Partial disassembly line balancing under uncertainty: robust optimisation models and an improved migrating birds optimisation algorithm

Partial Disassembly Line Balancing Under Uncertainty: Robust Optimization Models and an Improved Migrating Birds Optimization Algorithm

ARTICLE HISTORY
Compiled March 11, 2020

ABSTRACT
A partial disassembly line balancing problem under uncertainty is studied in this paper, which concerns the allocation of a sequence of tasks to workstations such that the overall profit is maximized. We consider the processing time uncertainty and develop robust solutions to accommodate it. The problem is formulated as a non-linear robust integer program, which is then converted into an equivalent linear program. Due to the intractability of such problems, the exact algorithms are only applicable to small scale instances. We develop an improved migrating birds optimization algorithm. Two enhancement techniques are proposed. The first one finds the optimal number of tasks to be performed for each sequence rather than random selection used in the literature; while the second one exploits the specific problem structure to construct effective neighborhoods. The numerical results show the strong performance of our proposal compared to CPLEX and the improved gravitational search algorithm (IGSA), especially for large scale problems. Moreover, the enhancement due to the proposed techniques is obvious across all instances considered.

KEYWORDS
partial disassembly line balancing problem; uncertain processing time; robust optimization; migrating birds optimization.

1. Introduction

Disassembly, remanufacturing and recycling are the critical processes of End of Life (EOL) product recovery. With disassembly being the first key step (Gupta and Taleb 1994), it is important to properly design disassembly lines so as to achieve efficient product recovery. The disassembly line balancing problem (DLBP) concerns the optimal assignment of disassembly tasks to a number of workstations under various constraints, such as those on precedence relationships and cycle time. The objective is to utilize the disassembly resources as efficiently as possible while satisfying demand (Güngör and Gupta 1999).

As an important problem in reverse logistics, DLBP has attracted growing attention in recent years (Boysen et al. 2007). Even though DLBP appears to be similar to the assembly line balancing problem (ALBP), it cannot simply be considered as the reverse of ALBP due to the physical and operational characteristics between them (Güngör and Gupta 1999). Unlike the convergent flow process (all parts must be assembled to become a single product) of assembly lines, the disassembly flow process is divergent in that a single product is broken down into multiple subassemblies/parts, and the disassembly process does not have to be performed completely. Furthermore, the condition of the products received for disassembly is unknown; the part qualities and quantities, and arrival times of the products cannot be controlled, which leads to obvious variations in disassembly task times (Altekin and Akkan

2012). For a detailed comparison of assembly and disassembly lines, please refer to the book by Lambert and Gupta (2005) and a recent review paper by Özceylan et al. (2019).

In order to design an efficient disassembly line, different and usually complex factors need to be considered, and many factors should be incorporated into the objective functions of DLBP (Güngör and Gupta 2002). It has been shown by (McGovern and Gupta 2007) that DLBP problems are NP-complete, and thus most of the works in the literature concern the heuristics or meta-heuristic methods. McGovern and Gupta (2007) proposed a formulation with five objectives, which include minimizing the number of workstations, ensuring the balance of workstation idle times, removing hazardous parts early, removing high-demand parts early, and minimizing the number of part removal direction changes. A genetic algorithm was developed to solve the problem. Ding et al. (2010) designed a Pareto-based ant colony algorithm for a multi-objective DLBP problem to minimize the number of workstations, ensure the balance, and satisfy the demand. Kalayci and Gupta (2013a) first introduced the sequence-dependent concept in DLBP, where task times may be influenced by the order in which they are performed. For a sequence-dependent disassembly line balancing problem (SDDLBP) with multiple objectives, ant colony optimization (Kalayci and Gupta 2013a), particle swarm optimization (Kalayci and Gupta 2013b), artificial bee colony optimization (Kalayci and Gupta 2013c), tabu search (Kalayci and Gupta 2014) and hybrid genetic algorithm (Kalayci and Gupta 2016) have all been proposed to solve these problems.

The above-mentioned studies all consider complete disassembly processes. However, with the increasing structural complexity of products, it is expensive and usually unnecessary to completely disassemble a product into individual parts; partial disassembly is more common in practice. The partial disassembly line balancing problem (PDLBP) has not received much attention in the literature. Most of existing researches about PDLBP aim to maximize the profit. Altekin et al. (2004) first considered a partial disassembly DLBP problem to recycle valuable parts. They developed two mathematical models to maximize the profit in each disassembly cycle as well as the whole planning horizon. Altekin et al. (2008) defined a profit-oriented PDLBP which aimed at maximizing the profit while simultaneously deciding the number of stations and the cycle time. Then they built a mixed integer programming model that was solved by a lower and upper-bound scheme. Recently, Ren et al. (2017) first gave the cycle time for the profit-oriented PDLBP, and solved it with an improved gravitational search algorithm. Kalaycılar et al. (2016) studied a PDLBP problem with a fixed number of workstations. They developed a mixed integer programming model and proposed some upper and lower bounding procedures to assign the tasks and maximize revenue. The task failure in partial disassembly lines was considered by Altekin and Akkan (2012), who proposed a predictive-reactive approach to re-select and re-assign tasks after the failure so that the line was rebalanced.

Note that all the literature reviewed so far concerns various DLBP problems in a deterministic setting. However, parameters could be well uncertain in real life. For instance, the disassembly task time could vary due to the particularities of manual operations, the heterogeneity among operators may well exist, unexpected events could take place and have an impact on the disassembly process, and etc. Recently some researchers investigated the PDLBP problems in which the time of each disassembly task is a random variable with known probability distribution. To accommodate the uncertainty, the cycle time constraint was relaxed but required to be satisfied above a predetermined probability level. Bentaha et al. (2013) first constructed a chance constrained program formulation for such a problem to maximize the profit, and solved the model with the interior-point algorithm of CPLEX. The same problem was solved by a lower and upper-bounding scheme based on the second order cone programming and convex piecewise linear approximation in Bentaha et al. (2015). Bentaha et al. (2014, 2018) considered the penalty costs incurred due to cycle time constraint violation. Bentaha et al. (2014) formulated this problem into a stochastic program, and proposed Monte Carlo sampling based exact solution method to maximize the line profit. In order to maximize the revenue under task time uncertainty, an exact solution approach combining the L-shaped algorithm and Monte Carlo sampling techniques was developed by Bentaha et al. (2018).

Robust optimization is widely used in dealing with uncertainties when the probability distribution of the random variables is unknown (Gorissen et al. 2015). The uncertainties are described in the form of user-specified uncertainty sets and the aim is to find solutions that are feasible for all realizations of the uncertain parameters within the given set (Bertsimas et al. 2011). The solutions obtained thus have strong robustness and the sensitivity to parameter changes is low. Robust optimization was first proposed by Soyster (1973) to solve the parameter uncertainty in linear programming, but the solutions derived were too conservative for the sake of ensuring robustness. To address the over-conservativeness, Bertsimas and Sim (2003, 2004) developed a new robust optimization method (called B&S method hereafter) using the budget uncertainty set to describe the uncertain parameters. This uncertainty set is based on the relative value of the uncertainty parameter deviation, which can describe the parameter fluctuation more accurately, and it is more suitable to describe the uncertain task processing time.

The B&S method has clear advantages over the alternative approaches that are based on other uncertainty sets (e.g., ellipsoid uncertainty set (El Ghaoui et al. 1998)). Firstly its robust counterpart model takes a linear form, and it can be directly applied to discrete optimization problems. Secondly, this method offers full control on the degree of conservatism. A parameter Γ that limits the number of uncertain data being considered is specified by the decision maker, which in turn adjusts the level of conservativeness of the solutions. In other words, this approach aims to find the optimal solutions when at most Γ uncertain parameters are allowed to simultaneously change. Hazır and Dolgui (2013) and Pereira and Álvarez-Miranda (2018) have applied this approach to assembly line balancing problems. To the best of our knowledge, the B&S approach has not yet been applied to any DLBP problems.

Before proceeding we would like to highlight that robust optimization is in clear contrast to stochastic programming that seeks optimal solutions in some probabilistic sense, which requires the knowledge of the distributions. This requirement can be relaxed and many authors have considered distribution free stochastic programming approaches, which leads to a related but different research area, the distributionally robust optimization problems (see for example Bertsimas and Popescu (2005); Delage and Ye (2010); Adulyasak and Jaillet (2015); Zhang et al. (2018)). In these problems the distribution of the random parameters is unknown. Instead, they introduce a family of distributions that are assumed to include the true (but unknown) distribution. All the distributions in this set satisfy some given constraints, such as linear constraints on moments (Bertsimas and Popescu 2005). The objective is to optimize the worst case performance of the selected cost function over the distribution set. The distributionally robust optimization problems are different from the robust optimization problems in that the former only require the constraints are satisfied by a pre-defined probability (though for all distributions in the given set).

In this paper, we consider a partial disassembly line balancing problem under uncertainty (PDLBP-U), where the operational time for each disassembly task is uncertain but takes values from a known interval. Our aim is to find the minimum number of workstations for the given cycle time requirement, and in the meantime to maximize the profit of the disassembly line. We formulate the problem into a non-linear robust integer program. Following the B&S approach, the non-linear program is converted into a corresponding linear robust model which can be solved directly by the CPLEX solver. However, an optimal solution from CPLEX can be hardly found in polynomial time especially for larger scale problems. To this end, we develop a heuristic method based on *migrating birds optimization*(MBO), which is a novel nature-inspired metaheuristic algorithm proposed by Duman et al. (2012). It is suitable for discrete combinatorial optimization problems, and has been applied to various problems including the closed loop layout(Niroomand et al. 2015), the task allocation problems (Dindar 2017) and the flow shop problems (Sioud and Gagné 2018; Meng et al. 2018).

To summarize, our contributions are two-fold. Firstly, the probability distribution of the uncertain task time is often difficult to obtain in practice. Instead, the possible deviation from the standard operation time is much easier to estimate. In light of this we propose to use the budget uncertainty set to describe the uncertainty and develop a robust model following the B&S method. This approach is flexible in that the level of robustness can be adjusted by the decision-maker as required. Secondly, we propose an improved MBO algorithm to solve the PDLBP-U problem. Unlike the random selection of tasks to be performed for each sequence in the literature, we propose to find the optimal number of tasks for partial disassembly. We also propose effective neighborhoods to enhance the performance of the basic MBO algorithm. Our computational experiments over a wide range of instances show the clear and significant improvement due to these techniques.

The remaining of the paper is organized as follows. The problem description and the robust optimization models are presented in Section 2. Section 3 describes in detail the proposed heuristic method for the problem concerned. The computational experiments are reported in Section 4. Finally, conclusions and future research directions are provided in Section 5.

2. Problem description and robust optimization models

2.1. *The problem*

A single type of products are partially disassembled via a straight disassembly line. We aim to assign a set of disassembly tasks $i \in \{1, 2, ..., I\}$ to an ordered sequence of workstations $k \in \{1, 2, ..., K\}$, while satisfying cycle time and precedence relationships constraints. A single task cannot be divided between two workstations, and it may result in removal of one or more parts. It is also possible that no parts are released after a task. Denote by c_i and r_i the cost and revenue associated to task i. If a task does not release any parts, the revenue is zero. Cost is also incurred for opening and operating a workstation. In this study, the uncertainty of disassembly task times is taken into account. The processing time $\tilde{t}_i$ of task i is unknown but takes its value from a bounded interval $[t_i,\bar{t}_i]$, where the lower bound t_i is assumed as the nominal time of task i, which is the standard processing time of task i. $\tilde{t}_i$ is allowed to deviate the nominal time due to the particularities of manual operations and random event, and the maximum deviation value is d_i, so the upper bound $\bar{t}_i$=t_i+d_i. Let $d_i = p \cdot t_i$, where p is the deviation coefficient from the nominal time that determines the size of the interval. PDLBP-U looks for sequences that are feasible in all possible realizations at a predetermined uncertainty level, and in the mean time maximize the overall profit. We assume that the supply of products to be disassembled is infinite, and every product contains all their parts with no addition or removing of components.

The precedence relationships among tasks are depicted by the task-based AND/OR precedence diagram, which contains AND precedence relation and OR precedence relation. The AND precedence relation is the common precedence relation. For a task i, all of its predecessors must have been completed before it can start itself. As shown in Figure 1(a), tasks 1,2 and 3 are AND predecessors of task 4. The OR precedence relation is expressed by an arc. For task i, at least one of its predecessors must have been finished before it may start. In Figure 1(b), task 5, 6 and 7 are OR predecessors of task 8.

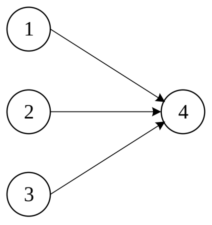


Figure 1(a). AND precedence relation.

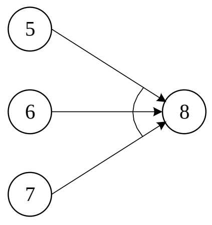


Figure 1(b). OR precedence relation.

2.2. *The models*

The notation to be used in the mathematical models is given in Table 1.

Table 1. The Notation.

Notation	Definition
I	number of disassembly tasks that is indexed by i
K	upper limit on the number of workstation that is indexed by k
C	operating cost per unit time for a workstation
F	fixed cost of opening a workstation
CT	cycle time
$PAND(i)$	the set of AND predecessors of task i
$POR(i)$	the set of OR predecessors of task i
r_i	the revenue of parts released by task i
c_i	the cost of task i
$\tilde{t}_i$	uncertain processing time of task i
x_{ik}	=1, if task i is assigned to workstation k;
	=0, otherwise
z_k	=1, if workstation k is open;
	=0, otherwise

The problem alluded to in the previous section can be formulated into an integer program as below.

$$\max \sum\nolimits_{i=1}^{I} \sum\nolimits_{k=1}^{K} (r_i - c_i) x_{ik} - (F + C \cdot CT) \sum\nolimits_{k=1}^{K} z_k \tag{1a}$$

$$s.t. \quad \sum\nolimits_{k=1}^{K} x_{ik} \leq 1, \forall i \in I \tag{1b}$$

$$\sum\nolimits_{i=1}^{I} \tilde{t}_i x_{ik} \leq z_k \cdot CT, \forall k \in K \tag{1c}$$

$$x_{ik} \leq \sum\nolimits_{h=1}^{k} x_{lh}, \forall i \in I, k \in K, l \in PAND(i) \tag{1d}$$

$$x_{ik} \leq \sum\nolimits_{h=1}^{k} \sum\nolimits_{l \in POR(i)} x_{lh}, \forall i \in I, k \in K \tag{1e}$$

$$x_{ik}, z_k \in \{0, 1\}, \forall i \in I, k \in K \tag{1f}$$

The objective function (1a) maximizes the total profit of disassembling one product. The first term is the profit due to the disassembly of the product, while the second term represents the total cost of opening and operating workstations. Each task can be assigned to at most one workstation by constraint (1b). Constraint (1c) ensures that the total (uncertain) processing time of all the tasks assigned to a workstation must stay within the cycle time. The precedence constraint (1d) ensures that task i is assigned only when all its AND predecessors are already assigned. Similarly, constraint (1e) ensures that task i is assigned only if at least one of its OR predecessor has been assigned. Constraint (1f) indicates that decision variables are 0-1 variables.

The problem cannot be readily solved due to the random variables involved in constraint (1c). We now follow the B&S robust approach to re-formulate it as a non linear robust model. By constraint (1c) the total processing time allocated to any workstation k can not go beyond the cycle time. The most stringent condition occurs when all the tasks' processing times take their upper bound values. (Therefore d_i can be deemed as the protective buffer to prevent workstation overload against a particular task uncertainty.)

We first introduce a parameter Γ that determines the number of tasks whose processing times are considered uncertain and take their upper bound values in the robust model, while the rest of tasks are deemed as deterministic and take their lower bound values. Therefore, by varying the value of Γ between 0 and I we can adjust the robustness of the solutions against their conservativeness. When $\Gamma = 0$, the uncertainty of all processing times are ignored and the PDLBP-U reduces to the deterministic PDLBP with nominal times. When Γ increases, more tasks are considered as uncertain and the level of conservativeness of the solution increases. When $\Gamma = I$, all processing times take upper bound values and the solution is the most conservative. In this case the PDLBP-U can be converted into the deterministic PDLBP in the worst case. In order to derive a solution that is feasible for up to Γ uncertain tasks, a non-linear robust integer programming model is formulated as follows.

$$\max \sum\nolimits_{i=1}^{I} \sum\nolimits_{k=1}^{K} (r_i - c_i)x_{ik} - (F + C \cdot CT) \sum\nolimits_{k=1}^{K} z_k \tag{2a}$$

$$s.t. \quad \sum\nolimits_{i=1}^{I} t_i x_{ik} + h_k(x) \leq z_k \cdot CT, \forall k \in K \tag{2b}$$

$$h_k(x) = \max \left\{ \sum\nolimits_{i=1}^{I} d_i x_{ik} q_{ik} : \sum\nolimits_{i=1}^{I} q_{ik} \leq \Gamma \right\}, \forall k \in K \tag{2c}$$

$$0 \leq q_{ik} \leq 1, \forall i \in I, k \in K \tag{2d}$$

$$(1b), (1d), (1e), (1f)$$

where the constraint (2c) ensures that at most Γ operation times deviate from their nominal values, and q_{ik} is an auxiliary continuous variable measuring the deviation of the processing time of task i when allocated to workstation k.

The non-linear constraint (2c) for workstation k is equivalent to the following problem for given x.

$$h_k(x) = \max \sum\nolimits_{i=1}^{I} d_i x_{ik} q_{ik} \tag{3a}$$

$$s.t. \quad \sum\nolimits_{i=1}^{I} q_{ik} \leq \Gamma, \tag{3b}$$

$$0 \leq q_{ik} \leq 1, \forall i \in I, \tag{3c}$$

Following the B&S approach, the dual of problem (3) can be solved to linearize the constraint (2c) . By associating dual variables θ_k and w_{ik} to constraints (3b) and (3c) respectively, the following dual of problem (3) is obtained.

$$h_k(x) = \min \sum\nolimits_{i=1}^{I} w_{ik} + \Gamma \cdot \theta_k \tag{4a}$$

$$s.t. \quad \theta_k + w_{ik} \geq d_i x_{ik}, \forall i \in I, \tag{4b}$$

$$w_{ik} \geq 0, \forall i \in I, \tag{4c}$$

$$\theta_k \geq 0. \tag{4d}$$

By strong duality $h_k(x)$ is equal to the objective function value of Problem (4) , so the non-linear robust model (2) can be transformed into linear model (5) as follows(Bertsimas and Sim 2003).

$$\max \sum\nolimits_{i=1}^{I} \sum\nolimits_{k=1}^{K} (r_i - c_i)x_{ik} - (F + C \cdot CT) \sum\nolimits_{k=1}^{K} z_k \tag{5a}$$

$$s.t. \quad \sum\nolimits_{i=1}^{I} t_i x_{ik} + \sum\nolimits_{i=1}^{I} w_{ik} + \Gamma \cdot \theta_k \leq z_k \cdot CT, \forall k \in K \tag{5b}$$

$$\theta_k + w_{ik} \geq d_i x_{ik}, \forall i \in I, k \in K \tag{5c}$$

$$w_{ik} \geq 0, \forall i \in I, k \in K \tag{5d}$$

$$\theta_k \geq 0, \forall k \in K \tag{5e}$$

$$(1b), (1d), (1e), (1f)$$

In theory the linear model (5) can be solved exactly by branch-and-cut algorithms. However, for practical PDLBP-U problems, the exact optimal solutions cannot be obtained within reasonable time, as we know DLBP problems have been proven to be NP-complete (McGovern and Gupta 2007). To this end, we propose an MBO based meta-heuristic approach to solve the non-linear robust model (2) of PDLBP-U. As we shall see in the numerical experiments, the proposed approach can find good feasible solutions in reasonable time.

3. The improved migrating bird optimization

In this section we propose an improved MBO (IMBO) method with a tailored decoding strategy and effective neighbourhoods. The standard MBO algorithm is briefly introduced, followed by the detailed description of IMBO.

3.1. *A brief introduction to the MBO algorithm*

MBO is a recent meta-heuristic approach developed for combinatorial optimization problems. It is motivated by the V formation during the long distance journey in birds' migratory process, which can save about 70% of energy consumption. In this formation, birds fly at an angle and at a specific distance from each other. The leader bird uses the most energy, while the followers on both the left and right line save energy from the air agitation created by the companions in front of them. When the leader is tired, it goes to the end of the line and a bird following it takes its place as the next leader.

In the MBO algorithm, each bird in the flock represents a solution. A number of parallel solutions are evolved to enlarge the solution space during the solution process. Moreover, MBO distinguishes from the other swarm intelligence algorithms in its different individual evolutionary mechanism. Individuals (birds) in MBO not only can generate better neighboring solutions by themselves, but also may obtain better solutions from the neighborhood solutions of the previous individual. Therefore, the algorithm has faster convergence and more diversification that contribute to its efficiency.

MBO mainly contains four basic steps: initialization, leader improvement, follower improvement and new leader selection, which are introduced as below.

(1) ***initialization***: The MBO parameters are initialized, which include the number of solutions in the group (N), the number of neighboring solutions to be explored (n_e), the number of neighboring solutions to be shared with the following solution (n_s), the number of tours (m) and the termination limit (T). The initial population is randomly generated. The first solution is called the leader in the flock, and the others are regarded as the followers on the left and right line.
(2) ***leader improvement***: At first, the leader explores n_e number of its neighboring solutions. Then, the best neighboring solution is selected to compare against the leader. If the leader is improved, it is replaced by the best neighboring solution; otherwise, the leader stays the same and the unused best n_s neighboring solutions are kept as the neighboring solutions of the followers.
(3) ***follower improvement***: The current follower remains n_s neighboring solutions from the previous, and regenerates n_e-n_s solutions by searching its neighborhood. If the best neighboring solution improves the current follower, the current follower must be replaced. Subsequently, the unused n_s best neighboring solutions are moved to the next follower.
(4) ***new leader selection***: Leader improvement and followers improvement are one tour, and after m tours are finished, the leader will be updated. The leader moves to the end of the flock as a new follower, and the first follower becomes a new leader.
(5) ***termination criterion***: If the number of objective function evaluations exceeds T, the algorithm terminates; otherwise, go to Step 2 and repeat.

3.2. *The decoding process*

In this paper, a solution is encoded as a task assignment string. Each element of the string corresponds to one disassembly task of the product. Denote by $\{O_1, O_2, ..., O_M\}$ the task assignment string that meets the precedence relation of the tasks, where $O_i (i \in \{1, ..., M\})$ indicates the task in the ith position of the string. M is the total number of tasks required to completely disassemble the product. This encoding method is the same for both DLBP and

PDLBP.

The decoding process is to assign the tasks of the string to workstations while satisfying the cycle time constraint. Unlike the CT constraint of PDLBP, the total uncertain tasks' time in each workstation should satisfy the most stringent CT constraint in PDLBP-U. Assume the number of the tasks assigned to workstation k is n_k, we have $\sum_{i=1}^{I} x_{ik} = n_k$. For workstation k, if Γ is not more than n_k, then $\sum_{i=1}^{I} q_{ik} = \Gamma$, and $h_k(x) = \max \sum_{i=1}^{I} d_i x_{ik} q_{ik}$ is equivalent to the sum of the Γ largest d_i values assigned to workstation k. If Γ is more than n, we have $\sum_{i=1}^{I} q_{ik} = n_k < \Gamma$, and $h_k(x) = \sum_{i=1}^{I} d_i x_{ik}$. For example, if task 1,2 and 3 are assigned to the same workstation, the constraint (2b) takes the following forms for different Γ.

(i) when Γ=1, $t_1 + t_2 + t_3 + \max\{d_1, d_2, d_3\} \leq CT$,
(ii) when Γ=2, $t_1 + t_2 + t_3 + \max\{d_1 + d_2, d_1 + d_3, d_2 + d_3\} \leq CT$,
(iii) when $\Gamma \geq 3$, $t_1 + t_2 + t_3 + d_1 + d_2 + d_3 \leq CT$,

In the decoding process, rather than all M tasks must be undertaken in DLBP, only S tasks need to be done in PDLBP/PDLBP-U, where $S \in \{1, ..., M\}$ represents the disassembly length. For example, an AND/OR precedence diagram of a product with 12 tasks is shown in Figure 2. One task assignment string is $\{0, 2, 11, 1, 9, 8, 7, 3, 5, 4, 10, 6\}$. In DLBP, task 0 is performed first, followed by the others in turn, and finally task 6 is disassembled. In PDLBP /PDLBP-U, if S=5, only the first five elements of the string need to be performed, i.e., $\{0, 2, 11, 1, 9\}$.

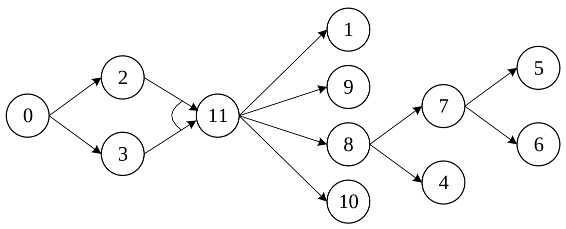


Figure 2. AND/OR precedence diagram.

In the PDLBP literature the value of S for each assignment string is usually randomly generated (Ren et al. 2017), so that the objective function value can be evaluated. The main advantage of this approach is fast, but it cannot guarantee the obtained partial sequence is optimal. In this study we propose to find the optimal S^*, which can be obtained by a slight modification of model (5). However, our numerical experiments suggest that this integer program is rather slow to solve. Therefore, we propose an exhaustive search algorithm to find S^* and the objective function value f_{best} of the solution, as shown in Algorithm 1.

Algorithm 1 The improved decoding process of solution

```
1: Given the value of $\Gamma$, open the first workstation($k$=1), set the best objective value $f_{best}$=0
2: for $S$=1 to $M$ do
3:     for $i$=1 to $S$ do
4:         if the CT constraint is not violated when the $i$th task is assigned to workstation $k$
           then
5:             assign task $i$ to workstation $k$;
6:             $i$=$i$+1;
7:         else
8:             $k$=$k$+1
9:             assign task $i$ to workstation $k$
10:            $i$=$i$+1
11:        end if
12:    end for
```

```
13:    calculate the objective value $f$
14:    **if** $f$ has an improvement than $f_{best}$ **then**
15:        $f_{best}$ is updated, $S$ is recorded $S^*$
16:    **end if**
17: **end for**
18: return $f_{best}$ and $S^*$
```

3.3. *Neighboring structures*

IMBO is a nature-inspired algorithm based on neighborhood search techniques. The new solutions are mainly determined by neighborhood structures, which make a solution move to its neighborhood one. In order to avoid invalid search and infeasible solutions, neighborhood structures must satisfy the precedence relation among tasks. Based on the basic insert and swap techniques in DLBP, four specific neighborhood structures with PDLBP-U features are defined as follow:

- ***Insert1***: a position i is randomly selected from the task assignment string, and a position interval $[h, j]$ within which task O_i can be inserted is found as follows. All tasks $O_l(h \leq l < i)$ are not AND predecessors of task O_i, if task O_l is OR predecessor of task O_i, then there must be task $O_k(k < h)$ is also OR predecessor of task O_l. In addition, task O_i is not the AND predecessor of tasks $O_l(i < l \leq j)$, if task O_i is OR predecessor of any task O_l, then there must be task $O_k(k < i)$ is OR predecessor of task O_l. Finally a position l is stochastically chosen in$[h, j]$, the task O_i is inserted into the position l.
- ***Insert2***: a position $i(r_i\text{-}c_i > 0)$ is randomly selected from an assignment string, and the position interval $[h, j]$ within which task O_i can be inserted is found in the same way as Insert1. Based on the principle that the task with high income is performed as early as possible, set $\Omega_1 = \{v|v \in [h, i)$ and $r_v\text{-}c_v < r_i\text{-}c_i\}$ and $\Omega_2 = \{v|v \in (i, j]$ and $r_v\text{-}c_v > r_i\text{-}c_i\}$ are constructed. Task O_i is inserted into the position v, which is selected randomly from set Ω_1 and Ω_2.
- ***Swap1***: a position i is randomly selected from the task assignment string, and the set $\Omega = \{j|j > i$, tasks O_j and O_i can be exchanged$\}$ is decided as follows. O_i is not AND predecessor of tasks $O_h(i < h \leq j)$, as long as task O_i is OR predecessor of any task O_h that task $O_k(k < i)$ is also OR predecessor of any task O_h. Meanwhile, all tasks $O_l(i \leq l < j)$ are not AND predecessors of task O_j, and if task O_l is OR predecessor of task O_j, then there must be task O_k is OR predecessor of task O_j. Finally a position j is stochastically chosen in set Ω, and the task O_i is exchanged with O_j.
- ***Swap2***: Starting from the first position($i = 1$), for any two adjacent position i and $j(j = i{+}1)$ from a task assignment string, if task O_i and task O_j met with the precedence relations, as long as the net income $r_j\text{-}c_j$ of task O_j is more than that of task O_i, then task O_i and task O_j are swapped.

In general, Insert1 and Swap1 can exploit new solutions more effectively, while Insert2 and Swap2 are beneficial to the exploration of the current solution. The detailed process to generate neighboring solutions is shown in Algorithm 2.

Algorithm 2 Generate n_e neighbor solutions for solution x

```
1: set $p$=1, $k$=1
2: **while** $k \leq n_e$ **do**
3: generate random number $\theta \in [0, 1]$
4: **if** $\theta$ <0.5
5:     **if** $p$=1, execute Insert1 for $x$, generate neighbor solution $x_k$
```

```
6:       if $x$ is improved by $x_k$, then $p = p$
7:       else $p$=$p$+1
8:       end if
9:    else Swap1 is done to $x$, neighbor solution $x_k$ is obtained
10:      if $x_k$ is better than $x$, then $p = p$
11:      else $p$=$p$-1
12:      end if
13:   end if
14: else
15:   if $p$=1, execute Insert2 for $x$, generate neighbor solution $x_k$
16:      if $x$ is improved by $x_k$, then $p = p$
17:      else $p$=$p$+1
18:      end if
19:   else Swap2 is done to $x$, neighbor solution $x_k$ is obtained
20:      if $x_k$ is better than $x$, then $p = p$
21:      else $p$=$p$-1
22:      end if
23:   end if
24: end if
25: $k = k$+1
26: end while
27: return $n_e$ neighbor solutions
```

3.4. *Leader update*

In general, a better solution can lead to better neighboring solutions, so an elite strategy is used to update the leader. In this paper, the global best solution is defined as the elite bird which may change by leader improvement and followers improvement of every tour. If the leader is not the elite bird after m tours, it is replaced by the elite bird. Otherwise the leader moves to the end and the solution behind the leader becomes a new leader.

3.5. *Follower reset*

In order to prevent premature convergence of the algorithm, a follower reset mechanism is designed. For each individual in the population, the age of solution is used to describe the evolutionary effect. For a newly generated follower the age of solution is set to one. After a tour is finished, the age is increased by one if the follower is not updated; otherwise the age is reset to one. When the age reaches the pre-set upper limit Age, it means that the follower has not been evolved yet. In such a situation the follower is regenerated. The follower reset mechanism can effectively expand the search space and increase the diversification of solutions.

3.6. *The IMBO algorithm*

The complete IMBO algorithm is presented in Algorithm 3.

Algorithm 3 The IMBO algorithm

```
1: Generate $N$ solutions, calculate the objective value of each solution via Algorithm 1, and
   select the best solution as the elite bird.
2: Initialize $T, m, n_e, n_s, Age, F_{best}$(the objective function of the elite bird); set $u$=0, the elite
   bird to be the leader
3: while $u \leq T$ do
4:   for $j$=1 to $m$ do
5:     Generate and evaluate $n_e$ neighboring solutions for the leader via Algorithm 2
6:     if the best neighboring solution improves the leader then
7:       the leader is replaced by the best neighboring solution
8:     end if
9:     move the unused best $n_s$ neighboring solutions to the follower solution as their
       neighbors
10:    if the leader improves the elite bird, then the elite bird is replaced by the leader, and
       $F_{best}$ is updated
11:    end if
12:    $u = u + n_e$
13:    for each follower solution do
14:      Generate $n_e - n_s$ neighbor solutions of the solution via Algorithm 2. Evaluate
         $n_e - n_s$ neighbors generated by the solution and $n_s$ neighbors are kept from the
         solution in the front
15:      if the best neighbor has an improvement then
16:        the current follower is updated, set $age = 1$
17:      else
18:        update the solution age, $age = age + 1$
19:        if $(age > Age)$, then regenerate the current follower
20:      end if
21:      if the current follower improves the elite bird, then the elite bird is replaced by
         the current follower, and $F_{best}$ is updated
22:      end if
23:      $u = u + n_e - n_s$
24:    end for
25:  end for
26:  if the leader is worse than the elite bird, then
27:    the leader is replace by the elite bird
28:  else move the leader to the end and the solution behind the leader becomes a new leader
29:  end if
30: end while
31: return the elite bird and $F_{best}$
```

4. Numerical studies

In this section, a case study with a real-life product is presented to validate the proposed model and illustrate key properties of PDLBP-U. The optimal solution of the case problem was obtained by solving the linear robust model (5) with the standard branch-and-cut algorithm. Further, to demonstrate its applicability and performance the proposed IMBO approach was tested on a number of instances and compared against the exact solutions, another heuristic, and multiple variants of the standard MBO. The non-linear robust models of these instances were solved by the heuristics while the corresponding linear robust models were solved by the branch-and-cut method in IBM-ILOG CPLEX 12.8. All experiments were run on a desktop with Intel(R) Core(TM) i7 CPU(1.80 GHz/8.00 G RAM).

4.1. *Case study*

In this case study a dishwasher with 44 disassembly tasks is considered. The precedence diagram is given in Figure 3 and the key parameters are adopted from Cevikcan et al. (2019), as shown in Table 2. We have let $p = 0.1$ and $d_i = \lceil 0.1 \cdot t_i \rceil$ for the purpose of this experiment. In Figure 3, task 1, 4, 6, 8 and 21 have no predecessor tasks. Tasks 1 and 4 are AND predecessors of a dummy task A_0. The dummy tasks (A_0, A_1 and A_2) are introduced to avoid crossing lines for clearer presentation. Additionally, F, C and CT are set to be 1, 0.05, and 60, respectively.

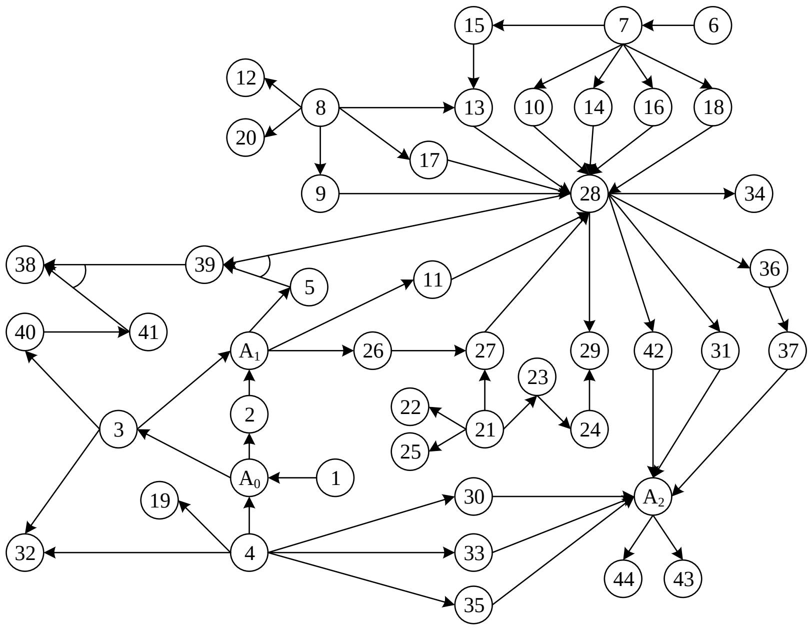


Figure 3. Precedence diagram of a dishwasher.

Table 2. The data of disassembly tasks for the dishwasher.

No	Disassembly task	t_i	d_i	r_i	c_i	No	Disassembly task	t_i	d_i	r_i	c_i
1	Detaching top cover and its insulation from main body	12	2	7.40	2.25	23	Disassembling control panel	9	1	1.31	2.54
2	Disassembly right panel and its insulation	19	2	3.28	0.34	24	Disassembling control unit	12	2	1.59	3.89
3	Disassembling left panel and its insulation	19	2	1.88	3.02	25	Detaching detergent and rinse aid dispenser	22	3	1.70	3.24

No	Disassembly task	t_i	d_i	r_i	c_i	No	Disassembly task	t_i	d_i	r_i	c_i
4	Removing front-lower access panel	23	3	1.75	5.43	26	Removing door spring and rope-pulley mechanism between the springs and hinge arms	22	3	9.12	1.86
5	Removing rear-lower panel	11	2	3.53	2.21	27	Detaching inner door panel from chassis	22	3	0.58	3.45
6	Removing silverware baskets	5	1	2.16	4.16	28	Disassembling tub	39	4	1.94	2.02
7	Removing bottom rack and bottom slide arm cover	9	1	4.15	3.36	29	Removing wire harnesses	33	4	2.37	3.43
8	Removing upper rack and upper slide arm cover	8	1	4.41	4.24	30	Disassembling blower	27	3	4.83	2.08
9	Disassembling upper rack s-lide	7	1	8.21	2.90	31	Removing heating element	29	3	5.24	1.77
10	Disassembling bottom rack slide	7	1	1.88	2.17	32	Disjointing drain hose	6	1	9.85	2.30
11	Disassembling slide support wheels	18	2	4.16	1.62	33	Detaching drain pump from sump	18	2	1.90	3.50
12	Detaching middle spray arm from upper rack	6	1	8.97	3.06	34	Detaching motor and circulation pump from heating element	16	2	5.10	1.60
13	Removing upper spray arm and water gateway	16	2	8.66	2.49	35	Disassembling turbidity sensor	14	2	7.48	1.23
14	Detaching lower spray arm	7	1	6.45	1.54	36	Detaching sump seal	6	1	0.80	4.52
15	Detaching filters	14	2	8.10	0.36	37	Removing sump	34	4	2.48	1.67
16	Removing salt box cover	6	1	7.83	1.38	38	Detaching intake line from inlet valve	7	1	8.29	2.19
17	Detaching regeneration reservoir cover from tub	7	1	1.61	3.38	39	Detaching inlet valve	10	2	3.21	3.33
18	Disassembling float valve	8	1	5.78	1.85	40	Detaching water flow meter	8	1	4.89	2.13
19	Disassembling float switch	14	2	2.11	2.57	41	Detaching regeneration reservoir (water pocket and air pocket)	12	2	5.77	1.56
20	Removing door gasket	13	2	2.00	3.43	42	Detaching salt box	14	2	4.78	4.76
21	Disassembling exterior door	30	3	7.04	1.75	43	Disassembling base plastic panel	11	2	4.18	4.39
22	Disjointing door latch	12	2	5.67	1.43	44	Removing levelling legs	12	2	5.45	2.45

For different Γ the optimal objective function values f_{opt} and the corresponding disassembly solutions are listed in Table 3. W_k represents the assigned task sequence for workstation k. When Γ=0, the processing time of each task takes their nominal values, and the optimal objective function value is 70.17. On the contrary, when Γ takes the maximum value of 44 the processing time of each task takes their upper bound values and the corresponding optimal objective value reduces to 67.60. Clearly, when Γ increases from 0 to 3, the optimal objective function value has dropped gradually from 70.17 to 67.60. After that the objective value does not change any more for larger Γ, and the results for the cases with $5 \leq \Gamma \leq 43$ are not included in Table 3. Moreover, the results show that once Γ is 3 or higher the profit remains the same, but the optimal disassembly sequences are different.

Table 3. The solutions for the real-life product.

Γ	The disassembly solution	f_{opt}
0	$W_1 = \{1\ 3\ 4\ 6\}$, $W_2 = \{2\ 7\ 16\ 32\ 40\ 41\}$, $W_3 = \{8\ 21\ 26\}$, $W_4 = \{9\ 12\ 15\ 22\ 35\ 38\}, W_5 = \{5\ 11\ 13\ 14\ 18\}$	70.17
1	$W_1 = \{1\ 4\ 6\ 7\ 18\}, W_2 = \{8\ 15\ 21\}, W_3 = \{3\ 13\ 40\ 41\}$, $W_4 = \{2\ 11\ 14\ 22\}, W_5 = \{16\ 26\ 32\ 35\ 38\}, W_6 = \{5\ 9\ 12\ 30\}$	68.92
2	$W_1 = \{1\ 3\ 4\}, W_2 = \{2\ 8\ 26\ 32\}, W_3 = \{6\ 7\ 12\ 16\ 18\ 40\ 41\}$, $W_4 = \{9\ 11\ 30\}, W_5 = \{5\ 15\ 21\}, W_6 = \{13\ 14\ 22\ 35\ 38\}$	68.92

Γ	The disassembly solution	f_{opt}
3	$W_1 = \{1\ 4\ 6\ 8\ 12\}, W_2 = \{2\ 3\ 9\ 40\}, W_3 = \{21\ 26\}$, $W_4 = \{7\ 11\ 18\ 22\ 32\}, W_5 = \{14\ 16\ 30\ 41\}, W_6 = \{13\ 15\ 35\ 38\}$	67.60
4	$W_1 = \{6\ 7\ 18\ 21\}, W_2 = \{4\ 8\ 9\ 15\}, W_3 = \{1\ 3\ 13\ 32\}$, $W_4 = \{2\ 35\ 40\ 41\}, W_5 = \{12\ 16\ 22\ 26\ 38\}, W_6 = \{11\ 14\ 30\}$	67.60
44	$W_1 = \{4\ 6\ 8\ 9\ 12\}, W_2 = \{1\ 3\ 35\ 40\}, W_3 = \{2\ 26\ 41\}$, $W_4 = \{7\ 13\ 15\ 16\ 18\}, W_5 = \{11\ 21\ 32\}, W_6 = \{14\ 22\ 30\ 38\ \}$	67.60

The results above show the cost of robust solutions, but not their benefit. To this end, we calculated the probability of constraint violation via Monte Carlo simulation. Specifically, we sampled $H = 100,000$ realisations of the task processing times. Given the optimal solution under a specific Γ, the number of violated constraints (1c) under each realisation was recorded. Denote by v the total number of constraint violations over all the H realisations. The probability of constraint violation can be approximated by $v/(zH)$, where $z = \sum_{k=1}^{K} z_k$ is the number of opening workstations in the optimal solution. As we did not require the distribution to be known, to facilitate this calculation, we considered four different distributions as follows.

- Uniform distribution $U[t_i, \bar{t}_i]$.
- Normal distribution $N(u_i, \sigma_i)$. We have let $u_i = (t_i + \bar{t}_i)/2$ and $\sigma_i = (\bar{t}_i - t_i)/6$ such that the interval $[t_i, \bar{t}_i]$ covers nearly all the values based on the three-sigma rule.
- Positive Skewed Triangular distribution in $[t_i, \bar{t}_i]$ with mode t_i.
- Negative Skewed Triangular distribution in $[t_i, \bar{t}_i]$ with mode $\bar{t}_i$.

The first two distributions are symmetric, while the other two are skewed to the left and right, respectively. For each distribution the corresponding violation probability was calculated and the results are reported in Table 4, where we also included the profit f_{opt}. The percentage in the parentheses represents the relative increase of profit from that of $\Gamma = 44$. Moreover, we tested multiple levels of uncertainty by varying the p value in $\{0.1, 0.3, 0.5\}$. Now that four different distributions were considered, we included into the comparison an alternative approach that replaces the uncertain processing times by their mean values. The resulting deterministic PDLBP problems were solved and the constraint violation probabilities were evaluated in the same way.

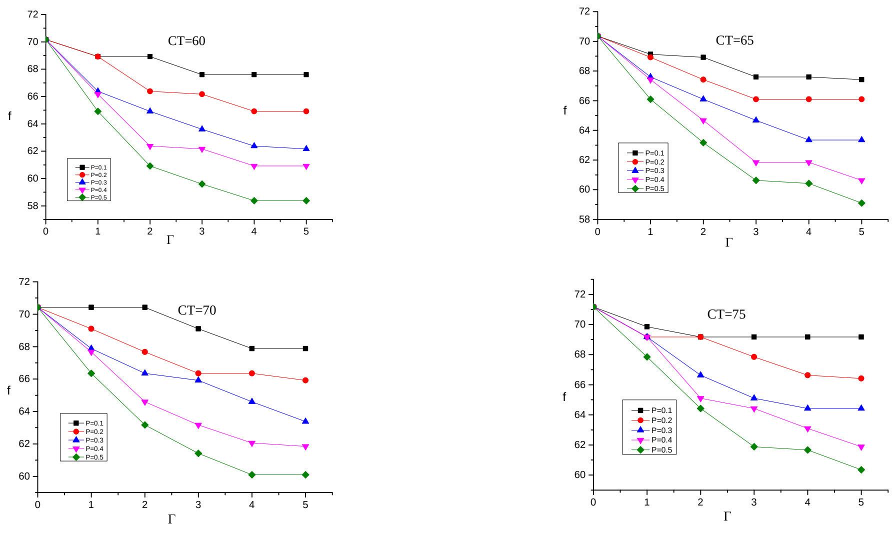
Table 4. The benefits and price of robustness.

		f_{opt}	Constraint violation probability(%)			
			Uniform	Normal	Triangular+(skewed)	Triangular-(skewed)
p=0.1	Γ=0	70.17(3.80%)	81.09	83.79	51.53	96.46
	Γ=1	68.92(1.95%)	0.00	0.01	0.00	0.01
	Γ=2	68.92(1.95%)	0.00	0.01	0.00	0.00
	Γ=3	67.60(0.00%)	0.00	0.00	0.00	0.00
	Γ=4	67.60(0.00%)	0.00	0.00	0.00	0.00
	Γ=44	67.60	0.00	0.00	0.00	0.00
	PDLBP (Uniform)	68.92(1.95%)	0.00	-	-	-
	PDLBP (Normal)	68.92(1.95%)	-	0.01	-	-
	PDLBP (Triangular+)	68.92(1.95%)	-	-	0.01	-
	PDLBP (Triangular-)	68.92(1.95%)	-	-	-	0.00
p=0.3	Γ=0	70.17(12.49%)	99.55	99.99	96.44	99.99
	Γ=1	67.60(8.37%)	40.47	39.69	9.42	70.5
	Γ=2	64.92(4.07%)	0.20	0.00	0.00	1.49
	Γ=3	63.60(1.96%)	0.00	0.00	0.00	0.00
	Γ=4	63.60(1.96%)	0.00	0.00	0.00	0.00
	Γ=44	62.38	0.00	0.00	0.00	0.00
	PDLBP (Uniform)	66.17(6.08%)	8.00	-	-	-
	PDLBP (Normal)	66.17(6.08%)	-	1.76	-	-
	PDLBP (Triangular+)	67.60(8.37%)	-	-	2.78	-

		f_{opt}	Constraint violation probability(%) uniform	normal	Triangular+(skewed)	Triangular-(skewed)
	PDLBP (Triangular-)	64.92(4.07%)	-	-	-	0.96
p=0.5	Γ=0	70.17(20.34%)	99.92	100.00	99.21	100.00
	Γ=1	66.17(13.48%)	55.44	56.76	20.79	79.97
	Γ=2	62.17(6.62%)	4.15	1.50	0.16	15.43
	Γ=3	59.63(2.26%)	0.39	0.01	0.00	4.05
	Γ=4	58.31(0.00%)	0.00	0.00	0.00	0.00
	Γ=44	58.31	0.00	0.00	0.00	0.00
	PDLBP (Uniform)	64.92(11.34%)	20.40	-	-	-
	PDLBP (Normal)	64.92(11.34%)	-	9.58	-	-
	PDLBP (Triangular+)	66.17(13.48%)	-	-	6.15	-
	PDLBP (Triangular-)	62.38(6.52%)	-	-	-	17.76

The results confirm that when Γ=44, the solution is feasible in all possible realizations, and thus the constraint violation probability is always zero regardless of the distributions. With the decrease of Γ, both the profit and the violation probability increase, showing clear trade-off between the benefits and price of robustness. For the same Γ, the probability is always the lowest for positive skewed Triangular distribution and the highest for the negative skewed Triangular distribution, which is not surprising given their definitions. The results also show that the larger the degree of uncertainty, the larger Γ should be selected to achieve the required violation probabilities. The performance of the PDLBP solutions using the mean task times is very interesting. They perform strongly when $p = 0.1$. In fact, the performance of all the solutions is similar apart from $\Gamma = 0$. In such situations one can use $\Gamma = 1$ or simply the deterministic PDLBP with the mean processing times, if available. In sharp contrast, when $p = 0.5$, the PDLBP solutions are rather poor. They may lead up to 20.40% constraint violations. The robust solutions could produce much lower constraint violations at the expense of some profit loss. For example, when $\Gamma = 3$ the violation probability is at most 4.05%. More importantly, the proposed approach provides multiple candidate solutions with a range of robustness and profit performance, which allows informed decision making based on specific needs.

CT is a key parameter in PDLBP-U, because it directly restrains the assignment of tasks and thus affects the profit. The other two key parameters that affect the solutions are Γ and the deviation coefficient p. Therefore, in order to analyze the sensitivity of the proposed approach, we generated five scenarios based on different values of CT (60,65,70,75,80) and p ($0.1 \sim 0.5$). The profits under different Γ values are displayed in the Figure 4.



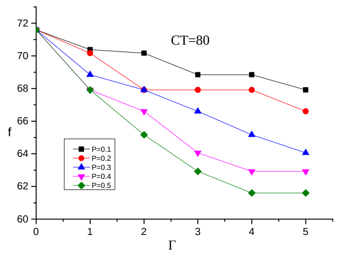


Figure 4. Sensitivity analysis under different CT, Γ and p.

Since in all scenarios the profit converges to the value in the worst case after $\Gamma = 5$, only the results of $\Gamma \leq 5$ are displayed in Figure 4. For each scenario the profit value is the same for different p as long as $\Gamma = 0$, because in this situation all the task times are at their nominal values. The profit then decreases as Γ increases, no matter how p changes. The downwards trend is much more obvious for larger p values, i.e., the profit value shows faster reduction. Therefore, the tighter the interval of the uncertain processing time, the smaller impact of uncertainty on the profit.

For the same value of p and Γ, the profit can increase with CT, because longer cycle time allows more flexibility for task allocation and requires less stations to be opened. However, the profit does not always increase with CT due to the uncertain task times and the dynamic number of tasks to be completed. In order to find the a reasonable CT value to allow more profit, the decision makers are suggested to experiment multiple values for the product structure and data concerned.

4.2. *Tuning of algorithm parameters*

There are five control parameters (N, m, n_e, n_s, Age) to be predetermined for IMBO. In this section a design-of-experiments (DOE) was implemented to find the best combination of these parameters. Since a full factorial method (Tian and Zhang 2019) for five parameters needs too many experiments, a representative combination, i.e. Taguchi method (Zhang et al. 2017) is more suitable. Since n_e and n_s are correlated ($n_e \geq 2n_s$) to guarantee enough neighbours of the leader to share with followers (Meng et al. 2018), a four factor and three level Orthogonal experiment array with 9 combinations is required, as shown in Table 5.

Table 5. Parameter values of IMBO.

Combination Number	N	m	n_e	n_s	Age
1	25	1	7	3	5
2	25	10	10	5	10
3	25	20	20	10	20
4	50	20	7	3	10
5	50	1	10	5	20
6	50	10	20	10	5
7	100	10	7	3	20
8	100	20	10	5	5
9	100	1	20	10	10

Each combination was tested on six cases (Γ = 0,1,2,3,4,44) of the real-life product in Table 2, and five parallel scenarios (CT=60,65,70,75,80) were considered for each case. Each test was solved 20 times independently, and 500,000 evaluations of the objective function was adopted as the stopping criterion. The difference between the average value obtained in 20 trials and the optimal solution from CPLEX was considered as the response factor. We report the average response values of five scenarios for the six cases in Table 6. It is shown that the performance of the fourth combination outperforms the others. Therefore the combination of N=50, m=20,

n_e=7, n_s=3, Age=10 were used in the following experiments.

Table 6. The Experimental Results.

Combination Number	Γ=0	Γ=1	Γ=2	Γ=3	Γ=4	Γ=44
1	0.76%	1.23%	2.96%	1.18%	0.68%	0.52%
2	1.72%	1.73%	2.33%	1.25%	0.46%	0.57%
3	0.87%	1.60%	2.56%	1.50%	0.59%	0.39%
4	0.22%	0.13%	0.63%	0.54%	0.03%	0.00%
5	0.72%	1.22%	1.48%	0.95%	0.35%	0.03%
6	0.47%	0.50%	0.73%	1.30%	1.74%	0.11%
7	0.48%	0.63%	1.32%	0.98%	0.15%	0.21%
8	0.28%	0.23%	0.68%	0.73%	1.60%	0.17%
9	0.00%	0.95%	1.10%	0.62%	0.13%	0.45%

4.3. *Performance comparison against the exact solution method*

In order to test the performance of IMBO, the solutions of the proposed algorithm in six problem instances were compared against the optimal/near optimal solutions from CPLEX. The first instance is an example with 12 tasks provided by Ren et al. (2017), and the precedence diagram is shown in Figure 2. The second and third one are the ball-point pen with 22 tasks and a network with 60 tasks, both from Kalaycılar et al. (2016). The other three larger-scale instances were generated according to the network rule introduced by Kalaycılar et al. (2016). These instances contain 120, 180 and 240 tasks, respectively, with CT set as 60, 80 and 100 accordingly. The cost, revenue and processing time of each task were generated from discrete uniform distributions DU[5,20], DU[10,50], and DU[1,20], respectively. For all instances F, C and p are set to be 10, 0.5, and 0.1.

The results are summarised in Table 7. The first column is the number of disassembly tasks in each instance and the second column the value of Γ. The next seven columns are the results from CPLEX and IMBO, with f_{opt} the best objective value obtained by CPLEX, f^* and f^a the best and the average solution of 20 executions of IMBO, β the percentage of solutions achieving the best, and $Gap = (f^* - f^a)/f^*$ the relative gap between the best and average IMBO solutions. The CPU(s) is the run time in second. CPLEX stops whenever an optimal solution is found or the time limit of 3600/7200s is reached, while IMBO is terminated after 500,000 evaluations of the objective function. The last column $\Delta(f^*, f_{opt})$ is the sub-optimality of f^* from the solution f_{opt}, as written below.

$$\Delta(f^*, f_{opt}) = (f^* - f_{opt})/f^* \tag{6}$$

Table 7. Comparison of the performance of the exact method and IMBO.

N	Γ	CPLEX		IMBO					$\Delta(f^*, f_{opt})(\%)$
		f_{opt}	CPU(s)	f^*	f^a	$\beta(\%)$	$Gap(\%)$	CPU(s)	
12	0	55∗	1.25	55	55	100	0.00	0.45	0.00
	1	55∗	3.70	55	55	100	0.00	0.53	0.00
	2	55∗	5.22	55	55	100	0.00	0.60	0.00
	3	55∗	5.27	55	55	100	0.00	0.64	0.00
	4	55∗	6.73	55	55	100	0.00	0.72	0.00
	12	55∗	4.03	55	55	100	0.00	0.48	0.00
22	0	157∗	51.08	157	156.60	90	0.25	1.19	0.00
	1	153∗	122.36	153	149.45	65	2.32	1.70	0.00
	2	141∗	2899.32	141	139.85	75	0.82	2.00	0.00
	3	136∗	4783.38	136	135.2	75	0.59	2.66	0.00
	4	136∗	3286.42	136	135.1	70	0.66	2.90	0.00
	22	136∗	359.69	136	135.55	90	0.33	1.56	0.00

N	Γ	CPLEX		IMBO					$\Delta(f^*, f_{opt})(\%)$
		f_{opt}	CPU(s)	f^*	f^a	$\beta(\%)$	$Gap(\%)$	CPU(s)	
60	0	568	3600	550	539.84	40	1.85	10.74	-3.27
	1	520	3600	530	519.17	30	2.04	12.11	1.89
	2	514	3600	520	515.42	25	0.88	15.54	1.15
	3	491	3600	493	487.10	35	1.20	18.43	0.41
	4	479	3600	491	480.63	30	2.11	20.82	2.44
	60	514	3600	486	474.20	50	2.43	10.83	-5.76
	0	568	7200	550	539.84	40	1.85	10.74	-3.27
	1	535	7200	530	519.17	30	2.04	12.11	-0.94
	2	530	7200	520	515.42	25	0.88	15.54	-1.92
	3	524	7200	493	487.10	35	1.20	18.43	-6.29
	4	514	7200	491	480.63	30	2.11	20.82	-4.68
	60	514	7200	486	474.20	50	2.43	10.83	-5.76
120	0	1344	7200	1305	1251.10	15	4.13	32.81	-2.99
	1	1224	7200	1251	1219.75	10	2.50	38.9	2.16
	2	1205	7200	1225	1190.85	5	2.79	47.67	1.63
	3	1158	7200	1205	1154.95	5	4.15	62.19	3.90
	4	1132	7200	1190	1150.60	5	3.31	74.08	4.87
	120	1238	7200	1170	1117.40	15	4.50	33.23	-5.81
180	0	1789	7200	1681	1609.10	10	4.28	74.76	-6.42
	1	1561	7200	1629	1583.7	5	2.78	77.15	4.17
	2	1489	7200	1605	1551.20	5	3.35	83.62	7.23
	3	1396	7200	1550	1486.00	5	4.13	104.93	9.94
	4	1302	7200	1542	1478.25	5	4.13	123.71	15.56
	180	1651	7200	1493	1428.70	5	4.31	75.35	-10.58
240	0	2523	7200	2467	2354.86	5	4.55	103.72	-2.27
	1	-	7200	2439	2299.10	5	5.74	121.32	-
	2	-	7200	2386	2188.70	5	8.27	154.99	-
	3	-	7200	2366	2248.66	5	4.96	172.59	-
	4	-	7200	2346	2230.61	5	4.92	208.06	-
	240	2334	7200	2261	2108.64	5	6.74	105.68	-3.23

The numbers with $*$ represent the exact optimal solutions.

For the first two instances, CPLEX obtained the optimal solutions under all Γ values, whereas IMBO found the same optimal solutions in most cases and with less time, as indicated by the large β values. For the second instance, it is obvious that the two extreme cases (Γ=0 and $\Gamma = I$) can be quickly solved by CPLEX, because the problems are equivalent to the deterministic PDLBP. Longer computational times are observed for the other cases ($0 < \Gamma < I$). The similar pattern of computing time is also observed in assembly line balancing problem (Hazır and Dolgui 2013). For the other four and larger instances, the optimal solutions cannot be found within the 2h time limit by CPLEX, even though the solutions of the two extreme cases are still better than those of the other cases. Therefore, for these two cases the solution obtained by IMBO is slightly worse than CPLEX, and thus the negative gaps. The inferiority increases with the problem size and the biggest is -10.58%. Despite this result the running time of IMBO is no more than 110s in both extreme cases and across all scenarios. For the other cases where Γ varying from 1 to 4, when N=60, the solutions obtained by IMBO outperform CPLEX for the time limit of 3600s, but they become worse than CPLEX when the time limit is increased to 7200s. As the problem size further increases IMBO's performance becomes stronger than CPLEX. For N=120 and 180, IMBO obtained better solutions than CPLEX in shorter CPU time. When N = 240, IMBO obtained feasible solutions in about 200s, whereas CPLEX was unable to find any solution within 2h. The results also indicate that the number of IMBO solutions achieving the best decreases with N, and β reduces from 100% sharply to 5%. This is not surprising given the exponential increase of the solution space. Nevertheless, the IMBO solution gap is always within 8.27% in all the experiments.

4.4. *Performance comparison against other heuristic algorithms*

The robust models of PDLBP-U have not been considered in the literature and the existing heuristics are not readily applicable to the problem. One potential method is the improved gravitational search algorithm (IGSA)(Ren et al. 2017) that has been developed to solve PDLBP. We extend IGSA as follows: add the objective function evaluation based on different Γ as the same as IMBO, and choose IGSA as one comparable algorithm. To understand the impact on the performance of IMBO due to the improved decoding strategy and effective neighborhoods, three variants of IMBO are chosen as the other comparable algorithms. The first variant N-MBO is derived from IMBO by disabling the improved decoding strategy; the second one H-MBO is obtained by replacing effective neighborhoods of IMBO with the swap and insert operations suggested in Ren et al. (2017); the third one is the standard MBO approach without either of these two enhancement techniques.

The number of agents for IGSA is set at 100, which is the best value suggested by Ren et al. (2017). Except the termination condition all the others parameters of IGSA take the same values as given in Ren et al. (2017). The parameter settings of all the IMBO variants are the same. All these five algorithms use the same termination criteria(CPU computing time) as shown in Table 7. Each algorithm was applied to the 6 problem instances 20 times and the best solutions were used to calculate the relative percentage deviations (RPD): $RPD = (f^* - f_i)/f^*$, where $f_i(i{=}1,2,3,4)$ represents the best solutions of H-MBO, N-MBO, MBO and ISGA, respectively. The results are presented in Table 8.

Table 8. RPD(%) values of alternative algorithms compared to IMBO.

N	Γ	H-MBO	N-MBO	MBO	IGSA
12	0	0.00	0.00	0.00	0.00
	1	0.00	0.00	0.00	0.00
	2	0.00	0.00	0.00	0.00
	3	0.00	0.00	0.00	0.00
	4	0.00	0.00	0.00	0.00
	12	0.00	0.00	0.00	0.00
22	0	0.00	2.55	2.55	5.73
	1	0.00	0.00	0.00	5.88
	2	3.55	3.55	3.55	8.51
	3	0.00	0.00	0.00	6.62
	4	0.00	0.00	0.00	9.56
	22	0.00	0.00	0.00	12.50
60	0	7.09	7.64	7.64	10.73
	1	8.49	8.67	9.62	13.40
	2	10.38	10.38	12.12	17.31
	3	6.90	9.13	10.75	13.39
	4	8.76	11.00	12.02	13.85
	60	9.05	11.93	12.76	20.58
120	0	3.37	3.91	4.98	7.05
	1	2.00	3.04	5.52	6.31
	2	3.10	3.67	5.06	6.94
	3	4.48	3.65	6.47	8.88
	4	4.96	3.53	5.29	9.33
	120	3.68	4.70	6.32	9.74
180	0	5.35	6.25	8.21	8.15
	1	4.73	2.21	8.35	7.61
	2	5.92	3.36	8.54	9.03
	3	6.32	3.87	8.52	6.77
	4	5.84	6.03	11.22	10.89
	180	6.43	5.89	9.11	11.25

N	Γ	H-MBO	N-MBO	MBO	IGSA
240	0	3.00	2.80	8.15	4.78
	1	3.81	3.32	9.18	5.99
	2	2.77	1.63	7.92	4.82
	3	2.92	3.00	8.37	5.37
	4	3.92	4.05	9.29	6.44
	240	4.07	4.60	8.27	6.99

It can be concluded from Table 8 that, for the smallest instance, all heuristics can obtain the optimal solutions. In all the other instances, IMBO obtains the best solutions, and comfortably outperforms the other four. In addition, when N=60 and 120, H-MBO and N-MBO are the second and third best, while IGSA is notably the worst. When N=180 and 240, N-MBO generates better solutions than H-MBO, and MBO is beaten by IGSA.

These results clearly demonstrate the effectiveness of the proposed enhancement techniques. Both the improved decoding strategy and the effective neighborhoods contribute to the much improved performance over the standard MBO. In particular, the effectiveness of the improved decoding strategy is more distinct for smaller problems, while that of the effective neighborhoods more distinct for larger problems. In all cases IMBO delivers much better performance than MBO, which is also beaten by IGSA in multiple scenarios, especially when the problem is large.

5. Conclusions

In this paper, a partial disassembly line balancing problem under uncertainty is studied, which aims to allocate a partial sequence of tasks to a number of workstations such that the overall profit is maximized. The processing time of the tasks is assumed to be uncertain and takes values from known intervals. To address the uncertainty, we follow the B&S robust optimization approach and formulate the problem as a non-linear robust model, which is then converted into an equivalent linear robust model. Even though in theory the resulting model can be solved by exact algorithms, the computational complexity means that the optimal solutions cannot be obtained within reasonable time for practical problems. To this end we have developed an improved migrating birds optimization algorithm with two enhancements. The first finds the optimal number of tasks to be performed for each sequence rather than random selection that is used in the literature. The second exploits the specific problem structure to construct effective neighborhoods. The performance of the proposed IMBO approach is firstly compared against the exact method across multiple test instances. The results show that IMBO can find the optimal solutions for small-scale instances, and can find better solutions in shorter time for large-scale instances. The proposed approach is then compared against another heuristic method IGSA; even though both find the optimal solutions for the smallest instances, the former is able to find better solutions than the latter in all the other instances considered. The results also clearly demonstrate the effectiveness of the proposed enhancement techniques. In particular, the effectiveness of the improved decoding strategy is more distinct for smaller problems, while that of the effective neighborhoods more distinct for larger problems. In all cases IMBO delivers much better performance than the basic MBO approach.

The future research may consider more practical problems such as the mix model disassembly line balancing problem under uncertainty, or the profit-oriented U-shape disassembly line balancing problem under uncertainty. The innovative solution approaches for these problems are also worth being investigated.

Funding

This research is supported by the National Natural Science Foundation of China [grant number 71471151].

References

Altekin F.T., L. Kandiller, and N.E. Özdemirel. 2004. "Disassembly Line Balancing with Limited Supply and Subassembly Availability." *Proceedings of SPIE - The International Society for Optical Engineering* 5262: 59-70.

Altekin F.T., L. Kandiller, and N.E. Özdemirel. 2008. "Profit-oriented disassembly-line balancing." *International Journal of Production Research* 46(10): 2675-2693.

Altekin F. T., and C. Akkan. 2012. "Task-Failure-Driven Rebalancing of Disassembly Lines." *International Journal of Production Research* 50 (18): 4955-4976.

Adulyasak Y., and P. Jaillet. 2015. "Models and Algorithms for Stochastic and Robust Vehicle Routing with Deadlines." *Transportation Science* 50 (2): 608-626.

Bertsimas D., and M. Sim. 2003. "Robust discrete optimization and network flows." *Mathematical Programming* 98: 49-71.

Bertsimas D., and M. Sim. 2004. "The price of robustnes." *Operations Research* 52 (1): 35-53.

Bertsimas D., and I. Popescu. 2005. "Optimal Inequalities in Probability Theory: A Convex Optimization Approach." *SIAM Journal on Optimization* 15 (3): 780-804.

Boysen N., M. Fliedner, and A. Scholl. 2007. "A Classification of Assembly Line Balancing Problems." *European Journal of Operational Research* 183 (2): 674-693.

Ben-Tal A., L.E. Ghaoui and A. Nemirovski. 2009. "Robust Optimization." *Princeton, NJ: Princeton University Press.*

Bertsimas D., D.B. Brown, and C. Caramanis. 2011. "Theory and applications of robust optimization." *SIAM REVIEW* 53(3):464-501.

Bentaha M.L., O. Battaïa, and A. Dolgui. 2013. "Chance Constrained Programming Model for Stochastic Profit-Oriented Disassembly Line Balancing in the Presence of Hazardous Parts." *Advances in Production Management Systems. Sustainable Production and Service Supply Chains*: 103-110.

Bentaha M.L., O. Battaïa, A. Dolgui and S.J. Hu. 2014. "Dealing with Uncertainty in Disassembly Line Design." *CIRP Annals-Manufacturing Technology* 63: 21-24.

Bentaha M.L., O. Battaïa, A. Dolgui, and S.J. Hu. 2015. "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, 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: 7220-7242.

Cevikcan E., D. Aslan and F.B. Yeni. 2019. "Disassembly line design with multi-manned workstations: a novel heuristic optimisation approach." *International Journal of Production Research*: 1-22.

Delage E., Y.Y. Ye. 2010." Distributionally robust optimization under moment uncertainty with application to data-driven problems". *Operations Research* 58(3): 595-612.

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." *The International Journal of Advanced Manufacturing Technology* 48(5-8): 761-771.

Duman E., M. Uysal, and A.F. Alkaya. 2012. "Migrating birds optimization: A new metaheuristic approach and its performance on quadratic assignment problem." *Information Sciences* 217(24): 65-77.

Dindar Oz. 2017. "An improvement on the Migrating Birds Optimization with a problem-specific neighboring function for the multi-objective task allocation problem." *Expert Systems With Applications* 67: 304-311.

El Ghaoui L., F. Oustry, and H. Lebret. 1998. “Robust solutions to uncertain semidefinite programs.” *SIAM Journal on Optimization* 9(1): 33-52.

Gupta S.M., and K.N. Taleb. 1994. “Scheduling disassembly.” *International Journal of Production Research* 32:1857-1866.

Güngör A., and S.M. Gupta. 1999. “Disassembly Line Balancing. ” *In Proceedings of the Annual Meeting of the Northeast Decision Sciences Institute*: 193-195.

Güngör A., and S.M. Gupta. 2002. “Disassembly line in product recovery.” *International Journal of Production Research* 40(11): 2569-2589.

Gorissen B.L., I. Yanıkoğlu and D.D. Hertog. 2015. “A practical guide to robust optimization” *Omega* 53: 124-137.

Hazır Ö., and A. Dolgui. 2013. “Assembly line balancing under uncertainty: Robust optimization models and exact solution method.” *Computers & Industrial Engineering* 65(2): 261-267.

Kalayci C.B., and S.M. Gupta. 2013. “Ant colony optimization for sequence-dependent disassembly line balancing problem.” *Journal of Manufacturing Technology Management* 24(3): 413-427.

Kalayci C.B., and S.M. Gupta. 2013. “A particle swarm optimization algorithm with neighborhood-based mutation for sequence-dependent disassembly line balancing problem.” *The International Journal of Advanced Manufacturing Technology* 69(1-4): 197-209.

Kalayci C.B., and S.M. Gupta. 2013. “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. 2014. “A tabu search algorithm for balancing a sequence-dependent disassembly line.” *Production Planning & Control* 25(2):149-160.

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: 321-354.

Kalaycılar E.G., M. Azizoğlu, and S. Yeralan. 2016. “A Disassembly Line Balancing Problem with Fixed Number of Workstations.” *European Journal of Operational Research* 249(2): 592-604.

Lambert, A.J.D., and S.M. Gupta. 2005. “Disassembly Modelling for Assembly, Maintenance, Reuse and Recycling.” *Boca Raton, FL: CRC Press.*

McGovern S.M., and S.M. Gupta. 2007. “A balancing method and genetic algorithm for disassembly line balancing.” *European Journal of Operational Research* 179(3): 692-708.

Meng T., Q.K. Pan, J.Q. Li, and H.Y. Sang. 2018. “An improved migrating birds optimization for an integrated lot-streaming flow shop scheduling problem.” *Swarm and Evolutionary Computation* 38: 64-78.

Mulvey, J.M., R.J. Vanderbei, and S.A. Zenios. 1995. “Robust optimization of large-scale systems.” *Operations Research* 43(2): 264-281.

Niroomand S., A. Hadi-Vencheh, R. Sahin, and B. Vizvári. 2015. “Modified migrating birds optimization algorithm for closed loop layout with exact distances in flexible manufacturing systems.” *Expert Systems with Applications* 42(19): 6586-6597.

özceylan E., C.B. Kalayci, A. Güngör and M.G. Surendra. 2019. “Disassembly line balancing problem: a review of the state of the art and future directions.” *International Journal of Production Research* 57: 15-16.

Pereira J., and E. Álvarez-Miranda. 2018. “An exact approach for the robust assembly line balancing problem.” *Omega* 78: 85-98.

Ren Y.P., D.Y. Yu, C.Y. Zhang, G.D. Tian, L. L. Meng, and X.Q. Zhou. 2017. “An improved gravitational search algorithm for profit-oriented partial disassembly line balancing problem.” *International Journal of Production Research* 55(24): 7302-7316.

Soyster A.L.. 1973. “Convex programming with set-inclusive constraints and applications to inexact linear programming.” *Operations Research* 21(5): 1154-1157.

Sioud A., and C. Gagné. 2018. “Enhanced migrating birds optimization algorithm for the permutation flow shop problem with sequence dependent setup times.” *European Journal* *of Operational Research* 264(1): 66-73.

Tian X.Y., Z.H. Zhang. 2019. "Capacitated disassembly scheduling and pricing of returned products with price-dependent yield." *Omega* 84: 160-174.

Zhang B., Q.K. Pan, L. Gao, X.L. Zhang, H.Y. Sang, and J.Q. Li. 2017. "An effective modified migrating birds optimization for hybrid flow shop scheduling problem with lot streaming." *Applied Soft Computing* 52: 14-27.

Zhang Y.L., S.Q. Shen, S.A. Erdogan. 2018. "Solving 0-1 semidefinite programs for distributionally robust allocation of surgery blocks." *Optimization Letters* 12: 1503-1521.