Society of Korea Industrial and Systems Engineering Vol. 36, No.3 : 95-108, September 2013 ISSN : 2005-0461(print)
http://dx.doi.org/10.11627/jkise.2013.36.3.95 ISSN : 2287-7975(online)

Setup Minimization Problem in a Diverging Point of the Conveyor System

Hyoungtae Kim* · Yong-Hee Han**†

*Department of Global Service management, Woosong University
**Department of entrepreneurship and Small Business, Soongsil University

컨베이어 시스템 분기점에서의 셋업 최소화 문제

김형태* · 한용희**†

*우송대학교 글로벌서비스경영학부
**숭실대학교 벤처중소기업학과

The problem of constrained sequencing of a set of jobs on a conveyor system with the objective of minimizing setup cost is investigated in this paper. 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 conveyor junctions. The problem is motivated by the paint purge reduction problem at a major U.S. automotive manufacturer. We first model a diverging junction with a sequence-independent setup cost and predefined attributes as an assignment problem and this model is then extended for a more general situation by relaxing the initial assumptions in various ways.

Keywords : Sequencing, Setup Minimization, Line Balancing

1. Introduction

1.1 Motivation

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 automobiles, electronics, 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.

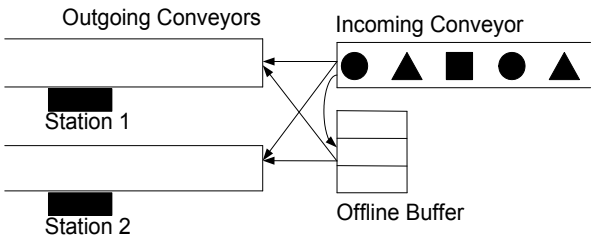
Conveyors are the most popular material transfer mechanism in high-volume production. Conveyors can transfer large amounts of material with simple motion control and also provide buffer space. However, simple conveyor segments are usually constrained to operate in a First-In-First-Out (FIFO) principle. 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, diverging conveyor junction points and/or off-line buffers (a conveyor itself can be considered as an on-line buffer) can also be used to change the sequence. Junction

Received 22 August 2013; Accepted 11 September 2013
† Corresponding Author : amade@ssu.ac.kr

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 the 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 Conveyor Example

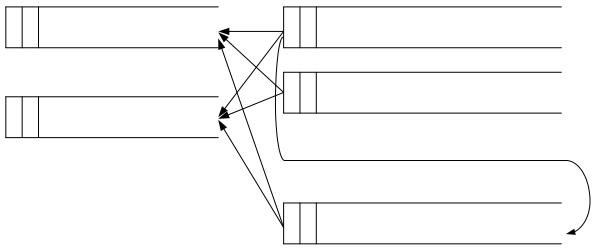
Our sequencing problem can be defined by the following two sets of elements. First, the design parameters include :

- number of downstream queues at the junction
- capacity of the upstream (and downstream) queues
- capacity of the off-line buffer at the junction - if one exists
- discipline supported by the upstream queues, the downstream queues, and the off-line buffer, respectively
- configuration of the junction.

Second, the known operational parameters include:

- attributes of the jobs in each queue,
- 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, multiple downstream conveyors and/or storage buffers is commonly observed in many real-world manufacturing environments, including the case study we conducted, as shown in <Figure 2>. In <Figure 2>, 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 2> Multiple Incoming and Outgoing Conveyors with an Off-line Buffer

While an optimal or a near-optimal solution may be found by observation for the problem of minimizing the number of color changes in <Figure 1>, finding an 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 therefore this paper is restricted to part of the conveyor system design and control with the objective of minimizing setup costs. More specifically, we restrict our attention to a constrained sequencing problem having one upstream (incoming) conveyor (queue) and no off-line buffer.

1.2 Problem Statement

Consider the problem of constrained sequencing of 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 inter-

face 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 paper is as follows. Relevant literature is reviewed in Section 2. Related problems are defined and analytical solution methodology is proposed in Section 3. Our research contributions are summarized and future research directions are discussed in Section 4.

2. Literature Review

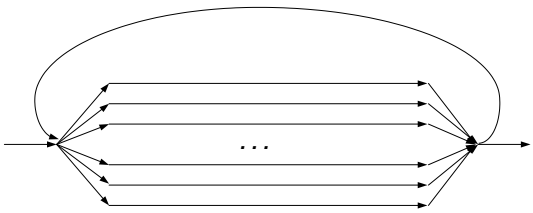
Morley and Schelberg [36], Morley [34], and Morley and Ekberg [35] discussed algorithms for assigning trucks to paint booths in a truck facility to minimize total make-span and the number of paint flushes. They applied market-based bidding algorithms to a GM plant and reported a 100% increase in average 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 et al. [12] compared Morley and Ekberg's market-based approach with an ant-inspired response threshold algorithm and used a genetic algorithm for getting parameter values for the above two algorithms. Kittithreerapronchai and Anderson [29] 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 [3] 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 [38] 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 et al. [16] 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.

A 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 're-humping'). 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 3>.

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

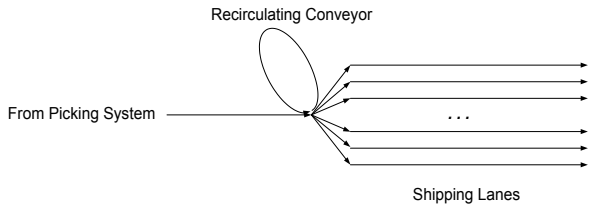


<Figure 3> The Rail Classification Problem as a Special Case of the Constrained Sequencing Problem

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 change-eovers should be counted on the rail for re-humping, 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 [43], who compared four sorting and train formulation schemes in a railroad classification yard. Yagar et al. [47] 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 et al. [21]. 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 [18], Daganzo [19], and Daganzo [20]) 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 [1], Assad [2], Choe et al. [16], and Cordeau et al. [17] for a general review of rail transportation problems. However, because the rail classification problem considers inherent differences 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 4>.



<Figure 4> 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 been assigned any lane.

Research on OAS is relatively scarce even though there exist many implementations of such systems in industry (see Johnson and Lofgren [28], Gould [22], Gould [23], Horrey [26], Schwind [41], and Witt [46]). In one of the first papers

to analyze OAS, Bozer and Sharp [9] 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 et al. [8] 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 [27] proves this result by an analytical model for OAS. Choe [13], Choe and Sharp [14], Choe et al. [15], and Choe et al. [16] 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 [32] 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 [37] for a general review and Bastani [4] 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 [5] and Yano and Bolat [48] 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 [31] and Yano and Rachamadugu [49]). 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 [11] and Bolat et al. [7] 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 [49] that considers 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 [33], Potts and Wassenhove [40], Unal and Kiran [44], and Webster and Baker [45]. In general, scheduling with sequence-dependent setups is NP-hard, with polynomial algorithms available only for a few special cases (see Bruno and Downey [10] and Laporte [30]).

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 problem discussed in this paper is 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.

3. Analytical Modeling

3.1 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. We assume as follows :

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. FIFO discipline is applied on all queues.
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 3.2 as the model is extended. We also need to introduce the following notations :

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

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

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

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

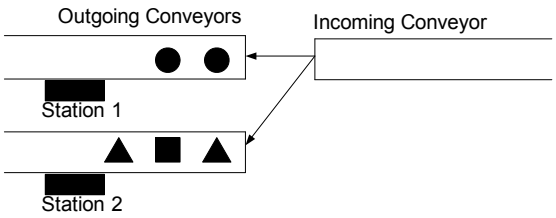
$$\bar{z}_{ij}=\begin{cases}1 & \text{if } i^{th} \text{ job in } U \text{ is the last item to be sent to a downstream queue}\\ 0 & \text{otherwise}\end{cases}$$

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

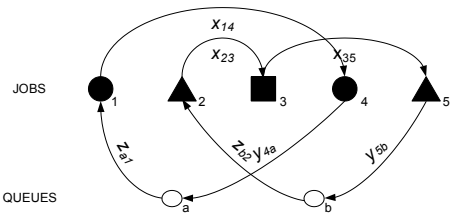
Property 1.1 : Dispatching Constraint
The FIFO discipline of U prohibits the i^{th} job in U from being dispatched before the $(i-1)^{th}$ job in U is dispatched.

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}. Furthermore, 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 5> shows a possible dispatching result from the example in <Figure 1>, where <Figure 6> is the associated network representation.



<Figure 5> A Possible Dispatching Result for the Example in <Figure 1>



<Figure 6> Network Representation Associated with <Figure 5>

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 6>, a downstream queue has jobs 1 and 4, and another downstream queue has jobs 2, 3, and 5, in sequence. Then as discussed in Han [25] and Han [24], the above net-

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

$$\text{Minimize} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \overline{C}_{ij} x_{ij} \tag{1}$$

subject to

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

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

$$\sum_{i=1}^{N} y_{iq} = 1 \qquad \forall\, q \tag{4}$$

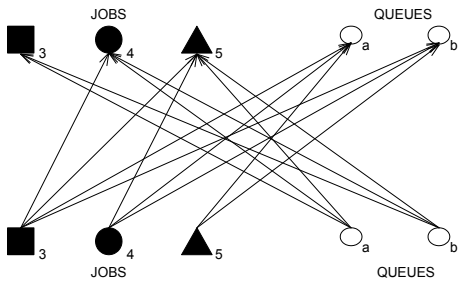
$$\sum_{j=1}^{N} z_{qj} = 1 \qquad \forall\, q \tag{5}$$

$$x_{ij},\ u_{iq},\ v_{qj},\ w_{q'q} \in \{0, 1\} \qquad \forall\, i, j, q, q \tag{6}$$

The objective function (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. (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. (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 (4) and (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 (4) and (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, $Q-1$, $Q-2$, $\cdots$ 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 3.2.

Constraint (6) is added because the meaning of the decision variables demands integrality. However, (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 et al. [6]). If our MIP formulation is represented as 'maximize c subject to $Ax = b$ where b is a binary variables vector', then 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 [39]). Therefore, A is totally unimodular because 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 A is totally unimodular and each element of b is integer-valued, then the optimal solution assigns all variables integer values (see Shapiro [42] for a proof). Therefore, (6) can be removed. With (6) removed, the formulation reduces to the well-solved assignment problem. The removal of (6) without loss of generality can also be proved by showing a one-to-one correspondence relationship between two equally sized sets composed of all jobs and queues as illustrated in <Figure 7>.



<Figure 7> 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.

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

3.2.1 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., 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 :

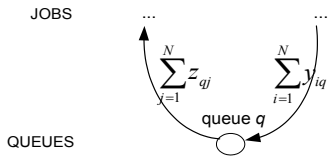
$$1 \le R \le K \tag{7}$$

Based on (7), 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

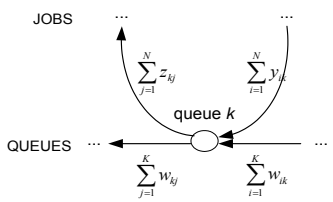
Recall from Section 3.1., constraints (4~5) restrict each queue to exactly one incoming unit flow and one outgoing unit flow, shown in <Figure 8>.



<Figure 8> Part of the Network Representation for the Model in Section 3.1

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 and some queues are allowed to 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 9>.



<Figure 9> Extension the Network Representation in <Figure 7>

For each queue node, constraints

$$\sum_{i=1}^{N} y_{ik} + \sum_{i=1, i\neq k}^{K} w_{ik} = 1 \text{ and } \sum_{j=1}^{N} z_{kj} + \sum_{j=1, j\neq k}^{K} w_{kj} = 1$$

are added. The former restricts a queue to have an entering arc either from a job, or from another queue. The latter sim-

ilarly 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.

3.2.2 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 :

$$AC_q = \frac{1}{2} C_q'' \left(\sum_{i=1}^{N} y_{iq} + \sum_{j=1}^{N} z_{iq}\right) + \frac{1}{2}\left[\frac{1}{2}\sum_{i=1, i\neq q}^{K} (C_q'' - C_i'') w_{iq} + \frac{1}{2}\sum_{j=1, j\neq q}^{K} (C_j'' - C_q'') w_{qj}\right]$$
$$= \frac{1}{2} C_q'' \left(\sum_{i=1}^{N} y_{iq} + \sum_{j=1}^{N} z_{qj}\right) + \frac{1}{4}\sum_{j=1, j\neq q}^{K} (C_j'' - C_q'') w_{qj} + \frac{1}{4}\sum_{j=1, j\neq q}^{K} (C_j'' - C_q'') w_{qj}$$

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.

It will be shown that all $f(w_{iq}, AC_i)$ and $f(w_{qj}, AC_j)$ terms in the above equation eventually cancel out if one sums AC_q over all queues, resulting in $\sum_q AC_q$ = total queue installation cost.

First, from

$\sum_{i=1}^{N} y_{ik} + \sum_{j=1, j\neq k}^{K} w_{ik} = 1$ and

$\sum_{j=1}^{N} z_{ik} + \sum_{j=1, j\neq k}^{K} w_{kj} = 1$, one can derive

$$0 \leq \sum_{i=1}^{N} y_{iq}, \sum_{i=1, i\neq q}^{K} w_{iq}, \sum_{j=1}^{N} z_{qj}, \sum_{j=1, j\neq q}^{K} w_{qj} \leq 1$$

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 $\sum_{i=1}^{N} y_{iq}, \sum_{j=1}^{N} z_{qj}, \sum_{i=1, i\neq q}^{K} w_{iq}, \sum_{j=1, j\neq q}^{K} w_{qj}$:

Case 1 **:**

$$\sum_{i=1}^{N} y_{iq} = \sum_{j=1}^{N} z_{qj} = 1 \quad \text{and} \quad \sum_{i=1, i\neq q}^{K} w_{iq} = \sum_{j=1, j\neq q}^{K} w_{qj} = 0 \quad \forall q$$

For Case 1,

$$AC_q = \frac{1}{2} C_q'' \left(\sum_{i=1}^{N} y_{iq} + \sum_{j=1}^{N} z_{qj}\right) = C_q''.$$

Because

$$\sum_{i=1, i\neq q}^{K} w_{iq} = \sum_{j=1, j\neq q}^{K} w_{qj} = 0,$$

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 6> 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 9>, $AC_q = C_q''$ and $\sum_q AC_q = C_a'' + C_b''$ which represents the total queue installation cost correctly.

Case 2 **:**

$$\sum_{i=1}^{N} y_{iq} = \sum_{j=1, j\neq q}^{K} w_{qj} = 1 \quad \text{and} \quad \sum_{j=1}^{N} z_{qj} = \sum_{i=1, i\neq q}^{K} w_{iq} = 0 \quad \forall q$$

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

$$\sum_{j=1,j\neq q}^{K} w_{qj}=1 \quad \text{and} \quad \sum_{i=1,i\neq q}^{K} w_{iq}=0,$$

there is only one queue, denoted r, connected with queue q and queue r belongs to either case 3 or case 4.

Therefore,

$$AC_q=\frac{1}{4}C_q''+\frac{1}{4}C_r''w_{qr} \tag{8}$$

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

$$\therefore AC_q+AC_r=\begin{cases}C_r'' & \text{if } r \text{ belongs to Case 3}\\ \frac{1}{4}\sum_{j=1,j\neq r}^{K}C_j''w_{qj} & \text{if } r \text{ belongs to Case 4}\end{cases}$$

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 (8). 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 (8~10), 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 10>, where $AC_q+AC_r=C_r''$; queue r is in case 3.

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

$$\sum_{i=1,i\neq q}^{K} w_{iq}=1 \quad \text{and} \quad \sum_{j=1,j\neq q}^{K} w_{qj}=0,,$$

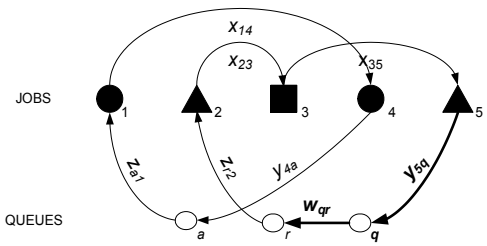
there is only one queue, p, connected with queue q and queue p belongs to either case 2 or case 4.

Therefore,

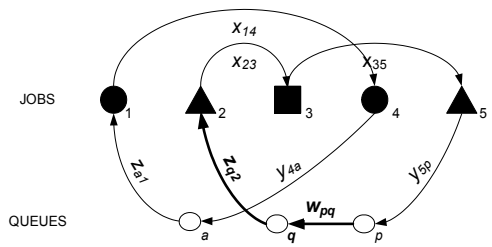
$$AC_p=\begin{cases}\frac{1}{4}C_q''-\frac{1}{4}C_p''w_{pq} & \text{if } p \text{ belongs to Case 2}\\ \frac{1}{4}C_q''-\frac{1}{4}\sum_{i=1,i\neq p}^{K}C_i''w_{ip} & \text{if } p \text{ belongs to Case 4}\end{cases} \tag{9}$$

$$\therefore AC_q+AC_p=\begin{cases}C_q'' & \text{if } p \text{ belongs to Case 2}\\ C_q''-\frac{1}{4}\sum_{i=1,i\neq r}^{K}C_i''w_{ip} & \text{if } r \text{ belongs to Case 4}\end{cases}$$

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 (9). 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 (8~10), 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 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 11> shows an example of case 3 where $AC_q+AC_p=C_q''$; queue p is in case 2.



<Figure 10> Example of Case 2



<Figure 11> Example of Case 3

Case 3 :

$$\sum_{j=1}^{N} z_{qj}=\sum_{i=1,i\neq q}^{K} w_{iq}=1 \quad \text{and} \quad \sum_{i=1}^{N} y_{iq}=\sum_{j=1,j\neq q}^{K} w_{qj}=0 \quad \forall q$$

Case 4 :

$$\sum_{j=1,j\neq q}^{K} w_{qj}=\sum_{i=1,i\neq q}^{K} w_{iq}=1 \quad \text{and} \quad \sum_{i=1}^{N} y_{iq}=\sum_{j=1}^{N} z_{qj}=0 \quad \forall q$$

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,

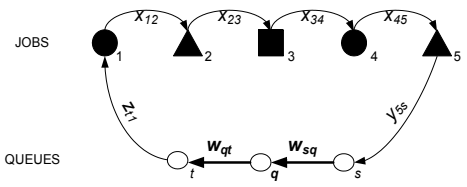
$$AC_q = \frac{1}{4}C_t'' + \frac{1}{4}C_s'' \qquad (10)$$

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

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

$$\therefore AC_q + AC_s + AC_t = \begin{cases} C_t'' & \text{if } s \text{ belongs to Case 2 and } t \text{ belongs to Case 3} \\ \frac{1}{4}C_t'' + \frac{1}{4}\sum_{j=1, j\neq r}^{K} C_j'' w_{qj} & \text{if } s \text{ belongs to Case 2 and } t \text{ belongs to Case 4} \\ C_t'' - \frac{1}{4}C_s'' - \frac{1}{4}\sum_{i=1, i\neq r}^{K} C_i'' w_{iq} & \text{if } s \text{ belongs to Case 2 and } t \text{ belongs to Case 4} \\ \frac{1}{4}\sum_{j=1, j\neq t}^{K} C_j'' w_{jt} - \frac{1}{4}\sum_{i=1, i\neq r}^{K} C_i'' w_{is} & \text{if } s \text{ belongs to} \end{cases}$$

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 12> where $AC_q + AC_s + AC_t = C_t''$; queue s is in case 2 and queue t is in case 3.



<Figure 12> 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 $\sum_{q=1}^{J} AC_q$= total queue installation cost.
- $\sum_{j=1, j\neq q}^{K} w_{qi} = 1$ 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 $\sum_{q=1}^{K} AC_q$

$$= \sum_{q=1}^{K}\left[\frac{1}{2}C_q''\left(\sum_{i=1}^{N} y_{iq} + \sum_{j=1}^{N} z_{qj}\right)\right] +$$

$$\frac{1}{2}\sum_{q=1}^{K}\left[\frac{1}{2}\sum_{i=1, i\neq q}^{K}(C_q'' - C_i'')w_{iq} + \frac{1}{2}\sum_{j=1, j\neq q}^{K}(C_j'' - C_q'')w_{qj}\right]$$

$$= \frac{1}{2}\sum_{q=1}^{K} C_q''\left(\sum_{i=1}^{N} y_{iq} + \sum_{j=1}^{N} z_{qj}\right) + \frac{1}{2}\sum_{i=1}^{K}\sum_{j=1, j\neq i}^{K}(C_j'' - C_i'')w_{ij}$$

3.2.3 Model Formulation

Summarizing the discussion in section Ⅲ, a MIP model that explicitly considers the number of available queues and the attribute-dependent setup cost can be formulated as follows :

$$Minimize \quad \sum_{i=1}^{N-1}\sum_{j=i+1}^{N} C_{ij}'' x_{ij} + \frac{1}{2}\sum_{q=1}^{K} C_q''\left(\sum_{i=1}^{N} y_{iq} + \sum_{j=1}^{N} z_{qj}\right) + \frac{1}{2}\sum_{i=1}^{K}\sum_{j=1, j\neq i}^{K}(C_j'' - C_i'')w_{ij}$$

subject to

$$\sum_{j=i+1}^{N} x_{ij} + \sum_{q=1}^{K} y_{iq} = 1 \qquad \forall\, i = 1, \cdots, N \qquad (11)$$

$$\sum_{i=1}^{j-1} x_{ij} + \sum_{q=1}^{K} z_{iq} = 1 \qquad \forall\, i = 1, \cdots, N \qquad (12)$$

$$\sum_{i=1}^{N} y_{iq} + \sum_{i=1, i\neq q}^{K} w_{iq} = 1 \qquad \forall\, q = 1, \cdots, K \qquad (13)$$

$$\sum_{j=1}^{N} z_{qj} + \sum_{j=1, j\neq q}^{K} w_{qj} = 1 \qquad \forall\, q = 1, \cdots, K \qquad (14)$$

$$x_{ij}, y_{iq}, z_{qj}, w_{iq}, w_{qj} \in \{0,1\} \qquad \forall\, i, j, q \qquad (15)$$

The first term in the objective function is the sum of the setup cost of all jobs in all downstream queues, as in the simple model in Section 3.1, 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. We also note that the objective function can be simplified as follows :

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

Explanations for constraints (11~12) are given in Section 3.1. For (13~15), newly added variables w are needed to identify unused queues. Again, (15) can be safely removed using the same reasoning as in Section 3.1 - total unimodularity. As a result, the problem again becomes the well-solved assignment problem. 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.

4. Contributions and Future Directions

In this paper, analytical models have been developed for a constrained sequencing problem with a diverging junction. We also modeled cases where the 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.

Even though this paper did extensive analysis of diverging junction cases, no analysis has been made of off-line buffer cases, converging junction cases, and off-line buffer cases. If making an optimization model for these 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.

Future research also needs to seek to incorporate other constraints such as those on cycle time limitations into the models. In the interview we conducted with the paint shop plant managers, 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 is 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 model discussed in this research :

- 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.

- 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).

References

[1] Assad, A.A., Models for rail transportation. *Transportation Research*, 1980, Vol. 14, No. 3, p 205-220.

[2] Assad, A.A., Analytical models in rail transportation : An annotated bibliography, INFOR, 1981, Vol. 19, p 59-80.

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

[4] Bastani, A.S. Closed-loop conveyor systems with breakdown and repair of unloading stations. *IIE Transactions*, 1990, Vol. 22, No. 4, p 351-360.

[5] Baybars, I., A survey of exact algorithms for the simple assembly line balancing problem. *Management Science*, 1986, Vol. 32, p 909-932.

[6] Bazaraa, M.S., Jarvis, J.J., and Sherali, H.D., Linear Programming and Network Flows, John Wiley and Sons, 1990.

[7] Bolat, A., Savsar, M., and Al-Fawzan, M.A., Algorithms for real-time scheduling of jobs on mixed model assembly lines. *Computers and Operations Research*, 1994, Vol. 21, p 487-498.

[8] Bozer, Y.A., Quiroz, M.A., and Sharp, G.P., An evaluation of alternative control strategies and design issues for automated order accumulation and sortation systems. *Material Flow*, 1988, Vol. 4, p 265-282.

[9] Bozer, Y.A. and Sharp, G.P., An empirical evaluation of general purpose automated order accumulation and sortation system used in batch picking. *Material Flow* 1985, Vol. 2, p 111-131.

[10] Bruno, J. and Downey, P., Complexity of task sequencing with deadlines, setup times, and changeover costs. *SIAM Journal of Computing*, 1978, Vol. 7, p 394-404.

[11] Burns, L.D. and Daganzo, C.F., Assembly line job sequencing principles. *International Journal of Production Research*, 1987, Vol. 25, p 71-99.

[12] Campos, M., Bonabeau, E., and Theraulaz, G., Dynamic scheduling and division of labor in social insects. *Adaptive Behavior*, 2001, Vol. 8, No. 2, p 83-92.

[13] Choe, K.I., 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, 1990.

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

[15] Choe, K.I. and Sharp, G.P., Performance estimation of an automated sorting system. *Proceedings of the 12th International Conference on Automation in Warehousing*, 1992, p 198-204.

[16] Choe, K.I. and Sharp, G.P., Aisle-based order pick systems with batching, zoning, and sorting. *Progress in Material Handling Research*, Material Handling Institute, Charlotte, 1993, p 245-276.

[17] Cordeau, J.F., Toth, P., and Vigo, D., A Survey of optimization models for train routing and scheduling. *Transportation Science*, 1998, Vol. 32, No. 4, p 380-404.

[18] Daganzo, C.F., Static blocking at railyards : sorting implications and track requirements. *Transportation Science*, 1986, Vol. 20, p 189-199.

[19] Daganzo, C.F., Dynamic blocking for railyards, part I. homogeneous traffic. *Transportation Research*, 1987a, 21B, p 1-27.

[20] Daganzo, C.F., Dynamic blocking for railyards, part II. heterogeneous traffic. *Transportation Research*, 1987b, 21B, p 29-40.

[21] Daganzo, C.F., Dowling, R.G., and Hall, R.W., Railroad classification yard throughput : the case of multi -stage triangular sorting. *Transportation Research*, 1983, Vol. 17, No. 2, p 95-106.

[22] Gould, L., New Sortation systems cuts postal costs $2 million per year. *Modern Materials Handling*, 1991, Vol. 46, No. 10, p 54-55.

[23] Gould, L., Sortation system doubles output capacity. *Modern Materials Handling*, 1991, Vol. 46, No. 6, p 50-51.

[24] Han, Y., Dynamic sequencing of jobs on conveyor systems for minimizing changeovers (Ph. D. Thesis), in School of Industrial and Systems Engineering Atlanta : Georgia Institute of Technology, 2004.

[25] Han, Y., Optimal Conveyor Selection Problem on a Diverging Conveyor Junction Point. *Journal of the*

Society of Korea Industrial and Systems Engineering, 2009, Vol. 32, No. 3, p 118-126.

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

[27] Johnson, M.E., The impact of sorting strategies on automated sortation system performance. *IIE Transactions*, 1998, Vol. 30, p 67-77.

[28] Johnson, M.E. and Lofgren, T., Model decomposition speeds distribution center design. *Interfaces*, 1994, Vol. 24, No. 5, p 95-106.

[29] Kittithreerapronchai, O. and Anderson, C., 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, 2003.

[30] Laporte, G., The traveling salesman problem : an overview of exact and approximate algorithms. *European Journal of Operational Research*, 1992, Vol. 59, p 231-247.

[31] Lee, C.Y. and Vairaktarakis, G., Workforce planning in mixed model assembly systems. *Operations Research*, 1997, Vol. 45, p 553-567.

[32] Meller, R.D., Optimal order-to-lane assignments in an order accumulation/sortation system. *IIE Transactions*, 1997, Vol. 29, p 293-301.

[33] Monma, C.L. and Potts, C.N., On the complexity of scheduling with batch setup times. *Operations Research*, 1999, Vol. 37, p 798-804.

[34] Morley, R., Painting trucks at General Motors : the effectiveness of a complexity-based approach. *Cambridge*, The Ernst and Young Center for Business Innovation, 1996, p 53-58.

[35] Morley, R. and Ekberg, G., Cases in chaos : complexity-based approaches to manufacturing. *The Ernst and Young Center for Business Innovation*, 1998, p 97-102.

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

[37] Muth, E.J., Conveyor theory : a survey. *AIIE Transactions*, 1979, Vol. 11, No. 4, p 270-277.

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

[39] Nemhauser, G.L. and Wolsey, L.A., Integer and Combinatorial Optimization, John Wiley and Sons, Inc., 1988.

[40] Potts, C.N. and Wassenhove, L.N.V., Integrating scheduling with batching and lot-sizing : a review of algorithms and complexity. *Journal of Operational Research Society*, 1992, Vol. 43, p 395-406.

[41] Schwind, G.F., High speed distribution at Walden books. *Material Handling Engineering November*, 1992, p 76-80.

[42] Shapiro, J., Mathematical programming : structures and algorithms. New York, Wiley, 1979.

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

[44] Unal, A.T. and Kiran, A.S., Batch sequencing. *IIE Transactions*, 1992, Vol. 24, p 73-83.

[45] Webster, S. and Baker, K.R., Scheduling groups of jobs on a single machine. *Operations Research*, 1995, Vol. 43, p 692-703.

[46] Witt, C.E., Reebok's distribution on fast track. *Material Handling Engineering March*, 1989, p 43-46.

[47] Yagar, S., Saccomanno, F.F., and Shi, Q., An efficient sequencing model for humping in a rail yard. *Transportation Research*, 1983, Vol. 17A, p 251-262.

[48] Yano, C.A. and Bolat, A., Survey, development, and application of algorithms for sequencing paced assembly lines. *Journal of Manufacturing Operations Management*, 1989, Vol. 2, p 172-198.

[49] Yano, C.A. and Rachamadugu, R., Sequencing to Minimize Work Overload in Assembly Lines with Product Options. *Management Science*, 1991, Vol. 37, No. 5, p 572-586.