Disassembly line balancing under high variety of end of life states using a joint precedence graph approach

Robert J. Riggs, Olga Battaïa, S. Jack Hu

Disassembly line balancing under high variety of end of life states using a joint precedence graph approach

Robert J. Riggs [a,b,*], Olga Battaïa [c], S. Jack Hu [a,d]

[a] Department of Industrial and Operations Engineering, The University of Michigan, Ann Arbor, MI, USA
[b] Department of Industrial Engineering, Clemson University, Clemson, SC, USA
[c] Laboratory of Informatics, École Nationale Supérieure des Mines de Saint-Etienne, Saint-Etienne, France
[d] Department of Mechanical Engineering, The University of Michigan, Ann Arbor, MI, USA

Keywords:
Disassembly line balancing (DLB)
Joint precedence graph
End of life (EOL) condition
Stochastic line balancing

A B S T R A C T

Disassembly is an important aspect of end of life product treatment, as well as having products disassembled in an efficient and responsible manner. Disassembly line balancing is a technique that enables a product to be disassembled as efficiently and economically viable as possible; however, considering all possible end of life (EOL) states of a product makes disassembly line balancing very difficult. The EOL state and the possibility of multiple recovery options of a product can alter both disassembly tasks and task times for the disassembly of the EOL product. This paper shows how generating a joint precedence graph based on the different EOL states of a product is beneficial to achieving an optimal line balance where traditional line balancing approaches are used. We use a simple example of a pen from the literature to show how a joint disassembly precedence graph is created and a laptop example for joint precedence graph generation and balancing. We run multiple scenarios where the EOL conditions have different probabilities and compare results for the case of deterministic task times. We also consider the possibility where some disassembly task times are normally distributed and show how a stochastic joint precedence graph can be created and used in a stochastic line balancing formulation.

1. Introduction

The end of life treatment of products is a very important topic concerning manufacturers, consumers, governments, and society as a whole. The waste from these products can have negative impacts on the environment creating a less sustainable future. Many national governments (e.g., Japan, Canada, and Taiwan), the European Union members, and 23 states in the US have adopted extended producer responsibility (EPR) principle based legislation for end-of-use treatment of products [1]. Due to this fact, manufacturers are increasingly recovering, remanufacturing, and reprocessing postconsumer products. For instance, the European Union (EU) created directives requiring companies to be responsible, free of charge, for the end of life treatment of many products including electronics and automobiles. The Waste Electrical and Electronic Equipment (WEEE) directive, together with the Restriction of Hazardous Substances (RoHS) directive, imposes the responsibility of disposal of electrical waste and electronic equipment to the manufacturers of the equipment. Disassembly in some form is the common thread for all forms of product recovery. Product recovery required by law is being discussed and implemented around the world as countries become conscious of the need for sustainable practices and the potential economic benefit of remanufacturing.

Developing a disassembly system enables products to be disassembled in a responsible and efficient manner. There are many decisions to be made for the design and layout of a disassembly system, such as the number of workstations and task assignment. Line balancing (LB) is a decision making tool that assigns manufacturing tasks to a set number of workstations with the objective of minimizing some performance objective, such as the maximum difference in total task time between workstations. LB was first considered in assembly settings but the approach can be extended to the application of disassembly [2,3]. Meacham et al. [4] determined the optimal disassembly configurations for both single and multiple product types for meeting a specified demand for recovered components and subassemblies. Gungor and Gupta [5] investigated the disassembly line balancing problem (DLBP) in the presence of task failures. Some disassembly tasks may not be completed due to a

* Corresponding author at: Department of Industrial and Operations Engineering, The University of Michigan, Ann Arbor, MI, USA. Tel.: +1 8322477620.
E-mail address: robriggs@umich.edu (R.J. Riggs).

http://dx.doi.org/10.1016/j.jmsy.2014.11.002

product defect that affects the rest of the disassembly process and some disassembly steps may need to be altered or skipped entirely. Gungor et al. [6] also discussed the challenges that come from disassembly line balancing, ranging from product, disassembly line, demand, and assignment complications.

Altekin et al. [7] investigated the DLBP as a partial disassembly with limited supply of a single product and its subassemblies. They created two formulations, one that maximizes profit over a single disassembly cycle and the other maximizes profit over the entire planning horizon. Tang and Zhou [8] created a disassembly Petri net model for the hierarchical modeling in order to derive a disassembly path with the maximal benefit in the presence of some defective components. Their algorithm for balancing disassembly lines seeks to maximize the productivity of a disassembly system. McGovern et al. [9] proved that the disassembly line balancing problem is NP-complete and presents a genetic algorithm for balancing disassembly lines and used priori instances to evaluate any disassembly line balancing technique.

Altekin et al. [10] developed an MIP formulation for solving partial disassembly-line balancing that determines simultaneously multiple decision points such as the cycle time and task assignment of the line and the number of stations to be opened. Tripathi et al. [11] proposed a fuzzy disassembly optimization model that determines the disassembly sequence and the depth of disassembly to maximize net revenue from EOL disposal of products. Fuzzy control theory is used to account for the uncertainty associated with the quality in returns. Koc et al. [12] developed both an integer and dynamic programming formulations for the DLBP by using AND/OR graphs to check for feasibility of the solution.

Altekin et al. [13] developed a two-step approach to rebalance a disassembly line when task failures occur that lead to successor tasks being infeasible. Ma et al. [14] created a model that simultaneously solves the decisions of disassembly level, sequence, and EOL options for components or subassemblies in a product in the parallel disassembly environment for the objective of maximizing profit. The parallel disassembly environment allows two or more parts of subassemblies to be disassembled at the same time. McGovern et al. [2] formulated the disassembly line balancing problem and presented case studies and solution methods in their book *The Disassembly Line: Balancing and Modeling*. Rickli et al. [15] created a multi-objective genetic algorithm that chose the optimal partial disassembly sequence with respect to operation and recovery costs, as well as revenue and environmental impact. Kalayci and Gupta [28] used an artificial bee colony optimization technique to solve the disassembly line balancing problem with sequence dependence. They compare their bee colony algorithm to six metaheuristic approaches from the literature and demonstrate the effectiveness of the algorithm in solving the disassembly line balancing problem with multi-objectives. Riggs et al. [29] developed a two-stage sequence generation approach for the partial disassembly of products that have sequence dependent task times.

Ilgin and Gupta [16] provided an extensive review of the state of the art for EOL product recovery and its relationship with disassembly for remanufacturing and recycling. Tuncel et al. [17] used a reinforcement learning approach for disassembly line balancing where uncertainty comes from a single source, demand fluctuations for the disassembled components. They assumed all incoming products have the same identical structure and are in the same EOL condition. Other approaches have focused on uncertainty coming from the EOL product itself, which results in task failures or additional processing [5,13].

There are many different approaches used in the literature to represent the relative order of disassembly of a product at its EOL. The approaches in the literature are referred to as either tree-based or state-based for the structure of disassembly [18]. A tree-based approach assumes there is only one way to disassemble a product into its components and subassemblies, similar to a precedence graph. The state-based approach shows multiple possible disassembly paths, such as shown in an AND/OR graph or transition matrix [19]. Many approaches in the literature use AND/OR graphs [20] to show the possibility of different disassembly paths; however, when using an AND/OR graph, only one possible disassembly path is chosen and the line balance resulting from that path must be used for all products at their EOL.

For stochastic disassembly line balancing, the literature has less breadth than the literature for deterministic disassembly line balancing. Agrawal et al. [21] developed an ant colony algorithm to solve for a mixed-model U-shaped disassembly line to handle the situation where completion times are stochastic and there is task time variability due to the human factor. Aydemir-Karadag and Turkbey [20] developed a multi-objective genetic algorithm to determine the best line balance that considered stochastic task times where station paralleling is allowed. Bentaha et al. [30,31,32] developed methods for disassembly line balancing where there is uncertainty in task processing times.

Many line balancing approaches for disassembly are the same as those used for assembly but with a simple inverse or alteration of the precedence graph [3]. "Disassembly modeling and planning can be more challenging than assembly because its terminal goal is not necessarily fixed" due to changing market demand for reused components and subassemblies [22]. The research for the disassembly line balancing problem (DLBP), up to this point, has not considered how the end of life (EOL) condition and treatment options for the same components will alter the precedence graph, either by representation or the time values for each task. At a product's EOL, disassembly of certain components or modules may have a different time distribution for the same disassembly task due to the variation in the EOL states. Some components can be grouped together and the entire module can be reused so complete disassembly is not required for each component. This will lead to tasks being skipped and having zero time. In addition, even if all EOL states are considered for the line balancing algorithm, it is difficult to include all this information and have the algorithm solve quickly. Previous approaches use AND/OR graphs to model the possibility of having different disassembly paths, but only one path is chosen and every product at its EOL takes this path for disassembly. Our approach is able to consider many disassembly paths that are weighted into a single graph.

Our objective in this paper is to develop and validate a method originally used in the area of mixed model assembly line balancing, joint precedence graph generation, and enhance it for the application of disassembly with multiple component/module EOL states. We treat each EOL state of a product as a different model variant and generate a disassembly joint precedence graph from all EOL states that can then be inputted into a line balancing algorithm. The contribution of our work is the development of a method that: (1) considers the existence of all EOL states with certain probabilities so that different line balancing techniques where a single precedence graph is required can be applied, (2) considers components/modules having multiple EOL treatment possibilities, and (3) is tractable and reduces the complexity of dealing with many EOL states and treatment possibilities during the optimization stage (i.e. we preprocess the complexity of many EOL states during the formulation of the joint precedence graph). To the best of our knowledge, this is the first method for balancing disassembly lines that is able to consider many EOL states and treatment options into a single precedence graph.

The remainder of the paper is organized as follows: Section 2 presents the method of generating disassembly joint precedence graphs and stochastic disassembly joint precedence graph. Section 3 outlines the deterministic and stochastic line balancing

formulations. Section 4 presents line balancing results from a theoretical example for disassembly of a laptop taken from the literature and Section 5 contains discussion. Conclusions are discussed in Section 6.

2. Generation of disassembly joint precedence graphs

A precedence graph is a type of directed graph where arcs connect nodes to one another. The nodes are the tasks required for disassembly and the connecting arcs have a particular direction that shows the order of assembly/disassembly tasks. Boysen et al. [23] use a joint precedence graph for balancing mixed-model assembly lines by considering all products to be assembled on the same assembly line. This approach can be used to model precedence relations for disassembly lines where instead of considering different models of a product, the joint precedence graph will reflect every possible EOL state of a product. The disassembly line can then be balanced using existing line balancing optimization techniques from the literature. The EOL condition refers to the condition of an individual component or module and the EOL state is the combination of all EOL conditions for all components and modules in the assembly. The EOL state will have a probability associated with it and the summation of all EOL states will add up to one.

2.1. Method for generating joint precedence graphs

The starting point for creating a disassembly joint precedence graph for a product is to create a baseline disassembly precedence graph based on the product having "off the assembly line" quality. A product with off the assembly line quality has no quality defects and the EOL condition is like new. The baseline disassembly precedence graph should reflect all disassembly tasks that need to be accomplished to have the product disassembled to the required ending state.

Through various sampling methods, products at their end of life can be collected and the impact of different types of damage to the core can be assessed for how it can affect disassembly task times and the probability that type of damage will present itself. Damage to a product can increase or decrease task times resulting in variation of individual task times. Damage may also cause multiple tasks times to change simultaneously. For example, there is a disassembly task to remove component X and a component Y, first X and then Y. Component X has a 25% chance of being damaged, and the task time to remove component X is greater than the baseline case if component X is damaged. If component X is damaged, then the chance that component Y is damaged is 50% and the task time to remove Y is also greater than the baseline case, but if component X is not damaged then component Y would not be damaged and the baseline case for each task time is unchanged. This is an example of a dependency between two tasks. An example of independence between disassembly tasks is that damage to component X has no effect on the condition of component Y. Another possible phenomenon when disassembling a product is a task failure due to either damage to the product or absence of a component or module. Certain components or modules may become loose or dislodged during transport to its EOL treatment site or consumers may remove certain components or modules before disposing of the product. Both these cases can cause certain disassembly task times to be zero or negligible.

Once all possible component and module EOL states are determined, including both the difference in disassembly task times and the probability of having a difference in certain task times versus the baseline case, through simple combinatorics the probability of each EOL state can be enumerated and subsequent EOL state disassembly precedence graphs created. Since every state is enumerated, no two states will have exactly the same EOL profile. The EOL state disassembly precedence graphs will be very similar to the baseline disassembly precedence graph, except now a probability will be associated with each precedence graph, some tasks may be skipped, and the task times may be different for certain tasks.

The joint precedence graph is created by a weighted average approach that considers all precedence graphs for each EOL state. Task times in each precedence graph will be referred to as t_{iq}, where i is the task number and q is the EOL state. Each EOL state will have a probability p_q, where $0 \le p_q \le 1$, and the sum of all p_q values for $q \in Q$ is equal to 1, Q being the set of all EOL states. The average task time for the joint precedence graph (1) and the joint precedence arc set (2) are found using the following definitions:

$$t_i = \sum_{q \in Q} p_q t_{iq} \quad \forall i \in I \tag{1}$$

$$E = \bigcup_{q \in Q} E_q \backslash \{reducdant \quad arcs\} \tag{2}$$

Eq. (1) creates a weighted average task time for each disassembly task in the final joint precedence graph that considers all EOL states Q. E is the arc set for the final joint precedence graph, and is the union of all arcs in the individual EOL state precedence graphs minus any redundant arcs. Redundant arcs are created when there is a task failure, or zero task time, and a task is skipped because it is no longer required for disassembly.

2.2. Method for generating a stochastic joint precedence graph

Disassembly is typically a manual removal process, which makes many of the task times a random variable. We assume that each task time that is a random variable will be normally distributed with a known mean and variance. The normally distributed task times will be referred to as μ_{iq} with variance σ^2_{iq}, for all EOL states q. The weighted task time and variance can be calculated using Eqs. (3) and (4), respectively.

$$\mu_i = \sum_{q \in Q} p_q \mu_{iq} \quad \forall i \in I \tag{3}$$

$$\sigma_i^2 = \sum_{q \in Q} p_q((\mu_{iq} - \mu_i)^2 + \sigma_{iq}^2) \quad \forall i \in I \tag{4}$$

Eqs. (3) and (4) are derived based on a mixture distribution of normal variables. When the task time is zero for a certain EOL state, there will be no variance associated with that particular task time in that state and the variance will very simply be zero. It is obvious that the mixture of two or more normal random variables is not always normally distributed; however, we use the calculation of a variance in (4) to show the relative spread of having a mixture of two or more task times.

2.3. Example: joint precedence graph generation

Fig. 1 contains a liaison graph and a cross-section view of a pen taken from a paper by De Fazio and Whitney [24]. Based on the information taken from their paper, a theoretical baseline disassembly precedence graph can be created and theoretical disassembly task times added for each component, shown in Fig. 2. For simplicity, the disassembly task for each component, represented as nodes in the precedence graph, will include all tasks to remove that particular component from the core. It is possible that multiple tasks are required for component removal but all task times pertaining to removal of a single component will be summed together into a single time. For this theoretical pen disassembly example, complete disassembly is required.

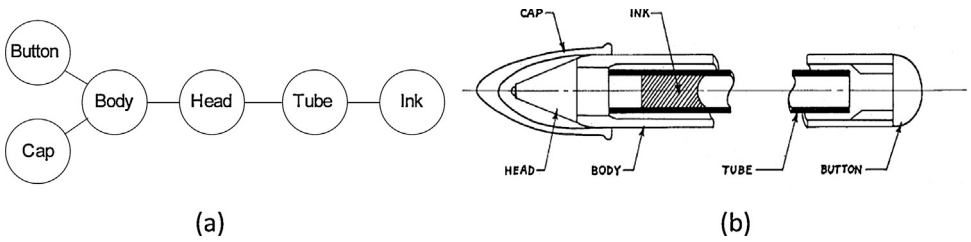


Fig. 1. (a) Pen liaison graph and (b) pen cross-section [24].

Table 1
(a) Coded EOL state and (b) disassembly probability of each EOL state.

EOL state	Cap	Head	Tube
(a)			
1	1	1	1
2	1	0	1
3	1	0	0
4	0	1	1
5	0	0	1
6	0	0	0
(b)			
1	0.75	0.80	1.00
2	0.75	0.20	0.50
3	0.75	0.20	0.50
4	0.25	0.80	1.00
5	0.25	0.20	0.50
6	0.25	0.20	0.50

The values shown next to each node in Fig. 2 are the baseline disassembly task times. The body of the pen will be the only remaining component after all other components are removed so the 1 time unit (tu) is the amount of time it will take the worker to place the body in a bin or receptacle. EOL data is collected for pens that will be disassembled and the following is the quality information:

- 25% chance that the cap is missing for disassembly, resulting in a zero task time.
- 20% chance that the head is damaged so disassembly time increases from 2 tu to 4 tu.
  - If the head is damaged then 50% chance the tube is damaged and the disassembly time increases from 1 to 3 tu.

Since there are 3 ($x = 3$) components that have a disassembly task different from the baseline and there are $2^x = 8$ possible precedence graphs to consider. However, since one of the EOL possibilities has a dependency between 2 components (head and tube), the 8 possible precedence graphs decrease to 6. The coded EOL condition table is shown in Table 1(a). If there is a 1 in the cell, then that component has the baseline precedence graph disassembly task time and if there is a 0, then that component has the altered disassembly time. Table 1(b) shows the actual probability of the EOL condition for each component. The cap and head components have probabilities that are all less than 1 because they are independent quality types but the tube has a probability of 0.5 or 1 because if the head is the baseline EOL state, the probability that the tube is in a state other than the baseline is 0. This adjustment allows the

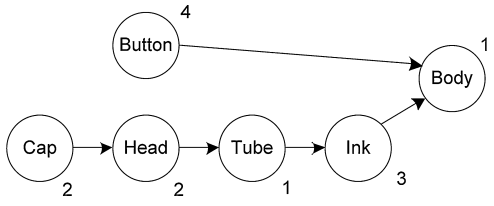


Fig. 2. Pen disassembly precedence graph.

Table 2
Probability and time data for each EOL state.

EOL state	Probability	Cap	Head	Tube
1 (baseline)	0.600	2	2	1
2	0.075	2	4	1
3	0.075	2	4	3
4	0.200	0	2	1
5	0.025	0	4	1
6	0.025	0	4	3

total probability of the 6 EOL states to add up to 1 since 2 of the EOL states are eliminated due to the dependency between two components. Table 2 shows the probability of each EOL state and the corresponding disassembly task time for the EOL state.

EOL state 1 is the baseline precedence graph and there is a 60% chance the pen will be in this EOL state. This EOL data can be incorporated into individual precedence graphs for each EOL state, shown in Fig. 3. Each precedence graph in the table has a number in parenthesis below that corresponds to its EOL state, and a probability of that EOL state. The cap node for EOL states 4, 5, and 6 are dashed because this is essentially a skipped task that results in 0 task time. After using Eqs. (1) and (2) to calculate the weighted average disassembly task time for each component i, t_i, and E, a joint precedence graph can be created, shown in Fig. 4. The disassembly task time for the cap decreased and the disassembly task times for the head and tube increased to reflect all the possible EOL states the pen can be in for disassembly.

Now let us assume that each task time that is non-zero is a normally distributed variable that has a standard deviation equal to 0.5 tu, or a 0.25 tu variance. Table 3 reflects the change that each non-zero task time has a variance of 0.25 tu. Applying Eqs. (3) and (4) from Section 2.2, we can generate a stochastic joint precedence graph, shown in Fig. 5.

The stochastic joint precedence graph in Fig. 5 is very similar the joint precedence graph in Fig. 4, except now each task time is a normal variable with a mean and variance.

3. Disassembly line balancing algorithm

Now that we can create a disassembly joint precedence graph, we introduce a line balancing algorithm that can use the joint precedence graph as an input. This is an example of one line balancing formulation but many can be used where a single precedence graph is utilized.

Table 3
Probability and time data for each EOL state.

EOL state	Probability	Cap, $N(\mu, \sigma^2)$	Head, $N(\mu, \sigma^2)$	Tube, $N(\mu, \sigma^2)$
1 (baseline)	0.600	$N(2, 0.25)$	$N(2, 0.25)$	$N(1, 0.25)$
2	0.075	$N(2, 0.25)$	$N(4, 0.25)$	$N(1, 0.25)$
3	0.075	$N(2, 0.25)$	$N(4, 0.25)$	$N(3, 0.25)$
4	0.200	0	$N(2, 0.25)$	$N(1, 0.25)$
5	0.025	0	$N(4, 0.25)$	$N(1, 0.25)$
6	0.025	0	$N(4, 0.25)$	$N(3, 0.25)$

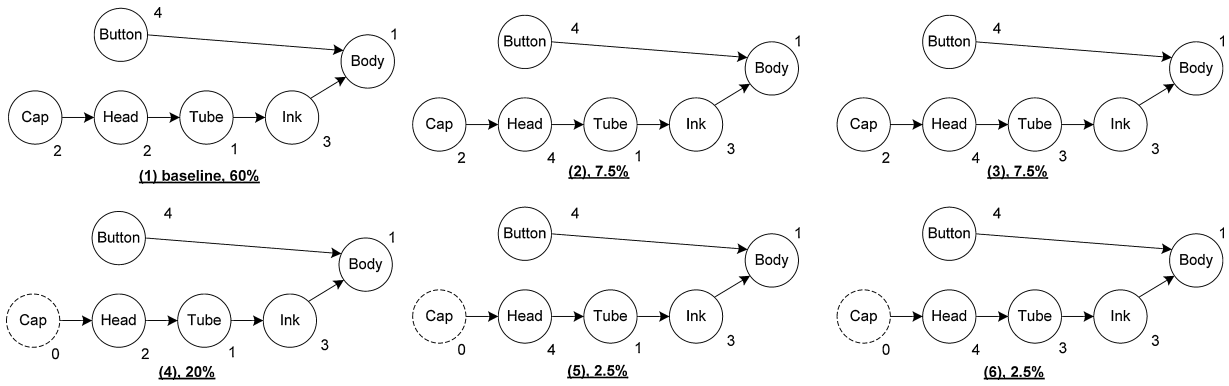


Fig. 3. EOL disassembly precedence graphs for pen.

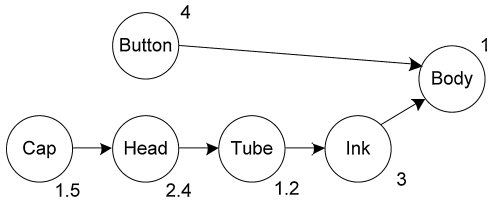


Fig. 4. Joint precedence graph for pen disassembly.

3.1. Deterministic disassembly line balancing formulation

Notation:

- Tasks: $i \varepsilon I$
- Workstations: $k \varepsilon K$
- Task Time: t_i
- Predecessor Set: $P(i)$
- Tool Sharing Set: $P(tool\_i)$
- Maximum Tasks Allowable: Z

Objective function:

$$\min \left(\max \left(\sum_{i \in I} X_{ik} t_i - \sum_{i' \in I} i' k' t_{i'} \right) \right) \quad \forall k, k' \in K \quad \text{and} \quad k' < k \tag{5}$$

Subject to:

$$\sum_{k \in K} X_{ik} = 1 \quad \forall i \in I \tag{6}$$

$$X_{ik} \leq \sum_{h=1}^{k} X_{i'h} \quad \forall i, \quad i' \in I, \quad i \neq i', \quad k \in K \quad \text{and} \quad i' \in P(i) \tag{7}$$

$$X_{ik} = 0, 1 \quad \forall i \in I \quad \text{and} \quad k \in K \tag{8}$$

The objective function (5) minimizes the maximum difference in total workstation time for a given number of workstations K. X_{ik} is the binary decision variable that is 1 if task i is assigned to workstation k and 0 otherwise. Constraint (6) is an occurrence constraint where every disassembly task must be assigned a workstation and only one workstation. Constraint (7) are the precedence constraints [25] where the workstation assignment of task i^l will be at the same workstation or an earlier workstation of task i, based on the predecessor set $P(i)$. Constraint (8) is a non-divisibility constraint that requires task i to not be split amongst multiple stations.

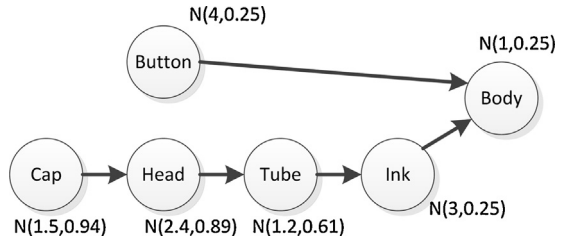


Fig. 5. Stochastic joint precedence graph for pen disassembly.

$$X_{ik} = X_{i'k} \quad \forall i, i' \in P(tool\_i), \quad k \in K \tag{9}$$

$$\sum_{i \in I} X_{ik} \leq Z \quad \forall k \in K \tag{10}$$

Constraint (9) is a tool sharing constraint where task i and i^l are tasks in a set $P(tool\_i)$ and have a common disassembly tool required to complete the task. In order to reduce cost for additional tooling, the disassembly tasks in this set must be completed at the same workstation. Constraint (10) is a total workstation task constraint that limits the number of disassembly tasks Z that can be performed at a single workstation. Since disassembly is typically manual, if workers are overloaded with too many tasks then the quality of their work can decrease and the time to complete tasks can increase.

To model the min-max function in CPLEX, the objective function is transformed and 2 additional constraints are added.

Objective function:

$$\min \quad Y \tag{5a}$$

Subject to:

$$\sum_{i \in I} X_{ik} t_i - \sum_{i' \in I} X_{i'k'} t_{i'} \geq -Y \quad \forall \quad k, \quad k' \in K, \quad \text{and} \quad k' < K \tag{5b}$$

$$\sum_{i \in I} X_{ik} t_i - \sum_{i' \in I} X_{i'k'} t_{i'} \leq Y \quad \forall \quad k, \quad k' \in K, \quad \text{and} \quad k' < K \tag{5c}$$

(6), (7), (8), (9), (10)

Objective function (5a) minimizes a float variable Y that is in constraint (5b) and (5c). Constraint (5b) ensures that the difference between total workstation time is greater than or equal to the negative of Y and constraint (5c) ensures that difference between the total workstation time is less than or equal to the positive value of Y.

3.2. Stochastic disassembly line balancing formulation

The stochastic disassembly line balancing formulation is very similar to the deterministic line balancing formulation in Section 3.1, except now we will take into consideration that each task time is a normal random variable and the total time variance for each workstation needs to be below some predetermined value. The

objective in (11) is to minimize the maximum difference between the total workstation times. The only change in nomenclature is now task times, V_i, are normal random variables with mean μ_i variance σ_i^2.

Objective function:

$$\min \left(\max \quad E\left[\sum_{i\in I} X_{ik} V_i - \sum_{i'\in I} X_{i'k'} V_{i'} \right] \right)$$
$$\forall k, k' \in K, \quad \text{and} \quad k' < k \tag{11}$$

Subject to:(6), (7), (8), (9), (10)

$$\sum_i X_{ik} \sigma_i^2 \leq \sum \forall k \in K \tag{12}$$

Constraint (12) is a total workstation variation constraint that requires each workstation to have a total variance below some value Σ. We take advantage that the sum of normally distributed variables is a normal distribution with mean that is the sum of all means and a variance that is the sum of all variances. For constraint (12), the value chosen for Σ will dictate how large the total workstation variation can be. The expectation can be removed from the objective function, based on Proposition 1, and we can transform the objective into two constraints so we can plug our formulation into CPLEX.

Objective function:

$$\min \quad M \tag{11a}$$

Subject to:

$$\sum_{i\in I} X_{ik}\mu_i - \sum_{i'\in I} X_{i'k'}\mu_{i'} \geq -M \quad \forall \quad k, \quad k' \in K, \quad \text{and} \quad k' < K \tag{11b}$$

$$\sum_{i\in I} X_{ik}\mu_i - \sum_{i'\in I} X_{i'k'}\mu_{i'} \leq M \quad \forall \quad k, \quad k' \in K, \quad \text{and} \quad k' < K \tag{11c}$$

(6), (7), (8), (9), (10), (12)

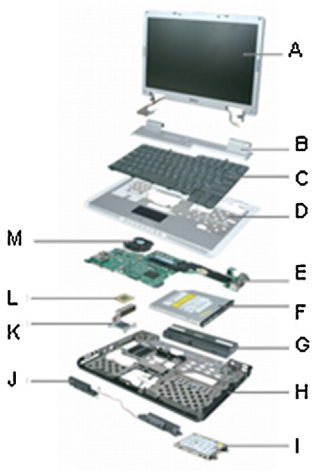
Proposition 1. *The* $E\left[\sum_{i\in I} X_{ik} V_i - \sum_{i'\in I} X_{i'k'} V_{i'}\right]$ $\forall i$ k, $k' \in K$, and $k' < K$, *where* μ_i *and* μ_i' *are the means for normally distributed variables, is equivalent to* $\sum_{i\in I} X_{ik}\mu_i - \sum_{i'\in I} X_{i'k'}\mu_{i'}$.

Proof. The expected value of a normal distribution is its mean, μ, and the difference between two normally distributed variables is a normal distribution with the mean being the difference in means. Therefore, $E[\sum_{i\in I} X_{ik} V_i - \sum_{i'\in I} X_{i'k'} V_{i'}] = E[\sum_{i\in I} X_{ik} V_i] - E[\sum_{i'\in I} X_{i'k'} V_{i'}] = \sum_{i\in I} X_{ik}\mu_i - \sum_{i'\in I} X_{i'k'}\mu_{i'}$. □

4. Disassembly line balancing: laptop example

The following laptop example is more realistic example compared to the pen example because laptops have components and modules that have tangible value for reuse and aftermarket sales. The disassembly times and EOL data are hypothetical and used to show the benefit of line balancing using a joint precedence graph versus a single precedence graph that does not consider every EOL state and component/module reuse possibilities. The laptop example has been used in a couple of papers as a demonstration of assembly and disassembly [26,27]. The exploded view of the laptop was found on dell's website. The dell laptop has 13 components, as shown in Fig. 6. A disassembly precedence graph was generated by analyzing the connections between the components and disassembly times estimated. The baseline precedence graph for the dell laptop is shown in Fig. 7.

The numeric values next to each node are the total disassembly task time for each task. An end node is added with 0 time so when



Parts List	
A	display assembly
B	hinge cover
C	keyboard
D	palm rest (with touch pad)
E	system board
F	optical drive
G	main battery
H	computer base
I	hard drive
J	speakers
K	microprocessor thermal-cooling assembly
L	microprocessor
M	fan

Fig. 6. Exploded view of dell laptop and parts list [21].

generating the different line balances, there is a common end task. The following is the collected EOL quality information:

- 30% chance component G (battery) is missing which results in a 0 task time.
- 35% chance component I (hard drive) is missing which results in a 0 task time.
- 20% chance component E (system board) has stripped screws so the time to remove E from the computer base increases from 5 tu to 8 tu.
- 30% chance components A (display), B (hinge), C (keyboard), and D (palm rest) do not need to be disassembled and can be left as a module to directly be reused in the computer aftermarket. Reusing the module will result in a 0 task time for A, B, and C but D will still take 4 tu.

The EOL information was derived to somewhat be based on realistic EOL scenarios, although the percentage of each EOL state and the task times are not based on an actual case study. It is very possible that consumers will keep the battery as a backup for their next computer if it is compatible, and the hard drive may be removed to save all data. Additionally, these components may be removed and sold in the aftermarket by individuals. It is assumed the system board is screwed into the computer base and small screws can present difficulties for workers to undue. The modularization of components A, B, C, and D for reuse is to simulate how different types of reuse are possible for the same components. If each

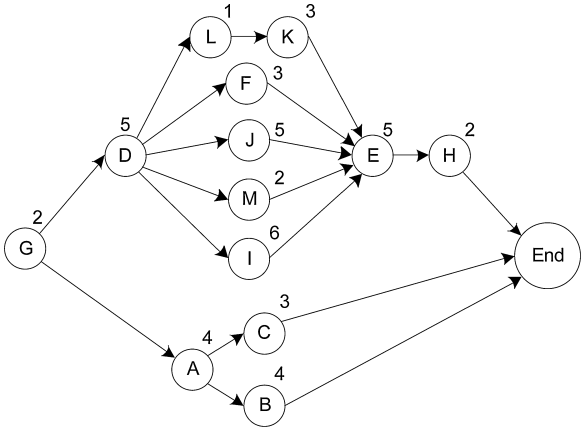


Fig. 7. Baseline disassembly precedence graph of laptop.

Table 4
Laptop EOL precedence graph information.

EOL State	Probability	Disassembly task times (tu)												
		A	B	C	D	E	F	G	H	I	J	K	L	M
1 (baseline)	0.207	4	4	3	5	5	3	2	2	6	5	3	1	2
2	0.089	0	0	0	5	5	3	2	2	6	5	3	1	2
3	0.111	4	4	3	5	8	3	2	2	6	5	3	1	2
4	0.048	0	0	0	5	8	3	2	2	6	5	3	1	2
5	0.089	4	4	3	5	5	3	0	2	6	5	3	1	2
6	0.038	0	0	0	5	5	3	0	2	6	5	3	1	2
7	0.048	4	4	3	5	8	3	0	2	6	5	3	1	2
8	0.020	0	0	0	5	8	3	0	2	6	5	3	1	2
9	0.111	4	4	3	5	5	3	2	2	0	5	3	1	2
10	0.048	0	0	0	5	5	3	2	2	0	5	3	1	2
11	0.060	4	4	3	5	8	3	2	2	0	5	3	1	2
12	0.026	0	0	0	5	8	3	2	2	0	5	3	1	2
13	0.048	4	4	3	5	5	3	0	2	0	5	3	1	2
14	0.020	0	0	0	5	5	3	0	2	0	5	3	1	2
15	0.026	4	4	3	5	8	3	0	2	0	5	3	1	2
16	0.011	0	0	0	5	8	3	0	2	0	5	3	1	2

Table 5
Results of laptop line balancing comparison.

EOL State	Probability	Line balance				Average throughput	Percent difference
		Wkst 1	Wkst 2	Wkst 3	Wkst 4		
Baseline	20.70%	GAD	CBLK	JI	FMEH	4.598	9.72
Joint	–	GADL	CBJ	KFI	MEH	5.045	

individual component is disassembled they can be inspected and determined fit or not for reuse, and if not fit they can be recycled; however, there is a chance the entire module is found fit as is and instead of completely disassembling the components, they can be left together and go straight into reuse.

There are $k = 4$ different components with various EOL states, so there are 16 possible EOL disassembly precedence graphs. Instead of showing each disassembly precedence graph, the information is shown in Table 4 with all relevant information.

Similar as before with the pen example, EOL state 1 is the baseline case and the probability of this EOL state presenting itself for disassembly is 20.7%, the highest of all EOL states. The data in Table 4 can be used to create a joint disassembly precedence graph for the laptop example, shown in Fig. 8. Versus the baseline case, the disassembly task time in the joint precedence graph for components A, B, C, I, and G decreased, while the disassembly task time for component E increased.

It is very difficult to include all 16 EOL states for the laptop example into a single line balancing algorithm and have it solve in a timely manner. For other realistic disassembly examples, the number of EOL states can increase dramatically as more EOL condition data and reuse possibilities are included into calculating EOL states. To show the effectiveness of creating a disassembly joint precedence graph for line balancing, we use CPLEX to determine two optimal line balances. The first line balance uses the joint precedence graph as an input and the second line balance uses the EOL state that has the highest probability of presenting, which in most cases will be the baseline precedence graph. We then created a simulation for each line balance and compared the average throughput between the two. When running CPLEX to obtain the optimal line balances, the solution time never exceeded 2 s.

For our simulation we assume the first workstation in the serial line cannot be starved and the final workstation cannot be blocked. We used the line balancing setup from Section 3 that minimized the maximum difference in total workstation time. We simulated the 2 line balances using Simul8 software and ran each simulation 5 times with different randomly generated arrival seeds. Each simulation was run for 2400 h to reach a steady state and the disassembly

Table 6
Workstation utilization results from simulation.

Baseline case			Joint case		
Workstation	Category	Percent	Workstation	Category	Percent
Wkst 1	Waiting (%)	0.00	Wkst 1	Waiting (%)	0.00
	Working (%)	70.70		Working (%)	86.09
	Blocked (%)	29.30		Blocked (%)	13.91
Wkst 2	Waiting (%)	0.00	Wkst 2	Waiting (%)	1.58
	Working (%)	68.36		Working (%)	83.39
	Blocked (%)	31.64		Blocked (%)	15.03
Wkst 3	Waiting (%)	0.00	Wkst 3	Waiting (%)	6.98
	Working (%)	68.34		Working (%)	83.11
	Blocked (%)	31.66		Blocked (%)	9.91
Wkst 4	Waiting (%)	0.00	Wkst 4	Waiting (%)	15.66
	Working (%)	100.00		Working (%)	84.34
	Blocked (%)	0.00		Blocked (%)	0.00
Average working percent		76.85	Average working percent		84.23

Table 7
Results of laptop line balancing sensitivity.

Component probability	EOL state	Probability	Line balance				Average throughput	Percent difference
			Wkst 1	Wkst 2	Wkst 3	Wkst 4		
50%	Baseline	6.25%	GAD	CBLK	JI	FMEH	4.444	13.07
	Joint	–	GDLM	ABJ	KFI	CEH	5.025	
40%	Baseline	12.96%	GAD	CBLK	JI	FMEH	4.546	12.30
	Joint	–	GADL	BJM	KFI	CEH	5.105	
30%	Baseline	24.01%	GAD	CBLK	JI	FMEH	4.65	7.31
	Joint	–	GADL	BFI	KJM	CEH	4.99	
20%	Baseline	40.96%	GAD	CBLK	JI	FMEH	4.762	2.29
	Joint	–	GDLF	ACI	BJM	KEH	4.871	
10%	Baseline	65.61%	GAD	CBLK	JI	FMEH	4.877	3.03
	Joint	–	GDLF	AMI	CKJ	BEH	5.025	
5%	Baseline	81.45%	GAD	CBLK	JI	FMEH	4.938	1.58
	Joint	–	GDLK	AMI	CFJ	BEH	5.016	

task times were turned into minutes, for example 2 tu equals 2 min. The time unit does not matter as we find the percent difference in throughput for the different simulations, which is unit less. Table 5 contains the line balancing task assignment for each workstation for the baseline and joint cases, as well as the average throughput and percent difference. We balanced for 4 workstations and this remained constant. Table 6 contains the average workstation utilization percentages (waiting, working, and blocking percents) for the simulation runs of each line balance.

4.1. Line balancing sensitivity

The results in Table 5 are for a combination of different probabilities of EOL states but it does not give any sort of understanding of the sensitivity between the probabilities of each EOL state versus balancing against the EOL state that has the highest probability of presenting for disassembly. We created and analyzed a series of line balances and simulations for having each of the 4 previously stated EOL component possibilities (battery and hard drive missing, system board damaged, monitor/keyboard module for reuse) where each condition has a 50%, 40%, 30%, 20%, 10%, and 5% probability of occurring. The same setup was performed for these scenarios as was done before with the simulation results shown in Table 7.

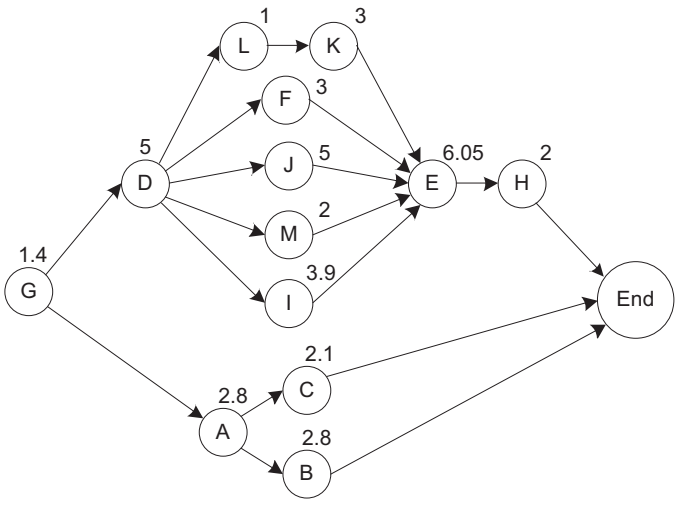


Fig. 8. Disassembly joint precedence graph for laptop example.

Table 8
Data input for the stochastic line balancing formulation, 0.5 tu standard deviation.

Task	Mean time	Variance	Standard deviation
A	2.80	3.54	1.88
B	2.80	3.54	1.88
C	2.10	2.07	1.44
D	5.00	0.25	0.50
E	6.05	2.30	1.52
F	3.00	0.25	0.50
G	1.40	1.02	1.01
H	2.00	0.25	0.50
I	3.90	8.35	2.89
J	5.00	0.25	0.50
K	3.00	0.25	0.50
L	1.00	0.25	0.50
M	2.00	0.25	0.50

4.2. Stochastic line balancing sensitivity

It was assumed before that all task times are deterministic but it is possible that each task can have a small amount of variation due to the manual nature of the work. We performed additional line balances and simulations for the original laptop example except now each non-zero task time is a normal random variable with a mean that is equal to the previously used task times and has a standard deviation of 0.5 tu and 1.0 tu. We used a value of $\Sigma = 14\ \text{tu}^2$ for the total variance in constraint (12) for the stochastic line balancing formulation in Section 3.2. Tables 8 and 9 show the mean, variance, and standard deviation for each task using Eqs. (3) and (4) with a task standard deviation of 0.5 and 1.0 tu, respectively, and the simulation results for the line balances are shown in Table 10. The line balancing task assignment in Table 7

Table 9
Data input for the stochastic line balancing formulation, 1.0 tu standard deviation.

Task	Mean time	Variance	Standard deviation
A	2.80	4.06	2.01
B	2.80	4.06	2.01
C	2.10	2.59	1.61
D	5.00	1.00	1.00
E	6.05	3.05	1.75
F	3.00	1.00	1.00
G	1.40	1.54	1.24
H	2.00	1.00	1.00
I	3.90	8.84	2.97
J	5.00	1.00	1.00
K	3.00	1.00	1.00
L	1.00	1.00	1.00
M	2.00	1.00	1.00

Table 10
Results of laptop line balancing for random variable task times.

Component standard deviation	EOL state	Probability	Line balance				Average throughput	Percent difference
			Wkst 1	Wkst 2	Wkst 3	Wkst 4		
0.5	Baseline	20.70%	GAD	CBLK	JI	FMEH	4.572	6.76
	Joint	–	GADL	CBJ	KFI	MEH	4.881	
1.0	Baseline	20.70%	GAD	CBLK	JI	FMEH	4.479	4.98
	Joint	–	GADL	CBJ	KFI	MEH	4.702	

Table 11
Results of laptop line balancing for only *E* as random variable.

Component standard deviation	EOL state	Probability	Line Balance				Average throughput	Percent difference
			Wkst 1	Wkst 2	Wkst 3	Wkst 4		
E = 1.0 only	Baseline	20.70%	GAD	CBLK	JI	FMEH	4.587	9.16
	Joint	–	GADL	CBJ	KFI	MEH	5.007	

is the same as in Table 4 from the deterministic case because the total workstation variance constraint did not require tasks to be reassigned to different workstations.

We also experimented with only 1 task being a normal variable, E (system board) in this case, with a standard deviation of 1.0 to see if the same results would occur as the case with all non-zero task times being normal random variables. The results from this set of experimentations are shown in Table 11.

5. Discussion

The results in Table 5 show that when creating and simulating a serial disassembly line with 4 workstations, a line balance utilizing the joint precedence graph has an average increase in throughput of 9.72% versus a line balance using the baseline precedence graph. The percent different in each table always compares the increase in throughput the joint line balance has when compared to the baseline line balance. Table 6 shows that the joint precedence graph line balance has an overall higher average workstation working percent, 84.23% versus 307.40%. It is interesting to note that for the baseline line balancing case, a single workstation (in this case workstation 4) will be the bottleneck for the disassembly line with a working percent of 100%. The workload for the joint line balance is more balanced with a working percent range of 83.11–76.85%. This conclusion was fairly constant for the line balancing results in the sensitivity study, where the joint line balance had a relatively small working percent range and a single workstation in the baseline line balance is the bottleneck.

The actual task assignment between the baseline and joint line balances have some identical task assignments with a few different task assignments that creates the more balanced work load for the joint line balance. For both the joint and baseline line balances, workstation 1 is assigned tasks G, A, and D; workstation 2 is assigned C and B; workstation 3 is assigned I; and workstation 4 is assigned M, E, and H. The difference in task assignments are workstation 1 for the joint case is also assigned task L while the baseline balance is not; workstation 2 for the joint case is assigned task J and workstation 2 for the joint case is assigned L and K, but not J; workstation 3 for the joint case is assigned K and F while the baseline balance for workstation 3 is assigned task J, but not tasks K and F; and workstation 4 for the baseline case is assigned F and the joint case is not. Due to the work balancing for the joint line balance, each workstation shifts between being the pacing workstation of the line while the baseline line balance is completely paced by the final workstation.

5.1. Line balancing sensitivity discussion

The line balancing sensitivity results in Table 7 are fairly expected. When moving from all EOL states having equal probability of occurring (50%) to diminishing percent of occurrence, the joint line balance decreases in throughput advantage versus the baseline case. The only result that does not match with other results is if every EOL state has a 20% chance of occurring. When each EOL state is 30%, the difference in throughput is 7.31%, for 20% case it drops to 2.29%, and 10% case it rises to 3.03%. When investigating why this result happens, the task assignment for workstation 2 for the joint line balance (tasks A, C, and I) has a 10.5% chance of resulting in a 0 total workstation time due to the task assignment. This result is shown in Table 12 and the EOL states that result in a total workstation time of zero is highlighted in gray.

This observation is typical for disassembly where some tasks should be ignored depending on the EOL state. Therefore, the following important constraint should be added to the model (5a)–(11a) to prevent such situations:

$$\sum_{j \in O} X_{jk} < \sum_{i \in I} X_{ik} \quad \forall k \in K, \quad \forall O \in \mathbf{O} \tag{13}$$

where **O** is a family of such sets *O* that *O* contains all tasks having 0-time for one coded EOL state. After the inclusion of this constraint, task J from workstation 3 was switched with task I to prevent the zero work time from happening. When simulating this new line balance, the average percent difference in throughput was 4.38% in favor of the joint line balance.

Table 12
Workstation #2 for 20% EOL state.

Attribute	Probability	A	C	I	Total work time
1	0.2070	4	3	6	13
2	0.0887	0	0	6	6
3	0.1115	4	3	6	13
4	0.0478	0	0	6	6
5	0.0887	4	3	6	13
6	0.0380	0	0	6	6
7	0.0478	4	3	6	13
8	0.0205	0	0	6	6
9	0.1115	4	3	0	7
10	0.0478	0	0	0	0
11	0.0600	4	3	0	7
12	0.0257	0	0	0	0
13	0.0478	4	3	0	7
14	0.0205	0	0	0	0
15	0.0257	4	3	0	7
16	0.0110	0	0	0	0

Table 13
Workstation utilization results for stochastic line balance, St dev = 0.5.

Baseline case (St dev = 0.5)			Joint case (St dev = 0.5)		
Workstation	Category	Percent	Workstation	Category	Percent
Wkst 1	Waiting (%)	0.00	Wkst 1	Waiting (%)	0.00
	Working (%)	70.01		Working (%)	83.02
	Blocked (%)	29.99		Blocked (%)	16.98
Wkst 2	Waiting (%)	0.30	Wkst 2	Waiting (%)	3.22
	Working (%)	67.46		Working (%)	80.50
	Blocked (%)	32.25		Blocked (%)	16.28
Wkst 3	Waiting (%)	0.68	Wkst 3	Waiting (%)	10.25
	Working (%)	67.86		Working (%)	80.64
	Blocked (%)	31.45		Blocked (%)	9.12
Wkst 4	Waiting (%)	0.59	Wkst 4	Waiting (%)	18.39
	Working (%)	99.41		Working (%)	81.61
	Blocked (%)	0.00		Blocked (%)	0.00
Average working percent		76.18	Average working percent		81.44

Table 14
Workstation utilization results for stochastic line balance, St dev = 1.0.

Baseline case (St dev = 1.0)			Joint case (St dev = 1.0)		
Workstation	Category	Percent	Workstation	Category	Percent
Wkst 1	Waiting (%)	0.00	Wkst 1	Waiting (%)	0.00
	Working (%)	68.46		Working (%)	79.98
	Blocked (%)	31.55		Blocked (%)	20.02
Wkst 2	Waiting (%)	1.11	Wkst 2	Waiting (%)	4.80
	Working (%)	66.16		Working (%)	77.51
	Blocked (%)	32.73		Blocked (%)	17.69
Wkst 3	Waiting (%)	2.89	Wkst 3	Waiting (%)	12.97
	Working (%)	66.47		Working (%)	77.15
	Blocked (%)	30.64		Blocked (%)	9.88
Wkst 4	Waiting (%)	2.69	Wkst 4	Waiting (%)	21.28
	Working (%)	97.31		Working (%)	78.72
	Blocked (%)	0.00		Blocked (%)	0.00
Average working percent		74.60	Average working percent		78.34

5.2. Stochastic line balancing sensitivity discussion

When comparing the task assignments in Tables 5 and 10, the baseline line balances are the same in each and the joint line balances are also the same. Table 5 contains results where each task time is deterministic, while Table 10 has task times that are normally distributed with a mean equal to the deterministic times and standard deviations of 0.5 and 1.0 tu, respectively. The percent difference in throughput between the joint versus baseline line balances decreases as the standard deviation increase from 0 (the deterministic case) to 0.5 and 1.0 tu. These results are shown in Fig. 9. These results are intuitive because if the total workstation

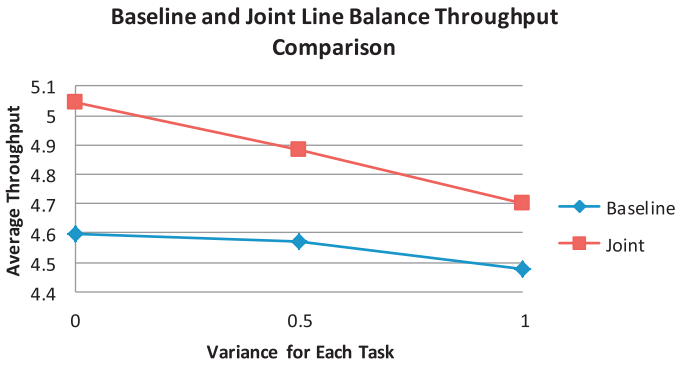


Fig. 9. Comparison of baseline and joint line balances average throughput.

times are continuous and the variation of workstation time can vary with a higher probability as the standard deviation increases, the interaction of workstations along the disassembly line that can cause waiting and blocking have an infinite amount of possibilities. This should, as expected, lead to a diminished average throughput. For the baseline case, the throughput decreases from 4.598 to 4.572 to 4.479 as the standard deviation increases, and the joint case throughput decreases from 5.045 to 4.881 to 4.702. What was not originally expected is how the decreases in average throughput are more drastic for the joint case versus the baseline case. The average throughput difference between each comparison decreases from 9.72% to 6.76% to 4.98%. The reason for this disparity is the baseline line balances have a single pacing workstation versus the joint line balances have pacing workstations that change throughout the simulation.

Tables 13 and 14 show how the workstation utilization changes for task time with different standard deviations. Because the baseline line balances have a single pacing workstation, this workstation stays working for close to 100% of the time for all three standard deviation cases (0.0, 0.5, and 1.0 tu) while the workstation working utilization for the joint cases decreases steadily. The range of working percent stays relatively small for each of the joint cases, but the average workstation working percentage decreases as the standard deviation values increase.

The situation where every task is a random variable with a common variance is unlikely, but it is very possible that at least one task time is a random variable. Table 11 in Section 4.2 shows the results for having task E have a standard deviation of 1.0 tu and the

throughput results do not change very much when compared to having all task times be deterministic.

6. Conclusions

This paper considers the variation of components end state and adapts an efficient method from mixed model assembly to plan the disassembly process of such products. Disassembly of products at their EOL and efficient disassembly systems are important topics and a method that is able to consider the full spectrum of EOL conditions and treatment possibilities can be a powerful tool for disassembly decision makers. The goal of this paper is to develop and validate a disassembly joint precedence graph creation method that is able to simultaneously consider all possible EOL conditions and states a product can be in. We also wanted to extend our approach to consider the possibility of stochastic task times and develop a method for generating a stochastic disassembly joint precedence graph.

The achievements of our proposed method is: (1) a method that can handle, through preprocessing, many different EOL states and treatment options for components in the same product into a single precedence graph, and (2) a method that can work with any line balancing algorithm where a single precedence graph is used. The impact our method has on disassembly line efficiency and throughput when using a joint precedence graph versus a precedence graph for only one EOL state is highly dependent on the percentage of EOL states, different treatment options, and task times, both deterministic and stochastic. However, any gain in efficiency for disassembly lines is important, no matter how small. Future work can focus on methods to mitigate the decrease in throughput percent difference when task times are continuous. Since disassembly is mostly manual operations, it makes sense that task times will have some sort of variance associated with them. Additional future work should focus on the possibility that the EOL condition and state probabilities can change over time and how can such changes be handled with respect to reconfiguration of the disassembly line.

References

[1] Ozdemir O, Denizel M, Guide VDR. Recovery decisions of a producer in a legislative disposal fee environment. Eur J Oper Res 2012;216:293–300.
[2] McGovern SM, Gupta SM. The disassembly line: balancing and modeling. New York: McGraw Hill; 2011.
[3] Battaïa O, Dolgui A. A taxonomy of line balancing problems and their solution approaches. Int J Prod Econ 2013;142:259–77.
[4] Meacham A, Uzsoy R, Venkatadri U. Optimal disassembly configurations for single and multiple products. J Manuf Syst 1999;18(5):311–22.
[5] Gungor A, Gupta SM. A solution approach to the disassembly line balancing problem in the presence of task failures. Int J Prod Res 2001;39:1427–67.
[6] Gungor A, Gupta SM, Pochampally K, Kamarthi SV. Complications in disassembly line balancing. Proc SPIE 2001;4193:289–98.
[7] Altekin FT, Kandiller L, Ozdemirel NE. Disassembly line balancing with limited supply and subassembly availability. Proc SPIE 2004;5262:59–70.
[8] Tang Y, Zhou M. A systematic approach to design and operation of disassembly lines. IEEE Trans Autom Sci Eng 2006;3(3):324–30.
[9] McGovern SM, Gupta SM. A balancing method and genetic algorithm for disassembly line balancing. Eur J Oper Res 2007;179:692–708.
[10] Altekin FT, Kandiller L, Ozdemirel NE. Profit-oriented disassembly-line balancing. Int J Prod Res 2008;46(10):2675–93.
[11] Tripathi M, Agrawal S, Pandey MK, Shankar R, Tiwari MK. Real world disassembly modeling and sequencing problem: Optimization by Algorithm of Self-Guided Ants (ASGA). Robot Comput Integr Manuf 2009;25:483–96.
[12] Koc A, Sabuncuoglu I, Erel E. Two exact formulations for disassembly line balancing problems with task precedence diagram construction using an AND/OR graph. IIE Trans 2009;41:866–81.
[13] Altekin FT, Akkan C. Task-failure-driven rebalancing of disassembly lines. Int J Prod Res 2012;50(18):4955–76.
[14] Ma YS, Jun HB, Kim HW, Lee DH. Disassembly process planning algorithms for end-of-life product recovery and environmentally conscious disposal. Int J Prod Res 2011;49(23):7007–27.
[15] Rickli JL, Cameilio JA. Multi-objective partial disassembly optimization based on sequence feasibility. J Manuf Syst 2013;32(1):281–93.
[16] Ilgin MA, Gupta SM. Environmentally conscious manufacturing and product recovery (ECMPRO): a review of the state of the art. J Environ Manage 2010;91(3):563–91.
[17] Tuncel E, Zeid A, Kamarthi S. Solving large scale disassembly line balancing problem with uncertainty using reinforcement learning. J Intell Manufact 2012;25(4):647–59.
[18] Lambert AJD. Disassembly sequencing: a survey. Int J Prod Res 2003;41(16):3721–59.
[19] Kang MK, Kwak MJ, Cho NW, Hong YS. Automatic derivation of transition matrix for end-of-life decision making. Int J Prod Res 2010;48(11):3269–98.
[20] Aydemir-Karadag A, Turkbey O. Multi-objective optimization of stochastic disassembly line balancing with station paralleling. Comput Ind Eng 2013;65:413–25.
[21] Agrawal S, Tiwari MK. A collaborative ant colony algorithm to stochastic mixed-model U-shaped disassembly line balancing and sequencing problem. Int J Prod Res 2008;46(6):1405–29.
[22] Tang Y, Zhou M, Zussman E, Caudill R. Disassembly modeling: planning, and application. J Manuf Syst 2002;21(3):200–17.
[23] Boysen N, Fliedner M, Scholl A. Assembly line balancing: joint precedence graphs under high product variety. IIE Trans 2009;41(3):183–93.
[24] De Fazio TL, Whitney DE. Simplified generation of all mechanical assembly sequences. IEEE J Robot Autom 1987;3(6):640–58.
[25] Baybars I. A survey of exact algorithms for the simple assembly line balancing problem. Manage Sci 1986;32(8):909–32.
[26] Hu SJ, Ko J, Weyand L, ElMaraghy HA, Lien TK, Koren Y, et al. Assembly system design and operations for product variety. CIRP Ann Manuf Technol 2011;60(2):715–33.
[27] Riggs RJ, Hu SJ. Disassembly liaison graphs inspired by word clouds. Procedia CIRP 2013;7:521–6.
[28] Kalayci CB, Gupta SM. Artificial bee colony algorithm for solving sequence-dependent disassembly line balancing problem. Expert Syst Appl 2013;40(18):7231–41.
[29] Riggs RJ, Jin X, Hu SJ. Two-stage sequence generation for partial disassembly of products with sequence dependent task times. Procedia CIRP 2015;29:698–703.
[30] Bentaha ML, Battaïa O, Dolgui A, Hu SJ. Dealing with uncertainty in disassembly line design. CIRP Ann Manuf Technol 2014;63(1):21–4.
[31] Bentaha ML, Battaïa O, Dolgui A. A sample average approximation method for disassembly line balancing problem under uncertainty. Comput Oper Res 2014;51:111–22.
[32] Bentaha ML, Battaïa O, Dolgui A. An exact solution approach for disassembly line balancing problem under uncertainty of the task processing times. Int J Prod Res 2015;53(6):1807–18.