Multi-objective evolutionary simulated annealing optimisation for mixed-model multi-robotic disassembly line balancing with interval processing time

Fang, Yilin; Ming, Hao; Li, Miqing; Liu, Quan; Pham, Duc Truong

DOI:
10.1080/00207543.2019.1602290

Multi-objective evolutionary simulated annealing optimisation for mixed-model multi-robotic disassembly line balancing with interval processing time

Yilin Fang [a,b*] Hao Ming [a,b], Miqing Li [c], Quan Liu [a,b] and Duc Truong Pham [d]

[a] School of Information Engineering, Wuhan University of Technology, Wuhan, People's Republic of China; [b] Key Laboratory of Fiber Optic Sensing Technology and Information Processing, Ministry of Education, Wuhan, People's Republic of China; [c] CERCIA, School of Computer Science, University of Birmingham, Birmingham, UK; [d] School of Engineering, University of Birmingham, Birmingham, UK
*Corresponding author. Email: fangspirit@whut.edu.cn

This paper considers the design and balancing of mixed-model disassembly lines with multi-robotic workstations under uncertainty. Tasks of different models are performed simultaneously by the robots which have different capacities for dis-assembly. The robots have unidentical task times and energy consumption respectively. Task precedence diagrams are used to model the precedence relations among tasks. Considering uncertainties in disassembly process, the task processing times are assumed to be interval numbers. A mixed-integer mathematical programming model is proposed to minimise the cycle time, peak workstation energy consumption, and total energy consumption. This model has a significant managerial impli-cation in real-life disassembly line systems. Since the studied problem is known as NP-hard, a metaheuristic approach based on an evolutionary simulated annealing algorithm is developed. Computational experiments are conducted and the results demonstrate the proposed algorithm outperforms other multi-objective algorithms on optimisation quality and computational efficiency.

Keywords: mixed-model disassembly line; multi-robotic workstation; uncertain processing time; multi-objective optimi-sation; simulated annealing algorithm

1. Introduction

As the consumption rates and the demand for new products dramatically increase, a great number of end-of-life (EOL) products are also continuously being disposed of, leading to a number of environmental problems (Vongbunyong and Chen 2015). Disassembly, as a mandatory step and most challenging part of EOL product recovery process, can be profitable and less harmful to the environment. There are several active research areas in the disassembly literature, with disassembly line balancing problem (DLBP) being one of the major research streams. The DLBP is to assign disassembly tasks to an ordered sequence of workstations optimally by satisfying specific constraints. It is one of the most important problems in designing disassembly lines. Since the first work on DLBPs by Güngör and Gupta (1999), many researches have dedicated their efforts to develop solution approaches for the simple DLBP. In recent years, several researchers tried to model more realistic and generalised forms of DLBP to narrow the gap between research and practice. These generalised DLBPs can be characterised by additional features such as mixed-model disassembly lines (Agrawal and Tiwari 2008), disassembly lines with multi-robotic workstation (Fang et al. 2019), disassembly under uncertainty (Bentaha et al. 2018), among many others. Detail reviews of such studies are given by Battaïa and Dolgui (2013), Bentaha, Dolgui, and Battaïa (2015a) and Özceylan et al. (2018).

In DLBP area, disassembly lines can be categorised into three types based on the variety of EOL products disassem-bled on the line: single-model, mixed-model and multi-model disassembly lines. In single-model disassembly lines, only one product type is allowed to be disassembled. Unlike the case of single-model lines, more than one type of product is disassembled simultaneously and continuously on mixed-model disassembly lines and therefore mixed-model line is more suitable to improve the flexibility to adapt to the product model changes (Paksoy et al. 2013). In multi-model lines, several products are disassembled in separate batches. Since setup times need to be considered between the lots, multi-model lines are used rarely. The vast majority of the studies in DLBP literature consider the single-model case due to natural complexity of DLBP. However, as the variability of the EOL products coming into disassembly lines become higher, remanufacturing companies try to convert single-model lines into mixed-model lines in order to provide more flexibility and capability of responding to variant disassembly demand and reduce line building and maintain costs. In contrast to the practical needs of mixed-

model disassembly lines from industries, the literature in this specific area is scarce and there are only five relevant studies which belong to Agrawal and Tiwari (2008), Paksoy et al. (2013), Riggs, Battaïa, and Hu (2015), Ilgin, Akcay, and Araz (2017) and Fang et al. (2019).

Another classification scheme of DLBPs is based on line type: straight lines, U-shaped lines and lines with various parallel layouts. In straight disassembly line, a set of workstations is linked together by a transport system whose mission is to move products or subassemblies among serially constructed workstations. A set of tasks is assigned to each workstation and these tasks are performed by a worker (robot) one after the other (Battaïa and Dolgui 2013). In DLBP literature, the majority of the previously published studies focus on the straight line type. However, the real-world industrial disassembly system may face many practical challenges, including large-sized EOL products, high demand of the recovered parts, com-plicated flow processes, and complex inventory problems. Under such a scenario efficient disassembly line designs cannot be achieved by considering straight line type. For example, most of the complex EOL products have dozens of disassembly tasks. If they are disassembled in a straight line, hundreds of workstations will need. Therefore, it is highly desirable to study other line configurations to overcome these complications. One of the promising strategies is paralleling. In DLBP literature, parallel layout is only considered in four references. Aydemir-Karadag and Turkbey (2013) investigated the introduction of parallel stations into work centres instead of straight lines in terms of several objective functions. This type of paralleling requires the duplication of all workers and tools/equipment needed for the tasks assigned to that work centre. Hezer and Kara (2015) and Mete et al. (2018) introduced line paralleling in which two or more straight assembly/disassembly lines are located in parallel and balanced simultaneously with common resources. Moreover, we firstly dealt with in Fang et al. (2019) generalised mixed-model DLBPs with parallel robots in each workstation, that is also considered in this paper. This problem is called mixed-model DLBP with multi-robotic workstation or mixed-model multi-robotic DLBP (MMDLBP). The analyses in Fang et al. (2019) show that using robot paralleling in each workstation can contribute to disassembly lines positively in terms of shorter line length, higher recovery rates and reduction of station idle time.

Apart from the above-mentioned practical challenges, successfully solving real-world DLBPs are usually subject to a great number of uncertainties such as uncertain task processing times. Task processing times, in fact, may vary during the disassembly process because of the high degree of uncertainty associated with the quality and the quantity of the products (Bentaha, Battaïa, and Dolgui 2015b). The uncertainty in task processing times of DLBP has been addressed using stochastic and fuzzy programming techniques. In stochastic approaches, task processing time is assumed to obey a known probability distribution by Agrawal and Tiwari (2008), Aydemir-Karadag and Turkbey (2013), Bentaha, Battaïa, and Dolgui (2014, 2015b, 2015c), Riggs, Battaïa, and Hu (2015), Altekin, Bayındır, and Gümü¸skaya (2016), Altekin (2017), Bentaha et al. (2018), and Zheng et al. (2018). In fuzzy approaches, task processing time is represented as a fuzzy value by Özceylan and Paksoy (2014), Kalayci et al. (2015), Seidi and Saghari (2016), and Zhang et al. (2017). In these studies, the probability distribution and fuzzy membership functions are usually assumed to be known, which is, however, not true in real-world situations. It is difficult to choose an appropriate function to model the uncertainty in advance. Compared with the two approaches, the upper and lower bounds of task processing time are often easy to know, which makes it very practical to use intervals to represent uncertain task processing time (Sotskov, Lai, and Werner 2013; Lu, Lin, and Ying 2014). To the best of authors' knowledge, there is no interval approach reported in DLBP literature to address this issue.

In view of uncertainty in task processing time and aforementioned advantages offered by mixed-model disassembly lines with multi-robotic workstation and interval approach, this study considers MMDLBP with interval task processing time. The single-model multi-robotic disassembly line (SMDL) and mixed-model multi-robotic disassembly line (MMDL) are respectively described in Figure 1(a) and (b). In the two types of lines, there is a possibility of assigning more different skilled robots than one to each workstation and these robots can simultaneously perform tasks on the same individual product. However, only the tasks for the same model can be handled during a cycle time (Ct) in SMDL while the tasks for different models can be handled simultaneously in MMDL. MMDLBP with interval task processing time is usually occurred in disassembly factories for several large-sized EOL products consisting of numerous components. For instance, in the automobile industry, automobiles are manufactured into different models within the same family. Due to high degree of complexity and uncertainty associated with the structure and the quality of the used or wrecked automobiles and high cost to build and maintain a disassembly line, the factories disassemble one automobile model with different features or several automobile models on a single disassembly line using various paralleling configurations and uncertainty processing techniques. Under these circumstances, MMDLBP with interval task processing time arises to smooth the disassembly process and decrease the cost.

In MMDL, robots are scarce resources and the robot energy consumption is a major expense. The reduction of energy consumption beneficially influences industry's impact on the environment (Nilakantan et al. 2017). Moreover, cycle time is commonly used for assessing line efficiency due to its direct relationship with production rate. Hence, three objectives are considered simultaneously in this study: minimisation of cycle time, minimisation of peak workstation energy consumption

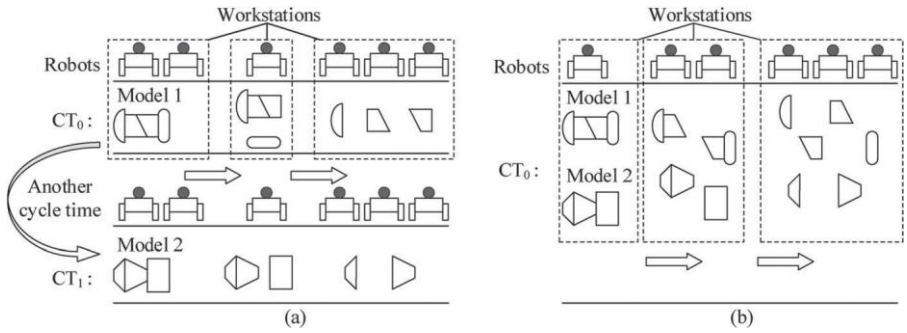

Figure 1. Single-model and mixed-model multi-robotic disassembly lines.

and minimisation of total energy consumption. Our purpose is to present a solution methodology to obtain solutions for MMDLBP with interval task processing time and enable the decision maker for evaluating a number of alternative solutions. Due to the DLBP's NP-hard nature (McGovern and Gupta 2007a) and the inclusion of interval task times, robot paralleling configuration and simultaneously considered multiple objectives listed above, the solving difficulty for our problem will increase geometrically with the increase of the problem scale. In recent years, various metaheuristic algorithms have been widely used in solving DLBP because of their excellent solution performance, such as genetic algorithms (McGovern and Gupta 2007b; Aydemir-Karadag and Turkbey 2013; Jiang et al. 2016; Kalayci, Polat, and Gupta 2016; Pistolesi et al. 2018; Fang et al. 2019; Wang, Li, and Gao 2019), ant colony algorithms (Agrawal and Tiwari 2008; Ding et al. 2010; Mete et al. 2018), artificial bee colony algorithms (Kalayci and Gupta 2013a; Kalayci et al. 2015; Gao et al. 2018), simulated annealing algorithms (Kalayci and Gupta 2013b; Wang, Li, and Gao 2019), artificial fish swarm algorithms (Zhang et al. 2017), and firefly algorithms (Zhu, Zhang, and Wang 2018). Most of these studies deal with multi-objective DLBPs, but optimise each objective step by step by using the lexicographic method, or get a single objective by using the weighted method. However, objectives generally conflicting with each other, thus these approaches cannot guarantee the equilibrium of the optimisation of all objectives. In contrast, Pareto-based metaheuristic algorithms (Ding et al. 2010; Aydemir-Karadag and Turkbey 2013; Kalayci et al. 2015; Jiang et al. 2016; Zhang et al. 2017; Gao et al. 2018; Pistolesi et al. 2018; Zhu, Zhang, and Wang 2018; Fang et al. 2019; Wang, Li, and Gao 2019) overcome the disadvantage as they provide a uniform set of Pareto-optimal solutions per run, with no weights or orders to specify. Therefore, it is appropriate to design metaheuristic algorithms based on Pareto solution set to solve multi-objective DLBPs.

As mentioned above, this paper deals with MMDLBP under uncertainty of the disassembly task times which are represented as interval numbers. Task precedence diagram (TPD) is used to model the precedence relations among tasks. A mathematical formulation and a metaheuristic algorithm are proposed. In DLBP literature, the only study which addresses MMDLBP belongs to Fang et al. (2019). However, unlike the deterministic MMDLBP in Fang et al. (2019), MMDLBP under uncertainty is addressed in this study. To the best knowledge of the authors, this is the first study dealing with DLBP under uncertainty by using interval approach. The main contributions can be listed as follows: Firstly, the uncertain dis-assembly task processing time is assumed as interval number and it is solved by introducing interval possibility degree and order relations between intervals. Secondly, a mixed integer programming (MIP) model is developed for uncertain MMDLBP with the objectives of minimising the cycle time, the peak workstation energy consumption and the total energy consumption. Thirdly, an evolutionary simulated annealing algorithm using Pareto-domination based acceptance criterion (PDSA-EA) is proposed for multi-objective optimisation of MMDLBP with interval task processing time.

The remainder of this paper is organised as follows. The description of MMDLBP with interval task processing time and the mathematical formulation with an illustrative example are given in Section 2. In Section 3, the implementation details of proposed hybrid evolutionary simulated annealing algorithm are presented. Computational results are given in Section 4. A conclusion and possible future research avenues are presented in Section 5.

2. Mathematical formulation

2.1. Problem description

Considering the objectives and conditions, DLBPs mainly fall into two categories. DLBP-I has the objective of minimising the number of workstations within a given cycle time. And DLBP-II, addressed in our study, aims to minimise the cycle time with fixed number of workstations. In this paper, the maximum robots available for each workstation are predetermined by the system designer according to the product size, tools availability and workstation design (Fattahi, Roshani, and Roshani

2011). Thus, MMDLBP can be defined as that a sequence of tasks of mixed-model products are assigned to the robots in fixed number of multi-robotic workstations where the combined task precedence relations are satisfied and the cycle time, peak workstation energy consumption and total energy consumption are minimised.

The assumptions for the considered problem are given as follows:

- Similar products of various models are disassembled simultaneously.
- Parallel tasks of the same model are not allowed but are permitted among different models.
- The task times are given as intervals.
- Robots have different processing times on the same task.
- The operation and standby energy consumptions of each robot per time unit are given.
- The TPDs of different models are given.
- Workstations are aligned in a serial manner and the number of workstations is given.
- The maximum number of robots assigned to each workstation is given.
- Each task cannot be divided and each robot can perform at most one task at the same time.
- Each robot is available for all workstations, but can only be assigned to one workstation at most.
- Transmission times of products/subassemblies are ignored.

2.2. The MIP model of MMDLBP

2.2.1. Notations

Sets and parameters

- t_{ik}^{I} : interval time required to perform task i by robot k.
- Nt: total number of tasks.
- I: set of tasks,$I = \{1, \ldots, e, \ldots, i, \ldots, Nt\}$.
- Nw: total number of workstations.
- J: set of workstations,$J = \{1, \ldots, j, \ldots, Nw\}$.
- Nr: total number of robots.
- K: set of robots,$K = \{1, \ldots, k, \ldots, Nr\}$.
- $N R_{\max}$: maximum number of robots allowed at each workstation.
- OP_k : Operation energy consumption of the robot k per time unit.
- SP_k : Standby energy consumption of the robot k per time unit.
- $P(i)$: set of immediate predecessors of the task i.
- $P_a(i)$: set of all predecessors of the task i.
- $S(i)$: set of immediate successors of the task i.
- $S_a(i)$: set of successors of the task i.
- L: a large positive number.

Continuous variables

- C_i: completion time of task i.
- CW_j: completion time of workstation j.
- Ct: cycle time.
- EC_j^{I} : energy consumption of workstation j.
- PE^{I} : peak workstation energy consumption.
- TE^{I} : total energy consumption.

Binary variables

- X_{ijk} = 1if task i is assigned to robot k at workstation j (and 0 otherwise).
- W_{jk} = 1if robot k is used at workstation j (and 0 otherwise).
- Y_{ih} = 1if task i and h belong to the same model (and 0 otherwise).
- Z_{ih} = 1if task i is assigned earlier than task h (and 0 otherwise).

2.2.2. Objective functions

$$f_1 : \min Ct, \tag{1}$$

$$f_2 : \min PE^I = \{\max(EC_j^I),\ \ \forall j \in J\}, \tag{2}$$

$$f_3 : \min TE^I = EC_j^I, \tag{3}$$

$$EC_j^I = X_{ijk} \cdot (t_{ik}^I \cdot OP_k + (Ct - t_{ik}^I) \cdot SP_k),\ \ \forall j \in J. \tag{4}$$

Objective function (1) defines to minimise the cycle time. Objective function (2) and (3) mean to minimise the peak workstation energy consumption and the total energy consumption. Equation (4) represents the energy consumption at workstation j in operation mode and standby mode. Different from the multi-objective optimisation approach where the first objective is viewed as the main objective to be considered first and the remaining objectives as secondary objectives, all the objectives are viewed equally and optimised simultaneously in our study.

2.2.3. Constraints

$$X_{ijk} = 1,\ \forall i \in I, \tag{5}$$

$$j \cdot X_{hjk} \le j \cdot X_{ijk},\ \ \forall i \in I, \forall h \in P(i), \tag{6}$$

$$C_i - C_h + L(1 - X_{ijk}) + L(1 - X_{hjk}) \ge X_{ijk} \cdot t_{ik}^I,\ \ \forall i \in I,\ \ \forall h \in P(i),\ \ \forall j \in J, \tag{7}$$

$$Y_{ih} \cdot (C_i - C_h) + L(1 - X_{ijk}) + L(1 - X_{hjk}) + L \cdot Z_{ih} \ge Y_{ih} \cdot X_{ijk} \cdot t_{ik}^I,\ \ \forall i \in I,\ \ \forall h \in I \backslash \{P_a(i) \cup S_a(i)\},\ \ \forall j \in J, \tag{8}$$

$$Y_{ih} \cdot (C_h - C_i) + L(1 - X_{hjk}) + L(1 - X_{ijk}) + L(1 - Z_{ih}) \ge Y_{ih} \cdot X_{hjk} \cdot t_{hk}^I,\ \ \forall i \in I,\ \ \forall h \in I \backslash \{P_a(i) \cup S_a(i)\},\ \ \forall j \in J, \tag{9}$$

$$(1 - Y_{ih}) \cdot (C_i - C_h) + L(1 - X_{ijk}) + L(1 - X_{hjk}) + L(1 - Z_{ih}) \ge (1 - Y_{ih}) \cdot t_{ik}^I,\ \ \forall i \in I,\ \ \forall h \in I \backslash \{P_a(i) \cup S_a(i)\},\ \ \forall j \in J,\ \ \forall k \in K, \tag{10}$$

$$(1 - Y_{ih}) \cdot (C_h - C_i) + L(1 - X_{hjk}) + L(1 - X_{ijk}) + L(1 - Z_{ih}) \ge (1 - Y_{ih}) \cdot t_{hk}^I,\ \ \forall i \in I,\ \ \forall h \in I \backslash \{P_a(i) \cup S_a(i)\},\ \ \forall j \in J,\ \ \forall k \in K, \tag{11}$$

$$C_i \le CW_j + L(1 - X_{ijk}),\ \ \forall i \in I,\ \ \forall j \in J, \tag{12}$$

$$CW_j \le Ct,\ \ \forall j \in J, \tag{13}$$

$$X_{ijk} - Nt \cdot W_{jk} \le 0,\ \ \forall j \in J,\ \ \forall k \in K, \tag{14}$$

$$1 \le W_{jk} \le N R_{\max},\ \ \forall j \in J, \tag{15}$$

$$\hat{v}_{ijk} \in \{0, 1\},\ \ \forall i \in I,\ \ \forall j \in J,\ \ \forall k \in K, \tag{16}$$

$$w_{jk} \in \{0, 1\},\ \ \forall j \in J,\ \ \forall k \in K, \tag{17}$$

$$Y_{ih} \in \{0, 1\},\ \ \forall i, h \in I, \tag{18}$$

$$Z_{ih} \in \{0, 1\},\ \ \forall i \in I, \forall h \in I \backslash \{P_a(i) \cup S_a(i)\}. \tag{19}$$

Constraint (5) guarantees that each task is assigned to only one robot. Constraint (6) ensures that predecessor of task i like h is assigned to the same workstation as task i or it is assigned to the prior workstation. Constraint (7) ensures that if task i and its predecessor h are assigned to the same workstation, the constraint becomes $C_i - C_h \ge t_{ik}^I$. If they are assigned to

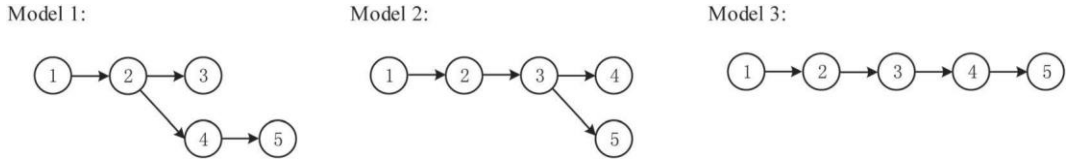

Figure 2. TPDs of three models.

different workstations, the constraint is always satisfied. Constraints (8) and (9) ensure that if tasks i and h (two tasks which are of the same model) which have no precedence relation are assigned to the same workstation, and if $Z_{ih} = 0$ (task h starts earlier than i), then (8) becomes $C_i - C_h \geq t_{ik}^{I}$ and (9) is always satisfied; otherwise (8) is always satisfied and (9) becomes $C_h - C_i \geq t_{hk}^{I}$. Constraints (10) and (11) ensure that if tasks i and h (two tasks which are of different models) are assigned to the same workstation, and if $Z_{ih} = 0$ (task h starts earlier than i), then (10) becomes $C_i - C_h \geq t_{ik}^{I}$ and (11) is always satisfied; otherwise (10) is always satisfied and (11) becomes $C_h - C_i \geq t_{hk}^{I}$. Constraint (12) determines the completion time of the whole tasks that are assigned to workstation j. Constraint (13) is the cycle time constraint. Constraint (14) ensures that if any task is assigned to robot k at workstation j, then W_{jk} must be equal to 1. Constraint (15) ensures that all the workstations are used and the number of robots assigned to each workstation is not more than $N\,R_{\max}$. Constraints (16)–(19) define the binary variables.

2.3. Approaches to deal with uncertain task processing times

The task processing times are assumed to be uncertain with unavailable probability distribution and membership functions due to some stochastic variability (Feng, Zheng, and Xu 2016). The only information is that processing time of task i by robot k lies in an interval $[t_{ik}^{L}, t_{ik}^{R}]$, where $0 \leq t_{ik}^{L} \leq t_{ik}^{R}$. Interval possibility degree describes the extent that an interval is greater than the other and it can be used to change the uncertain inequality with intervals to deterministic inequality (Jiang et al. 2008). Then constraints (7)–(11) can be transformed into deterministic constraints. For example, the constraint $C_i - C_h \geq t_{ik}^{I}$ can be transformed as Equation (20), where the value of γ is given between [0.5, 1]. The greater γ is, the more possible that the constraint is satisfied.

$$P(C_i - C_h \geq t_{ik}^{I}) \geq \gamma\,,\ \gamma \in [0.5, 1]. \tag{20}$$

Interval order relations determine whether an interval is superior to another one. Inspired by Han et al. (2016) who converted interval objectives into real-valued ones using a dynamically weighted sum of its midpoint and radius, we use the midpoint and radius values of the objective function to evaluate the performance between the different objective values. For example, objective function (3) can be transformed as follows:

$$f_2 : \min TE^{I} = \min(TE^{c}, TE^{w}), \tag{21}$$

where $\cdot^{c}$ and $\cdot^{w}$ are the midpoint and radius of an interval.

However, Equation (21) divides the single objective into two objectives. So a linear weighting method is used to further transform the objective function to Equation (22), where the value of the weight λ is given between $(0, 1)$.

$$\min TE^{I} = \lambda \cdot TE^{c} + (1 - \lambda) \cdot TE^{w},\ \lambda \in (0, 1). \tag{22}$$

2.4. Illustrative example

An example of MMDLBP is given in this section. TPDs of three models are presented in Figure 2, where the circles represent the tasks. Each model has five tasks with different task precedence. The operation energy consumption and standby energy consumption of robots per time unit are shown in Table 1. In this example, tasks of three models ($M = \{1, 2, 3\}$) are assigned to robots ($SetR = \{R1, R2, \ldots, R9\}$) at three multi-robotic workstations ($N_w = 3$) and the maximum number of robots allowed at each workstation is limited to two ($N\,R_{\max} = 2$).

Figure 3 gives two optimal solutions of the problem, where I_n^m refers to the task n of model m. The problem has been solved through the mathematical programming formulation in Section 2.2, and the optimal solutions are obtained by LINGO 11.0 software. The objective of solution (a) is to minimise cycle time and solution (b) minimises the cycle time and total energy consumption simultaneously. In Figure 3, the tasks assigned to each robot in each workstation are illustrated by white rectangles and the width of rectangle reflects the task time. Gray rectangles indicate idle times of robots. The task numbers are written inside the white rectangles and the index of each robot is shown on the left of the workstations.

Table 1. Operation energy consumption (OP) and standby energy consumption (SP) of nine robots per time unit.

Robot	R1	R2	R3	R4	R5	R6	R7	R8	R9
OP	1.0	1.5	2.0	2.5	3.0	3.5	4.0	4.5	5.0
SP	0.10	0.15	0.20	0.25	0.30	0.35	0.40	0.45	0.50

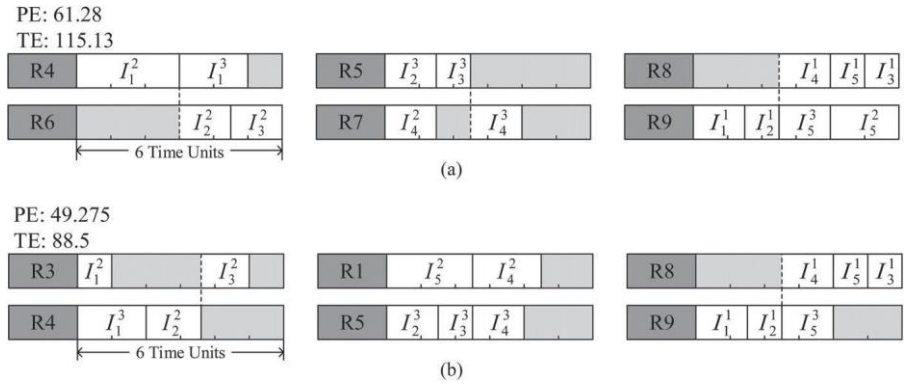

Figure 3. Two solutions for an example of MMDLBP.

As shown in Figure 3, each workstation shows six time units which correspond to the optimal cycle time while solution (b) has less total energy consumption than solution (a). In solution (b), robots are better used, and less energy is consumed. This instance shows the potential benefit of considering energy consumption on disassembly line, which increases the complexity of DLBP but has a practical significance to the design of disassembly line.

3. Evolutionary simulated annealing algorithm for MMDLBP

3.1. Procedure of PDSA-EA

The simulated annealing (SA) algorithm, which distinctly escapes from being trapped into local optima by accepting a worse solution with a probability function, was firstly designed by Kirkpatrick, Gelatt, and Veechi (1983). It starts with an initial solution which is regarded as the current solution. Then the neighbourhood solution of the current one is obtained by neighbourhood generation. If the new solution is better than the current one (evaluated using objective function), the new solution replaces the current one and best one. Otherwise, the new solution is accepted as the current one with a certain probability.

Multi-objective simulated annealing using Pareto-domination based acceptance criterion (PDMOSA), proposed by Suman (2004), defined a novel fitness: 1 + the number of solutions in Pareto set obtained so far dominating it. The worse solution has larger value of fitness. Without the calculation of multi-objective values, probability function becomes simpler which cuts much computational cost. The algorithm benefits from the feature that obtains a uniformly distributed solution in the trade-off surface. More specifically, in the early stage of the algorithm, the difference of fitness values between solu-tions would not be so much due to small size of current solution set and the high temperature, which makes it possible that any move is accepted so that the decision space is well explored. As the iteration increases, the size of current solution set expands and the temperature decreases, solutions that are worse than the current one are accepted with less possibility, mak-ing the move more selective. Consequently, the proposed algorithm converges to a set of well-diversified non-dominated solutions.

Evolutionary algorithm (EA), inspired by biological evolution such as reproduction, mutation, recombination and selec-tion, has been successfully applied to solve numerous optimisation problems for its global search ability. In this study, PDMOSA is hybridised with EA to optimise three objectives simultaneously. The core of the proposed approach is to com-bine the local search property of SA algorithm which ensures the diversity of the population by accepting a worse solution with a probability function, with the global search property of EA which ensures the convergence of algorithm.

As depicted in Algorithm 1, the proposed approach includes two populations (PC and PB) which are evolved and updated respectively. PC is the current solution set and PB is an archive storing non-dominated solutions. Each individual in PC is used

to generate a neighbouring solution independently, and each new solution will be either discarded or added into PC for the next iteration according to the Pareto-domination based acceptance criterion. PB is updated based on the Pareto dominance relationship.

Algorithm 1 . Framework of PDSA-EA algorithm.
Input: Data for MMDLBP; parameters of PDSA-EA (T0: the initial temperature, AT: the annealing times, the cooling rate

λ, L: the number of iterations at each temperature, N: the size of PC). Output: Non-dominated solutions. Begin:

Generate a set of initial solutions as PC;

Get PB from PC based on Pareto dominance relationship;
Set t = 0, l = 0, T = T0;
while (t < AT) do

t := t + 1;
while (l < L)do

l := l + 1;
for each solution Sc in solution set PC;
Get a new solution Sc using neighbourhood search strategy;
Update solution set PB by Sc based on Pareto dominance relationship;
Calculate f (Sc) and f (Sc) using Pareto-domination based acceptance criterion;
if (f (Sc) > f (Sc))
Sc = Sc;
else
[0, 1];
Generate a random number R ∼f(S) − f (Sc)
if (f (S) < f (S)) and EXP(− c) > R

c c kT
Sc = Sc;

if (f (Sc) = f (Sc)) andScis dominated bySc
Sc = Sc;
End while.

Add individuals of solution set PB (which do not exist in PC) to PC;

Keep the size of PC to N according to adaptive crowding distance;

End while.

End.

The procedure starts with generating solution set PC and solution set PB randomly using the encoding and decoding schemes. Then the main loop begins until the termination criterion is reached. The main loop consists of neighbourhood search strategy, application of Pareto-domination based acceptance criterion, population maintaining strategy based on adaptive crowding distance, and solution set updating.

3.2. Encoding and decoding

Three integer vectors are used for the encoding: robot-workstation assignment vector *RW*, task sequence vector *TS* and task-workstation assignment vector *TW*.

Before building a task sequence vector, the fixed global ID (GID) of each task is defined as a number in a set {1, 2, . . . , $N\,t^1$, $N\,t^1$ + 1, $N\,t^1$ + 2, . . . , $N\,t^1$ + $N\,t^2$, . . . }where $N\,t^m$ is the total number of tasks of model *m*. Note that task sequence vector *TS* only determines the order of assignment. Task-workstation assignment vector *TW* determines number of tasks assigned to each workstation and each element of *TW* is an integer between 1 and *Nw*. Moreover, there is a one-to-one mapping between the same index in *TS* and *TW*. That is to say, if the *d*th element in *TW* is *j* and the *d*th element in *TS* is *i*, task *i* will be assigned to workstation *j*. Robot-workstation assignment vector *RW* determines the types and number of robots assigned to each workstation. If the *r*th element of *RW* is *j*, the robot *r* will be assigned to workstation *j* (*j* = −1 means the robot is not used). Though the assignment of tasks and robots to workstations is determined, the detailed assignment of tasks to robots remains unsolved. Thus a robot-selection mechanism is utilised to decide which robot the current task should be assigned to. The following heuristic method is used for decoding.

(1) Set *j* = 0, as the counter for workstations.

(2) If *j* = *Nw*, then stop; else set *j* := *j* + 1.

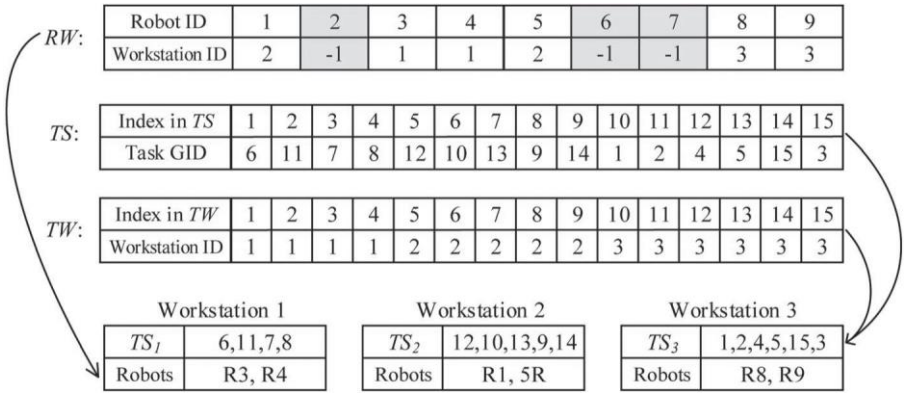

Figure 4. Solution representation for MMDLBP.

(3) Decide the number of tasks assigned to *j*th workstation according to *TW*, then the task sequence *TS* can be divided into TS_j. Allocate each task in TS_j to robots based on the order of occurrence in the TS_j.

(4) Decide the robots assigned to *j*th workstation according to *RW*. Set S_r = 0 as the starting time of robot *r*.

(5) Calculate the completion time of task *b* by each robot at *j*th workstation using Equation (23), and assign task *b* to the robot which has the minimum completion time (if tie occurs, select the robot with the less energy consumption)

$$C_b = \max\{S_r, C_{p(b)}\} + p_{br}, \tag{23}$$

where p_{br} is the processing time of task *b* by robot *r*. $C_{p(b)}$ is the completion time of the task, which is of the same model as task *b* and is nearest to task *b* in TS_j. S_r is the starting time of the task which is to be performed by robot *r*.

(6) If all tasks in TS_j has been assigned, go to (2); else go to (5) and select a robot for the next task in TS_j.

Figure 4 illustrates the encoding of solution (b) in Section 2.4. From *RW*, it is obvious that nine robots are available for three workstations and only six robots are used: R3 and R4 are assigned to the first workstation, R1 and R5 are assigned to the second workstation, and R8 and R9 are assigned to the third workstation. Fifteen tasks of three models are assigned to workstations according to *TS* and *TW*. Apparently each task is assigned to a robot by the proposed decoding scheme. More importantly, the decoding scheme is rigorous which means there is only one decoding result corresponding to one encoding.

3.3. Solution initialisation

To initialise a solution, reasonable values are assigned to *RW*, *TS* and *TW*. To generate a feasible *RW*, robots are randomly selected from the assignable robots list one by one and then the list is updated by deleting the selected robots. The number of tasks assigned to each workstation is randomly generated to determine *TW*. The details of generating *TS* are described in the following steps.

step 1 Generate TS^m (m = 1, 2, . . . , M)

(1) Set m = 0 as the counter for model.
(2) if $m = M$, then stop; else set $m := m + 1$.
(3) Determine the unassigned task set $SetUT^m$.
(4) Determine the assignable task set $SetAT^m$ from $SetUT^m$, of which all the tasks have no predecessor, or their predecessors have already been assigned.
(5) Add random sort of all the tasks in $SetAT^m$ to the end of TS^m, then initialise $SetAT^m$ as an empty set.
(6) If $SetU\ T^m = \phi$, go to (2); else go to (3).

step 2 Get *TS* from TS^m (m = 1, 2, . . . , M)

(1) Set s = 0 as the index of *TS*.
(2) If $s = Nt$, then stop; else set $s := s + 1$.
(3) Randomly select a model x in the list of model (1, 2, . . . , M). Put the first element of TS^x to the sth position of *TS* and remove this task in TS^x.
(4) If $T\ S^x = \phi$, delete the index of model x in the list of models. Go to (2).

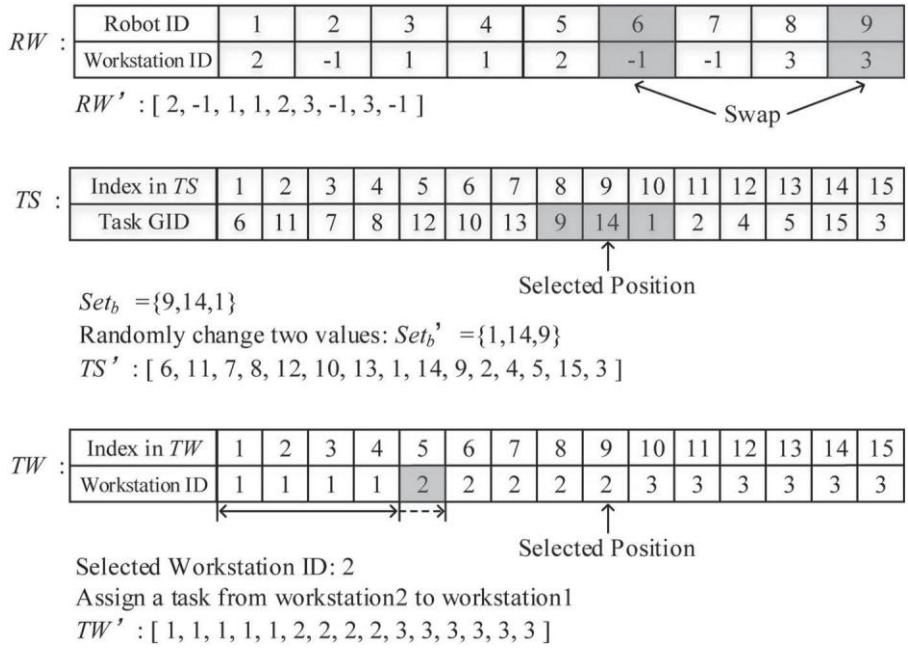

Figure 5. An example of neighbour structures.

Three vectors are randomly generated instead of using heuristic initialisation. To some extent, heuristic initialisation may accelerate the convergence but it also may drive down the diversity and distribution uniformity of initial solutions which would easily get stuck at local optimum and premature convergence.

3.4. Neighbourhood search strategy

Since three vectors in Section 3.2 are involved, different neighbour structures are utilised to obtain neighbour solutions. In this study, each neighbourhood generation operator is selected with the same possibility and all add up to 1. An example depicted in Figure 5 is to demonstrate the neighbourhood search strategy. Swap operator is used for *RW*. Firstly, randomly select a position of *RW*. Then exchange the value (workstation ID) with another randomly selected position, of which the value is not equal to the first. Since all usable robots are contained in the *RW*, including the robots which are not used, thus a robot being used is possible to exchange with an unused one. A dividing operator is used for *TW*. Firstly, randomly select a position of *TW* and get its workstation ID. Then remove a task from the selected workstation and allocate this task to one of its neighbouring workstations randomly.

Since the tasks are of different models, it will be easy to obtain infeasible solutions by swapping the values of two posi-tions of *TS* randomly or selecting a position and inserting its value to another random position. Considering the feasibility and diversity of the generated *TS*, a partial sequence swap operator is used as following.

(1) Randomly select a task (bth task in *TS*, where $1 \le b \le Nt$) as a starting-point. Add this task to an empty task set Set_b.

(2) Set $p := b - 1$, $s := b + 1$.

(3) Generate a random number r($r \in [0, 1]$). If $r \le 0.5$, execute (a) firstly then (b); else execute (a) then (b) by default.

(a) If $1 \le p \le Nt$ and pth task in *TS* is not the predecessor of bth task in *TS*, then add pth task to Set_b and set $p := p - 1$, then continue (a); else terminate (a).

(b) If $1 \le s \le Nt$ and sth task in *TS* is not the successor of bth task in *TS*, then add sth task to Set_b and set $s := s + 1$, then continue (b); else terminate (b).

(4) If the size of Set_b is 1, then go to (1); else randomly pick two positions of Set_b and change their values, then insert this part of task sequence to the corresponding position in *TS*.

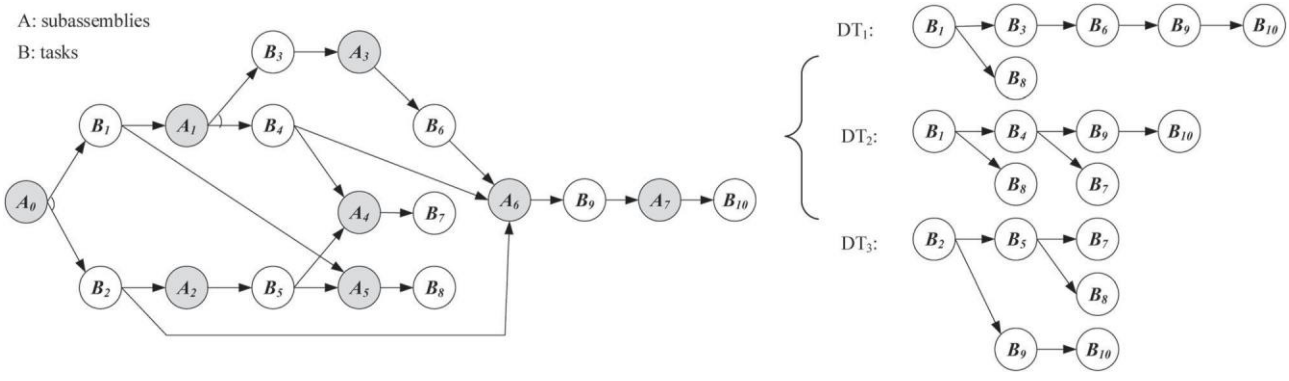

Figure 6. TAOG and DTs of a handlight.

3.5. Population update based on crowding distance

Crowding distance, defined in the fast elitist non-dominated sorting genetic algorithm (NSGA-II) (Deb et al. 2002), evaluates the individuals' density and is used to maintain diversity of population. After neighbourhood generation, if a new solution is not dominated by any solution in PB, then add it to PB and remove those dominated by it. As the iterative generation increases, the multiple objectives make it easy to obtain non-dominated solutions which will be added to PC. The size of PC expands, thus a population maintaining strategy based on adaptive crowding distance is utilised to keep the size of PC. The adaptive crowding distance D_i of ith solution is computed with Equations (24) and (25). Some solutions may have obvious difference on one objective but have less difference on other objectives, which results in the smaller value of crowding distance. So the variance of crowding distance is introduced to ensure that the solutions, which have more difference on objectives, have more chance to be preserved.

$$D_i = d_i \cdot EXP(S_i), \tag{24}$$

$$S_i = \frac{1}{m} \sum_{k=1}^{m} \left(f_k^{i+1} - f_k^{i-1} - d_i \right)^2, \tag{25}$$

where d_i is the crowding distance defined in NSGA-II, S_i is the variance of d_i along each of the objectives, m refers to the number of objectives, and f_k^i is the value of kth objective of ith solution. Note that all the values of objectives have been normalised and sorted before calculating the crowding distance.

4. Experimental study

4.1. Data set

First of all, the generation of problem instances need to be considered. To the best knowledge of the authors, this paper is the first one exploring the uncertain MMDLBP with interval task processing times. Since there is no standard data set available for the considered problem, nine tested problem instances are generated based on the disassembly benchmark instances that have been used before, such as the Ballpoint (Lambert 2007), Sample product (Koc, Sabuncuoglu, and Erel 2009), Radio (Lambert 1999), Automatic pencil (Ma et al. 2011), Ball-point pen (Lambert 1999), and Copy machine (Lambert 2007).

The transformed AND/OR graph (TAOG) (Koc, Sabuncuoglu, and Erel 2009) with practical cases has been widely used in DLBP recently. Inspired by Yagmahan (2011) who used the same precedence relation but different tasks times to get mixed models from a single model, in this paper, it is assumed that disassembly trees (DTs) with different disassembly structures and the same number of tasks can be used to represent task precedence relations of different models. Figure 6 shows TAOG of a handlight (Tang et al. 2002) which contains three DTs and these DTs are used as three TPDs of different models.

The nine problem instances are generated including three small-sized instances, P1, P2 and P3, three medium-sized instances, P4, P5 and P6, and three large-sized instances, P7, P8 and P9. The details of the nine problem instances are shown in Table 2. There are 30 robots considered in our experimental studies. The initial processing times of tasks by robots are randomly generated between $[T_k \times 0.5, T_k \times 1.5]$ where T_k is the base value randomly generated between [10, 30]. The lower bound of interval time is the initial value multiplied by 0.9 while 1.1 for upper bound. The operation energy consumptions of robots per time unit are randomly generated between [5, 10] and the standby energy consumption of a

Table 2. Descriptions of nine generated problem instances in data set.

Instance	Number of models	Number of total tasks	Scale	NR_{max}	Nw	Product
P1	3	15	small	2	2	Ballpoint (Lambert 2007)
P2	3	18	small	2	3	Sample product (Koc, Sabuncuoglu, and Erel 2009)
P3	2	18	small	3	2	Radio (Lambert 1999)
P4	6	60	medium	3	3	Automatic pencil (Ma et al. 2011)
P5	9	75	medium	3	3	Products from P1 and P4
P6	9	81	medium	3	4	Ball-point pen (Lambert 1999)
P7	15	114	large	4	3	Products from P1, P2 and P6
P8	14	126	large	4	4	Copy machine (Lambert 2007)
P9	15	141	large	4	5	Products from P4 and P6

robot per time unit is equal to 10% of the operation energy consumption (Nilakantan et al. 2017). For space limitation, the generated data including task precedence relationship data, interval task times and energy consumption data are exhibited in supplementary part of this paper.

4.2. Parameter settings and evaluation metrics

To evaluate the performance of the proposed algorithm, the restarted simulated annealing (RSA) algorithm (Li, Tang, and Zhang 2016) and the widely-used NSGA-II are re-implemented for the algorithm comparative study. All the algorithms and experiments were coded in Java programming language and tested on Windows 7 configured with JDK 1.8. The computer is equipped with Intel(R) Core(TM) CPU 3.00 GHz and 8.00 GB RAM. All the results presented in the experimental study are obtained by executing 20 independent runs for each algorithm. To make the tests reasonable, each algorithm is terminated when the number of newly generated solutions reaches 150 thousand times. All the parameters of the three algorithms have been determined experimentally and set to be suitable for each scale of the problems above. For PDSA-EA, the initial temperature, the annealing time, the cooling rate, the number of iterations under each temperature level, and the size of PC population were specified as 25.0, 300, 0.985, 10, and 50, respectively. For RSA, the annealing time, the number of iterations under each temperature level, and the number of moves before start were set to 300, 500 and 50. For NSGA-II, the population size was set to 150 and the maximum number of iterations to 1000.

$$RPF(P_S) = \frac{r_s}{} - \{ X \in P_s \mid \forall Y \in PF : Y \xrightarrow{Pareto\ dominant} X \} \mid / \mid PF \mid \quad (26)$$

Four performance metrics are used to compare the results. Since each algorithm terminates under a fixed number of gen-eration, the shorter execution time (ET) and better performance on each metric indicates higher efficiency of algorithm. ET shows the computational times (CPU times) consumed by an algorithm. The ratio of Pareto front (RPF) of solution set P_S is formulated with Equation (26), where PF is the Pareto front, and $|*|$ is the number of solutions in set $*$. For the non-dominated set P_S, only a part of solutions can reach the Pareto front. So a higher value of RPF indicates better quality of the obtained solutions. In addition, two commonly used indicators, convergence of the Pareto-optimal set (CP) (Li, Tang, and Zhang 2016) and Hypervolume (HV) (Zitzler and Thiele 1999), are introduced here to provide a combined information of convergence and diversity of the obtained solution sets. For the solution sets being of the same convergence, CP, which is calculated on the basis of uniformly distributed points along the Pareto front, prefers the one having uniformly distributed individuals, while HV, which calculates the volume of the objective space between the obtained solution set and a reference point, has a bias towards the one having good extensity.

Note that the actual Pareto fronts of the tested problem instances are not known. Hence, for each problem instance, the approximated Pareto front is formed by gathering all non-dominated solutions found by all the implemented algorithms in all runs. In addition, we normalised the objective values of the obtained solutions according to the range of the approximated Pareto front and calculated the performance metrics based on the normalised values.

4.3. Performance comparison among algorithms

Figure 7(a), (b) and (c) show the normalised non-dominated solution sets obtain by the three algorithms, NSGA-II, RSA and PDSA-EA, on one small-sized instance, P1, one medium-sized instance, P4, and one large-sized instance, P8, respectively. Note that each algorithm can obtain many solutions within twenty independent runs, and we only give the normalised non-dominated solutions to have a better visualisation. For each problem instance, normalised non-dominated solution sets

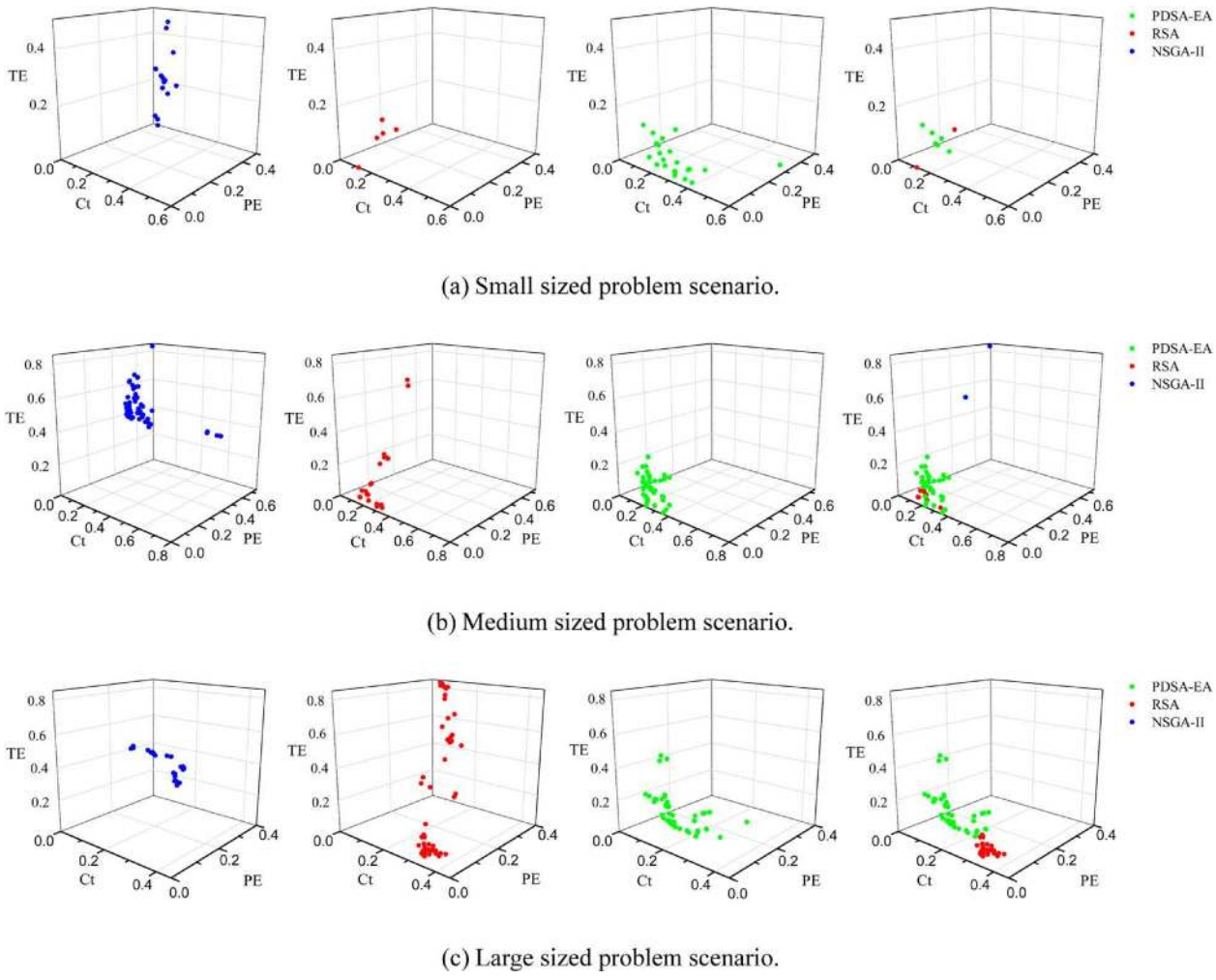

Figure 7. Normalised non-dominated solution sets obtained by the three algorithms on different problem scenarios, as shown by scatter plots (Ct: cycle time; PE: peak workstation energy consumption; TE: total energy consumption).

Table 3. The best, mean and worst objective values obtained by the three algorithms for different scale problems.

			Cycle time			Peak workstation energy consumption			Total energy consumption		
Scale	Algorithm	*N*	Best	Mean	Worst	Best	Mean	Worst	Best	Mean	Worst
Small-	NSGA-II	0	54.41	60.93	66.16	858.48	966.93	1090.94	1617.14	1790.21	2045.34
sized	RSA	2	50.07	**53.32**	54.60	716.19	811.13	872.78	1404.50	1503.30	1554.96
problem	PDSA-EA	**7**	48.69	59.95	76.08	719.04	**783.13**	971.72	1410.67	**1455.71**	1550.63
Medium-	NSGA-II	2	121.01	140.76	191.92	2791.58	2977.41	3565.67	7966.37	8279.14	9319.73
sized	RSA	10	127.31	134.95	147.08	2279.58	2484.27	3039.21	6674.02	7067.38	8735.55
problem	PDSA-EA	**41**	123.69	**134.51**	152.13	2263.50	**2434.39**	2710.76	6686.37	**6873.74**	7298.88
Large-	NSGA-II	0	144.72	155.90	164.78	4354.47	4546.58	4799.84	16720.67	16978.68	17320.71
sized	RSA	27	144.45	157.98	174.33	3964.26	4413.95	4932.01	15234.90	16591.73	18942.83
problem	PDSA-EA	**46**	137.39	**151.54**	166.02	4012.74	**4183.33**	4505.47	15619.50	**15911.82**	16998.92

obtained by the three algorithms are depicted in three colours respectively on the left and the Pareto front is depicted on the right. Apparently, for either the convergence or the diversity of solution set, PDSA-EA outperforms RSA and NSGA-II in whichever scale of the problems. Most solutions by RSA and NSGA-II are dominated by the non-dominated solution set by PDSA-EA for each problem instance. In addition, as the scale of the problems increases, NSGA-II shows good performance in convergence while RSA shows great ability of exploring the decision space and can easily find the boundary solutions.

Table 4. RPF, CP, HV and ET results of the three algorithms for the nine problem instances.

Instance	Algorithm	ET (s)	RPF	CP (mean [standard deviation])	HV (mean [standard deviation])
P1	NSGA-II	23.362	0.000	6.363E-1 [6.128E-2]	2.593E-1 [2.352E-2]
	RSA	**0.658**	0.250	1.553E-1 [2.372E-3]	8.560E-1 [2.354E-3]
	PDSA-EA	1.212	**0.875**	**1.277E-1 [4.800E-4]**	**9.333E-1 [2.161E-4]**
P2	NSGA-II	23.761	0.000	6.398E-1 [6.727E-2]	2.364E-1 [1.976E-2]
	RSA	**0.879**	0.000	1.830E-1 [8.404E-3]	7.295E-1 [1.072E-2]
	PDSA-EA	2.095	**1.000**	**4.792E-2 [2.448E-4]**	**9.316E-1 [1.019E-3]**
P3	NSGA-II	23.611	0.000	5.268E-1 [2.677E-2]	1.277E-1 [8.399E-3]
	RSA	**0.868**	0.056	2.300E-1 [1.574E-2]	6.920E-1 [2.335E-2]
	PDSA-EA	1.886	**1.000**	**3.803E-2 [3.080E-4]**	**9.933E-1 [3.014E-4]**
P4	NSGA-II	38.943	0.038	2.801E-1 [1.709E-2]	1.580E-1 [6.687E-3]
	RSA	**3.787**	0.189	2.710E-1 [1.963E-2]	5.820E-1 [2.448E-2]
	PDSA-EA	7.871	**0.774**	**1.079E-1 [4.505E-3]**	**7.701E-1 [9.758E-3]**
P5	NSGA-II	49.403	0.000	7.698E-1 [5.885E-2]	1.461E-1 [1.135E-2]
	RSA	**4.231**	0.345	2.630E-1 [1.471E-2]	6.422E-1 [1.315E-2]
	PDSA-EA	8.594	**0.971**	**1.049E-1 [4.132E-3]**	**7.697E-1 [9.826E-3]**
P6	NSGA-II	59.516	0.182	2.643E-1 [9.837E-3]	2.314E-1 [7.210E-3]
	RSA	**5.800**	0.364	2.422E-1 [1.590E-2]	5.801E-1 [2.004E-2]
	PDSA-EA	12.989	**0.455**	**1.385E-1 [4.359E-3]**	**6.715E-1 [4.677E-3]**
P7	NSGA-II	72.215	0.000	4.046E-1 [2.585E-2]	1.279E-1 [7.798E-3]
	RSA	**12.954**	**0.735**	2.053E-1 [1.412E-2]	6.586E-1 [2.541E-2]
	PDSA-EA	19.942	0.324	**1.374E-1 [3.925E-3]**	**6.950E-1 [8.528E-3]**
P8	NSGA-II	65.642	0.000	4.110E-1 [1.831E-2]	2.496E-1 [1.211E-2]
	RSA	**12.885**	0.370	2.853E-1 [1.377E-2]	5.333E-1 [1.732E-2]
	PDSA-EA	18.141	**0.630**	**1.409E-1 [6.422E-3]**	**6.332E-1 [1.162E-2]**
P9	NSGA-II	70.241	0.000	6.197E-1 [3.159E-2]	1.393E-1 [6.679E-3]
	RSA	**14.359**	0.448	2.282E-1 [1.988E-2]	5.884E-1 [1.615E-2]
	PDSA-EA	24.403	**0.891**	**9.859E-2 [2.395E-3]**	**6.304E-1 [8.300E-3]**

Note: ET (s): execution time (seconds); RPF: ratio of Pareto front; CP: convergence of Pareto set; HV: hypervolume.

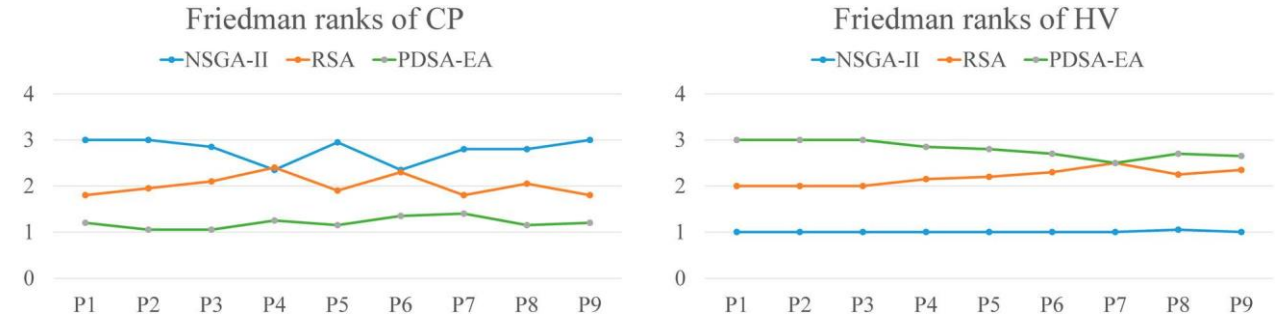

Figure 8. The Friedman ranks of CP and HV.

To further describe the distribution of solutions in Figure 7, the best, mean and worst objective values obtained by the three algorithms for the three problem instances, P1, P4 and P8, are listed as shown in Table 3. The three objectives are cycle time, peak workstation energy consumption, and total energy consumption. Note that the symbol 'N' in Table 3 denotes the number of solutions obtained by each algorithm in Pareto front of each problem instance. The better value of N and the better mean of each objective for each problem instance are highlighted in boldface. In Table 3, it is observed that PDSA-EA outperforms RSA and NSGA-II regarding to the mean energy consumption including peak workstation energy consumption and total energy consumption for all the problem instances, and the PDSA-EA outperforms its competitors in terms of the mean cycle time for medium and large-sized problem instances. In addition, the solutions in Pareto front of each problem instance are overwhelmingly obtained by PDSA-EA. The above results prove that the proposed PDSA-EA can reduce cycle time, peak workstation energy consumption and total energy consumption simultaneously and effectively.

Since multi-objective optimisation instead of single-objective optimisation is considered in this paper, some comprehensive metrics of Pareto-based algorithms for multi-objective optimisation such as RPF, CP and HV need to be evaluated to further test the performance of the proposed PDSA-EA. Hence, the RPF, CP, HV and ET results of the three algorithms for all nine problem instances are shown in Table 4. The better mean for each problem instance is highlighted in boldface.

From Table 4, it is observed that RSA and PDSA-EA show better performance on ET. As the scale of the problem increases, RSA and PDSA-EA consume less computational times than NSGA-II to reach solutions. For most problems in Table 4, PDSA-EA provides excellent performance on RPF, which indicates that PDSA-EA can reach to the Pareto front more easily than NSGA-II and RSA. As for the metrics CP and HV, which reflect the performance of the obtained solution set in terms of convergence and extensity, PDSA-EA shows better values than NSGA-II and RSA. By arranging each set of values of the variate in order of size, numbering them 1, 2, and so forth, then adding the numbers up, the Friedman ranks (Friedman 1937) are obtained. Here Friedman ranks were used to illustrate the quality of the metrics CP and HV. Figure 8 shows the average Friedman ranks of CP and HV which demonstrates PDSA-EA performs more stable on the quality of the solutions than RSA and NSGA-II.

5. Conclusion

MMDLBP under uncertainty, in which different models of EOL products are disassembled simultaneously at multi-robotic workstations and the disassembly task times are uncertain, has been addressed in this paper. A mathematical model for MMDLBP with interval task processing time is provided, in which the cycle time, the peak workstation energy consump-tion and total energy consumption are minimised simultaneously, and an illustrative example is given to demonstrate this model. Intervals are considered to represent the uncertain task times, and interval possibility degree and order relation between intervals are introduced to compare uncertain objectives and transform uncertain constraints with intervals to the common inequalities. Since MMDLBP falls into NP-hard, a hybrid evolutionary simulated annealing algorithm using Pareto-domination based acceptance criterion called PDSA-EA is developed to obtain Pareto-optimal solutions. In order to evaluate the proposed algorithm, PDSA-EA is compared with RSA and NSGA-II on small-, middle- and large-sized problem instances. The experimental results indicate that PDSA-EA outperforms RSA and NSGA-II on different evaluation metrics.

For further studies, more realistic problems such as the route planning, the recognition and classification of disassem-bly robots can be considered into the objective functions or constraints. Moreover, mathematical approaches to deal with uncertainty in MMDLBP will also be a good starting point for future work.

Acknowledgements

The authors wish to thank the editor and the anonymous referees for their comments which have helped to improve this paper.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This work was supported by Engineering and Physical Sciences Research Council [Grant Number EP/N018524/1]; National Natural Science Foundation of China [Grant Number 51475347]; Hubei Provincial Natural Science Foundation [Grant Number 2018CFB094].

References

Agrawal, S., and M. K. Tiwari. 2008. “A Collaborative ant Colony Algorithm to Stochastic Mixed-Model U-Shaped Disassembly Line Balancing and Sequencing Problem.” *International Journal of Production Research* 46 (6): 1405–1429.

Altekin, F. T. 2017. “A Comparison of Piecewise Linear Programming Formulations for Stochastic Disassembly Line Balancing.” *International Journal of Production Research* 55 (24): 7412–7434.

Altekin, F. T., Z. P. Bayındır, and V. Gümü¸skaya. 2016. “Remedial Actions for Disassembly Lines with Stochastic Task Times.” *Computers and Industrial Engineering* 99: 78–96.

Aydemir-Karadag, A., and O. Turkbey. 2013. “Multi-objective Optimization of Stochastic Disassembly Line Balancing with Station Paralleling.” *Computers and Industrial Engineering* 65 (3): 413–425.

Battaïa, O., and A. Dolgui. 2013. “A Taxonomy of Line Balancing Problems and Their Solution Approaches.” *International Journal of Production Economics* 142 (2): 259–277.

Bentaha, M. L., O. Battaïa, and A. Dolgui. 2014. "A Sample Average Approximation Method for Disassembly Line Balancing Problem Under Uncertainty." *Computers and Operations Research* 51: 111–122.

Bentaha, M. L., O. Battaïa, and A. Dolgui. 2015b. "An Exact Solution Approach for Disassembly Line Balancing Problem Under Uncertainty of the Task Processing Times." *International Journal of Production Research* 53 (6): 1807–1818.

Bentaha, M. L., O. Battaïa, A. Dolgui, and S. J. Hu. 2015c. "Second Order Conic Approximation for Disassembly Line Design with Joint Probabilistic Constraints." *European Journal of Operational Research* 247 (3): 957–967.

Bentaha, M. L., A. Dolgui, and O. Battaïa. 2015a. "A Bibliographic Review of Production Line Design and Balancing Under Uncertainty." *IFAC-PapersOnLine* 48 (3): 70–75.

Bentaha, M. L., A. Dolgui, O. Battaïa, R. J. Riggs, and J. Hu. 2018. "Profit-oriented Partial Disassembly Line Design: Dealing with Hazardous Parts and Task Processing Times Uncertainty." *International Journal of Production Research* 56 (24): 7220–7242.

Deb, K., A. Pratap, S. Agarwal, and T. Meyarivan. 2002. "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II." *IEEE Transactions on Evolutionary Computation* 6 (2): 182–197.

Ding, L. P., Y. X. Feng, J. R. Tan, and Y. C. Gao. 2010. "A new Multi-Objective ant Colony Algorithm for Solving the Disassembly Line Balancing Problem." *International Journal of Advanced Manufacturing Technology* 48 (5–8): 761–771.

Fang, Y., Q. Liu, M. Li, Y. Laili, and D. T. Pham. 2019. "Evolutionary Many-Objective Optimization for Mixed-Model Disassembly Line Balancing with Multi-Robotic Workstations." *European Journal of Operational Research* 276 (1): 160–174.

Fattahi, P., A. Roshani, and A. Roshani. 2011. "A Mathematical Model and ant Colony Algorithm for Multi-Manned Assembly Line Balancing Problem." *The International Journal of Advanced Manufacturing Technology* 53 (1): 363–378.

Feng, X., F. Zheng, and Y. Xu. 2016. "Robust Scheduling of a Two-Stage Hybrid Flow Shop with Uncertain Interval Processing Times." *International Journal of Production Research* 54 (12): 3706–3717.

Friedman, M. 1937. "The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance." *Journal of the American Statistical Association* 32: 675–701.

Gao, Y., Q. Wang, Y. Feng, H. Zheng, B. Zheng, and J. Tan. 2018. "An Energy-Saving Optimization Method of Dynamic Scheduling for Disassembly Line." *Energies* 11 (5): 1261–1278.

Güngör, A., and S. M. Gupta. 1999. "Disassembly Line Balancing." *Proceedings of the 1999 Annual Meeting of the Northeast Decision Sciences Institute*, Newport, Rhode Island: 193–195.

Han, Y., D. Gong, Y. Jin, and Q. Pan. 2016. "Evolutionary Multi-Objective Blocking Lot-Streaming Flow Shop Scheduling with Interval Processing Time." *Applied Soft Computing* 42: 229–245.

Hezer, S., and Y. Kara. 2015. "A Network-Based Shortest Route Model for Parallel Disassembly Line Balancing Problem." *International Journal of Production Research* 53 (6): 1849–1865.

Ilgin, M. A., H. Akcay, and C. Araz. 2017. "Disassembly Line Balancing Using Linear Physical Programming." *International Journal of Production Research* 55 (20): 6108–6119.

Jiang, C., X. Han, G. R. Liu, and G. P. Liu. 2008. "A Nonlinear Interval Number Programming Method for Uncertain Optimization Problems." *European Journal of Operational Research* 188 (1): 1–13.

Jiang, H., J. J. Yi, S. L. Chen, and X. M. Zhu. 2016. "A Multi-Objective Algorithm for Task Scheduling and Resource Allocation in Cloud-Based Disassembly." *Journal of Manufacturing Systems* 41: 239–255.

Kalayci, C. B., and S. M. Gupta. 2013a. "Artificial Bee Colony Algorithm for Solving Sequence-Dependent Disassembly Line Balancing Problem." *Expert Systems with Applications* 40 (18): 7231–7241.

Kalayci, C. B., and S. M. Gupta. 2013b. "Simulated Annealing Algorithm for Solving Sequence-Dependent Disassembly Line Balancing Problem." *IFAC Proceedings Volumes* 46 (9): 93–98.

Kalayci, C. B., A. Hancilar, A. Güngör, and S. M. Gupta. 2015. "Multi-Objective Fuzzy Disassembly Line Balancing Using a Hybrid Discrete Artificial Bee Colony Algorithm." *Journal of Manufacturing Systems* 37: 672–682.

Kalayci, C. B., O. Polat, and S. M. Gupta. 2016. "A Hybrid Genetic Algorithm for Sequence-Dependent Disassembly Line Balancing Problem." *Annals of Operations Research* 242 (2): 321–354.

Kirkpatrick, S., C. D. Gelatt, and M. P. Veechi. 1983. "Optimization by Simulated Annealing." *Science* 220: 671–680.

Koc, A., I. Sabuncuoglu, and E. Erel. 2009. "Two Exact Formulations for Disassembly Line Balancing Problems with Task Precedence Diagram Construction Using and AND/OR Graph." *IIE Transactions* 41 (10): 866–881.

Lambert, A. J. D. 1999. "Linear Programming in Disassembly/Clustering Sequence Generation." *Computers and Industrial Engineering* 36 (4): 723–738.

Lambert, A. J. D. 2007. "Optimizing Disassembly Processes Subjected to Sequence-Dependent Cost." *Computers and Industrial Engineering* 34 (2): 536–551.

Li, Z. X., Q. H. Tang, and L. P. Zhang. 2016. "Minimizing Energy Consumption and Cycle Time in Two-Sided Robotic Assembly Line Systems Using Restarted Simulated Annealing Algorithm." *Journal of Cleaner Production* 135: 508–522.

Lu, C. C., S. W. Lin, and K. C. Ying. 2014. "Minimizing Worst-Case Regret of Makespan on a Single Machine with Uncertain Processing and Setup Times." *Applied Soft Computing* 23: 144–151.

Ma, Y. S., H. B. Jun, H. W. Kim, and D. H. Lee. 2011. “Disassembly Process Planning Algorithms for End-of-Life Product Recovery and Environmentally Conscious Disposal.” *International Journal of Production Research* 49 (23): 7007–7027.

McGovern, S. M., and S. M. Gupta. 2007a. “Combinatorial Optimization Analysis of the Unary NP-Complete Disassembly Line Balancing Problem.” *International Journal of Production Research* 45 (18-19): 4485–4511.

McGovern, S. M., and S. M. Gupta. 2007b. “A Balancing Method and Genetic Algorithm for Disassembly Line Balancing.” *European Journal of Operational Research* 179 (3): 692–708.

Mete, S., Z. A. Çil, E. Özceylan, K. Agpak,˘ and O. Battaïa. 2018. “An Optimisation Support for the Design of Hybrid Production Lines Including Assembly and Disassembly Tasks.” *International Journal of Production Research* 56 (24): 7375–7389.

Nilakantan, J. M., Z. X. Li, Q. H. Tang, and P. Nielsen. 2017. “Multi-objective Cooperative co-Evolutionary Algorithm for Minimizing Carbon Footprint and Maximizing Line Efficiency in Robotic Assembly Line Systems.” *Journal of Cleaner Production* 156: 124–136.

Özceylan, E., C. B. Kalayci, A. Güngör, and S. M. Gupta. 2018. “Disassembly Line Balancing Problem: A Review of the State of the art and Future Directions.” *International Journal of Production Research* 1–23. doi: 10.1080/00207543.2018.1428775.

Özceylan, E., and T. Paksoy. 2014. “Interactive Fuzzy Programming Approaches to the Strategic and Tactical Planning of a Closed-Loop Supply Chain Under Uncertainty.” *International Journal of Production Research* 52 (8): 2363–2387.

Paksoy, T., A. Güngör, E. Özceylan, and A. Hancilar. 2013. “Mixed Model Disassembly Line Balancing Problem with Fuzzy Goals.” *International Journal of Production Research* 51 (20): 6082–6096.

Pistolesi, F., B. Lazzerini, M. D. Mura, and G. Dini. 2018. “EMOGA: A Hybrid Genetic Algorithm with Extremal Optimization Core for Multiobjective Disassembly Line Balancing.” *IEEE Transactions on Industrial Informatics* 14 (3): 1089–1098.

Riggs, R. J., O. Battaïa, and S. J. Hu. 2015. “Disassembly Line Balancing Under High Variety of End of Life States Using a Joint Precedence Graph Approach.” *Journal of Manufacturing Systems* 37: 638–648.

Seidi, M., and S. Saghari. 2016. “The Balancing of Disassembly Line of Automobile Engine Using Genetic Algorithm (GA) in Fuzzy Environment.” *Industrial Engineering and Management Systems* 15 (4): 364–373.

Sotskov, Y. N., T.-C. Lai, and F. Werner. 2013. “Measures of Problem Uncertainty for Scheduling with Interval Processing Times.” *OR Spectrum* 35 (3): 659–689.

Suman, B. 2004. “Study of Simulated Annealing Based Algorithms for Multiobjective Optimization of a Constrained Problem.” *Computers and Chemical Engineering* 28 (9): 1849–1871.

Tang, Y., M. C. Zhou, E. Zussman, and R. Caudill. 2002. “Disassembly Modeling, Planning and Application.” *Journal of Manufacturing Systems* 21 (3): 200–217.

Vongbunyong, S., and W. H. Chen. 2015. *Disassembly Automation - Automated Systems with Cognitive Abilities (Series: Sustainable Production, Life Cycle Engineering and Management)*. Berlin: Springer.

Wang, K., X. Li, and L. Gao. 2019. “Modeling and Optimization of Multi-Objective Partial Disassembly Line Balancing Problem Considering Hazard and Profit.” *Journal of Cleaner Production* 211: 115–133.

Yagmahan, B. 2011. “Mixed-model Assembly Line Balancing Using a Multi-Objective ant Colony Optimization Approach.” *Expert Systems with Applications* 38 (10): 12453–12461.

Zhang, Z., K. Wang, L. Zhu, and Y. Wang. 2017. “A Pareto Improved Artificial Fish Swarm Algorithm for Solving a Multi-Objective Fuzzy Disassembly Line Balancing Problem.” *Expert Systems with Applications* 86 (1): 165–176.

Zheng, F., J. He, F. Chu, and M. Liu. 2018. “A New Distribution-Free Model for Disassembly Line Balancing Problem with Stochastic Task Processing Times.” *International Journal of Production Research* 56 (24): 7341–7353.

Zhu, L., Z. Zhang, and Y. Wang. 2018. “A Pareto Firefly Algorithm for Multi-Objective Disassembly Line Balancing Problems with Hazard Evaluation.” *International Journal of Production Research* 56 (24): 7354–7374.

Zitzler, E., and L. Thiele. 1999. “Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach.” *IEEE Transactions on Evolutionary Computation* 3 (4): 257–271.