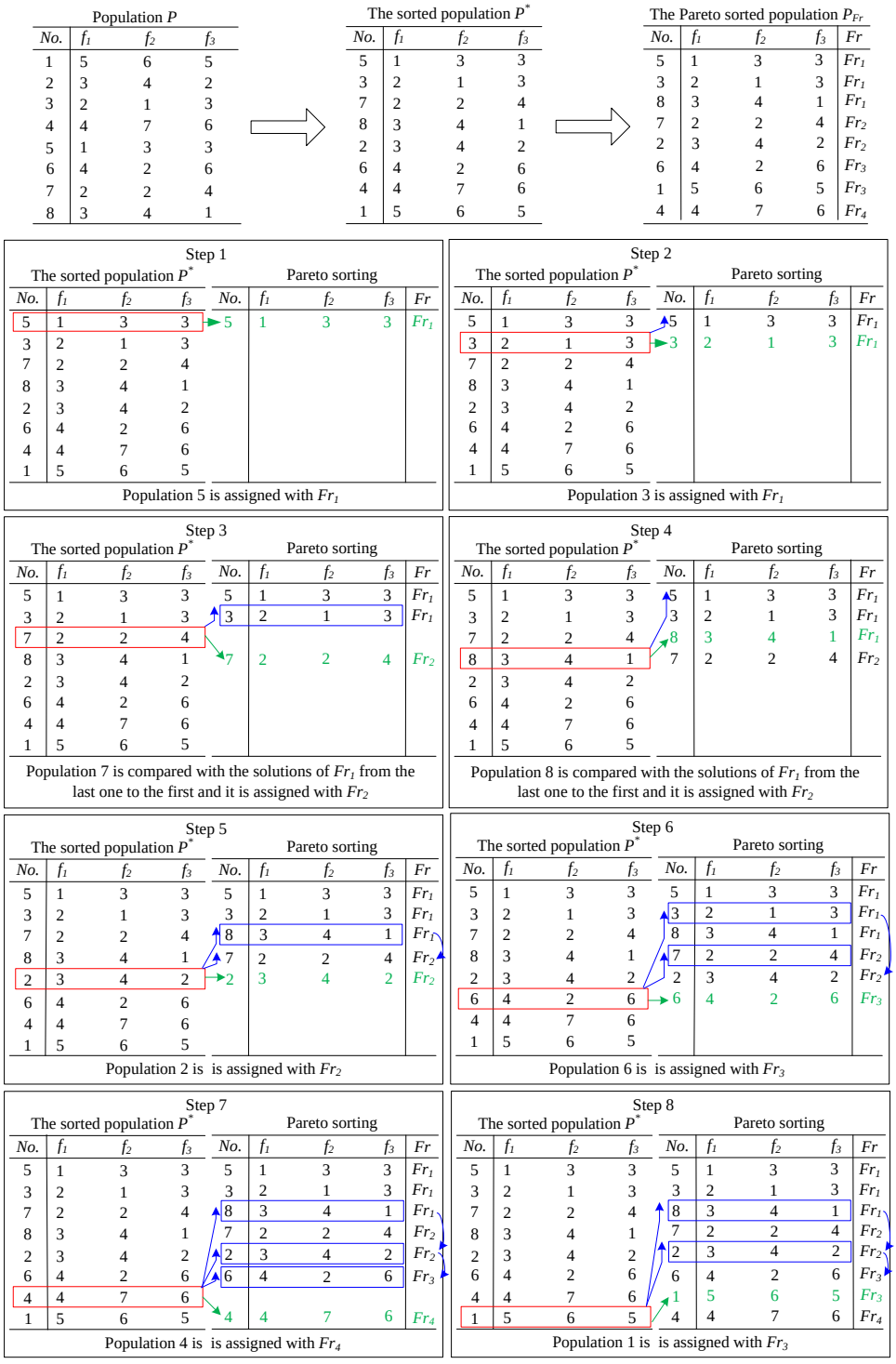
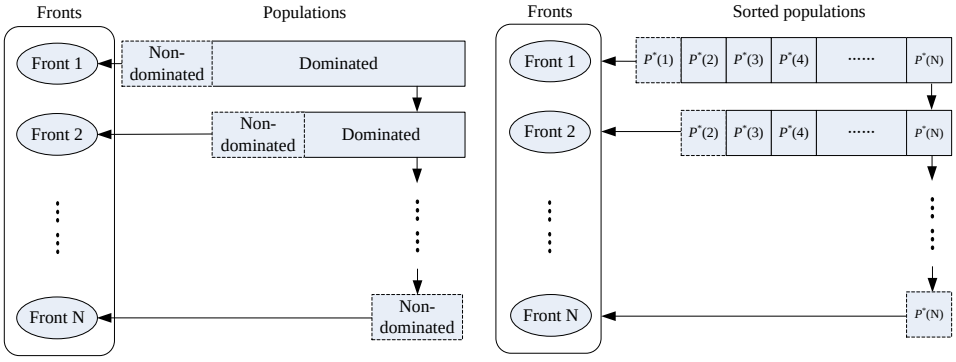



**Fig. 12** An example of ENS method

assigned $Fr_k$ dominates $P^*(i)$, then $P^*(i)$ is assigned $Fr_k$. If any solution which has been assigned $Fr_k$ dominates $P^*(i)$, $k$ increases by 1. If $k$ is not greater than the length of set $Fr$, solution $P^*(i)$ should be compared with the solutions which have been assigned $Fr_k$ from the last solution to the first solution. If $k$ is greater than the length of set $Fr$, it means there is no front ranking $Fr_k$ in set $Fr$. Hence, front ranking $Fr_k$ is added in set $Fr$ and $P^*(i)$ is assigned $Fr_k$. This procedure continues until all the solutions are assigned front rankings. Finally, Pareto sorted population $P_{Fr}$ is obtained.

An example is used here to show the workflow of ENS. As shown in step 1 (Figure 12), based on the same population used in Figure 10, the sorted population $P^*$ is firstly obtained. Obviously, the first solution (solution 5) in the sorted population $P^*$ is assigned $Fr_1$. In step 2, the second solution (solution 3) is compared with solutions which have been assigned $Fr_1$ from the last one to the first one (only solution 5 is assigned $Fr_1$). However, solution 3 is not dominated by solution 5. Thus, solution 3 is assigned $Fr_1$. In step 3, solution 7 is compared with solutions which have been assigned $Fr_1$ from the last one to the first one (in order of solutions 3 and 5). Because solution 3 dominates solution 7 and there is no front ranking $Fr_2$ in set $Fr$. $Fr_2$ is added to set $Fr$ and solution 7 is assigned $Fr_2$. In step 4, solution 8 is compared with solutions which have been assigned $Fr_1$ from the last one to the first one (in order of solutions 3 and 5). There is no solution which has been assigned $Fr_1$ dominates solution 8. Thus, solution 8 is assigned $Fr_1$. In step 5, solution 2 is compared with solutions which have been assigned $Fr_1$ from the last one to the first one (in order of solutions 8, 3 and 5). Because solution 2 is dominated by solution 8, it should be compared with solutions which have been assigned $Fr_2$ from the last one to the first one (only solution 7 is assigned $Fr_2$). It is obvious that solution 2 is not dominated by solution 7. Thus, solution 2 is assigned $Fr_2$. In step 6, solution 6 is compared with solutions which have been assigned $Fr_1$ from the last one to the first one (in order of solutions 8, 3 and 5). It is dominated by solution 3 and then it should be compared with solutions which have been assigned $Fr_2$ from the last one to the first one (in order of solutions 2 and 7). However, solution 6 is dominated by solution 7 and there is no front ranking $Fr_3$ in set $Fr$. Thus, $Fr_3$ is added to set $Fr$ and solution 6 is assigned $Fr_3$. This procedure continues until all the solutions are assigned front rankings as shown in Figure 12.

The differences between the traditional Pareto sorting and ENS are described in Figure 13. It is obvious that the traditional Pareto sorting method assigns the front rankings to several solutions at a time and ENS assigns front rankings to the solutions individually.




**(a)** The traditional Pareto sorting method **(b)** The efficient non-dominated sorting method

**Fig. 13** The differences between traditional Pareto sorting and ENS

5.2.5 Crowding-distance computation

After all the solutions are assigned front rankings, the solutions which have the same front rankings are sorted by crowding-distance [54]. Firstly, it sorts the solutions according to each objective function value in ascending order. For each objective value, the solutions which have the smallest and largest objective values are assigned infinite distance value. For the other intermediate solutions, distance value (the absolute normalized difference of two adjacent solutions' objective value) is assigned. The crowding-distance of all the solutions is calculated by the sum of distance values of each objective. The procedure of crowding-distance computation is described in Figure 14.

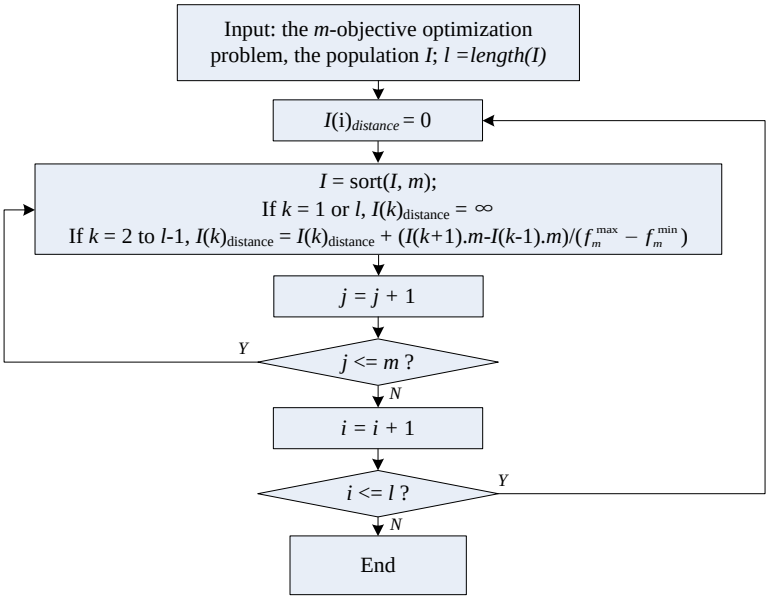



**Fig. 14** The procedure of crowd-distance computation

Based on Pareto sorting and crowding-distance computation, if two solutions have different front rankings, the solution with better front ranking is preferred (such as $Fr_1$ is better than $Fr_2$). If two solutions have the same front rankings, the solution with greater crowding distance is preferred.

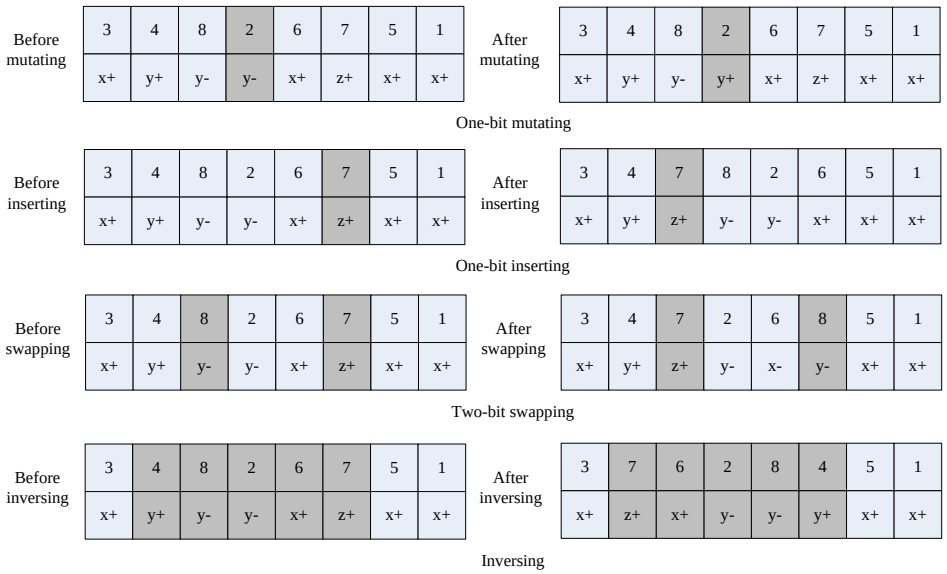



**Fig. 15** The neighborhood search strategy

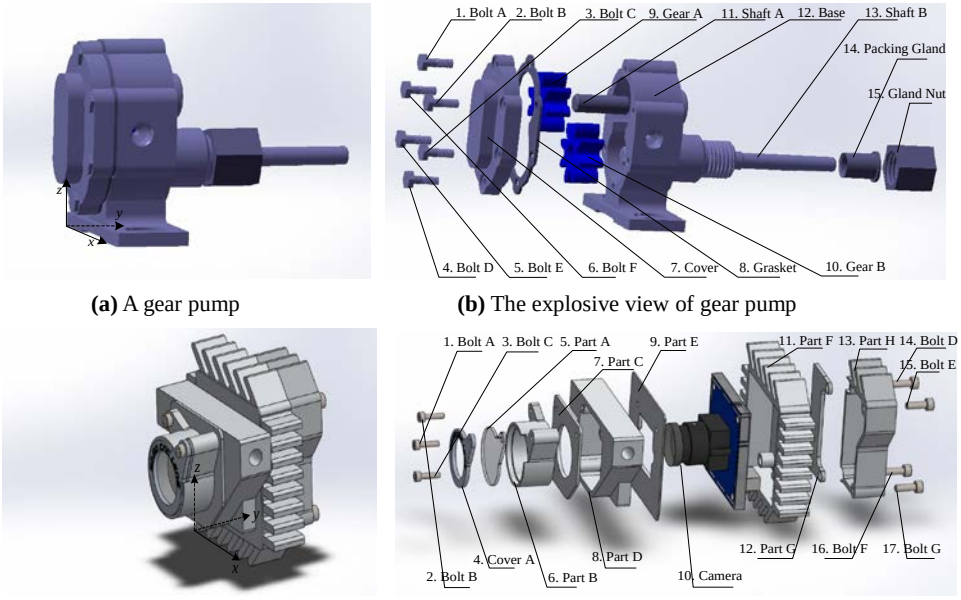
5.2.6 Neighborhood search strategy

Each follower bee randomly searches the neighborhood of the sites visited by scout bees. The neighborhood search strategy which includes one-bit mutating operator, one-bit inserting operator, two-bit swapping operator and inversing operator is used in this paper as shown in Figure 15.

One-bit mutating operator acts on random bit of disassembly direction array and the corresponding direction changes 180 degrees (such as from x+ direction to x- direction). One-bit inserting operator randomly selects one bit of disassembly sequence and disassembly direction arrays and this bit inserts to random position of disassembly sequence and disassembly direction arrays. Two-bit swapping operator is used to exchange random two bits of disassembly sequence and disassembly direction arrays. Inversing operator is used to invert random part of disassembly sequence and disassembly direction arrays. The feasibility of follower bee needs to be checked by SIMM. If the follower bee is unfeasible disassembly sequence, this follower bee continues to search the neighborhood until it is feasible disassembly sequence.

## 6. Experiments and Results

6.1 Case studies




**(a)** A gear pump **(b)** The explosive view of gear pump

**(c)** A camera **(d)** The explosive view of camera

**Fig. 16** Case studies based on a gear pump and a camera

As shown in Figure 16, cases studies based on a gear pump and a camera [56] are used to verify the effectiveness of proposed method. The workflow of proposed method is described in Figure 17. The properties of all the parts of the gear pump and the camera are listed in Table 2.

EoLPs are disassembled by robotic disassembly line as shown in Figure 18. *CT* of this robotic disassembly line is 30 s. The direction-change and tool-change time are respectively described by Equations (14) and (15).

**Table 2** The properties of all the parts of the gear pump and the camera

| EoLP | Parts | Task name | *bt*/s | Disassembly point (*mm*) | Tools | $h_{pi}$ |
|---|---|---|---|---|---|---|
| Gear pump | 1 | Unscrew the Bolt A | 3 | [49.4, -12.6, 105.5] | Spanner-I | 1 |
| | 2 | Unscrew the Bolt B | 3 | [74.4, -12.6, 81] | Spanner-I | 1 |
| | 3 | Unscrew the Bolt C | 3 | [74.4, -12.6, 45] | Spanner-I | 1 |
| | 4 | Unscrew the Bolt D | 3 | [49.4, -12.6, 20.5] | Spanner-I | 1 |
| | 5 | Unscrew the Bolt E | 3 | [24.4, -12.6, 45] | Spanner-I | 1 |
| | 6 | Unscrew the Bolt F | 3 | [24.4, -12.6, 81] | Spanner-I | 1 |
| | 7 | Remove the Cover | 4 | [49.4, -20.6, 63] | Gripper-II | 2 |
| | 8 | Remove the Gasket | 3 | [49.4, 1.4, 105.5] | Gripper-I | 2 |
| | 9 | Remove the Gear A | 6 | [49.4, 3.4, 81] | Gripper-I | 3 |
| | 10 | Remove the Gear B | 6 | [49.4, 3.4, 45] | Gripper-I | 3 |
| | 11 | Remove the Shaft A | 4 | [49.4, -7.6, 81] | Gripper-I | 1 |
| | 12 | Remove the Base | 8 | [49.4, 49.4, 81] | Gripper-II | 4 |

| | | | | | | |
|---|---|---|---|---|---|---|
| | 13 | Remove the Shaft B | 4 | [49.4, 152.4, 45] | Gripper-I | 2 |
| | 14 | Remove the Packing Gland | 2 | [49.4, 91.4, 45] | Gripper-I | 2 |
| | 15 | Unscrew the Gland Nut | 3 | [49.4, 96.4, 45] | Spanner-II | 1 |
| | 1 | Unscrew the Bolt A | 3 | [37.4, -28, 39.9] | Spanner-I | 1 |
| | 2 | Unscrew the Bolt B | 3 | [12.7, -18.5, 39.9] | Spanner-I | 1 |
| | 3 | Unscrew the Bolt C | 3 | [12.7, -18.5, 16] | Spanner-I | 1 |
| | 4 | Remove the Cover A | 2 | [28.1, -26, 29.8] | Gripper-I | 2 |
| | 5 | Remove the Part A | 5 | [30.3, -24.5, 32.4] | Gripper-I | 3 |
| | 6 | Remove the Part B | 4 | [23.1, -25.5, 26.6] | Gripper-II | 3 |
| | 7 | Remove the Part C | 3 | [24.4, -13.5, 28.8] | Gripper-I | 2 |
| | 8 | Remove the Part D | 4 | [25.1, -12.5, 43.4] | Gripper-II | 2 |
| Camera | 9 | Remove the Part E | 3 | [24.5, -0.5, 43.4] | Gripper-I | 2 |
| | 10 | Remove the Camera | 8 | [24.5, -20.1, 27.8] | Gripper-II | 5 |
| | 11 | Remove the Part F | 4 | [23, 0, 56.4] | Gripper-II | 2 |
| | 12 | Remove the Part G | 2 | [10.7, 13.5, 28.1] | Gripper-I | 1 |
| | 13 | Remove the Part H | 3 | [30, 24, 56.4] | Gripper-II | 2 |
| | 14 | Unscrew the Bolt D | 3 | [22.5, 26.5, 45.9] | Spanner-II | 1 |
| | 15 | Unscrew the Bolt E | 3 | [42.6, 26.5, 45.9] | Spanner-II | 1 |
| | 16 | Unscrew the Bolt F | 3 | [21, 26.5, 10] | Spanner-II | 1 |
| | 17 | Unscrew the Bolt G | 3 | [36.3, 26.5, 10] | Spanner-II | 1 |

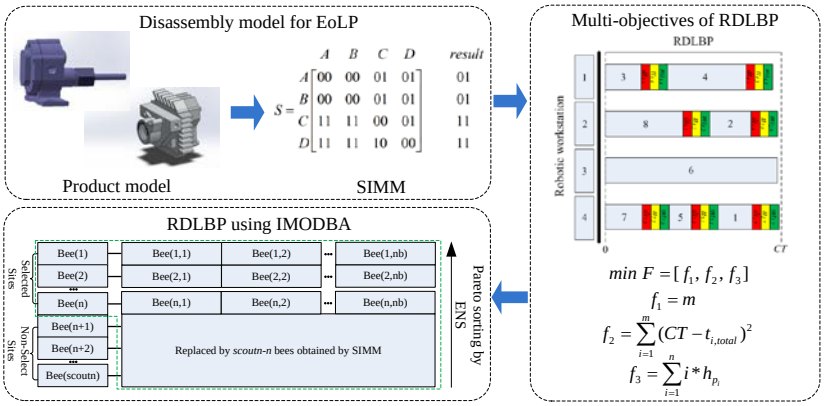



**Fig. 17** The workflow of proposed method

The safe distance between the contour of EoLP and the moving path of industrial robot's end-effector is 10 mm. The moving time matrix ($MT = [mt_{ij}]$; $i, j = 1, 2\ldots 15$) between different parts is calculated by moving distance matrix ($MD = [md_{ij}]$; $i, j = 1, 2\ldots 15$) which is manually calculated and moving speed ($ms$) which is assumed to be 12 cm/s® [57]. For example, as shown in Figure 19, the moving time $mt_{1,13}$ is calculated by Equation (24).

$$\begin{aligned} mt_{1,13} &= md_{1,13} / ms \\ &= (P_1P_2 + P_2P_3 + P_3P_4 + P_4P_5 + P_5P_6 + P_6P_7) / ms \\ &= (10 + 18.92 + 72.01 + 127.67 + 20 + 10) / 12 = 21.55s \end{aligned} \tag{24}$$

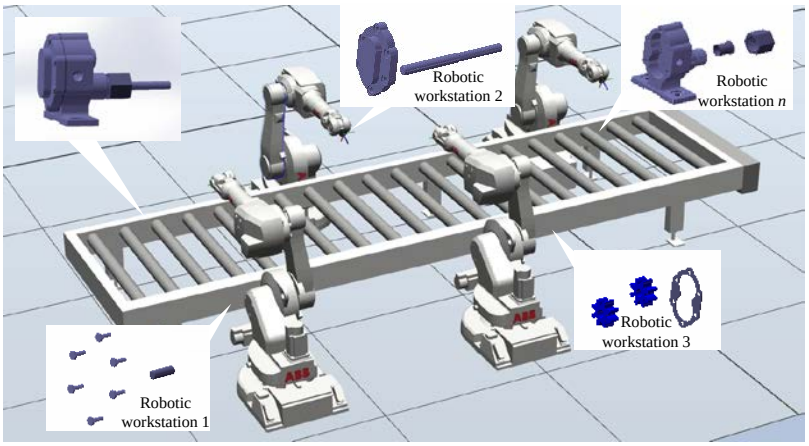



**Fig. 18** The robotic disassembly line

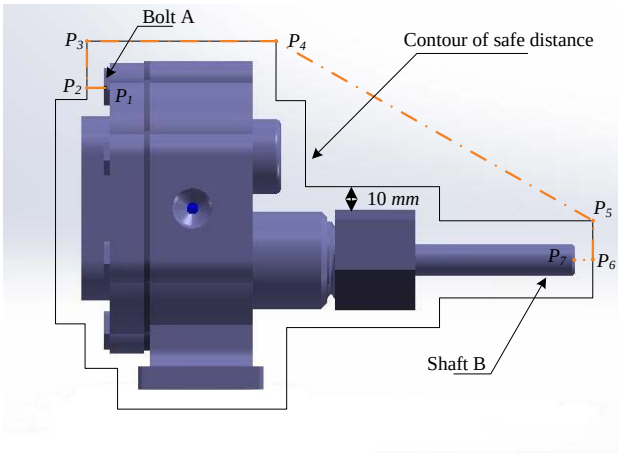



**Fig. 19** The moving path from Bolt A to Shaft B

6.2 Performance analysis

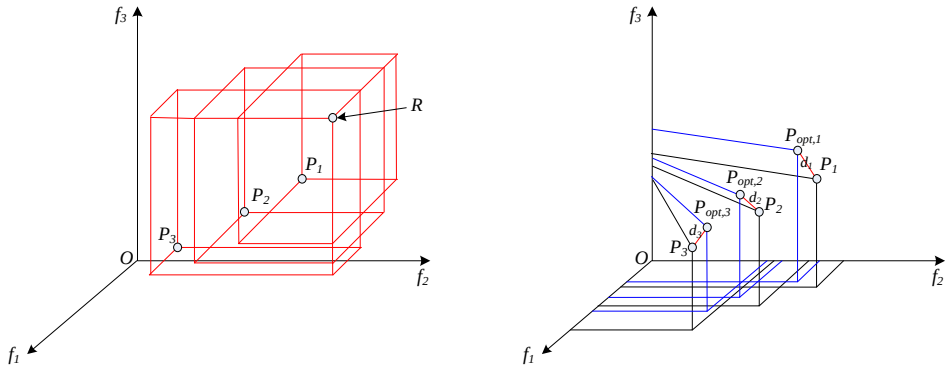
In this section, simulations are taken on personal computer with 2.3GHz Intel core i5-6200U CPU, 4 GB memory using Matlab 2014b® [58]. This section consists of three parts: 1. performance analysis of IMODBA under different iterations and populations; 2. performance comparisons between IMODBA and the other multi-objective optimization algorithms; 3. comparisons of Pareto optimal fronts of three cases.

For single-objective optimization problem, it is easy to evaluate the quality of solutions through fitness value. For multi-objectives optimization algorithm, auxiliary methods need to be used to evaluate the quality of non-dominated solutions [59]. Existing performance indexes (PI) used for evaluating the quality of non-dominated solutions contain distance-based accuracy PI, volume-based accuracy PI, *etc* [60]. In this paper, hypervolume indicator (HI) and generational distance (GD) are used. HI [61] is a widely used volume-based accuracy PI. It can evaluate the convergence and distribution of non-dominated solutions by quantitative comparisons without knowing the real Pareto optimal fronts [62]. As shown in Figure 20(a), HI of non-dominated solutions ($P_1$, $P_2$ and $P_3$ are the obtained non-dominated solutions) is the volume of the areas surrounded by cuboids $P_1R$, $P_2R$ and $P_3R$ ($R$ is the reference point) as

shown in Equation (25). For minimum optimization problem, the non-dominated solutions with greater HI value are preferred. GD [60] is used to represent the distance between the obtained non-dominated solutions ($P_1$, $P_2$ and $P_3$ in Figure 20(b)) and the Pareto optimal solutions ($P_{opt,1}$, $P_{opt,2}$ and $P_{opt,3}$ in Figure 20(b)). It is calculated by Equation (26), where *nd* is the number of obtained non-dominated solutions, $d_i$ (the red dotted line in Figure 20(b)) is the Euclidean distance between $i^{th}$ member of obtained non-dominated solutions and its nearest member of Pareto optimal solutions. The non-dominated solutions with lower GD value are preferred.

$$HI = Volume(cuboid_{P_1R} \cup cuboid_{P_2R} \cup cuboid_{P_3R}) \tag{25}$$

$$GD = (\sum_{i=1}^{nd} d_i^2)^{1/2} / nd \tag{26}$$



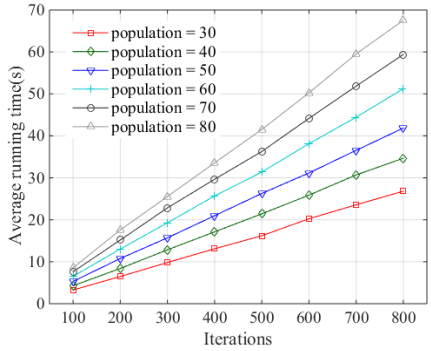

(a) HI of non-dominated solutions (b) GD of non-dominated solutions

**Fig. 20** performance indicators of non-dominated solutions

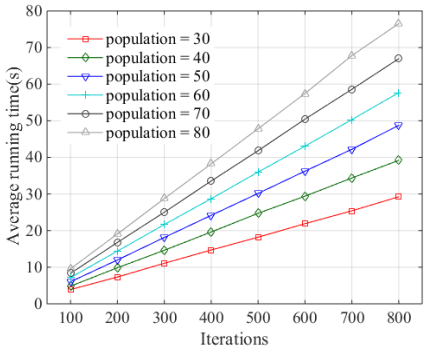
Before calculating HI and GD values of non-dominated solutions, it is important to normalize multi-objective values by Equation (27). For the gear pump, $f_{1,min}$, $f_{1,max}$, $f_{2,min}$, $f_{2,max}$, $f_{3,min}$ and $f_{3,max}$ are respectively 3, 5, 0.0548, 765.6372, 228 and 259. For the camera, $f_{1,min}$, $f_{1,max}$, $f_{2,min}$, $f_{2,max}$, $f_{3,min}$ and $f_{3,max}$ are respectively 3, 4, 1.0411, 858.3914, 268 and 338. For both the gear pump and the camera, the reference point for calculating HI value is [1.2, 1.2, 1.2] and the optimal Pareto solutions for calculating GD value are listed in Tables 7 and 8.

$$f_{i,norm} = (f_i - f_{i,\min}) / (f_{i,\max} - f_{i,\min}) \quad i = 1,2,3 \tag{27}$$

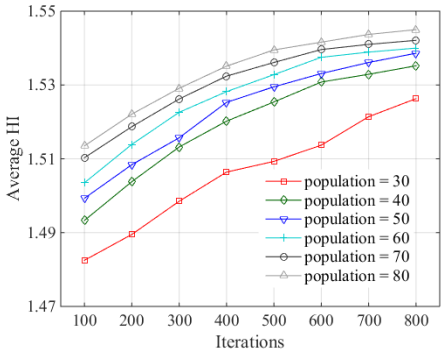
For both the gear pump and the camera, the selected sites number *ns*, follower bees number *rns* of IMODBA are respectively 15 and 1. The average running time, average HI and average GD are compared under different iterations (from 100 to 800) and populations (from 30 to 80). Each simulation is repeated 50 times. As shown in Figure 21, when the iteration number and the population number are respectively 80 and 800, it has the best quality of solutions (for the gear pump, average HI and average GD are respectively 1.5450 and 0.00; for the camera, average HI and average GD are respectively 0.8612 and 0.0025). But it takes the longest running time (for the gear pump, it is 67.55s and for the camera, it is 76.55s). When iteration number and population number are respectively 30 and 100, it takes the shortest running time (for the gear pump, it is 3.26s and for the camera, it is 3.89s) but it has the worst quality of solutions (for the gear pump, average HI and average GD are respectively 1.4825 and 0.0149; for the camera, average HI and average GD are respectively 0.5529 and 0.016).



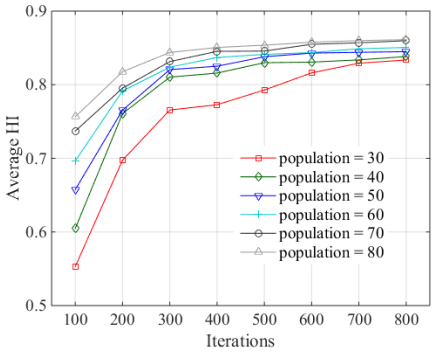

(a) Average running time of IMODBA under different iterations and populations based on the gear pump



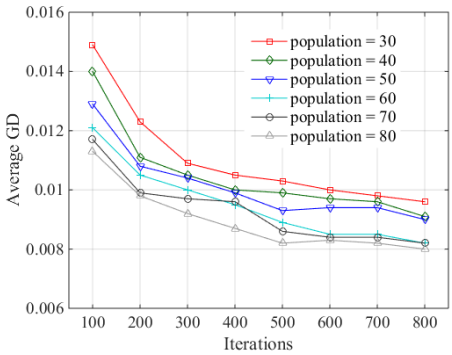

(b) Average running time of IMODBA under different iterations and populations based on the camera



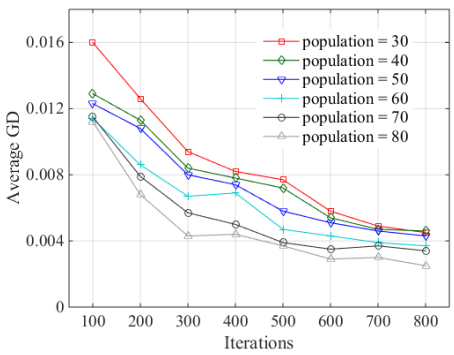

(c) Average HI of IMODBA under different iterations and populations based on the gear pump




(d) Average HI of IMODBA under different iterations and populations based on the camera




(e) Average GD of IMODBA under different iterations and populations based on the gear pump




(f) Average GD of IMODBA under different iterations and populations based on the camera

**Fig. 21** Performance analysis of IMODBA under different iterations and populations

For the performance comparisons between IMODBA and the other multi-objective optimization algorithms, the average running time, average HI and average GD under different iterations and populations are analyzed.

- IMODBA: the proposed method in this paper is used. The selected sites number *ns*, follower bees number *rns* are 15 and 1 respectively.
- MODBA: multi-objective discrete Bees Algorithm (MODBA) uses the traditional Pareto sorting method [54] and the input parameters are the same with IMODBA.

- Multi-objective artificial Bees colony (MOABC): it consists of employed bees, onlooker bees and scout bees. The neighborhood search strategies of MOABC are the same with IMODBA. For onlooker bees, the probability value is calculated by the method used in [63]. When a food source has not been updated over *thred* iterations (*thred* is 10 here), employed bee becomes scout bee to randomly explore new food source.
- Multi-objective genetic algorithm (MOGA): it uses selection procedure (roulette wheel method), the Pareto sorting methods [64], two point crossover procedure [65] and mutation procedure (the same with neighborhood search strategies of IMODBA). The parameter of chromosome selection is 0.1 [64]. The crossover rate and mutation rate are 0.8 and 0.2 respectively.

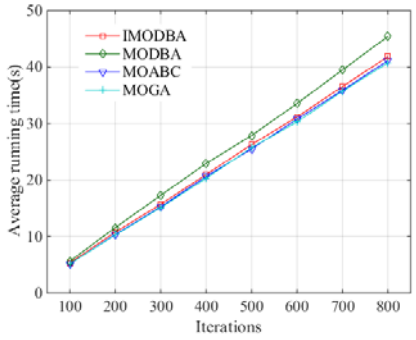
**Table 3** Improvement of running time of IMODBA under different iterations

| EoLP | Iterations | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 |
|---|---|---|---|---|---|---|---|---|---|
| Gear pump | IMODBA | 5.31s | 10.77s | 15.71s | 20.98s | 26.18s | 31.16s | 36.55s | 41.86s |
| | MODBA | 5.63s | 11.57s | 17.29s | 22.93s | 28.19s | 33.54s | 39.50s | 45.42s |
| | Improvement | 0.32s | 0.80s | 1.58s | 1.95s | 2.01s | 2.38s | 2.95s | 3.56s |
| | Percentage | 5.68% | 6.91% | 9.14% | 8.50% | 7.13% | 7.10% | 7.47% | 7.84% |
| Camera | IMODBA | 6.04s | 11.98s | 18.12s | 24.17s | 30.15s | 36.23s | 42.21s | 48.76s |
| | MODBA | 6.68s | 12.94 | 19.22s | 25.67s | 31.99s | 38.36s | 44.86s | 51.28s |
| | Improvement | 0.64s | 0.96s | 1.10s | 1.50s | 1.84s | 2.13s | 2.65s | 2.52s |
| | Percentage | 9.58% | 7.42% | 5.72% | 5.84% | 5.75% | 5.55% | 5.91% | 4.91% |

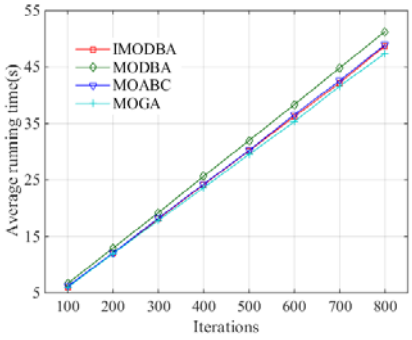
**Table 4** Improvement of running time of IMODBA under different populations

| EoLP | Populations | 30 | 40 | 50 | 60 | 70 | 80 |
|---|---|---|---|---|---|---|---|
| Gear Pump | IMODBA | 16.19s | 21.41s | 26.18s | 31.38s | 36.27s | 40.92s |
| | MODBA | 17.83s | 22.85s | 28.19s | 33.02s | 38.14s | 43.03s |
| | Improvement | 1.64s | 1.44s | 2.01s | 1.64s | 1.87s | 2.11s |
| | Percentage | 9.20% | 6.30% | 7.13% | 4.97% | 4.90% | 4.90% |
| Camera | IMODBA | 18.20s | 24.72s | 30.15s | 35.98s | 41.82s | 47.83s |
| | MODBA | 20.08s | 26.01s | 31.99s | 37.55s | 43.28s | 49.12s |
| | Improvement | 1.88s | 1.29s | 1.84s | 1.57s | 1.46s | 1.29s |
| | Percentage | 9.36% | 4.96% | 5.75% | 4.18% | 3.37% | 2.63% |

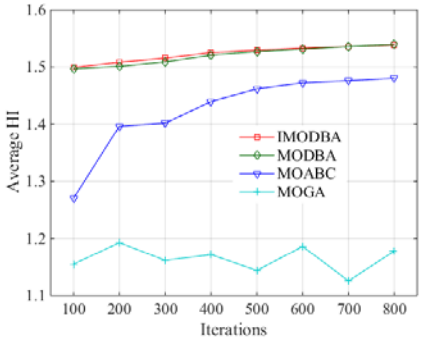
When the population number is 50, the average running time, average HI and average GD are compared under different iterations (from 100 to 800) as shown in Figures 22(a) ~ 22(f). Each simulation is repeated 50 times. From Figures 22(a) and 22(b), for both the gear pump and the camera, MODBA needs the longest running time than the others. With the help of ENS, IMODBA needs less running time than MODBA. The improvement of running time of IMODBA under different iterations is listed in Table 3. From Figures 22(c) ~ 22(f), for both the gear pump and the camera, it is obvious that IMODBA and MODBA can find better quality of non-dominated solutions than MOABC and MOGA. The quality of



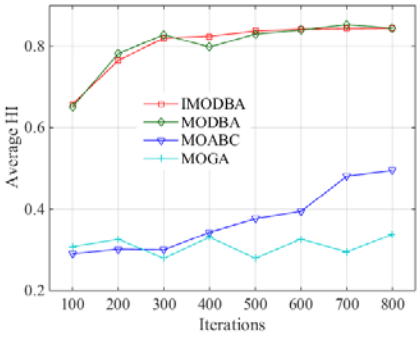

(a) Average running time of four optimization under different iterations based on the gear pump



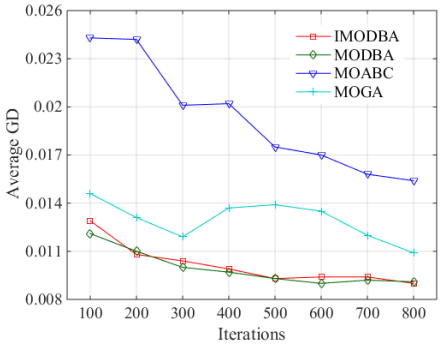

(b) Average running time of four optimization under different iterations based on the camera



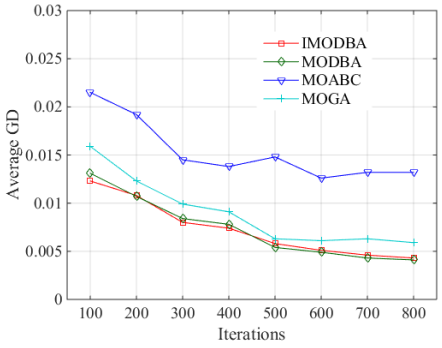

(c) Average HI of four optimization under different iterations based on the gear pump




(d) Average HI of four optimization under different iterations based on the camera




(e) Average DI of four optimization under different iterations based on the gear pump

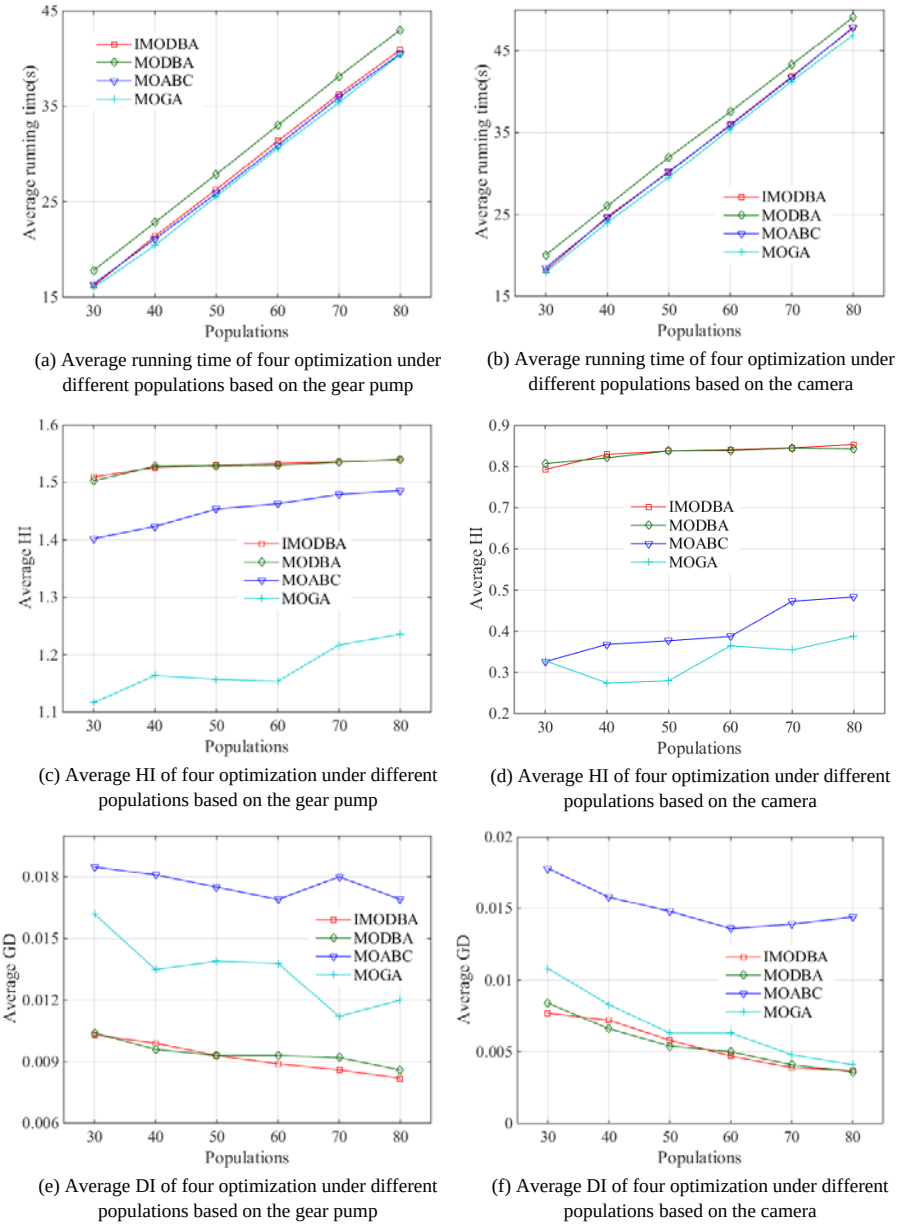



(f) Average DI of four optimization under different iterations based on the camera

**Fig. 22** Comparisons of the four optimization algorithms under different iterations based on the gear pump and the camera

solutions obtained by IMODBA is nearly the same with MODBA, because ENS sorts the solutions using less running time without changing the Pareto sorting results. From Figures 22(c) and 22(d), MOABC performs better than MOGA in terms of HI value while MOGA performs better than MOABC in terms of GD value from Figures 22(e) and 22(f). It is because MOABC performs better than MOGA in the diversity of solutions while MOGA performs better than MOABC in the convergence of solutions. When the iteration number is 500, simulations are made under different populations (from 30 to 80). Each simulation is repeated 50 times. From Figures 23(a) and 23(b), for both the gear pump and the camera, the

average running time of IMODBA is less than MODBA and it is comparable with MOABC and MOGA. The improvement of running time of IMODBA under different populations is listed in Table 4. From Figures 23(c) ~ 23(f), for both the gear pump and the camera, it is obvious that IMODBA and MODBA can also find better quality of solutions than MOABC and MOGA. MOGA performs worse than MOABC in terms of HI value from Figures 23(c) and 23(d), but it performs better than MOABC in terms of GD value from Figures 23(e) and 23(f). It is because MOGA performs better than MOABC in the convergence of solutions and it performs worse than MOABC in the diversity of solutions.




(a) Average running time of four optimization under different populations based on the gear pump

(b) Average running time of four optimization under different populations based on the camera

(c) Average HI of four optimization under different populations based on the gear pump

(d) Average HI of four optimization under different populations based on the camera

(e) Average DI of four optimization under different populations based on the gear pump

(f) Average DI of four optimization under different populations based on the camera

**Fig. 23** Comparisons of the four optimization algorithms under different populations based on the gear pump and the camera

The Pareto fronts obtained by the following three cases are also compared to verify the effectiveness of proposed method.

- Case 1: it considers the basic disassembly time of each part, the tool-change time and the direction-change time between different parts, but the moving time between different parts is ignored.
- Case 2: it considers the basic disassembly time of each part, the tool-change time, the direction-change time and the moving time between different parts. The moving time is calculated by Euclidean distance between different parts and moving speed of industrial robot's end-effector (ElSayed et al. 2012).
- Case 3: RDLBP is solved by the proposed method in this paper.

The iteration number, scout bees number, the selected sites number and follower bees number of IMODBA are 800, 80, 15 and 1 respectively. Simulations are repeated 100 times. 100 groups of non-dominated solutions are sorted to get the Pareto optimal solutions. Because ENS sorts the solutions without changing the Pareto sorting results, Pareto fronts obtained by MODBA are not listed in Tables 5 and 6. In Tables 5 and 6, for both the gear pump and the camera, it is obvious that Pareto fronts obtained by case 3 are different from the other cases. Case 1 ignores the moving time between different parts. Case 2 uses the Euclidean distance to calculate the moving time. It ignores the obstacle caused by the contour of EoLP. Thus, compared with cases 1 and 2, Pareto fronts obtained by case 3 are more applicable for robotic disassembly line. Besides, in Table 5, in terms of case 3, the $4^{th}$ solution of MOABC is dominated by the $4^{th}$ solution of IMODBA, the $5^{th}$ and $6^{th}$ solutions of MOABC are dominated by the $5^{th}$ solution of IMODBA and the $8^{th}$ solution of MOABC is dominated by the $7^{th}$ solution of IMODBA. The number of Pareto fronts obtained by IMODBA is greater than MOGA. In Table 6, in terms of case 3, the $1^{st}$ and $2^{nd}$ solutions of MOABC is dominated by the $2^{nd}$ solution of IMODBA and the $9^{th}$ solution of MOABC is dominated by the $8^{th}$ solution of IMODBA. The number of Pareto fronts obtained by IMODBA is also greater than MOGA. Thus, IMODBA performs better than MOABC and MOGA in terms of Pareto fronts. Finally, the Pareto optimal solutions of RDLBP based on the gear pump and the camera are listed in Tables 7 and 8 (for the disassembly direction, '1' and '2' mean 'Y+' and 'Y-' respectively).

## 7. Conclusion

In this paper, RDLBP is solved by IMODBA. Firstly, SIMM is used to generate feasible disassembly sequence and disassembly direction. After that, the feasible disassembly sequence and disassembly direction are used to obtain robotic disassembly line solution by robotic workstation assignment method. The multi-objectives used in this paper are to minimize the number of robotic workstations, balance the workload of robotic workstations and disassemble high demand parts as early as possible. With the help of ENS, IMODBA is proposed to solve RDLBP using less running time. Based on a gear pump and a camera, simulations are carried out to show the performance of IMODBA under different iterations and populations. The performance of IMODBA is also compared with MODBA, MOABC and MOGA under different iterations and populations. After that, Pareto fronts obtained by these optimization algorithms under different cases are also compared. It shows that IMODBA generates better quality of solutions than MOABC and MOGA. However, in this paper, SIMM is used to build disassembly model of EoLP which can be disassembled along orthogonal axes. In the future, more works should be finished to build disassembly model of EoLP with complex structure. In addition, the obstacle avoiding trajectory planning of industrial robot and the uncertainties in disassembly process will also be studied in the future.

**Table 5** Pareto fronts of three cases based on the gear pump

| | | Case 2 | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MOGA | | IMODBA | | | MOABC | | | MOGA | | | IMODB | |
| $f_2$ | $f_3$ | $f_1$ | $f_2$ | $f_3$ | $f_1$ | $f_2$ | $f_3$ | $f_1$ | $f_2$ | $f_3$ | $f_1$ | $f_2$ |
| 1 | 234 | 3 | 0.5262 | 238 | 3 | 0.5262 | 238 | 3 | 0.5262 | 238 | 3 | 0.0548 |
| 5 | 232 | 3 | 2.9193 | 235 | 3 | 2.9193 | 235 | 3 | 2.9193 | 235 | 3 | 0.0860 |
| 6 | 230 | 3 | 3.2023 | 234 | 3 | 3.2023 | 234 | 3 | 3.2023 | 234 | 3 | 0.1899 |
| 10 | 228 | 3 | 7.5069 | 233 | 3 | 7.5069 | 233 | 3 | 14.3724 | 232 | 3 | 1.0637 |
| | | 4 | 14.3724 | 232 | 3 | 14.3724 | 232 | 4 | 198.9339 | 231 | 3 | 2.0211 |
| | | 4 | 198.9339 | 231 | 4 | 198.9339 | 231 | 4 | 251.9134 | 230 | 3 | 3.8500 |
| | | 4 | 251.9134 | 230 | 4 | 251.9134 | 230 | 4 | 355.8085 | 229 | 3 | 6.3106 |
| | | 4 | 355.8085 | 229 | 4 | 355.8085 | 229 | 4 | 448.1330 | 228 | 4 | 27.0273 |
| | | 4 | 448.1330 | 228 | 4 | 448.1330 | 228 | | | | 4 | 27.2973 |
| | | | | | | | | | | | 4 | 27.4623 |
| | | | | | | | | | | | 4 | 263.5929 |
| | | | | | | | | | | | 4 | 284.1421 |

**Table 6** Pareto fronts of three cases based on the camera

| | | | | Case 2 | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MOGA | | | IMODBA | | | MOABC | | | MOGA | | | IMODBA | | |
| $f_3$ | $f_1$ | $f_2$ | $f_3$ | $f_1$ | $f_2$ | $f_3$ | $f_1$ | $f_2$ | $f_3$ | $f_1$ | $f_2$ | $f_3$ | $f_1$ | $f_2$ | $f_3$ |
| 310 | 3 | 0 | 310 | 3 | 0.2506 | 321 | 3 | 0.4466 | 294 | 3 | 0.4466 | 294 | 3 | 1.0411 | 331 |
| 292 | 3 | 1 | 292 | 3 | 0.4466 | 294 | 3 | 0.4501 | 293 | 3 | 0.4501 | 293 | 3 | 1.3141 | 319 |
| 284 | 3 | 2 | 284 | 3 | 0.4501 | 293 | 3 | 0.7571 | 275 | 3 | 0.7571 | 275 | 4 | 2.3049 | 318 |
| 280 | 3 | 5 | 280 | 3 | 0.7571 | 275 | 3 | 2.1059 | 274 | 3 | 2.1059 | 274 | 4 | 2.8017 | 312 |
| 275 | 3 | 9 | 275 | 3 | 2.1059 | 274 | 3 | 37.7046 | 271 | 3 | 37.7046 | 271 | 4 | 2.8432 | 309 |
| 274 | 3 | 14 | 274 | 3 | 37.7046 | 271 | 3 | 58.2641 | 269 | 3 | 58.2641 | 269 | 4 | 6.5750 | 307 |
| 270 | 3 | 52 | 270 | 3 | 58.2641 | 269 | 4 | 842.9434 | 268 | 4 | 842.9434 | 268 | 4 | 6.7485 | 306 |
| 268 | 3 | 68 | 268 | 4 | 842.9434 | 268 | | | | | | | 4 | 7.2913 | 303 |
| | | | | | | | | | | | | | 4 | 7.7011 | 288 |
| | | | | | | | | | | | | | 4 | 17.3577 | 287 |
| | | | | | | | | | | | | | 4 | 29.1523 | 275 |
| | | | | | | | | | | | | | 4 | 81.4263 | 273 |
| | | | | | | | | | | | | | 4 | 188.9248 | 272 |
| | | | | | | | | | | | | | 4 | 336.9715 | 270 |
| | | | | | | | | | | | | | 4 | 360.4361 | 268 |

**Table 7** Pareto optimal solutions of RDLBP obtained by different optimization algorithms based on the gear pump

| Disassembly sequence | Disassembly direction | Robotic worksta assignments |
|---|---|---|
| 5-6-2-3-1-7-15-8-14-12-9-13-11-10 | 2-2-2-2-2-2-1-2-1-1-2-2-1-1 | 1-1-1-1-1-1-1-2-2-2- |
| 2-3-5-6-1-7-15-8-14-12-10-9-13-11 | 2-2-2-2-2-2-1-2-1-1-2-1-2-2 | 1-1-1-1-1-1-1-2-2-2- |
| 2-3-5-6-1-7-15-14-12-8-10-9-13-11 | 2-2-2-2-2-2-1-1-1-2-2-1-2-2 | 1-1-1-1-1-1-1-2-2-2- |
| -14-3-5-4-6-1-2-7-9-8-12-13-10-11 | 1-1-2-2-2-2-2-2-2-2-1-1-1-1 | 1-1-1-1-1-2-2-2-2-2- |
| -14-13-2-6-3-5-1-4-7-8-12-10-9-11 | 1-1-1-2-2-2-2-2-2-2-1-1-1-1 | 1-1-1-1-1-2-2-2-2-2- |
| 6-5-3-2-4-7-10-9-15-14-12-8-13-11 | 2-2-2-2-2-2-2-2-2-1-1-1-1-1 | 1-1-1-1-1-1-1-2-2-2- |
| -14-13-5-6-1-2-3-4-7-9-12-10-8-11 | 1-1-1-2-2-2-2-2-2-2-1-1-1-1 | 1-1-1-1-1-2-2-2-2-2- |
| -14-3-6-13-5-1-4-2-12-9-7-8-10-11 | 1-1-2-2-1-2-2-2-2-1-1-2-1-2-1 | 1-1-1-1-2-2-2-2-3-3- |
| -14-3-6-13-4-1-5-2-12-9-7-10-8-11 | 1-1-2-2-1-2-2-2-2-1-1-2-1-2-1 | 1-1-1-1-2-2-2-2-3-3- |
| -14-5-2-13-3-1-4-6-12-9-7-8-11-10 | 1-1-2-2-1-2-2-2-2-1-1-2-1-2-1 | 1-1-1-1-2-2-2-2-3-3- |
| -14-13-3-4-1-5-2-6-12-9-7-10-8-11 | 1-1-1-2-2-2-2-2-2-1-1-2-1-2-1 | 1-1-1-1-2-2-2-2-2-3- |
| -14-13-3-4-1-5-2-6-12-9-10-7-8-11 | 1-1-1-2-2-2-2-2-2-1-1-1-2-2-1 | 1-1-1-1-2-2-2-2-2-3- |
| -14-13-6-2-3-5-1-4-7-11-12-9-10-8 | 1-1-1-2-2-2-2-2-2-2-2-1-1-1-1 | 1-1-1-1-1-2-2-2-2-2- |
| -14-13-2-3-5-1-4-6-7-8-12-10-11-9 | 1-1-1-2-2-2-2-2-2-2-2-1-1-1-1 | 1-1-1-1-1-2-2-2-2-2- |
| -14-13-2-3-5-6-1-4-7-8-12-10-9-11 | 1-1-1-2-2-2-2-2-2-2-2-1-1-1-1 | 1-1-1-1-1-2-2-2-2-2- |
| -14-3-6-13-4-1-5-2-12-9-8-7-11-10 | 1-1-2-2-1-2-2-2-2-1-1-1-2-1-2 | 1-1-1-1-2-2-2-2-3-3- |
| The remaining solutions are the same with the $1^{st} \sim 3^{rd}$ and $7^{th} \sim 12^{th}$ solutions | | |
| The Pareto optimal solutions are the same with the $1^{st} \sim 3^{rd}$ and $5^{th} \sim 12^{th}$ solutio | | |

**Table 8** Pareto optimal solutions of RDLBP obtained by different

| EoLP | Algorithms | No. | Disassembly sequence | Disassembly |
|---|---|---|---|---|
| Camera | IMODBA | 1 | 1-3-17-14-15-16-4-5-2-6-7-8-13-12-9-11-10 | 2-2-1-1-1-1-2-2-2- |
| | | 2 | 15-16-1-3-17-14-4-5-2-6-7-8-9-10-13-11-12 | 1-1-2-2-1-1-2-2-2- |
| | | 3 | 15-3-17-1-4-2-5-16-6-14-13-12-11-10-7-9-8 | 1-2-1-2-2-2-2-1-2- |
| | | 4 | 1-4-16-5-17-3-15-2-14-13-6-12-11-10-7-9-8 | 2-2-1-2-1-2-1-2-1- |
| | | 5 | 1-4-16-5-17-2-15-3-6-14-13-12-11-10-7-9-8 | 2-2-1-2-1-2-1-2-2- |
| | | 6 | 1-4-16-5-2-17-14-3-6-7-15-8-13-9-10-12-11 | 2-2-1-2-2-1-1-2-2- |
| | | 7 | 1-4-16-5-15-3-17-2-6-7-8-14-13-9-10-12-11 | 2-2-1-2-1-2-1-2-2- |
| | | 8 | 1-4-17-5-3-2-6-16-15-7-14-8-13-9-10-12-11 | 2-2-1-2-2-2-2-1-1- |
| | | 9 | 1-4-5-3-15-2-6-16-7-8-9-14-10-17-11-13-12 | 2-2-2-2-1-2-2-1-2- |
| | | 10 | 1-4-5-2-15-3-6-7-16-8-9-14-10-17-11-13-12 | 2-2-2-2-1-2-2-2-1- |
| | | 11 | 1-4-2-5-3-6-7-8-16-9-10-14-15-17-11-13-12 | 2-2-2-2-1-2-2-2-1- |
| | | 12 | 1-4-5-3-2-6-7-8-16-9-10-14-15-17-11-13-12 | 2-2-2-2-2-2-2-2-1- |
| | | 13 | 1-4-5-3-2-6-7-8-9-16-10-14-15-17-11-13-12 | 2-2-2-2-2-2-2-2-2- |
| | | 14 | 1-4-2-5-3-6-7-8-9-10-16-15-14-17-11-13-12 | 2-2-2-2-2-2-2-2-2- |
| | | 15 | 1-4-5-2-3-6-7-8-9-10-16-15-14-17-11-13-12 | 2-2-2-2-2-2-2-2-2- |
| | MOABC | 1 | 15-16-14-1-3-17-4-5-2-6-7-8-13-12-9-11-10 | 1-1-1-2-2-1-2-2-2- |
| | | 2 | 3-17-14-15-16-1-4-5-2-6-7-8-9-10-11-13-12 | 2-1-1-1-1-2-2-2-2- |
| | | 3 | 1-4-16-5-3-2-6-17-15-7-14-8-13-9-10-12-11 | 2-2-1-2-2-2-2-1-1- |
| | | 4-16 | The remaining solutions are the sa | |
| | MOGA | 1-14 | The Pareto optimal solutions are the | |

**Acknowledgements** This work is supported by the National Natural Science Foundation of China (Grant Nos. 5177539951675389 and 51475343), the Keygrant Project of Hubei Technological Innovation Special Fund (Grant No. 2016AAA016), Engineering and Physical Sciences Research Council (EPSRC), UK (Grant No. EP/N018524/1) and the China Scholarship Council (201606950054).

## References

1. Tao F, Cheng Y, Zhang L, Nee AYC (2017) Advanced manufacturing systems: socialization characteristics and trends. J Intell Manuf 28(5):1079-1094. https://doi.org/10.1007/s10845-015-1042-8
2. Wang LH, Wang XV, Gao L, Váncza J (2014) A cloud-based approach for WEEE remanufacturing. CIRP Ann-Manuf Techn 63(1):409-412. https://doi.org/10.1016/j.cirp.2014.03.114
3. D'Adamo I, Rosa P (2016) Remanufacturing in industry: advices from the field. Int J Adv Manuf Tech 86(9-12): 2575-2584. http://doi.org/10.1007/s00170-016-8346-5
4. Xu BS (2010) State of the art and future development in remanufacturing engineering. Trans Mat Heat T 31(1):10-14
5. Guide VDR (2000) Production planning and control for remanufacturing: industry practice and research needs. J Oper Manag 18(4):467-483. https://doi.org/10.1016/S0272-6963(00)00034-6
6. Xu BS (2010) Recent progress of remanufacturing industry and technology in China. Therm Spray Technol 2(3):1-6
7. Savaskan RC, Bhattacharya S, Van WLN (2004) Closed-loop supply chain models with product remanufacturing. Manage Sci 50(2):239-252. https://doi.org/10.1287/mnsc.1030.0186
8. Shakourloo A (2017) A multi-objective stochastic goal programming model for more efficient remanufacturing process. Int J Adv Manuf Tech 91(1-4): 1007-1021. https://doi.org/10.1007/s00170-016-9779-6
9. Priyono A, Ijomah W, Bititci U (2016) Disassembly for remanufacturing: A systematic literature review, new model development and future research needs. J Ind Engineering Manage 9(4):899-932. https://doi.org/10.3926/jiem.2053
10. Alavidoost MH, Zarandi MF, Tarimoradi M, Nemati Y (2017) Modified genetic algorithm for simple straight and U-shaped assembly line balancing with fuzzy processing times. J Intell Manuf 28(2):313-336. https://doi.org/10.1007/s10845-014-0978-4
11. Kim HW, Lee DH (2018) A sample average approximation algorithm for selective disassembly sequencing with abnormal disassembly operations and random operation times. Int J Adv Manuf Tech 1-14. https://doi.org/10.1007/s00170-018-1667-9
12. Vongbunyong S, Kara S, Pagnucco M (2012) A framework for using cognitive robotics in disassembly automation. In: David AD (ed) Leveraging Technology for a Sustainable World. Spring, Berkeley, pp 173-178. https://doi.org/10.1007/978-3-642-29069-5_30
13. Vongbunyong S, Kara S, Pagnucco M (2013) Basic behaviour control of the vision-based cognitive robotic disassembly automation. Assembly Autom 33(1):38-56. https://doi.org/10.1108/01445151311294694
14. Vongbunyong S, Kara S, Pagnucco M (2013) Application of cognitive robotics in disassembly of products. CIRP Ann-Manuf Techn 62(1):31-34. https://doi.org/10.1016/j.cirp.2013.03.037
15. Vongbunyong S, Kara S, Pagnucco M (2015) Learning and revision in cognitive robotics disassembly automation. Robot Cim-Int Manuf 34:79-94. https://doi.org/10.1016/j.rcim.2014.11.003
16. Wang BX, Guan ZL, Ullah S, Xu XH, He ZD (2017) Simultaneous order scheduling and

mixed-model sequencing in assemble-to-order production environment: a multi-objective hybrid artificial bee colony algorithm. J Intell Manuf 28(2):419-436. https://doi.org/10.1007/s10845-014-0988-2

17. Wang LH, Schmidt B, Givehchi M, Adamson G (2015) Robotic assembly planning and control with enhanced adaptability through function blocks. Int J Adv Manuf Tech 77(1-4):705-715. https://doi.org/10.1007/s00170-014-6468-1
18. Ullah S, Guan ZL, Zhang L, Zhang F, Wang BX, Mirza J (2017) Multi-objective artificial bee colony algorithm for order oriented simultaneous sequencing and balancing of multi-mixed model assembly line. J Intell Manuf 1-26. https://doi.org/10.1007/s10845-017-1316-4
19. Gungor A, Gupta SM (1999) Disassembly line balancing. In Proceedings of 1999 Annual Meeting of the Northeast Decision Sciences Institute, Newport, USA, 24-26 March 1999, pp 24-26.
20. Mcgovern SM, Gupta SM (2007) A balancing method and genetic algorithm for disassembly line balancing. Eur J Oper Res 179(3):692-708. https://doi.org/10.1016/j.ejor.2005.03.055
21. Ilgin MA, Akçay H, Araz C (2017) Disassembly line balancing using linear physical programming. Int J Prod Res 55(20):1-12. https://doi.org/10.1080/00207543.2017.1324225
22. Ding LP, Feng YX, Tan JR, Gao YC (2010) A new multi-objective ant colony algorithm for solving the disassembly line balancing problem. Int J Adv Manuf Tech 48(5-8):761-771. https:/doi.org/10.1007/s00170-009-2303-5
23. Ayyuce AK, Turkbey O (2013) Multi-objective optimization of stochastic disassembly line balancing with station paralleling. Comput Ind Eng 65(3):413-425. https://doi.org/10.1016/j.cie.2013.03.014
24. Tuncel E, Zeid A, Kamarthi S (2014) Solving large scale disassembly line balancing problem with uncertainty using reinforcement learning. J Intell Manuf 25(4):647-659. https://doi.org/10.1007/s10845-012-0711-0
25. Bentaha ML, Battaïa O, Dolgui A (2014) A sample average approximation method for disassembly line balancing problem under uncertainty. Comput Oper Res 51:111-122. https://doi.org/10.1016/j.cor.2014.05.006
26. Hezer S, Kara Y (2015) A network-based shortest route model for parallel disassembly line balancing problem. Int J Prod Res 53(6):1849-1865. https://doi.org/10.1080/00207543.2014.965348
27. Kalayci CB, Hancilar A, Gungor A, Gupta SM (2015) Multi-objective fuzzy disassembly line balancing using a hybrid discrete artificial bee colony algorithm. J Manuf Syst 37:672-682. https://doi.org/10.1016/j.jmsy.2014.11.015
28. Mete S, Çil ZA, Ağpak K, Özceylan E, Dolgui A (2016) A solution approach based on beam search algorithm for disassembly line balancing problem. J Manuf Syst 41:188-200. https://doi.org/10.1016/j.jmsy.2016.09.002
29. Kalayci CB, Gupta SM (2013) Ant colony optimization for sequence-dependent disassembly line balancing problem. J Manuf Technol Manage 24(3):413-427. https://doi.org/10.1108/17410381311318909
30. Kalayci CB, Gupta SM (2013) A particle swarm optimization algorithm with neighborhood-based mutation for sequence-dependent disassembly line balancing problem. Int J Adv Manuf Tech 69(1-4):197-209. https://doi.org/10.1007/s00170-013-4990-1
31. Liu J, Wang S (2017) Balancing Disassembly Line in Product Recovery to Promote the Coordinated Development of Economy and Environment. Sustainability 9(2):309-323. https://doi.org/10.3390/su9020309
32. Kalayci CB, Gupta SM (2014) A tabu search algorithm for balancing a sequence-dependent

disassembly line. Prod Plan Control 25(2):149-160. https://doi.org/10.1080/09537287.2013.782949

33. Kalayci CB, Polat O, Gupta SM (2016) A hybrid genetic algorithm for sequence-dependent disassembly line balancing problem. Ann Oper Res 242(2):321-354. https://doi.org/10.1007/s10479-014-1641-3
34. Kalayci CB, Gupta SM (2013) Artificial bee colony algorithm for solving sequence-dependent disassembly line balancing problem. Expert Syst Appl 40(18):7231-7241. https://doi.org/10.1016/j.eswa.2013.06.067
35. Alshibli M, ElSayed A, Kongar E, Sobh TM, Gupta SM (2016) Disassembly sequencing using tabu search. J Intell Robot Syst 82(1):69-79. https://doi.org/10.1007/s10846-015-0289-9
36. ElSayed A, Kongar E, Gupta SM (2010) A genetic algorithm approach to end-of-life disassembly sequencing for robotic disassembly. In Proceedings of the 2010 Northeast Decision Sciences Institute Conference, Alexandria, USA, 26-28 March 2010, pp 402-408
37. ElSayed A, Kongar E, Gupta SM, Sobh T (2011) An online genetic algorithm for automated disassembly sequence generation. In Proceedings of the ASME 2011 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Washington DC, USA, 28-31 August 2011, pp 657-664. https://doi.org/10.1115/DETC2011-48635
38. ElSayed A, Kongar E, Gupta SM, Sobh T (2012) A robotic-driven disassembly sequence generator for end-of-life electronic products. J Intell Robot Syst 68(1):43-52. https://doi.org/10.1007/s10846-012-9667-8
39. Agrawal S, Tiwari MK (2008) A collaborative ant colony algorithm to stochastic mixed-model u-shaped disassembly line balancing and sequencing problem. Int J Prod Res 46(6):1405-1429. https://doi.org/10.1080/00207540600943985
40. Pham DT, Ghanbarzadeh A (2007) Multi-objective optimisation using the bees algorithm. In Proceedings of the 3rd International Virtual Conference on Intelligent Production Machines and Systems, Cardiff, UK, 3-14 July 2007, pp 529-533
41. Tapkan P, Özbakır L, Baykasoğlu A (2012) Bees algorithm for constrained fuzzy multi-objective two-sided assembly line balancing problem. Optim Lett 6(6):1-11. https://doi.org/10.1007/s11590-011-0344-9
42. Ercin O, Coban R (2011) Comparison of the artificial bee colony and the bees algorithm for PID controller tuning. In Proceedings of 2011 International Symposium on Innovations in Intelligent Systems and Applications, Istanbul, Turkey, 15-18 June 2011, pp 595-598. https://doi.org/10.1109/INISTA.2011.5946157
43. Mastrocinque E, Yuce B, Lambiase A, Packianather MS (2013) A multi-objective optimization for supply chain network using the bees algorithm. Int J Eng Bus Manage 5(38):1-11. https://doi.org/10.5772/56754
44. Lu W, Quan Z, Liu Q, Zhang D, Xu W (2015) QoE based spectrum allocation optimization using bees algorithm in cognitive radio networks. In Proceedings of 2015 International Conference on Algorithms and Architectures for Parallel Processing, Zhang Jiajie, China, 18-20 November 2015, pp 327-338. https://doi.org/10.1007/978-3-319-27119-4_23
45. Xu WJ, Tian SS, Liu Q, Xie YQ, Zhou ZD, Pham DT (2016) An improved discrete bees algorithm for correlation-aware service aggregation optimization in cloud manufacturing. Int J Adv Manuf Tech 84(1-4):17-28. https://doi.org/10.1007/s00170-015-7738-2
46. Minella G, Ruiz R, Ciavotta M (2008) A review and evaluation of multiobjective algorithms for the flowshop scheduling problem. Informs J Comput 20(3):451-471. https://doi.org/10.1287/ijoc.1070.0258

47. Tian GD, Zhou MC, Chu JW, Liu YM (2012) Probability evaluation models of product disassembly cost subject to random removal time and different removal labor cost. IEEE T Autom Sci Eng 9(2):288-295. https://doi.org/10.1109/TASE.2011.2176489
48. Laili YJ, Tao F, Zhang L, Sarker BR (2012) A study of optimal allocation of computing resources in cloud manufacturing systems. Int J Adv Manuf Tech 63(5-8): 671-690. https://doi.org/10.1007/s00170-012-3939-0
49. Guo XW, Liu SX, Zhou MC, Tian GD (2016) Disassembly sequence optimization for large-scale products with multiresource constraints using scatter search and petri nets. IEEE T Cybernetics 46(11):2435-2446. https://doi.org/10.1109/TCYB.2015.2478486
50. Jin, GQ, Li WD, Xia K (2013) Disassembly matrix for liquid crystal displays televisions. Proc CIRP 11:357-362. https://doi.org/10.1016/j.procir.2013.07.015
51. Jin GQ, Li WD, Wang S, Gao SM (2015) A systematic selective disassembly approach for Waste Electrical and Electronic Equipment with case study on liquid crystal display televisions. P I Mech Eng B-J Eng special issue:1-18. https://doi.org/10.1177/0954405415575476
52. Pham QT, Pham DT, Castellani M (2012) A modified bees algorithm and a statistics-based method for tuning its parameters. P I Mech Eng I-J Sys 226(3):287-301. https://doi.org/10.1177/0959651811422759
53. Zhang XY, Tian Y, Cheng R, Jin YC (2015) An efficient approach to nondominated sorting for evolutionary multiobjective optimization. IEEE T Evolut Comput 19(2):201-213. https://doi.org/10.1109/TEVC.2014.2308305
54. Deb K, Pratap A, Agarwal S, Meyarivan TAMT (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE T Evolut Comput 6(2):182-197. http://doi.org/10.1109/4235.996017
55. Roy PC, Islam MM, Deb K (2016) Best Order Sort: A new algorithm to non-dominated sorting for evolutionary multi-objective optimization. In Proceedings of 2016 on Genetic and Evolutionary Computation Conference Companion, Colorado, USA, 20-24 July 2016, pp 1113-1120. http://doi.org/10.1145/2908961.2931684
56. Igor S (2017) CNC Camera box #ARIADNE. https://grabcad.com/library/cnc-camera-box-ariadne-1. Accessed 14 September 2017
57. KUKA (2017) KUKA LBR linear axis. https://www.kuka.com/en-de/products/robot-systems/robot-periphery/linear-units/lbr-linear-axis. Accessed 01 January 2017
58. Mathworks (2014) R2014b release highlights. https://www.mathworks.com/products/new_products/release2014b.html. Accessed 01 January 2014
59. Zitzler E, Thiele L, Laumanns M, Fonseca CM, Da FVG (2003) Performance assessment of multiobjective optimizers: An analysis and review. IEEE T Evolut Comput 7(2):117-132. https://doi.org/10.1109/TEVC.2003.810758
60. Okabe T, Jin Y, Sendhoff B (2003) A critical survey of performance indices for multi-objective optimisation. In Proceedings of 2003 Congress on IEEE on Evolutionary Computation, Canberra, Australia, 8-12 December 2003, pp 878-885. https://doi.org/10.1109/CEC.2003.1299759
61. Zitzler E, Thiele L (1998) Multiobjective optimization using evolutionary algorithms—a comparative case study. In Proceedings of 1998 International Conference on Parallel Problem Solving from Nature, Amsterdam, Netherlands, 27-30 September 1998, pp 292-301. https://doi.org/10.1007/BFb0056872
62. Beume N, Fonseca CM, López-Ibáñez M, Paquete L, Vahrenhold J (2009) On the complexity of computing the hypervolume indicator. IEEE T Evolut Comput 13(5):1075-1082. https://doi.org/10.1109/TEVC.2009.2015575

63. Akbari R, Hedayatzadeh R, Ziarati K, Hassanizadeh, B (2012). A multi-objective artificial bee colony algorithm. Swarm Evol Comput 2: 39-52. https://doi.org/10.1016/j.swevo.2011.08.001
64. Yang CL, Kuo RJ, Chien CH, Quyen NTP (2015) Non-dominated sorting genetic algorithm using fuzzy membership chromosome for categorical data clustering. Appl Soft Comput 30:113-122. https://doi.org/10.1016/j.asoc.2015.01.031
65. Akpınar S, Bayhan GM (2011) A hybrid genetic algorithm for mixed model assembly line balancing problem with parallel workstations and zoning constraints. Eng Appl Artif Intel 24(3):449-457. https://doi.org/10.1016/j.engappai.2010.08.006