L-Shaped Algorithm for Stochastic Disassembly Line Balancing Problem
Mohand Lounes Bentaha, Olga Battaïa, Alexandre Dolgui

To cite this version:
Mohand Lounes Bentaha, Olga Battaïa, Alexandre Dolgui. L-Shaped Algorithm for Stochastic Disassembly Line Balancing Problem. 7th IFAC Conference on Manufacturing Modelling, Management, and Control, 2013, Jun 2013, St Petersburg, Russia. pp. 407-411, 10.3182/20130619-3-RU-3018.00500 . emse-00881583

HAL Id: emse-00881583
https://hal-emse.ccsd.cnrs.fr/emse-00881583
Submitted on 22 Mar 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
L-shaped Algorithm for Stochastic Disassembly Line Balancing Problem

M. L. Bentaha * O. Battaïa * A. Dolgui *

* École Nationale Supérieure des Mines, EMSE-FAYOL, CNRS UMR6158, LIMOS, F-42023 Saint-Étienne, France (e-mail: {bentaha,battaia,dolgui}@emse.fr).

Abstract: Disassembly Line Balancing Problem considered here both chooses the best disassembly alternative for an end of life product and assigns the corresponding disassembly tasks to the workstations of the line with the aim to reduce the line cost. Precedence and cycle time constraints are observed. Task times are assumed stochastic with known normal probability distributions. The line cost includes the investment and operation costs for workstations as well as penalty costs generated by the cycle time constraints violations. To deal with uncertainties, a stochastic linear mixed integer formulation is developed. To solve efficiently the problem, L-shaped algorithm combined with Latin Hypercube Sampling is proposed.

Keywords: Disassembly, Disassembly Line Design, DLBP, Stochastic Programming, Sample Average Approximation, L-shaped Method, LHS, Sustainable manufacturing.

1. INTRODUCTION

Disassembly plays an important role in the efficient management of the waste electrical and electronic equipment as well as end-of-life vehicles. The industrial interest to this process is increasing not only due to the environmental awareness of consumers and governments, but also due to its economical attractiveness (Zorbas and Inglezakis (2012), Mayyas et al. (2012), Güngör and Gupta (1999)). Disassembly aims at the separation of an End of Life (EOL) product into its components or sub-assemblies for product and material recovery. Product recovery includes the parts reuse and remanufacturing, while material recovery consists of recycling and material reuse. The disassembly line is the most efficient solution for disassembling complex products (like vehicles) and small products available in large quantities (like cell phones). Installing a disassembly line is the most efficient solution for disassembly tasks can be performed, but only one must be chosen, this subassembly is related to these disassembly tasks by Or–type arcs. To calculate the number of these subassemblies by And–type arcs (in bold on the Figure 1). If, for a given subassembly, one or more disassembly tasks can be performed, but only one must be chosen, this subassembly is related to these disassembly tasks by Or–type arcs. In order to calculate the number of workstations (after optimization), a dummy task 5 is introduced into the precedence graph as a sink node, see Figure 1. Disassembly task times $t_i$, $i \in I$, are assumed stochastic with known normal probability distributions: mean $\mu$ and variance $\sigma^2$ are known for each task time, i.e. $t_i \sim N(\mu_i, \sigma_i)$, $t_i > 0$, $i \in I$; (Dolgui and Proth (2010), Silverman and Carter (1986)). It is modeled by a random vector $\xi = (t_1, t_2, \ldots, t_N)$ varying over a set $\Xi \subset \mathbb{R}^N$; consider a given probability space $(\Xi, \mathcal{F}, P)$ introduced by the random vector $\xi$. Let $t_i = \xi_i(\xi)$, $i \in I$. The disassembly tasks have to be assigned to the workstations of the line in such a way that precedence constraints given by And/Or graph are respected and at the same time the line cost is minimized. The line cost includes two components: first one is related to the cost of workstations used, the second types of arcs define the precedence relations between the subassemblies and the disassembly tasks: And–type and Or–type arcs. For example, if a disassembly task generates two subassemblies, or more, then it is related to these subassemblies by And–type arcs (in bold on the Figure 1). If, for a given subassembly, one or more disassembly tasks can be performed, but only one must be chosen, this subassembly is related to these disassembly tasks by Or–type arcs. In order to calculate the number of workstations (after optimization), a dummy task 5 is introduced into the precedence graph as a sink node, see Figure 1. Disassembly task times $t_i$, $i \in I$, are assumed stochastic with known normal probability distributions: mean $\mu$ and variance $\sigma^2$ are known for each task time, i.e.

$$t_i \sim N(\mu_i, \sigma_i), \ t_i > 0, \ i \in I;$$

(Dolgui and Proth (2010), Silverman and Carter (1986)). It is modeled by a random vector $\xi = (t_1, t_2, \ldots, t_N)$ varying over a set $\Xi \subset \mathbb{R}^N$; consider a given probability space $(\Xi, \mathcal{F}, P)$ introduced by the random vector $\xi$. Let $t_i = \xi_i(\xi), \ i \in I$. The disassembly tasks have to be assigned to the workstations of the line in such a way that precedence constraints given by And/Or graph are respected and at the same time the line cost is minimized. The line cost includes two components: first one is related to the cost of workstations used, the second

2. PROBLEM FORMULATION

The case of complete disassembly of a single type of EOL product is considered. All possible alternatives for disassembly process are given by an And/Or Graph (AOG), (Ma et al. (2011), Altekin et al. (2008), Koc et al. (2009)). An example for such a graph is given in Figure 1. To simplify the graph, without information loss, subassemblies with one component are not represented. Each subassembly of the product to be disassembled is represented by an auxiliary node $A_k, k \in K$, in the AOG, and each basic node $B_i, i \in I$, represents a disassembly task. Two

Fig. 1. And/Or precedence graph
one is entailed by exceeding cycle time $C$. In fact, each time the duration of tasks assigned to one workstation exceeds the given line cycle time, a corrective action is needed, and its cost is included in the objective function. For the problem described, the following mathematical model has been developed.

**Stochastic Linear Mixed Integer Program**

**Parameters**

$I$: Disassembly tasks’ index set: $I = \{1, 2, \ldots, N\}$, $N \in \mathbb{N}^*$;

$J$: Workstations’ index set: $J = \{1, 2, \ldots, M\}$, $M \in \mathbb{N}^*$;

$K$: Index set of the And/Or precedence graph’s auxiliary nodes: $K = \{0, 1, \ldots, K\}$, $K \in \mathbb{N}$;

$A_k$: Auxiliary node of the And/Or graph, $k \in K$;

$B_i$: Disassembly task, $i \in I$;

$S$: The And/Or graph’s sink node, $S = 0$;

$C$: Cycle time, $C > 0$;

$P_k$: Predecessors index set of $A_k$, $k \in K$, i.e. $P_k = \{i | B_i$ precedes $A_k\}$;

$S_k$: Successors index set of $A_k$, $k \in K$, $S_k = \{i | A_k$ precedes $B_i\}$.

**Decision Variables**

$x_{ij} = \begin{cases} 1, & \text{if disassembly task } B_i \text{ is assigned to workstation } j; \\ 0, & \text{otherwise.} \end{cases}$

$x_{sj} = \begin{cases} 1, & \text{if dummy task } S \text{ is assigned to workstation } j; \\ 0, & \text{otherwise.} \end{cases}$

$z_j = \begin{cases} C, & \text{if } x_{sj} = 1; \\ 0, & \text{otherwise.} \end{cases}$

The objective considered in this paper is to minimize the line cost including fixed workstation operating costs and recourse costs, $q_j, j \in J$ (per unit time), caused by exceeding $C$. A recourse variable $y_j(\xi), j \in J$, measures the amount of time exceeding $C$ if there is any. The following model is used for the Stochastic DLBP presented.

**SLMIP**

$$\min \left\{ F_c \cdot \sum_{j \in J} j \cdot z_j + \mathbb{E}_{\xi} \left( \sum_{j \in J} q_j \cdot y_j(\xi) \right) \right\}$$

s.t.

$$\begin{align*}
  z_j &= C \cdot x_{sj}, \forall j \in J \\
  \sum_{i \in S j} x_{ij} &= 1 \\
  \sum_{j \in J} x_{ij} &\leq 1, \forall i \in I \\
  \sum_{i \in S_k} \sum_{j \in J} x_{ij} &= \sum_{i \in P_k} \sum_{j \in J} x_{ij}, \forall k \in K \setminus \{0\} \\
  \sum_{i \in S_k} \sum_{j \in J} x_{ij} &\leq \sum_{i \in P_k} \sum_{j = 1} x_{ij}, \forall k \in K \setminus \{0\}, \forall v \in J
\end{align*}$$

The term of the objective function represents the cost of operating workstations and the expected recourse cost. If the dummy task $S$ is assigned to station $j$, which defines the number of processed workstations, then $\sum_{j \in J} j \cdot z_j = j \cdot C$ and the workstations operating cost becomes $j \cdot (F_c \cdot C)$. Constraints (1) ensure the value of $z_j$ to be $C$ when dummy task $S$ is assigned to station $j$. Constraint (2) imposes the selection of only one disassembly task (Or–successor) to begin the disassembly process. Constraint set (3) indicates that a task is to be assigned to at most one workstation. Constraints (4) ensure that only one Or–successor is selected. Constraint set (5) defines the precedence relations among tasks: the selected successor is assigned to upper–indexed station (or the same) than the one to which the selected predecessor is assigned. Constraint (6) imposes the assignment of the dummy task $S$ to one station. Constraints (7) ensure that all the disassembly tasks, preceding $S$, are assigned to lower or equal–indexed workstations than the one to which $S$ is assigned. The constraints (8) enforces the station operating time to remain within the cycle time under uncertainty. Finally, sets (9)–(11) represent the trivial constraints. $\mathbb{E}_{\xi}$ stands for the expected value with the respect to the distribution of $\xi$:

$$\mathbb{E}_{\xi} \left( \sum_{j \in J} q_j \cdot y_j(\xi) \right) = \int_{\xi} \left( \sum_{j \in J} q_j \cdot y_j(\xi) \right) dP$$

Note that the integral (12) makes the program (I) a nonlinear one, that is one source of difficulty in solving this problem.

Let $x$ be a vector representing a solution of problem (I) and $X = \{x |$ constraints (1)–(7), (9)–(10) are satisfied$\}$. The program (II), given below, represents an equivalent stochastic version of program (I). It is a two–stage stochastic linear mixed integer program with deterministic recourse, (Birge and Tang (1993), Kall and Wallace (1994)).

$$\min \left\{ F_c \cdot \sum_{j \in J} j \cdot z_j + \mathbb{E}_{\xi} \left[ Q(x, \xi) \right] \right\}$$

s.t.

$$x \in X$$

where for an outcome $\xi$ of $\tilde{\xi}$, we have:
\[ Q(x, \xi) = \min \left\{ \sum_{j \in J} q_j \cdot y_j(\xi) \right\} \]  
\text{subject to} \quad \sum_{i \in I} \zeta_i(\xi) \cdot x_{ij} - y_j(\xi) \leq C, \forall j \in J \]  
\[ y_j(\xi) \geq 0, \forall j \in J \]  
Let \( L = \{1, 2, \ldots, L\}, L \in \mathbb{N}^+ \). If \( \xi \) has a finite discrete distribution \( \{\xi_l^i, p_l\}, l \in L, p_l > 0, \forall l \in L, \) then the realization probability of \( \xi_l^i \) of \( \xi \), then \( (\text{I}) \) is an ordinary linear program with a dual decomposition structure, as shown below.

**Deterministic Equivalent**

\[ \min \left\{ F_c \cdot \sum_{j \in J} j \cdot z_j + \sum_{l=1}^{L} p_l \sum_{j \in J} y_j(\xi_l^i) \right\} \]  
\text{subject to} \quad \sum_{i \in I} \zeta_i(\xi_l^i) \cdot x_{ij} - y_j(\xi_l^i) \leq C, \forall j \in J, \forall l \in L \]  
\[ y_j(\xi_l^i) \geq 0, \forall j \in J, \forall l \in L \]  
\[ x \in X \]

Depending on the number of realizations of \( \xi \), i.e. \( L \), this linear mixed integer program may become very large in scale, but its particular block structure can be exploited by specially designed algorithms such as the \( L \)-shaped method, (Ahmed and Shapiro (2002), Birge (1997); Birge and Tang (1993), Kall and Wallace (1994), Tait (2013)).

3. SOLUTION METHOD

As mentioned earlier, computing the expected value \( \mathbb{E}_\xi [Q(x, \xi)] \), for a given disassembly task assignment to workstations, is quite impossible. Even, for discrete distributions of \( \xi \), the number of linear programs of type \( (\text{II}') \) to solve may be a huge one. Although the exact evaluation of the expectation term in \( (\text{II}') \) is possible, its optimization presents serious difficulties, (Birge and Louveaux (1997), Santoso et al. (2005)). Indeed, \( \mathbb{E}_\xi [Q(x, \xi)] \) is implicitly defined. To deal with these difficulties, we present a solution strategy combining a method, latin hypercube sampling, with the \( L \)-shaped algorithm. We begin by introducing the latin hypercube sampling and the \( L \)-shaped algorithm, then, the solution strategy developed is detailed.

3.1 Latin Hypercube Sampling

The Latin hypercube sampling is a Monte Carlo technique, having the efficient manner to sample the range of each random variable. Under the assumption of random variables independence, latin hypercube sampling generates, in the following manner, a sample of size \( \Lambda \) from \( \xi = (t_1, t_2, \ldots, t_N) \) in consistency with the probability space \( (\Xi, \mathcal{F}, P) \), (Helton and Davis (2003), Helton et al. (2006)). The range of each variable \( t_i, i = 1, 2, \ldots, N \), is divided into \( \Lambda \) disjoint intervals of equal probability and one value is randomly selected from each interval. Then, the \( \Lambda \) values obtained for \( t_1 \) are paired at random, without replacement, with the \( \Lambda \) values obtained for \( t_2 \). These \( \Lambda \) pairs are randomly combined without replacement with the \( \Lambda \) values of \( t_3 \) to form \( \Lambda \) triples. This process is continued until a set of \( \Lambda N \)–tuples is formed. These \( N \)–tuples constitute the Latin hypercube sample.

3.2 The \( L \)-shaped Algorithm

The main idea of the \( L \)-shaped method is to approximate the nonlinear term in the objective function of the two-stage stochastic problems, (Birge (1997), Birge and Tang (1993)). Consider the formulation \( (\text{II}) \) of the SDLBP. Assume a finite realizations set \( \Xi \) of the stochastic vector \( \xi \) such as \( |\Xi| = L \). The \( L \)-shaped algorithm, adapted to the SDLBP, processes as follows. Counters \( r, h, v \) are respectively used for the feasibility and optimality cuts as well for the algorithm iterations. Constraint (16) is the matrix version of the cycle time constraints (13); \( I \) represents the identity matrix.

**L–shaped Algorithm**

**Step 0.** Set \( r = h = v = 0 \)

**Step 1.** Set \( v = v + 1 \). Solve the following LP:

\[ \min \left\{ F_c \cdot \sum_{j \in J} j \cdot z_j + \varphi \right\} \]  
\text{subject to} \quad x \in X \]  
\[ D_{r^v} \cdot x = d_{r^v}, v = 1, 2, \ldots, r \]  
\[ E_{h^v} \cdot x + \varphi \geq e_{h^v}, v = 1, 2, \ldots, h \]  
\[ x \text{ mixed binary}, \varphi \geq 0 \]

Let \( (x^v, \varphi^v) \) be an optimal solution.

**Step 2.** For \( l = 1, 2, \ldots, L \), solve the following LP:

\[ \min Z = a^T \cdot u \]  
\text{subject to} \quad T_i \cdot x^v - I \cdot y - I \cdot u \leq C \]  
\[ y \geq 0, u \geq 0 \]

\( a = (1, \ldots, 1)^T \), until for some \( l \) the optimal value \( Z > 0 \). In this case, let \( \sigma^v \) be the associated simplex multipliers; define

\[ D_{r^v+1} = (\sigma^v)^T \cdot T_i \]

and

\[ d_{r^v+1} = (\sigma^v)^T \cdot h_i \]

in order to generate constraint called a feasibility cut of type (14). Set \( r = r + 1 \), add constraint type (15) and return to **Step 1.** If \( \forall l \in L, Z = 0 \), go to **Step 3.**

**Step 3.** For \( l = 1, 2, \ldots, L \), solve the LP:

\[ \min W = q_l^T \cdot y \]  
\text{subject to} \quad T_i \cdot x^v - I \cdot y \leq C \]  
\[ y \geq 0 \]

Let \( \omega_l^v \) be the simplex multipliers associated with an optimal solution of problem \( l \) above; define
Step 1 SAA Procedure

The problem (explicitly defined by an optimization problem, the program Step 2) optimality cuts (15): if for a given solution \(\hat{x}\) isn't feasible, then, a constraint of type (14) cutting \(\hat{x}\) is generated.

2) optimality cuts (15): in Step 1 of the L-shaped algorithm, a new variable, \(\varphi\), is introduced. It verifies the inequality \(\varphi \geq \sum_{l=1}^{L} \rho_{l} \cdot Q(\bar{x}, \xi_{l})\). Since each \(Q(x, \xi)\) is implicitly defined by an optimization problem, the program in Step 1 is solved without this inequality. If for a given feasible solution, \((\bar{x}, \hat{\varphi})\), of Step 2, we have \(\hat{\varphi} \geq \sum_{l=1}^{L} \rho_{l} \cdot Q(\bar{x}, \xi_{l})\), then, \((\bar{x}, \hat{\varphi})\) is an optimal solution, otherwise, a constraint of type (15) cutting \(\bar{x}\) is generated.

3.3 Sample Average Approximation Strategy

The SAA strategy deals with an approximate computation of the expected recourse cost. It allows the computation of lower and upper bounds of the optimal solution of problem (II). A random sample of \(\xi\) generated by the LHS having size \(\Lambda\), is considered. Then, the term \(E_{\xi}[Q(\bar{x}, \xi)]\) is approximated by the sample average function \(\frac{1}{\Lambda} \sum_{l=1}^{\Lambda} Q(\bar{x}, \xi_{l})\). Thus, the problem (II) is approximated by the problem (III) while \(p_{l} = \frac{1}{\Lambda}, \forall l = 1, 2, \ldots, L\). The problem (III) is solved with the L-shaped algorithm.

SAA Procedure

Step 1. Lower Bound Estimation:
generate, using LHS, \(\Omega\) independent random samples \((\xi_{1,1}^{n}, \xi_{1,2}^{n}, \ldots, \xi_{1,\Lambda}^{n})\), \(n = 1, 2, \ldots, \Omega; \Lambda\) is the size of each sample. Solve the corresponding problem (III) with the L-shaped algorithm \(\forall n = 1, 2, \ldots, \Omega\). Compute an optimal solution \(x_{n}^{\Lambda}\) and the corresponding objective value \(\gamma_{n}^{\Lambda}\), for each value of \(n\).

Compute a lower bound: \(L_{B_{\Lambda\Omega}} = \frac{1}{\Omega} \sum_{n=1}^{\Omega} \gamma_{n}^{\Lambda}\).

If \(\gamma^{*}\) represents the optimal objective value of (II), then, we have \(E(\gamma_{\Lambda}) \leq \gamma^{*}\) and \(\gamma_{\Lambda} \rightarrow \Lambda \rightarrow \infty \gamma^{*}\) with probability 1; \(\gamma_{\Lambda}\) represents the optimal objective value of problem (III), where \(L = \Lambda\) and \(p_{l} = 1/\Lambda, \forall l = 1, 2, \ldots, L\). As \(L_{B_{\Lambda\Omega}}\) is an unbiased estimator of \(E(\gamma_{\Lambda})\), we have \(L_{B_{\Lambda\Omega}} \leq \gamma^{*}\), (Mak et al. (1999)), Santoso et al. (2005).

The variance \(\sigma_{L_{B_{\Lambda\Omega}}}^{2}\) of \(L_{B_{\Lambda\Omega}}\) is calculated as follows:

\[
\sigma_{L_{B_{\Lambda\Omega}}}^{2} = \frac{1}{\Omega - 1} \sum_{n=1}^{\Omega} (\gamma_{n}^{\Lambda} - L_{B_{\Lambda\Omega}})^{2}.
\]

Step 2. Upper Bound Estimation:
genenerate, using LHS, a sample \((\xi_{1}, \xi_{2}, \ldots, \xi_{\Lambda'})\) of size \(\Lambda'\) independent from those generated in Step 1. Let \(x_{\Lambda'}^{n}\) be a feasible solution obtained in Step 1; \(x_{\Lambda'}^{n}\) should be the one for which the objective value of problem (III) is minimum, with \(L = \Lambda', p_{l} = 1/\Lambda', \forall l = 1, 2, \ldots, \Lambda'\). Let \(c_{n}^{\Lambda'} = F_{c} \sum_{j \in J} j \cdot z_{j}\) and \(Q_{n}^{\Lambda'} = \frac{1}{\Lambda'} \sum_{l=1}^{\Lambda'} Q(x^{n}, \xi_{l})\), under the solution \(x_{\Lambda'}^{n}\).

Compute an upper bound: \(U_{B_{\Lambda'}} = c_{n}^{\Lambda'} + Q_{n}^{\Lambda'}\).

\(U_{B_{\Lambda'}}\) is an unbiased estimator of \(\gamma_{n}^{\Lambda'}\), where \(\gamma_{n}^{\Lambda'} = c_{n}^{\Lambda'} + E_{\xi} [Q(\bar{x}^{n}, \xi)]\); \(\gamma_{n}^{\Lambda'} \geq \gamma^{*}\), we have then, \(U_{B_{\Lambda'}} \geq \gamma^{*}\).

The variance \(\sigma_{U_{B_{\Lambda'}}}^{2}\) of \(U_{B_{\Lambda'}}\) can be estimated with:

\[
\sigma_{U_{B_{\Lambda'}}}^{2} = \frac{1}{\Lambda' - 1} \sum_{l=1}^{\Lambda'} (c_{n}^{\Lambda'} + Q(x^{n}, \xi_{l}) - U_{B_{\Lambda'}})^{2}.
\]

4. ILLUSTRATIVE EXAMPLE

The SAA procedure has been applied to the compass example illustrated on Figure 2. The product is made of seven components: (1) wheel, (2) left leg, (3) right leg, (4) left fixation screw, (5) lead, (6) tip and (7) right fixation screw. The ways of completely disassembling the product is modeled with the AOG shown on the Figure 1. The data associated with the product are grouped in Table 1 below. The method presented in this study was implemented in Microsoft Visual C++ 2008. ILOG CPLEX 12.4 was used to solve the model on a PC with Pentium(R) Dual-Core CPU 2.30 GHz and 3Go RAM.

Solution results

Lower and upper bounds values, after optimization for the compass example, are summarized in Table 2. Table 3 represents the solution corresponding to the upper bound. Note that just a subset of disassembly tasks is selected to completely disassemble the compass.

Fig. 2. Compass example
Table 1. The sample product associated data.

<table>
<thead>
<tr>
<th>Task</th>
<th>$\mu$</th>
<th>$\sigma$</th>
<th>subassembly</th>
<th>component</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.21</td>
<td>0.05</td>
<td>1 : 5</td>
<td>6 : 7</td>
</tr>
<tr>
<td>2</td>
<td>0.21</td>
<td>0.05</td>
<td>1 : 3 : 6 , 7</td>
<td>4 : 5</td>
</tr>
<tr>
<td>3</td>
<td>0.50</td>
<td>0.10</td>
<td>2, 4, 5</td>
<td>1 : 3</td>
</tr>
<tr>
<td>4</td>
<td>0.21</td>
<td>0.05</td>
<td>1 : 3</td>
<td>4 : 5</td>
</tr>
<tr>
<td>5</td>
<td>0.50</td>
<td>0.10</td>
<td>2, 4, 5/3, 6, 7</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>0.21</td>
<td>0.05</td>
<td>1 : 3</td>
<td>6 : 7</td>
</tr>
<tr>
<td>7</td>
<td>0.50</td>
<td>0.10</td>
<td>3, 6, 7</td>
<td>1 : 2</td>
</tr>
<tr>
<td>8</td>
<td>0.21</td>
<td>0.05</td>
<td>–</td>
<td>2 : 4 : 5</td>
</tr>
<tr>
<td>9</td>
<td>0.50</td>
<td>0.10</td>
<td>–</td>
<td>1 : 2 : 3</td>
</tr>
<tr>
<td>10</td>
<td>0.21</td>
<td>0.05</td>
<td>–</td>
<td>3 : 6 : 7</td>
</tr>
</tbody>
</table>

Under the upper bound solution, the number of workstations is 2. Tasks $\{2, 6\}$ are assigned to the 1st workstation and task 9 to the second one. The overall idle time of the disassembly line, if mean time of each task selected is considered, is 0.10. This solution was provided in 10.5 seconds.

Table 3. Upper bound solution

<table>
<thead>
<tr>
<th>Task</th>
<th>$\mu$</th>
<th>$\sigma$</th>
<th>subassembly</th>
<th>component</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.21</td>
<td>0.05</td>
<td>1 : 3 : 6, 7</td>
<td>4 : 5</td>
</tr>
<tr>
<td>6</td>
<td>0.21</td>
<td>0.05</td>
<td>1 : 3</td>
<td>6 : 7</td>
</tr>
<tr>
<td>9</td>
<td>0.50</td>
<td>0.10</td>
<td>–</td>
<td>1 : 2 : 3</td>
</tr>
</tbody>
</table>

5. CONCLUSION

In this paper, the DLBP was studied under uncertainty and a two-stage stochastic linear mixed integer program with fixed recourse was developed. Disassembly task times were assumed random variables with known normal probability distributions. To solve the problem efficiently, the SAA strategy was proposed. This strategy combined the L-shaped algorithm and the LHS Monte Carlo technique. The SAA procedure provided lower and upper bounds of the optimal solution for the SDLBP. Deviations of these bounds were also provided. To illustrate the SAA strategy, a case example was presented.

Our next objective is to test the SAA strategy on a data set of industrial problems in order to evaluate its practical efficiency. Such an evaluation should show if the developed method can afford the solution of the real-size problems characterized by a large number of alternatives and subassemblies in reasonable computational time. Further, the case of partial disassembly under uncertainty will be investigated.

REFERENCES
