MODELING AND OPTIMIZATION OF DISASSEMBLY SYSTEMS WITH A HIGH VARIETY OF END OF LIFE STATES

by

Robert J. Riggs

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Industrial and Operations Engineering)
in the University of Michigan
2015

Doctorial Committee:

Professor S. Jack Hu, Chair
Assistant Professor Eunshin Byon
Professor Jun Ni
Professor Mark Van Oyen

DEDICATION

To my Wife and Family

ACKNOWLEDGMENTS

I would first like to thank my advisor, Professor S. Jack Hu, for accepting me into his lab and working with me these past five plus years. He has been so incredibly patient and understanding with me through some very rough health problems in the final year of my thesis and I could not have asked for a better advisor and mentor.

I would also like to thank all my committee members for all their help and insight. Thank you Assistant Professor Eunshin Byon, Professor Jun Ni, and Professor Mark Van Oyen, for serving on my committee and taking time out of your busy schedules to allow me to present my research to you and for giving me such valuable feedback.

I would like to thank my family for always encouraging me to continue on and finish even when times were tough in graduate school.

Lastly, to the most important person of all, I would like to acknowledge and thank my wife, Sara Lu Riggs, for being by my side from the start of graduate school, first as a friend, and then as my wife. You are a blessing unto my life and you make every day worth living to the fullest. You are my best friend, my rock, my soul mate, and I look forward to all the years we will have together at Clemson University and beyond. I love you!

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Remanufacturing is a promising product recovery method that brings new life to cores that otherwise would be discarded thus losing all value. Disassembly is a sub-process of remanufacturing where components and modules are removed from the core, sorted and graded, and directly reused, refurbished, recycled, or disposed of. Disassembly is the backbone of the remanufacturing process because this is where the reuse value of components and modules is realized. Disassembly is a process that is also very difficult in most instances because it is a mostly manual process creating stochastic removal times of components. There is a high variety of EOL states a core can be in when disassembled and an economic downside due to not all components having reuse potential. This thesis focuses on addressing these difficulties of disassembly in the areas of sequence generation, line balancing, and throughput modeling.

In Chapter 2, we develop a series of sequence generation models that considers the material properties, partial disassembly, and sequence dependent task times to determine the optimal order of disassembly in the presence of a high variety of EOL states. In Chapter 3, we develop a joint precedence graph method for disassembly that models all possible EOL states a core can be in that can be used with a wide variety of line balancing algorithms. We also develop a stochastic joint precedence graph method in the situation where some removal times of components are normal random variables. In Chapter 4, we further advance the analytical

modeling framework to analyze transfer lines that perform routing logics that result from a high variety of EOL states, such as a restrictive split routing logic and the possibility that disassembly and split operations can be performed at the same workstation.

CHAPTER 1

INTRODUCTION

1.1 Motivation

The end of life (EOL) treatment of consumer and industrial products is an important challenge as well as an opportunity that societies around the globe face each and every day. The challenge is finding the best way to properly dispose of a wide variety of products and the opportunity is the potential benefits that can achieved, both economically and environmentally, from responsible and sustainable reuse, recovery, and disposal. The EPA estimated that Americans generated about 251 million tons of trash in 2012, and that 75% of this solid waste was re-usable or recyclable. This solid waste fills up landfills every year and precious resources are no longer usable. However, it is estimated that there are over 73,000 firms (Lund, 1996) that undergo product recovery in the United States alone with a total annual sales volume of $53 billion, so there is at least a start to the solution with room to grow.

Remanufacturing, which many of these United States based recovery firms practice, is one such promising recovery process than enables the extraction of value from products that have been deemed ready for disposal. Remanufacturing is a multi-step process consisting of (1) acquiring products or cores for EOL treatment, (2) sorting and sometimes cleaning of the cores,

(3) disassembly of the cores into components or modules, and (4) assessing the reuse potential of each component and module (Lund, 1984). Some components and modules can be directly reused, although they might require some cleaning and minor repairs, while other components and modules might need to be refurbished and repaired to be reused, and still others recycled or disposed of. The process of remanufacturing is to make recovered components and modules in a like new condition when they are reused. The remanufacturing process is rooted in the product life cycle shown in Figure 1-1. Remanufacturing occurs at the end-of-life (EOL) treatment portion of the cycle, where components and modules can be re-used, recovered, recycled, or disposed of.

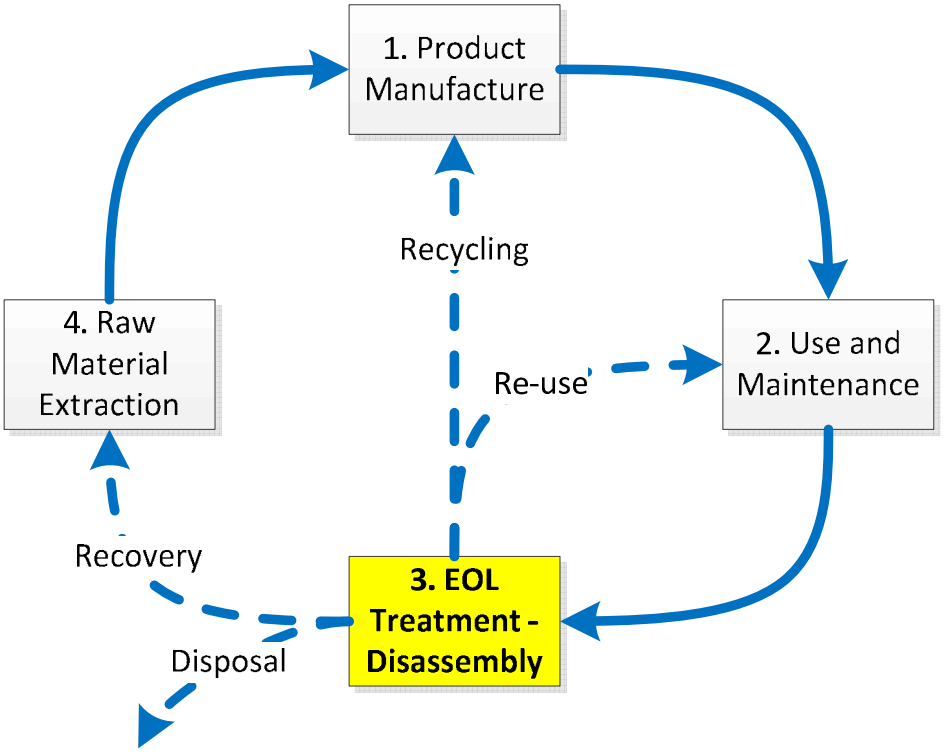


Figure 1-1 Product life cycle. Source: adapted from the Australian and New Zealand Government Framework for Sustainable Procurement, 2007

Disassembly is the backbone of the entire remanufacturing process, because this is where the actual real EOL value of components and modules is achieved. When the core is not disassembled, there is potential for reuse value but additional work is required to extract that value. In a way, disassembly adds value to a core; but there are many challenges that come with the disassembly process. These challenges include, but are not limited to: (1) the variability of the processing parameters for a disassembly line due to the quality variation in core returns; (2) process and system efficiency issues due to the nature of the type of work disassembly, typically manual; and (3) the economic issues when compared to (forward) assembly since not all components and modules have reuse potential.

Disassembly is typically a very manual type of work. The work can be done individually or in a line format. When the work is done manually, there is great variability in the task times to disassemble a product. Balancing a line is very straightforward when the task times are deterministic, but when task times are a stochastic, line balancing is more difficult. In addition, workers are prone to making errors when everything is done manually. For an automated assembly line, the task time is practically identical with each cycle of assembly. For complicated disassembly, each process cycle can be quite different from the previous and operator errors are more likely to occur.

If all cores were returned in a "like new" state for disassembly, the remanufacturing process would be very efficient and very profitable; however, this is not the case for obvious reasons. The quality of returned cores can be in a number of overall states. For example, a core could have components that are:

(a) missing (or not),

(b) damaged (or not damaged) which will affect the time for disassembly of that component, or

(c) damaged so it has no reuse value (or not damaged so it has reuse value).

Each of these is an example of an EOL condition for one component. Since a core is made up of many different components, the combination of all of these EOL conditions would constitute many EOL states. An EOL state is made up of a combination of different EOL conditions for each of the components. Given the possibility of so many EOL states, it is difficult to plan a disassembly line that is both efficient and predictable.

There are four major differences when comparing disassembly lines to assembly lines.

1. Disassembly involves mostly manual operations and any form of automation is very difficult outside of shredding and very automatic destructive processes. Automation is very prevalent for assembly where many assembly processes are mostly automated with very few manual operations.
2. A new product off the assembly line has some retail value. In order for that product to have that full retail value, all components need to be present in that assembly and each of those components needs to be in the correct working condition. For cores at disassembly, only certain components and modules have reuse value, and very typically those values are not even as high as a brand new component unless the reuse condition is like new. This is a major drawback of disassembly.
3. At each step of assembly, value is being added to the assembled product. For disassembly, time is being spent removing certain components that have no real value in order to extract components and modules that do have value.

4. Another less thought about problem that can occur during disassembly is a phenomenon called disassembly damage (Gungor et al., 1998). The disassembly process can cause damage to components that had reuse value. The damage can occur due to the harshness of the disassembly process (Jovane et al., 1993; Bras et al., 1996) or due to operator error (Williams et al., 2001).

These four differences to disassembly system design are what motivate the research in this thesis. It is critical that disassembly be as efficient as possible in order for it to be a sustainable practice. In many places in the world, cores are required by law to be taken back by the original manufacturers and a certain weight percentage of the core must be reused or recycled (Errington et al., 2013). An example of this is for certain products, like automobiles and electronics, in the European Union. In order to enable disassembly to be as efficient and sustainable as possible, mathematical methods are needed to determine the optimal disassembly sequence, line balance, configuration, and continuous throughput monitoring that take into consideration the high variety of EOL states a core can be in.

This thesis will focus on disassembly system design issues in the areas of sequence generation, line balancing, and throughput performance evaluation.

1.2 State of the Art, Literature Review

The relevant literature in the areas of (1) disassembly sequence generation, (2) line balancing; and (3) throughput modeling are reviewed. In the literature, the three main topics of this thesis are usually treated as completely separate research areas even though they are closely related and at times build upon each other. The reason for focusing on these three areas versus

other possible research areas is because these three research areas were viewed as having the greatest impact on the overall goal of this thesis, i.e., to develop methods and algorithms for modeling, balancing, and analyzing disassembly lines that enable disassembly to be as efficient as possible. To develop a "best practice" for establishing a sequence for disassembly tasks and to assigning the tasks to workstations in order to enable the disassembly line to be as efficient and cost effective as possible is an important and impactful goal.

The inter-relationship among these three areas is demonstrated using Figure 1-2, which is inspired from a survey paper by Lambert (2003). Each of these four levels is highly related yet requires very different decisions.

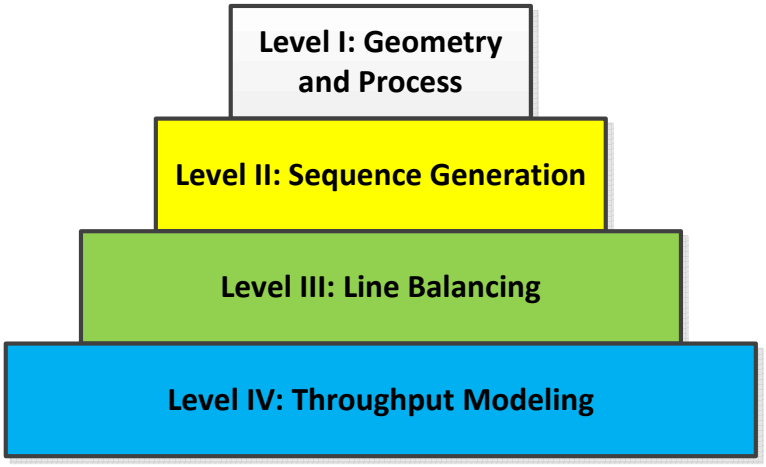


Figure 1-2 The adjacent levels of aggregation for disassembly modeling, inspired by (Lambert, 2003)

- At level I, the geometry and process level, component geometry can be recaptured.
- At level II, the sequence generation level, the optimal disassembly order can be determined based on some criteria.

- At level III, the line balancing level, an optimal disassembly line balance and line configuration are determined. Level III relates to level II because the order of tasks in the line balance with a certain configuration is a disassembly sequence, but now the user gives credence to the decision that each workstation has a balanced workload.
- At level IV, the throughput modeling level, the line balance and configuration from level III is analyzed on a deeper level, referred to as throughput modeling in this case. At this level, workstations may have a probability of failure and repair based on real life scenarios and the throughput and other measures such as work in process can be analyzed and determined very quickly.

The levels in the pyramid are related but contain very different decisions. For example, at level I the component geometry can be optimized to allow for easier (dis)assembly, while at level II the component geometry is likely fixed and the decision will be to find an optimal disassembly sequence. The optimal disassembly sequence from level II can be assigned to a series of workstations, but it may not be the optimal sequence to balance because at level III the optimal sequence is chosen with respect to balancing the workload at each workstation. At level IV, the chosen line balance is analyzed at a deeper level where workstations can be in failure, repair, blocked, or starved modes. At level I, the decisions are solely focused on the core while as the level number increases, the decisions are focused on the system level.

1.2.1 Disassembly Sequence Generation

Research on disassembly sequence generation puts an emphasis on the "design and optimization of disassembly lines" and "optimum product design regarding the product's end-of-life phase" (Lambert, 2003). The first application for sequence generation was assembly and it was later applied to disassembly. Many of the sequence generation methods and algorithms for assembly are very similar for disassembly; however, there will be noticeable differences between the two. For example, assembly is always complete while disassembly might selective or partial instead of complete.

Since disassembly and assembly sequence generation are closely related and one begot the other, when reviewing the literature for disassembly sequence generation there are many key pieces of research in assembly sequence research that are relevant. Many of the approaches for assembly sequence generation will be used for disassembly sequence generation, just with some changes made to reflect the differences between the two processes. The determination of all possible assembly sequences is an important and critical stage in the total design process of a product. One of the pioneers for the automatic generation of all assembly sequences algorithmically is Bourjault. Bourjault (1984) whose early work used a series of rules that are determined by a series of "yes" or "no" questions that are answered about the mating of components for an assembly. Bourjault represented a product by using the information contained in a part list and an assembly drawing to form a liaison graph, where the components are the nodes and the liaisons are the arc representing the determined mates. All assembly sequences are determined algorithmically using the liaison graphs.

De Fazio and Whitney (1987) extended Bourjault's work by simplifying the determination of the set of rules, or precedence constraints, by asking two specific questions

about liaison precedence. Questions are specifically asked about "what liaisons must be done prior to doing liaison i" and "what liaisons must be left to be done after doing liaison i". De Fazio and Whitney's work significantly reduced the question count for determining all possible assembly sequences. Their later work with Baldwin et al. (1991) took advantage of using a computer as aid for automatic assembly sequence generation. Other work of computer aided assembly sequence generation is that of Khosla and Mattikali (1989). They developed a methodology that uses software to automatically determine the assembly sequence from a 3D model of the assembly. Further, Kanai, Takahashi, and Makino (1996) developed a computer aided Assembly Sequence Planning and Evaluation system (ASPEN) that takes all the solid-model components of a product and automatically determines all feasible sequences by decomposition. ASPEN also determines the optimum sequence using Methods Time Measurement (MTM) as time standards for operating time determination. Choi and Zha (1998) also conducted further research for computer aided automatic assembly sequence generation with their work on automated sequence planning. They use an And/Or graph and the identification of leveled feasible subassemblies to determine the assembly sequence.

Built upon previous research, de Mello and Sanderson (1991) treated an assembly sequence generation problem as a disassembly sequence generation problem. The problem is then further decomposed into sub-problems where subassemblies are joined one at a time. Gupta and Krishnan (1998) created a methodology to determine the largest subassembly in an assembly problem for a product family where some components differ. They used De Fazio and Whitney's algorithm for finding all assembly sequences and implemented their searching algorithm to find the maximum generic subassembly.

Dini, Failli, Lazzerini, and Marcelloni (1999) made use of genetic algorithm to create and evaluate assembly sequences. They created a fitness function which takes into account geometrical constraints of the assembly and other optimization aspects and using their genetic algorithm decreased the time for computation. Marian, Luong, and Abhary (2003) also looked at optimizing the assembly sequence planning problem by using genetic algorithms. Wang and Ceglarek (2007) made use of graph theory by developing a methodology that generates all the sequences for a k-ary assembly process. The authors used a k-piece graph to represent assemblies without precedence constraints and a k-piece mixed-graph to represent assemblies with precedence information. Using this approach, all feasible subassemblies can be identified, and all of the assembly sequences for a k-ary assembly process are generated iteratively.

Focusing solely on disassembly sequence generation, Lambert (1997) presented an AND/OR graphical representation to aid in generating the optimal disassembly sequence. Lambert's method also takes into account selective disassembly, where total disassembly is not required. Kuo (2000) provided the disassembly sequence and cost analysis for electromechanical products during the design stage. The disassembly planning is divided into four stages, starting with geometric assembly representation, next cut-vertex search analysis, followed by disassembly precedence matrix analysis, and ending with disassembly sequences and plan generation.

Torres, Puente, and Aracil (2003) developed an algorithm that can compute a disassembly sequence of a product automatically, based only on the information introduced by an operator who knows the product very well. The algorithm creates a representation of the product to establish a partial, non-destructive disassembly sequence. Hula et al. (2003) presented a methodology to find the Pareto set of optimal end of life strategies using a multi-objective

genetic algorithm. The algorithm found the Pareto set of the tradeoff between cost and environmentally conscious actions and was able to include different scenarios, such as cost of labor, government regulation, etc., for comparison. Capelli et al. (2007) developed two algorithms. The first analyzes the physical constraints of a product from its 3D CAD representation and creates all possible disassembly solutions based on an AND/OR graph. The second algorithm explores these solutions and selects the optimum. Lambert (2003) presented a disassembly sequencing survey paper that discusses all relevant research ranging from liaison graph representation of disassemblies and precedence graphs. Dong and Arndt (2003) also presented a survey paper reviewing the state of the art for disassembly sequence generation and computer aided approaches.

Li, Khoo, and Tor (2006) developed a representation scheme known as a disassembly constraint graph for disassembly sequencing for maintenance. The approach can be used to generate possible disassembly sequences for maintaining machines, etc. Kara, Pornprasitpol, and Kaebernick (2006) created a methodology for selective disassembly from the methodology created by Nevins and Whitney for assembly sequence. The new methodology for disassembly sequence generation and selective part recovery is performed by computer software.

Dong, Gibson, and Arndt (2007) used Petri nets to generate an optimal disassembly based on accessibility and end of life strategy. Their approach used AND/OR graphs for representation to determine all possible disassembly sequences, and then the AND/OR graphs are transferred into Petri net graphs and the accessibility and life span values of components are taken into account to obtain the optimal disassembly sequence. Andres, Lozano, and Adenso (2007) used AND/OR graphs to model precedence relations for disassembly in a cell based layout. Tripathi et

al. (2009) used a self-guided ants optimization algorithm to find the optimal disassembly sequence and takes into account the quality of the products returned to be disassembled.

1.2.2 Disassembly Line Balancing

Similar to assembly and disassembly sequence generation, assembly line balancing preceded disassembly line balancing and naturally some differences exist between the two. Some examples of the differences between the two are as follows: disassembly is not always the reverse of assembly, the quality of components is only required for disassembly and not assembly, etc. But through the years, the optimization approaches for both assembly and disassembly line balancing have been very similar. The line balancing field in general is quite mature, dating back to 1955 with Salveson's paper on assembly line balancing (Salveson, 1955). The author assigned tasks to a workstation by developing a cycle time constraint with the objective to assign all tasks to a workstation with respect to precedence and cycle time constraints while minimizing the number of workstations. This problem was later referred to as the simple assembly line problem (Baybars, 1986).

The development of the disassembly line balancing problem has a similar history. There have been two very important survey papers in the area of disassembly line balancing, one by Boysen et al. (2008) and the other by Battaia et al. (2013). Boysen et al. reviewed over 150 papers from between 1966 and 2006, while the paper by Battaia et al. reviewed 267 papers since the time the Boysen paper was published, showing the explosion of interest in disassembly line balancing in recent years. In general, the disassembly line balancing problem can have many variations in how the problem is solved and the type of solution. Solution methods, such as exact methods or approximate methods, i.e. bounded exact methods, simple heuristics, or meta-

heuristics, can all be employed to solve the optimization problem. These solution methods can solve problems that seek to find the optimal configuration, such as a single disassembly line, a U-shaped line, or even a parallel or hybrid organized line. In addition, the attributes for task times can be constant, dynamic, stochastic, or even assignment-dependent. Workstations can have their own attributes as well, and certain formulations may only consider some problem constraints, such as cycle time or performance measure constraints to name a couple. And lastly, there are many variations of objective functions employed using the wide range of solution methods, such as minimize the number of workstations, minimize the line cycle time, or maximize line efficiency, just to name a few. Given the large amount of combinations possible, solution methods, problem attributes, objective function, etc., it is very easy to understand why there are so many recent disassembly line balancing papers published.

1.2.3 Disassembly Throughput Modeling

Throughput analysis is important in the "design, operation, and management of production systems" because it enables users to quickly analyze simple to complex transfer lines (Li et al., 2009). The research area of throughput modeling is very broad and the possibility of analyzing a wide range of problem types is possible using the same methods, as shown in Figure 1-3. Production systems can have reliable or unreliable machines, finite or infinite buffers, constant or random processing times, and the lines can be homogeneous or inhomogeneous.

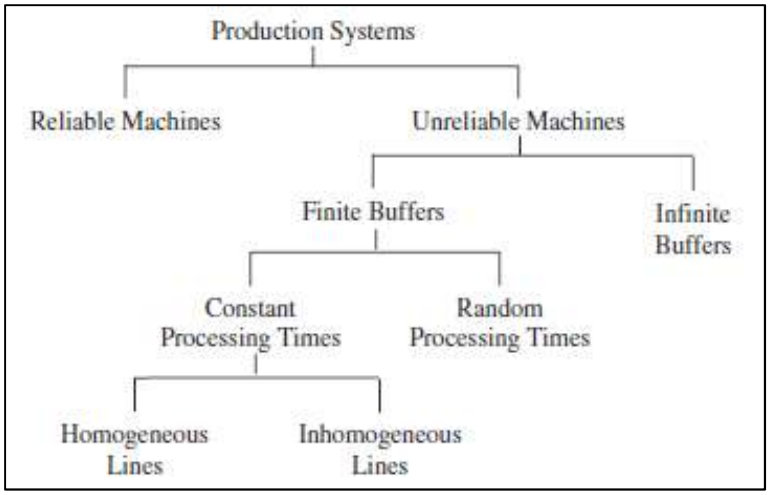


Figure 1-3 Categorizations of production systems with throughput analysis,

Source: Li et al., 2009

There are also two types of analytical methods for analyzing transfer lines, the "Meerkov aggregation method and the Gershwin decomposition method" (Li et al., 2009). The Meerkov aggregation method consists of backward and forward aggregation. For forward aggregation, the first machine-buffer-machine building block at the start of the line is aggregated into a single machine, and this process continues until all machines and buffers are aggregated into a single machine to be analyzed. Backward aggregation is the same except aggregation starts at the end of the transfer line (Li and Meerkov, 2003, 2007). The main advantage of Meerkov aggregation is that the aggregation procedure can be analytically proven to be convergent (Li et al., 2009). The Gershwin decomposition method is the opposite of the aggregation method because the entire transfer line is broken down into individual building blocks for analysis. The machine at the start of each building block become a pseudo machine that uses failure and repair probabilities to mimic the flow dynamics of what occurs upstream from the building block. The machine at the end of each building block also becomes a pseudo machine and mimics the flow

dynamics of what occurs downstream of the building block (Gershwin, 1994). To fully analyze the transfer line that is decomposed into a series of building blocks, an algorithm is used, called the Dallery-David-Xie (DDX) algorithm (Dallery et al., 1988 and 1989) that traverses up and down the decomposed transfer line to determine the production rate of the line, under the assumption of the conservation of flow. When the production rate of the line from one iteration to the next is smaller than some alpha value, the algorithm stops and the production rate is found along with the total work in processes (WIP) of the line.

In Chapter 4, the Gershwin decomposition method is used, along with a modified version of the DDX algorithm using a discrete time Markov chain framework; therefore, the papers summarized in this sub-section will focus on similar solution approaches. Tolio et al. (2002) presented an analytical method for determining the performance of production lines with finite buffers and two unreliable machines. This paper contributed to the throughput analysis literature by allowing each machine to have multiple failure modes instead of one single failure mode. Colledani et al. (2005) developed an approximate analytical method where buffers are finite, multiple failure modes exist, but instead of the line consisting of two machines and one buffer, the transfer line can contain many machines and buffers and can process multiple part types. This research contributed to analyzing split routing systems using the Gershwin decomposition method and created what is known as the competition failure. This work was extended in Colledani et al. (2008) where Z different types of products can be produced. The authors also used an aggregation technique where the behavior of several products is modeled by an aggregate product. There are other analytical approaches to analyze split systems. Liu et al. (2010) developed analytical methods to analyze split and merge systems with different routing policies, such as circulate and priority policies.

1.3 Research Objective

The overall research objective of this thesis is to develop methods and algorithms for the design, analysis and optimization of disassembly systems under the presence of a high variety of end of life (EOL) states.

The specific research tasks are proposed below:

1. Develop models for disassembly sequence generation that considers material properties, partial disassembly, and sequence dependent task times that will determine the optimal order of disassembly in the presence of a high variety of EOL states.
2. Design a method for disassembly line balancing that gives full consideration to the high quality variation of EOL returns of cores.
3. Further advance the analytical modeling framework to analyze transfer lines that perform restrictive split routing logic and disassembly and split operations at the same workstation.

1.4 Dissertation Organization

The thesis is organized in a multiple manuscript format as shown in Figure 1-4. In Chapter 2, multiple sequence generation models are developed for disassembly sequence generation that considers material properties, partial disassembly, sequence dependent task times, and the possibility that components of the core can have multiple EOL conditions. A two-stage approach and a component hedging policy are developed to determine the optimal disassembly sequence to maximize the weighted average value. In Chapter 3, a method to create

a disassembly joint precedence graph is developed that is able to decompose all EOL state information for a core into a single precedence graph. A stochastic joint precedence graph method is also developed, along with two line balancing algorithms, to show the effectiveness of using a disassembly joint precedence graph. In Chapter 4, a decomposition method is developed that enables the performance analysis of production systems performing a restrictive split routing logic and disassembly and split at the same workstation. This research further advances the analytical modeling framework. In Chapter 5, conclusions and future work are presented.

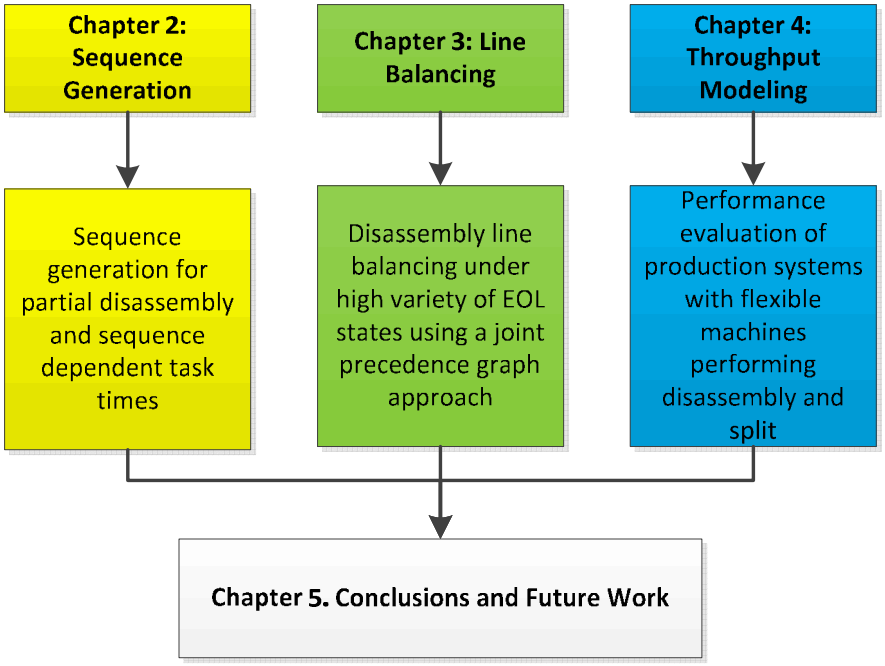


Figure 1-4 Organization of the dissertation

CHAPTER 2

SEQUENCE GENERATION FOR PARTIAL DISASSEMBLY OF PRODUCTS WITH SEQUENCE DEPENDENT TASK TIMES[1]

Abstract: Disassembly is an important process during the life cycle of products. During disassembly, value is mined from cores through reuse of components and raw materials. Disassembly can also be very difficult, inefficient, and even remove reuse value from components due to damage during the disassembly process. We developed a series of models to aid in finding optimal disassembly sequences in the presence of many EOL states. The first model orders a disassembly sequence such that components and modules that have higher reuse value will be removed from the core before lesser valued components in order to reduce the risk of possible damage during the disassembly process. We also consider the material properties of the components and create a value loss function to model the likelihood a component will lose value if damaged. We include partial disassembly and sequence dependent task times and developed a two stage model that first determines the optimal partial disassembly sequence, followed by the second stage that finds the optimal partial disassembly sequence where sequence dependent task times are included. We prove the optimality of the two stage approach under

[1] This chapter is based on a working paper titled "Sequence generation for partial disassembly of products with sequence dependent task times" to be submitted and a peer reviewed Conference Paper submitted to the 22nd CIRP conference on Life Cycle Engineering titled "Two stage sequence generation for partial disassembly of products with sequence dependent task times"

certain conditions and developed an effective heuristic hedging policy if certain conditions cannot be met.

Keywords: Disassembly sequence generation, End of Life, EOL, partial disassembly, sequence dependent task times

2.1 Introduction

When products and assemblies reach the end of their useful lives to a consumer, they are disposed of in some way. A promising and responsible process is remanufacturing, which is the process of recovering products through sorting, cleaning, and disassembling into modules or components, so that components or modules are reused directly or recycled (Lund, 1984). Disassembly is the backbone of the remanufacturing process because this is the stage where the end of life (EOL) value of components and modules can be realized. An assembly or core at its EOL contains value in select components and modules, but this value has to be mined through disassembly.

Disassembly is typically a very manual process where workers use their hands and tools to disassemble a product to a predetermined point (McGovern et al., 2010). There is also a lot of variation associated with the disassembly of cores. The task time to disassemble each component is typically a distribution since the work is manually done and the condition of the cores can range in condition. There is also variation in the actual EOL condition of each component. Components can be missing, damaged, either have reuse value or no reuse value depending on the damage to it, and these combination of EOL conditions creates many possible EOL states the core can be in. In addition to the variation of EOL states the core can be in, the actual

disassembly process can be quite harsh and damaging to the core's components (Jovane et al., 1993; Bras et al., 1996). "Disassembly damage" describes the process where aged parts in a core of the returned products are likely to break during the disassembly process (Gungor et al., 1998). Damage can occur because of gravity where parts fall due to the orientation during disassembly, or there is improper handling either due to operator error or during a long and difficult disassembly process (Williams et al., 2001).

One method to help reduce the possibility of disassembly damage is the design of fixtures (Jovane et al., 1993), but it can be difficult to design a fixture for all the possible orientations a core might need to be in during the disassembly process. Another idea is to extract the most valuable components and modules from the core as early as possible in the disassembly sequence to prevent the possibility of damage to the most valuable components or modules. This is one of the primary goals of this Chapter, to create a disassembly sequence model that seeks to extract the most valuable components and modules as early in the process as possible. There is another consideration, however, when considering the risk of losing value for individual components. Some components and modules may be reused directly, so even the smallest of damages, even cosmetic damage but not functional damage, can be detrimental to the value. Some components may be recycled into their raw material forms so damage to them is not costly to the EOL value at all. Some components may be robust, others delicate, and others might be hazardous if damaged and can be dangerous to the health of the workers. In our model, we also want to consider that components and modules may have different value-loss functions that is related to where they are in the disassembly sequence order, meaning the risk of losing value increases the further in the sequence the component is removed.

For disassembly sequence generation, quite often the end point of the disassembly sequence is either complete or selective, meaning a pre-determined end point is chosen before the optimal sequence is determined. We want to consider partial disassembly for our model, where the mathematical program determines the best disassembly stopping point in order to optimize a certain objective. Lastly, we want to create a disassembly sequence model that is able to consider the possibility of sequence dependent task times. Sequence dependent task times refer to the relationship certain components have with one another based on the order they are removed from the core. For example, if component A is removed before component B, then the removal time of component B may decrease from 8 time units to 5 time units; however, if component A is removed after component B, then the removal time of component B remains 8 time units. Depending on the removal order of components, the overall disassembly time can decrease and we want the disassembly sequence model to take this into consideration.

Smith et al. (2011) developed a rule-based recursive method for disassembly sequence generation that can find near-optimal selective disassembly sequences. Their method eliminates unrealistic disassembly solutions with certain disassembly rules. Tian et al. (2012) used a stochastic disassembly network graph and combine different disassembly decision-making criterion to analyze stochastic time models. With their method, different disassembly probability density functions can be used.

Tian et al. (2012) considered the cost of disassembly with their method for disassembly planning where removal times are random and labor cost can be different. They use a solution algorithm based on stochastic simulation to solve the proposed probability models. Edmunds et al. (2012) developed a method that converts AND/OR graph representations into precedence relations to reduce the overall problem size and enable the use of previously developed solution

techniques. The authors also developed two strategies for converting and AND/OR graph into precedence relations, one implementing dummy nodes and the second strategy creates a pseudo-precedence representation.

The remainder of the Chapter is organized as follows: Section 2 presents several sequence generation methods, including a greedy model, a multi-trend loss model, and a two stage model for partial disassembly and sequence dependent task times. Section 3 contains an example problem to further explain each sequence model. Section 4 discusses the results, and Section 5 concludes the paper and summarizes possible future work.

2.2 Method

The assumptions for each model are the same. The EOL condition possibilities for each component or set of components are assumed to be known and the probability of that condition is also assumed; therefore, the EOL state probabilities are also known.

The notation for each model is the following:

Indexes:

- i for the component number (1-N);
- j for the EOL state;
- k for the position in the sequence (1-K).

Parameters:

- N for total set of components to be disassembled ($i = 1,\ldots,|N|$);
- J for the total number of EOL states ($j = 1,\ldots,|J|$);

- K for the total number of positions in the sequence ($k = 1,\ldots,|K|$);
- v_{ij} for the profit/value of component i in EOL state j;
- p_j for the probability of EOL state j;
- w for the weight on the objective;
- $P(i)$ for the immediate successor set for task i;

Variables:

- $x_{ik} = 1$, if component i is removed in the sequence at position k, 0 otherwise;
- $z_i \in (1,..,K)$, component i is assigned to the $(1,\ldots,K)$ position in the sequence.

2.2.1 Greedy EOL Value Optimization Model with Linear Value Loss

The greedy EOL value model only considers the possibility that each component or set of components have the possibility of losing some value over the course of the sequence. The earlier a component is removed toward the start of the sequence, the less likely the value of that component will be diminished due to the disassembly process. The value loss function is assumed to be linear with the same value of w for every component, EOL state, and position in the sequence. Due to this fact, a component with a higher initial value will have the possibility of "losing" more value for each position in the sequence k as compared to a component with a lower initial value. This will force the components with the highest initial reuse value to the beginning of the sequence, hence the name "greedy" model. We also give consideration to the probability of each EOL state, so each reuse value is weighted based on the probability p_j.

The objective function (2.1) maximizes the total EOL value for all EOL states j.

$$maximize\ Y = \sum_{i=1}^{N} \sum_{j=1}^{J} \sum_{k=1}^{K} p_j * \left(x_{ik} * v_{ij} - \left(w * v_{ij} * \left(\frac{\mathrm{k} * x_{ik}}{K} \right) \right) \right) \quad (2.1)$$

The objective function (2.1) can be re-written as (2.1b).

$$maximize\ Y = \sum_{i=1}^{N} \sum_{j=1}^{J} \sum_{k=1}^{K} p_j * v_{ij} \left(x_{ik} - \mathrm{w} * \left(\frac{\mathrm{k} * x_{ik}}{K} \right) \right) \quad (2.1b)$$

For the objective function (2.1b), the value v_{ij} is weighted by the term p_j, the probability of each EOL state *j*. The position the component *i* is removed in the sequence z_i, which goes from 1 to *K*, is divided by *K*, the total number of spots in the sequence, making it a decimal number or equal to 1. This term is weighted by the value of w and subtracted from the binary variable x_{ik}, which is 1 if component *i* is in position *k* of the sequence. The constraints for the mixed integer program are the following:

Constraint (2.2) ensures x_{ik} is a binary integer variable.

$$x_{ik} \in \{0,1\}\ \forall\ i \in N\ and\ k \in K \quad (2.2)$$

Constraint (2.3) ensures that task *i* holds only one place in the sequence.

$$\sum_{k=1}^{K} x_{ik} = 1\ \ \forall\ i \in N \quad (2.3)$$

Constraint (2.4) guarantees that each place in the sequence only contains one task *i*.

$$\sum_{i=1}^{N} x_{ik} = 1 \ \forall k \in K \tag{2.4}$$

Constraint (2.5) defines the value of z_i.

$$\sum_{k=1}^{K} x_{ik} * k = z_i \ \forall i \in N \tag{2.5}$$

Constraint (2.6) ensures z_i is an integer in the range of 1 to K.

$$z_i \in \{1,2, \dots, K\} \ \forall i \in N \tag{2.6}$$

Constraint (2.7) are the precedence constraints.

$$z_i < z_{i`} \ \forall i, i` \in N, i \neq i`, i` \in P(i) \tag{2.7}$$

2.2.2 Multi-Trend Value Loss Models

The components that make up a core for disassembly have their own unique physical characteristics. Some components may be very robust, such as metal components whose value comes from being recycled, and are very unlikely to lose any possible value due to damage during the disassembly process no matter where in the sequence they are removed. Other components are more fragile, such as electronics whose value can be completely lost due to any damage to them, and their order in the disassembly sequence is more sensitive.

In addition, the removal of some components can be hazardous to workers if something is damaged so the "cost" is not really measured in terms of the value loss of the component but the

danger posed to workers. There can also be components that have an increasing chance of being damaged the further into the disassembly sequence they are removed based on the physical location that component has in the assembly. The value loss trends to model each of these scenarios may be linear, quadratic, and square-root. Some guidelines for which trend to use for the value loss function are below. It is important to note that these are guidelines and not standards. It is difficult to assess the potential "value loss" of each component depending on its position in the sequence and even more difficult to compare different components to each other in terms of their loss functions.

Linear Trend:

- Should be selected for components/modules that are not hazardous to workers during disassembly.
- Components that do not have an increasing or decreasing chance of being damaged.
- If damage to the component removes all reuse value, then *w* can equal 1.0.
- If damage to component does not remove reuse value (i.e. component is to be recycled), the slope should be flat.

Quadratic Trend

- Used for components that can have an increasing chance of becoming damaged during the sequence.

Square Root Trend:

- Used for components that can pose a hazard to workers if the component is damaged during disassembly.
- Since this trend is estimated by a linear trend, the shape does not have to follow a true square root curve and can be customized to any trend or shape.

2.2.2.1 Linear/Quadratic Combination Model

We can model a linear and quadratic loss function with a very minor alteration to the objective function (2.1b). The only changes we need to make is to add a parameter for the exponent pertaining to which loss function is required for each component and index the weight term based on the component. This is reflected in the objective function (2.1c).

$$maximize\ Y = \sum_{i=1}^{N} \sum_{j=1}^{J} \sum_{k=1}^{K} p_j * v_{ij} \left(x_{ik} - \left(w_i * \left(\frac{k * x_{ik}}{K} \right) \right)^{E_i} \right) \qquad (2.1c)$$

Subject to (2.2), (2.3), (2.4), (2.5), (2.6), and (2.7)

The parameter E_i will take value 1 for a linear loss trend or 2 for a quadratic loss trend with each based on the component index i, and w_i is the weight term based on component i. For the linear trend, if the component removed in the final position of the sequence means it has zero reuse value, then $w = 1$.

For the quadratic trend, if the component removed in the final position of the sequence has zero reuse value, then $w = \sqrt{1/v_{ij}}$ for component i in state j. If the component removed in the final position of the sequence has some proportional value x of the initial value $(x*v_{ij})$, then

$w = \sqrt{(1-x)/v_{ij}}$ for component *i* in state *j*. The value of w_i will be constant for both the linear and quadratic trends.

2.2.2.2 Linear, Quadratic, and Square Root Value Loss Trend Model

To model the square root value loss trend and subsequently all three value loss trend models with the same objective function, the exponent E_i needs to be manipulated. If the parameter E_i takes a value of ½, there will be solvability issues when using CPLEX. The square root value loss trend can be estimated with a piece-wise linear function. The value of E_i is set to 1 and the weight term will take on different values based on what position k the component is removed at. The objective function that considers all three value loss trends is shown in (2.1d).

$$maximize\ Y = \sum_{i=1}^{N}\sum_{j=1}^{J}\sum_{k=1}^{K} p_j * v_{ij}\left(x_{ik} - \left(w_{ijk} * \left(\frac{k * x_{ik}}{K}\right)\right)^{E_i}\right) \qquad (2.1d)$$

Subject to (2.2), (2.3), (2.4), (2.5), (2.6), and (2.7)

where w_{ijk} is a matrix of lookup values for component *i* in state *j* in position *k*. Based on how the user wants to model the square root trend, the value of w_{ijk} will change based on the position *k* chosen in the sequence for component *i* in state *j*. Note: if a linear or quadratic trend is chosen for certain components, then w_{ijk} will be the same value for all positions *k*. Below are guidelines when using square root value loss trend and how to find the values for w_{ijk}.

- If the component removed in the final position of the sequences means it has zero reuse value, then $w = v_{ij}$ for component *i* in state *j*.

- If the component removed in the final position of the sequence means it has some proportional value x of the initial value $(x*v_{ij})$, then $w = v_{ij} * (1 - 2x + x^2)$ for component *i* in state *j*.

- To determine the value of w_{ijk} when using the linear trend to estimate the square root trend, a table should be made that shows the value component i would have in position. These values will be used to find the weight term by the formula $w_{ijk} = \left(\frac{K}{k}\right) * \left(1 - \left(\frac{y_{ik}}{v_{ij}}\right)\right)$ where y_{ik} is the value of component *i* in position *k*.

2.2.3 Linear/Quadratic/Square Root Combination Model with Sequence Dependent Task Times for Partial Disassembly

The concept of sequence dependent task times is based on the idea that certain components have a relation such that when one component is removed, the other component can be removed with a shorter amount of time. This concept of sequence dependence is realistic for disassembly because as more and more components are removed from the core, fewer components will be present and it is possible that some of those remaining components will now be more accessible and easy to be removed. In addition, partial disassembly refers to the idea that the point when disassembly should be discontinued is not known and the optimization model will tell the user what components should not be removed. This defers from complete and selective

disassembly where all and a chosen end point is decided upon by the decision maker, respectively.

2.2.3.1 Sequence Dependent Task Times Model

For the sequence dependent task time model, the concept of disassembly time will now be brought into the equation. There will be a parameter for the task time and difference in individual task time if a certain component is removed before another component. In the objective function, the task time and time difference parameters will be multiplied by a parameter for the unit time cost for disassembly. An additional variable is also required to determine if a certain component is removed before another component or not.

Parameters:

- sd_{ijm} for the sequence dependent time difference if component *i* is removed before component *m* in state *j*
- c_i for the unit time cost for component *i*
- t_{ij} for the task time for component *i* in state *j*

Variables:

- y_{im} = 1 if component *i* is removed before component *m*, 0 otherwise.

$$maximize\ Y = \left(\sum_{i=1}^{N} \sum_{j=1}^{J} \sum_{k=1}^{K} p_j v_{ij} \left(x_{ik} - \left(w_{ijk} \left(\frac{\mathrm{k} x_{ik}}{K} \right)^{E_i} \right) \right) - \sum_{i=1}^{N} \sum_{j=1}^{J} \sum_{k=1}^{K} p_j c_i x_{ik} t_{ij} - \sum_{\substack{i,m=1 \\ i \neq m}}^{N} \sum_{j=1}^{J} \sum_{k=1}^{K} p_j c_i sd_{ijm} y_{im} \right) \quad (2.1e)$$

It is important to note that in the objective (2.1e), the parameter c_i could have a negative value so it would be plus the second term instead of minus. The first part of the objective function is very similar to each of the previous objectives and is exactly the same as (2.1d). The second term in the objective takes into consideration the "cost" of the total task time for the sequence. The third term in the objective brings in the concept of sequence dependent task times. The unit cost c is multiplied by the time difference parameter and the new variable that states if component i is removed in the sequence before component m or not. The time difference parameter sd_{ijm} does not have to be positive, meaning there is a time savings, but it can be a negative value for certain combinations of components meaning it can take longer time to remove a certain component if another component is removed first. The objective function (2.1e) is subject to the following constraints:

Constraint (2.2) ensures x_{ik} is a binary integer variable.

$$x_{ik} \in \{0,1\} \; \forall \; i \in N \; and \; k \in K \tag{2.2}$$

Constraint (2.3) ensures task i holds only one place in the sequence.

$$\sum_{k=1}^{K} x_{ik} = 1 \;\; \forall \; i \in N \tag{2.3}$$

Constraint (2.4) guarantees each place in the sequence only contains one task i.

$$\sum_{i=1}^{N} x_{ik} = 1 \ \forall k \in K \tag{2.4}$$

Constraint (2.5) defines the value of z_i.

$$\sum_{k=1}^{K} x_{ik} * k = z_i \ \forall i \in N \tag{2.5}$$

Constraint (2.6) ensures z_i is an integer in the range of 1 to K.

$$z_i \in \{1,2, \dots, K\} \ \forall i \in N \tag{2.6}$$

Constraint (2.7) is the precedence constraints.

$$z_i < z_{i`} \ \forall i, i` \in N, i \neq i`, i` \in P(i) \tag{2.7}$$

Constraints (2.8) and (2.9) determine which components/modules are removed before one another.

$$z_i - z_m \leq M * (1 - y_{im}) \ \forall i, m \in N, i \neq m \tag{2.8}$$

$$z_m - z_i \leq M * y_{im} \ \forall i, m \in N, i \neq m \tag{2.9}$$

Constraint (2.10) ensures that y_{im} is integer and binary.

$$y_{im} \in \{0,1\} \; \forall \; i, m \in N \tag{2.10}$$

Constraints (2.11) and (2.12) guarantees y_{im} and y_{mi} are not equivalent.

$$y_{im} \neq y_{mi} \;\; \forall \; i, m \in N, i \neq m \tag{2.11}$$

$$y_{im} + y_{mi} = 1 \;\; \forall \; i, m \in N, i \neq m \tag{2.12}$$

Constraints (2.8) – (2.12) are where the variables are linked together.

2.2.3.2 Two Stage Approach for Partial Disassembly Model with Sequence Dependent Costs

Simply combining the sequence dependent costs model in 2.3.1 and the partial disassembly model in 2.3.2 does not produce a desirable result. The sequence dependent costs model requires that every task be assigned a place in the sequence and this contradicts the requirement of a partial disassembly model that not all tasks must be included in the sequence. To allow the two models to coexist together, a two stage approach is proposed.

In the first stage, the "greedy" disassembly model is used:

Objective Function: (2.1b)

Subject To: Constraints (2.2), (2.3), (2.4), (2.5), (2.6), and (2.7)

The first stage is run for the given problem and a sequence is determined. As long as every component does not have reuse value, there will likely be a position in the sequence, called position Ω, where the component in the sequence has reuse value but every component removed after that position does not have reuse value. This is due to the way the objective function is structured, pushing components with higher reuse value to the front of the sequence and components with zero reuse value to the back of the sequence. All components removed after the Ω position will be hedged from the sequence. If all components are found by the optimization to be needed in the disassembly sequence, the entire component list is fed to the second stage of the problem.

In the second stage, only the components found in stage 1 to be a part of the disassembly sequence will be used as an input. The components not a part of the disassembly sequence in stage 1 will be hedged. The sequence dependent task time model is used for the second stage:

Objective Function: (2.1e)

Subject To: Constraints (2.2), (2.3), (2.4), (2.5), (2.6), (2.7), (2.8), (2.9), (2.10), (2.11), and (2.12)

Proposition 1: *The final disassembly sequence solution using the two stage approach is guaranteed to be at least as good as any other disassembly sequence solution to the problem under the condition that every component that has some EOL value must be included in the final sequence and the components hedged from the sequence do not have sequence dependent task times.*

Proof: The proof for this proposition is intuitive. If given the condition that every component that has any EOL value must be included in the final disassembly sequence and only components were hedged from the sequence without sequence dependent task times, then only components that have no EOL value will be removed from the sequence. Stage 1 of the two stage approach only removes components from the sequence with no EOL value, and all component removal tasks are hedged from the sequence at the point when (working backwards from the end of the sequence) the first component has EOL value.

In addition, the second stage will find an optimal sequence only using components that are required to be included in the final disassembly sequence and will include sequence dependent task time criteria. □

2.2.3.3 Heuristic Two Stage Policy

Under the condition that not every component with any EOL value must be included in the final disassembly sequence, the two stage approach is no longer guaranteed optimal and a heuristic approach is developed. The approach is the same for the first stage; all components that have no EOL value are pushed to the end of the sequence and are hedged after the position in the sequence where the final component that does have EOL value is assigned. This is shown in Figure 2-1. In position 10, component B is the final component in the sequence that has any EOL value, so all components after position 10 (positions 11 and 12) are removed from the sequence, and all components earlier in the sequence will remain in the sequence for now.

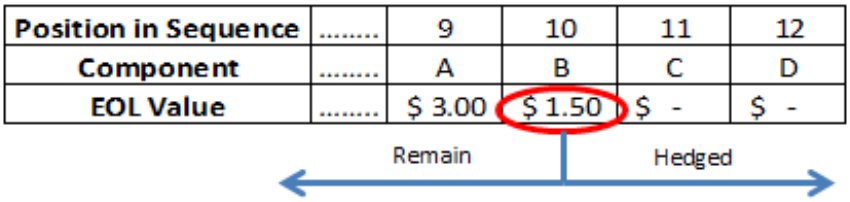


Figure 2-1 First Stage Hedging Example

For the second part of the first stage, a stopping point in the sequence needs to be determined, meaning at what point in the sequence is it still worth it financially to remove a component. To do so, we work our way from the back of the sequence. Based on the example in Figure 2-1, first component B will be assessed in position 10. If the weighted EOL value of component B for all EOL states is greater than or equal to the total cost to remove component B, then component B will remain in the sequence. If it is more expensive to remove component B than the weighted EOL value, then component B will be hedged from the sequence and we will assess the next component in the sequence in position 9. The previously described assessment is shown in equations (2.13) and (2.14).

$$\sum_{j=1}^{J} p_j v_{ij} \geq \sum_{j=1}^{J} p_j t_{ij} c_i \tag{2.13}$$

$$\sum_{j=1}^{J} p_j v_{ij} < \sum_{j=1}^{J} p_j t_{ij} c_i \tag{2.14}$$

If equation (2.13) holds true, then the component remains in the sequence, no other component needs to be assessed, and the process moves on to the second stage of the two stage approach. If equation (2.14) holds true, then the component being assessed needs to be hedged

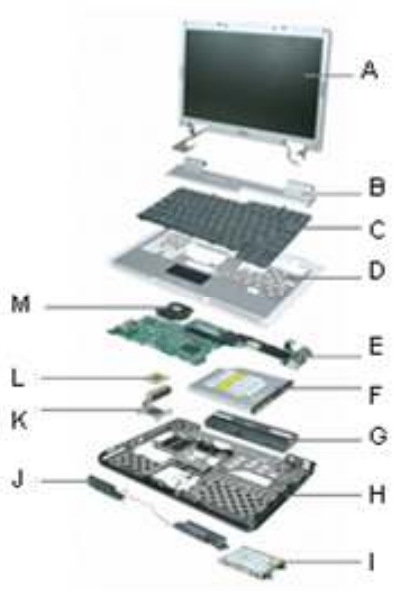
from the sequence and the assessment will continue until a stopping point is found, and then the second stage can commence.

2.2.4 Solution Methodology

The method to solve each of the individual value loss models and the two stage approach is any general optimization software package such as CPLEX. Depending on the value loss function chosen, such as the quadratic value loss function, there can be a non-linear objective function, but CPLEX and most general optimization software packages can solve such non-linear objectives.

2.3 Case Study

To demonstrate how the previously discussed models work, we use the disassembly of a laptop as an example. The laptop example consists of 13 total parts, which is simplified to help explain the different model types. An exploded view of the laptop is shown in figure 2-2(a) and figure 2-2(b) is the part list. A disassembly precedence graph is shown in figure 2-3.



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

(a) (b)

Figure 2-2 (a) exploded view of laptop and (b) the parts list

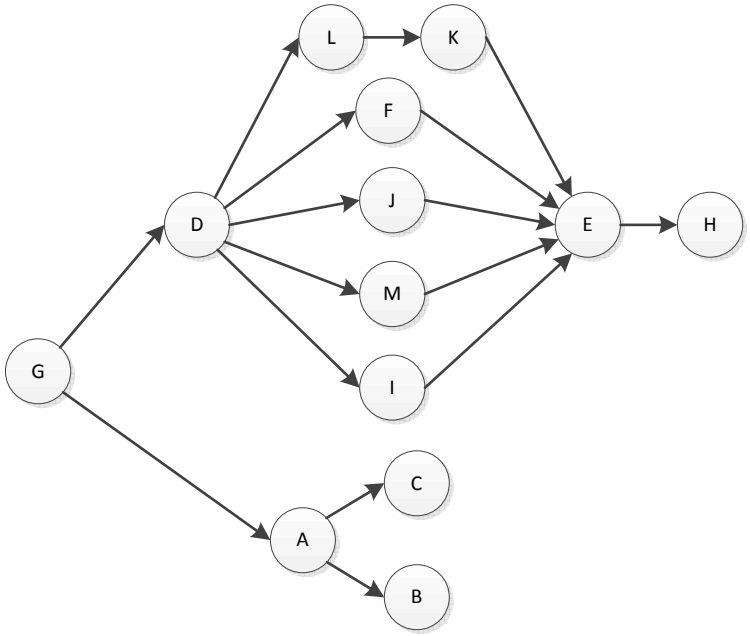


Figure 2-3 Disassembly precedence graph of laptop

The laptop example is used to show results for three different sequence generation models discussed in section 2. These three models are (1) the greedy model, (2) the multi-trend value loss model, and (3) the multi-trend value loss model for partial disassembly and sequence dependent task times (two stage approach). Each model will have the same 13 components of the laptop as an input. Our assumption for each model is that the EOL conditions and the probability of that condition for each component are known. For our example, we created three EOL conditions:

1) Condition 1: There is a 65% probability that the hard-drive (component I) is present in the assembly and a 35% chance the hard-drive is missing.
2) Condition 2: The system board (component E) can be damaged with a 35% probability and if damaged, will have no reuse value.
3) Condition 3: The optical drive (component F) has a 25% probability of being more difficult to remove than normal and results in a longer standard task time.

With the three EOL conditions we can calculate the probability of each EOL state for the core. Since there are three EOL conditions to consider, there will be a total of 8 EOL states for the core, found by raising 2 to the 3rd power for the number of EOL conditions. The 8 total EOL states are shown in table 2-1.

Table 2-1 Probabilities for each EOL state

EOL State	EOL Condition			EOL State Probability
	Hard Drive (I)	System Board (E)	Optical Drive (F)	
1	0.65	0.65	0.75	0.317
2	0.65	0.65	0.25	0.106
3	0.65	0.35	0.75	0.171
4	0.65	0.35	0.25	0.057
5	0.35	0.65	0.75	0.171
6	0.35	0.65	0.25	0.057
7	0.35	0.35	0.75	0.092
8	0.35	0.35	0.25	0.031

The probability of each EOL state is found by multiplying the probabilities of each condition in that state. For example, EOL state 1 has the hard-drive present, the system board not damaged, and the optical drive is not more difficult to remove than normal (0.65*0.65*0.75=0.317). Each EOL state has a different combination of each EOL condition possibility. The EOL state probabilities will be the p_j values that are used in each sequence generation model.

Lastly, each component has a certain value at its EOL, either through direct reuse or recycling. Some components do not have any tangible reuse value, or it might cost more money to recycle the component than it is worth. Table 2-2 contains the EOL values for each component. It is worth noting that these EOL values are only for components that are present in the assembly and not damaged.

Table 2-2 EOL value for each component

Component		EOL Value
A	display assembly	$ 16.00
B	hinge cover	$ -
C	keyboard	$ -
D	palm rest (with touch pad)	$ -
E	system board	$ 12.00
F	optical drive	$ 7.00
G	main battery	$ 8.00
H	computer base	$ 2.50
I	hard drive	$ 11.00
J	speakers	$ -
K	microprocessor thermal-cooling assembly	$ -
L	microprocessor	$ -
M	fan	$ 1.50

2.3.1 Greedy EOL Value Optimization Model with Linear Value Loss

For the greedy sequence generation model, we only take into consideration the EOL value of each component. Therefore, the model will push the components with the highest weighted EOL value towards the front of the sequence. The sequence order for the greedy model is shown in table 2-3 below.

Table 2-3 Disassembly sequence for the "greedy" model

Objective	W	Run time	Disassembly Sequence Order												
			1	2	3	4	5	6	7	8	9	10	11	12	13
41.6397	0.5	0:24	G	A	D	I	F	M	L	K	J	E	H	C	B
EOL Value			$8.00	$16.00	$-	$11.00	$7.00	$1.50	$-	$-	$-	$12.00	$2.50	$-	$-

2.3.2 Multi-Trend Value Loss Models

Table 2-4 contains the model used for each component's value-loss function. The *w* term in each row is the inputted value in the optimization program. The *w* is different based on the

loss function used and within each type of loss function. The *w* value is constant for each component type except for the square root model because it is estimated using a piecewise linear trend with different values of *w* at each possible location in the sequence. The components that have dashes for model type and the value of *w* are because these components have no EOL value and will not affect the disassembly sequence or objective value. The results for the sequence are show in table 2-5.

Table 2-4 Components and model used for loss functions

Component		Model	w
A	display assembly	Square Root	varies
B	hinge cover	-	-
C	keyboard	-	-
D	palm rest (with touch pad)	-	-
E	system board	Quadratic	0.289
F	optical drive	Quadratic	0.447
G	main battery	Linear	0.8
H	computer base	Linear	0.1
I	hard drive	Quadratic	0.302
J	speakers	-	-
K	microprocessor thermal-cooling assembly	-	-
L	microprocessor	-	-
M	fan	Linear	0.5

Table 2-5 Multi-trend disassembly sequence results

Objective	W	Run time	Disassembly Sequence Order												
			1	2	3	4	5	6	7	8	9	10	11	12	13
36.077	-	0:16	G	A	D	F	I	M	J	L	K	E	H	C	B
EOL Value			$8.00	$16.00	$-	$ 7.00	$11.00	$1.50	$-	$-	$-	$12.00	$2.50	$-	$-

2.3.3 Two Stage Approach for Partial Disassembly Model with Sequence Dependent Costs

For the two stage approach, we run the model in section 2.3.1 to first determine which disassembly tasks will be removed from the disassembly sequence. The first stage of the two stage approach is to only determine the partial disassembly sequence. These results are shown in table 2-6. Component H is in the Ω position and every component removed after component H will be hedged from the sequence. Components B and C (the hinge cover and the keyboard, respectively) are removed after component H, so they will be hedged from the sequence and will no longer be required in the disassembly sequence. Every other component will be removed in the sequence.

Table 2-6 First stage disassembly sequence order

Objective	W	Run time	Disassembly Sequence Order												
			1	2	3	4	5	6	7	8	9	10	11	12	13
41.6397	0.5	0:24	G	A	D	I	F	M	L	K	J	E	H	C	B
EOL Value			$8.00	$16.00	$-	$11.00	$7.00	$1.50	$-	$-	$-	$12.00	$2.50	$-	$-

For the second stage, we remove all information for components B and C from the model and move ahead with the second stage. For demonstration purposes, we use a constant value of w=0.5 and for E_i=1. For this model, additional information is required in the form of removal times and sequence dependent task times. Table 2-7 contains the removal times for each component.

Table 2-7 Removal times for each component

Component	Task Time (tu)
A	4
D	5
E	5
F	3
G	2
H	2
I	6
J	5
K	3
L	1
M	2

It is determined that if component J is removed before component L, there will be a time savings of 1 time unit. The results for the optimal disassembly sequence are shown in table 2-8.

Table 2-8 Second stage disassembly sequence order

Objective	W	Run time	Disassembly Sequence Order												
			1	2	3	4	5	6	7	8	9	10	11	12	13
27.999	0.5	2.74	G	A	D	I	F	M	J	L	K	E	H	-	-
EOL Value			$8.00	$16.00	$-	$11.00	$ 7.00	$1.50	$-	$-	$-	$12.00	$2.50	$-	$-

2.4 Discussion

The results for each disassembly sequence for the three models are very similar in terms of the disassembly sequence order. The order of the sequence is highly dependent on how open the precedence graph is, meaning if the disassembly order of components is highly substitutable or somewhat strictly ordered. The disassembly sequences are also similar because the laptop example only contains 13 components.

There are some differences in the disassembly sequences that are worth discussing, however. The sequence in table 2-3, the greedy model, has a sequence that is very much in order of the EOL value for each component. Component G has an EOL value of $8 and component A has a value of $16 and G is earlier in the sequence before A, but based on the precedence graph G must be removed before A can be removed. An example of where the order is strictly based on EOL value, the sequence order of component I, then F, and then M. This is in descending order of EOL value, $11, $7, and $1.50. Components I, F, and M can be removed in any order based on the precedence graph and these components have no impact on the objective function since they have no reuse value.

The disassembly sequence in table 2-5 is based upon the greedy model except now we include material and component properties into the model. The value-loss function of the components can now be a linear, quadratic, or square root trend. The only important difference between the sequences in table 2-3 and table 2-5 is that now component F is removed before component I. For the greedy disassembly sequence, component I has a higher EOL value than component F so it must come earlier in the sequence. When considering component properties for a relative value loss function, the model determined it is better to have component F earlier in the sequence than component I to attain a higher probable value retention level. In addition, it is worth noting that components J, L, and K, which have no EOL value, are in a slightly different order than the sequence in table 2-3. Since these components have no EOL value, their order in the sequence does not impact the objective value. Also, components C and B remain in the final two positions of the sequence and these components also have no EOL value.

The results in table 2-6 show the optimal disassembly sequence for the first stage problem in the two stage model. For the first stage sequence, components that have no EOL

value can be hedged from the sequence if they are removed after the Ω position. Components B and C were in the final two positions of the sequence for each of the previous models and as expected the removal of these components does not add value to the sequence since they have no EOL value. Components B and C will be removed from the sequence and this information will be passed to the second stage of the model. For the second stage of the model, the sequence dependent task times are required between each of the components, and the only sequence dependent task times for this stage is the relationship between components J and L. If component J is earlier in the sequence than L, then there is a time savings of 1 time unit to remove component L. The optimal sequence in table 2-8 states component J should be removed before component L, and in the previous optimal sequences component J could be removed before or after L without an impact on the objective value since both component J and L have zero reuse value. For example, the objective value in table 2-5 would be the exact same if J was in the sequence earlier than component L or if it came after. The objective value in table 2-8 would not be the same if component L was assigned earlier in the sequence than component J due to the process times of each component and the sequence dependent task time between components being included in the model.

2.5 Conclusions and Future Work

In this paper we presented a series of disassembly sequence generation models that built upon each other. We first presented a "greedy" sequence model that ordered the sequence so the components with the highest EOL value would be pushed towards the front end of the sequence and components with little to no EOL value pushed towards the back of the sequence. Next, we added upon this model by including the possibility that some components may not lose their

value as easily as others and modeled this using various value-loss functions with different shape parameters. Lastly, we developed a two stage model that would first determine the partial disassembly sequence, meaning which components do not need to be removed in the sequence, and followed by a second stage that would determine the optimal disassembly sequence when including sequence depending task times.

The future work should focus on the limitations of the two stage model. For a product with a few components to consider for the disassembly sequence, such as the laptop example, each stage of the model runs very quickly. But for larger problems with many components, having to run two integer programs to come to a final solution may be too time consuming and not efficient. Eliminating the need for a two stage problem and reducing it into a single model can be an important improvement. In addition, a heuristic approach to this problem can also give acceptable sequence solutions while decreasing the run time significantly for larger problem sets.

Lastly, more work is needed to understand the relationship the value-loss functions have between each other. Currently, the value-loss functions are relative and there are simple guidelines of which type of loss function to use for a certain type of material property or component.

CHAPTER 3

DISASSEMBLY LINE BALANCING UNDER HIGH VARIETY OF EOL STATES USING A JOINT PRECEDENCE GRAPH APPROACH[2]

Abstract: 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

[2] This chapter is based on a paper titled "Disassembly line balancing under high variety of EOL states using a joint precedence graph approach" that has been accepted in the Journal of Manufacturing Systems

how a stochastic joint precedence graph can be created and used in a stochastic line balancing formulation.

Keywords: Disassembly line balancing (DLB), joint precedence graph, end of life (EOL) condition, stochastic line balancing

3.1 Introduction

The end of life (EOL) 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 (Ozdemir et al., 2012). Due to this fact, manufacturers are increasingly recovering, remanufacturing, and reprocessing post-consumer 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. 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.

Disassembly in some form is the common thread for all forms of product recovery. 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 (McGovern et al., 2011; Battaïa et al., 2013). Meacham et al. (1999) 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 (2001) investigated the disassembly line balancing problem (DLBP) in the presence of task failures. Some disassembly tasks may not be completed due to a 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. (2001) also discussed the challenges that come from disassembly line balancing, ranging from product, disassembly line, demand, and assignment complications.

Altekin et al. (2004) 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 (2006) 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. (2007) 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. (2008) developed a 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. (2009) 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. (2009) developed both an integer and dynamic program formulations for the DLBP by using AND/OR graphs to check for feasibility of the solution.

Altekin et al. (2012) developed a two-step approach to rebalance a disassembly line when task failures occur that lead to successor tasks being infeasible. Ma et al. (2011) 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. (2011) 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. (2013) 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.

Ilgin and Gupta (2010) 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. (2012) 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 (Gungor et al., 2001; Altekin et al., 2012)

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 (2003). 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 (2010). Many approaches in the literature use AND/OR graphs (2013) 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.

The literature on stochastic disassembly line balancing is not as broad as the literature for deterministic disassembly line balancing. Agrawal et al. (2008) 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 et al. (2013) developed a multi-objective genetic algorithm to determine the best line balance that considered stochastic task times where station paralleling is allowed.

Many line balancing approaches for disassembly are the same as used for assembly but with a simple inverse or alteration of the precedence graph (Battaïa et al., 2013). "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 (Tang et al., 2002). 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 Chapter 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 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.

3.2 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, Fliedner and Scholl (2009) 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.

3.2.1 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 $0 \leq p_q \leq 1$, and the sum of all p_q values for $q \epsilon Q$ is equal to 1, Q being the set of all EOL states. The average task time for the joint precedence graph (3.1) and the joint precedence arc set (3.2) are found using the following definitions:

$$t_i = \sum_{q \in Q} p_q t_{iq} \quad \forall\, i \in I \tag{3.1}$$

$$E = \bigcup_{q \in Q} E_q \setminus \{reducdant \;\; arcs\} \tag{3.2}$$

Equation (3.1) 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.

3.2.2 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 equations (3.3) and (3.4), respectively.

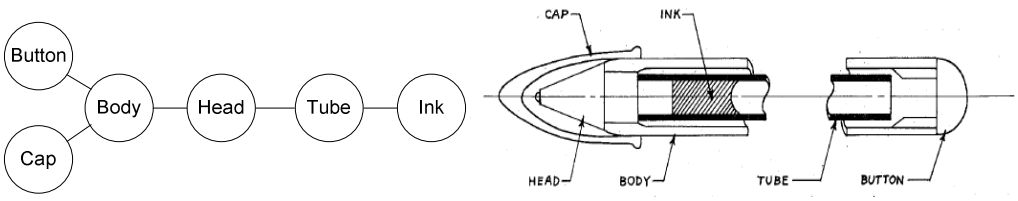
$$\mu_i = \sum_{q \in Q} p_q \mu_{iq} \quad \forall\, i \in I \tag{3.3}$$

$$\sigma_i^2 = \sum_{q \in Q} p_q \left(\left(\mu_{iq} - \mu_i \right)^2 + \sigma_{iq}^2 \right) \quad \forall\, i \in I \tag{3.4}$$

The equations (3.3) and (3.4) 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 (3.4) to show the relative spread of having a mixture of two or more task times.

3.2.3 Example: joint precedence graph generation

Figure 3-1 contains a liaison graph and a cross-section view of a pen taken from a paper by De Fazio et al. (1987). 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 figure 3-2. 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.



(a) (b)

Figure 3-1 (a) pen liaison graph and (b) pen cross-section (De Fazio et al., 1987)

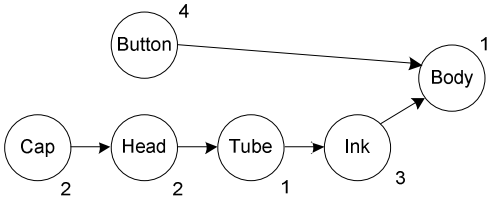


Figure 3-2 Pen disassembly precedence graph

The values shown next to each node in Figure 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 have a dependency between 2 components (head and tube), the 8 possible precedence graphs decreases 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 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 3-2 shows the probability of each EOL state and the corresponding disassembly task time for the EOL state.

Table 3-1 (a) coded EOL state and (b) disassembly probability of each EOL state

(a)

EOL State	Cap	Head	Tube
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)

EOL State	Cap	Head	Tube
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

Table 3-2 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

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 figure 3-3. 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 equations (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 figure 3-4.

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.

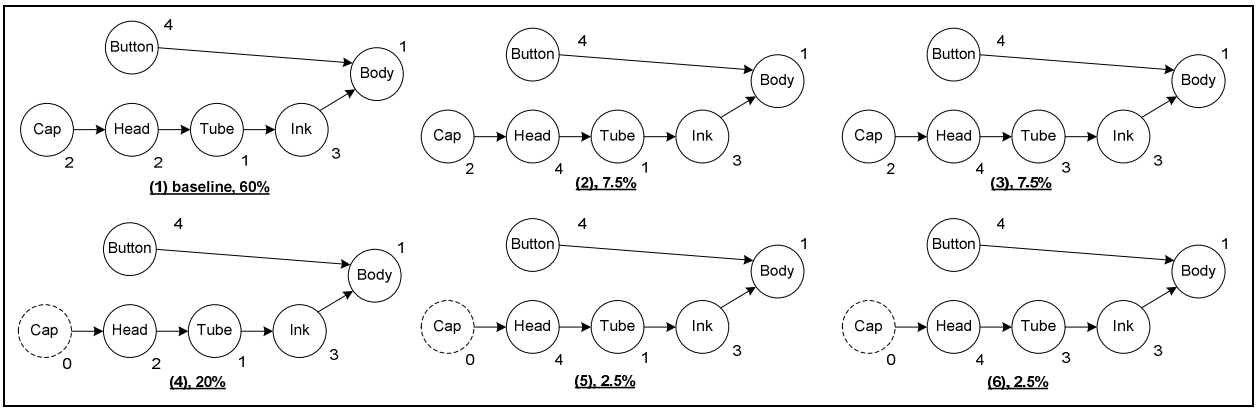


Figure 3-3 EOL disassembly precedence graphs for pen

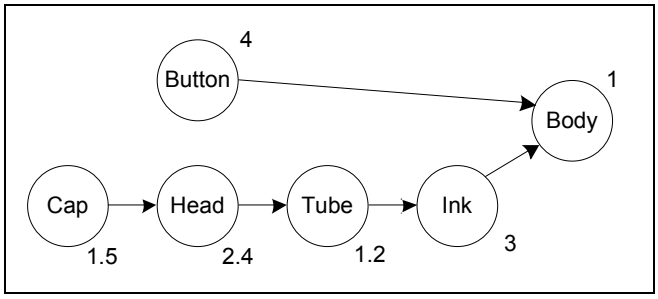


Figure 3-4 Joint precedence graph for pen disassembly

Now let's 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 3-3 reflects the change that each non-zero task time has a variance of 0.25 tu. Applying equations (3) and (4) from section 2.2, we can generate a stochastic joint precedence graph, shown in figure 3-5.

Table 3-3 Probability and time data for each EOL state

EOL State	Probability	Cap, N(μ,σ2)	Head, N(μ,σ2)	Tube, N(μ,σ2)
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)

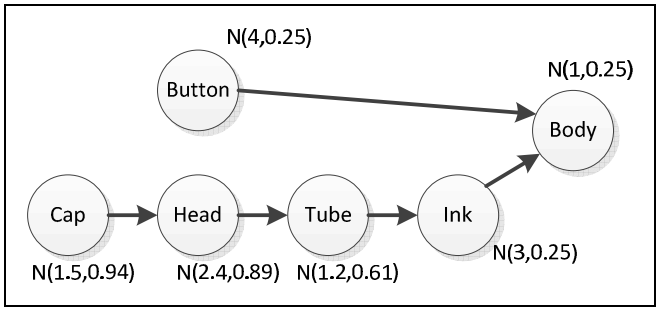


Figure 3-5 Stochastic joint precedence graph for pen disassembly

The stochastic joint precedence graph in figure 3-5 is very similar the joint precedence graph in figure 3-4, except now each task time is a normal variable with a mean and variance.

3.3 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.

3.3.1 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:

$$min\left(max\left(\sum_i X_{ik} t_i - \sum_{i'} X_{i'k'} t_{i'}\right)\right) \quad \forall\, i, i' \in I \text{ and } k, k' \in K, \; i \neq i' \text{ and } k \neq k' \tag{3.5}$$

Subject to:

$$\sum_k X_{ik} = 1 \quad \forall\, i \in I \tag{3.6}$$

$$X_{ik} \leq \sum_{h=1}^{k} X_{i'h} \quad \forall\, i, i' \in I, \; i \neq i', \; k \in K \text{ and } i' \in P(i) \tag{3.7}$$

$$X_{ik} = 0{,}1 \; \forall\, i \in I \text{ and } k \in K \tag{3.8}$$

The objective function (3.5) 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 (3.6) is an occurrence constraint where every disassembly task must be assigned a workstation and only one workstation. Constraint

(3.7) are the precedence constraints (Baybars, 1986) 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 (3.8) is a non-divisibility constraint that requires task *i* to not be split amongst multiple stations.

$$X_{ik} = X_{i'k} \quad \forall\, i, i' \in P(tool\_i), k \in K \tag{3.9}$$

$$\sum_i X_{ik} \leq Z \quad \forall\, k \in K \tag{3.10}$$

Constraint (3.9) 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 (3.10) 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:

$$\min\ Y \tag{3.5a}$$

Subject to:

$$\sum_i X_{ik} t_i - \sum_{i'} X_{i'k'} t_{i'} \geq -Y \quad \forall\, i, i' \in I \text{ and } k, k' \in K,\ i \neq i' \text{ and } k \neq k' \qquad (3.5b)$$

$$\sum_i X_{ik} t_i - \sum_{i'} X_{i'k'} t_{i'} \leq Y \quad \forall\, i, i' \in I \text{ and } k, k' \in K,\ i \neq i' \text{ and } k \neq k' \qquad (3.5c)$$

(3.6), (3.7), (3.8), (3.9), (3.10)

Objective function (3.5a) minimizes a float variable *Y* that is in constraint (3.5b) and (3.5c). Constraint (3.5b) ensures that the difference between total workstation time is greater than or equal to the negative of *Y* and constraint (3.5c) ensures that difference between the total workstation time is less than or equal to the positive value of *Y*.

3.3.2 Stochastic disassembly line balancing formulation

The stochastic disassembly line balancing formulation is very similar to the deterministic line balancing formulation in 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 (3.11) 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^2_i.

Objective Function:

$$min(\max\ E\ [\sum_i X_{ik}V_i - \sum_{i'} X_{i'k'}V_{i'}])\ \ \forall\ i, i' \in I\ and\ k, k' \in K,\ \ i \neq i'\ and\ k \neq k' \quad (3.11)$$

Subject To:

(3.6), (3.7), (3.8), (3.9), (3.10)

$$\sum_i X_{ik}\sigma_i^2 \leq \Sigma\ \ \forall\ k \in K \quad (3.12)$$

Constraint (3.12) 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 (3.12), 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:

$$\min\ M \quad (3.11a)$$

Subject To:

$$\sum_i X_{ik}\mu_i - \sum_{i'} X_{i'k'}\mu_{i'} \geq -\mathrm{M} \quad \forall\, i, i' \in I \text{ and } k, k' \in K,\ i \neq i' \text{ and } k \neq k' \qquad (3.11b)$$

$$\sum_i X_{ik}\mu_i - \sum_{i'} X_{i'k'}\mu_{i'} \leq \mathrm{M} \quad \forall\, i, i' \in I \text{ and } k, k' \in K,\ i \neq i' \text{ and } k \neq k' \qquad (3.11c)$$

(3.6), (3.7), (3.8), (3.9), (3.10), (3.12)

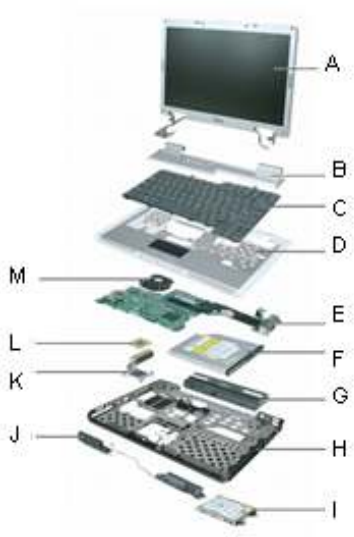
Proposition 1*: The* $E[\sum_i X_{ik}V_i - \sum_{i'} X_{i'k'}V_{i'}]\ \forall\, i, i' \in I \text{ and } k, k' \in K, i \neq i' \text{ and } k \neq k'$*, where* μ_i *and* $\mu_{i'}$ *are the means for normally distributed variables, is equivalent to* $\sum_i X_{ik}\mu_i - \sum_{i'} X_{i'k'}\mu_{i'}$.

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, $E[\sum_i X_{ik}V_i - \sum_{i'} X_{i'k'}V_{i'}] = E[\sum_i X_{ik}V_i] - E[\sum_i X_{i'k'}V_{i'}] = \sum_i X_{ik}\mu_i - \sum_{i'} X_{i'k'}\mu_{i'}$. □

3.4 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 (Hu et al, 2011; Riggs et al., 2013). The exploded view of the laptop was found on dell's website. The dell laptop has 13 components, as shown in figure 3-6. 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 figure 3-7.



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

Figure 3-6 Exploded view of dell laptop and parts list (Hu et al., 2013)

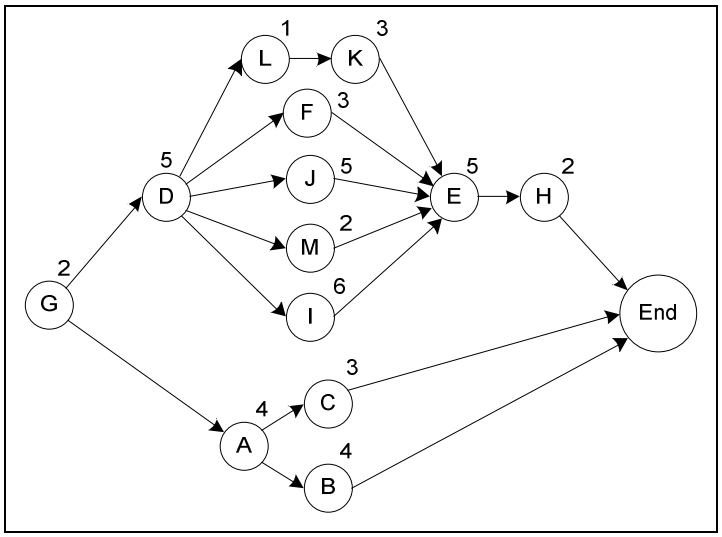


Figure 3-7 Baseline disassembly precedence graph of laptop

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 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 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 3-4 with all relevant information.

Table 3-4 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

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 3-4 can be used to create a joint disassembly precedence graph for the laptop example, shown in figure 3-8. 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.

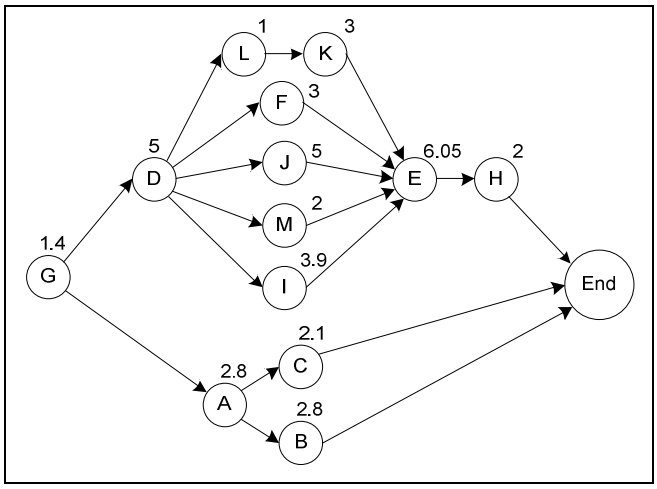


Figure 3-8 Disassembly joint precedence graph for laptop example

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 seconds.

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 hours to reach a steady state and the disassembly 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. The average of the 5 simulation runs was used as the throughput value for each scenario. The standard deviation of the 5 simulation runs for each scenario was less than 0.010 cores per hour, so we only report and use the average for throughput comparison purposes because the standard deviation is so low due to the long simulation run times. Table 3-5 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 3-6 contains the average workstation utilization percentages (waiting, working, and blocking percents) for the simulation runs of each line balance.

Table 3-5 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	

Table 3-6 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%

3.4.1 Line balancing sensitivity

The results in table 3-5 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 3-7.

Table 3-7 Results of laptop line balancing sensitivity

Component Probabilty	EOL State	Probability	Line Balance				Average Throughput	Percent Differenc
			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	

3.4.2 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 $\Sigma = 14$ tu^2 for the total variance in constraint (3.12) for the stochastic line balancing formulation in 3.2. Tables 3-8 and 3-9 shows the mean, variance, and standard deviation for each task using equations (3.3) and (3.4) 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 3-10. The line balancing task assignment in table 3-7 is the same as in table 3-4 from the deterministic case because the total workstation variance constraint did not require tasks to be reassigned to different workstations.

Table 3-8 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

Table 3-9 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 3-10 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	

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 3-11.

Table 3-11 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	

3.5 Discussion

The results in table 3-5 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 difference in each table always compares the increase in throughput the joint line balance has when compared to the baseline line balance. Table 3-6 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% to 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.

3.5.1 Line balancing sensitivity discussion

The line balancing sensitivity results in table 3-7 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 3-12 and the EOL states that result in a total workstation time of zero is highlighted in gray.

Table 3-12 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

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:

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

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.

3.5.2 Stochastic line balancing sensitivity discussion

When comparing the task assignments in tables 3-5 and 3-10, the baseline line balances are the same in each and the joint line balances are also the same. Table 3-5 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 figure 3-9. These results are intuitive because if the total workstation 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.

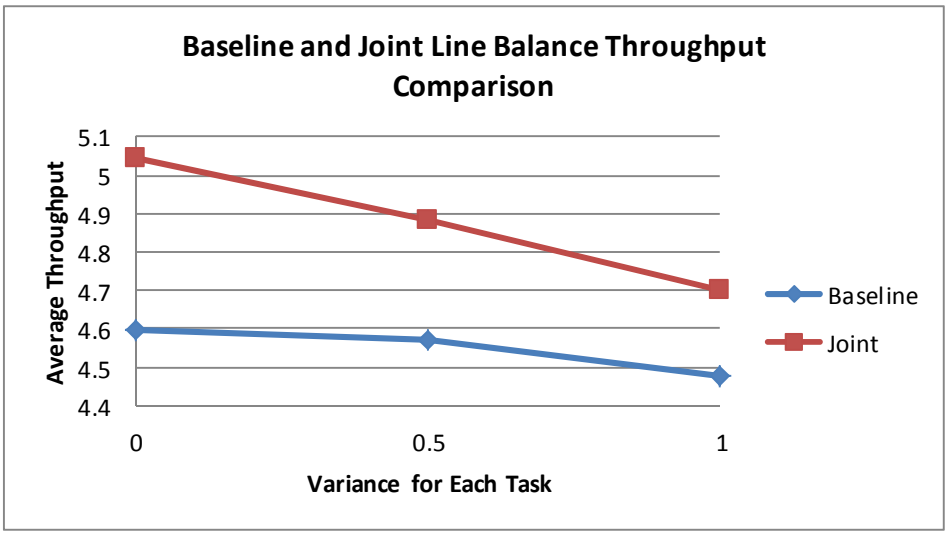


Figure 3-9 Comparison of baseline and joint line balances average throughput

Tables 3-13 and 3-14 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.

Table 3-13 Workstation utilization results for stochastic line balance, stdev=0.5

Baseline Case (Stdev=0.5)			Joint Case (Stdev=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 3-14 Workstation utilization results for stochastic line balance, stdev=1.0

Baseline Case (Stdev=1.0)			Joint Case (Stdev=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%

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 3-11 in section 3.2

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.

3.6 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.

CHAPTER 4

PERFORMANCE EVALUATION OF PRODUCTION SYSTEMS WITH FLEXIBLE MACHINES PERFORMING DISASSEMBLY AND SPLIT[3]

Abstract: This paper presents an approximate analytical method for evaluating the performance of production systems in which a machine has both (1) a restrictive split routing logic and (2) simultaneous split and disassembly operations which are realized by the same machine. The proposed method is based on the decomposition of the complex system into a set of simpler building blocks. The set of building blocks models the entire behavior of the original system, and the performance of each building block is evaluated by using decomposition equations. The numerical results reported in this paper demonstrate the accuracy of the method.

Keywords: Analytical modeling, disassembly, flow lines, performance evaluation, split

4.1 Introduction

This paper presents a new decomposition approach related to flow lines with unreliable machines and finite buffers that have a restrictive split and a simultaneous split and disassembly

[3] This chapter is based on a paper that was submitted to IEEE titled "Performance evaluation of production systems with flexible machines performing disassembly and split"

operation in the flow of material. A split is the operation in which a single upstream machine feeds one of the two or more parallel downstream buffers with the same part type. A disassembly is the operation in which a single upstream machine feeds all parallel downstream buffers with a different part type each. Disassembly lines are becoming more flexible and dynamic in order to become more cost effective, and this presents a problem with how to model such flexible and dynamic routing logics using analytical modeling techniques to determine performance measures. For example, a machine has the task of disassembling a subassembly into three smaller modules for reuse. Module A is always going to be routed downstream for further processing, module B will be routed to 1 of 2 downstream split streams based on if the module can be reused or needs to be recycled, and module C will be routed to 1 of 3 downstream split streams based on its end of life (EOL) condition for reuse. The system layout for this example is represented in figure 4-1. The squares are machines (Mi) and the circles are buffers (Bij). In this case, M1, M3, M4, M5, M6, M7 and M8 are dedicated machines, i.e. they can produce only one part type. On the contrary, M2 is a flexible machine that can produce more than one part type. For each split stream, the probabilities α^{24} and α^{25} will add up to one and β^{26}, β^{27}, and β^{28} as well. The selection of which part type to produce depends on the state of the system and on a dispatching rule. Determining the performance of manufacturing systems with this kind of flexibility and complexity quickly is important for the design, implementation, and operations stages of the systems.

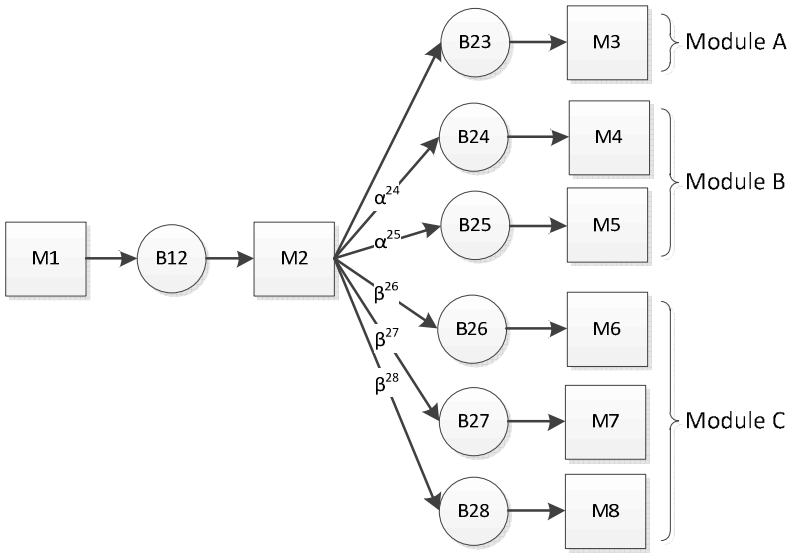


Figure 4-1 System layout for a "split" workstation

Throughput performance of production systems have been analyzed in the literature by using several tools, among them simulation and analytical methods. These tools were developed to predict the productivity performance of manufacturing systems, such as average throughput, work in process (WIP), etc. The diffusion of these techniques contributed to creating a solid knowledge in the manufacturing system design field. Simulation is very flexible since it can handle any degree of detail in the description of the system behavior. Unfortunately, it requires a considerable modeling effort and long simulation runs to determine steady state behavior (Banks et al., 2009; Taylor et al., 2005). In practice, it can take a long time to develop a detailed simulation model to capture the effects of main factors related to performance measures. In addition, the output of the simulation run is the outcome of a statistical experiment since a simulation run generates one of many possible realizations of the system's dynamic behavior. On the other hand, analytical methods have an advantage in their synthesis of the main behavior of complex systems in a few related variables, most of the time implicitly, by dynamic equations.

Analytical methods are very fast in evaluating the performance of a system and generally they provide accurate results.

Li et al. (2009) provides an extensive literature review for the state of the art of throughput analysis techniques. Their paper details the literature for performance evaluation of serial lines for both discrete time and continuous time models. Gershwin (1994) modeled manufacturing lines using Markov chains, and the book presents manufacturing systems as both discrete and continuous time models. Gershwin's work provided detailed derivation of balance equations for a standard building block consisting of two machines and a finite sized buffer in between, and the machines have a single failure and repair probability. The book also described how to decompose a long transfer line to determine the performance of the line. Gershwin focused on a decomposition method while work by Jacobs and Meerkov (1995) focused on an aggregation approach to evaluate the throughput of long transfer lines.

Tolio, Gershwin, and Matta (2002) modeled a two machine line with multiple failure modes. The results are similar to modeling two machines with single failures, except multiple failure models provide more exact approximate results from decomposition when decomposing a long transfer line and determining various performance measures, like throughput and average WIP.

Colledani, Gandola, and Matta (2008) created an approximate analytical method for determining performance measures for a system producing multiple products. The performance of each building block is evaluated by using an aggregation technique. Colledani, Matta, and Tolio (2005) created a performance evaluation setup for manufacturing lines and developed the competition failure idea that occurs in systems performing either split or merge. Chiang, Kuo, and Meerkov (1998) address bottlenecks in productions lines and their relationship to the

machines reliability. Feit and Wu (2000) considered the situation where transfer lines are under high uncertainty and focus on a procedure for information gathering to reduce the uncertainty. Li and Huang (2005) study split/merge systems making multiple products and they analyze the system using a system-theoretic approach, also known as an overlapping decomposition. Liu and Li (2010) further the research of split/merge systems by modeling these systems with different policies, such as circulate, semi-circulate, priority, and percentage policies.

The current literature for split systems does not reflect a method for analyzing systems with restrictive split routing logic. An example of a restrictive split routing logic is the treatment of module B in figure 4-1 having 2 routing streams based on its EOL condition. Module B can be reused or recycled, but if the situation arises where the recycling buffer for module B is full, module B cannot be simply rerouted to the reuse buffer. In addition, the literature for split/merge and assembly/disassembly systems are strictly separate and no method exists to approximate the performance of systems that have flexible workstations that perform both a split and disassembly routing logic at the same workstation.

The remainder of the paper is organized as follows: section 2 outlines the assumptions and notations used in the paper. Section 3 contains a specific description of the method used broken down into 3 subsections, a section on the method of modeling a restrictive split, a section on the method of modeling disassembly and split occurring together at the same workstation, and a section on the solution algorithm used in this paper to evaluate the performance of transfer lines with these layouts. Section 4 reports numerical results for both methods described in this paper and a discussion of the results. Section 5 closes the paper with conclusions.

4.2 Assumptions and notations

The method developed in this paper to analyze flexible transfer lines with complex routing logic is applied to relatively simple layouts for ease of presenting numerical results, but it can be applied to transfer lines with multiple machines. Only one buffer is located between a machine and an immediately downstream machine, and the capacity of each buffer is finite. All machines are unreliable and can fail in multiple modes. The methods developed in this paper are approximate and the transfer lines are broken down into building blocks (BBij) for purposes of decomposition.

The nomenclature and notation used in this paper are outlined below:

i) Nomenclature for machines, buffers, and building blocks.

Mi: machine i in the original line is represented graphically as a square in figures

$M^U(ij)$: refers to all pseudo-upstream machines from buffer Bij

$M^D(ij)$: refers to all pseudo-downstream machines from buffer Bij

Bij: buffer between machine i and j in the original line is represented graphically as a circle in figures

BBij: building block for upstream machine i and downstream machine j considered in the decomposition of the original line

ii) Nomenclature for failure/repair probabilities and probability of being in certain states.

$p_{i,x}$: the probability machine Mi fails in mode x, for $x=1,\ldots,F_i$

$r_{i,x}$: the probability machine Mi repairs in mode x, for $x=1,\ldots,F_i$

F_i: the total number of failure and repair modes for machine i

n_{ij}: buffer capacity of Bij

$\pi(\ldots)$: expression for the probability of being in a particular state

E(i,j): expression for the probability BBij is working, i.e. average throughput. Calculated either as the sum of all states where the pseudo-upstream machine of BBij is working and not blocked, or the sum of all the states where the pseudo-downstream machine of BBij is working and not starved.

$\pi(n, 1/0/u_i, 1/0/d_j)$: expression for the probability of being in a state where the upstream machine is either working (1), in a failed state (0), or in any state (u_i). The downstream machine can be working (1), in a failed state (0), or any state (d_j). The buffer between the upstream and downstream machine has capacity n_{ij} but can be in a state where the amount of material in the buffer is between 0 and n_{ij}. See [4] for building block calculations with one failure mode and [6] for building block calculations with multiple failure modes.

$P^b(i,j)$: sum of all states where the pseudo-upstream machine in BBij is blocked

$P^s(i,j)$: sum of all states where the pseudo-downstream machine in BBij is starved

iii) Nomenclature for the Markov chain states used to model a split in this paper.

$W^{i,j}$: BBij is in a working state

$B^{i,j}$: BBij is in a blocked state

S: BBij is in a starved state

R: BBij is down and needs to be repaired

Detailed assumptions used in this paper are listed next. These assumptions are very similar to assumptions used in (Colledani et al., 2005)

- A multiple failure discrete time model is used for each model presented in section 4.
- Machine processing times are identical, discrete, and scaled to a time unit. Homogenous processing times are used for ease of computation but can be relaxed.
- Machines can only fail while in operation, i.e. cannot fail while starved or blocked.
- The last machine is never blocked and the first machine is never starved.
- Probabilities to fail and repair are calculated using a mean time to fail (MTTF) and mean time to repair (MTTR), respectively. MTTF and MTTR are geometrically distributed and have average values of $1/p_{i,x}$ and $1/r_{i,x}$, respectively, for $i=1,\ldots,K$ and $x=1,\ldots,F_i$.

4.3 Description of method

4.3.1 Restrictive split

For ease of explanation, figure 4-2(a) is used to explain the method for solving transfer lines with a restrictive split routing logic. Machine M2 is the machine where a restrictive split will occur. The equations derived in this section will use the numbering scheme of figure 4-2(a) for machines and buffers (such as M1, B12, etc.) and a single failure and repair probability for each machine to clearly explain the developed method. The method can be extended for machines that have multiple local failure modes.

Machine M2 can be modeled as a 12 state Markov chain, shown in figure 4-3. The transition probabilities and states that are bolded and underlined are different from the previous

work for modeling an unrestrictive split by (Colledani et al., 2005). The Markov chain contains combinations of states for working ($W^{2,j}$), starved (S), down (R), and two types of blocking states ($B^{2,j}$ and $B^{*2,j}$). M2 can be in two different blocking states. $W^{2,3}B^{2,4}$ is the state where M2 can work for BB23 but is blocked for BB24 because buffer B24 is full. This state can transition to many other states depending on a number of possibilities. On the other hand, $B^{*2,4}$ is the blocking state where buffer B24 is full and the product is required to go to B24. Notice the only possible states to be transitioned to from this state are back to itself or to $W^{2,3}W^{2,4}$, since material flow will only continue if M2 is able to pass a product to B24.

The method for analyzing a restrictive split routing logic switches between decomposition and aggregation. Figure 4-2(a) is the split system to be analyzed with failure and repair probabilities listed below each machine, and figure 4-2(b) is the decomposed line with the relevant failure and repair probabilities to model the dynamics of a restrictive split.

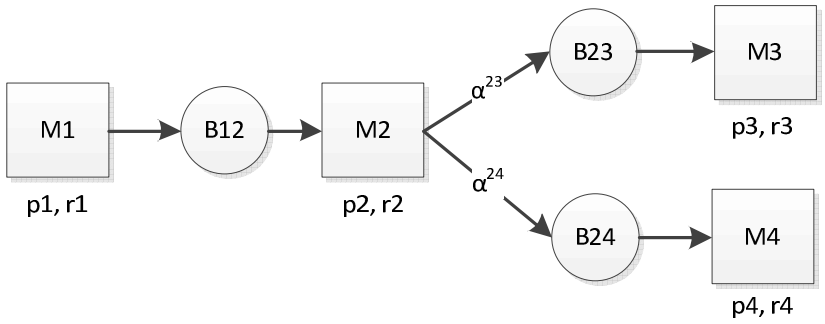


Figure 4-2(a) Restrictive split system layout

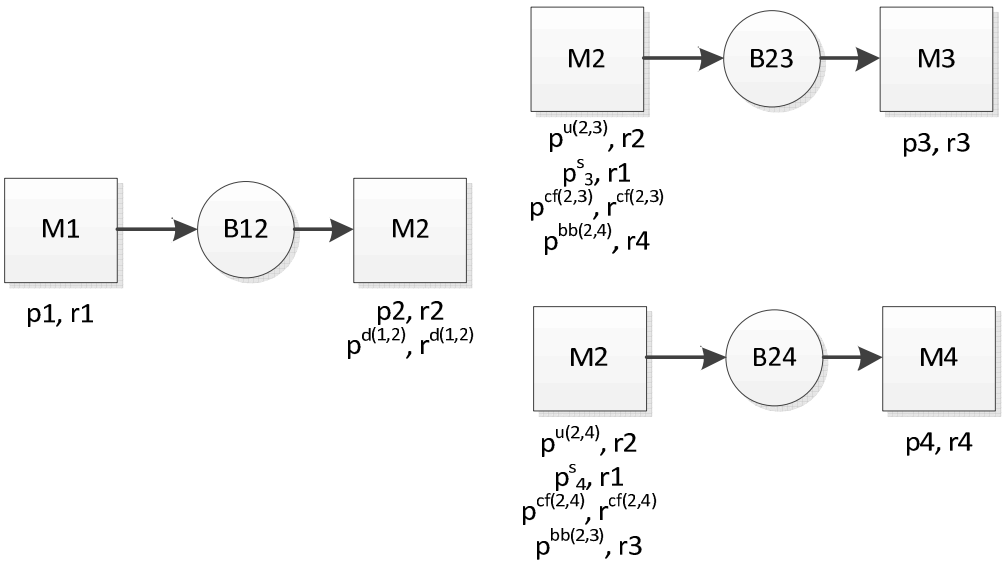


Figure 4-2(b) Decomposed restrictive split layout

The Markov chain in figure 4-3 models the dynamics of machine 2 from a machine level standpoint. When the restrictive split transfer line is decomposed, states in the original M2 Markov chain needs to be aggregated, as shown with equations (4.1)-(4.4). The equations shown are for BB23.

$$\pi(W^{2,3}) = \alpha^{23} * \pi(W^{2,3}W^{2,4}) + \alpha^{2,3} * \pi(W^{2,3}B^{2,4}) \tag{4.1}$$

$$\pi(B^{23}) = \pi(B^{23}B^{24}) + \pi(RB^{23}) + \pi(SB^{23}) + \pi(W^{24}B^{23}) + \ldots$$

$$\ldots + \pi(B^{*24}) + \pi(B^{*23}) + (1-a23) * \pi(W^{24}B^{23}) \tag{4.2}$$

$$\pi(S^{2,3}) = \pi(S) + \pi(SB^{2,4}) \tag{4.3}$$

$$\pi(R^{2,3}) = \pi(R) + \pi(RB^{2,4}) \tag{4.4}$$

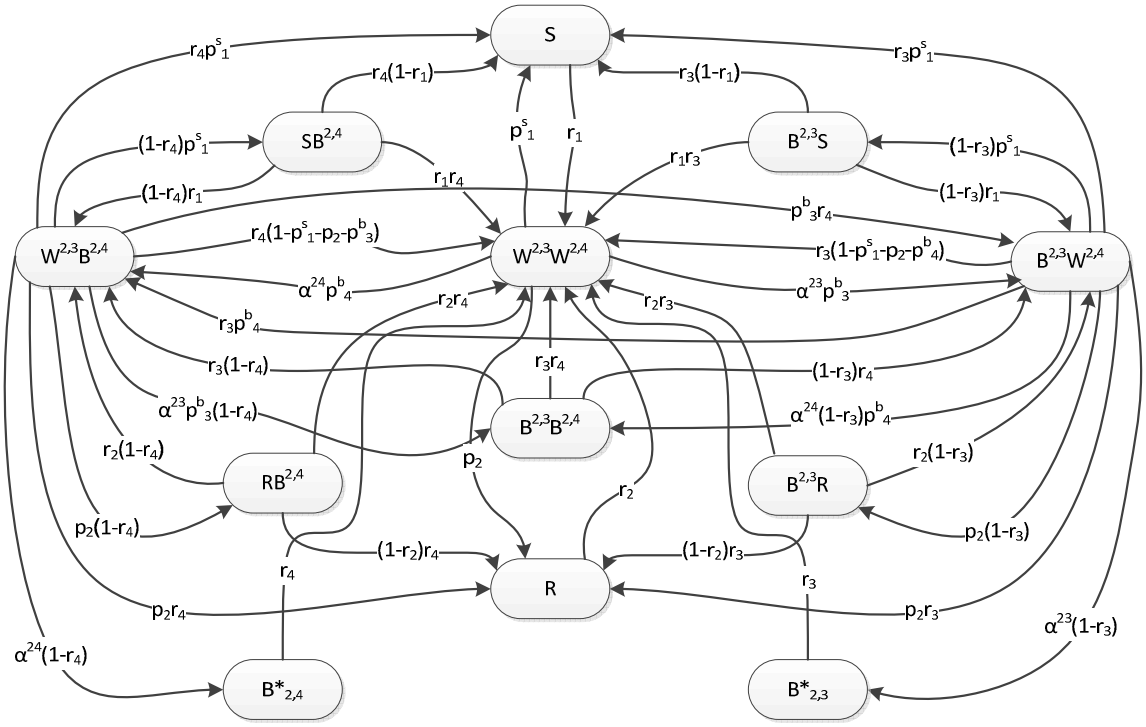


Figure 4-3 Markov chain for M2

The equations for BB24 are very similar to the equations for BB23. For example, equation (4.1) can be used for BB24 by interchanging the superscript 2,3 with 2,4 and vice versa. The aggregation for building blocks BB23 and BB24 are used to derive failure and repair probabilities for each machine to model the impact upstream and downstream building blocks have on each other.

It is worth noting that all 3 building blocks in figure 4-2(b) have machine M2 in it, and each M2 have different failure and repair probabilities that impact the performance of that building block. M2 in BB12 has 2 failure probabilities, p_2 (the local failure for M2) and $p^{d(1,2)}$ (a failure derived in equation (4.7)). The failure $p^{d(1,2)}$ replicates the situation of both downstream buffers (B23 and B24) being full, and the situation that one of the buffers is full and M2 is required to pass product to the buffer due to an α probability value. The matching repair

probability ($r^{d(1,2)}$) is derived similarly in equation (4.8) except the denominator for the equation is all the states where BB12 is blocked. Equations (4.7) and (4.8) differ from the unrestrictive split equations derived in [8] because of the possibility that BB12 can be blocked even though both downstream buffers are not full. Note: Since equations (4.7) and (4.8) are so long, we shorten them by creating equations (4.5) and (4.6) and making these variables *g* and *h*, respectively, and include the variables in the equations for (4.7) and (4.8).

$$g = p^{b}_{3} * \alpha^{23} * (1 - r_4) * \pi(W^{2,3}B^{2,4}) + p^{b}_{4} * \alpha^{24} * (1 - r_3) * \pi(W^{2,4}B^{2,3}) \tag{4.5}$$

$$h = \alpha^{24} * (1 - r_4) * \pi(W^{2,3}B^{2,4}) + \alpha^{23} * (1 - r_3) * \pi(W^{2,4}B^{2,3}) \tag{4.6}$$

$$p^{d(1,2)} = \frac{g + h + (1 - r_4) * \pi(B^{*2,4}) + (1 - r_3) * \pi(B^{*2,3})}{\pi(W^{2,3}W^{2,4}) + \alpha^{23} * \pi(W^{2,3}B^{2,4}) + \alpha^{24} * \pi(W^{2,4}B^{2,3})} \tag{4.7}$$

$$r^{d(1,2)} = \frac{g + h + (1 - r_4) * \pi(B^{*2,4}) + (1 - r_3) * \pi(B^{*2,3})}{\pi(B^{2,3}B^{2,4}) + \pi(B^{*2,4}) + \pi(B^{*2,3})} \tag{4.8}$$

The failure and repair probabilities for BB23 and BB24 are very similar to each other and the derivations for the equations are only different with switching 2,3 to 2,4 and vice versa. Machine M2 in BB23 has 4 failure and repair probabilities. The first failure listed, $p^{u(2,3)}$, is similar to the original local failure probability p_2 except it is calculated using the aggregation required to switch between the Markov chain for M2 to the buffer level decomposition. The equation for it is shown in equation (4.9). The second failure listed, p^{s}_{3}, is derived similarly to

$p^{u(2,3)}$ except now it is using the aggregated states for being starved by BB12. Equation (4.10) lists the equation to calculate failure p^{s}_{3}.

$$p^{u(2,3)} = \frac{\pi(R^{2,3})}{\pi(W^{2,3})} * r_2 \tag{4.9}$$

$$p^{s}_{3} = \frac{\pi(S^{2,3})}{\pi(W^{2,3})} * r_1 \tag{4.10}$$

The third failure probability listed under M2 in BB23 is the competition failure between the two building blocks. This failure is derived in (Colledani et al., 2005). Equation (4.12) lists how to calculate the competition failure probability and equation (4.13) lists the corresponding repair probability. The competition failure simulates the competition between BB23 and BB24. Since there is a split in the flow at M2, a product can be passed to either B23 or B24 but not to each buffer at the same time; therefore, the two building blocks compete and a pseudo type failure models this dynamic.

$$k = (1 - (p^{s}_{1} + p_2 + p^{b}_{3})) * \alpha^{24} \tag{4.11}$$

$$p^{cf(2,3)} = k * \frac{\alpha^{23} * \pi(W^{2,3}W^{2,4}) + \alpha^{23} * r_4 * \pi(W^{2,3}B^{2,4})}{\alpha^{23} * \pi(W^{2,3}W^{2,4}) + \alpha^{23} * \pi(W^{2,3}B^{2,4})} \tag{4.12}$$

$$r^{cf(2,3)} = \frac{\pi(W^{2,3})}{a24 * \pi(W^{2,3}W^{2,4})} * p^{cf(2,3)} \tag{4.13}$$

The fourth failure listed under M2 in BB23 is used to model the situation of BB24 causing BB23 to be blocked because buffer B24 is full and the product from M2 is required to pass to it. This failure calculation is located in equation (4.15). This failure type is different from the competition failure because with this failure both building blocks are now blocked. With the competition failure, one of the building blocks is blocked but the other one is not.

$$m = p^{b}{}_{4} * \alpha^{24} * \pi(W^{2,4}B^{2,3}) + \alpha^{24} * (1 - r_4) * \pi(W^{2,3}B^{2,4}) \tag{4.14}$$

$$p^{bb(2,3)} = \frac{m + (1 - r_4) * \pi(B^{*2,4})}{\pi(W^{2,4})} \tag{4.15}$$

Other failure probabilities used are equations (4.16)-(4.18). Equation (4.16) simulates the situation of M1 starving M2, equation (4.17) simulates the situation of M3 causing blockage to M2, and equation (4.18) simulates the situation of M4 causing blockage to M2.

$$p^{s}{}_{1} = \frac{P^{s}{}_{1}(1,2)}{E(1,2)} * r_1 \tag{4.16}$$

$$p^{b}{}_{3} = \frac{P^{b}{}_{3}(2,3)}{E(2,3)} * r_3 \tag{4.17}$$

$$p^{b}{}_{4} = \frac{P^{b}{}_{4}(2,4)}{E(2,4)} * r_4 \tag{4.18}$$

4.3.2 Disassembly and split

This section presents the method for analyzing transfer lines where a flexible machine performs a disassembly and split action. The equations derived in this section will use the numbering scheme of figure 4-4 for machines and buffers to clearly explain the developed method.

The dynamics of this system are a little more complex than with just a split routing logic. Now combinations of how different streams affect one another need to be considered. For example, building block BB25 will become blocked if both buffers B23 and B24 are full, or buffer B23 is full and the product leaving machine M2 is required to pass to B23 and vice versa for the situation that B24 is full. Since B25 will need to receive a product every time unit, once B25 becomes full and a product is ready to be passed to it, buffers B23 and B24 cannot receive a product. The problem arises with how to calculate the effect BB25 being full has on BB23 and BB24. A remote probability of failure and repair need to be calculated and localized to BB23 and BB24 to model the effect BB25 has in terms of blocking.

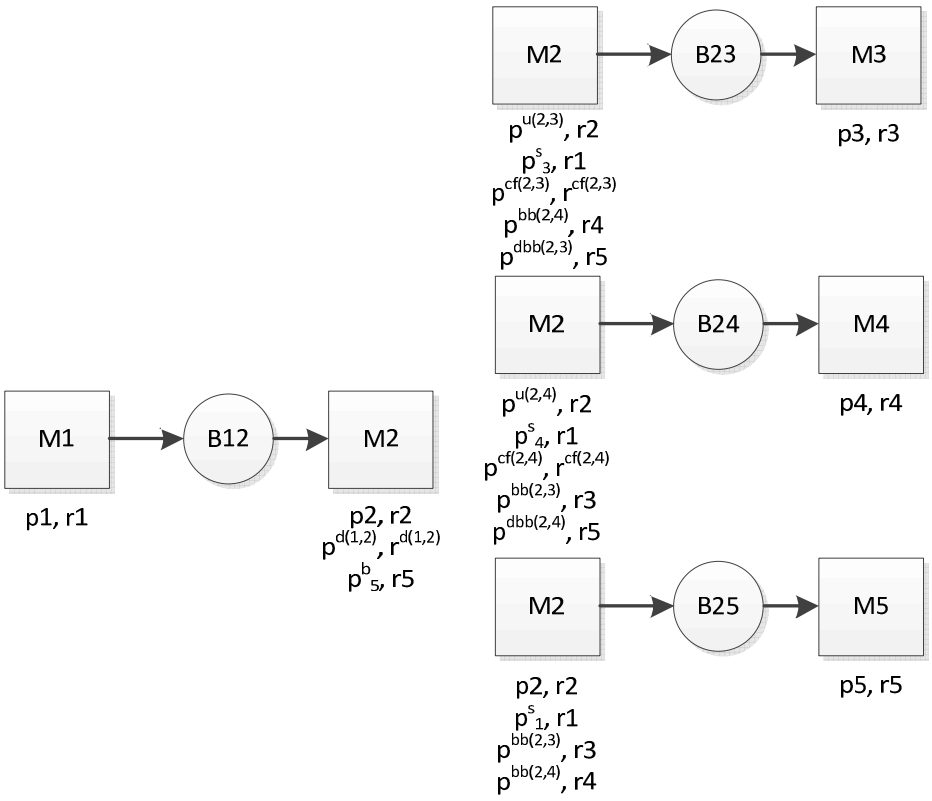


Figure 4-4 Decomposed transfer line for flexible workstation performing disassembly and split

The number of failure and repair probability pairs increases with modeling a flexible workstation performing both disassembly and split but this is expected. There is now an additional building block, BB25, added to the mix that is a disassembly building block. BB12 contains the same failure and repair probability pairs and a third one is listed under M2, p^{b}_{5} calculated by equation (4.19), that simulates M2 in BB12 being blocked if buffer B25 is full. Since BB25 is a disassembly building block, if B25 is full and a product is ready to be passed to it then M2 will be blocked for other downstream buffers.

$$p^{b}_{5} = \frac{P^{b}_{5}(2,5)}{E(2,5)} * r_{5} \tag{4.19}$$

Building blocks BB23 and BB24 have 4 of the same failure and repair probability pairs. The fifth failure probability listed under both M2 for BB23 ($p^{dbb(2,3)}$) and BB24 ($p^{dbb(2,4)}$) simulates the situation discussed previously of how a disassembly building block effects split building blocks. The derivation for these failure probabilities is very similar to equation (4.19) except the α probability values need to be considered. This is very intuitive. For example, buffer B25 becomes full so M2 will not be able to pass product to B23 or B24. But M2 will only pass product to B23 with probability α^{23} and B24 with probability α^{24}. Since for split building blocks M2 can only be working for one or the other, B25 becoming full and blocking M2 will only appear to affect the split building block that is set to receive the product and not the other one. In actuality, both building blocks are impacted and the α probability needs to be included in the calculation.

$$p^{dbb(2,3)} = \frac{P^{b}{}_{5}(2,5)}{\alpha^{23} * E(2,5)} * r_5 \tag{4.20}$$

$$p^{dbb(2,4)} = \frac{P^{b}{}_{5}(2,5)}{\alpha^{24} * E(2,5)} * r_5 \tag{4.21}$$

Building block BB25's machine M2 has 4 failure and repair probability pairs. The first being for local failure, the second for upstream starvation, the third for blockage caused by BB23, and the fourth for blockage caused by BB24.

4.3.3 Algorithm and convergence

The unknown failure and repair probabilities for the decomposed transfer line, such as $p^{d(1,2)}$, p^{b}_{3}, etc., can be determined by an iterative algorithm that is a modification of the DDX algorithm. The DDX algorithm (Dallery et al., 1988) is used to evaluate the performance of long transfer lines. A strength of the developed methods in this paper are they use an existing modified DDX algorithm presented by (Colledani et al., 2005).

The algorithm has 3 steps.

- The first step is initialization of unknown failure and repair probabilities. Each unknown failure probability is initialized to a small value, such as 0.05, and unknown repair probabilities to 0.8.
- Step 2 of the algorithm is evaluation of the performance of each building block. Evaluation begins with the most upstream building block and moves downstream to subsequent building blocks. Failure and repair probabilities can be estimated based on the performance results from each building block and are continuously updated.
- Step 3 is repeating this process of moving up and down the decomposed transfer line on a building block by building block basis, updating failure and repair probabilities. Once the percent difference in throughput for the line after each iteration is less than some value ε, the algorithm terminates. Average throughput and total WIP can be estimated from this algorithm.

The modified DDX algorithm used in (Colledani et al., 2005) is shown to be accurate and converge by running many different scenarios and comparing the analytical results to simulation results. Since we add to the modified DDX algorithm with our newly derived equations, we present just a few scenarios to show the accuracy of our methods.

4.4 Numerical results and discussion

4.4.1 Restrictive split

To evaluate the accuracy of our proposed methods, a simulation model was created using ARENA and MATLAB was used for the analytical modeling techniques. Table 4-1 contains parameter values for 3 scenarios. The input parameters are the failure and repair probability pairs for each machine, the split percentage for M2, as well as the capacity for each buffer. Scenario 1 has all the machines have relatively the same machine efficiency, between 0.86-0.89, and α^{23}=.60 and α^{24}=.40. Machine efficiency is defined as $r_i/(p_i+r_i)$ and is a measure of what percent of the time the machine is available to work. Scenario 2 has all the same parameter values as scenario 1 except M3 has a machine efficiency of 0.63. This means M2 will 63% of the time pass material downstream to B23, the buffer that feeds M3, and the rest of the time it will be in a failed state. Scenario 3 is the same as scenario 2 except α^{23}=.75 and α^{24}=.25. Table 4-2 contains the comparison for the simulation and analytical model results for throughput and average WIP levels for each machine and buffer. The percent difference is calculated as the absolute value difference between the analytical and simulation model divided by the results from the simulation model.

Table 4-1 Values for restrictive split model

Machine (Mi) Buffer (Bij)	Failure Prob. (pi) Repair Prob. (ri) Split Prob. (α^{ij}) Buffer Capac. (n_{ij})	Scenario #		
		1	2	3
M1	p1	0.08	0.08	0.08
	r1	0.68	0.68	0.68
B12	n_{12}	10	10	10
M2	p2	0.1	0.1	0.1
	r2	0.75	0.75	0.75
B23	α^{23}	0.6	0.6	0.75
	n_{23}	5	5	5
M3	p3	0.15	0.3	0.3
	r3	0.9	0.5	0.5
B24	α^{24}	0.4	0.4	0.25
	n_{24}	5	5	5
M4	p4	0.12	0.12	0.12
	r4	0.8	0.8	0.8

Table 4-2 Restrictive split comparison results

Scenario	Performance Measure	Analytical Model	Simulation Model	% Difference
1	Throughput	0.870	0.874	0.50%
	Total WIP	6.898	6.886	0.17%
2	Throughput	0.855	0.856	0.09%
	Total WIP	8.750	8.804	0.61%
3	Throughput	0.810	0.806	0.52%
	Total WIP	10.712	10.762	0.46%

4.4.2 Disassembly and split together

Table 4-3 contains parameter values for two scenarios. The input parameters are the failure and repair probability pairs for each machine, the split percentage for M2, as well as the

capacity for each buffer. Scenario 1 for the disassembly and split model is very similar to scenario 2 for the restricted split model, except now there is an additional buffer (B25) and an additional machine (m5). M5 is a disassembly stream and M5 has a machine efficiency of 0.85, very similar to the machine efficiencies of M1, M2, and M4. Scenario 2 is the same as scenario 1 except the machine efficiency of M5 is 0.70. Table 4-4 contains the comparison for the simulation and analytical model results for throughput and average WIP levels for each machine and buffer.

Table 4-3 Input values for disassembly and split model

Machine (Mi) Buffer (Bij)	**Failure Prob. (pi) Repair Prob. (ri) Split Prob. (α^{ij}) Buffer Capac. (n_{ij})**	Scenario #	
		1	**2**
M1	p1	0.08	0.08
	r1	0.68	0.68
B12	n_{12}	10	10
M2	p2	0.1	0.1
	r2	0.75	0.75
B23	α^{23}	0.6	0.6
	n_{23}	5	5
M3	p3	0.3	0.3
	r3	0.5	0.5
B24	α^{24}	0.4	0.4
	n_{24}	5	5
M4	p4	0.12	0.12
	r4	0.8	0.8
B25	n_{25}	10	10
M5	p5	0.15	0.3
	r5	0.85	0.7

Table 4-4 Disassembly/split comparison results

Scenario	Performance Measure	Analytical Model	Simulation Model	% Difference
1	Throughput	0.832	0.837	0.55%
	Total WIP	14.85	15.09	1.58%
2	Throughput	0.708	0.702	0.93%
	Total WIP	18.93	19.06	0.65%

4.4.3 Discussion

The analytical modeling results for both models matched the simulation results very well for each scenario. The percent difference errors for throughput are consistently below 1% and below 2% for total WIP. The scenarios for both the restricted split and disassembly and split together models reveal some interesting results. For example, scenario 2 for the disassembly and split model shows that the disassembly stream (M5) can become the limiting machine for the transfer line if its efficiency goes below a certain threshold compared to the other machines in the line. For the restricted split model, the only parameter difference between scenario 2 and 3 is the α split values but there is a difference in transfer line throughput and total WIP.

4.5 Conclusions

This paper presented an approximate analytical method for modeling both restrictive split routing logic and flexible machines performing multiple types of tasks, specifically disassembly and split together. The developed method for modeling both restrictive split routing logic and disassembly and split together routing logic is very accurate and this is demonstrated numerically by comparing the analytical results with a simulation model. Our method is approximated using a multiple failure discrete time model. One of the strengths of our method is it can be nested into

the existing literature of discrete time model decomposition methods, such as the modified DDX algorithm in (Colledani et al., 2005), for evaluating the performance of long transfer lines.

This paper presented a method for disassembly and restrictive split together but this method can be extended to transfer lines with disassembly and unrestrictive split routing logic as well. The difference between the two methods would be the Markov chain used to model the machine where the split takes place will be different based on the routing logic, and this in turn will affect the calculation for the various failure and repair probabilities based on the particular Markov chain used. One weakness for modeling a split using a Markov chain is that if the split machine has more than 2 downstream buffers, then a new Markov chain needs to be created and the amount of states needed in the Markov chain will increase dramatically. Future work is needed to model complex routing logic for continuous time models to continue to bridge the gap between the disadvantages of analytical modeling techniques and the advantages of simulation.

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

5.1 Summary and conclusions

This dissertation focuses on modeling and analyzing disassembly systems under a high variety of EOL states. The chapters within the dissertation studied three adjacent levels of aggregation for disassembly modeling, chapter 2 focused on sequence generation, chapter 3 focused on line balancing, and chapter 4 focused on throughput modeling. Although each chapter focused on a different portion of disassembly modeling, each chapter was concerned with providing algorithms and methods to support informed disassembly decision making at the line and system level.

The major achievements of this dissertation can be summarized as follows:

1. The development of disassembly sequence generation models that considers material properties, partial disassembly, and sequence dependent task times that will determine the optimal order of disassembly in the presence of a high variety of EOL states.
2. The development of a method for disassembly line balancing that gives full consideration to the high quality variation of EOL returns of cores.

3. Further advancement of the analytical modeling framework to analyze transfer lines that perform restrictive split routing logic and disassembly and split operations at the same workstation.

5.2 Proposed future work

The future work for this dissertation if focused on two aspects, (1) bring cohesion and unity to research areas that are usually treated separately, and (2) add more uncertainty into the sequence generation and line balancing methods.

- Traditionally, sequence generation and line balancing are treated as two separate research areas; however, the two areas are highly related where sequence generation focuses on finding the optimal sequence at the product level while line balancing seeks to find the optimal sequence that will be assigned to workstations at the system level. A method is needed to bridge this gap where the focus is still on return of cores that have a high variety of EOL states and partial disassembly and sequence dependent task times are still considered.
- Chapters 2 and 3 make the assumption that when cores are returned they can be sampled and therefor the percentages for each EOL state can be known and the disassembly times for each operation found to be either deterministic or a normal variable. In all practicality, this is very difficult in the real world and a method is needed to determine the best sequence and line balance if a product has EOL states with unknown percentages and disassembly removal times that are not deterministic and/or unknown.
- In chapter 4, we used a discrete time Markov chain to model a restrictive split routing logic, but there is a large research area that uses continuous time Markov chains to

formulate transfer lines, including already built software that can analyze transfer lines using a continuous time Markov chain framework, PAMS. A method needs to be developed than can analyze both restrictive split routing logic and multiple routing operations at the same workstation (such as split and disassembly) using the continuous time Markov chain framework. Once these methods are developed, it can be incorporated into PAMS.

BIBLIOGRAPHY

Agrawal S and MK Tiwari, 2008. A collaborative ant colony algorithm to stochastic mixed-model U-shaped disassembly line balancing and sequencing problem, *International Journal of Production Research*. 46(6): p. 1405-1429.

Altekin FT and C Akkan, 2012. Task-failure-driven rebalancing of disassembly lines, *International Journal of Production Research*. 50(18): p. 4955-4976.

Altekin FT, L Kandiller, and NE Ozdemirel, 2004. Disassembly line balancing with limited supply and subassembly availability, *Proceedings of SPIE - The International Society for Optical Engineering*. 5262: p. 59-70.

Altekin FT, L Kandiller, and NE Ozdemirel, 2008. Profit-oriented disassembly-line balancing, *International Journal of Production Research*. 46(10): p. 2675-2693.

Aydemir-Karadag A and O Turkbey, 2013. Multi-objective optimization of stochastic disassembly line balancing with station paralleling, *Computers and Industrial Engineering*. 65: p. 413-425.

Banks J, JS Carson, BL Nelson, and DM Nicole, 2009. *Discrete-Event System Simulation*, 5th ed. Prentice Hall.

Battaïa O and A Dolgu, 2013. A taxonomy of line balancing problems and their solution approaches, *International Journal of Production Economics*. 142: p. 259-277.

Baybars I, 1986. A survey of exact algorithms for the simple assembly line balancing problem, *Management Science*. 32(8): p. 909-932.

Boysen N, M Fliedner, A and Scholl, 2009. Assembly line balancing: Joint precedence graphs under high product variety, *IIE Transactions*. 41(3): p. 183-193.

Chiang SY, CT Kuo, and SM Meerkov, 1998. Bottlenecks in Markovian Production Lines: A Systems Approach, *IEEE Transactions on Robotics and Automation*. 14(2): p. 352-359.

Colledani M, A Matta, and T Tolio, 2005. Performance evaluation of production lines with finite buffer capacity producing two different products, *OR Spectrum*. 27(2-3): p. 243-263.

Colledani M, F Gandola, A Matta, and T Tolio, 2008. Performance evaluation of linear and non-linear multi-product multi-stage lines with unreliable machines and finite homogeneous buffers, *IIE Transactions*. 40(6): p. 612-626.

Dallery Y, R David, and XL Xie, 1988. An efficient algorithm for analyses of transfer lines with unreliable machines and finite buffers, *IIE Transactions*. 20(3): p. 280-283.

De Fazio TL and DE Whitney, 1987. Simplified generation of all mechanical assembly sequences, *IEEE Journal of Robotics and Automation*. 3(6): p. 640–658.

Edmunds R, M Kobayashi, and M Higashi, 2012. Using constraint-satisfaction to optimize disassembly sequence generated from AND/OR information, *International Journal of Production Research*. 50(15): p. 4105-4126.

Errington M and SJ Childe, 2013. A business process model of inspection in remanufacturing, *Journal of Remanufacturing*. 3(7).

Feit EM and SD Wu, 2000. Transfer Line Design with Uncertain Machine Performance Information, *IEEE Transactions on Robotics and Automation*. 16(5): p. 581-587.

Gungor A and SM Gupta, 2001. A solution approach to the disassembly line balancing problem in the presence of task failures. *International Journal of Production Research*. 39: p. 1427–1467.

Gungor A, SM Gupta, K Pochampally, and SV Kamarthi, 2001. Complications in disassembly line balancing, *Proceedings of SPIE – The International Society for Optimal Engineering*. 4193: p. 289-298.

Hu SJ, J Ko, L Weyand, HA ElMaraghy, TK Lien, Y Koren, H Bley, G Chryssolouris, N Nasr, and M Shpitalni, 2011. Assembly system design and operations for product variety, *CIRP Annals – Manufacturing Technology*. 60(2): p. 715-733.

Ilgin MA and SM Gupta, 2010. Environmentally conscious manufacturing and product recovery (ECMPRO): A review of the state of the art, *Journal of Environmental Management*. 91(3): p. 563-591.

Jacobs DA and SM Meerkov, 1995. A system-theoretic property of serial production lines: improvability, *International Journal of System Science*. 26: p. 95–137.

Kang MK, MJ Kwak, NW Cho, and YS Hong, 2010. Automatic derivation of transition matrix for end-of-life decision making, *International Journal of Production Research*. 48(11): p. 3269-3298.

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

Lambert AJD, 2003. Disassembly sequencing: a survey, *International Journal of Production Research*. 41(16): p. 3721-3759.

Li J and N Huang, 2005. Modeling and analysis of a multiple product manufacturing system with split and merge, *Int. J. Prod. Res*. 43(19): p. 4049–406.

Li J, DE Blumenfeld, N Huang, and JM Alden, 2009. Throughput analysis of productions systems: recent advances and future topics, *Int. J. Prod. Res.* 41(14): p. 3823-3851.

Liu Y and J Li, 2010. Split and merge production systems: performance analysis and structural properties, *IIE Transactions*. 42(6): p. 422 – 434.

Lund RT, 1984. Remanufacturing, *Technology Review.* 87(2): p. 18-23.

Lund RT, 1996. The Remanufacturing Industry: Hidden Giant. Boston, Massachusetts: Boston University.

Ma YS, HB Jun, HW Kim, and DH Lee, 2011. Disassembly process planning algorithms for end-of-life product recovery and environmentally conscious disposal, *International Journal of Production Research.* 49(23): p. 7007-7027.

McGovern SM and SM Gupta SM, 2011. The Disassembly Line: Balancing and Modeling, McGraw Hill, New York.

McGovern SM and SM Gupta, 2007. A balancing method and genetic algorithm for disassembly line balancing, *European Journal of Operations Research*. 179: p. 692-708.

Meacham A, R Uzsoy, and U Venkatadri, 1999. Optimal disassembly configurations for single and multiple products, *Journal of Manufacturing Systems.* 18(5): p. 311-322.

Ozdemir O, M Denizel, and VDR Guide, 2012. Recovery decisions of a producer in a legislative disposal fee environment, *European Journal of Operations Research*. 216: p. 293-300.

Rickli JL and JA Cameilio, 2013. Multi-objective partial disassembly optimization based on sequence feasibility, *Journal of Manufacturing Systems*. 32(1): p. 281-293.

Riggs RJ and SJ Hu, 2013. Disassembly liaison graphs inspired by word clouds, *Procedia CIRP*. 7: p. 521-526.

SB Gershwin, 1994. *Manufacturing Systems Engineering*. Prentice Hall.

SJE Taylor, L Behli, X Wang, SJ Turner, and J Ladbrook, 2005. Investigating distributed simulation at the Ford Motor Company, *Ninth IEEE International Symposium on Distributed Simulation and Real-Time Applications*. p. 139–147.

Smith SS and WH Chen, 2011. Rule-based recursive selective disassembly sequence planning for green design, *Advanced Engineering Informatics*. 25: p. 77-87.

Tang Y and M Zhou, 2006. A systematic approach to design and operation of disassembly lines, *IEEE Transactions on Automation Science and Engineering*. 3(3): p. 324-330.

Tang Y, M Zhou, E Zussman, and R Caudill, 2002. Disassembly modeling, planning, and application, *Journal of Manufacturing Systems*. 21(3): p. 200-217.

Tian G, MC Zhou, J Chu, and Y Liu, 2012. Probability evaluation models of product disassembly cost subject to random removal time and different removal labor cost. *IEEE Transactions on Automation Science and Engineering*. 9(2): p. 288-295.

Tian G, Y Liu, Q Tian, and J Chu, 2012. Evaluation model and algorithm of product disassembly process with stochastic feature, *Clean Techn Environ Policy*. 14: p. 345-356.

Tolio T, SB Gershwin, and A Matta, 2002. Analysis of two-machine lines with multiple failure modes, *IIE Transactions*. 34(1): p. 51–62.

Tripathi M, S Agrawal, MK Pandey, R Shankar, and MK Tiwari, 2009. Real world disassembly modeling and sequencing problem: Optimization by Algorithm of Self-Guided Ants (ASGA), *Robotics and Computer-Integrated Manufacturing*. 25: p. 483-496.

Tuncel E, A Zeld, and S Kamarthi S, 2012. Solving large scale disassembly line balancing problem with uncertainty using reinforcement learning, *Journal of Intelligent Manufacturing*. p. 1-13.