Dynamic Sequencing of Jobs on Conveyor Systems for Minimizing Changeovers

A Thesis
Presented to
The Academic Faculty

By

Yong-Hee Han

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Industrial and Systems Engineering

Georgia Institute of Technology
November 2004

Dynamic Sequencing of Jobs on Conveyor Systems for Minimizing Changeovers

November 18, 2004

To my family:

My mother, Jung Hwan Kim

My wife, Hyun Jin Lee

For their sacrifice and love.

ACKNOWLEDGMENT

I cannot express with words my deep appreciation and gratitude to my advisor, Dr. Chen Zhou – his unreserved help and advice was crucial for me to complete my Ph. D. program successfully. I also thank Dr. Leon F. McGinnis, Dr. Gunter P. Sharp, Dr. Joel Sokol, and Dr. Bert Bras for serving on my committee and for their constructive comments.

Specifically, I wish to thank Dr. Bras, Dr. McGinnis, and Dr. Sokol. Dr. Bras generously provided me help and guidance throughout my long years in the Environmentally Conscious Design and Manufacturing (ECDM) laboratory. Dr. McGinnis always gave me unreserved help in seeing the core of the problems encountered in all research projects I conducted in the ECDM laboratory, and helped me to go into the right direction. Dr. Sokol provided me with helpful resources and directions in my research topic (color changeover reduction problem in the paint shop).

Thanks also go to Carol Carmichael, P. J. Newcomb, and Tina Guldberg in the ECDM laboratory for their kind help. This dissertation was also supported by the National Science Foundation under Grant No. DMI-0085253.

Finally, the only regret I have is that my mother, who gave up virtually everything for her son, passed away recently, leaving him no chance to pay back any of the enormous debt he owes to her.

Mom, I love you.

TABLE OF CONTENTS

ACKNOWLEDGMENT iv

LIST OF TABLES xi

LIST OF FIGURES ix

LIST OF FIGURES ix

LIST OF SYMBOLS OR ABBREVIATIONS xi

CHAPTER 1 INTRODUCTION 1

1.1 Motivation 1

1.2 Problem Statement 5

CHAPTER 2 LITERATURE REVIEW 7

CHAPTER 3 ANALYTICAL APPROACH 16

3.1 Models for Diverging Junctions 16

3.1.1 Constant Setup Cost with a Fixed Number of Downstream Queues 16

3.1.2 Attribute Dependent Setup Cost with a Variable Number of Downstream Queues 22

3.1.2.1 Introduction 22

3.1.2.2 Installation Cost Calculation 25

3.1.2.3 Model Formulation 33

3.2 Models for an Off-line Buffer 35

3.2.1 Generalized Sequence-dependent Setup Cost 35

3.2.2 Basic Sequence-dependent Setup Cost 48

3.3 Model Customization 49

3.3.1 Generalization of the Model in Section 3.1.2 50

CHAPTER 4 NUMERICAL RESULTS 51

4.1 Sensitivity Analysis on the Models 52

4.2 Solution Time 56

4.3 Solution Quality of the Heuristic Algorithm 59

4.4 Results of the Case Study Data 63

CHAPTER 5 CASE STUDY 65

5.1 Paint Purge Reduction in Automotive Assembly 65

5.2 Simulation Study 68

5.2.1 Need for Simulation Modeling 68

5.2.2 Simulation Model Details 70

5.2.2.1 Input Data 70

5.2.2.2 Simulation Model Language Selection 71

5.3 Proposed Changes and Effect 73

CHAPTER 6 CONTRIBUTIONS AND FUTURE DIRECTIONS 75

6.1 Contributions 75

6.2 Future Research 76

APPENDIX A: MODELS FOR CONSTRAINED SEQUENCING & ATTRIBUTE ASSIGNING PROBLEM 78

APPENDIX B: MODEL FOR DIVERGING JUNCTION & OFF-LINE BUFFER 85

APPENDIX C: PROOF of THEOREM 2.2 88

BIBLIOGRAPHY 90

VITA 95

LIST OF TABLES

Table 1. Classification of Arcs in Shortest Path Problem Network 44

Table 2. Numerical Experiments Results for the Analytical Model (in # of Changeovers) 53

Table 3. CPLEX Solution Times (in seconds) 57

Table 4. Numerical Experiments Results for the Heuristic Algorithm (in # of Changeovers) 61

Table 5. Results of Benchmarking Heuristic Algorithm-based Solutions against Optimal Solutions (in Percentage) 62

LIST OF FIGURES

Figure 1. Diverging Conveyors with an Off-line Buffer Example....................................3

Figure 2. Converging Conveyors with an Off-line Buffer Example 3

Figure 3. Multiple Incoming and Outgoing Conveyors with an Off-line Buffer 4

Figure 4. the Rail Classification Problem as a Special Case of the Constrained Sequencing Problem 9

Figure 5. Lane Assignment Problem as a Special Case of the Constrained Sequencing Problem 12

Figure 6. A Possible Dispatching Result for the Example in Figure 1............................18

Figure 7. Network Representation Associated with Figure 6 18

Figure 8. Assignment Problem Example of 3 Jobs and 2 Queues 21

Figure 9. Part of the Network Representation for the Model in Section 4.1.1 24

Figure 10. Extension the Network Representation in Figure 8.......................................25

Figure 11. Network Representation Associated with Figure 6: Revisited.......................27

Figure 12. Example of Case 2 29

Figure 13. Example of Case 3 30

Figure 14. Example of Case 4 32

Figure 15. Network Representation Example of Constrained Sequencing Problem with an Off-line Buffer of Capacity 1 37

Figure 16. Simple Example of Shortest Path Problem Network.....................................45

Figure 17. Relationship between # of Queues and # of Changeovers.............................54

Figure 18. Relationship between # of Queues and # of Reduced Changeovers 55

Figure 19. Relationship between # of Attributes and # of Changeovers..........................56

Figure 20. Relationship between # of Attributes and Solution Time 58

Figure 21. Relationship between # of Jobs and Solution Time 59

Figure 22. Relationship Between # of Queues and Solution Time 59

Figure 23. Impact of # of Queues on # of Reduced Changeovers (Case Study Data) 64

Figure 24. Simulation Model Screenshot - Prime Storage Area......................................72

Figure 25. Simulation Model Screenshot - Prime Scuff Area 72

Figure 26. Simulation Run Results...74

Figure 27. Network Relationship among Redefined Variables 79

Figure 28. Relationship between Off-line Buffer Model and Diverging Queues Model ..86

LIST OF SYMBOLS OR ABBREVIATIONS

CONSTANTS

$U =$ upstream queue

$D =$ downstream queue (for the case of one downstream queue)

$Q =$ number of downstream queues

$N =$ number of jobs in U

$K =$ number of attributes

$B =$ number of slots in an offline buffer

$C =$ changeover cost, a constant for each changeover

$$\bar{C}_{ij} = \begin{cases} C & \text{if attribute of the } i^{\text{th}} \text{ job in } U \text{ is different from attribute of the } j^{\text{th}} \text{ job in } U \\ 0 & \text{otherwise} \end{cases}$$

$C'_{ij} =$ changeover cost when the i^{th} job in U is located right before the j^{th} job in U on a downstream queue

(for the case that a specific attribute is given to each job in U and it is known in advance)

$C_k =$ changeover cost from attribute k

$C''_q =$ cost of installing downstream queue q

$$A'_{ik} = \begin{cases} 1 & \text{if attribute } k \text{ can be assigned to the } i^{\text{th}} \text{ job in } U \\ 0 & \text{otherwise} \end{cases}$$

$$A_{ik} = \begin{cases} 1 & \text{if attribute } k \text{ is assigned to the } i^{\text{th}} \text{ job in } U \\ 0 & \text{otherwise} \end{cases}$$

VARIABLES

$$x_{ij} = \begin{cases} 1 & \text{if the } i^{\text{th}} \text{ job in } U \text{ is located right before the j}^{\text{th}} \text{ job in } U \text{ on a downstream queue} \\ 0 & \text{otherwise} \end{cases}$$

$$x_{ijk} = \begin{cases} 1 & \text{if the } i^{\text{th}} \text{ job with attribute } k \text{ in } U \text{ is sent to the } j^{\text{th}} \text{ position in } D \\ 0 & \text{otherwise} \end{cases}$$

$$x_{ijkl} = \begin{cases} 1 & \text{if the } i^{\text{th}} \text{ job with attribute } k \text{ in } U \text{ is located right before the } j^{\text{th}} \text{ job} \\ & \text{with attribute } l \text{ in } U \text{ on a downstream queue} \\ 0 & \text{otherwise} \end{cases}$$

$$x_{hijk} = \begin{cases} 1 & \text{if the } h^{\text{th}} \text{ job with attribute } k \text{ in } U \text{ is located right before the } j^{\text{th}} \text{ job in } U \text{ on a} \\ & \text{downstream queue (index } i \text{ represents temporary position in offline buffer)} \\ 0 & \text{otherwise} \end{cases}$$

$$y_{iq} = \begin{cases} 1 & \text{if the } i^{\text{th}} \text{ job in } U \text{ is the last item to be sent to a downstream queue} \\ 0 & \text{otherwise} \end{cases}$$

$$y_{ikq} = \begin{cases} 1 & \text{if the } i^{\text{th}} \text{ job with attribute } k \text{ in } U \text{ is the last item to be sent to a downstream queue} \\ 0 & \text{otherwise} \end{cases}$$

$$z_{qj} = \begin{cases} 1 & \text{if the } i^{\text{th}} \text{ job in } U \text{ is the first item to be sent to a downstream queue} \\ 0 & \text{otherwise} \end{cases}$$

$$z_{qjl} = \begin{cases} 1 & \text{if the } j^{\text{th}} \text{ job with attribute } l \text{ in } U \text{ is the first item to be sent to a downstream queue} \\ 0 & \text{otherwise} \end{cases}$$

w_{ij} = variable for 'unused' downstream queue (refer to section 4.1.2 for details)

$$t_{ijk} = \begin{cases} 1 & \text{if the } i^{\text{th}} \text{ job with attribute } k \text{ in } U \text{ is located right before the } j^{\text{th}} \text{ item} \\ & \text{(attribute of the } j^{\text{th}} \text{ item is not } k\text{) in } U \text{ on a downstream queue} \\ 0 & \text{otherwise} \end{cases}$$

$$v_{jkl} = \begin{cases} 1 & \text{if the } j^{\text{th}} \text{ job in } D \text{ has attribute } k \text{ while the } j+1^{\text{th}} \text{ job in } D \text{ has attribute } l \\ 0 & \text{otherwise} \end{cases}$$

$$u_{jk} = \begin{cases} 1 & \text{if the } j^{\text{th}} \text{ job in } D \text{ has attribute } k \text{ while the } j+1^{\text{th}} \text{ job in } D \text{ does not have attribute } k \\ 0 & \text{otherwise} \end{cases}$$

SUMMARY

This research investigates the problem of constrained sequencing of a set of jobs on a conveyor system with the objective of minimizing setup cost. A setup cost is associated with extra material, labor, or energy required due to the change of attributes in consecutive jobs at processing stations. A finite set of attributes is considered in this research. Sequencing is constrained by the availability of two elements – storage buffers and conveyor junctions. The problem is motivated by the paint purge reduction problem at a major U.S. automotive manufacturer. First, a diverging junction with a sequence-independent setup cost and predefined attributes is modeled as an assignment problem and this model is extended by relaxing the initial assumptions in various ways. We also model the constrained sequencing problem with an off-line buffer and develop heuristics for efficiently getting a good quality solution by exploiting the special problem structure. Finally, we conduct sensitivity analysis using numerical experiments, explain the case study, and discuss the use of the simulation model as a supplementary tool for analyzing the constrained sequencing problem.

CHAPTER 1

INTRODUCTION

1.1 Motivation

Information technology and computerized automation have greatly improved flexibility in modern manufacturing. Today, most high-volume production systems may appear to be old fashioned transfer lines but in fact have become highly flexible, producing a large family of products such as electronics, automobiles, and other consumer goods. One objective of striving for flexibility is to reduce the setup cost or time-to-respond to the ever-increasing diversity of customer demands. However, even the most flexible systems may still incur some setup cost in job changes. It is often desirable to change the job sequence to further reduce the setup cost and time.

Conveyors are the most popular material transfer mechanism in high-volume production. Conveyors can transfer large amounts of material with simple motion control and also provide buffer space. However, simple conveyor segments are usually constrained to operate in a First-In-First-Out (FIFO) principle. The sequence of materials on a simple conveyor segment cannot be changed by the conveyor itself. To change the sequence in a conveyor system, one needs special mechanisms such as bypass, transfer, and spur. The use of special mechanisms costs money and takes up floor space, especially when transporting large jobs, making it important to minimize their use and to maximize their utilization in operation.

However, merging or diverging conveyor junction points or off-line buffers (a conveyor itself can be considered as an on-line buffer) can also be used to change the sequence. Junction points and off-line buffers are frequently observed in manufacturing facilities and changing the control logic of such equipment is relatively inexpensive. Therefore, using junction points or off-line buffers is preferred to using special mechanisms because of the reduced initial investment cost and floor space usage.

Use of junction points and off-line buffers to change the job sequence can be found in the paint shop operation of automobile manufacturing where reducing the number of car color changes is desired. Figure 1 shows a diverging junction with an off-line buffer where a single upstream conveyor feeds a finite sequence of cars into two downstream conveyors that lead to the paint booth. Each car at the end of the upstream conveyor is allowed to visit off-line buffer before it is fed to the downstream conveyors. It is desirable that color changes are minimized in the downstream conveyors. In Figure 1, the number of setups to paint the five cars can vary from zero to four, depending on the dispatch sequence at the diverging point. Note that the minimum number of setups becomes two if an off-line buffer does not exist.

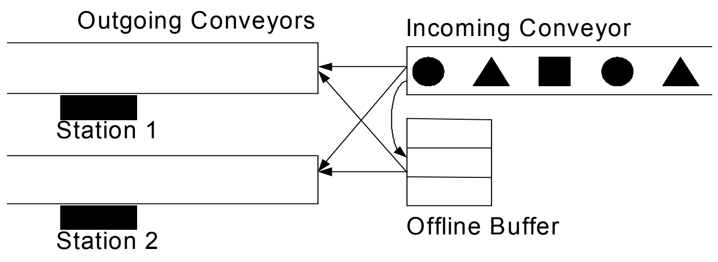

Figure 1. Diverging Conveyors with an Off-line Buffer Example

Figure 2 is an example of a converging junction with an off-line buffer where two upstream conveyors send a finite sequence of cars into a single downstream conveyor. In Figure 2, the number of setups can be two to four. Note that the minimum number of setups is three if no off-line buffer is available.

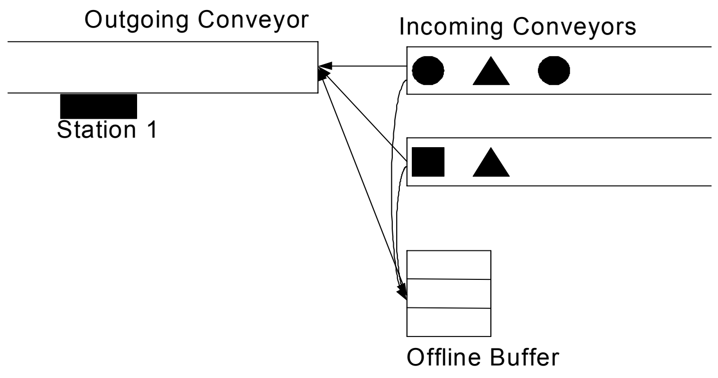

Figure 2. Converging Conveyors with an Off-line Buffer Example

Our sequencing problem can be defined by the following two sets of elements. First, the design parameters include:

1) the number of upstream and downstream queues at the junction,
2) the capacity of the upstream (and downstream) queues,
3) the capacity of the off-line buffer at the junction—if one exists,

4) the discipline supported by the upstream queues, the downstream queues, and the off-line buffer, respectively,
5) the configuration of the junction.

Second, the known operational parameters include:

1) the attributes of the jobs in each queue,
2) the setup cost between consecutive jobs in the downstream queue.

The operational decision is the dispatch sequence at the junction. The setting of multiple upstream conveyors and multiple downstream conveyors—with or without storage buffers—is commonly observed in many real-world manufacturing environments, including the case study we conducted, as shown in Figure 3. In Figure 3, for the car at the end of any upstream conveyor, this car may visit an off-line buffer, or bypass an off-line buffer and go directly to one of the three downstream conveyors. Since the car visiting off-line buffer will be ready to be released to one of downstream queues after some time, based on the queue discipline and transfer time of the off-line buffer, the off-line buffer can be used to further reduce the number of color changes.

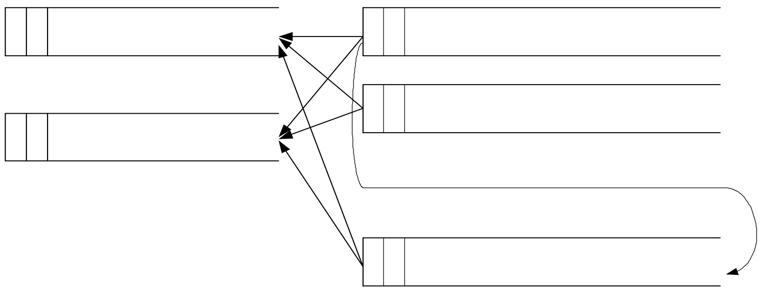

Figure 3. Multiple Incoming and Outgoing Conveyors with an Off-line Buffer

While the optimal solution can be found by observation for the problem of minimizing the number of color changes in Figures 1 and 2, finding the optimal solution becomes very difficult when the number of incoming cars increases in a more complex junction. The objective in production also may include minimizing cycle times or work in process. As can be seen, the minimization of setup alone is rather complex and this research is restricted to part of the conveyor system design and control with the objective of minimizing setup costs. Diverging junctions, off-line buffers, and flexible attribute assignments are investigated while converging junctions are not covered.

1.2 Problem Statement

Consider the problem of constrained sequencing a finite set of jobs on a conveyor system with the objective of minimizing setup cost. A *setup* occurs whenever two consecutive jobs do not share the same attribute at a processing station served by the conveyor system. The conveyor system consists of FIFO conveyor segments and special mechanisms such as junctions and off-line buffers. A *junction* is an interface between M upstream conveyors and N downstream conveyors with or without an off-line buffer. It is assumed that $M = 1$ and $N > 1$ with or without an off-line buffer. Time-based measures such as cycle time are not considered.

The organization of this dissertation is as follows. Relevant literature is reviewed in Chapter 2. Related problems are defined and analytical solution methodology is proposed in Chapter 3. In Chapter 4, a sensitivity analysis is conducted for the models described in Chapter 3 using numerical experiments on randomly generated data as well as case study data. In Chapter 5, details of the case study in an automobile paint shop as well as a discrete event simulation model as a supplementary tool for analyzing the constrained sequencing model is discussed. Our research contributions are summarized and future research directions are discussed in Chapter 6.

CHAPTER 2

LITERATURE REVIEW

Morley and Schelberg (1993), Morley (1996), and Morley and Ekberg (1998) discussed algorithms for assigning trucks to paint booths in a truck facility to minimize total makespan and the number of paint flushes. They applied market-based bidding algorithms to a GM plant and reported a 100% increase of average color block size in their case study. Their heuristic method turned out to have many advantages—easy to implement, robust with respect to schedule changes or machine breakdowns, and effective reduction of paint changeovers. Campos, Bonabeau et al. (2001) compared Morley and Ekberg's market-based approach with ant-inspired response threshold algorithm and used genetic algorithm for getting parameter values for the above two algorithms. Kittithreerapronchai and Anderson (2003) simulated a market-based algorithm as well as an ant-inspired algorithm. They found that some parameters in each algorithm could be removed since they are very insensitive to the objective function value. All the above approaches use artificial intelligence (AI) techniques to tackle the constrained sequencing problem. These AI approaches are easy to implement and robust to system disruptions such as paint booth breakdowns.

Atassi (1996) proposed the use of temporary re-sequencing, facilitated by an automated storage and retrieval system (AS/RS). The AS/RS acts as a buffer that can store cars before and after painting. Using this buffer, a plant can perturb the order for painting cars to create larger paint blocks and then restore the original sequence after painting. Myron (1996) examined the effect of forming large blocks of cars with the same color at an automotive assembly plant. Using discrete event simulation, he showed that a simple block protection rule could significantly reduce setup cost when it is coupled with pre- and post-sequencing using a fully flexible AS/RS.

However, AI or simulation approaches have the drawback of an inability to provide any optimality guarantee or upper or lower bound. In optimization modeling approaches, Choe, Sharp et al. (1993) was the first to model the constrained sequencing problem as an optimization problem and to get a upper bound. He used an AS/RS to increase the size of paint blocks while maintaining a workload-balanced vehicle sequence. More specifically, the problem is how to perturb the original car flow around the vehicle painting station to reduce color setup with the constraint of not violating maximum allowable deviation from the original sequence. He modeled the problem as a traveling salesman problem with time windows, and succeeded in reducing the model to a manageable size and getting very tight bounds—empirically within 2.5% of optimality—by exploiting the special problem structure. He also discussed various relaxations of the problems for getting a near-optimal solution within a reasonable time. To our knowledge, he was the first to model the color change reduction problem using an optimization formulation. However, his model is different from the models in this research in that he used an AS/RS while we use diverging conveyors as well as off-line buffer to re-sequence the incoming cars.

Similar setup reduction problem with constrained sequencing occurs in rail classification yards. In rail classification yards, freight cars are separated, sorted according to their final destination, and assembled to form new outbound train blocks. Because the classification process requires considerable resources, one of the objectives is to minimize reclassification. Typically, cars with different final destinations but sharing some initial portion of their trips are assembled into blocks. In each rail classification yard, blocks are built and staged on classification tracks where they wait for the departure of an outbound train. The list of potential blocks that may go into each outbound train is specified by the makeup policy. Therefore, one needs to send each to the appropriate classification track based on the sorting strategy. In this rail classification problem, if a block at the converging junction fails to join the desired train, it recirculates back to the diverging junction (called 'rehumping'). Therefore, the rail classification problem can be regarded as a special case of the constrained sequencing problem with a diverging junction and a diverging/converging junction shown in Figure 4.

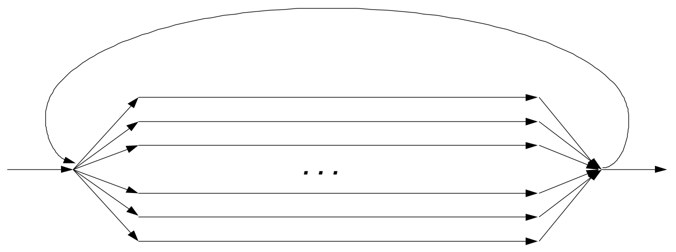

Figure 4. the Rail Classification Problem as a Special Case of the Constrained Sequencing Problem

The rail classification problem is similar to the constrained sequencing problem to be addressed in this research in terms of the decisions to be made. However, it has two different features as follows. First, departing and arriving train schedule constraints as well as the capacity of each classification track should be explicitly considered. Second, the number of changeovers should be counted on the rail for rehumping, not on the rail exiting the classification yard because attaching any ‘wrong’ car in a departing train is now allowed.

This topic was explored by Siddiqee (1972), who compared four sorting and train formulation schemes in a railroad classification yard. Yagar, Saccomanno et al. (1983) suggested a dynamic programming approach as well as a screening technique to optimize sorting and assembly operations. The relative performance of different multistage sorting strategies were investigated by Daganzo, Dowling et al. (1983). Each classification track is assigned several blocks and cars should be resorted during train formulation in multistage sorting. They derive equations for the service time per car of triangular sorting in classification yards. Three papers written by Daganzo (Daganzo (1986), Daganzo (1987), and Daganzo (1987)) also analyze and compare different classification strategies and give expressions for the switching work and space requirements. Dynamic blocking, in which the assignment of blocks to classification tracks is allowed to vary through time, is considered in the last two papers. See Assad (1980), Assad (1981), Choe, Sharp et al. (1993), and Cordeau, Toth et al. (1998) for a general review of rail transportation problems. However, because the rail classification problem considers

of inherent difference between the rail classification problem and the constrained sequencing problem, all sorting strategies discussed in the above cited papers are unrealistic to apply to the constrained sequencing problem (either with a diverging junction only, or with a random-access off-line buffer only) that are modeled in this dissertation.

Another decision-making problem can be found in the Order Accumulation / Sortation System (OAS) in a typical automated distribution center. In a common order picking system design, the sortation functions are separated from the order picking functions. To retrieve the items of an order from the warehouse, picking systems are used and many of these systems use ‘pick-wave’ where a group of orders is picked simultaneously with each picker being responsible for picking a single group of items for all the orders in a wave. Such a wave approach has been found to be more efficient than a serial picking scheme (where each picker selects all the items for one or more orders) in many systems. However, wave picking requires further sorting that is not required by serial picking systems. After retrieval by the sorting system from the warehouse, items move as a wave to OAS where they are assigned to one of shipping lanes for sortation into orders. Assignments are made based on the adopted lane assignment strategy and if recirculation is allowed, an item recirculates OAS until a shipping lane is assigned to that item. Therefore, identifying the optimal lane assignment strategy can be considered as a special case of the constrained sequencing problem with a diverging junction and an off-line buffer (i.e. recirculating conveyor) as in Figure 5.

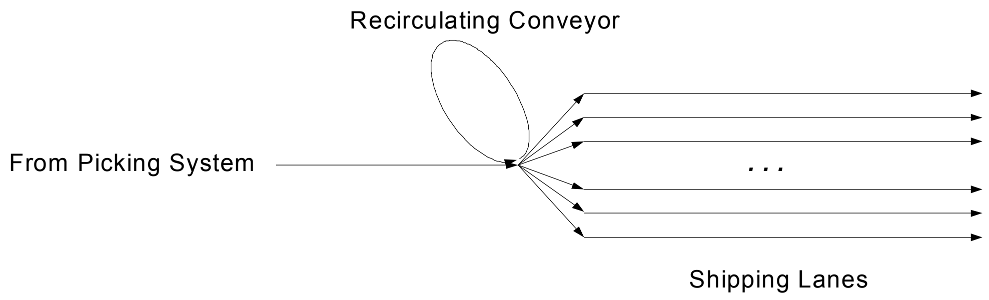

Figure 5. Lane Assignment Problem as a Special Case of the Constrained Sequencing Problem

However, like the rail classification problem, the lane assignment problem in OAS has a few characteristics different from the constrained sequencing problem as follows. First, capacity of each shipping lane needs to be explicitly considered. Second, an order may be pre-assigned to a specific shipping lane (e.g. a shipping lane dedicated for FedEx). Finally, if an order is not pre-assigned, usually a nonempty lane is dedicated to an order until the lane receives all items of that order. As a result, there are two common categories of lane assignment strategies – fixed priority rules and the next available rules (i.e. incidental rules). Fixed priority rules include such popular rules as 'sort the largest (or smallest) orders first' while the next available rules assign the next available lane to the item belonging to an order that has not yet assigned any lane.

Research on OAS is relatively scarce even though there exist many implementation of such systems in industry (see Johnson and Lofgren (1994), Gould (1991), Gould (1991), Horrey (1983), Schwind (1992), and Witt (1989)). In one of the first papers to analyze OAS, Bozer and Sharp (1985) used simulation to evaluate the throughput of OAS as a function of the number and length of lanes, the presence of a recirculation conveyor, the control system, and the induction capacity with the assumption that each lane is assigned to one order. Bozer, Quiroz et al. (1988) also used simulation to examine various line assignment strategies as well as wave release strategies under the assumption that there are more orders than lanes in OAS, finding incidental rules consistently outperforms fixed priority rules. Johnson (1998) proves this result by an analytical model for OAS. Choe (1990), Choe and Sharp (1991), Choe, Sharp et al. (1992), and Choe, Sharp et al. (1993) deal with questions on the design of both the picking system and its relationship to OAS. They developed approximate queueing models for the picking and OAS subsystems and incorporated those models into an overall analysis of the effect of picking schemes. Meller (1997) developed an algorithm for finding optimal lane assignment strategy when truck-loading requirements governs the sequence of order sortation. Customer orders are reverse loaded to company owned trucks with pre-specified delivery routes in the appropriate sequence based on the truck's route. A binary integer program is formulated and solved and this model assigns trucks and orders to shipping lanes with the objective of minimizing the total sorting time. Apart from the research specific to OAS, research on conveyor theory has been widely conducted (Muth (1979) for a general review and Bastani (1990) for recent works). However, all of those works deal with issues on material flow on the conveyor system (e.g. throughput, number of loading/unloading stations, time delay, and capacity).

Despite the fact that there is a large body of literature on sequencing assembly lines, most work adopts a static approach as a basic assumption of the problem (see Baybars (1986) and Yano and Bolat (1989) for a general review) and does not consider the constraints imposed by the material handling devices, such as strict FIFO constraints on a conveyor. The primary concern of a static approach is how to determine a single job sequence for the entire line for an available or recurring set of jobs, whereas the objective is, typically, to balance workload among the different processing departments (see Lee and Vairaktarakis (1997) and Yano and Rachamadugu (1991)). However, a dynamic approach would change the sequence on the fly for a given line without mechanical sequencing constraints.

Few papers, especially those that deal with mixed assembly lines, consider sequence-dependent setups. For example, Burns and Daganzo (1987) and Bolat, Savsar et al. (1994) consider lines where different jobs have different attributes or options and a setup occurs whenever two jobs with different options follow each other. They develop heuristics for sequencing these jobs with the objective of minimizing total setup cost. However, they do not consider the issue of constrained sequencing and assume that jobs have unique attributes. The issue of sequencing jobs with options is also discussed by Yano and Rachamadugu (1991) that consider cases where jobs with different options have different processing times. However, they assume that there is no setup between jobs with different options.

The issue of sequence-dependent setups has been addressed extensively in the traditional scheduling literature. Most such papers consider a single machine problem with multiple jobs, where individual jobs may belong to different families. A setup occurs whenever two consecutive jobs belong to different families. The individual jobs, irrespective of family membership, may carry different weights and have different due dates. The objective is to determine a sequence of jobs that optimizes one or more performance measures—typically, a function of job completion time such as maximum lateness, weighted completion time, or weighted tardiness. Examples of this work include Monma and Potts (1999), Potts and Wassenhove (1992), Unal and Kiran (1992), and Webster and Baker (1995). In general, scheduling with sequence-dependent setups is NP-hard, with polynomial algorithms available only for a few special cases (see Bruno and Downey (1978) and Laporte (1992)).

In all of the above literature on scheduling, mixed assembly line, or sequence-dependent setups, it is assumed that there is full flexibility as to how jobs are sequenced—i.e. not constrained by the sequence change mechanism. It is also assumed that setups are family- or lot-specific, with family or lot membership being known. The problems discussed in this dissertation are different from all those in above literature in two aspects. First, it is assumed that there is only constrained flexibility in how jobs can be re-sequenced. Second, some flexibility is allowed in assigning attributes that determine family membership among jobs. To our knowledge, this is the first attempt to consider the issue of sequencing with constrained flexibility and job-dependent setups.

CHAPTER 3

ANALYTICAL APPROACH

3.1 Models for Diverging Junctions

3.1.1 Constant Setup Cost with a Fixed Number of Downstream Queues

The simplest diverging junction is a single upstream queue feeding Q identical downstream queues. The problem is to decide which job is sent to which downstream queue to minimize total setup cost on all downstream queues. The following are assumed:

1. The number of downstream queues is fixed.
2. Setup cost is independent of job attributes and downstream queues.
3. The number of jobs in the upstream queue is fixed.
4. The attributes of all jobs in the upstream queue are fixed.
5. The capacity of each downstream queue is unlimited.
6. There is no setup cost for the first and last jobs sent to each downstream queue.
7. Queues follow the FIFO discipline.
8. All setups are done instantaneously—no setup time.

Assumption 6 means that if all jobs in a downstream queue have identical attributes, then no setup cost is assumed for that queue. This assumption can be relaxed by adding constraints that explicitly consider setup costs for the first and last jobs. Assumption 1 and 2 are relaxed in Section 3.1.2 and 3.2 as the model is extended. The following notations are introduced:

$U =$ upstream queue
$Q =$ number of downstream queues
$N =$ number of jobs in U
$C =$ changeover cost, a constant for each changeover

$$\overline{C}_{ij} = \begin{cases} C & \text{if the attribute of } i^{\text{th}} \text{ job in } U \text{ is different from the attribute of } j^{\text{th}} \text{ job in } U \\ 0 & \text{otherwise} \end{cases}$$

$$x_{ij} = \begin{cases} 1 & \text{if } i^{\text{th}} \text{ job in } U \text{ is located right before the j}^{\text{th}} \text{ job in } U \text{ on a downstream queue} \\ 0 & \text{otherwise} \end{cases}$$

$$y_{jq} = \begin{cases} 1 & \text{if } j^{\text{th}} \text{ job in } U \text{ is the last item to be sent to a downstream queue} \\ 0 & \text{otherwise} \end{cases}$$

$$z_{qi} = \begin{cases} 1 & \text{if } i^{\text{th}} \text{ job in } U \text{ is the first item to be sent to a downstream queue} \\ 0 & \text{otherwise} \end{cases}$$

One can find the following properties of the constrained sequencing problem.

Property 1.1 – Dispatching Constraint

The FIFO discipline of U prohibits the i^{th} job in U from being dispatched before the $(i\text{-}1)^{\text{th}}$ job in U is dispatched.

Property 1.2 – Conservation of Precedence Relationship

Due to *Property 1.1*, the precedence relationship among jobs in U is maintained in each downstream queue. That is, if the i^{th} job precedes the j^{th} job in U and both are dispatched to the same downstream queue, then the i^{th} job precedes the j^{th} job in the downstream queue.

The above two properties restrict the range of x_{ij} so that index i is always less than j. In addition, if each job as well as each downstream queue is represented as a node in a network, then represent each feasible arc in this network can be associated with x_{ij}, y_{jq}, or z_{qi}. In addition, the definitions of x_{ij}, y_{jq}, and z_{qi} require that each node has exactly one incoming arc and one outgoing arc. This way the constrained sequencing problem can be transformed as a network problem. Figure 6 shows a possible dispatching result from the example in Figures 1, where Figure 7 is the associated network representation.

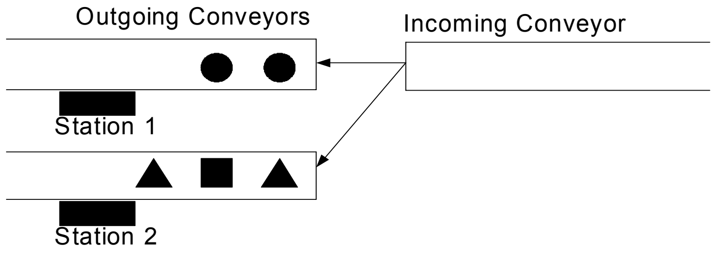

Figure 6. A Possible Dispatching Result for the Example in Figure 1

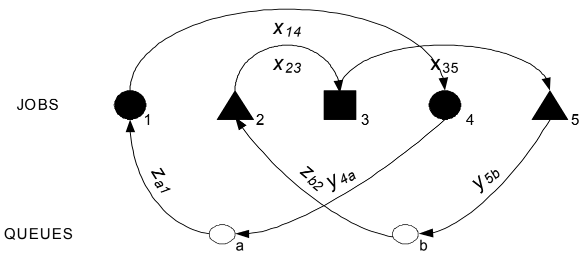

Figure 7. Network Representation Associated with Figure 6

The network described above can be interpreted such that if a job is directly connected with another job by variable *x*, then these two jobs are sent to the same downstream queue and are adjacent to each other in that queue. For the first job sent to a downstream queue, the incoming arc to the associated node is represented by *z*. For the last job, the outgoing arc from the associated node is represented by *y*. In Figure 7, a downstream queue has jobs 1 and 4, and another downstream queue has jobs 2, 3, and 5, in sequence.

The above network representation can be transformed into the following mixed-integer programming (MIP) formulation.

$$\textit{Minimize} \qquad \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \overline{C}_{ij} x_{ij} \qquad (1-1)$$

subject to

$$\sum_{j=i+1}^{N} x_{ij} + \sum_{q=1}^{Q} y_{iq} = 1 \qquad \forall\, i = 1, \cdots, N \qquad (1-2)$$

$$\sum_{i=1}^{j-1} x_{ij} + \sum_{q=1}^{Q} z_{qj} = 1 \qquad \forall\, j = 1, \cdots, N \qquad (1-3)$$

$$\sum_{i=1}^{N} y_{iq} = 1 \qquad \forall\, q = 1, \cdots, Q \qquad (1-4)$$

$$\sum_{j=1}^{N} z_{qj} = 1 \qquad \forall\, q = 1, \cdots, Q \qquad (1-5)$$

$$x_{ij},\, y_{iq},\, z_{qj} \text{ is binary} \qquad \forall\, i = 1, \cdots, N, \quad j = 1, \cdots, N, \quad q = 1, \cdots, Q \qquad (1-6)$$

The objective function (1-1) is the total setup cost of all jobs after dispatching. The Property 1.1 and 1.2 are ensured by not defining variables x_{ij} if index *i* is equal to or bigger than *j*. (1-2) ensures that any job *i* is assigned a successor job among all jobs after the i^{th} job in *U*, or is assigned as the last job in a downstream queue. (1-3) ensures that any job j is assigned a predecessor job among all jobs preceding the i^{th} job in U, or is assigned as the first job in a downstream queue. The intention of (1-4) and (1-5) is that each downstream queue is assigned exactly one first job and one last job, respectively. Note that if the first job is the same as the last job, then only one job is assigned to that downstream queue. The unintended result of (1-4) and (1-5) is that each downstream queue is utilized—i.e. at least one job is assigned—even when it is not necessary. In reality, achieving the minimum setup cost may not require all Q queues to be utilized. To find the minimum number of queues for achieving the minimum cost, one may try to find an optimal solution with Q, $Q - 1$, $Q - 2$, … queues until the minimum cost starts to increase. Another possible way of simultaneously identifying the number of queues to be used as well as the minimum cost is explained in Section 3.1.2.

Constraint (1-6) is added because the meaning of the decision variables demands integrality. However, (1-6) can be removed without loss of generality because of the following reasoning. A matrix is called *totally unimodular* if the determinant of every square submatrix formed from it has determinant –1, 0, or +1 (Bazaraa, Jarvis et al. (1990)). If our MIP formulation is represented as 'maximize $\boldsymbol{c}$ subject to $\boldsymbol{Ax} = \boldsymbol{b}$ where $\boldsymbol{b}$ is a binary variables vector', then $\boldsymbol{A}$ is a node-arc incidence matrix of a network because our formulation can be represented as a network. Note that the node-arc matrix is composed of (0, 1, –1), has two no-zero entries in each column, and the summation of each column equals zero.

In addition, if a matrix composed of (0, 1, –1) has no more than two no-zero entries in each column and the summation of all elements in column j equals zero if column j contains two no-zero coefficients, then this matrix is totally unimodular (Nemhauser and Wolsey (1988)). Therefore, $\boldsymbol{A}$ is totally unimodular because $\boldsymbol{A}$ is composed of (0, 1, –1), has two no-zero entries in each column, and the summation of each column equals zero. Furthermore, if $\boldsymbol{A}$ is totally unimodular and each element of $\boldsymbol{b}$ is integer-valued, then the optimal solution assigns all variables integer values (see Shapiro (1979) for a proof). Therefore, (1-6) can be removed. With (1-6) removed, the formulation reduces to the well-solved assignment problem. The removal of (1-6) without loss of generality can also be proved by showing one-to-one correspondence relationship between two equally sized sets composed of all jobs and queues as illustrated in Figure 8.

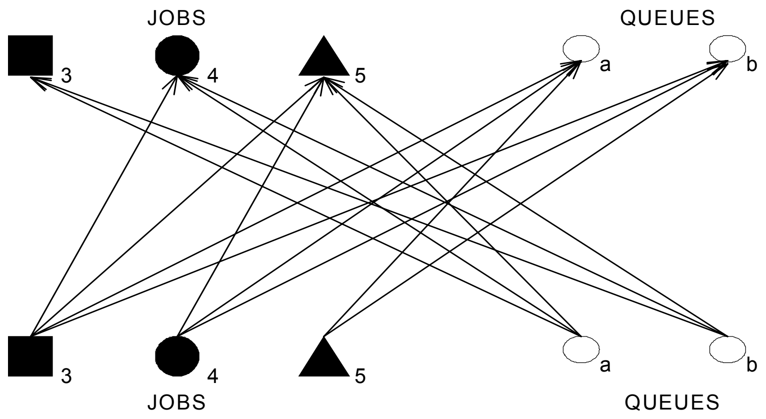

Figure 8. Assignment Problem Example of 3 Jobs and 2 Queues

Because the Hungarian method can solve the assignment problem optimally in O(N^3), modeling it as an assignment problem—compared to modeling it as an MIP or LP problem—has advantages in terms of speed and implementation cost. For speed, practical problems can be solved with a few hundred jobs to optimality in several seconds, allowing for on-the-spot optimal control in many applications. For implementation cost, dispatching logic in diverging or converging junctions of conveyors is usually implemented by Programmable Logic Controllers (PLCs). A PLC has limited memory, usually a few megabytes, and CPU power. Therefore, in most cases implementing an MIP- or LP-based algorithm in PLC environment requires an external system—where the MIP/LP solver is loaded—and a network module connecting the external system and PLC, causing high hardware and software costs. In contrast, implementing an assignment-problem-based algorithm can save implementation time and cost since it can be implemented in a PLC standalone environment and is relatively easy to program and debug.

3.1.2 Attribute Dependent Setup Cost with a Variable Number of Downstream Queues

3.1.2.1 Introduction

In reality, setup cost often depends on downstream queues and the attributes of the job to be processed. For example, the setup cost for color change in the paint shop usually depends on the paint color to be changed. In general, setup cost depends on both the job just finished and the job to be processed next.

In addition, in the simple constrained sequencing problem model discussed in Section 3.1.1, each downstream queue is forced to have at least one job. In the conveyor system design stage, the number of downstream conveyors is a design parameter. The cost of an additional conveyor, processing station, and extra floor space can outweigh the cost savings from setup reduction. In system operation, utilizing each additional downstream queue and workstation may incur extra costs for the labor, energy, and initial process setup. At one extreme, if the number of downstream queues is equal to one, no re-sequencing is possible. At the other extreme, if the number of queues is equal to the number of attributes, no setup cost is necessary because each queue can have jobs with identical attributes. Therefore, for a given set of jobs and cost values, if R and K denote the optimum number of downstream queues and the number of attributes, respectively, then the following condition holds:

$$1 \le R \le K \tag{2-1}$$

Based on (2-1), the maximum number of downstream queues one needs to consider is K. Each downstream queue is designated to a specific attribute if K queues are available for use. However, when the queue installation cost is considered, the optimum number of downstream queues is a variable and is often less than K. The rationale here is to start with maximum K, then to decide which queue to use and which queue not to use to minimize total cost. First, define the cost parameters as follows:

C'_{ij} = changeover cost when i^{th} job in U is located right before the j^{th} job in U on a downstream queue
(for the case that a specific attribute is given to each job in U and it is known in advance)
C''_{q} = cost of installing downstream queue q

Recall from Section 3.1.1, constraints (1-4) and (1-5) restrict each queue to exactly one incoming unit flow and one outgoing unit flow, shown in Figure 9.

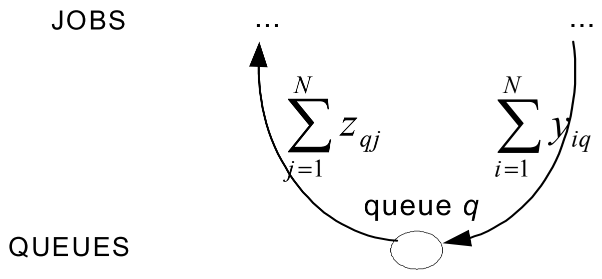

Figure 9. Part of the Network Representation for the Model in Section 4.1.1

A side effect of these constraints is that each queue is assigned at least one job. In this Section, the number of queues to be constructed is an integer in and some queues are allowed have no assigned job. The challenge is to allow some queues have no assigned job and, at the same time, to maintain a node-arc model.

Although there might be different ways of modeling such a situation, the approach taken here is to use all K queue nodes. Virtual arcs w_{qj} are defined to designate the outgoing flow from the unused queue node q to other queue node. Similarly, virtual arcs w_{iq} are defined to designate the incoming flow to queue node q—q may or may not be used—to go from other queue node. The extended queue node in this definition is depicted in Figure 10.

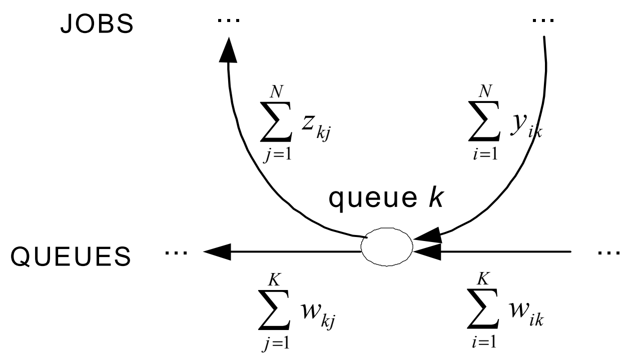

Figure 10. Extension the Network Representation in Figure 8

For each queue node, constraints: $\sum_{i=1}^{N} y_{ik} + \sum_{i=1, i\neq k}^{K} w_{ik} = 1$ and $\sum_{j=1}^{N} z_{kj} + \sum_{j=1, j\neq k}^{K} w_{kj} = 1$ are added. The former restricts a queue to have an entering arc either from a job, or from another queue. The latter similarly restricts the exiting arc. With the addition of virtual arcs and the above two constraints, a node-arc model can be built for the problem.

3.1.2.2 Installation Cost Calculation

The fact that no job is assigned to a queue with virtual links complicates the calculation for installation cost. Furthermore, a queue with or without an assigned job may be connected with other queues via virtual arcs. One way to handle this is to assign half of the installation cost to each arc connecting a job and a queue.

In this way, if queue q is connected from a job and connects to a job, the total cost becomes correct. If queue q is connected from queue i ($q \neq i$) and connects to a job, an adjustment need to be made by associating cost $\frac{C''_q - C''_i}{2}$ to virtual arc w_{iq}. This treatment makes up the installation cost for queue q ($\frac{C''_q}{2}+\frac{C''_q}{2}=C''_q$) which cancels out the installation cost for queue i ($\frac{C''_i}{2}-\frac{C''_i}{2}=0$).

Before the other cases are explained, the *auxiliary installation cost* associated with queue q, denoted as AC_q, is defined. AC_q represents the summation of costs of the arcs that start from or end at queue q. Then AC_q is defined as follows:

$$\begin{aligned}AC_q &= \frac{1}{2}C''_q(\sum_{i=1}^{N} y_{iq}+\sum_{j=1}^{N} z_{qj})+\frac{1}{2}\left[\frac{1}{2}\sum_{i=1,i\neq q}^{K}(C''_q-C''_i)w_{iq}+\frac{1}{2}\sum_{j=1,j\neq q}^{K}(C''_j-C''_q)w_{qj}\right]\\ &= \frac{1}{2}C''_q(\sum_{i=1}^{N} y_{iq}+\sum_{j=1}^{N} z_{qj})+\frac{1}{4}\sum_{i=1,i\neq q}^{K}(C''_q-C''_i)w_{iq}+\frac{1}{4}\sum_{j=1,j\neq q}^{K}(C''_j-C''_q)w_{qj}\end{aligned}$$

Note that terms $\left[\frac{1}{2}\sum_{i=1,i\neq q}^{K}(C''_q-C''_i)w_{iq}+\frac{1}{2}\sum_{j=1,j\neq q}^{K}(C''_j-C''_q)w_{qj}\right]$ are divided by *2* because these terms are double-counted when AC_q is summed for all queues for use in the MIP formulation. Since queues are interconnected by variable *w*, AC_q inevitably takes the following recursive form:

AC_q = true installation cost of queue q + $f(w_{iq}, AC_i)$ + $f(w_{qj}, AC_j)$, where $f(\ )$ denotes a function. (2-2)

It will be shown that all $f(w_{iq}, AC_i)$ and $f(w_{qj}, AC_j)$ terms in (2-2) eventually cancel out if one sums AC_q over all queues, resulting in ($\sum_q AC_q$ = total queue installation cost). First, from $\sum_{i=1}^{N} y_{ik}+\sum_{i=1,i\neq k}^{K} w_{ik}=1$ and $\sum_{j=1}^{N} z_{kj}+\sum_{j=1,j\neq k}^{K} w_{kj}=1$, one can derive

$0 \le \sum_{i=1}^{N} y_{iq},\quad \sum_{i=1, i\ne q}^{K} w_{iq},\quad \sum_{j=1}^{N} z_{qj},\quad \sum_{j=1, j\ne q}^{K} w_{qj} \le 1$ and all these variables are integers by definition. Therefore, each queue q can be classified into the following four cases based on all possible combinations made by $\sum_{i=1}^{N} y_{iq}, \sum_{j=1}^{N} z_{qj}, \sum_{i=1, i\ne q}^{K} w_{iq}$, and $\sum_{j=1, j\ne q}^{K} w_{qj}$:

<u>*Case 1*</u>: $\sum_{i=1}^{N} y_{iq} = \sum_{j=1}^{N} z_{qj} = 1$ and $\sum_{i=1, i\ne q}^{K} w_{iq} = \sum_{j=1, j\ne q}^{K} w_{qj} = 0 \qquad \forall\, q$

For Case 1, $AC_q = \frac{1}{2} C''_q (\sum_{i=1}^{N} y_{iq} + \sum_{j=1}^{N} z_{qj}) = C''_q$. Because $\sum_{i=1, i\ne q}^{K} w_{iq} = \sum_{j=1, j\ne q}^{K} w_{qj} = 0$, queue q is not connected with any other queue and it is clear that AC_q = the true installation cost of queue q = C''_q. Figure 11 shows an example for Case 1. This example—as well as other examples corresponding to other Cases—is one of the possible dispatching results from the situation explained in Figure 1, without an off-line buffer. In Figure 10, $AC_q = C''_q$ and $\sum_q AC_q = C''_a + C''_b$ which represents the total queue installation cost correctly.

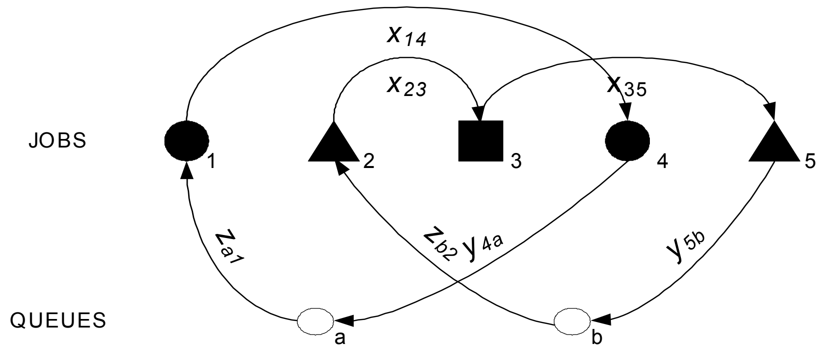

Figure 11. Network Representation Associated with Figure 6: Revisited

<u>*Case 2*</u>: $\sum_{i=1}^{N} y_{iq} = \sum_{j=1, j\neq q}^{K} w_{qj} = 1$ and $\sum_{j=1}^{N} z_{qj} = \sum_{i=1, i\neq q}^{K} w_{iq} = 0 \qquad \forall\, q$

For Case 2, $AC_q = \frac{1}{4}C''_q + \frac{1}{4}\sum_{j=1, j\neq q}^{K} C''_j w_{qj}$. Because $\sum_{j=1, j\neq q}^{K} w_{qj} = 1$ and $\sum_{i=1, i\neq q}^{K} w_{iq} = 0$, there is only one queue, denoted *r*, connected with queue *q* and queue *r* belongs to either Case 3 or Case 4. Therefore,

$$AC_q = \frac{1}{4}C''_q + \frac{1}{4}C''_r w_{qr}$$

$$AC_r = \begin{cases} \frac{3}{4}C''_r - \frac{1}{4}C''_r w_{qr} & \text{if } r \text{ belongs to Case 3} \\ \frac{1}{4}\sum_{j=1, j\neq r}^{K} C''_j w_{qj} - \frac{1}{4}C''_r w_{rq} & \text{if } r \text{ belongs to Case 4} \end{cases}$$

$$\therefore AC_q + AC_r = \begin{cases} C''_r & \text{if } r \text{ belongs to Case 3} \\ \frac{1}{4}\sum_{j=1, j\neq r}^{K} C''_j w_{qj} & \text{if } r \text{ belongs to Case 4} \end{cases} \qquad (2\text{-}3)$$

If queue *r* belongs to Case 3, there is only one queue, *q*, that is connected with queue *r* and $AC_q + AC_r = C''_r$ by (2-3). If queue *r* belongs to Case 4, queue *r* is connected with a queue belonging either to Case 2 or Case 4. The sub-network of the associated problem network, composed of queues including *q* and *r* and arcs connecting these queues takes form by sequentially connecting one queue in Case 2, some queues in Case 4, and one queue in Case 3; the number of queues in Case 4 ranges from zero to *Q*–2. By using (2-3), (2-4), and (2-5), one can get $\sum_b AC_b = C''_r$, where $b \in$ set of all queues in this sub-network. If this sub-network is interpreted such that queue *r* is used and all other queues in the sub-network are not used, *AC* for each queue in this sub-network represents queue installation cost correctly. An illustrative example of Case 2 is shown in Figure 12, where $AC_q + AC_r = C''_r$; queue r is in Case 3.

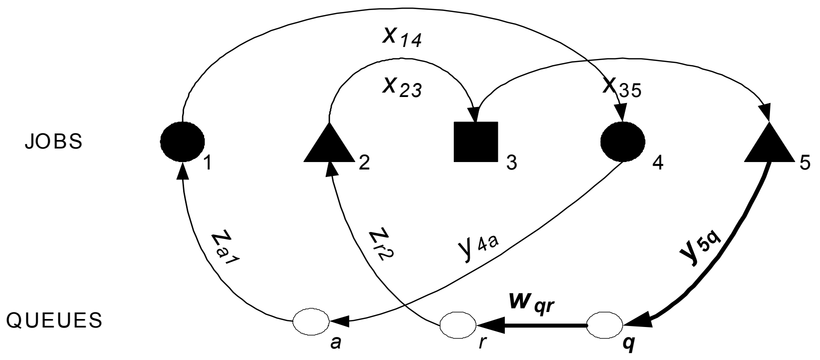

Figure 12. Example of Case 2

<u>*Case 3*</u>: $\sum_{j=1}^{N} z_{qj} = \sum_{i=1,i\neq q}^{K} w_{iq} = 1$ and $\sum_{i=1}^{N} y_{iq} = \sum_{j=1,j\neq q}^{K} w_{qj} = 0 \qquad \forall\, q$

For Case 3, $AC_q = \frac{3}{4}C''_q - \frac{1}{4}\sum_{i=1,i\neq q}^{K} C''_i w_{iq}$. Because $\sum_{i=1,i\neq q}^{K} w_{iq} = 1$ and $\sum_{j=1,j\neq q}^{K} w_{qj} = 0$, there is only one queue, p, connected with queue q and queue p belongs to either Case 2 or Case 4. Therefore,

$$AC_p = \begin{cases} \frac{1}{4}C''_q + \frac{1}{4}C''_p w_{pq} & \text{if } p \text{ belongs to Case 2} \\ \frac{1}{4}C''_q - \frac{1}{4}\sum_{i=1,i\neq p}^{K} C''_i w_{ip} & \text{if } p \text{ belongs to Case 4} \end{cases}$$

$$\therefore AC_q + AC_p = \begin{cases} C''_q & \text{if } p \text{ belongs to Case 2} \\ C''_q - \frac{1}{4}\sum_{i=1,i\neq p}^{K} C''_i w_{ip} & \text{if } p \text{ belongs to Case 4} \end{cases} \qquad (2\text{-}4)$$

If queue p belongs to Case 2, there is only one queue, q, that is connected with queue p and $AC_q + AC_p = C''_p$ by (2-4). If queue p belongs to Case 4,it is connected with a queue belonging either to Case 3 or Case 4. The sub-network composed of queues that include q and p and arcs connecting these queues takes form by sequentially connecting one queue in Case 3, some queues in Case 4, and one queue in Case 2; the number of queues in Case 4 ranges from zero to Q–2. By using (2-3), (2-4), and (2-5), one can get $\sum_b AC_b = C''_q$, where $b \in$ set of all queues in this sub-network. If this sub-network is interpreted such that queue q is used and all other queues in this sub-network are not used, then AC for each queue in this sub-network represents the queue installation cost correctly. Figure 13 shows an example of Case 3 where $AC_q + AC_p = C''_q$; queue p is in Case 2.

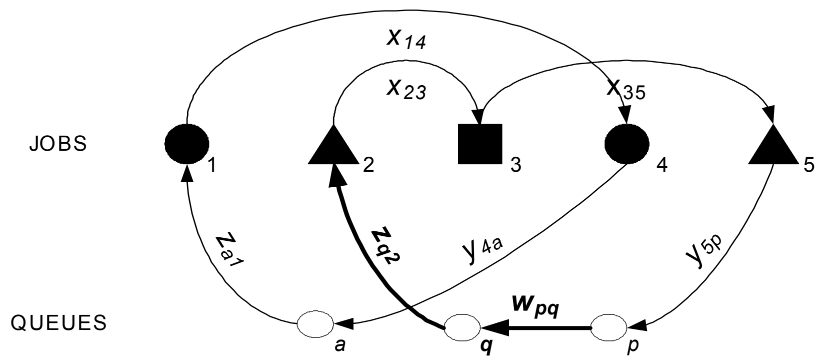

Figure 13. Example of Case 3

<u>*Case 4*</u>: $\sum_{j=1,j\neq q}^{K} w_{qj} = \sum_{i=1,i\neq q}^{K} w_{iq} = 1$ and $\sum_{i=1}^{N} y_{iq} = \sum_{j=1}^{N} z_{qj} = 0 \qquad \forall q$

For Case 4, $AC_q = \frac{1}{4}\sum_{j=1,j\neq q}^{K} C''_j w_{jq} - \frac{1}{4}\sum_{i=1,i\neq q}^{K} C''_i w_{iq}$. Because $\sum_{i=1,i\neq q}^{K} w_{iq} = \sum_{j=1,j\neq q}^{K} w_{qj} = 1$,

queue *q* is connected from a queue, *s*, belonging to Case 2 or Case 4 and is connected to a queue, *t*, belonging to Case 3 or Case 4. Therefore,

$$AC_q = \frac{1}{4}C''_t - \frac{1}{4}C''_s$$

$$AC_s = \begin{cases} \frac{1}{4}C''_q + \frac{1}{4}C''_s & \text{if } s \text{ belongs to Case 2} \\ \frac{1}{4}C''_q - \frac{1}{4}\sum_{i=1,i\neq p}^{K} C''_i w_{ip} & \text{if } s \text{ belongs to Case 4} \end{cases}$$

$$AC_t = \begin{cases} \frac{3}{4}C''_t - \frac{1}{4}C''_q & \text{if } t \text{ belongs to Case 3} \\ \frac{1}{4}\sum_{j=1,j\neq t}^{K} C''_j w_{qj} - \frac{1}{4}C''_q & \text{if } t \text{ belongs to Case 4} \end{cases}$$

$$\therefore AC_q + AC_s + AC_t$$

$$= \begin{cases} C''_t & \text{if } s \text{ belongs to Case 2 and } t \text{ belongs to Case 3} \\ \frac{1}{4}C''_t + \frac{1}{4}\sum_{j=1,j\neq t}^{K} C''_j w_{qj} & \text{if } s \text{ belongs to Case 2 and } t \text{ belongs to Case 4} \\ C''_t - \frac{1}{4}C''_s - \frac{1}{4}\sum_{i=1,i\neq s}^{K} C''_i w_{iq} & \text{if } s \text{ belongs to Case 3 and } t \text{ belongs to Case 4} \\ \frac{1}{4}\sum_{j=1,j\neq t}^{K} C''_j w_{jt} - \frac{1}{4}\sum_{i=1,i\neq s}^{K} C''_i w_{is} & \text{if } s \text{ belongs to Case 4 and } t \text{ belongs to Case 4} \end{cases} \qquad (2\text{-}5)$$

Using similar reasoning to that for Case 2 and Case 3, one can think of a sub-network composed of queues including *q* and *s*, and *t* and arcs connecting these queues. In addition, one can conclude that *AC* for each queue in this sub-network represents the queue installation cost correctly. An example of Case 4 is shown in Figure 14 where $AC_q + AC_s + AC_t = C''_t$; queue s is in Case 2 and queue t is in Case 3.

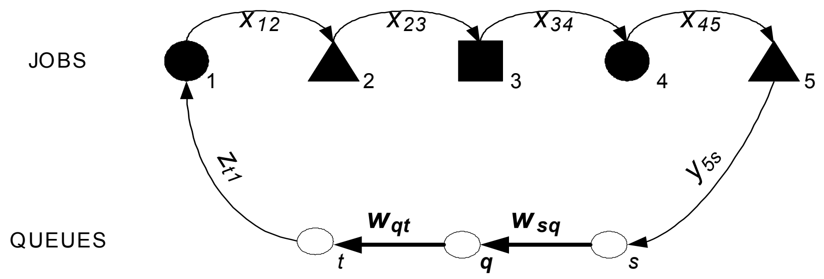

Figure 14. Example of Case 4

Summarizing the discussions in the above four Cases, the following facts can be derived:

- $f(w_{iq}, AC_i)$ and $f(w_{qj}, AC_j)$ will eventually cancel out to zero if one sums up AC_q over all queues, resulting in $\sum_{q=1}^{K} AC_q$ = total queue installation cost.
- $\sum_{j=1, j\neq q}^{K} w_{qj} = 1$ means that queue q is a queue with no assigned job, and

  $\sum_{j=1, j\neq q}^{K} w_{qj} = 0$ means that queue q is a queue with at least one job assigned.

Note that if cost $\frac{C''_i - C''_j}{2}$ (not $\frac{C''_j - C''_i}{2}$) is associated with variable w_{ij}, then

$\sum_{i=1, i\neq q}^{K} w_{iq} = 1$ means that queue q is a queue with no assigned job, and

$\sum_{i=1, i\neq q}^{K} w_{iq} = 0$ means that queue q is a queue with at least one job assigned.

Finally, one needs to calculate $\sum_{q=1}^{K} AC_q$

$$= \sum_{q=1}^{K}\left[\frac{1}{2}C''_q(\sum_{i=1}^{N} y_{iq} + \sum_{j=1}^{N} z_{qj})\right] + \frac{1}{2}\sum_{q=1}^{K}\left[\frac{1}{2}\sum_{i=1, i\neq q}^{K}(C''_q - C''_i)w_{iq} + \frac{1}{2}\sum_{j=1, j\neq q}^{K}(C''_j - C''_q)w_{qj}\right]$$

$$= \frac{1}{2}\sum_{q=1}^{K} C''_q(\sum_{i=1}^{N} y_{iq} + \sum_{j=1}^{N} z_{qj}) + \frac{1}{2}\sum_{i=1}^{K}\sum_{j=1, j\neq i}^{K}(C''_j - C''_i)w_{ij} .$$

3.1.2.3 Model Formulation

Now one can formulate the MIP model that explicitly considers the number of available queues and the attribute-dependent setup cost as follows:

$$\textit{Minimize} \quad \sum_{i=1}^{N-1}\sum_{j=i+1}^{N} C'_{ij}x_{ij} + \frac{1}{2}\sum_{q=1}^{K} C''_q(\sum_{i=1}^{N} y_{iq} + \sum_{j=1}^{N} z_{qj}) + \frac{1}{2}\sum_{i=1}^{K}\sum_{j=1, j\neq i}^{K}(C''_j - C''_i)w_{ij} \qquad (2-6)$$

subject to

$$\sum_{j=i+1}^{N} x_{ij} + \sum_{q=1}^{K} y_{iq} = 1 \qquad \forall\, i = 1, \cdots, N \qquad (2-5)$$

$$\sum_{i=1}^{j-1} x_{ij} + \sum_{q=1}^{K} z_{qj} = 1 \qquad \forall\, i = 1, \cdots, N \qquad (2-8)$$

$$\sum_{i=1}^{N} y_{iq} + \sum_{i=1, i\neq q}^{K} w_{iq} = 1 \qquad \forall\, q = 1, \cdots, K \qquad (2-9)$$

$$\sum_{j=1}^{N} z_{qj} + \sum_{j=1, j\neq q}^{K} w_{qj} = 1 \qquad \forall\, q = 1, \cdots, K \qquad (2-10)$$

$$x_{ij}, y_{iq}, z_{qj}, w_{iq}, w_{qj} \in \{0, 1\} \qquad \forall\, i,\ j,\ q \qquad (2-11)$$

The first term in (2-6) is the sum of the setup cost of all jobs in all downstream queues, as in the simple model in Section 3.1.1, while the second term is the cost of adding queue q and the third term is for canceling the cost correction due to over accounting in the second term. (2-6) can be simplified by using (2-7) and (2-8) as follows:

$$\sum_{i=1}^{N-1}\sum_{j=i+1}^{N}C'_{ij}x_{ij}+\frac{1}{2}\sum_{q=1}^{K}C''_{q}(\sum_{i=1}^{N}y_{iq}+\sum_{j=1}^{N}z_{qj})+\frac{1}{2}\sum_{i=1}^{K}\sum_{j=1,j\neq i}^{K}(C''_{j}-C''_{i})w_{ij}=$$

$$\sum_{i=1}^{N-1}\sum_{j=i+1}^{N}C'_{ij}x_{ij}+\frac{1}{2}\sum_{q=1}^{K}C''_{q}(1-\sum_{i=1,i\neq q}^{K}w_{iq}+1-\sum_{j=1,j\neq q}^{K}w_{qj})+\frac{1}{2}\sum_{i=1}^{K}\sum_{j=1,j\neq i}^{K}(C''_{j}-C''_{i})w_{ij}=$$

$$\sum_{i=1}^{N-1}\sum_{j=i+1}^{N}C'_{ij}x_{ij}+\sum_{q=1}^{K}C''_{q}-\frac{1}{2}\sum_{q=1}^{K}C''_{q}\sum_{i=1,i\neq q}^{K}w_{iq}-\frac{1}{2}\sum_{q=1}^{K}C''_{q}\sum_{j=1,j\neq q}^{K}w_{qj}=$$

$$\sum_{i=1}^{N-1}\sum_{j=i+1}^{N}C'_{ij}x_{ij}+\frac{1}{2}\sum_{q=1}^{K}C''_{q}(2-\sum_{i=1,i\neq q}^{K}w_{iq}-\sum_{j=1,j\neq q}^{K}w_{qj}) \qquad (2\text{-}6)'$$

An explanation for constraints (2-7), (2-6), and (2-11) is given in Section 3.1.1. For (2-6), (2-9), and (2-10), newly added variables w are needed to identify unused queues. Again, (2-11) can be safely removed using the same reasoning as in Section 3.1.1—totally unimodular. As a result, the problem becomes the well-solved assignment problem again. Note that it is meaningless to differentiate the cost of installing downstream queues from the cost of using downstream queues. Differentiating these two costs is meaningful only when it is possible to have a downstream queue that is not used in dispatching, which is not the case in our model—for any solution having an unused downstream queue, an equal or better solution having no unused downstream queue can be found. As a simple proof, for any solution having an unused queue, think of the modified solution with one job assigned to the unused queue. Because of the assumption that there is no setup cost for the first job assigned to a downstream queue, the modified solution has an equal or lesser total setup cost, depending whether or not the predecessor job and the successor job of the assigned job have the same attributes.

3.2 Models for an Off-line Buffer

3.2.1 Generalized Sequence-dependent Setup Cost

Now let us consider the constrained sequencing problem with one upstream queue, one downstream queue, and one off-line random access buffer between them. In this case, the job can bypass or visit off-line buffer. Furthermore, the job in off-line buffer stays for a while and is released to downstream queue later, making it possible to re-sequence the original sequence. In manufacturing systems, buffers allow workstations to operate more independently, cushioning against machine failures, worker or part shortages, and production rate differences (Askin and Standridge (1993)). Use of an off-line random access buffer or multiple off-line buffers with single capacity can be observed in many manufacturing systems.

Our objective is still to reduce the total number of attribute changes between adjacent jobs. Note that this model can be applied to multiple off-line random access buffers case because these multiple buffers can be modeled as one buffer problem without loss of generality, if these multiple buffers share one input/output point. The old definitions of U, D, A, and v, are used as they are, while B is defined as buffer size and x redefined as follows. Note that the definition of x_{ij} has been modified because the old definitions used in previous Sections are not appropriate for modeling off-line buffer cases.

D = downstream queue

B = number of slots in an offline buffer

$$A_{ik} = \begin{cases} 1 & \text{if attribute } k \text{ is assigned to the } i^{\text{th}} \text{ job in } U \\ 0 & \text{otherwise} \end{cases}$$

$$v_{jkl} = \begin{cases} 1 & \text{if the } j^{\text{th}} \text{ job in } D \text{ has attribute } k \text{ while the } j+1^{\text{th}} \text{ job in } D \text{ has attribute } l \\ 0 & \text{otherwise} \end{cases}$$

$$x_{ijk} = \begin{cases} 1 & \text{if the } i^{\text{th}} \text{ job with attribute } k \text{ in } U \text{ is sent to the } j^{\text{th}} \text{ position in } D \\ 0 & \text{otherwise} \end{cases}$$

An off-line buffer usually does not provide full sequencing flexibility, and the following Theorem 2.1 differentiates feasible instances from infeasible ones. Note that if $B \geq N$, full sequencing flexibility is always guaranteed.

Theorem 2.1

For the constrained sequencing problem with an off-line buffer with capacity B, there exists a limit on how much a job in U can be moved forward in D. Specifically,

$$x_{ijk} = 0 \quad \forall\, i, j, k \quad \text{where } i - j > B, \quad i = 1, \cdots, N, \quad j = 1, \cdots, N, \quad k = 1, \cdots, K$$

Proof

For any i, let us consider the range of values that j can take. If i^{th} job of U visits off-line buffer, $j > i$ should hold because off-line buffer is randomly accessible and i^{th} job can be reinserted at j^{th} position in D as long as $j > i$. If i^{th} job bypasses off-line buffer, $i - B \leq j \leq i$ holds because at most B jobs can stay in off-line buffer at the same time. More specifically, when i^{th} job bypasses off-line buffer, $j = i$ holds if off-line buffer doesn't contain any job, and $j = i - B$ holds if off-line buffer contains B jobs.

Combining these two possible cases, one can conclude that $x_{ijk} = 0$ for all i and j where $i - j > B$ by the definition of x_{ijk}. □

Figure 15 shows an illustrative example with $B = 1$ derived from the problem discussed in Figure 1. It shows all possible decisions available on 3rd job of U where solid lines represent cases of bypassing off-line buffer (i.e. x_{32} or $x_{33} = 1$) and dotted lines represent cases of visiting off-line buffer (i.e. x_{34} or $x_{35} = 1$).

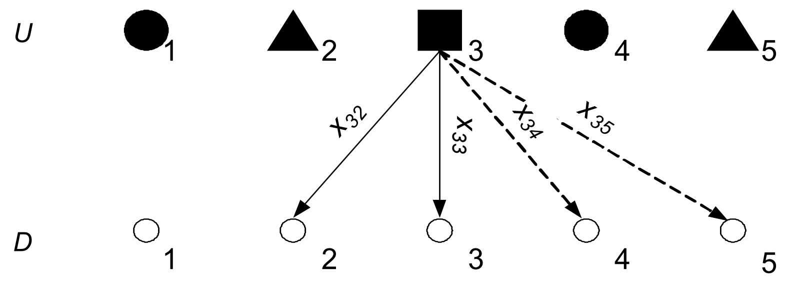

Figure 15. Network Representation Example of Constrained Sequencing Problem with an Off-line Buffer of Capacity 1

Using theorem 2.1 and the network model described in Figure 14, the MIP model can be formulated for one downstream queue and one off-line buffer as follows. Note that this formulation is for the case where the setup cost depends on both the attribute of the job just finished and the attribute of the job to be processed next.

$$Minimize \quad \sum_{j=1}^{N-1}\sum_{k}^{K}\sum_{l=1}^{K} C'_{kl} v_{jkl} \qquad (4-1)$$

subject to

$$\sum_{j=1}^{N}\sum_{k=1}^{K} x_{ijk} = 1 \qquad \forall\, i = 1, \cdots, N \qquad (4-2)$$

$$\sum_{i=1}^{N}\sum_{k=1}^{K} x_{ijk} = 1 \qquad \forall\, j = 1, \cdots, N \qquad (4-3)$$

$$x_{ijk} = 0 \qquad \forall\, i,\ j,\ k$$

$$where \quad i - j > B, \quad i = 1, \cdots, N, \quad j = 1, \cdots, N, \quad k = 1, \cdots, K \qquad (4-4)$$

$$v_{jkl} \geq \sum_{i=1}^{N} x_{ijk} + \sum_{i=1}^{N} x_{i,j+1,l} - 1 \qquad \forall\, j = 1, \cdots, N-1, \quad k = 1, \cdots, K, \quad l = 1, \cdots, K \qquad (4-5)$$

$$\sum_{j=1}^{N} x_{ijk} \leq A_{ik} \qquad \forall\, i = 1, \cdots, N, \quad k = 1, \cdots, K \qquad (4-6)$$

$$x_{ij}, v_{jkl} \text{ is binary} \quad \forall\, i = 1, \cdots, N, \quad j = 1, \cdots, N-1, \quad k = 1, \cdots, K, \quad l = 1, \cdots, K \ (4-7)$$

(4-2) and (4-3) are flow conservation constraints. (4-2) makes sure that each job is sent downstream exactly once while (4-3) is for forcing that each slot in the downstream queue receives one job. (4-4) is from theorem 2.1, preventing any re-sequencing caused by having more than *B* slots in off-line buffer. (4-5) counts the number of attribute changes by forcing v_{jkl} = 1 only when the job at *j*th position of *D* has attribute *k* and the job at *j+1*th position of *D* has attribute *l*. More specifically, if $\sum_{i=1}^{N} x_{ijk}$ or $\sum_{i=1}^{N} x_{i,j+1,l}$ becomes zero, because $0 \leq \sum_{i=1}^{N} x_{ijk} \leq 1$, $0 \leq \sum_{i=1}^{N} x_{i,j+1,l} \leq 1$, v_{jkl} is forced to be zero to minimize the objective function (4-1). That means v_{jkl} = 1 only when $\sum_{i=1}^{N} x_{ijk} = \sum_{i=1}^{N} x_{i,j+1,l} = 1$. (4-6) restricts the range of values that v_{jkl} can take, like (3-6) and (3'-6) in Appendix A.

Since the above formulation is a MIP, generally the constrained sequencing problem with an off-line buffer is an NP-hard problem. Therefore solving the problem with large instances takes prohibitively long time, especially when each dispatching decision should be made for each incoming job on rolling horizon time window (in our case study, average time between arrivals of adjacent jobs was 1 minute, meaning each decision should be made in 30 ~ 45 seconds). Therefore we need to develop heuristics with reasonably short solution time and good average performance compared to optimal solution.

First, it will be shown that constrained sequencing problem with an off-line buffer of capacity B is equivalent to constrained sequencing problem with B off-line buffers of capacity 1. Then a method of optimally solving constrained sequencing problem with an off-line buffer of capacity 1 (with polynomial time) will be developed. Finally, to obtain a good quality feasible solution in a reasonable time, we will propose heuristics of sequentially solving constrained sequencing problem with an off-line buffer of capacity 1 for B times.

Theorem 2.2

Constrained sequencing problem with a random-access off-line buffer of capacity B is equivalent to constrained sequencing problem with B off-line buffers of capacity 1.

This can be proved by showing random deposit to and random pick from the buffers, or by showing the relationship between adjacent off-line buffers of capacity 1. Here, the former can be proved as follows (the formal proof of the latter is in the Appendix). First of all, let us denote constrained sequencing problem with an off-line buffer of capacity B as problem α, constrained sequencing problem with B off-line buffers of capacity 1 as problem β, respectively. Then in problem α, for any job in the incoming sequence, if that job reaches junction point, the decision whether that job bypasses or visits off-line buffer needs to be made. If off-line buffer is visited, one needs to decide when that job is released to downstream queue. Since all slots of off-line buffer in problem α, as well as all off-line buffers in problem β are randomly accessible, problem α and problem β are identical in available options (bypassing or visiting off-line buffer). Furthermore, in problem α, if i^{th} job of U visits off-line buffer (of capacity B), $x_{ijk} = 0 \quad \forall\ i, j, k$ where $i > j, \quad i = 1, \cdots, N, \quad j = 1, \cdots, N, \quad k = 1, \cdots, K$ holds by theorem 2.1, and in problem β, exactly the same restriction holds if i^{th} job of U visits one of B buffers of capacity 1 because all B off-line buffers are randomly accessible and i^{th} job can be reinserted at j^{th} position in D as long as $j > i$. Therefore, for any of three different kinds of decisions (bypassing, visiting, or leaving off-line buffer) on each job in the incoming sequence, it can be shown that problem α and problem β are identical in all available options. As a result, we can find one (and the only one) feasible solution of problem α for any feasible solution of problem β, and vice versa. Therefore problem α and problem β are identical.

To develop polynomial time algorithm of optimally solving constrained sequencing problem with an off-line buffer of capacity 1, the following theorem that is unique to the case of buffer capacity 1 need to be exploited.

Theorem 2.3

For constrained sequencing problem with an off-line buffer of capacity 1, there exists an unchanged sequence block from *i*th job to *j – 1*th job of *U*. Specifically, if

$x_{ijk} = 1$ where $i < j, \quad i = 1, \cdots, N-1, \quad j = 2, \cdots, N, \quad k = 1, \cdots, K$, then

$x_{i+1,i,k} = x_{i+2,i+1,k} = \cdots = x_{j,j-1,k} = 1$.

Proof

As explained in proof of Theorem 2.1, $x_{ijk} = 1$ where $i < j$ means *i*th job of *U* visits off-line buffer and is placed at *j*th position of *D*. Since buffer capacity is one, as soon as *i*th job of *U* visits off-line buffer, no other job can visit off-line buffer until that job is released from off-line buffer. Therefore *i+1*, …, *j*th job of *U* are forced to bypass off-line buffer, making $x_{i+1,i,k} = x_{i+2,i+1,k} = \cdots = x_{j,j-1,k} = 1$. □

Theorem 2.3 enables all possible options to be identified for each job in *U*. Specifically, for each job in the conveyor junction point, bypassing off-line buffer is always possible while visiting off-line buffer is possible only when off-line buffer is empty. Therefore, if status of an off-line buffer (either empty or occupied) as well as status of each job (either bypassing or visiting an off-line buffer) is tracked, constrained sequencing problem with an off-line buffer of capacity 1 can be modeled as a shortest path problem. To model the problem as shortest path problem, two kinds of nodes need to be defined first: (α, G) and (β, S), $\alpha \in \{1, \ldots, N\}$, $\beta \in \{1, \ldots, N-1\}$, where

G bypasses off-line buffer,

S visits off-line buffer, and

α and β is job number (i.e. α^{th} job) in U.

We assume that the changeover cost depends on both the job just finished and the job to be processed next. Therefore, to calculate the changeover cost between these two jobs, all of four possible combinations (case of bypassing or visiting an off-line buffer for each job) need to be identified and changeover cost needs to be calculated for each combination. For example, if i^{th} job of U bypasses but $(i+1)^{\text{th}}$ job visits an off-line buffer, changeover cost depends on the i^{th} job and $(i+2)^{\text{th}}$ job because $(i+2)^{\text{th}}$ job bypasses an off-line buffer by Theorem 2.3, making i^{th} job and $(i+2)^{\text{th}}$ job adjacent in D. Therefore, arcs need to be defined in eight different categories as follows:

1) Arc from (α, G) to $(\alpha+1, G)$ with cost $C'_{\alpha,\alpha+1}$, meaning that $(\alpha+1)^{\text{th}}$ job of U bypasses off-line buffer, where $\alpha \in \{1, \ldots, N-1\}$.

2) Arc from (α, G) to $(\alpha+1, S)$ with cost $C'_{\alpha,\alpha+2}$, meaning that $(\alpha+1)^{\text{th}}$ job of U visits off-line buffer, where $\alpha \in \{1, \ldots, N-2\}$.

3) Arc from (β, S) to ($\beta+\sigma$, G) with cost $\sum_{i=1}^{\sigma-2} C'_{\beta+i,\beta+i+1} + C'_{\beta+\sigma-1,\beta} + C'_{\beta,\beta+\sigma}$, meaning that βth job of U that was in off-line buffer is reinserted at $(\beta+\sigma-1)$th position of D, and $\beta+\sigma$th job of U bypasses off-line buffer, where

$\beta \in \{1, \dots, N-1\}$, $\sigma \in \{2, \dots, N-1\}$, $\beta+\sigma \leq N$.

4) Arc from (β, S) to ($\beta+\sigma$, S) with cost $\sum_{i=1}^{\sigma-2} C'_{\beta+i,\beta+i+1} + C'_{\beta+\sigma-1,\beta} + C'_{\beta,\beta+\sigma+1}$, meaning that βth job of U that was in off-line buffer is reinserted at $(\beta+\sigma-1)$th position of D, and $\beta+\sigma$th job of U visits off-line buffer, where

$\beta \in \{1, \dots, N-1\}$, $\sigma \in \{2, \dots, N-2\}$, $\beta+\sigma \leq N$.

5) Arc from (β, S) to sink node with cost $\sum_{i=1}^{N-\beta-1} C'_{\beta+i,\beta+i+1} + C'_{N,\beta}$, meaning that βth job of U that was in off-line buffer is reinserted at the end (i.e. $(\beta+\sigma-1)$th position) of D, where $\beta \in \{1, \dots, N-1\}$.

6) Arc from source node to (1, G) with cost zero, meaning that 1st job of U bypasses off-line buffer.

7) Arc from source node to (1, S) with cost zero, meaning that 1st job of U visits off-line buffer.

8) Arc from (N, G) to sink node with cost zero, meaning that the last (i.e. N^{th}) job of U bypasses off-line buffer.

Arc 1 means bypassing while arc 2 means visiting an off-line buffer. Arc 6 and 8 are special cases of arc 1 while arc 7 is a special case of arc 2. Note that the costs associated with arc 6, 7, and 8 are zero because of the assumption that no changeover cost is associated with the first and last jobs in D. Also note that node (N, S) is undefined because any arc connected with this node means that the N^{th} job visits an off-line buffer which is not allowed by assumption. Arc 3 and 4 represent decisions related with when the job leaves an off-line buffer and is reinserted in D. If β^{th} job of U that was in off-line buffer is reinserted before $(\beta+\sigma)^{\text{th}}$ job, then all jobs between β^{th} job and $(\beta+\sigma)^{\text{th}}$ job should bypass an off-line buffer by Theorem 2.3. Therefore, total changeover cost for these jobs can be calculated, as shown in arc 3 and 4 arc cost. Note that once a job visits an off-line buffer, it does not leave the off-line buffer until the following job is sent to D. Therefore, no arc from (β, S) to ($\beta+1$, S) is defined. Table 1 shows arcs classification based on the starting and ending nodes of each arc.

Table 1. Classification of Arcs in Shortest Path Problem Network

		Ending Node	
		G	*S*
Starting Node	*G*	Bypass (Arc 1, 6, 8)	Visit (Arc 2, 7)
	S	Leave (Arc 3, 5)	Leave & Visit (Arc 4)

Then for any feasible solution of the original problem (constrained sequencing problem with an off-line buffer of capacity 1), a unique matching path from source to sink node can be found, and vice versa. In addition, in that case objective function value of the original problem is the same as cost of the matching path. Figure 16 shows a simple example of such a shortest path problem network corresponding the original problem with 4 jobs. In this example, for a feasible path from source → (1, *G*) → (2, *S*) → (4, *G*) → sink node, there exists a unique feasible solution of the original problem, and that solution means that 2nd job of *U* visits off-line buffer and it is reinserted in 4th position of *D* (no other job visits off-line buffer). Furthermore, the reverse (for any feasible solution of the original problem, there exists a unique path in the network) also holds and the costs of these two solutions are the same ($= C'_{13} + C'_{32} + C'_{24}$). Therefore it can be proved that the original problem is equivalent to the associated shortest path problem. Then the constrained sequencing problem can be solved with an off-line buffer of capacity one in polynomial time with complexity $O(N^2)$ by applying Dijkstra's algorithm (Ahuja, Magnanti et al. (1993)).

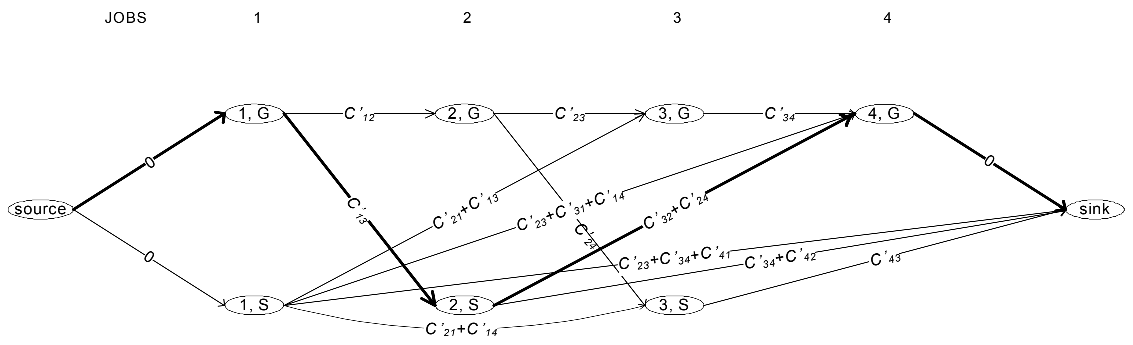

Figure 16. Simple Example of Shortest Path Problem Network

Furthermore, solution time for the above shortest path problem can be reduced by removing some arcs through exploiting the special structure of the constrained sequencing problem. First, for constrained sequencing problem with an off-line buffer of capacity 1, let *attribute block* be the ith job, …, $(i + e)$th job of U, where $e \geq 1$ and all jobs in the block have identical attributes.

Theorem 3.1

It is optimal for all jobs in an attribute block of U to bypass an off-line buffer.

Proof

Let S^α be a feasible solution where a, …, $(a + b)$th job of D have attribute l (attribute block) and S^β be a feasible solution identical to S^α except sending the $(a + n)$th job of S^α to an off-line buffer and reinserting it at the mth position of D, $1 \leq n < b$, $a + b < m \leq N$. Furthermore, if we define

$$X_{jk}^{\ i} = \begin{cases} 1 & \text{for } S^i \text{, if the attribute of the } j^{\text{th}} \text{ job in } D \text{ is } k \\ 0 & \text{otherwise} \end{cases} \text{, then}$$

$X_{jl}^{\alpha} = X_{jl}^{\beta}$ for $1 \leq j < a + n,\ m < j \leq N$

$X_{jl}^{\alpha} = X_{j+1,l}^{\beta}$ for $a + n < j < m$.

Sending the $(a + n)$th job of U to an off-line buffer does not change the number of changeovers while reinserting at the mth position of D increases the number of changeovers by up to two. Because the sequence of S^α and S^β are identical except on $(a + n)$th and mth position of U,

the number of changeovers of S^α $\leq$ the number of changeovers of S^β.

Furthermore, by Theorem 2.3, the maximum number of jobs in an attribute block that can be reinserted outside the attribute block is one. Therefore, for any feasible solution sending any job of an attribute block to an off-line buffer, equally good or better solution can be found by forcing all jobs in the attribute block to bypass an off-line buffer. □

Using Theorem 3.1, all nodes (*, S) representing jobs in an attribute block can be removed from the shortest path problem network. Consequently, all arcs connected with those nodes can also be removed.

Theorem 3.2

It is optimal not to insert any job (with different attribute) inside an attribute block.

Proof

In addition to the definitions in the proof of Theorem 3.1, redefine S^{β} as a feasible solution where the attribute block in S^{α} is split by sending the mth job (with attribute other than l) of U to an off-line buffer and reinserting it at the $(a + n)$th position of D, $1 \leq m < a$, $1 \leq n < b$. Then

$X_{jl}^{\alpha} = X_{jl}^{\beta}$ for $1 \leq j < m, a + n < j \leq N$

$X_{jl}^{\alpha} = X_{j+1,l}^{\beta}$ for $m \leq j < a + n.$

Sending the mth job of U to an off-line buffer can decrease the number of changeovers by at most two while reinserting at the $(a + n)$th position of D increases the number of changeovers by two. Because the sequence of S^{α} and S^{β} are identical except on mth and $(a + n)$th position of U,

$\therefore$ the number of changeovers of S^{α} $\leq$ the number of changeovers of S^{β} □

Using Theorem 3.2, all instances of arc 3 and 4 that go from a job with different attribute to a job belonging to an attribute block can be removed from the shortest path problem network.

Finally, a feasible solution can be obtained by sequentially solving shortest path problem associated with each off-line buffer (of capacity 1) B times. Note that even though this heuristic algorithm is fast with $O(N^2B)$ complexity and based on local optimal solution from each stage (shortest path problem), there is no guarantee of performance or optimality for the constrained sequencing problem with an off-line buffer of capacity B.

3.2.2 Basic Sequence-dependent Setup Cost

The formulation discussed in Section 3.2.1 is for the generalized sequence-dependent setup cost. If setup cost depends only on the attribute of the job just finished, or only on the attribute of the job to be processed, the size of the formulation can be reduced. Such reduction is especially important because our formulation is NP-hard. First, an indicator variable u_{jk} needs to be defined:

$$u_{jk} = \begin{cases} 1 & \text{if } j^{\text{th}} \text{ job in } D \text{ has attribute } k \text{ while } j+1^{\text{th}} \text{ job in } D \text{ does not have attribute } k \\ 0 & \text{otherwise} \end{cases}$$

Then the MIP formulation can be modified as follows:

$$\text{Minimize} \quad \sum_{j=1}^{N-1}\sum_{k=1}^{K} C_k u_{jk} \qquad (5-1)$$

subject to

$$\sum_{j=1}^{N}\sum_{k=1}^{K} x_{ijk} = 1 \qquad \forall\, i = 1, \cdots, N \qquad (5-2)$$

$$\sum_{i=1}^{N}\sum_{k=1}^{K} x_{ijk} = 1 \qquad \forall\, j = 1, \cdots, N \qquad (5-3)$$

$$x_{ijk} = 0 \qquad \forall\, i,\ j,\ k$$
$$\text{where} \quad i - j > B, \quad i = 1, \cdots, N, \quad j = 1, \cdots, N, \quad k = 1, \cdots, K \qquad (5-4)$$

$$u_{jk} \geq \sum_{i=1}^{N} x_{ijk} - \sum_{i=1}^{N} x_{i,j+1,k} \qquad \forall\, j = 1, \cdots, N-1,\ k = 1, \cdots, K \qquad (5-5)$$

$$\sum_{j=1}^{N} x_{ijk} \leq A_{ik} \qquad \forall\, i = 1, \cdots, N,\ k = 1, \cdots, K \qquad (5-6)$$

$$x_{ijk} \text{ is binary} \qquad \forall\, i,\ j,\ k$$
$$\text{where} \quad i = 1, \cdots, N, \quad j = 1, \cdots, N, \quad k = 1, \cdots, K \qquad (5-7)$$

(5-5) makes $u_{jk} = 1$ only when jth job in D has attribute k while $j+1$th job in D does not have attribute k. Note that for all the other constrains except (5-5), the same constraints shown in Section 3.2.1 are used. Also note that there is no advantage of using the simplified cost structure discussed in Section 3.2.2 for the heuristic discussed in Section 3.2.1.

3.3 Model Customization

Since all models discussed in this dissertation use either MIP or LP formulation and the structure of these models are relatively simple, various ways of customizing the model can be found to meet the requirements specific to each application. In this Section a few examples of customizing the models are illustrated.

3.3.1 Generalization of the Model in Section 3.1.2

In Section 3.1.2 we discussed the case with explicit consideration of downstream queue addition cost and the setup cost with implicit assumption that any queue is subject to selection. However, in general cases there may be some queues that should be always used along with queues that may or may not be used. To model such a situation, the following constraint needs to be added:

$$w_{qi} = 0 \qquad q \neq i, \quad \forall\, q \in Q', \quad \forall\, i \in Q$$
$$Q' = \text{set of downstream queues that should be used}$$

By the definition of w_{qi}, it is clear that $w_{qi} = 1$ means queue q has been left out for use. Therefore we need to make sure that $w_{qi} = 0$ for all queues that should be used. Note that similar results can be found by setting $C''_q = 0, \quad \forall\, q \in Q'$.

CHAPTER 4

NUMERICAL RESULTS

Many different kinds of variables and parameters are used in defining and modeling constrained sequencing problem, making it difficult to explore all possible combinations. The case study at an automobile assembly plant showed that certain issues are more important than others in reality. Therefore, numerical experiments have been conducted focusing on answering the following questions:

(1) How the characteristics of the input sequence of jobs (number of attributes and number of jobs) affect the performance of the model?
(2) How is the solution affected by system characteristics, such as number of downstream queues, off-line queue capacity, and changeover costs?
(3) More specifically, how much cost saving can be realized by increasing the number of downstream queues and off-line queue capacity and it is ever desirable to have full re-sequencing flexibility?
(4) How computationally efficient is our solution approach, and is it amenable to environments where a solution must be obtained in several seconds?
(5) What are the performance characteristics of our models to the case study data?
(6) How good is the quality of the solutions obtained from the heuristic algorithm and how is it affected by problem characteristics? Is the use of the optimal solution justifiable compared to the solution from the heuristic algorithm?

4.1 Sensitivity Analysis on the Models

To answer questions (1) ~ (3), a series of experiments have been conducted for randomly generated problems with varying number of attributes, jobs, downstream queues on the formulation described in Section 3.1.1. Setup costs are assumed identical for any attribute changeover to minimize effects of setup costs on the model.

Experimental experiments results are shown in Table 2, Figure 16, 17, and 18. To minimize fluctuations of objective function value (due to different attribute distribution of the input sequence) and solution time (due to multi-user environment), 30 experiments have been made per each combination of parameter values, resulting in a total of 6600 experiments.

Table 2. Numerical Experiments Results for the Analytical Model (in # of Changeovers)

Number of Attributes	Number of Jobs	Number of Downstream Queues								
		1	2	3	4	5	6	7	8	9
3	100	65	22	0	0	0	0	0	0	0
	200	127	42	0	0	0	0	0	0	0
	300	197	63	0	0	0	0	0	0	0
	400	269	87	0	0	0	0	0	0	0
	500	342	120	0	0	0	0	0	0	0
4	100	75	35	12	0	0	0	0	0	0
	200	157	73	29	0	0	0	0	0	0
	300	228	104	42	0	0	0	0	0	0
	400	298	141	52	0	0	0	0	0	0
	500	369	176	66	0	0	0	0	0	0
5	100	85	47	23	10	0	0	0	0	0
	200	164	97	52	21	0	0	0	0	0
	300	242	126	68	28	0	0	0	0	0
	400	322	171	93	39	0	0	0	0	0
	500	404	223	123	50	0	0	0	0	0
6	100	85	49	28	14	6	0	0	0	0
	200	168	95	58	30	10	0	0	0	0
	300	255	157	97	54	23	0	0	0	0
	400	324	198	122	68	29	0	0	0	0
	500	417	247	149	85	33	0	0	0	0
7	100	88	56	39	23	11	3	0	0	0
	200	164	101	67	41	24	10	0	0	0
	300	268	167	106	64	34	12	0	0	0
	400	338	208	138	88	50	21	0	0	0
	500	409	255	166	106	59	25	0	0	0
8	100	78	52	34	24	15	8	3	0	0
	200	175	119	83	56	35	20	9	0	0
	300	248	159	110	74	51	31	13	0	0
	400	345	221	149	100	67	40	18	0	0
	500	441	295	203	136	87	48	20	0	0
9	100	87	65	47	34	25	18	11	5	0
	200	182	127	94	70	50	34	21	10	0
	300	273	186	132	96	69	45	25	10	0
	400	341	235	169	123	89	60	37	16	0
	500	438	303	221	159	111	73	42	19	0
10	100	92	63	45	34	26	19	13	8	3
	200	181	126	89	67	49	34	23	14	6
	300	264	190	144	111	83	59	39	23	11
	400	336	230	174	132	95	69	47	27	12
	500	447	312	236	179	131	90	58	34	16

In Table 2, column with one downstream queue means cases where no re-sequencing can be made. Note that if number of downstream queues is equal or bigger than the number of total attributes in the incoming sequence, the number of changeovers can be reduced to zero. Figure 17 shows that the number of changeovers reduces as the number of queues increases. Furthermore, Figure 18 shows data on reduced changeovers (compared to the original sequence) indicating that the benefit of adding additional downstream queue quickly diminishes as total number of downstream queues increases. This means that a limited number of downstream queues would be sufficient in general for reducing changeovers and that full re-sequencing capability (10 or more downstream queues in the following figures) would be rarely justified.

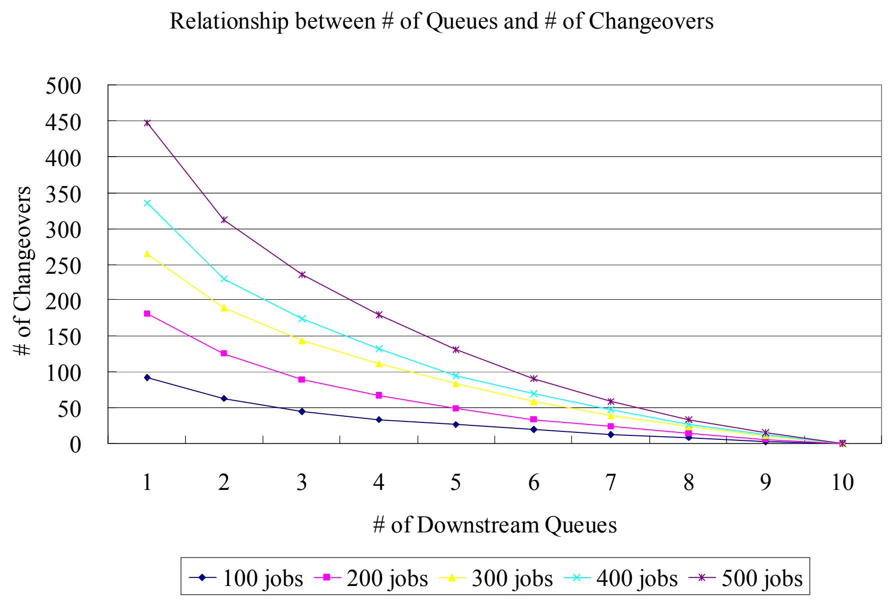

Figure 17. Relationship between # of Queues and # of Changeovers

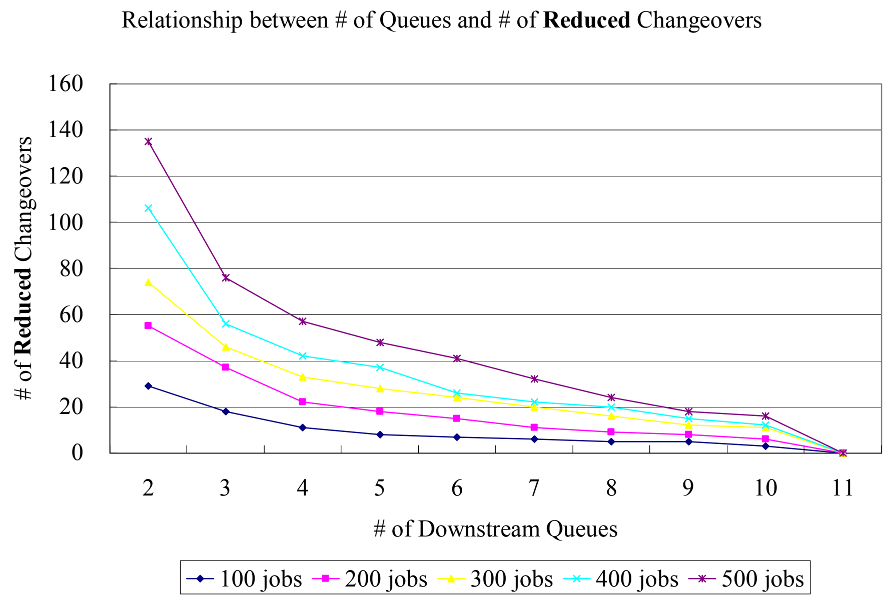

Figure 18. Relationship between # of Queues and # of Reduced Changeovers

Figure 19 shows that the number of attributes is positively correlated with the number of changeovers. This phenomenon can be explained as follows. Since all jobs were randomly generated with equal probability for all attributes, more number of attributes means there is less number of jobs in each attribute set if the number of jobs is fixed. As a result, the average distance between adjacent jobs with the same attribute increases. Finally, the number of changeovers increases because of reduced probability to find the job with the same attribute on dispatching.

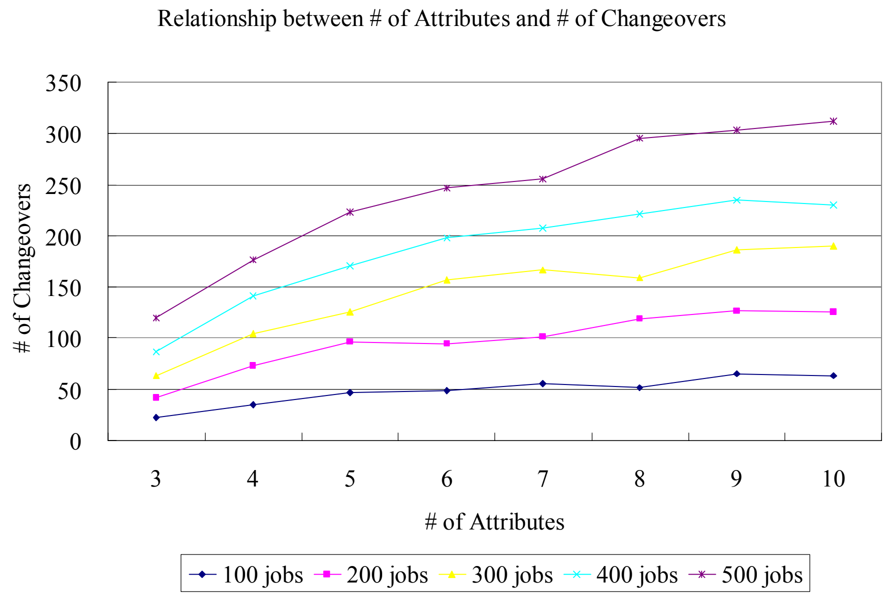

Figure 19. Relationship between # of Attributes and # of Changeovers

4.2 Solution Time

All experiments have been done on a multi-user Unix platform (Sun 280R with dual 900MHz UltraSparc-III-Cu CPU's and 2GB RAM) and CPLEX version 8.1.0 environment. CPLEX Solution times for varying problem sizes, number of attributes, and downstream queues are shown in Table 3.

Table 3. CPLEX Solution Times (in seconds)

Number of Attributes	Number of Jobs	Number of Downstream Queues								
		1	2	3	4	5	6	7	8	9
3	100	0.05	0.08	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	200	0.40	0.41	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	300	1.42	1.22	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	400	3.65	2.73	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	500	6.47	5.55	N/A	N/A	N/A	N/A	N/A	N/A	N/A
4	100	0.06	0.10	0.05	N/A	N/A	N/A	N/A	N/A	N/A
	200	0.44	0.54	0.39	N/A	N/A	N/A	N/A	N/A	N/A
	300	1.54	1.61	1.15	N/A	N/A	N/A	N/A	N/A	N/A
	400	4.29	3.79	2.62	N/A	N/A	N/A	N/A	N/A	N/A
	500	8.03	7.53	5.34	N/A	N/A	N/A	N/A	N/A	N/A
5	100	0.06	0.09	0.07	0.07	N/A	N/A	N/A	N/A	N/A
	200	0.49	0.60	0.50	0.39	N/A	N/A	N/A	N/A	N/A
	300	1.81	1.81	1.53	1.16	N/A	N/A	N/A	N/A	N/A
	400	4.52	4.30	3.50	2.60	N/A	N/A	N/A	N/A	N/A
	500	8.47	8.51	7.10	5.21	N/A	N/A	N/A	N/A	N/A
6	100	0.05	0.11	0.08	0.07	0.06	N/A	N/A	N/A	N/A
	200	0.48	0.65	0.59	0.50	0.40	N/A	N/A	N/A	N/A
	300	1.78	2.06	1.72	1.43	1.16	N/A	N/A	N/A	N/A
	400	4.97	4.85	4.12	3.32	2.70	N/A	N/A	N/A	N/A
	500	9.28	9.95	8.32	7.09	5.00	N/A	N/A	N/A	N/A
7	100	0.06	0.09	0.09	0.08	0.07	0.06	N/A	N/A	N/A
	200	0.54	0.67	0.63	0.55	0.48	0.41	N/A	N/A	N/A
	300	2.01	2.32	2.22	1.90	1.60	1.21	N/A	N/A	N/A
	400	5.19	5.24	4.69	4.19	3.56	2.48	N/A	N/A	N/A
	500	9.58	10.57	9.35	7.98	6.67	5.23	N/A	N/A	N/A
8	100	0.06	0.11	0.09	0.08	0.07	0.08	0.08	N/A	N/A
	200	0.49	0.71	0.63	0.59	0.54	0.51	0.41	N/A	N/A
	300	1.97	2.33	2.01	1.66	0.15	1.48	1.13	N/A	N/A
	400	5.17	5.40	4.84	4.23	3.87	3.28	2.72	N/A	N/A
	500	10.58	11.97	11.12	9.37	8.24	6.86	5.13	N/A	N/A
9	100	0.05	0.12	0.10	0.09	0.08	0.08	0.08	0.07	N/A
	200	0.53	0.75	0.64	0.63	0.59	0.56	0.54	0.48	N/A
	300	2.10	2.54	2.26	2.06	2.02	1.78	1.48	1.21	N/A
	400	5.33	5.35	5.18	4.78	4.24	3.82	3.67	2.92	N/A
	500	10.75	11.87	11.10	10.05	8.90	7.69	6.70	5.28	N/A
10	100	0.06	0.11	0.09	0.08	0.08	0.08	0.09	0.09	0.07
	200	0.53	0.78	0.76	0.72	0.67	0.61	0.53	0.52	0.46
	300	1.99	2.46	2.13	2.15	1.98	1.81	1.59	1.32	1.23
	400	5.51	5.60	5.04	4.96	4.51	3.97	3.79	3.31	2.65
	500	10.94	12.35	11.59	10.23	9.22	8.15	7.24	6.24	5.17

As expected, solution time is positively correlated with number of attributes (see Figure 20) and number of jobs (see Figure 21) increases. However, Figure 22 shows that solution time decreases as number of downstream queues increases.

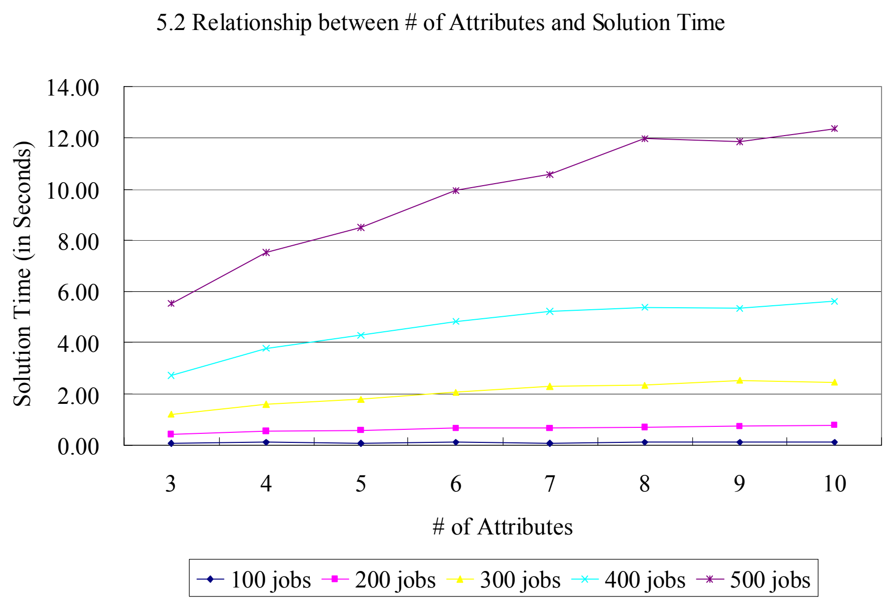

Figure 20. Relationship between # of Attributes and Solution Time

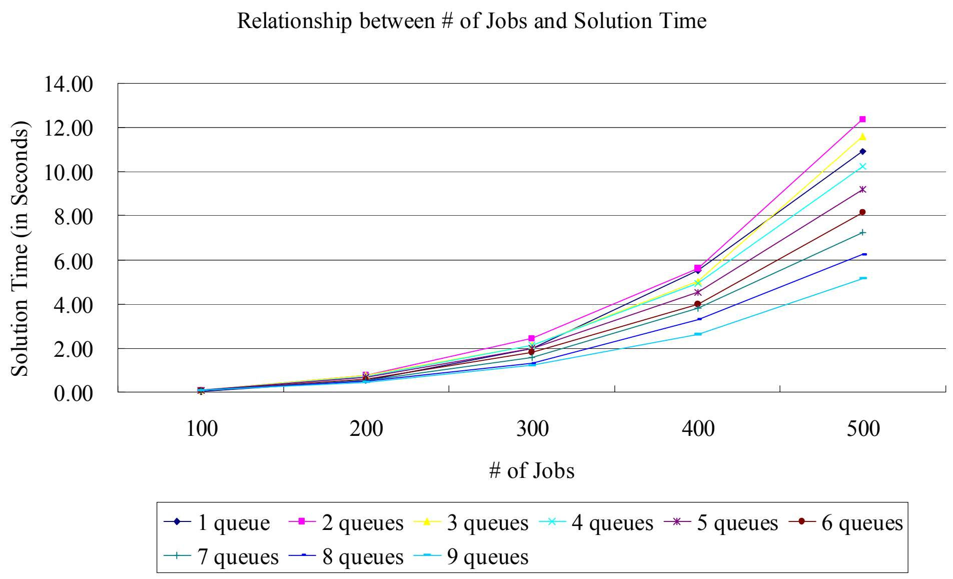

Figure 21. Relationship between # of Jobs and Solution Time

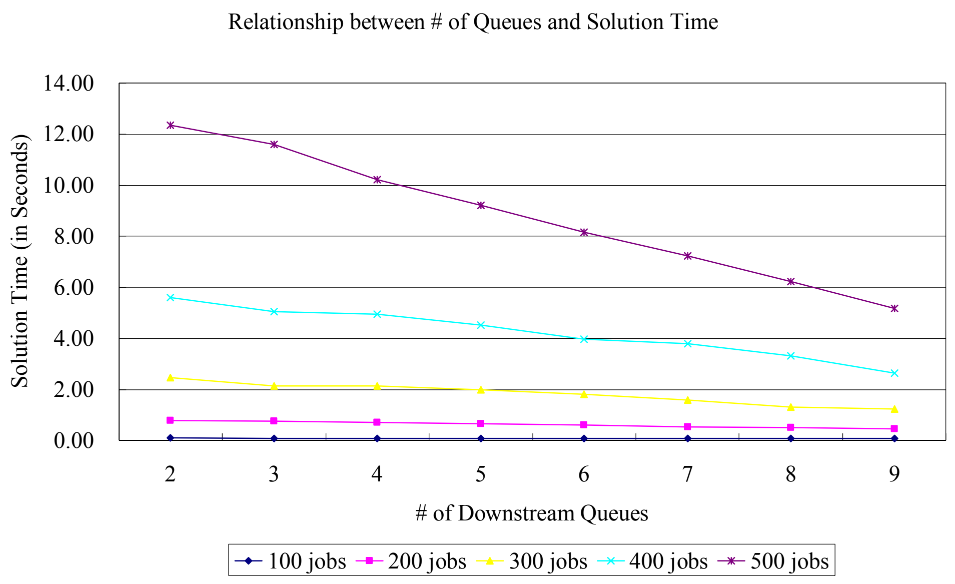

Figure 22. Relationship Between # of Queues and Solution Time

Note that the maximum solution time for 1000 jobs was kept within three minutes on several informal experiments that have been conducted. In summary, numerical results show that solution time is not a big concern for most practical applications including cases where decisions should be made on the spot.

4.3 Solution Quality of the Heuristic Algorithm

The numerical experiments results for the heuristic algorithm are shown in Table 4. In addition, Table 5 shows results of benchmarking heuristic algorithm-based solutions against optimal solutions in Table 1. Note that the numbers in Table 5 represent the increased percentage of changeovers (compared to the optimal solution) due to using the heuristic algorithm.

Table 4. Numerical Experiments Results for the Heuristic Algorithm (in # of Changeovers)

Number of Attributes	Number of Jobs	Number of Downstream Queues								
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
3	100	65	39	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	200	127	57	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	300	197	90	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	400	269	134	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	500	342	182	N/A	N/A	N/A	N/A	N/A	N/A	N/A
4	100	75	48	27	N/A	N/A	N/A	N/A	N/A	N/A
	200	157	109	50	N/A	N/A	N/A	N/A	N/A	N/A
	300	228	156	69	N/A	N/A	N/A	N/A	N/A	N/A
	400	298	200	99	N/A	N/A	N/A	N/A	N/A	N/A
	500	369	253	129	N/A	N/A	N/A	N/A	N/A	N/A
5	100	85	63	48	18	N/A	N/A	N/A	N/A	N/A
	200	164	128	78	44	N/A	N/A	N/A	N/A	N/A
	300	242	165	115	63	N/A	N/A	N/A	N/A	N/A
	400	322	246	159	76	N/A	N/A	N/A	N/A	N/A
	500	404	310	214	102	N/A	N/A	N/A	N/A	N/A
6	100	85	66	46	32	13	N/A	N/A	N/A	N/A
	200	168	124	94	61	29	N/A	N/A	N/A	N/A
	300	255	207	149	108	61	N/A	N/A	N/A	N/A
	400	324	254	194	118	61	N/A	N/A	N/A	N/A
	500	417	322	238	169	79	N/A	N/A	N/A	N/A
7	100	88	74	63	44	26	12	N/A	N/A	N/A
	200	164	142	109	89	48	32	N/A	N/A	N/A
	300	268	213	171	111	84	34	N/A	N/A	N/A
	400	338	280	222	161	116	47	N/A	N/A	N/A
	500	409	340	257	203	133	68	N/A	N/A	N/A
8	100	78	62	49	41	36	21	8	N/A	N/A
	200	175	151	127	99	72	44	24	N/A	N/A
	300	248	200	175	131	95	73	33	N/A	N/A
	400	345	282	234	189	133	102	51	N/A	N/A
	500	441	376	308	242	184	113	67	N/A	N/A
9	100	87	76	66	56	48	27	19	12	N/A
	200	182	159	139	112	95	73	51	19	N/A
	300	273	235	205	170	133	95	69	38	N/A
	400	341	296	245	209	172	122	88	38	N/A
	500	438	385	329	280	233	175	129	45	N/A
10	100	92	76	69	58	41	35	30	19	12
	200	181	154	125	109	84	69	48	31	19
	300	264	237	211	185	143	130	79	62	27
	400	336	281	250	217	182	143	100	65	38
	500	447	402	339	296	233	185	135	105	42

Table 5. Results of Benchmarking Heuristic Algorithm-based Solutions against Optimal Solutions (in Percentage)

Number of Attributes	Number of Jobs	Number of Downstream Queues							
		2	**3**	**4**	**5**	**6**	**7**	**8**	**9**
3	100	77%	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	200	36%	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	300	43%	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	400	54%	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	500	52%	N/A	N/A	N/A	N/A	N/A	N/A	N/A
4	100	37%	125%	N/A	N/A	N/A	N/A	N/A	N/A
	200	49%	72%	N/A	N/A	N/A	N/A	N/A	N/A
	300	50%	64%	N/A	N/A	N/A	N/A	N/A	N/A
	400	42%	90%	N/A	N/A	N/A	N/A	N/A	N/A
	500	44%	95%	N/A	N/A	N/A	N/A	N/A	N/A
5	100	34%	109%	80%	N/A	N/A	N/A	N/A	N/A
	200	32%	50%	110%	N/A	N/A	N/A	N/A	N/A
	300	31%	69%	125%	N/A	N/A	N/A	N/A	N/A
	400	44%	71%	95%	N/A	N/A	N/A	N/A	N/A
	500	39%	74%	104%	N/A	N/A	N/A	N/A	N/A
6	100	35%	64%	129%	117%	N/A	N/A	N/A	N/A
	200	31%	62%	103%	190%	N/A	N/A	N/A	N/A
	300	32%	54%	100%	165%	N/A	N/A	N/A	N/A
	400	28%	59%	74%	110%	N/A	N/A	N/A	N/A
	500	30%	60%	99%	139%	N/A	N/A	N/A	N/A
7	100	32%	62%	91%	136%	300%	N/A	N/A	N/A
	200	41%	63%	117%	100%	220%	N/A	N/A	N/A
	300	28%	61%	73%	147%	183%	N/A	N/A	N/A
	400	35%	61%	83%	132%	124%	N/A	N/A	N/A
	500	33%	55%	92%	125%	172%	N/A	N/A	N/A
8	100	19%	44%	71%	140%	163%	167%	N/A	N/A
	200	27%	53%	77%	106%	120%	167%	N/A	N/A
	300	26%	59%	77%	86%	135%	154%	N/A	N/A
	400	28%	57%	89%	99%	155%	183%	N/A	N/A
	500	27%	52%	78%	111%	135%	235%	N/A	N/A
9	100	17%	40%	65%	92%	50%	73%	140%	N/A
	200	25%	48%	60%	90%	115%	143%	90%	N/A
	300	26%	55%	77%	93%	111%	176%	280%	N/A
	400	26%	45%	70%	93%	103%	138%	138%	N/A
	500	27%	49%	76%	110%	140%	207%	137%	N/A
10	100	21%	53%	71%	58%	84%	131%	138%	300%
	200	22%	40%	63%	71%	103%	109%	121%	217%
	300	25%	47%	67%	72%	120%	103%	170%	145%
	400	22%	44%	64%	92%	107%	113%	141%	217%
	500	29%	44%	65%	78%	106%	133%	209%	163%

Table 5 shows that finding optimal solution is justifiable in most cases, while Table 4 indicates that the heuristic algorithm is still worth implementing if getting optimal solution is impossible or requires excessive initial investment cost. Note that the solution times for the heuristic algorithm were less than 0.7 second when implemented in Java and run on a Pentium IV PC with a Pentium 1.5 GHz processor.

4.4 Results of the Case Study Data

To verify if our solution methodology works well on the real world manufacturing system, data have been collected on a sequence of 4897 cars with a total of 9 colors going to the paint booth of the paint shop in an automobile manufacturing facility. Case study details are discussed in the next Chapter. Figure 23 shows that the introduction of the second downstream queue results in an average cost reduction of 41% and that only 19% of the maximum possible cost reduction is realized with the third downstream queue, with the incremental benefit of additional queues quickly diminishing. Because the total number of colors is 9, it is predicted that the number of reduced changeovers drops to zero for 10 or more downstream queues and Figure 22 confirms our prediction.

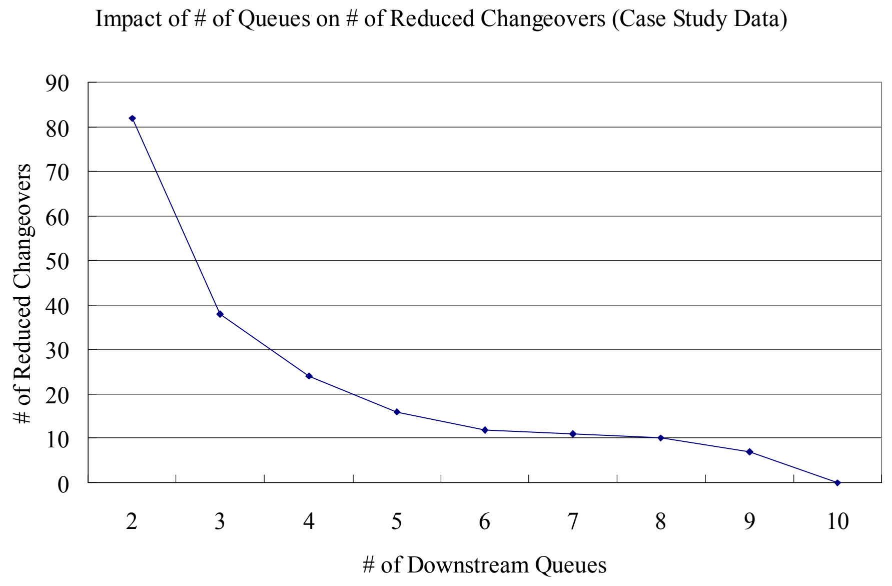

Figure 23. Impact of # of Queues on # of Reduced Changeovers (Case Study Data)

Note that the total cost reduction amount obtainable on the plant level is expected to be much higher since the plant in the case study has several diverging or converging conveyor points in the paint shop.

CHAPTER 5

CASE STUDY

The constrained sequencing problem on conveyors for setup reduction was motivated by a project with an automotive assembly plant in US. The initial objective of the project was to try to reduce the number of setups via changing the control logic at conveyor diverging / converging points because control logic change incurs relatively less interruption in production and is inexpensive in implementation. The long term objectives of the project include changing physical configuration of the plant as well as offering insights for future system designs. Due to system complexity, we initially observed and applied the existing control logic in the simulation model. We then built the mathematical model for each of the isolated junctions (diverging junctions and junctions with an off-line buffer) as discussed in the prior chapters and verified solution quality as well as applicability to real life production environment using the simulation model. In this chapter, we first report the initial case study and then apply our method to explore improvement.

5.1 Paint Purge Reduction in Automotive Assembly

Automotive manufacturing is a complex task involving many steps of machining and assembly. Among them, the painting process is an important part of the entire automobile manufacturing system. Because most car models offer multiple colors and orders normally consists of cars with different colors, frequent color change is unavoidable if the order sequence is to be maintained. However, color change in the painting process is expensive because of wasted paint, solvent (for removing residual paint from paint gun nozzles), and time. Moreover, the solvent as well as the paint often contain environmental pollutants such as Volatile Organic Compounds (VOCs).

There have been several approaches utilized in the automotive industry to deal with this problem. Some plants batch cars with the same color in the production order. This approach requires excessive storage space and equipment and as well as causing higher WIP and longer order cycle times. Therefore, some other plants try to reduce the time and/or cost of each color change setup. In our case study, the plant adopted the latter approach. They installed advanced equipment eliminating setup time to change the color. However, color change still results in wasted paint and solvent. Therefore, it is desirable to reduce the number of color changes by re-sequencing cars via conveyor junctions. Besides its original function of transporting cars among processing stations, the conveyor system can be also used for temporary buffering and re-sequencing cars to maximize average color block size, which is equivalent to minimizing the total number of color changes.

The paint shop consists of a few phases. Vehicles enter the first phase as solid sheet metal bodies (Body-In-White or BIW) that are fed from the body shop and exit the last phase painted (at several paint booths) and burnt (at a few ovens). The main processes include prime spray, prime oven, dry sand, enamel spray, and enamel oven. The bodies flow through these processes on conveyors.

Throughout the automotive assembly system, various processing steps on the vehicles such as parallel ovens and metal repair disrupt the original sequence as well as existing color blocks. In fact, all operations done in any assembly plant can introduce disruptions to the scheduled job sequence. Disruptions can be classified as controllable and uncontrollable. Controllable disruptions can be further classified as facility-related (i.e. assembly stations and paint booths), process-related (i.e. off-line processes), and operation-related (i.e. dedicated handling requirement of specific units). Uncontrollable disruptions are mostly related to repairs and rework operations. Painting is a delicate process prone to various quality problems. As a result, over half of the disruption points are located in the paint shop of an automobile assembly plant.

The system has over 10 diverging conveyor junctions including one in front of the prime ovens, the prime storage area, the enamel spray booths, and the enamel ovens to which the analytical models discussed in Section 3.1 can be applied. It also has more than 10 converging junctions and one FIFO off-line buffer. However, it does not have a random access off-line buffer, preventing us from applying the analytical model in Section 3.2 without conveyor configuration changes.

In terms of information technology infrastructure, the conveyor system in the paint shop is controlled by more than 20 PLCs, each managing one control point in the system.

Each PLC is linked to some sensors nearby to provide local view of the entire conveyor system in the paint shop. There is no data communication among the PLCs. The decision at each control point should be made within half minute of a status change because of the dynamically changing car sequence. Furthermore, the logic in the PLC is implemented as ladder diagram. Ladder diagram is convenient for logical operation but limited in computation, data handling, and communications. Therefore, implementing optimization algorithm requires additional hardware and interface, resulting in not being considered in the initial phase because of the associated investment cost.

5.2 Simulation Study

5.2.1 Need for Simulation Modeling

Simulation can accommodate much more realistic situations than that is possible with analytical models, such as time delays, control logic, parallel processing, shared resources, and variety of probability distributions to simulate randomness in the system. Therefore, simulation has been extensively used for simulating production activities including automotive production processes. Example of successful applications can be found in Park, Matson et al. (1998) and Graehl (1992). Ulgen and Gunal (1998) discusses the use of computer simulation in design and operation of automobile assembly plants as well as automotive components manufacturing plants. With its inherent ability for modeling randomness, computer simulation is an ideal tool for evaluating different rule sets and for predicting the throughput capability of a selectivity system. Simulation modeling provides an easier option for evaluating different scenarios without affecting the current operation of the actual system.

In general, the mathematically optimal solution of an analytical model may not be an optimal solution for the modeled system because of the abstraction in the modeling process. First, analytical model cannot address all aspects of the real system. One example is that the cycle time of each conveyor has been one of the top concerns of the engineers on the field while our analytical models are unable to handle any time-related constraint. Another example is the handling of rework. Second, analytical models are usually able to model partial system while optimizing the whole system is typically desired. In our case study, the entire painting processes can be thought of as a collection of conveyors connected via various junctions. While our analytical model yields the optimal or near-optimal solution for each subsystem composed of one diverging junction (with or without off-line buffer), the set of these local optimal solutions may not constitute the global optimal solution set of the whole system. Therefore, simulation results can be used to find the performance of the partial solutions from the analytical model. These limitations of the mathematical model make our simulation model indispensable for evaluating solutions including the solution from the mathematical model.

Simulation model can also be used to conduct what-if analysis and sensitivity analysis. For example, sometimes simply observing simulation animation may enable identifying bottlenecks in the manufacturing process flow. Furthermore, required changes to achieve a goal (e.g. getting the desired throughput) can be identified manually (by trial and errors) or automatically (systematic parameter optimization functionality that some commercial simulation software packages supports) through repeated simulation runs with different set of parameter values.

Finally, since physical change in existing conveyor system is very expensive considering equipment downtime and installation cost, the use of simulation model to study the effect of system changes is indispensable even though building a high-fidelity simulation model itself is a large undertaking. However, state-of-the-art simulation software packages provide sophisticated programming constructs and abilities allowing intricate operating details of such complex systems to be modeled with relative ease and accuracy (Jayaraman, Narayanaswamy et al. (1997)).

5.2.2 Simulation Model Details

5.2.2.1 Input Data

In our case study, we need information on conveyor configuration, incoming car sequence, processing time on each workstation, and rework. For conveyor configuration information, the plant CAD file was linked to the simulation model for dimensional accuracy. For car sequence information, the plant uses in-line sensors to identify the color of the incoming car. However, the plant database stores only the frequency distribution of colors, not the sequence of the car colors that we need. Therefore, we manually collected the color sequence data for 4897 cars – equivalent to 82 hours of production volume. The color frequency distribution of the manually collected data matched historical data stored in the database. Based on this, we assume that the collected sequence is representative of long-run averages. We then create pseudo random sequences based on the color distribution and use them in the simulation model. All other information, including data on conveyor speed and processing time for each workstation, we used the data derived from the plant database. However, we note that the simulation model shows that the number of reduced color changeovers is slightly bigger when manually collected sequence data are used (compared to the pseudo random sequence). It is suspected that such difference is due to the fact that pseudo random sequence is independent and identically distributed while the actual sequence manually collected data was not.

5.2.2.2 Simulation Model Language Selection

AutoMod was chosen as the simulation model platform mainly because its 3D animation capability, convenience for simulating manufacturing processes and support for customizable control logic in junction points although it does not provide connectivity to general-purpose programming languages such as Java, C/C++, or Visual Basic. However, the limited embedded programming language for model customization in AutoMod caused more than half of our model development time spent on developing and debugging user-defined libraries (written in the embedded programming language of AutoMod) and it is believed that at least some of these libraries, or APIs (Application Programming Interfaces) for general-purpose languages need to be provided by the software vendor. Figure 24 and Figure 25 shows our simulation model screenshots.

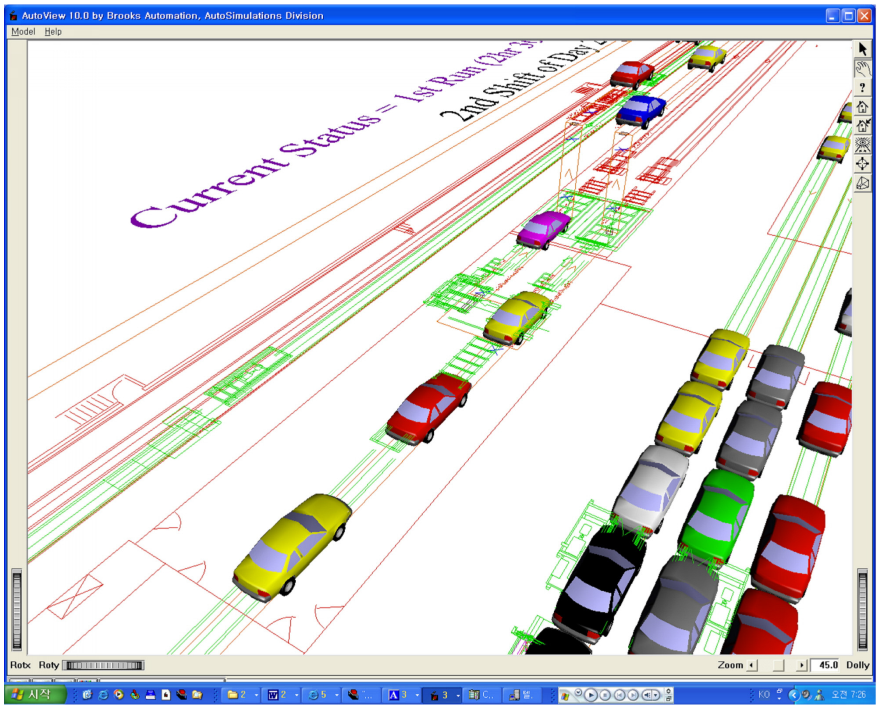
AutoView 10.0 by Brooks Automation, AutoSimulations Division
Model Help
Current Status = 1st Run (2hr 3
2nd Shift of Day 1
Rotx Roty
Zoom 45.0 Dolly

Figure 24. Simulation Model Screenshot - Prime Storage Area

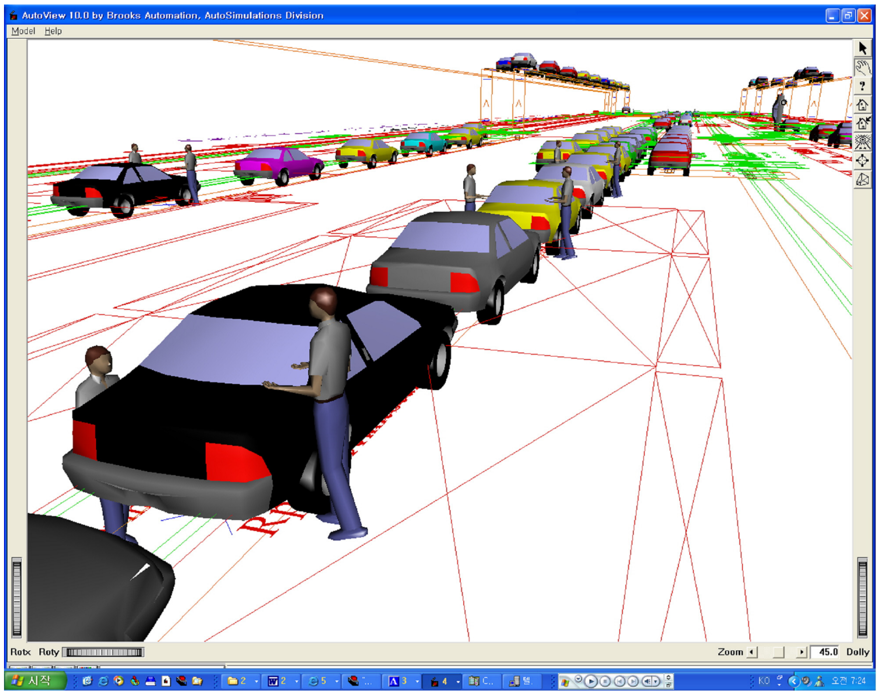
AutoView 10.0 by Brooks Automation, AutoSimulations Division
Model Help
Rotx Roty
Zoom 45.0 Dolly

Figure 25. Simulation Model Screenshot - Prime Scuff Area

5.3 Proposed Changes and Effect

The diverging control point before the enamel ovens was selected for this case study. Because of the practical difficulties in obtaining an optimal solution as well as in implementing PC-based algorithm discussed in Section 5.1, in addition to the analytical model discussed in Chapter 3, we also developed heuristic algorithm for the diverging conveyor junction and evaluated it using the simulation model. The current heuristic algorithm is described as follows:

Let *L* be the set of last jobs sent to each of the downstream queues.

Step 1. For each job at the exit of *U*, search *L* for the same attribute.

Step 2. If a job with the same attribute is found in *L*, send the current job to that downstream queue and go to Step 4.

Step 3. If no job with the same attribute is found in *L*, send the current job to a downstream queue containing the minimum number of jobs.

Step 4. Repeat this algorithm until no job is left in *U*.

Our revised heuristic algorithm is as follows:

Let *L2* be the set of second to last jobs sent to each downstream queue.

Step 1. For each job at the exit of *U*, search *L* for the same attribute.

Step 2. If a job with the same attribute is found in *L*, send the current job to that downstream queue and go to Step 5.

Step 3. If no job with the same attribute is found in *L*, search *L2* for the same attribute.

Step 4. If a job with the same attribute is found in *L2*, send the current job to a downstream queue containing the minimum number of jobs, as long as this queue is the queue associated with the job found in *L2*.

Step 5. Repeat this algorithm until no job is left in *U*.

Figure 26 is the simulation run results using the randomly generated data of 500 cars with 9 colors and 2 downstream conveyors. They show the number of changeovers resulting from the original sequence, the existing heuristic, the revised heuristic and the analytical model in Section 3.1.1. The result show that the existing algorithm reduced the change over by 12%, the revised algorithm by 20% and the optimization model by 31%. Therefore, the revised algorithm is a good alternative to using the analytical model, especially when getting the optimal solution is not possible (in the case of using a PLC).

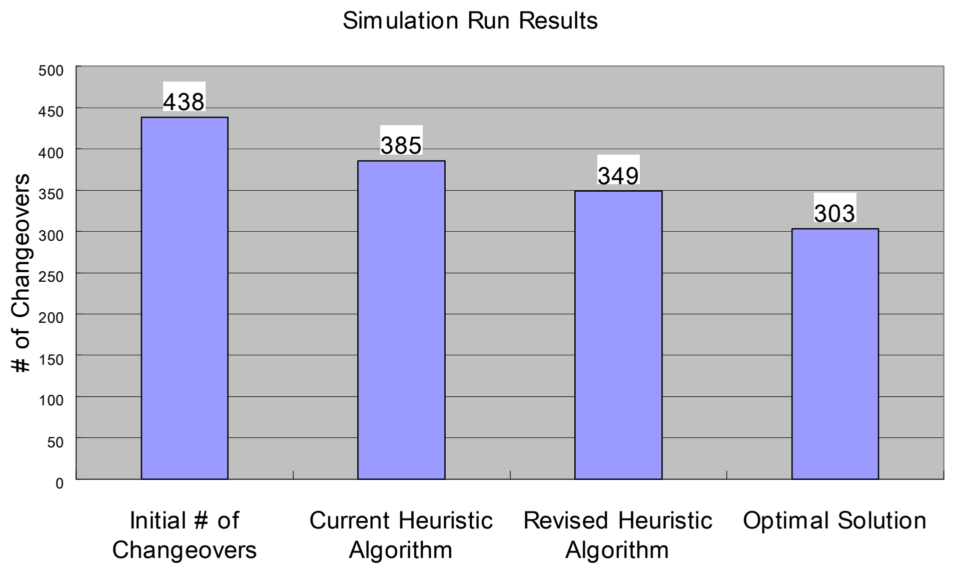

Figure 26. Simulation Run Results

CHAPTER 6

CONTRIBUTIONS AND FUTURE DIRECTIONS

6.1 Contributions

This dissertation is the first attempt to define, classify, and model the constrained sequencing problem and to generalize the problem on various assumptions. Specifically, analytical models have been developed for constrained sequencing problem with a diverging junction as well as with random access off-line buffer. We also modeled cases where number of downstream queues needs to be decided simultaneously as well as cases where setup cost depends on both the job just finished and the job to be processed next. It has been proved that problems with practical size such as several hundred jobs can be solved quickly by these models. In addition, for the constrained sequencing problem with an off-line buffer, special problem structure has been identified and a practical solution algorithm based on that structure has been developed.

The numerical experiments on those models helped revealing the characteristics of the models. The numerical experiment results showed that our formulations could be used for real time control. A case study also has been conducted on the paint shop project in an automotive plant to verify the validity of our approach on the manufacturing environment. Finally, a discrete event simulation model has been developed. In the case study environment, this model has been used to test solutions from our analytical models as well as from other heuristic algorithms.

6.2 Future Research

Even though this dissertation did extensive analysis on diverging junction cases and one off-line buffer cases with random access principle, no analysis has been made on converging junction cases as well as off-line buffer cases with other disciples such as FIFO. If making an optimization model for converging junction cases can be done successfully, a whole conveyor system can be analyzed systematically by dividing the system into each diverging or converging junction and applying the appropriate model to each junction. This way heuristics can be devised to obtain a feasible solution for the whole system and the performance of the heuristics may be good because the solution is composed of local optimal solutions for each junction point.

Other constraints such as those on cycle time limitations and line balancing also need to be integrated into the models discussed in this research. In the case study, reducing the number of color changeovers (i.e. increasing average color block size) was usually not on the highest priority in making production decisions. In other words, the solution optimized only for reducing the number of changeovers is very likely to be declined on implementation stage because performance of other criteria (that are considered more important than the number of changeovers) would be degraded greatly.

Finally, using diverging or converging junction alone may not be the best solution for all cases requiring re-sequencing, especially when full re-sequencing capability is desired. In fact, many manufacturing facilities use AS/RS or selectivity bank for re-sequencing purpose. However, since it believed that the approach of using junctions for re-sequencing is almost always beneficial even with the existence of AS/RS or selectivity bank, the following two functionalities need to be included in the analytical models as well as the simulation model discussed in this research:

1) Optimal capacity of AS/RS or selectivity bank in facility design phase needs to be decided. Since AS/RS or selectivity bank is expensive, minimizing its capacity is desired and use of junctions can greatly contribute to reducing required capacity while meeting re-sequencing requirements.
2) Optimal dispatching decision needs to be made on all diverging/converging conveyor junctions as well as on all diverging points (where jobs can go to AS/RS or to the next processing facility) and all converging points (where jobs can be received from AS/RS or from the previous processing facility).

APPENDIX A: MODELS FOR CONSTRAINED SEQUENCING & ATTRIBUTE ASSIGNING PROBLEM

In many cases, some attributes can be assigned at the time of being transferred to the downstream queues. In other cases, an attribute, such as the earmarked color of a car, can be swapped with another car with different color later in the sequence, to further reduce the number of setups. If the assumption that attributes of the incoming sequence are given in advance is relaxed, the number of setups can be reduced further. However, in such cases the decision making process is more complex because decisions how to assign or swap attributes for the incoming sequence also need to be made. To model the above situation, subscripts need to be added to the old definition of x_{ij} to designate a pair of jobs with a pair of earmarked attributes:

$$x_{ijkl} = \begin{cases} 1 & \text{if } i^{\text{th}} \text{ job with attribute } k \text{ in } U \text{ is located right before the } j^{\text{th}} \text{ job} \\ & \text{with attribute } l \text{ in } U \text{ on a downstream queue} \\ 0 & \text{otherwise} \end{cases}$$

The main reason for extending the dimension of x_{ij} into x_{ijkl} is that the attributes of the adjacent jobs in downstream queues need to be considered explicitly because they are dynamically assigned or swapped during problem solving process. Note that $x_{Njkl} = 0$ is undefined for all j, k, and l. In addition, a definition needs to be made as follows:

$$A'_{ik} = \begin{cases} 1 & \text{if attribute } k \text{ can be assigned to } i^{\text{th}} \text{ job in } U \\ 0 & \text{otherwise} \end{cases}$$

Finally, y_{iq} and z_{qj} need to be redefined as follows:

$$y_{jlq} = \begin{cases} 1 & \text{if } j^{\text{th}} \text{ job with attribute } l \text{ in } U \text{ is the last item to be sent to a downstream queue} \\ 0 & \text{otherwise} \end{cases}$$

$$z_{qik} = \begin{cases} 1 & \text{if } i^{\text{th}} \text{ job with attribute } k \text{ in } U \text{ is the first item to be sent to a downstream queue} \\ 0 & \text{otherwise} \end{cases}$$

The relationship among the above redefined variables in the model is illustrated in an example shown in Figure 27 where queue *a* receives job 2 and 4 while queue *b* receives job 2, 3, and 5 sequentially.

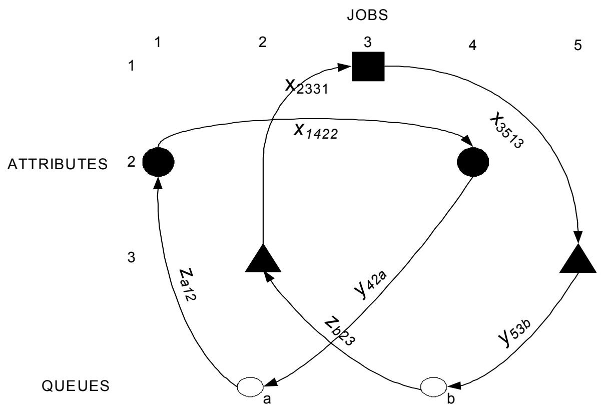

Figure 27. Network Relationship among Redefined Variables

Then the MIP based on extended network representation can be formulated as follows. Note that in this formulation the setup cost depends on both the attribute of the job just finished and the attribute of the job to be processed next.

$$\textit{Minimize} \quad \sum_{i=1}^{N-1}\sum_{j=i+1}^{N}\sum_{k=1}^{K}\sum_{l=1}^{K} C'_{kl}x_{ijkl} + \frac{1}{2}\sum_{q=1}^{Q} C''_q\Big(\sum_{i=1}^{N}\sum_{k=1}^{K} y_{ikq} + \sum_{j=1}^{N}\sum_{l=1}^{K} z_{qjl}\Big) + \frac{1}{2}\sum_{i=1}^{Q}\sum_{j=1, j\neq i}^{Q} w_{ij}(C''_j - C''_i)$$

subject to

$$\sum_{j=i+1}^{N}\sum_{k=1}^{K}\sum_{l=1}^{K} x_{ijkl} + \sum_{k=1}^{K}\sum_{q=1}^{Q} y_{ikq} = 1 \qquad \forall\, i = 1, \cdots, N \quad (3-2)$$

$$\sum_{i=1}^{j-1}\sum_{k=1}^{K}\sum_{l=1}^{K} x_{ijkl} + \sum_{l=1}^{K}\sum_{q=1}^{Q} z_{qjl} = 1 \qquad \forall\, j = 1, \cdots, N \quad (3-3)$$

$$\sum_{i=1}^{N}\sum_{k=1}^{K} y_{ikq} + \sum_{i=1, i\neq q}^{Q} w_{iq} = 1 \qquad \forall\, q = 1, \cdots, Q \quad (3-4)$$

$$\sum_{j=1}^{N}\sum_{l=1}^{K} z_{qjl} + \sum_{j=1, j\neq q}^{Q} w_{qj} = 1 \qquad \forall\, q = 1, \cdots, Q \quad (3-5)$$

$$\sum_{j=i+1}^{N}\sum_{l=1}^{K} x_{ijkl} \le A'_{ik} \qquad \forall\, i = 1, \cdots, N-1, \quad k = 1, \cdots, K \quad (3-6)$$

$$\sum_{h=1}^{i-1}\sum_{g=1}^{K} x_{higk} + \sum_{l=1}^{K}\sum_{q=1}^{Q} z_{qil} = \sum_{j=i+1}^{N}\sum_{l=1}^{K} x_{ijkl} + \sum_{k=1}^{K}\sum_{q=1}^{Q} y_{ikq} \qquad \forall\, i = 1, \cdots, N, \quad k = 1, \cdots, \quad (3-7)$$

$$x_{ijkl}, y_{ikq}, z_{qjl} \in \{0, 1\} \qquad \forall\, i, j, k, l \quad (3-8)$$

(3-6) is for restricting the range of values that ith job in U can take. $\sum_{q=1}^{Q} y_{iq} = 1$ for any i means that ith job is the last job to be sent to a downstream queue and by (3-2), $x_{ijkl} = 0$ for all j, k, and l. If $\sum_{q=1}^{Q} y_{iq} = 0$, by (3-6) x_{ijkl} is forced to be zero for all j and l when $A_{ik} = 0$. Note that (3-6) can be generalized by being replaced by $x_{ijkl} \le A'_{ijkl}, \ \forall\, i, j, k, l$. This alternative enables more sophisticated control on the range that x_{ijkl} can take, but it also increases the number of constraints by $\frac{(N-1)K(NK-2)}{2}$. (3-7) guarantees that exactly one attribute is assigned to each job in the incoming sequence. Contrary to models in Section 3.1.1 and 3.1.2, (3-8) cannot be removed in this model because there is no guarantee that LP relaxation of the above formulation has integer optimal solution. All the other constraints in this formulation are simply the extended version of the counterparts found in the formulation of Section 3.1.1 and 3.1.2 where detailed explanation of each constraint is available.

The above formulation assumes that the setup cost depends on both the attribute of the job just finished and the attribute of the job to be processed. However, sometimes the setup cost depends only on the attribute of the job just finished, or only on the attribute of the job to be processed. In either case, the formulation size can be reduced by modifying the formulation and notations. Note that the total number of variables of the formulation is reduced by $\frac{N(N-1)K(K-1)}{2}$ while the number of constraints remains the same. First, variables and constant are defined as follows:

$$x_{ijk} = \begin{cases} 1 & \text{if } i^{\text{th}} \text{ job with attribute } k \text{ in } U \text{ is sent to } j^{\text{th}} \text{ position in } D \\ 0 & \text{otherwise} \end{cases}$$

C_k = changeover cost from attribute k

$$t_{ijk} = \begin{cases} 1 & \text{if } i^{\text{th}} \text{ job with attribute } k \text{ in } U \text{ is located right before the } j^{\text{th}} \text{ item} \\ & \text{(attribute of } j^{\text{th}} \text{ item is not } k\text{) in } U \text{ on a downstream queue} \\ 0 & \text{otherwise} \end{cases}$$

Then the formulation can be modified as follows:

$$\text{Minimize} \quad \sum_{i=1}^{N-1}\sum_{j=i+1}^{N}\sum_{k=1}^{K} C_k t_{ijk} + \frac{1}{2}\sum_{q=1}^{Q} C_q''(\sum_{j=1}^{N} y_{jq} + \sum_{i=1}^{N} z_{qi}) + \frac{1}{2}\sum_{i=1}^{Q}\sum_{j=1}^{Q}(C_j'' - C_i'')w_{ij} \qquad (3'-1)$$

subject to

$$\sum_{j=i+1}^{N}\sum_{k=1}^{K} x_{ijk} + \sum_{q=1}^{Q} y_{iq} = 1 \qquad \forall\, i = 1, \cdots, N \qquad (3'-2)$$

$$\sum_{i=1}^{j-1}\sum_{k=1}^{K} x_{ijk} + \sum_{q=1}^{Q} z_{qj} = 1 \qquad \forall\, j = 1, \cdots, N \qquad (3'-3)$$

$$\sum_{j=1}^{N} y_{jq} + \sum_{j=1/q}^{Q} w_{jq} = 1 \qquad \forall\, q = 1, \cdots, Q \qquad (3'-4)$$

$$\sum_{i=1}^{N} z_{qi} + \sum_{i=1/q}^{Q} w_{qi} = 1 \qquad \forall\, q = 1, \cdots, Q \qquad (3'-5)$$

$$t_{ijk} \ge x_{ijk} - \sum_{m=j+1}^{N} x_{jmk} \qquad \forall\, i,\ j,\ k$$

$$\textit{where} \quad i < j, \quad i = 1, \cdots, N, \quad j = 1, \cdots, N, \quad k = 1, \cdots, K \qquad (3'-6)$$

$$\sum_{j=i+1}^{N} x_{ijk} \le A_{ik} \qquad \forall\, i = 1, \cdots, N-1, \quad k = 1, \cdots, K \quad (3'-7)$$

$$\sum_{h=1}^{i-1}\sum_{g=1}^{K} x_{higk} + \sum_{l=1}^{K}\sum_{q=1}^{Q} z_{qil} = \sum_{j=i+1}^{N}\sum_{l=1}^{K} x_{ijkl} + \sum_{k=1}^{K}\sum_{q=1}^{Q} y_{ikq} \qquad \forall\, i = 1, \cdots, N, \quad k = 1, \cdots, K \quad (3'-8)$$

$$x_{ijk},\ y_{jq},\ z_{qi} \text{ is binary} \qquad \forall\, i,\ j,\ k,\ q \qquad (3'-9)$$

(3’-6) is for checking attribute change. Because of (3’-1), (3’-2), and (3’-6), indicator variable t_{ijk} becomes one for any possible combination *i*, *j*, and *k* only when $x_{ilk} = 1$ and $\sum_{m=j+1}^{N} x_{jmk} = 0$ (in all other cases t_{ijk} becomes zero because of (3’-1)). In other words, t_{ijk} becomes one only when the following two conditions are met:

1. the queue is re-sequenced so that job *i* with attribute *k* is right before job *j* in a downstream queue ($x_{ijk} = 1$), and

2. attribute *k* is not assigned to job *j* ($\sum_{m=j+1}^{N} x_{jmk} = 0$).

Both of the above two formulations can also be modified for solving *constrained sequencing and swapping problem* defined as follows. Let us consider a constrained sequencing problem with determined attributes (as in Section 3.1.1 and 3.1.2). It is assumed that each job has additional attribute (denoted as *secondary attribute*) other than the attribute associated with setup cost (denoted as *primary attribute*). Then one can think of a set of jobs that have different primary attributes and identical secondary attribute. If "swapping" among the jobs in the incoming sequence is allowed, then the number of setups may be reduced further.

One example found on the manufacturing environment is the color change reduction problem (as described in Chapter 5) where number of color changes can be further reduced by swapping two cars in the incoming sequence if the colors to be painted (primary attribute) are different and all the other options (secondary attribute) are identical for these two cars. This method is used to alter the sequence of vehicles by changing their identification rather than physically switching their location. Distribution of secondary attributes directly impacts the effectiveness of vehicle swapping. At several points in the assembly process, a plant will read vehicle identifications and automatically swap vehicle identification tags. Tag swapping can be done physically, or electronically if RF (Radio Frequency) tags are used. Myron (1996) and Atassi (1996) report use of swapping technique at Ford Motor Company's Wixom Assembly Plant to create a new sequence that more closely matches the National Blend, Ford's way of making master production schedule to sequence cars to balance the workload in the final production area called trim and chassis.

To model the constrained sequencing and swapping problem, the following constraint needs to be added to the formulations in (3’–1) ~ (3’–9):

$$\sum_{i \in I_s} \sum_{j=i+1}^{N} \sum_{l=1}^{K} x_{ijkl} = |I_s| \qquad \forall\, s \in S,\ k \in K_s \text{, where} \tag{3-9}$$

S = secondary attribute set of jobs in U,

I_s = set of jobs having secondary attribute s in U,

$|I_s|$ = cardinality of I_s, and

K_s = primary attribute set of jobs in I_s.

Constraint (3-9) can be interpreted as follows. Since swapping among jobs having different primary attributes and the same secondary attribute is allowed, (3-9) needs to be added to make sure that for each primary attribute, the total number of jobs having that primary attribute does not change after swapping.

APPENDIX B: MODEL FOR DIVERGING JUNCTION & OFF-LINE BUFFER

By combining the results obtained in Section 3.1 and 3.2, general cases where diverging junctions and off-line buffers are used simultaneously can be modeled. The formulation discussed in this Section models the case where attributes of all jobs in the upstream queue are fixed. Notations in the previous Sections were reused except newly defining variable x_{hijk} as follows:

$$x_{hijk} = \begin{cases} 1 & \text{if } h^{\text{th}} \text{ job with attribute } k \text{ in } U \text{ is located right before the } j^{\text{th}} \text{ job in } U \text{ on a} \\ & \text{downstream queue (index } i \text{ represents temporary position in offline buffer)} \\ 0 & \text{otherwise} \end{cases}$$

The basic idea of modeling both off-line buffer and diverging conveyors is that the model for off-line buffer and the model for diverging conveyors can be combined without loss of generality. Specifically, the model for diverging conveyors in Section 3.1 enables us to find the optimal solution for a given input sequence while the model for off-line buffer in Section 3.2 allows us to identify all the possible downstream sequence for an initial input sequence. In addition, the only relationship between the above two models is that the output of the off-line buffer model becomes the input of the diverging queues model because jobs stored in the off-line buffer need to be released to one of downstream queues. Therefore, combining these two models can successfully identify optimal solution for the given initial sequence in upstream queue. Figure 28 shows how the relationship between the model for off-line buffer and the model for diverging conveyors can be used for developing integrated model for both off-line buffer and diverging queues.

Figure 27 is followed by the resulting formulation. It is assumed that setup cost depends only on the attribute of the job to be changed and that attributes of jobs in the input sequence are fixed for simpler formulation.

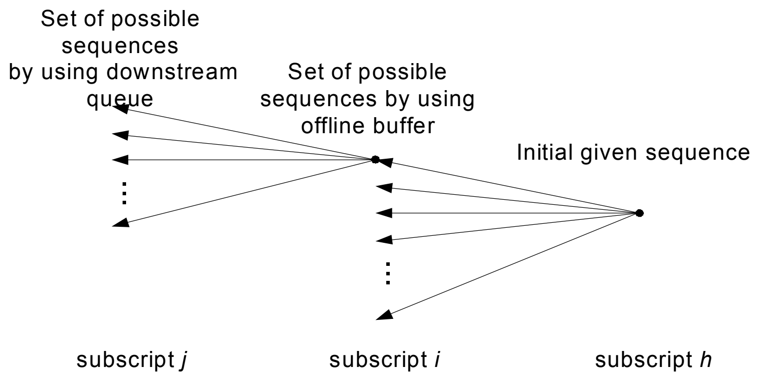
Set of possible sequences by using downstream queue
Set of possible sequences by using offline buffer
Initial given sequence
subscript *j*
subscript *i*
subscript *h*

Figure 28. Relationship between Off-line Buffer Model and Diverging Queues Model

$$Minimize \sum_{h=1}^{N-1}\sum_{i=h+1}^{N}\sum_{k=1}^{K} C_k v_{hik} + \frac{1}{2}\sum_{q=1}^{Q} C_q''(\sum_{j=1}^{N} y_{jq} + \sum_{i=1}^{N} z_{qi}) + \frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{M} w_{ij}(C_j'' - C_i'') \qquad (6-1)$$

subject to

$$\sum_{i=1}^{N}\sum_{j=i+1}^{N}\sum_{k=1}^{K} x_{hijk} = 1 \qquad \forall\ h = 1, \cdots, N \qquad (6-2)$$

$$\sum_{h=1}^{N}\sum_{j=i+1}^{N}\sum_{k=1}^{K} x_{ijk} = 1 \qquad \forall\ i = 1, \cdots, N \qquad (6-3)$$

$$\sum_{j=i+1}^{N} x_{hijk} = 0 \qquad \forall\ h,\ i,\ k$$

$$where \quad h - i > B, \quad h = 1, \cdots, N, \quad i = 1, \cdots, N, \quad k = 1, \cdots, K \qquad (6-4)$$

$$\sum_{i=1}^{N}\sum_{j=i+1}^{N} x_{hijk} \le A_{hk} \qquad \forall\ h = 1, \cdots, N,\ k = 1, \cdots, K \qquad (6-5)$$

$$\sum_{h=1}^{N}\sum_{j=i+1}^{N}\sum_{k=1}^{K} x_{hijk} + \sum_{q=1}^{Q} y_{iq} = 1 \qquad \forall\ i = 1, \cdots, N \qquad (6-6)$$

$$\sum_{h=1}^{N}\sum_{i=1}^{j-1}\sum_{k=1}^{K} x_{hijk} + \sum_{q=1}^{Q} z_{qj} = 1 \qquad \forall\ j = 1, \cdots, N \qquad (6-7)$$

$$\sum_{j=1}^{N} y_{jq} + \sum_{j=1/q}^{Q} w_{jq} = 1 \qquad \forall\ q = 1, \cdots, M \qquad (6-8)$$

$$\sum_{i=1}^{N} z_{qi} + \sum_{i=1/q}^{M} w_{qi} = 1 \qquad \forall\ q = 1, \cdots, M \qquad (6-9)$$

$$v_{ijk} \ge \sum_{h=1}^{N} x_{hijk} - \sum_{h=1}^{N}\sum_{m=j+1}^{N} x_{hjmk} \qquad \forall\ i,\ j,\ k$$

$$where \quad i < j, \quad i = 1, \cdots, N, \quad j = 1, \cdots, N, \quad k = 1, \cdots, K \qquad (6-10)$$

$$x_{hijk}, u_{ik}, y_{jq}, z_{qi} \text{ is binary} \qquad \forall\ h,\ i,\ j,\ k,\ q \qquad (6-11)$$

APPENDIX C: PROOF OF THEOREM 2.2

Theorem 2.2

Constrained sequencing problem with an off-line buffer of capacity *B* is equivalent to constrained sequencing problem with *B* off-line buffers of capacity 1.

Proof

In constrained sequencing problem with *B* off-line buffers of capacity 1, incoming job sequence goes through *B* buffers sequentially. *x* is redefined as follows:

$$x_{i_a j_b k} = \begin{cases} 1 & \text{if } i^{\text{th}} \text{ job with attribute } k \text{ of the incoming sequence to the junction} \\ & \text{with } a^{\text{th}} \text{ offline buffer is in } j^{\text{th}} \text{ position of the incoming sequence to} \\ & \text{the junction with } b^{\text{th}} \text{ offline buffer} \\ 0 & \text{otherwise} \end{cases}$$

,where $i_a = 1,\cdots,N,\quad j_b = 1,\cdots,N,\quad k = 1,\cdots,K,\quad a = 1,\cdots,B,\quad b = 1,\cdots,B+1$ (j_{B+1} means j^{th} position of *D*) . Then by theorem 2.1,

$$x_{h_{B-1} i_B k} = 0 \quad \forall\ h_{B-1}, i_B, k \quad \text{where } h_{B-1} > 1 + i_B\,. \tag{4-8}$$

Since the outgoing sequence from the junction with (*B-1*)th off-line buffer is the incoming sequence to the junction with *B*th off-line buffer,

$$x_{i_B j_{B+1} k} = 0 \quad \forall\ i_B, j_{B+1}, k \quad \text{where } i_B > 1 + j_{B+1}\,. \tag{4-9}$$

By (4-8) and (4-9),

$$x_{h_{B-1} j_{B+1} k} = 0 \quad \forall\ h_{B-1}, j_{B+1}, k \quad \text{where } h_{B-1} > 2 + j_{B+1}\,. \tag{4-10}$$

Similarly, from (4-10)

$x_{g_{B-2} j_{B+1} k} = 0 \quad \forall\, g_{B-2}, j_{B+1}, k \quad \text{where } g_{B-2} > 3 + j_{B+1}$.

Finally,

$$x_{e_1 j_{B+1} k} = 0 \quad \forall\, e_1, j_{B+1}, k \quad \text{where } e_1 > B + j_{B+1}. \tag{4-11}$$

(4-11) is the same as Theorem 2.1 which is for constrained sequencing problem with an off-line buffer of capacity *B*, meaning that all the possible re-sequencing options by using an off-line buffer with capacity *B* is exactly the same as options by using *B* off-line buffers of capacity 1. Therefore constrained sequencing problem with an off-line buffer of capacity *B* is equivalent to constrained sequencing problem with *B* off-line buffers of capacity 1. □

BIBLIOGRAPHY

AutoMod User's Manual v10.0. Bountiful, Brooks Automation, Inc. - AutoSimulations Division.

Ahuja, R. K., T. L. Magnanti, et al. (1993). Network Flows: Theory, Algorithms, and Applications, Prentice-Hall.

Askin, R. G. and C. R. Standridge (1993). Modeling and Analysis of Manufacturing Systems, John Wiley & Sons.

Assad, A. A. (1980). "Models for rail transportation." Transportation Research **14A**: 205-220.

Assad, A. A. (1981). "Analytical models in rail transportation: An annotated bibliography." INFOR **19**: 59-80.

Atassi, F. R. (1996). Implementation of block painting in Ford's in-line vehicle sequencing environment (MS Thesis). System Design and Management Program. Cambridge, MIT.

Bastani, A. S. (1990). "Closed-loop conveyor systems with breakdown and repair of unloading stations." IIE Transactions **22**(4): 351-360.

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

Bazaraa, M. S., J. J. Jarvis, et al. (1990). Linear Programming and Network Flows, John Wiley & Sons.

Bolat, A., M. Savsar, et al. (1994). "Algorithms for real-time scheduling of jobs on mixed model assembly lines." Computers and Operations Research **21**: 487-498.

Bozer, Y. A., M. A. Quiroz, et al. (1988). "An evaluation of alternative control strategies and design issues for automated order accumulation and sortation systems." Material Flow **4**: 265-282.

Bozer, Y. A. and G. P. Sharp (1985). "An empirical evaluation of general purpose automated order accumulation and sortation system used in batch picking." Material Flow **2**: 111-131.

Bras, B., S. Duncan, et al. (2001). Real-time integrated economic and environmental performance monitoring of a production facility. Society of Automotive Engineers**:** paper no.2000-ES-23.

Bruno, J. and P. Downey (1978). "Complexity of task sequencing with deadlines, setup times, and changeover costs." SIAM Journal of Computing **7**: 394-404.

Brzezinski, J. (2003). automobile manufacturing engineer**:** Personal communication.

Burns, L. D. and C. F. Daganzo (1987). "Assembly line job sequencing principles." International Journal of Production Research **25**: 71-99.

Campos, M., E. Bonabeau, et al. (2001). "Dynamic scheduling and division of labor in social insects." Adaptive Behavior **8**(2): 83-92.

Choe, K. I. (1990). Aisle-based order pick systems with batching, zoning, and sorting (Ph. D. Thesis). School of Industrial and Systems Engineering. Atlanta, GA, Georgia Institute of Technology.

Choe, K. I. and G. P. Sharp (1991). Small parts order picking: design and operations. Material Handling Research Center Technical Report MHRC-TR-89-07, Georgia Institute of Technology, Atlanta, GA.

Choe, K. I., G. P. Sharp, et al. (1992). Performance estimation of an automated sorting system. Proceedings of the 12th International Conference on Automation in Warehousing, 198-204.

Choe, K. I., G. P. Sharp, et al. (1993). Aisle-based order pick systems with batching, zoning, and sorting. Progress in Material Handling Research, Material Handling Institute, Charlotte, 245-276.

Cordeau, J.-F., P. Toth, et al. (1998). "A Survey of optimization models for train routing and scheduling." Transportation Science **32**(4): 380-404.

Daganzo, C. F. (1986). "Static blocking at railyards: sorting implications and track requirements." Transportation Science **20**: 189-199.

Daganzo, C. F. (1987). "Dynamic blocking for railyards. part I. homogeneous traffic." Transportation Research **21B**: 1-27.

Daganzo, C. F. (1987). "Dynamic blocking for railyards. part II. heterogeneous traffic." Transportation Research **21B**: 29-40.

Daganzo, C. F., R. G. Dowling, et al. (1983). "Railroad classification yard throughput: the case of multi-stage triangular sorting." Transportation Research **17A**: 95-106.

Gould, L. (1991). "New Sortation systems cuts postal costs $2 million per year." Modern Materials Handling **46**(10): 54-55.

Gould, L. (1991). "Sortation system doubles output capacity." Modern Materials Handling **46**(6): 50-51.

Graehl, D. (1992). Insights into carrier control: a simulation of a power and free conveyor through an automotive paint shop. Winter Simulation Conference.

Han, Y., C. Zhou, et al. (2002). Paint line color change reduction in automobile assembly through simulation. Winter Simulation Conference.

Horrey, R. J. (1983). Sortation systems: from push to high-speed fully automated applications. Proceedings of the 5th International Conference on Automation in Warehousing, Atlanta, GA.

Jayaraman, A., R. Narayanaswamy, et al. (1997). A sortation system model. Winter Simulation Conference.

Johnson, M. E. (1998). "The impact of sorting strategies on automated sortation system performance." IIE Transactions **30**: 67-77.

Johnson, M. E. and T. Lofgren (1994). "Model decomposition speeds distribution center design." Interfaces **24**(5): 95-106.

Kittithreerapronchai, O. and C. Anderson (2003). Do ants paint trucks better than chickens? Market versus response thresholds for distributed dynamic scheduling. Proceedings of the 2003 IEEE Congress on Evolutionary Computation, Canberra, Australia.

Laporte, G. (1992). "The traveling salesman problem: an overview of exact and approximate algorithms." European Journal of Operational Research **59**: 231-247.

Lee, C. Y. and G. Vairaktarakis (1997). "Workforce planning in mixed model assembly systems." Operations Research **45**: 553-567.

Meller, R. D. (1997). "Optimal order-to-lane assignments in an order accumulation/sortation system." IIE Transactions **29**: 293-301.

Monma, C. L. and C. N. Potts (1999). "On the complexity of scheduling with batch setup times." Operations Research **37**: 798-804.

Morley, R. (1996). Painting trucks at General Motors: the effectiveness of a complexity-based approach. Embracing complexity: exploring the application of complex adaptive systems to business. Cambridge, The Ernst & Young Center for Business Innovation**:** 53-58.

Morley, R. and G. Ekberg (1998). Cases in chaos: complexity-based approaches to manufacturing. Embracing complexity: a colloquium on the application of complex adaptive systems to business. Cambridge, The Ernst & Young Center for Business Innovation**:** 97-102.

Morley, R. and C. Schelberg (1993). An analysis of a plant-specific dynamic scheduler. Final Report, Intelligent, Dynamic Scheduling for Manufacturing Systems**:** 115-122.

Muth, E. J. (1979). "Conveyor theory: a survey." AIIE Transactions **11**(4): 270-277.

Myron, D. L. (1996). Paint blocking in Ford's in-line vehicle sequencing environment (MS Thesis). Leaders for Manufacturing Program. Cambridge, MIT.

Nemhauser, G. L. and L. A. Wolsey (1988). Integer and Combinatorial Optimization, John Wiley & Sons, Inc.

Park, Y., J. Matson, et al. (1998). Simulation and analysis of the Mercedes_Benz all activity vehicle (AAV) production facility. Winter Simulation Conference.

Potts, C. N. and L. N. v. Wassenhove (1992). "Integrating scheduling with batching and lot-sizing: a review of algorithms and complexity." Journal of Operational Research Society **43**: 395-406.

Schwind, G. F. (1992). "High speed distribution at Walden books." Material Handling Engineering **November**: 76-80.

Shapiro, J. (1979). Mathematical programming: structures and algorithms. New York, Wiley.

Siddiqee, M. W. (1972). Investigation of Sorting and Train Formation Schemes for a Railroad Hump Yard. Traffic Flow and Transportation. G. F. Newell. New York, Elsevier**:** 377-387.

Ulgen, O. and A. Gunal (1998). Simulation in the automobile Industry. Handbook of Simulation. J. Banks, John Wiley & Sons.

Unal, A. T. and A. S. Kiran (1992). "Batch sequencing." IIE Transactions **24**: 73-83.

Webster, S. and K. R. Baker (1995). "Scheduling groups of jobs on a single machine." Operations Research **43**: 692-703.

Witt, C. E. (1989). "Reebok's distribution on fast track." Material Handling Engineering **March**: 43-46.

Yagar, S., F. F. Saccomanno, et al. (1983). "An efficient sequencing model for humping in a rail yard." Transportation Research **17A**: 251-262.

Yano, C. A. and A. Bolat (1989). "Survey, development, and application of algorithms for sequencing paced assembly lines." Journal of Manufacturing Operations Management **2**: 172-198.

Yano, C. A. and R. Rachamadugu (1991). "Sequencing to minimize work overload in assembly lines with product options." Management Science **37**: 572-586.

VITA

Yong-Hee Han was born in Hamyeol, a small town in South Korea. He earned his B.S. degree from Industrial Engineering Department at Hanyang University in 1997, and M.S. from the School of Industrial and Systems Engineering at Georgia Institute of Technology in 1998. His research interests include scheduling, production planning, automotive manufacturing, and discrete event simulation. His email address is hyhkorea@hotmail.com.