An integrated approach for a new flexible multi-product disassembly line balancing problem

Peng Hu, Feng Chu, Ming Liu, Shijin Wang, Peng Wu

An integrated approach for a new flexible multi-product disassembly line balancing problem

Peng Hu[a,b], Feng Chu[a], Ming Liu[c], Shijin Wang[c], Peng Wu[b]

[a] *Laboratoire IBISC, Univ Évry, Université Paris-Saclay, Évry, France*
[b] *School of Economics & Management, Fuzhou University, Fuzhou, Peoples Republic of China*
[c] *School of Economics & Management, Tongji University, Shanghai, Peoples Republic of China*

Abstract

Flexible disassembly line design for end-of-life (EOL) products is a key issue in the remanufacturing industry. However, existing studies for disassembly line balancing have not simultaneously considered multiple EOL products, the identical parts of these products and uncertainty during disassembly, which are important characteristics of flexible disassembly lines. The present study addresses a new flexible multi-product disassembly line balancing problem in which 1) disassembly schemes need to be selected, 2) a workstation can disassemble multiple EOL products, 3) identical parts of multiple products can be treated as identical tasks, and 4) only partial probability distribution information of processing times is known. For the problem, an integrated approach is developed, which is composed of a chance-constrained program, a distribution-free model, efficient valid inequalities and an exact lifted cut-and-solve method. Numerical experiments are conducted on an illustrative example, 10 instances based on realistic products and 480 randomly generated instances with up to 20 products, 400 tasks and 86 workstations. Computational results show that the proposed valid inequalities can reduce about 75% computational time of the original model, and the lifted cut-and-solve method needs only 17.53% and 40.65% of the computational times required by the CPLEX and the classic cut-and-solve method, respectively.

Keywords: Disassembly line balancing, Multi EOL product, Chance-constrained programming, Valid inequalities, Cut-and-solve method

1. Introduction

Sustainable development has received extensive attention in recent years owing to the wide awareness of saving non-renewable resources and protecting the environment.

Email address: hupeng723@gmail.com (Peng Hu)

As an important branch of sustainable industries, the remanufacturing industry aims to create new economic, social, and environmental values by appropriately managing end-of-life (EOL) products. Disassembly is an important process in remanufacturing industries, and it widely exists in many recovery industries. Many vehicle companies, such as BMW, have disassembled EOL vehicles to remanufacture high-value components such as engines, starter motors, and alternators for many years (Thierry et al., 1995). Some electronic manufacturers, such as Apple and Dell, have set up disassembly lines for remanufacturing the multiple EOL electric products (Xu et al., 2019; Ndubisi et al., 2020).

Increasing demand for customized products results in various new products, and the scale of EOL products and their variants is rapidly expanding in the recycling market. Thus, the traditional single-product disassembly line is inappropriate and uneconomical to disassemble such increasing EOL product variants. In the latest review, Özceylan et al. (2019) point out that the researchers are expected to concentrate more on the consideration of multi-product to be disassembled in the future. Moreover, with the increasing variety of end-of-life products entering the recycling streams, Paksoy et al. (2013) state that investigating multi-product disassembly lines can offer better insight into practical disassembly issues. In addition, Fang et al. (2019) point out that multi-product line satisfies variant disassembly demand and reduces line building and maintaining costs. In the real application, Apple developed a disassembly line called Daisy to dismantle 9 iPhone variants and recycle high-quality components and metal materials in 2016 (N-dubisi et al., 2020). These multiple EOL products may have identical components that can be disassembled by the same machine or workstation to save material resources and improve disassembly line performance. For example, the Engine Electronic Control Unit (ECU) is one of the most valuable electronic devices of EOL vehicles (Cucchiella et al., 2016). In Figure 1, there are two types of ECUs (A and B). They have 4 identical parts, i.e., front plastic casing (1), left metal piece (2), right metal piece (3), rear metal casing (4), and different printed circuit boards (5 or 6). Because of the complicated and uncertain disassembly environments for EOL products and their variants, the designing of flexible disassembly lines that simultaneously consider multiple products, their identical parts and uncertainty during disassembly is a great challenge, although disassembly line balancing problems have been studied for two decades.

Approximately 96% of related studies in the literature concentrate on the single-product disassembly line balancing problem (DLBP) according to Özceylan et al. (2019). Multi-product DLBPs, also called mixed-model DLBP in most existing studies, have been investigated for the last 10 years. A mixed-model disassembly line means a disas-

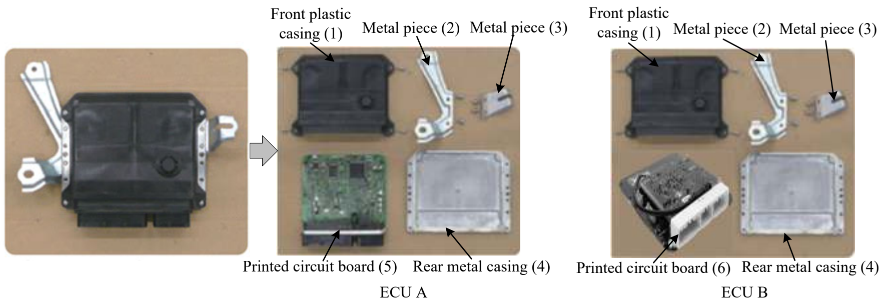


Figure 1: The example of two Toyota Vitz ECUs (Environmental Affairs Division, 2017)

sembly line that can handle more than one type of EOL product (Paksoy et al., 2013). Some of these studies examine deterministic multi-product DLBPs (Ilgin et al., 2017; Fang et al., 2019). Some works investigate multi-product DLBPs with uncertainty. Agrawal et al. (2008) study a stochastic mixed-model DLBP and sequencing problem, in which the uncertain task processing time follows a normal distribution, but disassembly scheme selection is ignored. Fang et al. (2020a) address a stochastic mixed-model DLBP with parallel robots in each workstation, in which only the interval of task processing time is given, and the disassembly scheme is predetermined. Liu et al. (2022) investigate a stochastic multi-product DLBP with limited distributional information of task processing time. Compared with our study, 1) They assume that one workstation can only dismantle one type of product, while one workstation can dismantle multiple products in our study. 2) They assume that the first two moments of task processing times are known, while in our study, the mean, standard deviation and upper bound of task processing times are known. 3) They formulate the problem by a distributionally robust model with a CVaR constraint, while we formulate the problem by a joint chance constraint. 4) In terms of solution method, they develop a cutting-plane method and an approximation method, while in our study, a lifted cut-and-solve method is proposed. In addition, these studies do not consider identical parts of EOL products, except the study with a deterministic environment (Ilgin et al., 2017). Pioneering works have advanced multi-product DLBP related research, but the characteristics of flexible multi-product disassembly lines, such as disassembly scheme selection, identical parts of EOL products and uncertainty during disassembly, have not been considered simultaneously.

The present study investigates a new flexible multi-product disassembly line balancing problem in which 1) disassembly schemes need to be selected, 2) a workstation and a disassembly scheme are capable of disassembling multiple EOL products, 3) identical parts of multiple products can be treated as identical tasks, and 4) only partial probability distribution information of processing times is known. This study focuses on proposing a novel approach to disassembly multiple EOL products, which aids operation managers in decision-making and improving the performance of the disassembly process. For the problem, an integrated approach is designed, which is composed of an appropriate chance-constrained program, an approximated distribution-free model, efficient valid inequalities, and a lifted cut-and-solve method. The developed approach is tested on an illustrative example, 10 instances based on realistic products and 480 randomly generated instances. The experiment results show that the approach can solve large-scale instances with up to 20 products, 400 tasks, and 86 workstations. The results of the present study provide the following main contributions.

- A new flexible multi-product DLBP is investigated, in which disassembly scheme selection, identical parts of EOL products and uncertain processing time are considered simultaneously;
- An integrated approach according to the problem characteristics is designed to solve the studied problem. Notably, the experimental results show that valid inequalities save approximately 75% of computational time on average, and the average computational time of the lifted cut-and-solve method is only approximately 17.5% of the time needed by the commercial solver CPLEX version 12.9;
- The proposed approach can solve large-scale instances with up to 20 EOL products, 400 tasks, and 86 workstations.

The remainder of this paper is organized as follows. Section 2 provides a literature review. Section 3 describes the studied problem and proposes a new joint chance-constrained program. Based on problem properties, the proposed model is approximately transformed into a distribution-free model and valid inequalities are developed in Section 4. An exact lifted cut-and-solve method is proposed in Section 5. Numerical experiments are conducted in Section 6 to evaluate the performances of the distribution-free model, the valid inequalities, and the lifted cut-and-solve method. Section 7 provides conclusion and future research directions.

2. Literature review

DLBP has been widely studied in the last two decades, and most existing works address the single-product DLBP (Altekin et al., 2008; Bentaha et al., 2014; Ren et al., 2017; Li et al., 2019; Kucukkoc, 2020; Edis, 2021). Because this work studies the multi-product DLBP, we focus on multi-product DLBP related studies in the literature.

First, the multi-product DLBP is classified into two categories, deterministic and uncertain. For deterministic multi-product DLBP, Ilgin et al. (2017) examine a multi-objective mixed-model DLBP in which the same parts of different EOL products are disassembled by so-called common tasks with processing times that may be different between EOL products. Their study assumes that the disassembly schemes of EOL products are predetermined. A mixed-integer linear programming model is proposed for the problem. Then a linear physical programming method is proposed to solve their problem. Fang et al. (2019) investigate a robotic mixed-model DLBP with parallel robots in each workstation to simultaneously minimize four objectives, the cycle time, total energy consumption, peak workstation energy consumption and the number of robots used. Identical parts of multiple EOL products are not considered in their study. A mixed-integer linear program is proposed to formulate the studied problem, and a knowledge-leveraging evolutionary algorithm is designed to solve it. Moreover, they extend their study by considering energy resource constraints (Fang et al., 2020b). A mixed-integer linear programming model is constructed for the problem, and an ε-constraint method based NSGA-II algorithm is proposed to solve it.

For the uncertain multi-product DLBP, Altekin et al. (2008) study a stochastic multi-objective mixed-model U-shaped DLBP where the task processing time follows a normal distribution. The authors assume that the EOL products have no identical parts and that the disassembly schemes are predetermined. A non-linear stochastic programming model is formulated for the problem. A collaborative ant colony optimization algorithm is proposed to solve the problem. Paksoy et al. (2013) investigate a multi-objective mixed-model DLBP with fuzzy objectives, in which EOL products have no identical parts. For the problem, a mixed-integer linear programming model is proposed and solved using the commercial solver LINGO 11.0. Two different fuzzy programming approaches, binary fuzzy goal programming and fuzzy multi-objective programming, are applied to address multiple objectives. Fang et al. (2020a) recently investigate a stochastic multi-objective mixed-model DLBP with bounded task processing time, in which products are disassembled by parallel robots in each workstation. However, the disassembly scheme selection and identical parts of EOL products are not considered in their study. A mixed-integer linear program is proposed for the problem, where the stochastic task processing times are represented by interval numbers. An evolutionary simulated annealing algorithm is developed to solve the problem. Liu et al. (2022) study a stochastic multi-product DLBP with workforce assignment. The uncertain task processing time is represented by partial information of the probability distribution, i.e., the mean and covariance matrix. Each workstation in their study can only handle one type of EOL product. A stochastic program with conditional value-at-risk constraints is developed for the problem. Valid inequalities are proposed to reduce the solution space, but their efficiency is not evaluated. An exact cutting-plane method is proposed to solve this problem. Yin et al. (2022) recently study a multi-product partial DLBP with parallel robots in each workstation to minimize the cycle time, energy consumption, and improved hazardous index. In their study, the uncertain task processing times are described by interval numbers. But the disassembly schemes of EOL products are predetermined, and identical parts of EOL products are not considered. A mixed-integer programming model is established for the studied problem. Then, a multi-objective hybrid driving algorithm is proposed to solve the problem.

Table 1: Comparisons of related studies on multi-product DLBP

Existing work	Problem setting						Modelling	Valid	Solution
	Deterministic	Stochastic	Fuzzy	DSS	MIP	ODM	method	inequality	method
Ilgin et al. (2017)	✓				✓	✓	MIPM		LPP+Solver
Fang et al. (2019)	✓			✓		✓	MIPM		Metaheuristic[1]
Fang et al. (2020b)	✓			✓		✓	MIPM		Metaheuristic[2]
Paksoy et al. (2013)			✓	✓		✓	MIPM		BFGP+FMOP+Solver
Altekin et al. (2008)		ND				✓	SPM		Metaheuristic[3]
Fang et al. (2020a)		Interval				✓	MIPM		Metaheuristic[4]
Liu et al. (2022)		MCM		✓			DRM-CVaR	✓	Exact method[1]
Yin et al. (2022)		Interval				✓	MIPM		Metaheuristic[5]
This work		**MDU**		✓	✓	✓	**DFM-C**	✓	**Exact method[2]**

ND: Normal distribution of task processing time; MCM: the mean and covariance matrix of task processing time; MDU: the mean, standard deviation and upper bound of task processing time; DSS: Disassembly schemes selection; ODM: One workstation can disassemble multiple products; MIP: Multiple products have identical parts; MIPM: Mixed integer linear programming model; SPM: Stochastic programming model; DFM-CVaR: Distributionally robust model with CVaR constraint; DFM-C: Distribution-free model with chance constraint; BFGP: Binary fuzzy goal programming; FMOP: Fuzzy multi-objective programming; LPP: Linear physical programming; Metaheuristic[1]: Knowledge-leveraging evolutionary algorithm; Metaheuristic[2]: ε-constraint method based NSGA-II algorithm; Metaheuristic[3]: Collaborative ant colony optimization algorithm; Metaheuristic[4]: Simulated annealing; Metaheuristic[5]: Multi-objective hybrid driving algorithm; Exact method[1]: Cutting-plane method; Exact method[2]: Lifted cut-and-solve method.

Table 1 summarises the related studies on multi-product DLBPs and shows the differences between existing works and the present study. We can find that (1) the stochastic multi-product DLBP with partial distributed information is rarely studied, (2) the main characteristics of a flexible disassembly line are not simultaneously considered, and (3) most constructed models are solved using commercial solvers or metaheuristics, but few studies develop exact methods.

3. Problem description and formulation

This section states and formulates the flexible multi-product DLBP using a joint chance-constrained program.

Consider that a set of EOL products have to be disassembled. The disassembly of identical parts of these products should be accomplished using identical tasks. To depict the studied problem, we use the ECU example mentioned in the Introduction. In Figure 2, the two products (denoted by A and B) have two alternative disassembly schemes that are represented by different colors, red and blue for A, and black and green for B. The underlined parts 5 and 6 are printed circuit boards and are assumed to be hazardous. A disassembly task and a subassembly (disassembly state) are represented by ◯ and □, respectively. I and N represent the sets of disassembly tasks and subassemblies, respectively. Specifically, N_1 and N_2 denote the initial subassemblies of A and B, and N_{FA} and N_{FB} represent the final states of A and B, respectively. T_i^k means that task i can be executed k times because it belongs to k products. For example, T_{10}^2, and T_{13}^2 are identical tasks and maybe executed twice because of identical parts of A and B. Moreover, tasks 6, 7, 11, 12 with gray color are assumed to be hazardous because they handle hazardous parts 5 and 6. For the studied problem, the basic assumptions are summarized as follows:

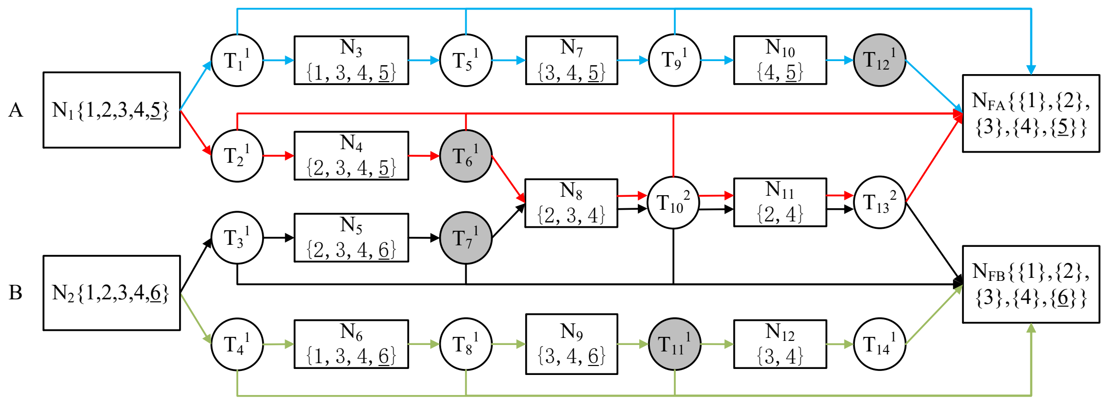


Figure 2: Disassembly schemes of products A and B

(1) EOL products may have identical parts, and the disassembly of these identical parts can be accomplished by identical tasks;

(2) An EOL product can have several disassembly schemes that are known in advance, but only one scheme can be selected;

(3) During disassembly, the given task precedence relationships in a disassembly scheme must be respected;

(4) Task processing times are assumed to be mutually independent and uncertain, and only their limited probability distribution information is available, i.e., the mean, standard deviation and upper bound;

(5) A predetermined cycle time C must be respected with a given risk level α, i.e., the probability that the task processing times of all workstations exceed the cycle time is less than α;

(6) One workstation can disassemble several tasks of multiple EOL products, and workstations are independent and homogeneous.

(7) Some parts are assumed to be hazardous and need specific treatments with a supplementary cost (McGovern & Gupta, 2007a; Bentaha et al., 2015), and the hazardous tasks are independent of disassembly schemes.

The flexible multi-product DLBP consists of optimal selection of a disassembly scheme for each EOL product, determination of the workstations to be opened and assignment of tasks to opened workstations respecting task precedence relationships and cycle time within a given risk level. The objective is to minimize the total disassembly cost, including the open workstation cost and cost of handling hazardous tasks.

Before the problem formulation, the related notations are presented as follows:

Indices:

i : index of disassembly tasks;

$j,\ j'$: index of workstations;

n : index of subassembly nodes;

Parameters:

J : set of workstations;

I : set of disassembly tasks;

H : set of hazardous tasks and $H \subset I$;

N_0: set of initial subassembly nodes;

N : set of subassembly nodes including initial subassembly nodes, $N_0 \subset N$;

P_n : set of immediate predecessors of subassembly node n, where $n \in N$;

S_n : set of immediate successors of subassembly node n, where $n \in N$;

t_i : stochastic processing time of task i, where $i \in I$;

k_i : the maximum number of times that task i can be executed, where $i \in I$;

C : the cycle time of all workstations;

α : a given risk level (probability) that the task processing time exceeds the cycle time for all workstations;

C_F : the cost for opening a workstation;

C_H : the cost for handing a hazardous task.

Decision variables:

x_{ij}: non-negative integer variable, number of times of task i is assigned to workstation j, where $i \in I, j \in J$;

y_j: binary variable, equal to 1 if workstation j is open, 0 otherwise, where $j \in J$.

For the problem, a joint chance-constrained program P1 is formulated as follows:

$$\textbf{P1:} \quad \min \quad C_F \sum_{j\in J} y_j + C_H \sum_{i\in H}\sum_{j\in J} x_{ij} \tag{1}$$

$$s.t. \quad \sum_{i\in S_n}\sum_{j\in J} x_{ij} = 1, \ \forall n \in N_0 \tag{2}$$

$$\sum_{j\in J} x_{ij} \le k_i, \ \forall i \in I \tag{3}$$

$$\sum_{i\in S_n}\sum_{j\in J} x_{ij} = \sum_{i\in P_n}\sum_{j\in J} x_{ij}, \ \forall n \in N\backslash N_0 \tag{4}$$

$$\sum_{i\in S_n}\sum_{j=1}^{j'} x_{ij} \le \sum_{i\in P_n}\sum_{j=1}^{j'} x_{ij}, \ \forall n \in N\backslash N_0, \ \forall j' \in J \tag{5}$$

$$\sum_{i\in I} x_{ij} \le \sum_{i\in I} k_i y_j, \ \forall j \in J \tag{6}$$

$$y_{j-1} \ge y_j, \ \forall j \in J\backslash\{1\} \tag{7}$$

$$Pr\left(\sum_{i\in I} t_i x_{ij} \le C, \ \forall j \in J\right) \ge 1-\alpha \tag{8}$$

$$y_j \in \{0,1\}, \forall j \in J \tag{9}$$

$$x_{ij} \text{ nonnegative integer}, \ \forall i \in I, \ j \in J \tag{10}$$

Objective (1) minimizes the total cost, including the workstation opening cost and the extra cost for handling hazardous tasks. Constraints (2) mean that only one task is selected at the beginning of disassembly for each EOL product. Constraints (3) indicate that task i may be executed at most k_i times. Constraints (4) ensure the flow conservation of each subassembly, i.e., equality of the executed times of preceding tasks and succeeding tasks of each subassembly. Naturally, constraints (2) and (4) can guarantee that only one disassembly scheme is selected for each EOL product. Constraints (5) respect the precedence relationships of tasks. Constraints (6) guarantee that a task may be assigned to a workstation only when the workstation is opened. Constraints (7) number the opened workstations from small to large and avoid empty workstations. Constraint (8) ensures the joint probability that the task processing time does not exceed the cycle time for all workstations is greater than or equal to 1-α. Constraints (9) - (10) define the domains of decision variables.

The above-mentioned joint chance-constrained model P1 is stochastic and non-linear because of its stochastic task processing time t_i, $i \in I$, and the joint chance constraint (8). To efficiently solve the studied problem, P1 is approximately transformed into a distribution-free model P2 in Section 4.

4. Distribution-free model and valid inequality

In this section, based on problem analysis, the joint chance-constrained programming model P1 is transformed into a distribution-free model P2 and valid inequalities are proposed to tighten its solution space.

4.1. Distribution-free model

To transform P1 into an approximated distribution-free model, the joint chance constraint (8) in P1 needs to be transformed into individual chance constraints based on the assumption that workstations are independent of each other and task processing times are mutually independent in Section 3. To this end, β_j is introduced to represent the individual risk level for respecting the cycle time of workstation j, $\forall j \in J$, and the joint chance constraint (8) can be equivalently transformed to (11) and (12) as follows:

$$Pr\left(\sum_{i\in I} t_i x_{ij} \le C\right) \ge 1-\beta_j,\ \forall j \in J \tag{11}$$

$$\prod_{j\in J}(1-\beta_j) = 1-\alpha \tag{12}$$

where $0 \le \beta_j \le 1, \forall j \in J$.

Constraints (11) mean that the workload of workstation j respects the cycle time with at least a possibility of $1-\beta_j$. Constraint (12) establishes the relationship between β_j and α. According to the commonly used method for determining the value of individual risk level (Bentaha et al., 2015; Zheng et al., 2018), which assumes equality of all individual risk levels, parameter β_j is calculated by $\beta_j = 1 - \sqrt[|J|]{1-\alpha}, \forall j \in J$, where $|J|$ is the number of workstations.

With given information of the stochastic task processing time t_i, the mean $\mu_i = E[t_i]$, the standard deviation σ_i and the upper bound $\mu_i(1+b_i)$, where b_i is the limited deviation ratio to the mean, and similar to Ng (2014) and Zheng et al. (2018), the stochastic task processing time can be expressed as:

$$t_i = \mu_i(1+Z_i),\ \forall i \in I \tag{13}$$

where Z_i is the deviation ratio to the mean of the task processing time and is limited by b_i, i.e., $Z_i \le b_i$.

Therefore, the individual chance constraints (11) are reformulated as:

$$Pr\left(\sum_{i\in I} \mu_i(1+Z_i) x_{ij} \le C\right) \ge 1-\beta_j,\ \forall j \in J \tag{14}$$

To construct the distribution-free model, constraints (14) are approximately transformed to the following constraints:

$$\sum_{i \in I} (\mu_i + \nu_i) x_{ij} \leq C, \ \forall j \in J \tag{15}$$

where ν_i is an auxiliary parameter that reflects the uncertainty of task processing time t_i, which will be explained later.

Then, distribution-free model P2 is proposed as follows.

$$\begin{aligned} \textbf{P2:} \qquad & \min \quad C_F \sum_{j \in J} y_j + C_H \sum_{i \in H} \sum_{j \in J} x_{ij} \\ s.t. \qquad & (2) - (7), (9), (10), (15) \end{aligned}$$

As long as the value of ν_i is determined, P2 becomes a mixed-integer linear program that may be solved using a commercial solver, such as CPLEX, at least for small-sized instances. If the following proposition is true, ν_i can be pre-determined.

Proposition 4.1. *As long as any ν_i satisfies the following equation*

$$f(\nu_i) := \min_{\lambda_i > 0} \left\{ e^{-\lambda_i \nu_i / \mu_i} \left(1 + \frac{E[Z_i^2]}{b_i^2} (e^{\lambda_i b_i} - \lambda_i b_i - 1) \right) - \beta_j \right\} = 0 \tag{16}$$

then, any solution of P2 must satisfy the individual chance constraints (14).

Proof. See Appendix A. □

The value of ν_i can be obtained by solving equality (16). To determine ν_i, Algorithm 1 is presented.

Algorithm 1 Calculation of the value of ν_i

```
Input: $\mu_i, b_i, \sigma_i^2, \nu_i = 0, \lambda_i = 0.001$
 1: $f_{value} = \widehat{f}(\nu_i, \lambda_i) = e^{-\lambda_i \nu_i / \mu_i} \left(1 + \frac{E[Z_i^2]}{b_i^2}(e^{\lambda_i b_i} - \lambda_i b_i - 1)\right) - \beta_j$;
 2: while $f_{value} > 0.001$ do
 3:     if $f \leq 1$ then
 4:         $\lambda_i = \lambda_i + 0.001$;
 5:         $f_{value} = \widehat{f}(\nu_i, \lambda_i)$;
 6:     else
 7:         $\nu_i = \nu_i + 0.1$;
 8:         $\lambda_i = 0.001$;
 9:         $f_{value} = \widehat{f}(\nu_i, \lambda_i)$;
10:     end if
11: end while
Output: The value of $\nu_i$
```

The basic idea of the algorithm is to determine a combination of $(\nu_i,\ \lambda)$ that satisfies $\widehat{f}(\nu_i, \lambda_i) = 0$. Although P2 is a mixed-integer linear programming model once ν_i is fixed, P2 is still NP-hard because the deterministic single-product DLBP is NP-hard (McGovern & Gupta, 2007b). Therefore, effective valid inequalities to improve P2 are proposed in the next section.

4.2. Improved distribution-free model

This section presents two valid inequalities to tighten the solution space of model P2.

4.2.1. Valid inequality 1

The first valid inequality attempts to limit the number of workstations to be opened. To propose the valid inequality, we relax the assumption that one workstation can execute several tasks of different EOL products in model P2 to form its relaxed model P2$'$, in which one workstation executes the tasks of only one EOL product. P (indexed by p) denotes the set of EOL products and $|J_p|$ denotes the number of workstations required for product p in a feasible solution of P2$'$. $\sum\limits_{p\in P} |J_p|$ denotes the total number of workstations of the feasible solution of P2$'$. The following proposition is formulated.

Proposition 4.2. *Let $\sum\limits_{j\in J} y_j$ be the number of opened workstations in a feasible solution of P2, then the following inequality:*

$$VI1: \quad \sum_{j\in J} y_j \le \sum_{p\in P} |J_p| \tag{17}$$

is a valid inequality for P2.

Valid inequality 1 shows that a feasible solution of model P2 exists in which the number of open workstations is not greater than $\sum\limits_{p\in P} |J_p|$. Therefore, the number of opened workstations in an optimal solution of P2 does not exceed $\sum\limits_{p\in P} |J_p|$.

Proof. See Appendix B. □

To determine $\sum\limits_{p\in P} |J_p|$, Algorithm 2 is proposed below. L_p (indexed by l), I_l^p and T are the disassembly scheme set of product p, the task set of disassembly scheme l in L_p and the available time of a workstation, respectively. $|J_p^l|$ denotes the number of open workstations of disassembly scheme l of product p. Let $c_i = \mu_i + \nu_i$ denote the processing time of task $i, \forall i \in I$. Recall that μ_i is the mean task processing time and ν_i reflects the uncertainty of task processing time.

Algorithm 2 Upper bound determination for opened workstations $\sum\limits_{p \in P} |J_p|$

```
Input: Disassembly schemes of EOL products ($L_p, \forall p \in P$), task processing times ($c_i, \forall i \in I$), cycle time $C$
1:  $p = 1$;
2:  while ($p \leq |P|$) do
3:      $l = 1$, $|J_p| = 0$;
4:      while ($l \leq |L_p|$) do
5:          $station_{num} = 1$, $i = 1$, $T = C$;
6:          while ($i \leq |I_l^p|$) do
7:              if $c_i \leq T$ then
8:                  $T = T - c_i$, $i = i + 1$;
9:              else
10:                 $station_{num} = station_{num} + 1$, $T = C$;
11:             end if
12:         end while
13:         $|J_p^l| = station_{num}$;
14:         if $|J_p^l| > |J_p|$ then
15:             $|J_p| = |J_p^l|$;
16:         end if
17:         $l = l + 1$;
18:     end while
19:     $p = p + 1$;
20: end while
Output: The value of $\sum\limits_{p \in P} |J_p|$
```

Algorithm 2 contains three loops: Lines 6 to 12 calculate the number of workstations of disassembly scheme l of product p in a feasible solution of P2$'$, Lines 4 to 18 determine $|J_p| = \max\limits_{l \in L_p} \left\{ |J_p^l| \right\}$, and Lines 2 to 20 obtain $\sum\limits_{p \in P} |J_p|$.

4.2.2. Valid inequality 2

The second valid inequality determines the workstations to which disassembly tasks cannot be assigned due to their precedence relationships. The task precedence relationship implies that a task cannot be executed before its predecessors and after its successors. For simplicity, let PT_i^l and ST_i^l denote the sum of processing times of the predecessors and successors of task i in disassembly scheme l, respectively, where $l \in L_i$ and L_i is the disassembly scheme set containing task i. Consequently, the minimum numbers of workstations for predecessors and successors of task i, n_{pi} and n_{si} may be determined by the following formulas: $n_{pi} = \min\limits_{l \in L_i} \left\{ \lceil PT_i^l / C \rceil \right\}$ and $n_{si} = \min\limits_{l \in L_i} \left\{ \lceil ST_i^l / C \rceil \right\}$, recall that C is the cycle time and $\lceil x \rceil$ is the smallest integer greater than or equal to x.

Without loss of generality, the second valid inequality can be analyzed via the case illustrated in Figure 3, in which tasks are numbered and represented by $\bigcirc$. The illustrated EOL product has two disassembly schemes that contain 11 tasks, and the corresponding processing times ($c_i = \mu_i + \nu_i$) are set as 8, 7, 8, 5, 4, 7, 6, 8, 5, 9, and 7 seconds. With the assumption in Section 3, only one scheme of a product can be selected. For the green scheme ($l = 1$), task 7 has predecessors $\{1, 2, 3\}$ and successor $\{8, 9\}$, and we obtain $PT_7^1 = c_1 + c_2 + c_3 = 23$ seconds and $ST_7^1 = c_8 + c_9 = 13$ seconds. Similarly for the blue scheme ($l = 2$), we have $PT_7^2 = c_4 + c_5 + c_6 = 16$ seconds and $ST_7^2 = c_{10} + c_{11} = 16$ seconds. We suppose that the cycle time is 10 seconds, and we have $J_{max} = 6$ according to Algorithm 1. Therefore, $n_{p7} = \min(\lceil 23/10 \rceil, \lceil 16/10 \rceil) = 2$ and $n_{s7} = \min(\lceil 13/10 \rceil, \lceil 16/10 \rceil) = 2$. Task 7 cannot be assigned before workstation $n_{p7} = 2$ and after workstation $|J_{max}| - n_{s7} + 1 = 5$. Therefore, for model P2, we have $x_{71} = x_{76} = 0$ and the following proposition.

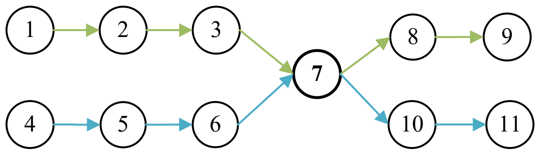


Figure 3: An illustration of valid inequality 2

Proposition 4.3. *With the pre-determined* $\sum_{p \in P} |J_p|$ *by Algorithm 2, the following inequality:*

$$VI2: \quad x_{ij} = 0, \; \forall i \in I, j < n_{pi}, j > \sum_{p \in P} |J_p| - n_{si} + 1 \tag{18}$$

is a valid inequality for P2.

Valid inequality 2 excludes the task-workstation assignments that are not in a feasible solution of P2. Specifically, task $i \in I$ cannot be assigned before workstation n_{pi} and after workstation $\sum_{p \in P} |J_p| - n_{si} + 1$.

Proof. See Appendix C. $\square$

4.2.3. Improved distribution-free model

The numerical experiments in Section 6 show that the model with VI1 and VI2 performs best. Therefore, an improved model of P2 is presented:

$$\textbf{P3:} \qquad \min \quad C_F \sum_{j \in J} y_j + C_H \sum_{i \in H} \sum_{j \in J} x_{ij}$$

$$s.t. \qquad (2)-(7),(9),(10),(15),(17),(18)$$

Although the improved model P3 is tighter than P2, it is still time-consuming for large-scale instances. Therefore, an exact lifted cut-and-solve method is proposed in the next section to efficiently solve model P3.

5. Lifted cut-and-solve method

This section presents an exact improved cut-and-solve (CS) method (termed lifted CS). The CS method is first introduced by Climer & Zhang (2006) to solve the asymmetric traveling salesman problem. CS method is a particular branch and bound algorithm. Compared with the branch, bound and remember (BB&R) algorithm (Sewell & Jacobson, 2012), which is an extension of the branch and bound algorithm that remembers all the searched sub-problems and has a new dominance rule based on the memorized sub-problems, there are the following differences: 1) At each node, the BB&R considers all sub-problems, while the CS needs to consider only two sub-problems; 2) The BB&R memories the information related to all explored sub-problems, but the CS memories only the best solution and its objective value. Hence, the CS method can reduce the size of the search tree and memory required. The CS method has been successfully used to solve many combinatorial optimization problems, such as facility location (Yang et al., 2012; Gadegaard et al., 2018), and lane reservation (Fang et al., 2013; Wu et al., 2017). In the following, the basic procedure of the CS method is first presented, and improvements are proposed to solve the studied problem.

The iterative procedure of the CS method is described as follows. At the $(i-1)$-*th* iteration, a *Piercing Cut* (PC_{i-1}) is constructed based on the linear relaxed solution of a *Dense Problem* (DP_{i-1}). At the *i-th* iteration, PC_{i-1} cuts the solution space of DP_{i-1} into two sub-spaces that correspond to a *Sparse Problem* (SP_i) and a new *Dense Problem* (DP_i). SP_i generally has a small solution space and may be exactly solved to yield a new feasible solution of the original problem. The current best upper bound (UB) may be updated. DP_i often has a large solution space, and its relaxed problem is generally solved to obtain a lower bound (LB) of the original problem. DP_0 is the original problem. The process continues until the current LB is greater than or equal to the best UB. Then, the solution of the best UB is output as an optimal solution of the original problem.

The efficiency of the CS method primarily depends on the effectiveness of the PC, the quality of the LB and the speed for solving sparse problems. To enhance the performance of the CS method for solving model P3, the present work devises a new lifted CS method.

Particularly, 1) a constructive heuristic (Algorithm 3) is proposed to obtain an initial UB that may be an optimal solution of the original problem, while a traditional CS does not do it. 2) double PCs (PC_{i-1}^1, PC_{i-1}^2) based on partially linear relaxation are proposed at each iteration to obtain a better LB, while in the traditional CS, only one PC is proposed at each iteration, and the solution space of SP may still be large and time-consuming. 3) SP is further divided into two sub-problems (SP_i^1, SP_i^2) by a second PC for an efficient resolution, whereas SP is solved directly in the classical CS. The framework and search tree of the lifted CS method are outlined in Figures 4 and 5, respectively.

The next sections present the heuristic for obtaining an initial UB, the double PCs and the formulations of sparse and dense problems.

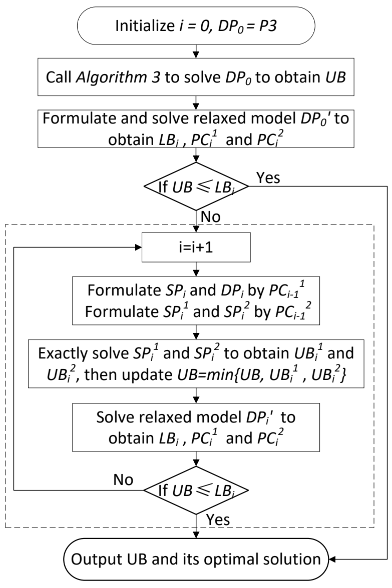

Initialize i = 0, DP0 = P3
Call Algorithm 3 to solve DP0 to obtain UB
Formulate and solve relaxed model DP0' to obtain LBi, PCi1 and PCi2
If UB ≤ LBi
Yes
No
i=i+1
Formulate SPi and DPi by PCi-11
Formulate SPi1 and SPi2 by PCi-12
Exactly solve SPi1 and SPi2 to obtain UBi1 and UBi2, then update UB=min{UB, UBi1, UBi2}
Solve relaxed model DPi' to obtain LBi, PCi1 and PCi2
No
If UB ≤ LBi
Yes
Output UB and its optimal solution

Figure 4: The framework of lifted cut-and-solve method

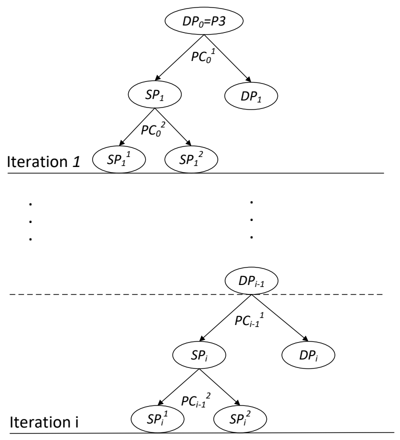


Figure 5: The search tree of lifted cut-and-solve method

5.1. Heuristic to determine an initial UB

The main purpose of the proposed heuristic is to select a disassembly scheme with the minimum sum of task processing times for each EOL product, open workstations for product p one at a time, $p = 1, ..., |P|$, and assign the related tasks to workstations respecting task precedence relationships and cycle time constraints.

Note that I_p^l denotes the ordered task set in which the precedence relationships of tasks are respected for $l \in L_p$, and L_p is the disassembly scheme set of product p. Recall that $c_i = \mu_i + \nu_i$ denotes the processing time of task $i, \forall i \in I$. The proposed heuristic is summarised as follows.

Algorithm 3 The heuristic to obtain an upper bound

```
Input: $c_i, \forall i \in I_p^l; I_p^l, \forall l \in L_p, p \in P$
1: Set $p = 1$;
2: while $p \leq |P|$ do
3:     for $l = 1 \to |L_p|$ do
4:         Calculate $T_l = \sum_{i \in I_p^l} c_i$;
5:     end for
6:     $I_p^* = I_p^{l^*}$, where $l^* = arg \min_{l \in L_p} \{T_l\}$;
7:     $p = p + 1$;
8: end while
9: Set $j = 0, T = 0, IP = \bigcup_{p \in P} I_p^*$;
10: while $(i \leq |IP|)$ do
11:     if $c_{IP[i]} \leq T$ then
12:         $T = T - c_{IP[i]}$;
13:         $x_{IP[i]j} = 1$, $i = i + 1$;
14:     else
15:         $j = j + 1$;
16:         $y_j = 1$, $T = C$;
17:     end if
18: end while
19: Calculate $UB = C_F \sum_{j \in J} y_j + C_H \sum_{i \in H} \sum_{j \in J} x_{ij}$;
Output: An $UB$ and its corresponding solution
```

Algorithm 3 consists of two parts. Lines 1 to 8 select the disassembly scheme with the minimum sum of task processing times for each EOL product, and Lines 9 to 18 decide the workstations to be opened and assign the selected tasks to the opened workstations.

5.2. Double piercing cuts

Existing CS methods generally define one PC, and the corresponding SP may still be difficult to solve. The double PCs (PC_{i-1}^1, PC_{i-1}^2) at the $(i-1)$-*th* iteration are designed based on an optimal solution of linear relaxation of DP_{i-1} in this study, denoted as $(\widetilde{x}_{ij}, \widetilde{y}_j)$. Note that the objective function value is primarily determined by the number of workstations to be opened, i.e., $\sum_{j \in J} y_j$. Because the cost of handling hazardous tasks is the same regardless of which disassembly scheme is selected. A better LB may be obtained with the partial relaxation of DP_{i-1}. Preliminary experiments show that task

i has a larger probability of being assigned to workstation j in the optimal solution of $P3$ when $\widetilde{x}_{ij}$ has a large value. Therefore, we propose 1) a partial relaxation strategy for DP_{i-1}, 2) the first PC with the value of $\widetilde{y}_j$, 3) the second PC with the value of $\widetilde{x}_{ij}$.

At the $(i-1)$-*th* iteration, with an optimal solution of the relaxed DP_{i-1}, the first piercing cut, PC^1_{i-1}, can be defined as follows.

$$\sum_{y_j \in \Omega^1_{i-1}} y_j = 0 \tag{19}$$

where $\Omega^1_{i-1} = \{y_j | \widetilde{y}_j = 0, \forall j \in J\}$.

Let I_{i-1} denote the set of tasks whose values of $\widetilde{x}_{ij}$ are fractions. For any task $i \in I_{i-1}$, the combination of task i assigned to workstation j, i.e., (i,j) corresponding to the largest values of $\widetilde{x}_{ij}$ is presented below:

$$\Psi_{i-1} = \left\{ (i,j) | (i,j) = arg \max_{i \in I_{i-1}, j \in J} \{\widetilde{x}_{ij}\} \right\} \tag{20}$$

Consequently, the second piercing cut PC^2_{i-1} can be defined as follows:

$$\sum_{x_{ij} \in \Omega^2_{i-1}} x_{ij} \leq l_{i-1} - 1 \tag{21}$$

where $\Omega^2_{i-1} = \{x_{ij} | (i,j) \in \Psi_{i-1}\}$ and l_{i-1} is an integer that takes the value of $\sum\limits_{(i,j) \in \Psi_{i-1}} \lceil \widetilde{x}_{ij} \rceil$.

The first piercing cut (PC^1_{i-1}) divide the current DP_{i-1} into two sub-problems DP_i and SP_i. The second piercing cut (PC^2_{i-1}) further divide SP_i into two sub-problems SP^1_i and SP^2_i. The formulations of sparse and dense problems are presented below.

5.3. Sparse and dense problem formulations

According to the above defined double PCs, the formulations of the dense problem (DP_i) at the *i-th* iteration can be presented as follows.

$$\begin{aligned} DP_i: \quad & \min \quad C_F \sum_{j \in J} y_j + C_H \sum_{i \in H} \sum_{j \in J} x_{ij} \\ s.t. \quad & (2)-(7), (9), (10), (15), (17), (18) \\ & \sum_{y_j \in \Omega^1_{i-1}} y_j \geq 1 \end{aligned} \tag{22}$$

Two sparse problems SP^1_i and SP^2_i are defined as follows:

$$\begin{aligned} SP^1_i: \quad & \min \quad C_F \sum_{j \in J} y_j + C_H \sum_{i \in H} \sum_{j \in J} x_{ij} \\ s.t. \quad & (2)-(7), (9), (10), (15), (17), (18) \end{aligned}$$

$$\sum_{y_j \in \Omega_{i-1}^1} y_j = 0 \tag{19}$$

$$\sum_{x_{ij} \in \Omega_{i-1}^2} x_{ij} \geq l_{i-1} \tag{23}$$

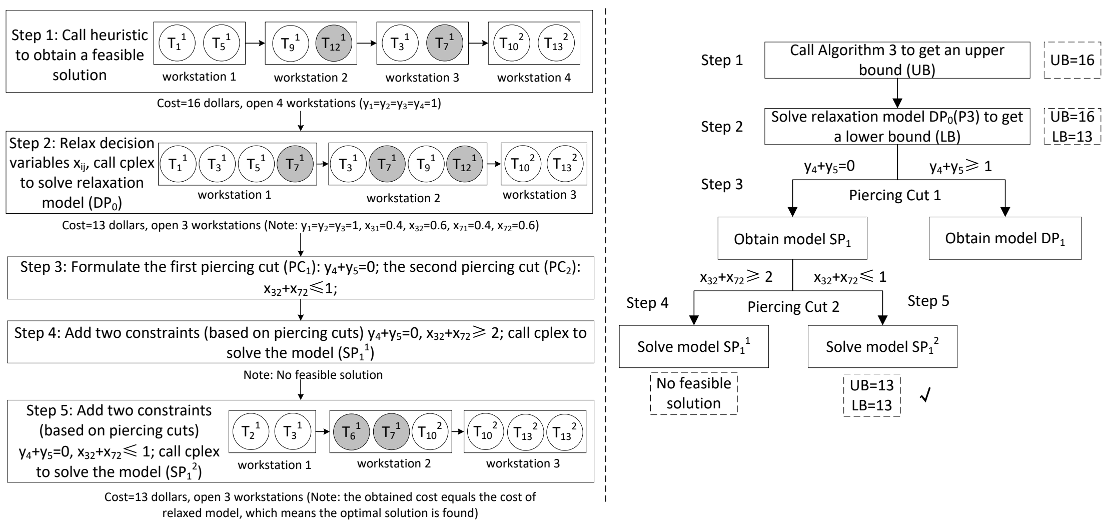
$$SP_i^2 : \quad \min \quad C_F \sum_{j \in J} y_j + C_H \sum_{i \in H} \sum_{j \in J} x_{ij}$$

$$s.t. \quad (2)-(7), (9), (10), (15), (17), (18)$$

$$\sum_{y_j \in \Omega_{i-1}^1} y_j = 0 \tag{19}$$

$$\sum_{x_{ij} \in \Omega_{i-1}^2} x_{ij} \leq l_{i-1} - 1 \tag{21}$$

To better understand the proposed method, the ECU example mentioned in the Introduction is given below. In the example, there are 2 products and 14 tasks with processing times (in seconds) of {30, 35, 45, 35, 35, 20, 18, 35, 31, 38, 50, 20, 25, 25}. The costs of opening a workstation (C_F) and handling a hazardous task (C_H) are 3 and 2 dollars, respectively. There are 6 workstations available, and the cycle time C is 90 seconds. The detailed process of the lifted cut-and-solve method is depicted below in Figure 6.



Step 1: Call heuristic to obtain a feasible solution
T_1^1 T_5^1 → T_9^1 T_{12}^1 → T_3^1 T_7^1 → T_{10}^2 T_{13}^2
workstation 1
workstation 2
workstation 3
workstation 4
Cost=16 dollars, open 4 workstations ($y_1=y_2=y_3=y_4=1$)
Step 2: Relax decision variables x_{ij}, call cplex to solve relaxation model (DP_0)
T_1^1 T_3^1 T_5^1 T_7^1 → T_3^1 T_7^1 T_9^1 T_{12}^1 → T_{10}^2 T_{13}^2
workstation 1
workstation 2
workstation 3
Cost=13 dollars, open 3 workstations (Note: $y_1=y_2=y_3=1$, $x_{31}=0.4$, $x_{32}=0.6$, $x_{71}=0.4$, $x_{72}=0.6$)
Step 3: Formulate the first piercing cut (PC_1): $y_4+y_5=0$; the second piercing cut (PC_2): $x_{32}+x_{72}\leqslant 1$;
Step 4: Add two constraints (based on piercing cuts) $y_4+y_5=0$, $x_{32}+x_{72}\geqslant 2$; call cplex to solve the model (SP_1^1)
Note: No feasible solution
Step 5: Add two constraints (based on piercing cuts) $y_4+y_5=0$, $x_{32}+x_{72}\leqslant 1$; call cplex to solve the model (SP_1^2)
T_2^1 T_3^1 → T_6^1 T_7^1 T_{10}^2 → T_{10}^2 T_{13}^2 T_{13}^2
workstation 1
workstation 2
workstation 3
Cost=13 dollars, open 3 workstations (Note: the obtained cost equals the cost of relaxed model, which means the optimal solution is found)
Step 1
Call Algorithm 3 to get an upper bound (UB)
UB=16
Step 2
Solve relaxation model DP_0(P3) to get a lower bound (LB)
UB=16
LB=13
Step 3
$y_4+y_5=0$
$y_4+y_5\geqslant 1$
Piercing Cut 1
Obtain model SP_1
Obtain model DP_1
$x_{32}+x_{72}\geqslant 2$
$x_{32}+x_{72}\leqslant 1$
Step 4
Piercing Cut 2
Step 5
Solve model SP_1^1
Solve model SP_1^2
No feasible solution
UB=13
LB=13
✓

Figure 6: The detailed process of the algorithm for solving the instance

Next, to evaluate the performance of the proposed lifted CS method, numerical experiments are performed, and the results are presented in Section 6.

6. Numerical experiments

Numerical experiments are performed on an illustrative example, 10 instances based on realistic products and 480 randomly generated instances to evaluate the performance of the proposed integrated approach. The program is coded using the C++ programming language in Microsoft Visual Studio 2019. All numerical experiments are performed on a personal computer with Core I5 and 3.20 GHz CPU with 12GB RAM. All models in Sections 6.1, 6.3, 6.4 and model P3 in Section 6.5 are solved using the CPLEX solver version 12.9. The CPLEX code and instance data are available on $https://www.researchgate.net/publication/359266654\_Instance\_data\_and\_cplex\_code$.

6.1. An illustrative example

In this part, the example of the engine ECU mentioned in the Introduction is investigated to test our models and methods. The disassembly schemes of the two ECUs are shown in Figure 2, which contain 14 tasks and 12 subassemblies. The mean task processing times (in seconds) are {30, 35, 45, 35, 35, 20, 18, 35, 31, 38, 50, 20, 25, 25}. The upper bound, standard deviation and risk level are 0.1, 0.01 and 5%, respectively. The costs of opening a workstation and handling a hazardous task are 3 and 2 dollars, respectively. Assume that there are 6 workstations available, and the cycle time C is 90 seconds.

Figure 7 reports the results of the illustrative instance. In the optimal solution, 3 workstations are opened, and the total cost is 13 dollars. For product A, the selected disassembly scheme contains tasks $\{1, 5, 9, 12\}$, and the selected disassembly scheme of product B contains tasks $\{3, 7, 10, 13\}$, corresponding to the blue and black lines in Figure 2, respectively.

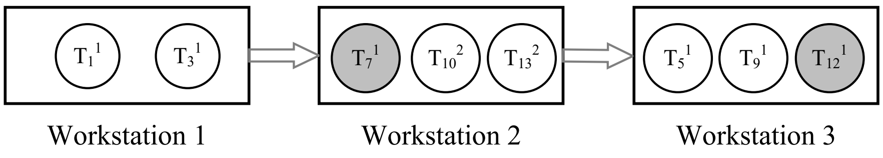


Figure 7: The result of the illustrative example

6.2. Instance data and parameter setting

This paper is the first to study the multi-product DLBP with identical parts and stochastic task times. Since there is no standard data set that can be directly used, we generate 10 instances based on 7 realistic EOL products from previous literature (see Table 2) by emerging different EOL products to generate the multi-product instances, which is similar to Fang et al. (2020b) and Liu et al. (2022). The basic information of these multi-product instances is summarized in Table 3, where each instance scale is primarily determined by three parameters, i.e., the number of products $|P|$, the number of tasks $|I|$, and the number of available workstations $|J|$.

Table 2: Information of realistic EOL products

Number	EOL product	Reference
1	Hand light	Tang et al. (2002)
2	Compass	Bentaha et al. (2012)
3	Ballpoint pen	Lambert (1999)
4	Sample product	Koc et al. (2009)
5	Piston and connecting rod	Bentaha et al. (2013)
6	Radio set	Lambert (1999)
7	Automatic pencil	Ma et al. (2011)

Table 3: Information of the multi-product instances based on realistic EOL products

Number	Products	$\|P\|$	$\|I\|$	$\|J\|$
1	1,2	2	20	10
2	1,3	2	30	10
3	1,2,3	3	40	15
4	2,3,4	3	53	15
5	1,2,3,4	4	63	20
6	2,3,4,5	4	78	20
7	1,2,3,4,5	5	88	25
8	2,3,4,5,6	5	108	25
9	1,2,3,4,5,6	6	118	30
10	2,3,4,5,6,7	6	145	30

Moreover, to thoroughly evaluate the performances of the integrated method, 96 problem sets, each with 5 instances, for a total of 480 instances are randomly generated. For each EOL product, 20% of tasks are set as identical tasks, and 20% are set as hazardous tasks. The mean task processing times (μ_i) (in seconds) are randomly generated from $[10, 50]$ consistent with He et al. (2020). Similar to Ng (2014), the upper bound and standard deviation of the task processing time ($b_i, E[Z_i^2]$) are set as $(0.1, 0.01)$, $(0.2, 0.05)$ and $(0.3, 0.10)$ to represent different uncertainty levels. As in Bentaha et al. (2015) and He et al. (2020), the unit fixed cost of a workstation C_F and the unit cost for handling hazardous task C_H are set as 3 and 2 dollars, respectively. The risk level α and the cycle time C are set as 5% and 90 seconds, respectively. The input parameters are summarized in Table 4.

Table 4: Input parameters of random instances

Parameters	Values
Number of products ($\|P\|$)	2, 4, ..., 20
Number of tasks ($\|I\|$)	20, 40, ..., 400
number of available workstations ($\|J\|$)	6, 10, ..., 86
Cost to open a workstation (C_F) (dollars)	3
Cost to handle a hazardous task (C_H) (dollars)	2
Cycle time (C) (seconds)	90
Risk level (α)	5%
Mean task times (μ_i) (seconds)	[10,50]
$(b_i, E[Z_i^2])$	$(0.1, 0.01)$, $(0.2, 0.05)$, $(0.3, 0.10)$

6.3. Evaluation of the distribution-free model

The approximated distribution-free model $P2$ (solved via CPLEX) is evaluated by comparison with the corresponding deterministic model (DM)(solved via CPLEX) and a sampling average approximation (SAA) method (see Appendix D for its presentation). The task processing times of the DM are set as the means of the stochastic ones in P2. The task processing times in SAA are assumed to follow normal distributions, in which their means and standard deviations are identical to these values in P2. The number of samples is set to 20 based on preliminary experiments. Numerical experiments are first performed on the realistic EOL products instances.

Tables 5-7 report the computational results of realistic EOL product instances under 3 uncertainty levels, where Obj_{DM}, T_{DM}, Obj_{SAA}, T_{SAA} and Obj_{P2}, T_{P2} denote the average objective values and computational times of DM, SAA and P2, respectively. Specially, the computational time unit is seconds denoted by s. We can observe that the average objective value of the DM is the smallest, which takes the value of 29.5, and the objectives of SAA and P2 are almost the same. On the other hand, the SAA requires more computational time than P2 and DM. Moreover, with the increase of the uncertainty, the objective values and the computational times also increase. Therefore, in general, more workstations are needed to deal with the uncertain task times.

To have an observation of the performances on different methods, experiments are further conducted on random instance sets with up to 6 products, 120 tasks and 30 workstations. Computational results are illustrated in Figures 8-10, in which sub-figures (a) and (b) report the objective values and computational times of DM, SAA and P2, respectively. The detailed results are presented in Tables E1-E3 in Appendix E. Figures 8-10 show that the curves of the objective values of P2 and SAA almost overlap, and they are superior to the values of DM. This result indicates that the solution qualities of P2 and SAA are nearly identical, and DM proposes a better solution because there are

Table 5: The results of realistic EOL product instances with ($b_i = 0.1, E[Z_i^2] = 0.01$)

		DM		SAA		P2	
Instance	(P, I, J)	Obj_{DM}	$T_{DM}(s)$	Obj_{SAA}	$T_{SAA}(s)$	Obj_{P2}	$T_{P2}(s)$
1	(2,20,10)	14.2	0.10	14.2	0.20	14.8	0.09
2	(2,30,10)	19.0	0.19	20.8	1.05	21.4	0.16
3	(3,40,15)	22.8	0.35	24.0	2.29	24.0	0.36
4	(3,53,15)	22.8	0.43	25.6	17.60	25.6	0.66
5	(4,63,20)	29.6	2.35	30.2	18.78	31.4	2.69
6	(4,78,20)	27.8	1.14	29.6	305.60	29.6	1.21
7	(5,88,25)	34.0	2.83	37.0	2411.27	37.0	2.68
8	(5,108,25)	36.4	4.47	37.4	2782.47	39.4	11.99
9	(6,118,30)	43.8	4.99	45.8	3626.41	47.4	8.40
10	(6,145,30)	44.8	8.72	50.8	4837.71	54.0	17.50
Average	-	29.5	2.56	31.5	1400.35	32.5	4.58

Table 6: The results of realistic EOL product instances with ($b_i = 0.2, E[Z_i^2] = 0.05$)

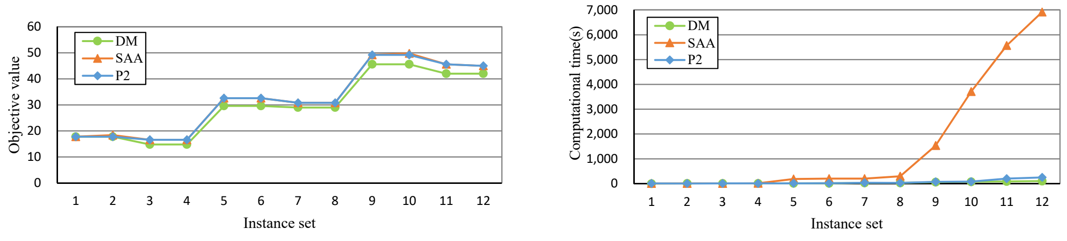
		DM		SAA		P2	
Instance	(P, I, J)	Obj_{DM}	$T_{DM}(s)$	Obj_{SAA}	$T_{SAA}(s)$	Obj_{P2}	$T_{P2}(s)$
1	(2,20,10)	14.2	0.10	16.0	0.30	15.4	0.10
2	(2,30,10)	19.0	0.19	23.2	2.41	22.6	0.22
3	(3,40,15)	22.8	0.35	25.8	12.77	26.4	0.41
4	(3,53,15)	22.8	0.43	28.2	153.42	27.6	0.88
5	(4,63,20)	29.6	2.35	32.6	6824.53	33.2	0.87
6	(4,78,20)	27.8	1.14	30.8	119.36	32.6	11.04
7	(5,88,25)	34.0	2.83	38.5	2333.57	40.0	3.33
8	(5,108,25)	36.4	4.47	43.0	5148.70	41.2	11.56
9	(6,118,30)	43.8	4.99	49.8	6283.63	49.8	32.48
10	(6,145,30)	44.8	8.72	51.7	7218.16	54.0	15.26
Average	-	29.5	2.56	34.0	2209.69	34.3	7.61

no uncertain factors. Based on Tables E1-E3, the average gaps of the objective values between DM and P2 can be calculated by the formula $(Obj_{P2} - Obj_{DM})/Obj_{DM} \times 100\%$. The gaps under the 3 uncertainty levels are 7.72%, 16.44% and 24.83%, respectively, which indicate that the gaps increase with levels of uncertainty. Sub-figures (b) shows that the curves of the computational times of P2 and DM almost overlap, and the computational time of SAA increases sharply. The average computational times of the SAA are 474, 313 and 258 times of those needed by P2 under 3 different uncertainty levels. These results indicate that DM and P2 perform better than SAA in terms of computational time.

Moreover, we utilize the Wilcoxon signed-rank test, a flexible non-parametric sta-

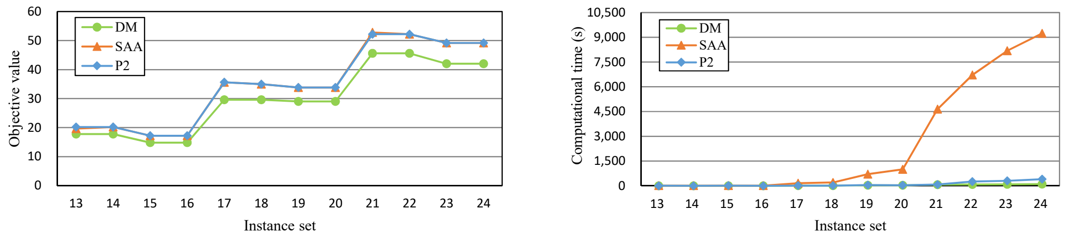
Table 7: The results of realistic EOL product instances with ($b_i = 0.3, E[Z_i^2] = 0.1$)

		DM		SAA		P2	
Instance	(P, I, J)	Obj_{DM}	$T_{DM}(s)$	Obj_{SAA}	$T_{SAA}(s)$	Obj_{P2}	$T_{P2}(s)$
1	(2,20,10)	14.2	0.10	16.6	0.33	17.2	0.11
2	(2,30,10)	19.0	0.19	24.4	3.19	24.4	0.16
3	(3,40,15)	22.8	0.35	27.0	34.31	28.2	0.63
4	(3,53,15)	22.8	0.43	28.6	115.71	28.6	0.73
5	(4,63,20)	29.6	2.35	36.2	1528.23	36.2	2.91
6	(4,78,20)	27.8	1.14	33.2	233.04	33.2	2.79
7	(5,88,25)	34.0	2.83	42.0	4097.66	42.4	134.70
8	(5,108,25)	36.4	4.47	44.0	5444.01	43.6	68.44
9	(6,118,30)	43.8	4.99	54.0	7874.54	53.3	15.27
10	(6,145,30)	44.8	8.72	56.7	9235.37	55.5	100.11
Average	-	29.5	2.56	36.3	2856.64	36.3	32.58



(a) Objective value (b) Computational time

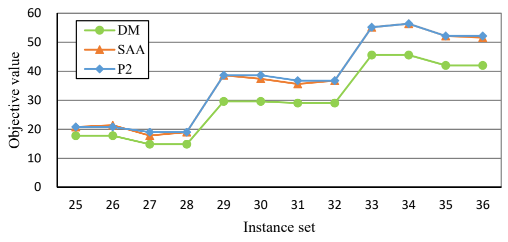
Figure 8: Comparison results with ($b_i = 0.1, E[Z_i^2] = 0.01$)



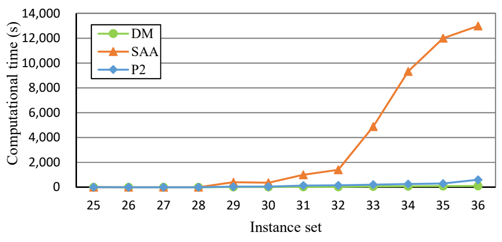
(a) Objective value (b) Computational time

Figure 9: Comparison results with ($b_i = 0.2, E[Z_i^2] = 0.05$)

tistical hypothesis test (García et al., 2009), to compare the performance of the three methods. The principle of the Wilcoxon signed-rank test is to calculate the differences between the two compared methods and analyze these differences to test if they are statistically significantly different. We prefer the Wilcoxon test because it does not assume the normality of the samples (Chica et al., 2010) and it has already been demonstrated to help analyze the behavior of evolutionary algorithms, and it has been well adopted in



(a) Objective value



(b) Computational time

Figure 10: Comparison results with ($b_i = 0.3, E[Z_i^2] = 0.1$)

the disassembly line balancing problems (Fang et al., 2019; Li et al., 2020).

The null-hypothesis H_0 is that the compared two methods have equal performance. The significance level considered in all the tests to be presented is $p = 0.05$. A p-value is smaller than 0.05, denoting a rejection of the null-hypothesis, i.e., the compared two methods perform significantly differently. The results of the Wilcoxon test are illustrated in Table 8. We can obtain that DM outperforms P2 and SAA statistically because the p-values are smaller than 0.0001. In comparison, P2 and SAA are not statistically different in terms of the objective values according to the p-value of 0.3047.

Table 8: Results of Wilcoxon signed-rank test for three different methods

	Negative difference ($A > B$)		Positive difference ($A < B$)		Equal difference ($A = B$)		p-value
Compared pairs	Number	Ranks	Number	Ranks	Number	Ranks	
$A = DM, B = P2$	0	0	34	595	2	-	< 0.0001
$A = DM, B = SAA$	0	0	35	630	1	-	< 0.0001
$A = P2, B = SAA$	5	31	4	31	27	-	0.3047

DM: deterministic model solved via CPLEX; P2: P2 model solved via CPLEX; SAA: sampling average approximation method

6.4. *Evaluation of valid inequalities*

To evaluate the proposed valid inequalities, numerical experiments are performed on realistic EOL products instances and randomly generated instance sets with up to 10 products, 200 tasks and 46 workstations.

Table 9 reports the computational results of realistic EOL product instances, in which columns 2 and 3 represent the instance parameters and objective values. Columns 3-6 represent the computational times of model P2, P2 with valid inequality 1 (P2+VI1), P2 with valid inequality 2 (P2+VI2) and P2 with valid inequalities 1 and 2 (P2+VI1+VI2), which are denoted by T_{P2}, T_{P2+VI1}, T_{P2+VI2}, and $T_{P2+VI1+VI2}$, respectively. From

Table 9, we can see that the average computational times of these 4 models are 2.56, 1.75, 1.74, and 1.46 seconds, respectively. It indicates that the proposed two inequalities are valid.

Table 9: Computational results of valid inequalities on realistic EOL product instances

Instance	(P, I, J)	Obj	$T_{P2}(s)$	$T_{P2+VI1}(s)$	$T_{P2+VI2}(s)$	$T_{P2+VI1+VI2}(s)$
1	(2,20,10)	14.2	0.10	0.09	0.07	0.05
2	(2,30,10)	19.0	0.19	0.10	0.10	0.11
3	(3,40,15)	22.8	0.35	0.24	0.17	0.19
4	(3,53,15)	22.8	0.43	0.31	0.22	0.22
5	(4,63,20)	29.6	2.35	1.58	1.98	1.09
6	(4,78,20)	27.8	1.14	0.75	0.53	0.50
7	(5,88,25)	34.0	2.83	1.55	1.51	1.45
8	(5,108,25)	36.4	4.47	2.40	1.68	1.66
9	(6,118,30)	43.8	4.99	3.83	4.51	3.45
10	(6,145,30)	44.8	8.72	6.66	6.70	5.92
Average	-	29.5	2.56	1.75	1.74	1.46

Moreover, the results of random instances are reported in Figure 11 and Table E4 in Appendix E. Specifically, the curves in Figure 11 represent the computational time ratios that are calculated via the formula $R_i = T_i/T_{P2} \times 100\%$, where $i = P2$, $P2+VI1$, $P2+VI2$ and $P2+VTI1+VI2$. The results show that each proposed inequality can save computational time, and P2+VI1+VI2 (P3) is the most efficient. Specifically, the average computational time ratios of P2+VI1, P2+V2 and P2+VI1+VI2 (P3) are only 33.93%, 33.80% and 25.17%, respectively. Therefore, the performance of model P3, i.e., P2+VI1+VI2 (P3), is verified.

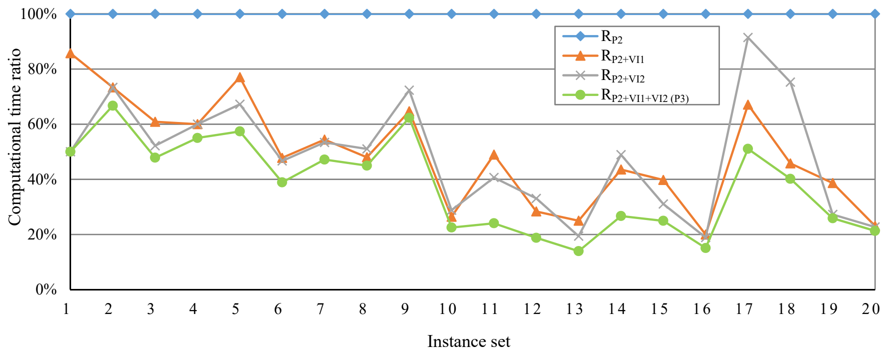


Figure 11: Comparison of valid inequalities

6.5. Evaluation of the lifted CS method

The proposed lifted CS method is first tested on realistic EOL product instances, and then tested on 20 small-scale instance sets with up to 10 products, 200 tasks and

46 workstations and 20 large-scale instance sets with up to 20 products, 400 tasks and 86 workstations. The proposed method is compared with P3 solved by CPLEX (version 12.9) and a classic CS method.

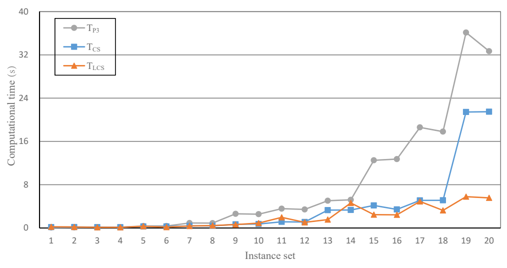
Table 10 reports the results on instances based on realistic products. Columns 3-5 illustrate the computational times of the CPLEX, the classic CS method and the lifted CS method, which are represented by T_{P3}, T_{CS} and T_{LCS}, respectively. From Table 10, we can observe that all instances are solved within a short time, and the average computational times of CPLEX, classic CS method, and lifted CS method are 1.46, 0.71, and 0.55 seconds, respectively. It implies that the lifted CS method can efficiently solve the instances and can reduce 62.3% of the computational time required by CPLEX on average.

Table 10: Computational results of lifted cut-and-solve method on realistic EOL product instances

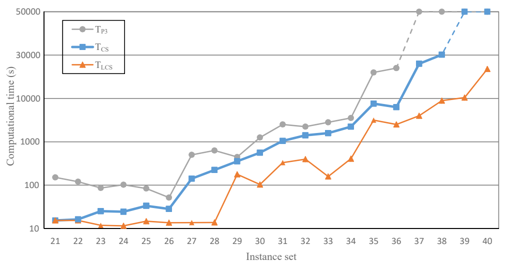
Instance set	(P, I, J)	*Obj*	$T_{P3}(s)$	$T_{CS}(s)$	$T_{LCS}(s)$
1	(2, 20, 10)	14.2	0.05	0.11	0.06
2	(2, 30, 10)	19.0	0.11	0.16	0.11
3	(3, 40, 15)	22.8	0.19	0.27	0.10
4	(3, 53, 15)	22.8	0.22	0.30	0.19
5	(4, 63, 20)	29.6	1.09	1.08	0.81
6	(4, 78, 20)	27.8	0.50	0.38	0.35
7	(5, 88, 25)	34.0	1.45	0.89	0.50
8	(5, 108, 25)	36.4	1.66	0.70	0.69
9	(6, 118, 30)	43.8	3.45	1.07	0.87
10	(6, 145, 30)	44.8	5.92	2.19	1.78
Average	-	29.5	1.46	0.71	0.55

The experimental results on random instances are illustrated in Figure 12, and sub-figures (a) and (b) report the results of small-scale and large-scale instances, respectively. Sub-figure (a) shows that the computational times of all methods are less than 40 seconds and the curves of T_{CS} and T_{LCS} are below the T_{P3} curve in most cases, except for instances 1-4. Table E5 in Appendix E indicates that the average computational times of CPLEX, classical CS method and lifted CS method are 7.79, 3.64 and 1.84 seconds, respectively. This result signifies that the lifted CS method globally outperforms CPLEX and the classical CS method. For large-scale instances, sub-figure (b) shows that the computational times of all 3 methods increase with instance set size, but the computational time of the lifted CS method increases much slowly than the classical CS method and CPLEX. Notably, CPLEX and the classical CS method cannot propose feasible solutions for instance sets 37-40 and instance sets 39-40 within 50,000 seconds.

The computational times of these instances are presented by dotted lines. Table E6 in Appendix E shows that the average computational times of CPLEX, the classical CS and the lifted CS methods are 6356.84, 1657.09 and 673.23 seconds, respectively. For instance sets 1-36, the lifted CS method needs only 17.53% and 40.65% of the computational times required by the CPLEX and the classic CS method, respectively.



(a) Instances with up to 10 products, 200 tasks and 46 available workstations



(b) Instances with up to 20 products, 400 tasks and 86 available workstations

Figure 12: Comparison with P3 and the classic CS method

In summary, the proposed lifted CS method is more efficient than CPLEX and the classical CS method, particularly for large-scale instances.

7. Conclusions and future research work

This paper examines a new flexible multi-product DLBP by considering identical parts of multiple products and uncertain task processing times, in which only partial information of their probability distribution is known. To efficiently solve the problem, an integrated approach is devised in which 1) the problem is formulated using an appropriate joint chance-constrained model, 2) a distribution-free model is proposed based on problem analysis, 3) efficient valid inequalities are developed to reduce the solution space of the distribution-free model, and 4) a new lifted CS method is provided to solve the problem. Numerical experiments demonstrate the high performance of the integrated approach.

In this study, we consider uncertain task processing times in the disassembly line balancing problem but ignore that uncertain hazardous risk of a component may coexist in reality. We focus on the line balancing problem, while the sequences of tasks that may result in operation direction changes are ignored. Consequently, the cost of operation direction changes are not considered. Due to the NP-hardness of the problem, the proposed exact method is still time-consuming for large-size instances. Therefore, one future research direction is to consider uncertain task times and the hazardous risk of a component. Another possible research direction is integrating the sequence of disassembly tasks into the line balancing problem. In addition, developing metaheuristics or problem-specific heuristics may be another promising direction. Besides, the constraint programming (CP) model, as one of the powerful paradigms for solving combinatorial optimization problems, may be applied to solve the studied problem.

Acknowledgements

This work was partially supported by the China Scholarship Council (202006650025), the China Franco-Chinese Cai Yuanpei project (Grant No. 43956QF), the National Natural Science Foundation of China (Grant No. 71871159, 72071144), the Humanities and Social Science Foundation of the Chinese Ministry of Education (21YJA630096), and the 2nd Fujian Young Eagle Program Youth Top Talent Program.

References

Agrawal, S. et al. (2008). A collaborative ant colony algorithm to stochastic mixed-model u-shaped disassembly line balancing and sequencing problem. *International Journal of Production Research*, *46*, 1405–1429.

Altekin, F. T., Kandiller, L., & Ozdemirel, N. E. (2008). Profit-oriented disassembly-line balancing. *International Journal of Production Research*, *46*, 2675–2693.

Bentaha, M. L., Battaïa, O., & Dolgui, A. (2012). A stochastic formulation of the disassembly line balancing problem. In *IFIP International Conference on Advances in Production Management Systems* (pp. 397–404). Springer.

Bentaha, M. L., Battaïa, O., & Dolgui, A. (2013). A decomposition method for stochastic partial disassembly line balancing with profit maximization. In *2013 IEEE International Conference on Automation Science and Engineering (CASE)* (pp. 404–409). IEEE.

Bentaha, M. L., Battaïa, O., & Dolgui, A. (2014). A sample average approximation method for disassembly line balancing problem under uncertainty. *Computers & Operations Research*, *51*, 111–122.

Bentaha, M. L., Battaïa, O., Dolgui, A., & Hu, S. J. (2015). Second order conic approximation for disassembly line design with joint probabilistic constraints. *European Journal of Operational Research*, *247*, 957–967.

Chica, M., Cordon, O., Damas, S., & Bautista, J. (2010). Multiobjective constructive heuristics for the 1/3 variant of the time and space assembly line balancing problem: ACO and random greedy search. *Information Sciences*, *180*, 3465–3487.

Climer, S., & Zhang, W. (2006). Cut-and-solve: An iterative search strategy for combinatorial optimization problems. *Artificial Intelligence*, *170*, 714–738.

Cucchiella, F., DAdamo, I., Rosa, P., & Terzi, S. (2016). Scrap automotive electronics: A mini-review of current management practices. *Waste Management & Research*, *34*, 3–10.

Edis, E. B. (2021). Constraint programming approaches to disassembly line balancing problem with sequencing decisions. *Computers & Operations Research*, *126*, 105111.

Environmental Affairs Division, T. M. C. (2017). Vehicle recycling. `https://global.toyota/pages/global_toyota/sustainability/report/kururisa_en.pdf`.

Fang, Y., Chu, F., Mammar, S., & Che, A. (2013). An optimal algorithm for automated truck freight transportation via lane reservation strategy. *Transportation Research Part C: Emerging Technologies*, *26*, 170–183.

Fang, Y., Liu, Q., Li, M., Laili, Y., & Pham, D. T. (2019). Evolutionary many-objective optimization for mixed-model disassembly line balancing with multi-robotic workstations. *European Journal of Operational Research*, *276*, 160–174.

Fang, Y., Ming, H., Li, M., Liu, Q., & Pham, D. T. (2020a). Multi-objective evolutionary simulated annealing optimisation for mixed-model multi-robotic disassembly line balancing with interval processing time. *International Journal of Production Research*, *58*, 846–862.

Fang, Y., Xu, H., Liu, Q., & Pham, D. T. (2020b). Evolutionary optimization using epsilon method for resource-constrained multi-robotic disassembly line balancing. *Journal of Manufacturing Systems*, *56*, 392–413.

Gadegaard, S. L., Klose, A., & Nielsen, L. R. (2018). An improved cut-and-solve algorithm for the single-source capacitated facility location problem. *EURO Journal on Computational Optimization*, *6*, 1–27.

García, S., Molina, D., Lozano, M., & Herrera, F. (2009). A study on the use of nonparametric tests for analyzing the evolutionary algorithms behaviour: a case study on the cec2005 special session on real parameter optimization. *Journal of Heuristics*, *15*, 617–644.

He, J., Chu, F., Zheng, F., Liu, M., & Chu, C. (2020). A multi-objective distribution-free model and method for stochastic disassembly line balancing problem. *International Journal of Production Research*, *58*, 5721–5737.

Ilgin, M. A., Akçay, H., & Araz, C. (2017). Disassembly line balancing using linear physical programming. *International Journal of Production Research*, *55*, 6108–6119.

Koc, A., Sabuncuoglu, I., & Erel, E. (2009). Two exact formulations for disassembly line balancing problems with task precedence diagram construction using an AND/OR graph. *IIE Transactions*, *41*, 866–881.

Kucukkoc, I. (2020). Balancing of two-sided disassembly lines: Problem definition, MILP model and genetic algorithm approach. *Computers & Operations Research*, *124*, 105064.

Lambert, A. (1999). Linear programming in disassembly/clustering sequence generation. *Computers & Industrial Engineering*, *36*, 723–738.

Li, J., Chen, X., Zhu, Z., Yang, C., & Chu, C. (2019). A branch, bound, and remember algorithm for the simple disassembly line balancing problem. *Computers & Operations Research*, *105*, 47–57.

Li, Z., Çil, Z. A., Mete, S., & Kucukkoc, I. (2020). A fast branch, bound and remember algorithm for disassembly line balancing problem. *International Journal of Production Research*, *58*, 3220–3234.

Liu, X., Chu, F., Zheng, F., Chu, C., & Liu, M. (2022). Distributionally robust and risk-averse optimisation for the stochastic multi-product disassembly line balancing problem with workforce assignment. *International Journal of Production Research*, *60*, 1973–1991.

Luedtke, J., & Ahmed, S. (2008). A sample approximation approach for optimization with probabilistic constraints. *SIAM Journal on Optimization*, *19*, 674–699.

Ma, Y.-S., Jun, H.-B., Kim, H.-W., & Lee, D.-H. (2011). Disassembly process planning algorithms for end-of-life product recovery and environmentally conscious disposal. *International Journal of Production Research*, *49*, 7007–7027.

McGovern, S., & Gupta, S. (2007a). Combinatorial optimization analysis of the unary np-complete disassembly line balancing problem. *International Journal of Production Research*, *45*, 4485–4511.

McGovern, S. M., & Gupta, S. M. (2007b). A balancing method and genetic algorithm for disassembly line balancing. *European Journal of Operational Research*, *179*, 692–708.

Ndubisi, N. O., Nygaard, A., & Chunwe N, G. (2020). Managing sustainability tensions in global supply chains: specific investments in closed-loop technology vs blood metals. *Production Planning & Control*, *31*, 1005–1013.

Ng, M. (2014). Distribution-free vessel deployment for liner shipping. *European Journal of Operational Research*, *238*, 858–862.

Özceylan, E., Kalayci, C. B., Güngör, A., & Gupta, S. M. (2019). Disassembly line balancing problem: a review of the state of the art and future directions. *International Journal of Production Research*, *57*, 4805–4827.

Paksoy, T., Güngör, A., Özceylan, E., & Hancilar, A. (2013). Mixed model disassembly line balancing problem with fuzzy goals. *International Journal of Production Research*, *51*, 6082–6096.

Qiu, F., & Wang, J. (2014). Chance-constrained transmission switching with guaranteed wind power utilization. *IEEE Transactions on Power Systems*, *30*, 1270–1278.

Ren, Y., Yu, D., Zhang, C., Tian, G., Meng, L., & Zhou, X. (2017). An improved gravitational search algorithm for profit-oriented partial disassembly line balancing problem. *International Journal of Production Research*, *55*, 7302–7316.

Sewell, E. C., & Jacobson, S. H. (2012). A branch, bound, and remember algorithm for the simple assembly line balancing problem. *INFORMS Journal on Computing*, *24*, 433–442.

Tang, Y., Zhou, M., Zussman, E., & Caudill, R. (2002). Disassembly modeling, planning, and application. *Journal of Manufacturing Systems*, *21*, 200–217.

Thierry, M., Salomon, M., Van Nunen, J., & Van Wassenhove, L. (1995). Strategic issues in product recovery management. *California Management Review*, *37*, 114–136.

Wu, P., Che, A., Chu, F., & Fang, Y. (2017). Exact and heuristic algorithms for rapid and station arrival-time guaranteed bus transportation via lane reservation. *IEEE Transactions on Intelligent Transportation Systems*, *18*, 2028–2043.

Xu, F., Li, Y., & Feng, L. (2019). The influence of big data system for used product management on manufacturing–remanufacturing operations. *Journal of Cleaner Production*, *209*, 782–794.

Yang, Z., Chu, F., & Chen, H. (2012). A cut-and-solve based algorithm for the single-source capacitated facility location problem. *European Journal of Operational Research*, *221*, 521–532.

Yin, T., Zhang, Z., Zhang, Y., Wu, T., & Liang, W. (2022). Mixed-integer programming model and hybrid driving algorithm for multi-product partial disassembly line balancing problem with multi-robot workstations. *Robotics and Computer-Integrated Manufacturing*, *73*, 102251.

Zheng, F., He, J., Chu, F., & Liu, M. (2018). A new distribution-free model for disassembly line balancing problem with stochastic task processing times. *International Journal of Production Research*, *56*, 7341–7353.

Appendix A.

Before demonstrating Proposition 4.1, Lemmas 1 and 2 are introduced (refer to Ng (2014)).

Firstly, Lemma 1 is introduced below to describe the relationship between Z_i and b_i.

Lemma 1. *For any $\lambda > 0$, the following inequality must hold:*

$$E[e^{\lambda Z_i}] \leq 1 + \frac{E[Z_i^2]}{b_i^2}(e^{\lambda b_i} - \lambda b_i - 1) \tag{A.1}$$

Proof. By the definition of the mean task processing time, we have $\mu_i = E[t_i] = E[\mu_i(1+Z_i)]$. Thus, it follows that $E[Z_i] = 0$. Then, $e^{\lambda b_i}$ can be expressed as $1 + \lambda b_i + \sum_{m=2}^{\infty} \frac{(\lambda b_i)^m}{m!}$ (according to the Taylor series expansion). Because $Z_i \leq b_i$ and $b_i > 0$,

$$\begin{aligned} E[e^{\lambda Z_i}] &= 1 + \sum_{m=2}^{\infty} \frac{\lambda^m E[Z_i^m]}{m!} \\ &\leq 1 + E[Z_i^2] \sum_{m=2}^{\infty} \frac{\lambda^m E[b_i^{m-2}]}{m!} \\ &= 1 + \frac{E[Z_i^2]}{b_i^2} \sum_{m=2}^{\infty} \frac{\lambda^m b_i^m}{m!} \\ &= 1 + \frac{E[Z_i^2]}{b_i^2}(e^{\lambda b_i} - \lambda b_i - 1) \end{aligned} \tag{A.2}$$

□

Next, Lemma 2 is proposed to depict the relationship between $E[Z_i^2]$ and σ_i^2.

Lemma 2. *With the given standard deviation of task processing time σ_i, we have $E[Z_i^2] = \frac{\sigma_i^2}{\mu_i^2}$.*

Proof. Let $D[t_i]$ denote the variance of processing time of task i, we have:

$$\begin{aligned} D[t_i] &= D[\mu_i(1 + Z_i)] \\ &= \mu_i^2 D[Z_i] \end{aligned} \tag{A.3}$$

Note that $E[Z_i] = 0$, hence $D[Z_i] = E[Z_i^2] - (E[Z_i])^2 = E[Z_i^2]$. Accordingly, the following equation

$$D[t_i] = \mu_i^2 E[Z_i^2] \tag{A.4}$$

must hold, thus $E[Z_i^2] = \frac{\sigma_i^2}{\mu_i^2}$. □

Now, we provide the proof of Proposition 4.1.

Proof. Given a feasible solution x_{ij} obtained by (15), i.e., $\sum_{i\in I}(\mu_i+\nu_i)x_{ij} \le C,\ \forall j \in J$, if $\sum_{i\in I}\mu_i(1+Z_i)x_{ij} > C$, it must follow that $\sum_{i\in I}\mu_i Z_i x_{ij} > \sum_{i\in I}\nu_i x_{ij}$. Moreover, if $\mu_i Z_i > \nu_i$, it must follows that $\sum_{i\in I}\mu_i Z_i x_{ij} > \sum_{i\in I}\nu_i x_{ij}$. Hence, for simplicity, $\sum_{i\in I}\mu_i Z_i x_{ij} > \sum_{i\in I}\nu_i x_{ij}$ is conservatively approximated by $\mu_i Z_i > \nu_i$, so the following inequality must hold:

$$Pr\left(\sum_{i\in I}\mu_i(1+Z_i)x_{ij} > C\right) \le Pr\left(\mu_i Z_i > \nu_i\right),\ \forall j \in J \tag{A.5}$$

Obviously, we have $Pr\left(\mu_i Z_i > \nu_i\right) = Pr\left(e^{\lambda Z_i} > e^{\lambda\nu_i/\mu_i}\right)$. Then, using the Markov inequality: $Pr(X \ge a) \le \frac{E(X)}{a}$, where $X \ge 0$, and $a > 0$. Let $X = e^{\lambda Z_i}$ and $a = e^{\lambda\nu_i/\mu_i}$. With Lemma 1, we have

$$\begin{aligned} Pr\left(e^{\lambda Z_i} > e^{\lambda\nu_i/\mu_i}\right) &\le e^{-\lambda\nu_i/\mu_i}E[e^{\lambda Z_i}] \\ &\le e^{-\lambda\nu_i/\mu_i}\left(1+\frac{E[Z_i^2]}{b_i^2}(e^{\lambda b_i}-\lambda b_i-1)\right) \end{aligned} \tag{A.6}$$

Since A6 is valid for any $\lambda > 0$, the following inequality must holds:

$$Pr\left(\sum_{i\in I}\mu_i(1+Z_i)x_{ij} > C\right) \le \min_{\lambda>0} e^{-\lambda\nu_i/\mu_i}\left(1+\frac{E[Z_i^2]}{b_i^2}(e^{\lambda b_i}-\lambda b_i-1)\right),\ \forall j \in J \tag{A.7}$$

Recall constraints (14) that can be rewritten as:

$$Pr\left(\sum_{i\in I}\mu_i(1+Z_i)x_{ij} > C\right) < \beta_j,\ \forall j \in J \tag{A.8}$$

We can always find a λ such that the right sides of (A7) and (A8) to be equal, we can obtain

$$\min_{\lambda>0} e^{-\lambda\nu_i/\mu_i}\left(1+\frac{E[Z_i^2]}{b_i^2}(e^{\lambda b_i}-\lambda b_i-1)\right) - \beta_j = 0,\ \forall j \in J \tag{A.9}$$

So any ν_i satisfies A9, then any solution x_{ij} of P2 must satisfy the individual chance constraint (14). □

Appendix B.

Proof. Valid inequality I indicates that the number of opened workstations of an optimal solution of $P2$ does not exceed $\sum_{p\in P}|J_p|$. Suppose that $\sum_{j\in J}y_j^{''}$ denotes the optimal number of workstations of $P2$ and $\sum_{j\in J}y_j^{''} > \sum_{p\in P}|J_p|$. Let $\sum_{j\in J}y_j^{'}$ denote the optimal number of workstations of $P2^{'}$, and we have $\sum_{p\in P}|J_p| \ge \sum_{j\in J}y_j^{'}$. Thus, $\sum_{j\in J}y_j^{''} > \sum_{j\in J}y_j^{'}$. As $P2^{'}$ is the relaxed model of $P2$, we can always find another optimal number of workstations of $P2$, denoted as $\sum\limits_{j\in J} y_j^*$ that is smaller than or equal to $\sum\limits_{j\in J} y_j^{'}$. Apparently, we have $\sum\limits_{j\in J} y_j^* \leq \sum\limits_{j\in J} y_j^{'} < \sum\limits_{j\in J} y_j^{''}$. So it is contradictory to the assumption that $\sum\limits_{j\in J} y_j^{''}$ is the optimal number of workstations of $P2$. Thus, (17) is a valid inequality of $P2$. □

Appendix C.

Proof. Valid inequality II means that task i cannot be assigned to the workstations smaller than n_{pi} or larger than $\sum\limits_{p\in P} |J_p| - n_{si} + 1$. Suppose that there exists an optimal solution ζ of model $P2$ with $x_{ij} \neq 0$, in which $j < n_{pi}$ or $j > \sum\limits_{p\in P} |J_p| - n_{si} + 1$. Naturally, we can always find a solution $\zeta^{'}$ that the processors of task i needs at least $n_{pi} - 1$ workstations. Thus, it can be observed that solution $\zeta^{'}$ requires less workstations than the optimal solution ζ. It is apparently contradictory with our former assumption. Hence, (18) is valid for model $P2$. □

Appendix D.

The model with SAA method mainly refers to Luedtke & Ahmed (2008) and Qiu & Wang (2014). Before the approximated model, some new parameters are introduced below:

Ω : the set of randomly generated Monte Carlo samples;

η : index of each sample;

$z_{\eta j}$: binary variable, equal to 1 if the workload of workstation $j \in J$ exceeds the cycle time under sample $\eta \in \Omega$, 0 otherwise ;

M: a big positive number;

The approximated model using SAA method is proposed in the following:

$$\textbf{P4:} \quad \min \quad C_F \sum_{j\in J} y_j + C_H \sum_{i\in H} \sum_{j\in J} x_{ij}$$

$$s.t. \quad (2) - (7), (9), (10)$$

$$\sum_{i\in I} t_i(\eta) x_{ij} - M z_{\eta j} \leq C, \ \forall j \in J, \ \forall \eta \in \Omega \tag{D.1}$$

$$\sum_{\eta\in\Omega} \sum_{j\in J} z_{\eta j} \leq \lfloor |\Omega| \alpha \rfloor \tag{D.2}$$

$$z_{\eta j} \in \{0, 1\}, \ \forall j \in J, \ \forall \eta \in \Omega \tag{D.3}$$

Constraints $D.1$ guarantee that the workloads of all workstations will not exceed the cycle time under all samples by adding a binary variable $z_{\eta j}$. Constraint $D.2$ restricts the sum value of $z_{\eta j}$, which is determined by the scale of Monte Carlo samples $|\Omega|$ and the probability α, where $\lfloor x \rfloor$ means the largest integer that does not exceed x. Constraints $D.3$ define the domains of decision variables.

Appendix E.

The detailed computational results for random instances with three uncertainty levels $(b_i, E[Z_i^2])$ in Section 6.3 are reported in Tables E1-E3. The detailed computational results of valid inequality evaluation in Section 6.4 are reported in Table E4. The detailed computational results for evaluating the lifted cut-and-solve method in Section 6.5 are reported in Tables E5-E6.

Table E.1: The results of random instances with $(b_i = 0.1, E[Z_i^2] = 0.01)$

		DM		SAA		P2	
Instance	(P, I, J)	Obj_{DM}	$T_{DM}(s)$	Obj_{SAA}	$T_{SAA}(s)$	Obj_{P2}	$T_{P2}(s)$
1	(2,20,6)	17.8	0.08	17.8	0.12	17.8	0.13
2	(2,20,10)	17.8	0.10	18.4	0.22	17.8	0.12
3	(2,40,10)	14.8	0.13	16.6	0.33	16.6	0.15
4	(2,40,14)	14.8	0.10	16.6	0.36	16.6	0.13
5	(4,40,14)	29.6	0.41	32.6	6.16	32.6	0.37
6	(4,40,18)	29.6	0.50	32.6	10.92	32.6	0.39
7	(4,80,18)	29.0	0.92	30.8	11.10	30.8	1.13
8	(4,80,22)	29.0	0.92	30.8	26.06	30.8	1.01
9	(6,60,22)	45.6	2.02	49.2	1538.67	49.2	2.56
10	(6,60,26)	45.6	2.57	49.8	3708.64	49.2	2.60
11	(6,120,26)	42.0	3.58	45.6	5568.77	45.6	14.00
12	(6,120,30)	42.0	5.93	45.0	6918.61	45.0	14.95
Average	-	29.8	1.44	32.2	1482.50	32.1	3.13

Table E.2: The results of random instances with ($b_i = 0.2, E[Z_i^2] = 0.05$)

		DM		SAA		P2	
Instance	(P, I, J)	Obj_{DM}	$T_{DM}(s)$	Obj_{SAA}	$T_{SAA}(s)$	Obj_{P2}	$T_{P2}(s)$
13	(2,20,6)	17.8	0.08	19.6	0.14	20.2	0.12
14	(2,20,10)	17.8	0.10	20.2	0.25	20.2	0.12
15	(2,40,10)	14.8	0.13	17.2	0.41	17.2	0.17
16	(2,40,14)	14.8	0.10	17.2	0.42	17.2	0.13
17	(4,40,14)	29.6	0.41	35.6	28.69	35.6	0.40
18	(4,40,18)	29.6	0.50	35.0	35.44	35.0	0.49
19	(4,80,18)	29.0	0.92	33.8	261.50	33.8	1.34
20	(4,80,22)	29.0	0.92	33.8	348.12	33.8	1.16
21	(6,60,22)	45.6	2.02	52.8	4641.25	52.2	7.85
22	(6,60,26)	45.6	2.57	52.2	6717.90	52.2	12.35
23	(6,120,26)	42.0	3.58	49.2	8182.52	49.2	15.87
24	(6,120,30)	42.0	5.93	49.2	9236.56	49.2	53.99
Average	-	29.8	1.44	34.7	2454.44	34.7	7.83

Table E.3: The results of random instances with ($b_i = 0.3, E[Z_i^2] = 0.10$)

		DM		SAA		P2	
Instance	(P, I, J)	Obj_{DM}	$T_{DM}(s)$	Obj_{SAA}	$T_{SAA}(s)$	Obj_{P2}	$T_{P2}(s)$
25	(2,20,6)	17.8	0.08	20.8	0.18	20.8	0.11
26	(2,20,10)	17.8	0.10	21.4	0.24	20.8	0.15
27	(2,40,10)	14.8	0.13	17.8	0.34	19.0	0.16
28	(2,40,14)	14.8	0.10	19.0	0.34	19.0	0.14
29	(4,40,14)	29.6	0.41	38.6	99.29	38.6	1.52
30	(4,40,18)	29.6	0.50	37.4	59.58	38.6	1.23
31	(4,80,18)	29.0	0.92	35.6	409.25	36.8	4.56
32	(4,80,22)	29.0	0.92	36.8	863.36	36.8	9.12
33	(6,60,22)	45.6	2.02	55.2	4875.63	55.2	11.79
34	(6,60,26)	45.6	2.57	56.4	9333.72	56.4	18.40
35	(6,120,26)	42.0	3.58	52.2	12005.20	52.2	20.30
36	(6,120,30)	42.0	5.93	51.6	12984.24	52.2	90.04
Average	-	29.8	1.44	36.9	3385.95	37.2	13.13

Table E.4: Computational results of valid inequalities on random instances

Instance	(P, I, J)	Obj	$T_{P2}(s)$	$T_{P2+VI1}(s)$	$T_{P2+VI2}(s)$	$T_{P2+VI1+VI2}(s)$
1	(2,20,6)	17.8	0.14	0.12	0.07	0.07
2	(2,20,10)	17.8	0.15	0.11	0.11	0.10
3	(2,40,10)	14.8	0.23	0.14	0.12	0.11
4	(2,40,14)	14.8	0.20	0.12	0.12	0.11
5	(4,40,14)	29.6	0.61	0.47	0.41	0.35
6	(4,40,18)	29.6	0.90	0.43	0.42	0.35
7	(4,80,18)	29.0	1.93	1.05	1.03	0.91
8	(4,80,22)	29.0	1.98	0.95	1.01	0.89
9	(6,60,22)	45.6	4.19	2.71	3.03	2.61
10	(6,60,26)	45.6	11.29	2.98	3.25	2.54
11	(6,120,26)	42.0	14.81	7.25	6.02	3.56
12	(6,120,30)	42.0	18.21	5.16	6.02	3.42
13	(8,80,30)	62.2	36.12	9.02	7.00	5.03
14	(8,80,34)	62.2	19.50	8.49	9.54	5.20
15	(8,160,34)	56.2	50.19	19.96	15.58	12.51
16	(8,160,38)	56.2	84.51	16.87	15.95	12.73
17	(10,100,38)	75.2	36.45	24.45	33.33	18.60
18	(10,100,42)	75.2	44.33	20.29	33.36	17.81
19	(10,200,42)	68.4	139.75	54.01	38.09	36.14
20	(10,200,46)	68.4	153.52	35.41	34.83	32.69
Average	-	44.1	30.95	10.50	10.46	7.79

Table E.5: Computational results for random instances with $|P| = 2 - 10$, $|I| = 20 - 200$ and $|J| = 6 - 46$

Instance set	(P, I, J)	Obj	$T_{P3}(s)$	$T_{CS}(s)$	$T_{LCS}(s)$
1	(2, 20, 6)	17.8	0.07	0.15	0.21
2	(2, 20, 10)	17.8	0.10	0.16	0.17
3	(2, 40, 10)	14.8	0.11	0.15	0.11
4	(2, 40, 14)	14.8	0.11	0.13	0.11
5	(4, 40, 14)	29.6	0.35	0.26	0.31
6	(4, 40, 18)	29.6	0.35	0.26	0.15
7	(4, 80, 18)	30	0.91	0.37	0.35
8	(4, 80, 22)	30	0.89	0.37	0.44
9	(6, 60, 22)	45.6	2.61	0.66	0.62
10	(6, 60, 26)	45.6	2.54	0.72	0.89
11	(6, 120, 26)	42.0	3.56	1.13	1.96
12	(6, 120, 30)	42.0	3.42	1.09	1.03
13	(8, 80, 30)	62.2	5.03	3.29	1.54
14	(8, 80, 34)	62.2	5.20	3.32	4.62
15	(8, 160, 34)	56.2	12.51	4.16	2.45
16	(8, 160, 38)	56.2	12.73	3.42	2.42
17	(10, 100, 38)	75.2	18.60	5.09	4.90
18	(10, 100, 42)	75.2	17.81	5.11	3.24
19	(10, 200, 42)	68.4	36.14	21.44	5.78
20	(10, 200, 46)	68.4	32.69	21.49	5.56
Average	-	44.18	7.79	3.64	1.84

Table E.6: Computational results for random instances with $|P| = 12 - 20$, $|I| = 120 - 400$ and $|J| = 46 - 86$

Instance set	(P, I, J)	*Obj*	$T_{P3}(s)$	$T_{CS}(s)$	$T_{LCS}(s)$
21	(12, 120, 46)	88.2	171.59	18.67	18.64
22	(12, 120, 50)	88.2	139.83	21.05	18.73
23	(12, 240, 50)	82.0	93.57	40.08	7.38
24	(12, 240, 54)	82.0	113.40	38.71	6.22
25	(14, 140, 54)	104.2	92.32	52.35	16.88
26	(14, 140, 58)	104.2	71.42	45.21	13.40
27	(14, 280, 58)	95.0	653.86	165.50	13.64
28	(14, 280, 62)	95.0	823.88	357.88	13.98
29	(16, 160, 62)	114.2	563.37	460.63	278.67
30	(16, 160, 66)	114.2	1505.53	722.52	114.81
31	(16, 320, 66)	109.2	3617.95	1299.23	435.62
32	(16, 320, 70)	109.2	3564.20	1875.17	504.30
33	(18, 180, 70)	132.0	3966.16	2315.86	221.47
34	(18, 180, 74)	132.0	5081.14	3153.85	511.58
35	(18, 360, 74)	132.2	18852.11	8524.85	4841.85
36	(18, 360, 78)	132.2	22182.63	7421.85	3754.55
37	(20, 200, 78)	156.2	-	26523.11	5527.60
38	(20, 200, 82)	156.2	-	30087.80	9526.51
39	(20, 400, 82)	156.0	-	-	10854.60
40	(20, 400, 86)	156.0	-	-	21672.45
Average	-	107.1	6356.84	1657.09	673.23

In the table, $'-'$ means that the optimal solution has not been found within 50000 seconds. Hence, the average value is obtained from sets 21 to 36.