2002

Analysis of delayed product differentiation under pull type policies

Heedong Kim
Iowa State University

INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

Analysis of delayed product differentiation under pull type policies

by

Heedong Kim

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Industrial Engineering

Program of Study Committee:
Sarah M. Ryan, Major Professor
Thomas Barta
Jo Min
Sigurdur Olafsson
Yoshinori Suzuki

Iowa State University

Ames, Iowa

2002

ProQuest Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

Graduate College
Iowa State University

This is to certify that the doctoral dissertation of

Heedong Kim

has met the dissertation requirements of Iowa State University

Signature was redacted for privacy.

Major Professor

Signature was redacted for privacy.

For the Major Program

TABLE OF CONTENTS

ABSTRACT

Delayed product differentiation (DPD) increases manufacturers' competitiveness in the market by enabling them to more quickly respond to changes in customers' demands. DPD has also been shown to require less Work-in-Process (WIP) than a non-DPD setup in some cases. Although there are many papers on this topic, previous research was mainly focused on the level of semi-finished and/or finished good inventory under a base-stock policy. The control of WIP inventory was not considered. DPD may also improve response times under pull inventory control schemes, in which the amount of WIP is controlled directly. Possible alternatives are kanban, CONstant-WIP (CONWIP) and multiple CONWIP loops. These systems can be modeled as closed queueing networks in which a fixed number of kanbans circulate as customers among each set of one or more processing stages.

The interfaces between kanban loops in the queueing networks greatly complicate the analysis of these systems. There are currently no exact methods to determine how many kanbans to have in each loop achieve specified throughputs. In this study, we first developed models to analyze the performance of simple kanban and CONWIP controlled systems and set the number of kanbans to achieve a specified performance level. The models help us better understand the behavior of pull systems. The performance evaluation method uses nonlinear programming (NLP) models to bound the throughput for fixed number of kanbans or minimize the number of kanbans necessary to achieve a specified throughput. These simple models can also serve as the foundation for developing analytical models of more complex systems. The model shows how random supplies and demands prevent equilibrium from occurring in a single-stage kanbans system.

We studied a model for a system of two products with unlimited supply and demand using three CONWIP loops to represent the common processes and the differentiated processes for each product. The same system after DPD has more common processes and fewer differentiated processes. The NLP model can determine numbers of kanbans for each loop to achieve specified throughput targets. Because the throughput bounds are not as tight as desired, we developed a heuristic algorithm that starts from the NLP solution and adjusts the kanbans using simulation to evaluate the performance. A comparison of the result of the heuristic algorithm for the systems with and without DPD indicates that DPD reduces the amount of WIP necessary to achieve a specified throughput. Furthermore, we show how models of systems with similar structure can be generalized.

CHAPTER 1 INTRODUCTION

The goal of delayed product differentiation (DPD) is to move the point of differentiation between products to the downstream operations as much as possible. Lee and Tang (1997) presented three types of product/process redesign approaches for DPD, namely standardization, modular design, and process restructuring. Standardization can be accomplished by designing a part used after the point of differentiation so that it can be commonly used by all products. In modular design, a part is divided into two modules such that the first module is completed in front of the point of differentiation. One common example of modular design is the use of a faceplate for electronics and appliances. The case of these products used to be designed in one piece. By dividing the case into a common module and a customizable faceplate, the point of differentiation can be delayed. Process restructuring involves changing the order of processes when it is feasible and does not require any change in the product design.

Figure 1 illustrates an example of how delayed product differentiation might be achieved. There are two different products with some common features. The figure on the left shows two common processes before implementation of DPD. The figure on the right

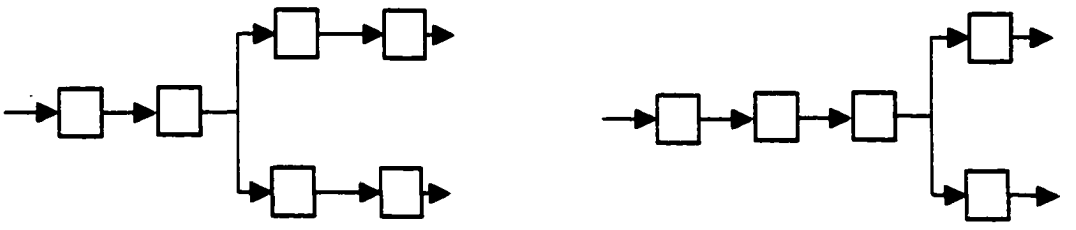

Figure 1. An example of delayed product differentiation

shows that common processes are increased from two to three after DPD. For example, consider a computer mouse manufacturer whose products are sold in U.S. and Korea. Since the computer mouse is an electrical device, it is required to display information showing that it satisfies the regulations of the country where it is sold. Until a few years ago, only information in the language of the market was specified. But, now, it is not uncommon to have information showing in each language of all of its market. Under the old design, the mouse would be differentiated when the country-specific casing was assembled. After the assembly, the mouse would be packed in a box with either Korean or English instructions. The new design postpones the point of differentiation until after the assembly of the casing since the same mouse can be sold in both Korea and U.S.

Delayed product differentiation (DPD) is currently being used in many automobile, electronic and appliance industries. Customers want to buy customized products; they want to be able to choose some functions and components in their products, but manufacturers want to benefit from the economics of large scale production. A common example of DPD is an innovation at Benetton, an apparel manufacturer. They were able to delay the point of differentiation by dyeing yarn before knitting it into garments (see Signorelli and Heskett 1989). Previously, the knitting was done first, then the dyeing. By reversing these operations, they were able to adapt to customer demands for different sizes and styles more quickly. One of the advanced automobile manufacturers, Toyota, calls it 'postponement' instead of DPD (see Federgruen 1993). Many vehicle options are installed by the dealers rather than in the factory. Also, the most common example of this customization is that personal computer buyers can configure devices in their computers. The manufacturers need to produce greater variety of computers with different options. This leads the manufacturer to a difficult

decision between satisfying the customers and maintaining low inventory. In the traditional approach to satisfy customers, the manufacturer must produce many items in advance of demand (customize-to-stock). Otherwise, they may not be able to deliver the desired products within the customers' delivery lead time expectation. When they produce in advance, they will have to keep more finished products. Also, it is possible that the actual demand is less than forecasted demand and the overproduced finished goods may be sold at lower prices to reduce the manufacturer's inventory. To maintain low inventory, the manufacturer may choose to produce after they receive orders from customers (customize-to-order). Although this guarantees low inventory, the delivery lead time will be much longer than in the customize-to-stock system. DPD can be used to reduce the amount of the time required for customization.

Production control systems are usually characterized as either 'push' or 'pull'. One characteristic of pull production systems is that there is a fixed population of Work-in-Process (WIP) in the system so that they can be modeled as closed queueing networks. The queueing network models include fork/join synchronization stations where supplies are matched with production signals and demands are matched with finished products. Interesting performance characteristics of pull production systems are throughput and cycle time (since the amount of WIP in the system is fixed). In contrast, in a push system the throughput and/or cycle time are fixed and the WIP is observed as a performance measure. A push system is controlled by a production plan. The plan specifies when to start production and how many to produce. On the other hand, a pull system production usually is driven by demand from downstream machines or customers while limiting the number of parts. The limit on the number of parts is enforced by fixing the number of kanbans in each kanban loop.

There are different types of pull control; kanban, hybrid kanban-CONWIP, CONWIP (CONstant-WIP; Spearman, Woodruff, and Hopp 1990) and POLCA (Paired-cell Overlapping Loops of Cards with Authorization; Suri 1998). The CONWIP approach differs from the kanban that CONWIP specifies a fixed number of kanbans for the whole production line while a kanban system has a fixed number of kanbans in each production stage. The hybrid kanban-CONWIP is somewhere between kanban and CONWIP in that it uses CONWIP for the whole system and kanbans for individual stages at the same time. POLCA uses multiple loops of CONWIP and connects each CONWIP loop by fork/join synchronization stations for pulling parts from the upstream CONWIP loop. In fact, only the kanban system, often called 'pure kanban' system, is a pull system. All others are mixture of push and pull because they limit the number of kanbans in the system while allowing finished parts/kanbans to be pushed downstream. The classification of the systems having multiple CONWIP loops is not yet clearly defined. In our study, we just call it a multiple CONWIP loop system.

1.1 Problem Statement

Some of the benefits of delayed product differentiation are low inventory, and shorter customer response time (see Silver and Peterson 1985, and Gupta and Krishnan 1998). By delaying the point of differentiation, manufacturers that use a base-stock system can reduce the amount of safety stock because they can respond to changed demand more quickly. Also, the WIP level can be reduced by delaying product differentiation (see Swaminathan and Tayur 1999). Furthermore, pull type control policies usually requires less WIP inventory than

push type control policies (Hopp and Spearman 2000). Less inventory again leads to a short cycle time of production. Studies that have been done in the past related to inventory of DPD systems assumed base-stock and focused on the finished product inventory level and/or semi-fished product inventory level before the point of differentiation. Although the inventory levels in a systems are different for the different inventory control policies, all studies either assumed a push type control policy or were focused only on the finished stock level. Furthermore, the decision on how to implement DPD may be different for a pull type control policy. It is clearly beneficial to study whether the same performance of a system can be obtained with less WIP when DPD is implemented under a pull type control policy.

1.2 Research Objectives

In order to answer the above questions, it is required to develop a method to evaluate performance of systems under pull type inventory policies. Performance of a system can be measured by the throughputs of different products. We need a method to determine how much inventory (WIP) is required and where the inventory should be allocated to achieve specified performance levels. Also, a model for how DPD could be implemented in a pull type system needs to be developed. Using the methods developed, this research compares amount of WIP required to achieve specified throughput with and without DPD.

As a primary study for developing a performance evaluation method, this research developed nonlinear programming (NLP) models for simpler systems, namely, one machine under kanban control, two serial machine CONWIP, and three serial machine kanban systems. These models are helpful to understand the pull type systems better. There are two

items of interest in this primary study. First, it is desired to modularize the mathematical model of kanban systems. Without a modularized kanban model, the number of constraints increases as the number of buffers and fork/join stations grows. Especially, the number of indexes and subscripts on variables also increases as the number of fork/join stations increases. This causes the problem size to quickly increase. Modularization can facilitate solving larger problems. Second, the model initially assumed that the service time at station, times between arrivals of supplies, and times between arrivals of demands are exponential. Since this assumption may not be appropriate in some situations, a model for Erlang service time distribution is developed.

We found very interesting results from the primary study. It is not possible to have steady state in a single-stage kanban system if the arrival of supply and demand are independent of each other and random. The result suggests that it may not be appropriate to use the pull control policy in some applications. Furthermore, in the analysis of two serial machine CONWIP systems, it was expected that the Erlang service time model would provide better performance bounds than the exponential service time model. But the study gives same values of upper and lower bound for the exponential service time model (that is, exact values for the system performance) while the gap between upper and lower bounds for the Erlang service time model is fairly wide.

In general, the throughput bounds found by the NLP models have a wide interval. For this reason, the NLP models can not be relied upon to find a kanban allocation guaranteed to satisfy a specified throughput requirement. A heuristic algorithm is developed to find the actual kanban allocation required to meet the customers demand. The heuristic algorithm uses NLP solutions as a starting point to quickly find a feasible kanban allocation and

evaluates the system performance by simulation. Using the heuristic algorithm, the performance of systems before and after delaying product differentiation are compared. The result shows that delaying product differentiation can reduce the inventory in the systems and shorten the cycle time is shorter.

Therefore, the models show that delaying product differentiation reduces inventory and response time for customers demand. Also, it finds a new kanban allocation when the demand of customer is changed in existing system. Furthermore, this research provides strategies for modeling and analyzing the delaying product differentiation in different environment.

This dissertation is organized as follows. In chapter 2, we discuss related research. In chapter 3, we present a performance evaluation model for queueing networks of kanban systems. In chapter 4, we present an evaluation model for queueing networks of CONWIP systems. In chapter 5, we present the evaluation model for queueing networks with and without DPD. Also, the simulation-based heuristic algorithm for kanban allocation is presented. In chapter 6, we compare systems with and without DPD using NLP models and the heuristic algorithm. We conclude with chapter 7.

CHAPTER 2 LITERATURE REVIEW

2.1 Delayed Product Differentiation

The concept of DPD is not new, in fact, it was first presented by Anderson (1950) in the marketing literature. In spite of the early appearance of the concept, researchers started to actively study the area of delayed product differentiation in the 1990s.

Swaminathan and Tayur (1998) discussed vanilla box configuration and inventory levels of vanilla boxes for delayed product differentiation. A vanilla box is a semi-finished product that can be used for producing several different products. The model has two submodels: one finds best the vanilla box configuration and the other one finds the best assembly sequences. The authors developed an algorithm for solving the two submodels simultaneously. Swaminathan and Tayur (1999) later presented a model for configuration of vanilla boxes.

There are other studies showing the benefits of delayed product differentiation. Lee and Tang (1998) showed that the performance of a system can be improved by reversing two consecutive operations. He, Kusiak and Tseng (1998) studied DPD strategies in the aspect of design of parts and its manufacturing processes. He and Babayan (2002) presented optimal and heuristic methods for scheduling production sequences for delayed product differentiation in agile manufacturing to minimize makespan. Ma, Wang and Liu (2001) studied the effect of lead time and procurement time on the decision of postponement under a base-stock policy. Garg and Tang (1997) developed model to examine the impact of DPD when there is more than one point of differentiation. Schraner and Hausman (1997)

developed method for sequencing production operations to minimize cost. Aviv and Federgruen (2001) characterized the benefits of DPD where the demand is not known or not accurate or where consecutive demands are correlated. Although many studies have been presented and published, most studies simply assumed push type inventory controls or did not consider the inventory control policy.

The importance of studying DPD under a pull policy is backed by Van Hoek's recent study (2001) in which he reviewed the literature on DPD and suggested directions of research. Van Hoek emphasized the importance of studying the cross-company dimension of the supply chain. He also pointed out the desirability of studying the integration of related supply chain concepts such as just-in-time (JIT) manufacturing and supply, vendor-managed inventory, efficient consumer response (ECR) and the associated quick-response distribution techniques.

2.2 Kanban and CONWIP

Our study considers a pull type inventory control policy. Although Krishnamurthy, Suri and Vernon (2000) showed that a push system may perform better in flexible manufacturing systems with low throughput requirements, all recent studies show that pull systems perform better than push systems in general (see Hopp and Spearman 2000, and Spearman and Zazanis 1992). The pull type inventory policy could be very useful in conjunction with DPD since production in a pull type system is driven by customers' demands. The inventory control policy CONWIP, used for controlling the systems with and without DPD in this study, is a relatively new concept (see Spearman, Woodruff, and Hopp

1990, Spearman, and Zazanis, 1992, and Muckstadt and Tayur 1995). Tayur (1992) discussed the properties of allocation of kanbans and partition under pull systems. Bonvik, Couch and Gershwin (1997) compared different types of inventory policies, namely base-stock, kanban, CONWIP and hybrid kanban-CONWIP, using simulations to show better performance of CONWIP and hybrid-CONWIP policies. Also, Muckstadt and Tayur (1995) compared kanban and CONWIP systems under different objectives and showed that one strategy can perform better than the other for different objectives and operating conditions. Herer and Masin (1997) developed a method for setting the order of the backlog list of CONWIP based production lines. There also have been studies to allocate kanbans in CONWIP systems under different types of production environments (Hopp and Roof 1998; Ryan, Baynat, and Choobineh 2000; Ryan and Choobineh 2002).

2.3 Performance Evaluation

Pull systems can be modeled as closed queueing networks. One of the methods used to evaluate performance of closed queueing network is Mean Value Analysis (MVA). There is much literature on MVA for different types of closed queueing network (see, for example, Buzacott and Shanthikumar 1993, Hopp and Spearman 2000, and Suri and Hilderbrant 1984). MVA is not suitable for our research because it can not handle fork/join stations. Krishnamurthy, Suri and Vernon (2001) developed a two-moment approximation model for analysis of systems with fork/join stations. They decompose the queueing network into manufacturing stations and fork/join stations. Then, they apply the two-moment approximation model. Although the concept of the model is quite easy to follow, the model is

still being improved for better accuracy. Also, Baynat and Dallery (1996) developed product-form approximation methods for the single chain closed queueing network that can handle fork/join stations (see also Baynat et al. 2001). The routings of parts at a fork/join station are probabilistically decided. Their method is difficult to use in this research because we are considering multichain queueing networks to model multiple product loops.

Kumar and Kumar (1994) presented a technique for finding upper and lower bounds of Markovian queueing networks. Ryan and Choobineh (2002) adapted the Kumar and Kumar approach to a CONWIP controlled job shop and developed a procedure to determine the total amount of inventory in the system and its allocation among the product types. They extended the method to obtain tighter bounds by enhancing the Kumar and Kumar model. Our analytical model for queueing network performance evaluation also is adapted from the methodology used in Kumar and Kumar study.

CHAPTER 3 MODELING OF PULL SYSTEMS

In this chapter, we will present nonlinear programming models for performance evaluation of kanban controlled queueing networks in steady state. We first present a simple model of a kanban controlled queueing network and explain the control mechanism. Then, we show equality and inequality constraints used in the models.

The following are the assumptions in the queueing models:

- Exponential distributions for service times, times between arrivals of raw material, and times between arrivals of demand unless it is assumed that there is infinite supply of raw material or infinite demand of final product.
- Each server has a distinct set of kanbans associated with it.

Figure 2 shows the queueing network for a single machine single product kanban system. In the figure, B_1 is the buffer for the supply of raw material (or finished parts from the upstream station) with raw material arrival rate of λ and B_5 is the buffer for the demand of end product (or demand for a finished part from the downstream station) with rate of ν. The network shows two fork/join synchronization stations. Each fork/join synchronization

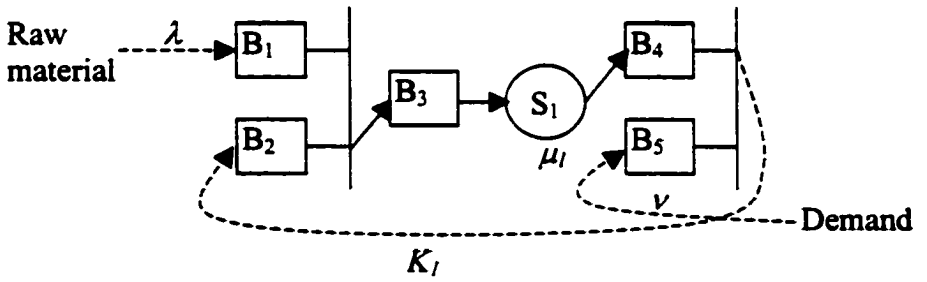

Figure 2. Closed queueing network model of single-stage kanban controlled system

station is composed of two types of buffers. One type is part buffer (B_1, B_4), and the other type is kanban buffer (B_2, B_5). Also, each machining station has a buffer (B_3) for the parts waiting for the machine to be available or in process by the machine. Throughout the work, S_p denotes a machining station p that processes products. The service rate of machine p is denoted by μ_p.

The kanbans are used to control the system operation as follows. At the upstream synchronization station, raw materials are synchronized with free kanbans. As soon as an entity is available in both B_1 and B_2, the kanban is attached to the part and sent to the machining station buffer, B_3. At the downstream synchronization station, finished parts are synchronized with customer demand. As soon as an entity is available in both B_4 and B_5, the finished part leaves the system and the detached kanban is then sent to the upstream kanban buffer, B_2, to signal another demand for raw material and authorize the raw material to be transferred to the processing station.

The variables for the nonlinear programming model are defined in terms of two types of stochastic processes:

$X_i(t)$ = number of entities in buffer B_i (including one in service if i is a machining station buffer) at time t

$$Y_i(t) = \begin{cases} 1, \text{ if } X_i(t) > 0 \\ 0, \text{ if } X_i(t) = 0 \end{cases}$$

The following are additional notations used throughout this research.

L_p: set of buffer indices in the kanban loop of machine station S_p, for example in Figure 2, $L_1 = \{2, 3, 4\}$

K_p : number of kanbans of the loop of the p^{th} machine station

The quantity K_p is constant when the objective of problem is to minimize or maximize throughputs in the system or of specific products. But K_p becomes a decision variable when the objective is to minimize the number of kanbans in the system while satisfying given throughput requirements. Throughout the study, the term *kanban configuration* denotes the number of kanbans in each loop. We assume that a steady state is reached. In section 3.2, we introduce constraints that follow from assuming stationary first, second and cross-moments of $\{X_i(t)\}$. Theses constraints motivate the definition of decision variables that represent the expected values of buffer populations and utilizations in steady state. Let

$$w_i = \lim_{t\to\infty} \mathrm{E}[Y_i(t)]$$

$$\beta_i = \lim_{t\to\infty} \mathrm{E}[X_i(t)]$$

when these limit exist. The variable β_i is the expected population of buffer B_i and w_i is its expected utilization in steady state. Additional decision variables that are needed to express stationary first and second moment constraints represent products of buffer populations and/or one or more buffer utilization. When the corresponding limits exist, the variables are

$$z_{ij} = \lim_{t\to\infty} \mathrm{E}[Y_i(t)Y_j(t)], \text{ where } i \le j$$

$$\alpha_{ijk} = \lim_{t\to\infty} \mathrm{E}[Y_i(t)Y_j(t)Y_k(t)] \text{ where } i \le j \le k$$

$$\gamma_{ij} = \lim_{t\to\infty} \mathrm{E}[X_i(t)Y_j(t)]$$

$$\delta_{ijk} = \lim_{t\to\infty} \mathrm{E}[X_i(t)Y_j(t)Y_k(t)] \text{ where } j \le k$$

$$\varepsilon_{ijkl} = \lim_{t\to\infty} \mathrm{E}\left[X_i(t)Y_j(t)Y_k(t)Y_l(t) \right] \text{ where } j \le k \le l$$

The constraints in sections 3.1 and 3.2 are developed in under the assumption that the system will reach in steady state. However, in section 3.3, we will see that when raw material and demand arrivals are random, this assumption is not valid.

3.1 General Constraints

The stationary first and second moment constraints to be derived in section 3.2 are specific to the queueing network being modeled. We start with presenting constraints common for all queueing networks. First, notice that $X_i(t)Y_i(t) = X_i(t)$ because $Y_i(t) = 1$ when $X_i(t) > 0$ Also, at any synchronization station, at most one of the kanban buffer and the part buffer can be occupied at any time. The following equalities hold for the kanban system when i is a part buffer and j is the corresponding kanban buffer.

$$X_i(t)X_j(t) = X_i(t)Y_j(t) = Y_i(t)X_j(t) = Y_i(t)Y_j(t) = 0 \tag{1}$$

For any i, p where $i \in L_p$, $Y_i(t) = 1$ implies $1 \le X_i(t) \le K_p$ and $Y_i(t) = 0$ implies $X_i(t) = 0$. Therefore, $X_i(t) \le K_p Y_i(t)$. Then, by taking expected value on both sides, the following inequalities can be obtained.

$$E\left[X_i(t)\right] \le K_p E\left[Y_i(t)\right], \text{ for all } i, p, t \text{ where } i \in L_p$$

The constraints for the system in steady state can be written as follows.

$$\beta_i \le K_p w_i, \text{ for all } i, p \text{ where } i \in L_p \tag{2}$$

The following constraints follow from the assumption that the population of a loop, L_p, is a constant equal to the number of kanbans.

$$\sum_{i \in L_p} E\left[X_i(t)\right] = K_p \text{ for all } p,\ t$$

Equivalently in steady state,

$$\sum_{i \in L_p} \beta_i = K_p, \text{ for all } p \tag{3}$$

The constant population property motivates additional constraints since β_i is the expected buffer population in steady state and not the actual buffer population. The population of a loop must be constant in any situation. The following must be true.

$$\sum_{i \in L_p} E\left[X_i(t)Y_j(t)\right] = K_p E\left[Y_j(t)\right], \forall (j,p)$$

$$\sum_{i \in L_p} E\left[X_i(t)Y_j(t)Y_k(t)\right] = K_p E\left[Y_j(t)Y_k(t)\right], \forall (j,k,p)$$

$$\sum_{i \in L_p} E\left[X_i(t)Y_j(t)Y_k(t)Y_l(t)\right] = K_p E\left[Y_j(t)Y_k(t)Y_l(t)\right], \forall (j,k,l,p)$$

For the system in steady state, the above equalities can be expressed in terms of the decision variables as follows.

$$\sum_{i \in L_p} \gamma_{ij} = K_p w_j, \forall (j,p)$$

$$\sum_{i \in L_p} \delta_{ijk} = K_p z_{jk}, \forall (j,k,p)$$

$$\sum_{i \in L_p} \epsilon_{ijkl} = K_p \alpha_{jkl}, \forall (j,k,l,p) \tag{4}$$

Schweitzer et al. (1986) defined a correction term in the expected sojourn time for mean value analysis (MVA) of a closed queueing network. In steady state, the expected time

that a part spends at a machining station is given by

$D = \frac{1}{\mu_p} + \frac{1}{\mu_p}\beta_j + e$, where β_j is the expected population in the buffer of machine station p

Here, $e < 0$ is a correction factor to avoid the possibility that the service time for a part may be counted twice in its sojourn time. Schweitzer et al. (1986) defined the correction factor as

$$e = -\frac{\beta_j}{\mu_p K_p}$$

From Little's Law, we can write the sojourn time as

$$D = \frac{\beta_j}{\mu_p w_j}$$

We can now derive the following equation.

$$\beta_j = w_j\left(1 + \beta_j\right) + \mu_p w_j e$$

Since the correction factor is not exact in all cases and $e < 0$, we can not obtain an equality constraint but instead have the following inequality constraints.

$$\beta_j \le w_j\left(1 + \beta_j\right), \forall j \tag{5}$$

The next two constraints ensure that the utilization of buffers does not exceed 100%. First, the steady state utilization of each buffer must be less than 1. This requirement can be expressed as

$$w_i \le 1, \text{ where } B_i \text{ is a machine station buffer} \tag{6}$$

Since both the kanban buffer and the part buffer of a fork/join synchronization station cannot be occupied at the same time, it is true that if B_i and B_j are corresponding part and kanban buffers, then

$$P(Y_i(t) = 1) = E\left[Y_i(t)\right]$$

$$Y_i(t)Y_j(t) = 0$$

and

$$P(Y_i(t) = 1 \text{ or } Y_j(t) = 1) + P(Y_i(t) = Y_j(t) = 0)$$

$$= P(Y_i(t) = 1) + P(Y_j(t) = 1) + P(Y_i(t) = Y_j(t) = 0)$$

$$= 1.$$

Then the following inequality can be obtained in steady state.

$$w_i + w_j \leq 1, \text{ where } i \text{ is part buffer and } j \text{ is the corresponding kanban buffer} \tag{7}$$

Also, since $X_i(t) \geq Y_i(t)$, the following inequalities can be obtained.

$$w_i \leq \beta_i, \ \forall i \tag{8}$$

$$z_{ij} \leq \gamma_{ji}, \ \forall i, j$$

$$z_{ij} \leq \gamma_{ij}, \ \forall i, j \tag{9}$$

$$\alpha_{ijk} \leq \delta_{ijk}, \ \forall i, j, k$$

$$\alpha_{ijk} \leq \delta_{jik}, \ \forall i, j, k$$

$$\alpha_{ijk} \leq \delta_{kij}, \ \forall i, j, k \tag{10}$$

Now, since $Y_i(t) \leq 1$ for all i, the following equations define the relationship between utilization variables.

$$z_{ij} \leq w_i, \ \forall i, j$$

$$z_{ij} \leq w_j, \ \forall i, j \tag{11}$$

$$\alpha_{ijk} \leq z_{ij}, \ \forall i, j, k \tag{12}$$

$$\alpha_{ijk} \le z_{ik},\ \forall i,j,k$$

$$\alpha_{ijk} \le z_{jk},\ \forall i,j,k$$

The following inequalities show relationships between buffer population variables.

$$\gamma_{ij} \le \beta_i,\ \forall i,j$$

$$\gamma_{ij} \le \beta_j,\ \forall i,j \tag{13}$$

$$\delta_{ijk} \le \gamma_{ij},\ \forall i,j,k$$

$$\delta_{ijk} \le \gamma_{ik},\ \forall i,j,k \tag{14}$$

Lastly, the inequality $X_i(t)Y_j(t) \le (K_p - 1)Y_i(t)$ must be true since

$$X_i(t)Y_j(t) = 0 \text{ when } Y_i(t) = 0 \text{ or } Y_j(t) = 0$$

and

$$X_i(t)Y_j(t) = X_i(t) \le X_i(t) + X_j(t) - 1 \le K_p - 1, \text{ if } Y_i(t) = Y_j(t) = 1$$

By taking expectation on both side of the inequality $X_i(t)Y_j(t) \le (K_p - 1)Y_i(t)$, we obtain the following constraint.

$$\gamma_{ij} \le (K_p - 1)w_i, \text{ where } i,j \in L_p \tag{15}$$

As mentioned before, the constraints derived in this section can be used to find throughput bounds for given a kanban configuration or to find a best kanban configuration for given throughput requirements. For the problem of finding throughput bounds, the buffer population, β_i, and buffer utilizations, w_i, are the primary variables and only inequality (5) generates nonlinear constraints. On the other hand, for the problem of finding a kanban configuration, many other constraints, i.e. (2), (4), become nonlinear because the kanban counts, K_p, are also decision variables.

3.2 Moment Constraints

Figure 3 shows a queueing network of a single-stage kanban system with Poisson supply and demand arrival rates of λ and ν. In addition to the constraints derived in section 3.1, constraints that follow from stationarity of the first and second moments of buffer population can be obtained. For the supply part buffer, B_1, the buffer population increases by one when a supply arrives and B_2 is not empty and it decreases by one when a kanban arrives at B_2 and B_1 is not empty. The buffer queue dynamics are presented in Table 1.

Now, the stationary first moment equalities of the first buffer can be found. We use a similar methodology as in Kumar and Kumar (1994). Let τ_n be the time point at which the n^{th} change, which would be raw material arrival, demand arrival or service completion, in the system state occurs. Then, let F_{τ_n} denote the σ-field generated by the events up to time τ_n. The buffer population at time τ_{n+1} is

$X_1(\tau_{n+1}) = X_1(\tau_n) + 1$: if the event at τ_{n+1} is an arrival of supply and $Y_2(\tau_n) = 0$

$= X_1(\tau_n) - 1$: if the event at τ_{n+1} is a real service completion at the buffer B_3 and

$Y_1(\tau_n) = 1$, $Y_5(\tau_n) = 1$, or

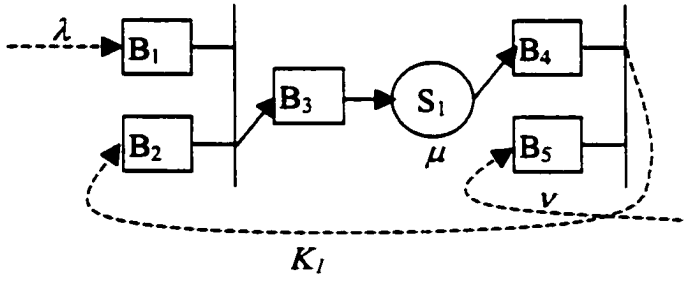

Figure 3. Queueing network of single-stage kanban system

Table 1. Queue dynamics by event

Event	Buffer Status	Buffer Change				
		B_1	B_2	B_3	B_4	B_5
Supply Arrival	$X_2=0$	↑				
	$X_2>0$		↓	↑		
Service Completion at S	$X_3>0, X_5=0$			↓	↑	
	$X_1=0, X_3>0, X_5>0$		↑	↓		↓
	$X_1>0, X_3>0, X_5>0$	↓				↓
Demand Arrival	$X_4=0$					↑
	$X_1=0, X_4>0$		↑		↓	
	$X_1>0, X_4>0$	↓		↑	↓	

if the event at τ_{n+1} is an arrival of demand and $Y_1(\tau_n)=1$, $Y_4(\tau_n)=1$

$= X_1(\tau_n)$: otherwise

For the single stage kanban system, the "total" rate at which events occur is $\lambda+\mu+\nu$. The system state changes when these events occur. But, note that some of these are virtual changes, e.g. if "service completion" occurs when $X_3(\tau_n)=0$. Given information about events up to time τ_n, the probability that $X_1(\tau_{n+1})=X_1(\tau_n)+1$ is $\frac{\lambda(1-Y_2(\tau_n))}{\lambda+\mu+\nu}$ where $\frac{\lambda}{\lambda+\mu+\nu}$ is the probability that the λ-Poisson process "wins" and $1-Y_2(\tau_n)$ is the probability that this results in an increase in X_1. The probability that $X_1(\tau_{n+1})=X_1(\tau_n)-1$ is $\frac{\mu Y_1(\tau_n)Y_5(\tau_n)Y_3(\tau_n)}{\lambda+\mu+\nu}+\frac{\nu Y_1(\tau_n)Y_4(\tau_n)}{\lambda+\mu+\nu}$. $Y_3(\tau_n)$ is the probability that the service completion is real, not virtual, and $Y_1(\tau_n)Y_5(\tau_n)$ is the probability that the kanban attached to this part is freed, then matched with raw material waiting in B_1. Lastly, the probability that

$X_1(\tau_{n+1}) = X_1(\tau_n)$ is $1 - \dfrac{\lambda(1-Y_2(\tau_n)) + \mu Y_1(\tau_n)Y_5(\tau_n)Y_3(\tau_n) + \nu Y_1(\tau_n)Y_4(\tau_n)}{\lambda+\mu+\nu}$. Then we can obtain the conditional expectation

$$E\left[X_1(\tau_{n+1})\middle|F_{\tau_n}\right] = X_1(\tau_n) + \frac{\lambda}{\lambda+\mu+\nu}\left(1-Y_2(\tau_n)\right) - \frac{\mu}{\lambda+\mu+\nu}Y_1(\tau_n)Y_3(\tau_n)Y_5(\tau_n) - \frac{\nu}{\lambda+\mu+\nu}Y_1(\tau_n)Y_4(\tau_n)$$

Then, we take unconditional expectations on both sides to obtain

$$E[X_1(\tau_{n+1})] = E[X_1(\tau_n)] + \frac{\lambda}{\lambda+\mu+\nu}\left(1-E[Y_2(\tau_n)]\right) - \frac{\mu}{\lambda+\mu+\nu}E[Y_1(\tau_n)Y_3(\tau_n)Y_5(\tau_n)]\backslash - \frac{\nu}{\lambda+\mu+\nu}E[Y_1(\tau_n)Y_4(\tau_n)]$$

and assume $E[X_1(\tau_{n+1})] = E[X_1(\tau_n)]$ in steady state. We have the following stationary first moment equality for the first buffer

$$\frac{\lambda}{\lambda+\mu+\nu}\left(1-E[Y_2(\tau_n)]\right) - \frac{\mu}{\lambda+\mu+\nu}E[Y_1(\tau_n)Y_3(\tau_n)Y_5(\tau_n)] - \frac{\nu}{\lambda+\mu+\nu}E[Y_1(\tau_n)Y_4(\tau_n)] = 0$$

In steady state, it is equivalent to

$$\lambda - \lambda w_2 - \nu z_{14} - \mu\alpha_{135} = 0$$

Similarly, first moment equalities for other buffers are obtained and shown in Table 2. Note

Table 2. First moment equalities for single stage kanban system

Buffer	Equalities
B_1	$\lambda - \lambda\omega_2 - \nu z_{14} - \mu\alpha_{135} = 0$
B_2	$\lambda\omega_2 + \nu z_{14} + \mu\alpha_{135} - \nu\omega_4 - \mu z_{35} = 0$
B_3	$\lambda\omega_2 + \nu z_{14} + \mu\alpha_{135} - \mu\omega_3 = 0$
B_4	$\mu\omega_3 - \nu\omega_4 - \mu z_{35} = 0$
B_5	$\nu\omega_4 + \mu z_{35} - \nu = 0$

that, taken together, these constraints require $\lambda = \nu$. Second moment equalities can be obtained in a similar fashion. The square of the population of buffer B_1 can be expressed as

$X_1^2(\tau_{n+1}) = (X_1(\tau_n)+1)^2$: if the event at τ_{n+1} is arrival of supply and $Y_2(\tau_n) = 0$

$= (X_1(\tau_n)-1)^2$: if the event at τ_{n+1} is a real service completion at the buffer B_3 and

$Y_1(\tau_n) = 1,\ Y_5(\tau_n) = 1$

if the event at τ_{n+1} is arrival of demand and $Y_1(\tau_n) = 1,\ Y_4(\tau_n) = 1$

$= (X_1(\tau_n))^2$: otherwise

Its conditional expectation can be written as

$$(\lambda+\mu+\nu)E\left[X_1^2(\tau_{n+1})\middle|F_{\tau_n}\right] = (\lambda+\mu+\nu)X_1^2(\tau_n) + 2\lambda X_1(\tau_n) - 2\mu X_1(\tau_n)Y_3(\tau_n)Y_5(\tau_n)$$
$$-2\nu X_1(\tau_n)Y_4(\tau_n) + \lambda - \lambda Y_2(\tau_n) + \mu Y_1(\tau_n)Y_3(\tau_n)Y_5(\tau_n)$$
$$+\nu Y_1(\tau_n)Y_4(\tau_n)$$

Then, we take unconditional expectations on both sides and obtain

$$(\lambda+\mu+\nu)E\left[X_1^2(\tau_{n+1})\right] = (\lambda+\mu+\nu)E\left[X_1^2(\tau_n)\right] + 2\lambda E\left[X_1(\tau_n)\right] - 2\mu E\left[X_1(\tau_n)Y_3(\tau_n)Y_5(\tau_n)\right]$$
$$-2\nu E\left[X_1(\tau_n)Y_4(\tau_n)\right] + \lambda - \lambda E\left[Y_2(\tau_n)\right] + \mu E\left[Y_1(\tau_n)Y_3(\tau_n)Y_5(\tau_n)\right]$$
$$+\nu E\left[Y_1(\tau_n)Y_4(\tau_n)\right]$$

and set $E\left[X_1^2(\tau_{n+1})\right] = E\left[X_1^2(\tau_n)\right]$.

The second moment constraint for the first buffer is

$$2\lambda E\left[X_1(\tau_n)\right] - 2\mu E\left[X_1(\tau_n)Y_3(\tau_n)Y_5(\tau_n)\right] - 2\nu E\left[X_1(\tau_n)Y_4(\tau_n)\right] + \lambda - \lambda E\left[Y_2(\tau_n)\right]$$

$$+\mu E\left[Y_1(\tau_n)Y_3(\tau_n)Y_5(\tau_n)\right]+\nu E\left[Y_1(\tau_n)Y_4(\tau_n)\right]=0$$

In steady state, it is equivalent to

$$2\lambda\beta_1-2\mu\delta_{135}-2\nu\gamma_{14}+\lambda-\lambda w_2+\mu\alpha_{135}+\nu z_{14}=0$$

The second moment constraint can be simplified by substituting the corresponding first moment constraint. Since $\nu z_{14}+\mu\alpha_{135}=\lambda(1-w_2)$ we get

$$\lambda\beta_1-\mu\delta_{135}-\nu\gamma_{14}+\lambda(1-w_2)=0$$

Other second moment equalities are shown in Table 3.

Similarly, cross moment equalities between two different buffers can be obtained. For B_1 and B_2, no meaningful equalities cannot obtained since one is a part buffer and the other one is the corresponding kanban buffer and $X_1(t)X_2(t)=0$ for all t. The equality second moment constraint of B_1 and B_3 can be derived from

$X_1(\tau_{n+1})X_3(\tau_{n+1})=\left(X_1(\tau_n)+1\right)X_3(\tau_n)$: if the event at τ_{n+1} is arrival of supply and $Y_2(\tau_n)=0$

$=\left(X_1(\tau_n)-1\right)X_3(\tau_n)$: if the event at τ_{n+1} is service completion and

$$Y_1(\tau_n)=1, Y_3(\tau_n)=1,\ Y_5(\tau_n)=1$$

$=\left(X_1(\tau_n)-1\right)\left(X_3(\tau_n)+1\right)$: if the event at τ_{n+1} is arrival of demand and

Table 3. Second moment equalities for single-stage kanban system

Buffer	Equalities
B_1, B_1	$\lambda\beta_1-\mu\hat{o}_{135}-\nu\gamma_{14}+\lambda(1-\omega_2)=0$
B_2, B_2	$\lambda\beta_2-\lambda\omega_2-\mu\hat{o}_{235}-\nu\gamma_{24}=0$
B_3, B_3	$\lambda\gamma_{32}+\nu\hat{o}_{314}+\lambda\omega_2+\nu z_{14}-\mu\beta_3+\mu\hat{o}_{315}=0$
B_4, B_4	$\mu\gamma_{43}-\nu\beta_4+\nu\omega_4=0$
B_5, B_5	$\nu\beta_5+\nu(1-\omega_4)-\mu\gamma_{53}=0$

$$Y_1(\tau_n) = 1,\ Y_4(\tau_n) = 1$$

$= X_1(\tau_n)\big(X_3(\tau_n)+1\big)$: if the event at τ_{n+1} is arrival of supply and $Y_2(\tau_n) = 1$

$= X_1(\tau_n)\big(X_3(\tau_n)-1\big)$: if the event at τ_{n+1} is service completion and

$$Y_3(\tau_n) = 1, Y_5(\tau_n) = 0$$

if the event at τ_{n+1} is service completion and

$$Y_1(\tau_n) = 0,\ Y_3(\tau_n) = 1,\ Y_5(\tau_n) = 1$$

$= X_1(\tau_n)X_3(\tau_n)$: otherwise

Following the same steps with the previous two constraints, setting $E\big[X_1(\tau_{n+1})X_3(\tau_{n+1})\big]$ $= E\big[X_1(\tau_n)X_3(\tau_n)\big]$, the following equality is obtained.

$$\lambda\beta_3 - \lambda\gamma_{32} - \mu\delta_{315} - \mu\delta_{314} + \nu\gamma_{14} - \nu\gamma_{13} + \mu\delta_{135} - \nu z_{14} = 0$$

Other cross moment equalities are shown in Table 4.

3.3 Infeasibility of the Single Stage Kanban System with Poisson Process Supply and Demand

The number of buffers required in a kanban system is three times the number of machines in the system. If the number of machines in a system is n, the number of stationary first and second moment constraints are $3n$ and $3n(3n+1)/2$. Furthermore, the number of variable subscripts required for the system is equal to $n+1$. Because the size of the NLP model quickly increases as the number of machines increases and it becomes very hard to

Table 4. Cross moment equalities between different pair of buffers for the single-stage kanban system

Buffers	Equalities
(B_1,B_2)	None
(B_1,B_3)	$\lambda\beta_3 - \lambda\gamma_{32} - \mu\delta_{315} - \nu\delta_{314} + \nu\gamma_{14} - \mu\gamma_{13} + \mu\delta_{135} - \nu z_{14} = 0$
(B_1,B_4)	$\lambda\beta_4 - \lambda\gamma_{42} - \nu\gamma_{41} + \mu\gamma_{13} - \mu\delta_{135} - \nu\gamma_{14} + \nu z_{14} = 0$
(B_1,B_5)	$\lambda\beta_5 - \lambda\gamma_{52} - \mu\delta_{513} + \nu\beta_1 - \nu\gamma_{14} - \mu\delta_{135} + \mu\alpha_{135} = 0$
(B_2,B_3)	$\mu\gamma_{35} - \mu\delta_{315} + \nu\gamma_{34} - \nu\delta_{314} - \lambda\gamma_{32} + \lambda\beta_2 - \mu\gamma_{23} - \mu z_{35} + \mu\alpha_{135} - \lambda\omega_2 = 0$
(B_2,B_4)	$\nu\beta_4 - \nu\gamma_{41} - \lambda\gamma_{42} + \mu\gamma_{23} - \mu\delta_{235} - \nu\gamma_{24} - \nu\omega_4 + \nu z_{14} = 0$
(B_2,B_5)	$\mu\gamma_{53} - \mu\delta_{513} - \lambda\gamma_{52} + \nu\beta_2 - \nu\gamma_{24} - \mu\delta_{235} - \mu z_{35} + \mu\alpha_{135} = 0$
(B_3,B_4)	$\lambda\gamma_{42} + \nu\gamma_{41} - \mu\gamma_{43} + \mu\beta_3 - \mu\gamma_{35} - \nu\gamma_{34}\ \mu\omega_3 + \mu z_{35} - \nu\gamma_{14} = 0$
(B_3,B_5)	$\lambda\gamma_{52} - \mu\gamma_{53} + \mu\delta_{513} + \nu\beta_3 - \nu\gamma_{34} - \mu\gamma_{35} + \mu z_{35} - \mu\alpha_{135} = 0$
(B_4,B_5)	None

track variables, it is desired to develop a modularized NLP model for kanban controlled queuing network. Figure 4 shows how a series of kanban controlled system can be decomposed into set of single machine kanban controlled system. The modularized kanban controlled queueing network is shown in the lower portion of Figure 4 with supply and demand arrival rates of λ and ν.

In the modularized system, λ would be the rate of arrivals of parts from the upstream station and ν would be the rate of demand. The processes governing these arrivals are not known. To explore the feasibility of the modularization approach, we study the single-stage system in which the supply and demand arrivals follow Poisson process with respective rates λ and ν. Unfortunately, this model is infeasible. The cause of this infeasibility is the first and second moment constraint equalities of the supply and demand buffers. The difficulty is seen by taking left hand sides of equalities and doing a little arithmetic. Let $S(B_i,B_j)$ be the second

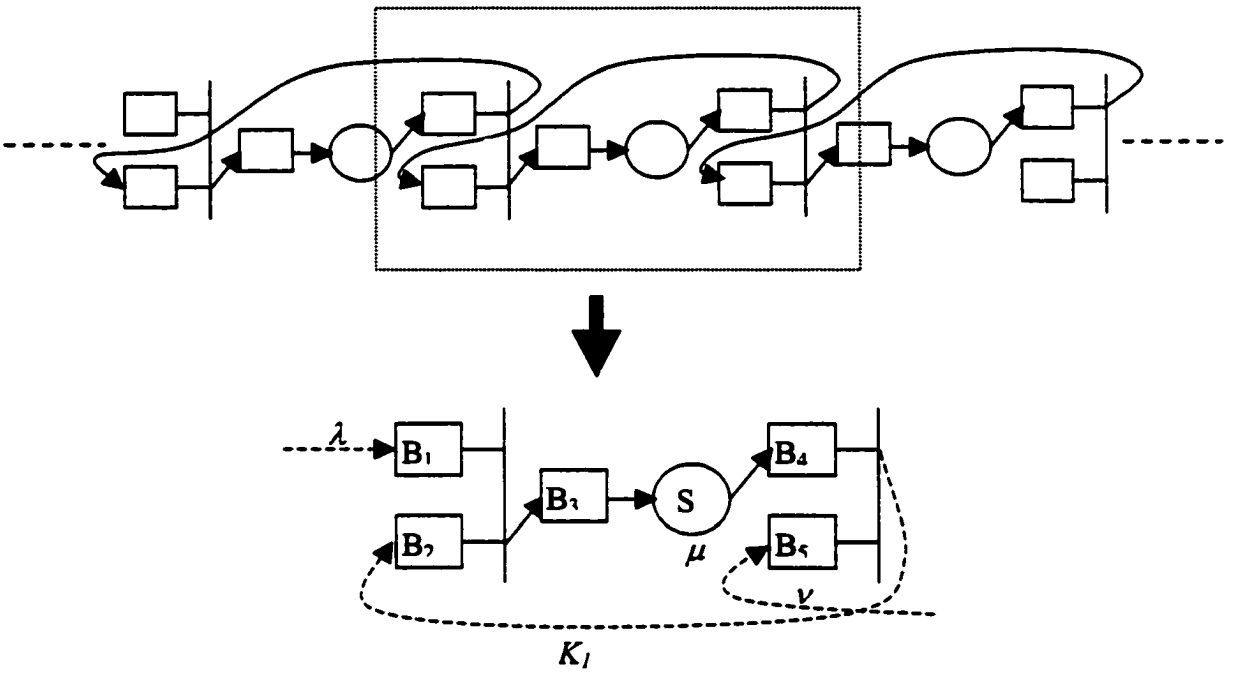

Figure 4. Modularized queueing network of a kanban control system

moment constraint of B_i and B_j and $S_L(B_i,B_j)$ to be the left hand side of $S(B_i,B_j)$. Then, when we calculate $S_L(B_1,B_5)$ - $S_L(B_1,B_1)$ - $S_L(B_5,B_5)$ - $S_L(B_2,B_5)$ + $S_L(B_2,B_2)$, the following is obtained

$$\lambda\beta_5 + \nu\beta_1 - \lambda\beta_1 - \lambda - \nu\beta_5 - \nu + \nu w_4 - \nu\beta_2 + \mu z_{35} + \lambda\beta_2 = 0$$

From the stationary first moment constraints, we know $\nu\omega_4 + \mu z_{35} - \nu = 0$ and $\lambda = \nu$. The above equality can be simplified as

$$-\lambda = 0$$

Also, when we set the number of kanbans, K_1, equal to 1, the values of a population variable and its corresponding utilization variables must be the same, $w_i = \beta_i$, $z_{ij} = \gamma_{ij}$

and $\alpha_{ijk} = \delta_{ijk}$. Let $F(B_i)$ be the stationary first moment constraint of B_i and $F_L(B_i)$ be the left hand side of $F(B_i)$. We can obtain the following equalities.

$$F_L(B_1)+S_L(B_1,B_1) = \lambda w_1 = 0$$

$$F_L(B_5)+S_L(B_5,B_5) = \nu w_5 = 0$$

To reach steady state where $\lambda = \nu > 0$, the utilization of supply and demand buffers must be equal to zero, $w_1 = 0$, $w_5 = 0$. Then, we can see from the first moment constraints, $F(B_1)$, $F(B_3)$ and $F(B_5)$, that $w_2 = w_4 = 1$, $w_3 = \lambda / \mu$. But this is not possible since it violates the condition $w_2 + w_3 + w_4 = 1$ that follows when the number of kanbans equals 1.

A simulation study was conducted to verify that steady state can really not be reached. The supply arrival and demand arrival rates are equal to 1 and the service rate is equal to 2. The length of each simulation run is 500000 time units. Figure 5 shows the change of supply and demand buffer populations after three different replications. The maximum population of buffers shown in the figure is 500 although there is no limit on buffer capacities in the

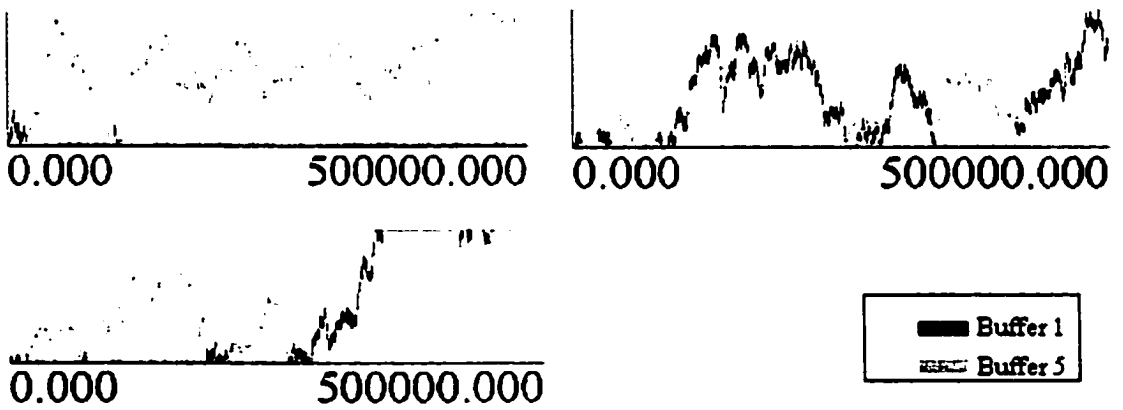

Figure 5. Buffer population change of single-stage kanban ($\lambda = \nu = 1, \mu = 2$)

simulation. It seems that there is no pattern of buffer populations. The possible cause of the infeasibility and its implication is further discussed in Chapter 7. Although the modularization method appears hopeless, we continue to study kanban policy because modularization idea could potentially work if we had the true upstream supply replacing the Poisson process with rate λ and the true downstream demand instead of the Poisson process with rate ν. But we do not know what the processes are. The next sections will show that a feasible solution exists corresponding to existence of steady state if either supply or demand is infinite.

3.4 Infinite Supply and Demand

Figure 6 shows the changes from Figure 2 when we assume that there are infinite quantities of raw materials available. Because there will be always raw material at the buffer 1, the raw material will be sent to buffer 3 as soon as a kanban is available. Buffers 1 and 2 can be omitted in our analysis of the queuing network. Similarly, Figure 7 shows the changes from Figure 2 when there is infinite demand. We only need to consider the first 3 buffers. Notice that there must be some kanbans in either or both of the buffers 3 and 4 in Figure 6.

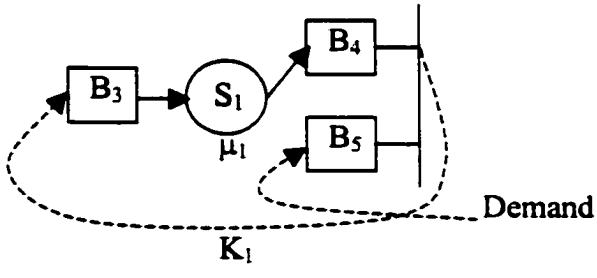

Figure 6. Closed queueing network model with infinite supply

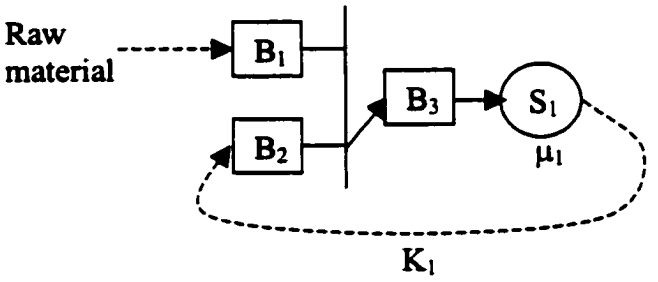

Figure 7. Closed queueing network model with infinite demand

Also, there are kanbans only in buffer 3 if buffer 5 is not empty. Now, we generalize the infinite supply case by denoting the server as S_p. Also, we label machine, part, and kanban buffers as B_i, B_j and B_k. In order to make the model tighter, additional constraints are developed for the case of the infinite supply of raw materials shown in Figure 6.

Since all kanbans are either in B_i or B_j, the joint utilization variable of B_i and B_j can be expressed as

$$z_{ij} = w_i + w_j - 1 \tag{16}$$

There are only three different conditions of buffers B_i and B_j: that either one of the buffers is occupied, $Y_i(t) = 1, Y_j(t) = 0$ or $Y_i(t) = 0, Y_j(t) = 1$, or both buffers are occupied, $Y_i(t) = 1, Y_j(t) = 1$. If either one of the buffers is occupied, the number of kanbans in the occupied buffer is equal to K_p. This property can be expressed the as following equalities.

$$\begin{aligned} \beta_i - \gamma_{ij} &= K_p(w_i - z_{ij}) \\ \beta_j - \gamma_{ji} &= K_p(w_j - z_{ij}) \end{aligned} \tag{17}$$

Lastly, we can express the buffer population $\beta_k = \gamma_{ki} + \gamma_{kj} - \delta_{kij}$. Since B_k and B_j are the

corresponding kanban and part buffers, we know that $\gamma_{kj} = 0, \delta_{kij} = 0$. The following equality is obtained.

$$\gamma_{ki} = \beta_k \tag{18}$$

In the case of infinite supply, there must be some kanbans in either or both of buffers 2 and 3. Also, there are kanbans only in buffer 3 if buffer 1 is not empty. Again, we generalize by denoting the server as S_p. Also, we denote part, kanban and machine buffers as B_i, B_j and B_k. In a similar fashion, we have the following constraints for the infinite demand.

$$z_{jk} = w_j + w_k - 1 \tag{19}$$

$$\beta_j - \gamma_{jk} = K_p(w_j - z_{jk})$$

$$\beta_k - \gamma_{kj} = K_p(w_k - z_{jk}) \tag{20}$$

$$\gamma_{ik} = \beta_i \tag{21}$$

3.5 Three Machine Flow Line under Kanban Policy

In this section, we present the NLP model for a single product flow line queueing network with three serial machines. The system is controlled by a kanban policy in which each machine has its own kanban loop. It is assumed that there is an infinite supply of raw materials and infinite demand of the final product. Figure 8 shows a queueing network model with three serial machining stations. There are two fork/join stations and three kanban loops. The first fork/join station synchronizes the first and second kanban loops and the second fork/join station synchronizes second and third kanban loops. The objective of the model is

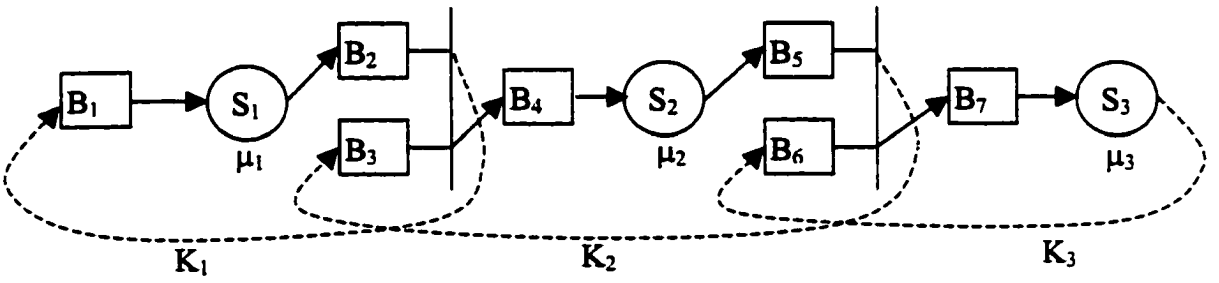

Figure 8. Closed queueing network model of three serial machines

to find the minimum number of kanbans for a given throughput requirement or the maximum throughput for a given number of kanbans, where servers may have different exponential service times. Now, we can find the stationary first moment equalities of the first buffer. These can be interpreted as equating the rate of traffic into and out of each buffer. The complete list of stationary first and second moment constraints are presented in Appendix 2.

The nonlinear programming model for the flow line presented in Figure 8 was examined. The model includes the constraints (2) – (21) and the stationary first and second moment constraints. We assumed that the model has infinite supply and demand. We performed optimization of the model in two different ways. First, we wanted to maximize and minimize the total throughput to find the bounds on throughput for a given set of kanban levels. Second, we minimized the total number of kanbans necessary to achieve a given throughput requirement.

The objective function for the throughput maximization can be written as Max $\mu_1 w_1$ $(=\mu_2 w_4 = \mu_3 w_7)$. The throughput can be maximized for a given set of service rates (μ_1, μ_2, and μ_3) and number of kanbans (K_1, K_2, and K_3). The nonlinear programming problems in this research were solved by LINGO software (LINDO Systems Inc., 1996).

Complete model in LINGO is presented in Appendix 2. We also constructed a simulation model using ARENA software to verify the NLP solution. The length of the warm-up period was 10,000 time units and the length of each simulation run was 100,000 time units.

Table 5 shows the comparison between optimization and simulation results. All service rates were equal to 1. The result shows that the NLP solution provides valid lower and upper bounds although the gap between the bounds is quite wide compared with the simulation confidence interval. The reason of the wide bounds may be explained that the bounds are the best and worst case performances. In the best case, the timing of the kanban-part synchronizations happens in the best possible way. In other words, the gap between an arrival of finished part from upstream and the arrival of kanban at a synchronization is the maximum possible.

Table 6 shows the kanban configuration that the objective is to minimize the total number of kanbans for a specified throughput requirements. Again all service rates were equal to 1. Unfortunately, there is no guarantee that the system with the kanban configuration

Table 5. Comparison between NLP solutions and simulation results

K_1	K_2	K_3	Optimization			Simulation	
			Min	Max	Avg.	Throughput	95% CI
1	1	1	0.333	0.667	0.500	0.561	± 0.007
1	2	1	0.556	0.857	0.707	0.648	± 0.008
2	1	1	0.382	0.750	0.566	0.610	± 0.007
1	1	2	0.500	0.667	0.584	0.610	± 0.007
2	2	2	0.473	0.875	0.674	0.722	± 0.008
5	5	5	0.737	0.969	0.853	0.867	± 0.010
10	10	10	0.859	0.991	0.925	0.930	± 0.010

Table 6. Minimization of total number of kanbans

Throughput Requirement	K_1	K_2	K_3	$\sum_{p=1}^{3} K_p$	Throughput (NLP)	Throughput (Simulation)
0.50	2	1	1	4	0.593	0.611 ± 0.004
0.60	2	1	1	4	0.600	0.611 ± 0.004
0.70	2	1	1	4	0.700	0.611 ± 0.004
0.80	2	2	1	5	0.800	0.686 ± 0.006
0.90	2	3	1	6	0.900	0.735 ± 0.005
0.95	2	4	1	7	0.953	0.766 ± 0.005
0.99	2	10	1	13	0.991	0.863 ± 0.006

given in Table 6 will actually satisfy the throughput requirements. The simulation result confirms that only one of the kanban configurations meets the throughput requirement. The reason is that the throughput in Table 6 only indicates that the maximum throughput of the kanban configuration in the solution exceeds the throughput requirements. The actual throughput may not meet the requirements. A heuristic method for obtaining the minimum total number of kanbans is presented in Chapter 5.

In this chapter, an analytical model for a three serial machine kanban queueing network was presented. Also, it was attempted to modularize kanban systems so that each kanban loop is analyzed separately in earlier section. While this modeling approach is feasible when there is infinite supply and demand, the NLP model grows in complexity very quickly as the number of stations and kanban loops increases. The infeasibility of the single-stage model with Poisson supply and demand suggests that decomposing a multistage system in to single-stage modules may not be practical. In the next chapter, we explore the same approach for CONWIP systems, in which a single kanban loop controls several machines.

CHAPTER 4 CONWIP SYSTEMS

A kanban policy is difficult to use when there are job orders with short production runs, significant setups, scrap loss or large, unpredictable fluctuations in demand (see Monden 1983, and Spearman, Woodruff and Hopp 1990). The kanban policy is relatively difficult to design because a kanban configuration specifying the number of kanbans for each machine must be obtained. On the other hand, designing a CONWIP policy only requires finding a single number of kanbans for the whole system. Also, adding new kanban(s) or removing them is much easier by simply adding them at the first station in the system or removing them at the last station. Furthermore, as Krishnamurthy, Suri and Vernon (2000) pointed out, a kanban system may require more WIP than push systems when low throughput is required. The reason is that each machine requires a certain number of kanbans even if the throughput requirement is low. On the other hand, CONWIP needs only a small number of kanbans for the whole system. In this chapter, we present a model for analyzing the performance of CONWIP systems.

4.1 Modeling CONWIP Systems

Modeling a flow line CONWIP system is much simpler than modeling a series of kanban systems because there is no fork/join station within the flow line. The dynamics of a CONWIP system are simple: completion of service on a machine reduces its buffer population by one and increases buffer population of next machine by one (see Table 7). Modeling a kanban system is much more complex because the complexity of the model

increases as the number of fork/join stations increases. Especially, the number of subscripts on variables increases as the number of machines increases in a kanban system and tracking all the variables is very complex.

Figure 9 shows a generalized queueing network under CONWIP control with infinite supply and demand. With infinite supply and demand assumptions, the dynamics of machines are all the same. We just need to make sure that the completion of service at the last server, B_i, decreases its buffer population by one and increases population of the first buffer, B_1, by one. The NLP model of the generalized single loop CONWIP requires equations (2) – (15) and first and second moment constraints. The number of first moment

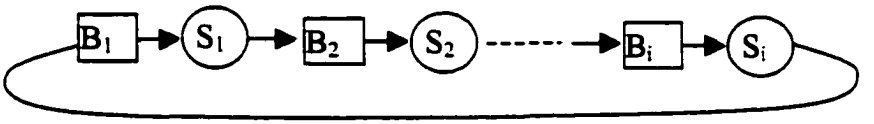

Figure 9. Generalized single loop CONWIP queueing network

Table 7. Dynamics of generalized single loop CONWIP queueing network

Event	Buffer Status	Buffer Change					
		1	2	3	...	i-1	i
Service completion at S_1	$X_1>0$	↓	↑				
Service completion at S_2	$X_2>0$		↓	↑			
Service completion at S_3	$X_3>0$			↓			
⋮							
Service completion at S_{i-1}	$X_{i-1}>0$					↓	↑
Service completion at S_i	$X_i>0$	↑					↓

constraints is equal to the number of servers, i, and the number of second moment constraints is equal to $i(i+1)/2$. Recalling that the number of stationary first and second moment constraints are $3i$ and $3i(3i+1)/2$, the CONWIP model requires fewer first and second moment constraints than a kanban system with the same number of servers.

Although, for comparison of performance of the systems with and without DPD in this study, three CONWIP loops are used to control kanbans in the systems, we first model a single CONWIP loop system. Modeling a single loop can help in understanding the performance of CONWIP systems. Also, we can check the accuracy of the performance bounds. To model more complex multiple CONWIP loop systems, we have to know how to model fork/join synchronization stations along with CONWIP systems. We already know how to model fork/join stations from the kanban model presented in previous chapter. We now discuss how to model CONWIP systems.

4.2 CONWIP Controlled System

The nonlinear programming model for a simple two serial machine CONWIP queueing network is presented in this section. Two models are developed: one with exponential service time and one with Erlang service time with 4 exponential phases. It is assumed that there are infinite supply and demand. Although the performance of a single CONWIP loop can be obtained analytically as a closed serial queueing network, it is helpful to see how well the models perform. Figure 10 shows the queueing network with exponential service times. There is only one loop for both machines.

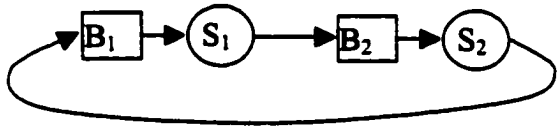

Figure 10. CONWIP queueing network with exponential service time

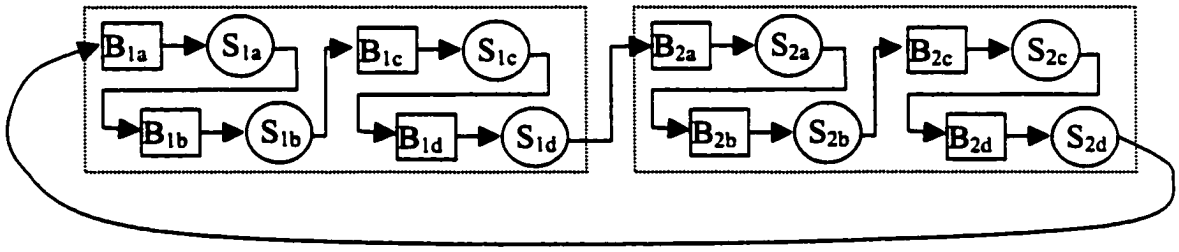

Figure 11. CONWIP queueing network with Erlang service time (k = 4)

Figure 11 shows the queueing network with Erlang service time. The four machines shown inside a dotted rectangle represent four phases of service although it is actually a single machine. For ease of illustration, we assume that the first buffers, B_{1a} and B_{2a} are real machining buffers and the remaining buffers are 'imaginary' buffers. The first buffers hold the parts waiting for the machine to be available and the parts being processed at the first phase. Only one part can be processed among four machines within a rectangle at any given time. This characteristic must be properly modeled in the NLP.

The NLP model for the exponential distribution is very precise because there are only two buffers. The dynamics of the buffers also are simple in that the population of a buffer increases as the other buffer finishes its processing and the other buffer's population is decreased by one (Table 8).

The first moment constraints for the two buffers are the same:

Table 8. Dynamics by service completion

Event	Buffer Status	Buffer Change	
		1	2
Service completion at S_1	$X_1>0$	↓	↑
Service completion at S_2	$X_2>0$	↑	↓

$$\mu_2 w_2 - \mu_1 w_1 = 0$$

The second moment constraints are also obtained as:

$$S(B_1,B_1) = \mu_2\gamma_{12} - \mu_1\beta_1 + \mu_2 w_2 = 0$$

$$S(B_2,B_2) = \mu_1\gamma_{21} - \mu_2\beta_2 + \mu_1 w_1 = 0$$

$$S(B_1,B_2) = \mu_2\beta_2 - \mu_2\gamma_{12} - \mu_2 w_2 - \mu_1\gamma_{21} + \mu_1\beta_1 - \mu_1 w_1 = - S(B_1,B_1) - S(B_2,B_2) = 0$$

In addition to the first moment and second moment constraints, equations (2) through (15) are used to find performance bounds of the queueing network. The model in LINGO is presented in Appendix 3. The model can be compared with simulation to see the accuracy of the NLP model.

The queueing network for Erlang service times is more complex in that each machine has four buffers and four stations representing four phases of the Erlang distribution. Table 9 shows the dynamics of buffer population changes. Service completion at the first servers, S_{1a} or S_{2a}, occurs only when all of the downstream buffers of the server, (S_{1b}, S_{1c}, S_{1d}) or (S_{2b}, S_{2c}, S_{2d}), respectively, are empty.

One of the properties of the Erlang model is that the buffer populations of imaginary buffers and their respective utilizations are the same since the imaginary buffers hold only one part being processed. Also, the imaginary buffer status represents the corresponding

Table 9. Dynamics of Erlang service time CONWIP

Event	Buffer Status	Buffer Change							
		1a	1b	1c	1d	2a	2b	2c	2d
SC at S_{1a}	$X_{1a}>0, X_{1b}=0, X_{1c}=0, X_{1d}=0$	↓	↑						
SC at S_{1b}	$X_{1b}>0$		↓	↑					
SC at S_{1c}	$X_{1c}>0$			↓	↑				
SC at S_{1d}	$X_{1d}>0$				↓	↑			
SC at S_{2a}	$X_{2a}>0, X_{2b}=0, X_{2c}=0, X_{2d}=0$					↓	↑		
SC at S_{2b}	$X_{2b}>0$						↓	↑	
SC at S_{2c}	$X_{2c}>0$							↓	↑
SC at S_{2d}	$X_{2d}>0$	↑							↓

(SC = Service Completion)

phase of a part's processing line and there can be at most one part in a set of imaginary buffers. This property can be expressed as

$$z_{1b1c} = z_{1b1d} = z_{1c1d} = 0$$
$$z_{2b2c} = z_{2b2d} = z_{2c2d} = 0 \tag{22}$$

The utilization of the real machine can expressed as

$$w_1 = w_{1a} + w_{1b} + w_{1c} + w_{1d} - z_{1a1b} - z_{1a1c} - z_{1a1d}$$
$$w_2 = w_{2a} + w_{2b} + w_{2c} + w_{2d} - z_{2a2b} - z_{2a2c} - z_{2a2d} \tag{23}$$

The number of constraints increases much more rapidly as the number of buffers (phases) increases in the Erlang service time model compared with the exponential service time model. Eight first moment constraints and 36 second moment constraints for the Erlang model are obtained while two first moment constraints and one second moment constraint for

the exponential model are obtained. Equations (2) – (15) and (23) also are required constraints for getting the performance bounds. A complete list of the constraints is shown in Appendix 4.

The two models presented in this section are compared. First, the two systems are compared for a balanced service rate such that the service rates, μ_i, are set at 1 for both machines. Table 10 shows the comparison of the two systems. The objective functions of both models are to maximize and minimize total throughput, $\mu_1 w_1 = \mu_2 w_2$. The exponential service time model provides the same values of upper bound and lower bound and these values are verified to be exact in the simulation test. On the other hand, the Erlang service time model only provides exact performance bounds when the number of kanbans in the system is 1. Then, the upper bounds reach 1 for larger numbers of kanbans in the system

Table 10. Comparison of balanced CONWIP queueing networks ($\mu_1 = \mu_2 = 1$)

K		Exponential Service Time				Erlang Service Time			
		β_1	β_2	w_1	w_2	β_1	β_2	w_1	w_2
1	**Simulation**	0.500	0.500	0.500	0.500	0.500	5.000	0.500	0.500
	UB	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500
	LB	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500
2	**Simulation**	1.000	1.000	0.667	0.667	1.002	0.998	0.786	0.784
	UB	1.333	0.667	0.667	0.667	1.000	1.000	1.000	1.000
	LB	1.333	0.667	0.667	0.667	1.000	1.000	0.500	0.500
5	**Simulation**	2.493	2.508	0.833	0.834	2.504	2.496	0.940	0.939
	UB	4.167	0.833	0.833	0.833	3.988	1.012	1.000	1.000
	LB	4.167	0.833	0.833	0.833	2.500	2.500	0.714	0.714
10	**Simulation**	4.997	5.002	0.909	0.909	4.947	5.054	0.972	0.973
	UB	9.090	0.909	0.909	0.909	7.809	2.191	1.000	1.000
	LB	9.090	0.909	0.909	0.909	5.001	4.999	0.833	0.833

while the lower bounds keep increasing the number of kanbans increases. One downfall of the model is that the NLP model can not provide accurate bounds for the expected buffer populations, β_1 and β_2.

The models are examined for unbalanced lines such that the service rate for the first machine is set to 1 and the rate of the second machine is set to 0.9. Table 11 shows the results that the exponential model provides better bounds than the Erlang model although the bounds for the exponential model are no longer exact when the number of kanbans is greater than two. Again, the upper bounds of the Erlang model quickly increases to the machines' capacities when the number of kanbans is greater than 1.

The reason why the Erlang NLP model gives less informative bounds lies in the complexity of the queueing network model. The Erlang model requires more buffers to represent it as queueing network. Many of constraints in the NLP model show relationships

Table 11. Comparison of unbalanced CONWIP queueing networks ($\mu_1 = 1, \mu_2 = 0.9$)

K		Exponential Service Time				Erlang Service Time			
		β_1	β_2	w_1	w_2	β_1	β_2	w_1	w_2
1	**Simulation**	0.473	0.527	0.473	0.527	0.474	0.526	0.474	0.526
	UB	0.474	0.526	0.474	0.526	0.474	0.526	0.474	0.526
	LB	0.474	0.526	0.474	0.526	0.474	0.526	0.474	0.526
2	**Simulation**	0.928	1.071	0.631	0.703	0.917	1.083	0.742	0.825
	UB	0.930	1.070	0.631	0.701	1.000	1.000	0.900	1.000
	LB	0.930	1.070	0.631	0.701	0.900	1.100	0.474	0.526
5	**Simulation**	2.194	2.806	0.788	0.875	1.887	3.113	0.879	0.976
	UB	3.326	1.673	0.805	0.895	1.900	3.100	0.900	1.000
	LB	1.340	3.660	0.772	0.858	2.050	2.950	0.672	0.747
10	**Simulation**	3.970	6.030	0.855	0.949	2.561	7.439	0.898	0.998
	UB	8.030	1.970	0.891	0.990	7.577	2.423	0.900	1.000
	LB	1.513	8.487	0.832	0.924	3.534	6.466	0.779	0.866

between two neighboring buffers. The exponential model has only two buffers and all constraints show relationships between these two while the Erlang model shows more relationship between phases and less between real machines.

CHAPTER 5 DELAYED PRODUCT DIFFERENTIATION

In this chapter, we will present two simple CONWIP controlled queueing networks, one with delayed product differentiation (DPD) and one without DPD. A simple production line is used to study the effect of delayed product differentiation on the relationship between throughput and inventory limit. It is assumed that there are two products and each product requires four operations. The first two operations are processed at two serial machines common for both products. After the common processes, the products are differentiated into two types. There is a separate line for each product, each with two serial machines. Implementation of DPD postpones the point of differentiation until after the third operation. After DPD, the two machines for the third operation can process both types of products. It is assumed that there are infinite supply and demand.

Three CONWIP loops are used to control inventory in the system. For a kanban policy, it is usually harder to allocate kanbans because it requires more variety of kanbans. Also, CONWIP systems tend to require less inventory than the kanban control policy. There are many ways to apply CONWIP control policies for analyzing systems. First, only one CONWIP loop can be used for all types of product. Kanbans are assigned to product types in proportion to the rate of demands of customers for each product. This type of policy, single chain multi-product, is known to be less efficient than having a separate loop for each product when their routes are different (see Duenyas 1994). The next alternative is to have same number of loops as the number of different products. Each loop would encompass both the common processes and the differentiated processes. Although this type of CONWIP policy requires the least inventory, the response time for demand takes longer. New kanbans

can not be added at the point of differentiation but only at the beginning of the production process. The CONWIP policy used in this research has one CONWIP loop for common processes and one CONWIP loop for each product's differentiated processes. A fork/join station at the point of differentiation synchronizes all CONWIP loops. In this way, kanbans can be added at the first buffer of each kanban loop and the configuration can quickly adapt when the demand of customers is changed. More alternative CONWIP policies can be obtained by dividing the CONWIP loops of previous alternatives into multiple CONWIP loops. This may not be desirable because dividing CONWIP loops may result in an increase of the number of kanbans in system.

As explained, the system for two products is controlled by three CONWIP loops. The first CONWIP loop is for the common processes and the other two loops are for each product after differentiation. A fork/join station is located at the differentiation point and it synchronizes those three CONWIP loops. Different from the fork/join station used in Chapter 3, the fork/join station now has one part buffer and two kanban buffers. The part buffer is for the parts having the common processes completed and the kanban buffers are for released kanbans after completion of the last differentiated process at the last station for each product. Comparing with the analytical model of the kanban queueing network, the analytical model of the CONWIP loop is much simpler since there is only one fork/join station. We showed that flow line CONWIP system can be easily generalized in Chapter 4. As for any CONWIP system, these models generalize easily to a large number of serial machines in any loop. The model without DPD can be generalized as a three loop CONWIP system with one fork/join station with any number of serial machines in any loop. Also, the model with DPD can be

generalized as three loop CONWIP system with one fork/join station and two parallel machines at the end of the common processes.

Figure 12 shows the queueing network representation before DPD implementation. S_1 and S_2 are the common process machines for both types of products. As soon as machine S_2 finishes service, the part goes to buffer B_3 for synchronization with either kanban buffers, B_4 or B_7. Buffer B_4 has priority over buffer B_7 so that, when both buffers have free kanbans waiting for a part to be available, a finished part attaches the kanban from B_4 and proceeds to B_5. S_3 and S_4 are the stations for the first type products and S_5 and S_6 are for the second type product.

Figure 13 shows the queueing network representation after DPD implementation. The

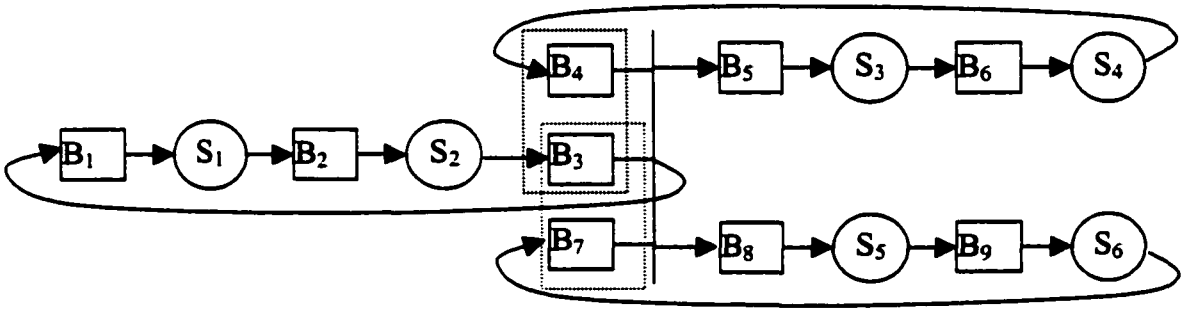

Figure 12. Queueing network without DPD

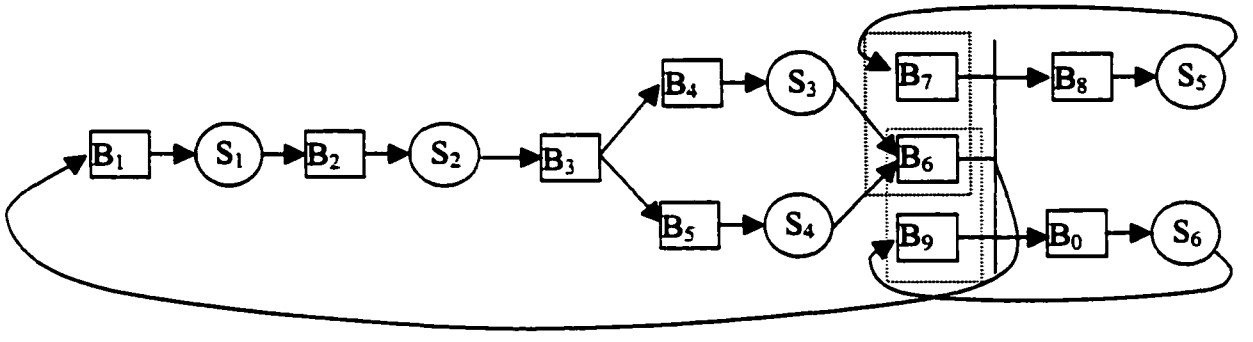

Figure 13. Queueing network with DPD

buffers and servers are named according to their locations such that lower CONWIP loop index and earlier processes have lower indexes. The servers, S_3, S_4, S_5, and S_6 in the system without DPD are now called S_3, S_5, S_4, and S_6. Also, the buffers, B_3, B_4, B_5, B_6, B_7, B_8, and B_9, in the system without DPD are now called B_6, B_7, B_4, B_8, B_9, B_5, and B_0.

There are now two more machines, S_3 and S_4, for common processes. The machining station buffers B_4 and B_5 hold at most one part each, namely, the part being processed at the machines so that all parts wait at buffer B_3 for either of the two machines to be available. It is assumed that B_4 has priority over B_5. When both machines, S_3 and S_4, are idle, the finished part from S_2 enters the B_4 to be processed by the machine S_3. For the best performance, we can set the faster machine as machine S_3 and the slower one as machine S_4. Finished parts from S_3 and S_4 are sent to the part buffer B_6 to be synchronized with kanban buffers B_7 and B_9. Again, the priority is given to the type 1 product so that B_7 has priority over B_9.

5.1 CONWIP without Delayed Product Differentiation (DPD)

The NLP model for the CONWIP system without DPD can be obtained by the same method used in the previous chapters. First, dynamics of the change in buffer populations of the system should be obtained (Table 12). For all machining station buffers except the last machining station buffer in a CONWIP loop, the dynamic is simply that the buffer population is decreased by one and the population of the next buffer is increased by one when a product completes service at the machining station. At the time of a service completion at the last machining station, S_2, of the common process loop, the buffer dynamics depend on the status of the fork/join station.

Table 12. Dynamics of CONWIP queueing network without DPD

Event Server	Buffer Status	Buffer Change								
		1	2	3	4	5	6	7	8	9
SC at S_1	$X_1>0$	↓	↑							
SC at S_2	$X_2>0, X_4>0$	↑	↓		↓	↑				
	$X_2>0, X_4=0, X_7>0$	↑	↓					↓	↑	
	$X_2>0, X_4=0, X_7=0$		↓	↑						
SC at S_3	$X_5>0$					↓	↑			
SC at S_4	$X_3>0, X_6>0$	↑		↓		↑	↓			
	$X_3=0, X_6>0$				↑		↓			
SC at S_5	$X_8>0$								↓	↑
SC at S_6	$X_3>0, X_9>0$	↑		↓					↑	↓
	$X_3=0, X_9>0$							↑		↓

(SC = Service completion)

In addition to the constraints (2) – (15) for the queueing network model without DPD, the stationary first and second moment constraints are obtained (see Appendix 5). The number of buffers in the system is nine, comprising the number of machines plus three fork/join station buffers. There are 9 first moment constraints and $9(9+1)/2 = 45$ second moment constraints.

The CONWIP model shown in this section can be generalized such that there can be any number of machines in each loop. Figure 14 shows a generalized CONWIP queueing network. There are *i* machines for the common processes, *j* machines for the type 1 product and *k* machines for the type 2 products. The buffers at the fork/join stations are named as follows: the part buffer is $B_{1,0}$, the kanban buffer for the type 1 products is $B_{2,0}$, and the kanban buffer for the type 2 products is $B_{3,0}$. The generalization is possible because the dynamics of the buffers in the middle of CONWIP loops are all the same. The buffer

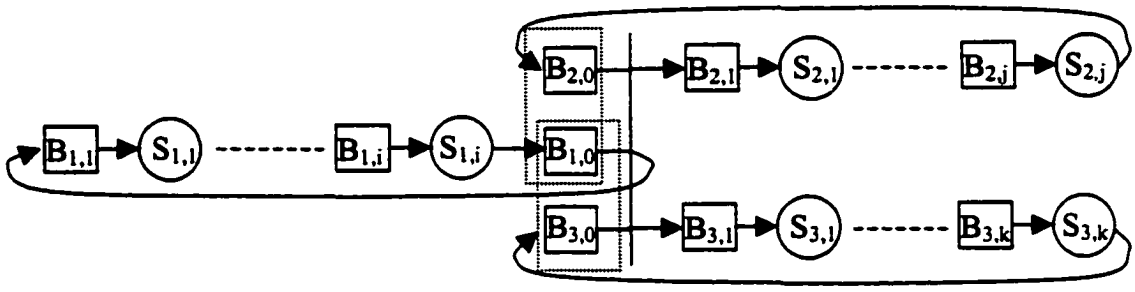

Figure 14. Generalized CONWIP queueing network without DPD

population of a machine in the middle decreases by one when the machine completes process and increases by one when its upstream server completes service.

Table 13 shows the dynamics of the generalized CONWIP queueing network. There are only four types of events: the first three types of events are completions of service at the last machines of CONWIP loops and the other one is completion of services at other machines. The status of the fork/join station needs to be checked for part/kanban

Table 13. Dynamics of generalized CONWIP queueing network without DPD

Event	Buffer Change		Buffer Status
	Increase	Decrease	
$S_{1,i}$	(1,1), (2,1)	(1,i), (2,0)	$X_{1,i}>0, X_{2,0}>0$
	(1,1), (3,1)	(1,i), (3,0)	$X_{1,i}>0, X_{2,0}=0, X_{3,0}>0$
	(1,0)	(1,i)	$X_{1,i}>0, X_{2,0}=0, X_{3,0}=0$
$S_{2,j}$	(1,1), (2,1)	(1,0), (2,j)	$X_{2,j}>0, X_{1,0}>0$
	(2,0)	(2,j)	$X_{2,j}>0, X_{1,0}=0$
$S_{3,k}$	(1,1), (3,1)	(1,0), (3,k)	$X_{3,k}>0, X_{1,0}>0$
	(3,0)	(3,k)	$X_{3,k}>0, X_{1,0}=0$
$S_{p,q}$	(p,q+1)	(p,q)	$X_{p,q}>0$

$\left((p,q) \notin \{(1,i), (2,j), (3,k)\}\right)$

synchronization when a process is completed at a last machine of a CONWIP loop. Completion of processing at other machines just sends the completed part to the next downstream buffer. Using the dynamics, first and second moment constraints can be obtained for the same type of CONWIP network. The number of first moment constraints is $i+j+k+3$. The number of second moment constraints is $\frac{(i+j+k+3)(i+j+k+4)}{2}$.

5.2 CONWIP with Delayed Product Differentiation (DPD)

Evaluating the performance of the CONWIP network with DPD (Figure 13) is more difficult than the CONWIP network without DPD (Figure 12) because of the existence of parallel machines. The model must reflect the properties of the parallel machines. A new part coming into the parallel machine subsystem remains in the outside buffer, B_3, when both machines are busy. The machines are not allowed to be idle unless there is no job waiting in B_3. When one of the parallel machines completes processing and starts to process another part received from B_3, the buffer population of B_3 changes, not the machining buffer population. Also, a part that finishes processing at S_2 will bypass B_3 and go to the higher priority machine if both machines are idle. Table 14 shows the dynamics of buffer changes for each event.

The NLP models are composed of equations (2) – (15), first moment constraints and second moment constraints. The NLP model in LINGO is presented in Appendix 6. There are total of 10 buffers: six machine buffers, three fork/join station buffers and one parallel machines' buffer.

Table 14. Dynamics of CONWIP queueing network with DPD

Event	Buffer Status	Buffer Change									
		1	2	3	4	5	6	7	8	9	0
SC at S_1	$X_1>0$	↓	↑								
SC at S_2	$X_2>0, X_4>0, X_5>0$		↓	↑							
	$X_2>0, X_4=0$		↓		↑						
	$X_2>0, X_4>0, X_5=0$		↓			↑					
SC at S_3	$X_3>0, X_4>0, X_7>0$	↑		↓				↓	↑		
	$X_3>0, X_4>0, X_7=0, X_9>0$	↑		↓						↓	↑
	$X_3>0, X_4>0, X_7=0, X_9=0$			↓			↑				
	$X_3=0, X_4>0, X_7>0$	↑			↓			↓	↑		
	$X_3=0, X_4>0, X_7=0, X_9>0$	↑			↓					↓	↑
	$X_3=0, X_4>0, X_7=0, X_9=0$				↓		↑				
SC at S_4	$X_3>0, X_5>0, X_7>0$	↑		↓				↓	↑		
	$X_3>0, X_5>0, X_7=0, X_9>0$	↑		↓						↓	↑
	$X_3>0, X_5>0, X_7=0, X_9=0$			↓			↑				
	$X_3=0, X_5>0, X_7>0$	↑				↓		↓	↑		
	$X_3=0, X_5>0, X_7=0, X_9>0$	↑				↓				↓	↑
	$X_3=0, X_5>0, X_7=0\ X_9=0$					↓	↑				
SC at S_5	$Y_6>0, X_8>0$	↑					↓				
	$Y_6=0, X_8>0$							↑	↓		
SC at S_6	$Y_6>0, X_9>0$	↑					↓				
	$Y_6=0, X_9>0$									↑	↓

The CONWIP queueing network also can be generalized as three CONWIP loops with two parallel machines at the end of the common process CONWIP loop. The number of machines in each loop can be any number except that there must be at least three machines in the common process loop. The first of the three machines is the last machine of common process loop of the CONWIP system without DPD and the other two machines are the parallel machines moved from the product specific loops. Figure 15 shows a generalized CONWIP queueing network having $i + 1$ common process machines, j type 1 product

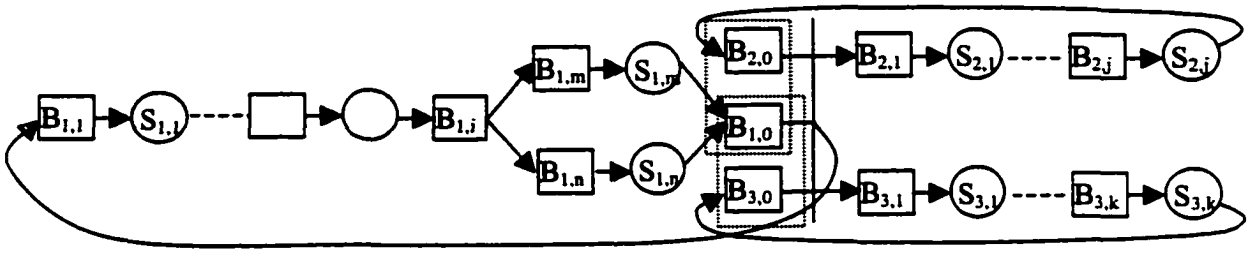

Figure 15. Generalized CONWIP queueing network with DPD

machines and k type 2 product machines. The index of the faster machine among the two parallel is given as (1,m) and slower machine's is (1,n). The fork/join station buffer indexes are the same and are denoted by $B_{1,0}$, $B_{2,0}$, and $B_{3,0}$. Note that the CONWIP queueing network with DPD should be drawn differently if more than one machine of each product specific loop become common process machines.

There are six different types of events. The dynamics of service completion at a machine in the middle of a loop, and the service completions at the end of each differentiated product loop remain the same as the dynamics explained in the previous section. The changes provoked by completion of service at machine $i-1$ depend on the status of buffers related to the parallel machines. The changes resulting from completion of services at the parallel machines, $S_{1,m}$ and $S_{1,n}$, depend on whether there is any part waiting for processing in the part buffer $B_{1,i}$. Table 15 shows the dynamics of the generalized CONWIP queueing network with DPD. The number of first moment constraints is $i+j+k+4$. The number of second moment constraints is $\frac{(i+j+k+4)(i+j+k+5)}{2}$.

Table 15. Dynamics of generalized CONWIP queueing network with DPD

Event	Buffer Change		Buffer Status
	Increase	Decrease	
$S_{1,i-1}$	(1,i)	(1,i-1)	$X_{1,i-1}>0, X_{1,m}>0, X_{1,n}>0$
	(1,m)	(1,i-1)	$X_{1,i-1}>0, X_{1,m}=0$
	(1,n)	(1,i-1)	$X_{1,i-1}>0, X_{1,m}>0, X_{1,n}=0$
$S_{1,m}$	(1,1), (2,1)	(1,m), (2,0)	$X_{1,i}=0, X_{1,m}>0, X_{2,0}>0$
	(1,1), (3,1)	(1,m), (3,0)	$X_{1,i}=0, X_{1,i}>0, X_{2,0}=0, X_{3,0}>0$
	(1,0)	(1,m)	$X_{1,i}=0, X_{1,i}>0, X_{2,0}=0, X_{3,0}=0$
	(1,1), (2,1)	(1,i), (2,0)	$X_{1,i}>0, X_{1,m}>0, X_{2,0}>0$
	(1,1), (3,1)	(1,i), (3,0)	$X_{1,i}>0, X_{1,i}>0, X_{2,0}=0, X_{3,0}>0$
	(1,0)	(1,i)	$X_{1,i}>0, X_{1,i}>0, X_{2,0}=0, X_{3,0}=0$
$S_{1,n}$	(1,1), (2,1)	(1,n), (2,0)	$X_{1,i}=0, X_{1,i}>0, X_{2,0}>0$
	(1,1), (3,1)	(1,n), (3,0)	$X_{1,i}=0, X_{1,n}>0, X_{2,0}=0, X_{3,0}>0$
	(1,0)	(1,n)	$X_{1,i}=0, X_{1,n}>0, X_{2,0}=0, X_{3,0}=0$
	(1,1), (2,1)	(1,i), (2,0)	$X_{1,i}>0, X_{1,n}>0, X_{2,0}>0$
	(1,1), (3,1)	(1,i), (3,0)	$X_{1,i}>0, X_{1,n}>0, X_{2,0}=0, X_{3,0}>0$
	(1,0)	(1,i)	$X_{1,i}>0, X_{1,n}>0, X_{2,0}=0, X_{3,0}=0$
$S_{2,j}$	(1,1), (2,1)	(1,0), (2,j)	$X_{2,j}>0, X_{1,0}>0$
	(2,0)	(2,j)	$X_{2,j}>0, X_{1,0}=0$
$S_{3,k}$	(1,1), (3,1)	(1,0), (3,k)	$X_{3,k}>0, X_{1,0}>0$
	(3,0)	(3,k)	$X_{3,k}>0, X_{1,0}=0$
$S_{p,q}$	(p,q+1)	(p,q)	$X_{p,q}>0$

5.3 Heuristic Algorithm for Kanban Configuration

The NLP models developed in the previous sections can be used to minimize the total number of kanbans in the system satisfying given throughput requirements. Although the NLP models identify a kanban configuration, i.e., the number of kanbans required for each loop, only the total number of kanbans is meaningful. The NLP solution finds the lower

bound for the total number of kanbans and not individual lower bounds of CONWIP loops. To optimally operate systems, one must know not only the total number of kanbans but also the allocation of kanbans.

A heuristic algorithm is presented in this section to find a kanban configuration satisfying given throughput requirements. Since the NLP model only provides bounds on throughput and the gap between them is wide, simulation is used to check for fulfillment of throughput requirement. Enumerating all possible kanban configurations is the only way to find a true minimum kanban configuration. The lower bounds of throughput tend to increase more slowly than the upper bounds. The minimum required number of kanbans quickly increases when the throughput requirements approach the machine capacities. The size of the solution space will also quickly increase. Total enumeration can be costly in this case. The heuristic algorithm presented in this section can reduce the effort. Although the NLP solution does not provide an exact solution, the solution of the problem to minimize the number of kanbans can be used as an initial starting point in the heuristic. The heuristic searches for a feasible kanban configuration starting from the kanban configuration given by the NLP model. A feasible kanban configuration is set as upper bound. Then, the heuristic searches for a feasible kanban configuration requiring fewer total kanbans.

The heuristic algorithm is shown in Figure 16. TH_p^{NLP} and TH_p denote throughput of the CONWIP loop L_p obtained by the NLP and simulation. R_p denotes throughput requirement for the CONWIP loop L_p. The initial step, Step 0, of the heuristic solves the NLP problem to minimize the total number of kanbans subject to constraint(s) on the throughput and sets the resulting kanban configuration as a starting point for the heuristic. Then, Step 1 uses simulation to check if the kanban configuration satisfies the throughput

Step 0: Solve NLP problem

Min. $\sum K_p$

s.t. $TH_p^{NLP} \geq R_p$, for $\forall\, p$

Set $SumK = \sum K_p$, $K_p^{LB} = K_p$, $S = 0$

Step 1: Run simulation with the current kanban configuration

If $TH_p \leq R_p$ for any p, then

For all p where $p \neq 1$ and $TH_p \leq R_p$, set $K_p = K_p + 1$,

$K_1 = K_1 + 1$

Set $SumK = \sum K_p$ and update $K_p^* = K_p$ for all p

Repeat **Step 1**

Step 2 : Set $K_1 = SumK - \sum_{i \neq 1} K_i^{LB} - 1$ and $K_p = K_p^{LB}$

Step 3: Run simulation with current kanban configuration

If there exists $TH_p \leq R_p$ for any p, then

If $K_1 = K_1^{LB}$ goto **Step 4**

If $K_1 < SumK - \sum_{i \neq 1} K_i^{LB} - 1$ and $S \leq R_p - TH_p$ for any $p \neq 1$, goto **Step 4**

Else, $K_1 = K_1 - 1$ and $K_i = K_i + 1$ where $R_i - TH_i \geq R_p - TH_p$, for all $p \neq 1$,

Set $S = R_i - TH_i$ and repeat **Step 3**

Else, set $SumK = \sum K_p$, update $K_p^* = K_p$ for all p, and goto **Step 2**

Step 4: Minimum kanban configuration: $K_p^* = K_p$ for all p. Stop.

Figure 16. Heuristic algorithm for kanban configuration

requirement. If any of the throughput requirements are not met, more kanbans are added to the offending CONWIP loops. The procedures of running the simulation and adding kanbans are repeated until all throughput requirements are met. Next, the heuristic tests if there is any feasible solution with one less kanban by setting the number of kanbans of each product specific CONWIP loop to its lower bound from the NLP solution. The remaining kanbans are assigned to the common process CONWIP loop. One kanban is moved to the product specific CONWIP loop with the greatest violation of its throughput requirement. The lower bounds are updated when a feasible kanban configuration is found. The heuristic algorithm ends if the number of kanbans of the common process loop reaches its lower bound or reallocation of kanbans does not improve throughput in the CONWIP loop of the newly added kanban. The last feasible solution found by the heuristic is taken as the minimum kanban configuration.

Table 16 shows an example of how the heuristic finds a kanban configuration for the CONWIP system without DPD shown in Figure 12. The service rates are $\mu_1 = 1$, $\mu_2 = 0.9$, $\mu_3 = 0.6$, $\mu_4 = 0.5$, $\mu_5 = 0.4$, and $\mu_6 = 0.5$. The throughput requirements for the system, type 1 product and type 2 product are equivalent to 90% of maximum capacities of each loop and are given by $R_1 = 0.81$, $R_2 = 0.45$ and $R_3 = 0.36$. The NLP solution found the minimum kanban configuration of $(K_1^{LB}, K_2^{LB}, K_3^{LB}) = (5, 4, 4)$. In Step 1, the algorithm repeatedly increases number of kanbans of CONWIP loops violating throughput requirements and examines the new kanban configuration until there is no violation. It proceeds to Step 2, when it finds a feasible kanban configuration, $(K_1^*, K_2^*, K_3^*) = (11, 6, 10)$ and assigns a minimum number of kanbans to the product specific CONWIP loops, K_2 and K_3, and the

maximum number of kanbans to the common process CONWIP loop, K_1, with one less than *SumK*. In Step 3, the heuristic tries to find a feasible kanban configuration by taking one kanban from the common process loop and reassigning it to the CONWIP loop with greater deficit. Because it found better kanban configuration $(K_1^*, K_2^*, K_3^*) = (10, 6, 10)$, it reduces the total number of kanbans by one and starts a new search. The algorithm stops when the number of kanbans of the common process loop reaches its lower bound without finding a

Table 16. Example heuristic algorithm procedure

	K_1	K_2	K_3	TH_1	TH_2	TH_3	*SumK*
Step 0	5	4	4	0.810	0.450	0.360	13
Step 1	5	4	4	0.715	0.415	0.300	13
Step 1	6	5	5	0.749	0.436	0.313	16
Step 1	7	6	6	0.773	0.451	0.322	19
Step 1	8	6	7	0.788	0.452	0.336	21
Step 1	9	6	8	0.799	0.452	0.347	23
Step 1	10	6	9	0.809	0.453	0.356	25
Step 1	**11**	**6**	**10**	**0.816**	**0.453**	**0.363**	**27**
Step 2,3	18	4	4	0.778	0.431	0.347	26
Step 3	17	5	4	0.791	0.447	0.343	26
Step 3	16	5	5	0.800	0.446	0.353	26
Step 3	15	5	6	0.805	0.445	0.359	26
Step 3	14	6	6	0.810	0.457	0.353	26
Step 3	13	6	7	0.813	0.455	0.357	26
Step 3	12	6	8	0.814	0.455	0.359	26
Step 3	11	6	9	0.813	0.454	0.359	26
Step 2,3	**10**	**6**	**10**	**0.812**	**0.452**	**0.360**	**26**
Step 2,3	9	6	10	0.810	0.453	0.357	25
Step 3,4	8	6	11	0.807	0.452	0.355	25

new feasible configuration or when moving a kanban from the common process does not help to reduce the deficit of the throughput requirement. The algorithm of this example stops for the latter reason. The best kanban configuration found by the algorithm is (K_1^*, K_2^*, K_3^*) = (10, 6, 10).

The heuristic algorithm starts from the solution to the NLP model designed for exponential service time. It would not perform well if the algorithm is applied for systems having different service time distributions. The kanban configuration obtained may already be feasible and it would just consider it as the best configuration. A minor modification can make the algorithm applicable to the systems with other service time distributions. The algorithm still uses the kanban configuration from exponential service time NLP models as its starting point. However, the lower bounds for all CONWIP loops are set to 1. The algorithm must search a wider solution space because of the lack of good lower bounds and a less reliable starting point. Figure 17 shows the modification made for the heuristic algorithm for Erlang service time. The rest of steps remain the same as in the original heuristic algorithm.

It is desired to have better NLP models for the exponential service time and to

Step 0: Solve NLP problem

Min. $\sum K_p$

s.t. $TH_p \geq R_p$, for $\forall\, p$

Set $SumK = \sum K_p$, $K_p^{LB} = 1$, $S = 0$

Figure 17. Modification of the heuristic algorithm for the Erlang service time

develop NLP models for other service time distributions to minimize the effort. The existing algorithm can be used for finding minimum kanban configuration for systems having more different product types.

CHAPTER 6 NUMERICAL RESULTS: CONWIP WITH AND WITHOUT DPD

In the previous chapter, we showed how nonlinear programming models are developed for the queueing networks with CONWIP policies, one with delayed product differentiation and one without delayed product differentiation. NLP models can also be obtained for generalized version of the two situations with arbitrary numbers of production stages. First and second moment constraints can be obtained using the generalized dynamics. Other constraints can be obtained by applying equations (2) – (15).

Using the NLP models of the two example queueing networks, upper and lower throughput bounds are obtained and compared with simulation results to see how useful the bounds are. Also, the models are examined to see if implementation of delayed product differentiation can improve system throughput. The results of the heuristic algorithm to find an efficient kanban configuration are compared for the two systems as well.

6.1 Performance Bounds

Performance bounds obtained from NLP models are compared with simulation results. An extended simulation study is performed for systems having Erlang service time distributions with varying number of phases while maintaining same service rates. These results help to see how the performance is changed as the variance of service time decreases. Also, it gives us some sense of practicality of using the models to evaluate non-exponential service time distribution. The results are presented for two cases, a balanced case in which

processing rates for the two products are equal, and an unbalanced case in which they differ.

CASE 1:

First, all the machines are balanced such that the two common process machines have service rates of 1 and all other machines have service rates of 0.5. In other words, the maximum capacity of the system is 1 and maximum capacities of both products are 0.5. The simulation results and NLP results are shown in Table 17 for the system without DPD. The models are examined for three different kanban configurations. The first kanban configuration, (K_1,K_2,K_3)= (3,2,2), was obtained by maximizing total throughput in the NLP models with seven kanbans in the system. TH_p denotes the throughput of CONWIP loop L_p. TH_2 ($= \mu_3 w_5 = \mu_4 w_6$) is the throughput of the type 1 product, TH_3 ($= \mu_5 w_8 = \mu_6 w_9$) is the throughput of the type 2 product, and TH_1 ($= \mu_1 w_1 = \mu_2 w_2$) is the throughput of the system which is equal to $TH_2 + TH_3$. The second kanban configuration, (K_1,K_2,K_3)= (6,9,4), was

Table 17. Performance bounds for balanced CONWIP system without DPD

Method	(K_1,K_2,K_3)								
	(3,2,2)			(6,9,4)			(9,9,9)		
	TH_1	TH_2	TH_3	TH_1	TH_2	TH_3	TH_1	TH_2	TH_3
Simulation (Exp)	0.603	0.312	0.291	0.782	0.447	0.335	0.851	0.447	0.405
Simulation (Erl. K=2)	0.683	0.349	0.333	0.859	0.474	0.387	0.913	0.470	0.443
Simulation (Erl. K=3)	0.727	0.371	0.357	0.894	0.480	0.414	0.937	0.479	0.459
Simulation (Erl. K=4)	0.757	0.384	0.373	0.914	0.485	0.429	0.951	0.484	0.467
Max TH_1	0.667	0.334	0.334	0.850	0.450	0.400	0.900	0.450	0.450
Min TH_1	0.387	0.194	0.194	0.598	0.347	0.251	0.692	0.346	0.346
Max TH_2	0.667	0.334	0.334	0.715	0.450	0.265	0.819	0.450	0.369
Max TH_3	0.564	0.231	0.334	0.831	0.431	0.400	0.889	0.439	0.450

obtained with throughput constraints, $TH_2 \geq 0.45$ and $TH_3 \geq 0.4$, from NLP. The third kanban configuration, $(K_1,K_2,K_3)= (9,9,9)$, obtained with throughput constraints, $TH_2 \geq 0.45$ and $TH_3 \geq 0.45$, from NLP. Four different objective functions, namely, maximize total throughput, minimize total throughput, maximize type 1 product throughput and maximize type 2 throughput, are used to obtain throughput bounds. Although the simulation result shows that the throughputs from simulation are within the bounds, the bounds have wide gaps. For all kanban configurations examined, the simulation results are closest to the upper bounds obtained with the objective functions, Max TH_1 and Max TH_2. In fact, the simulation results with Erlang service time (k = 2 phases) and the bounds of Max TH_1 are closest among all bounds. This suggests that the heuristic algorithm can use the kanban configuration from the NLP with this objectives as a good starting point for the Erlang service time.

The CONWIP system with DPD was also examined with the same service rates and throughput constraints. The kanban configurations are obtained by the same objectives and constraints as for the system without DPD. The results show that the gaps between bounds are wider than those of the CONWIP system without DPD (Table 18). The NLP solution suggests to move all kanbans except one kanban in each product specific loop to the common process loop. This suggestion makes sense because there is only one machine in each product loop. The simulation result with 4-phase Erlang service time distribution and the bounds of Max TH_1 are closest among all bounds.

As stated before, the first kanban configuration was obtained by maximizing system throughput with seven kanbans in the system. The other two kanban configurations were obtained by minimizing the total number of kanbans in the system subject to throughput requirements. For the first kanban configuration, both the simulation result and the NLP

Table 18. Performance bounds for balanced CONWIP system with DPD

Method	(K_1,K_2,K_3)								
	(5,1,1)			(6,1,1)			(9,1,1)		
	TH_1	TH_2	TH_3	TH_1	TH_2	TH_3	TH_1	TH_2	TH_3
Simulation (Exp)	0.657	0.358	0.299	0.698	0.375	0.322	0.772	0.406	0.366
Simulation (Erl. K=2)	0.733	0.391	0.341	0.773	0.408	0.366	0.844	0.436	0.408
Simulation (Erl. K=3)	0.774	0.409	0.365	0.814	0.424	0.390	0.880	0.451	0.429
Simulation (Erl. K=4)	0.801	0.418	0.382	0.841	0.434	0.407	0.901	0.459	0.442
Max TH_1	0.833	0.431	0.403	0.857	0.423	0.434	0.900	0.455	0.445
Min TH_1	0.333	0.167	0.167	0.351	0.176	0.176	0.387	0.193	0.193
Max TH_2	0.825	0.457	0.369	0.852	0.463	0.389	0.898	0.475	0.424
Max TH_3	0.833	0.396	0.438	0.857	0.411	0.446	0.900	0.437	0.463

result agree that systems with DPD provide improvement of throughput. Comparingthe simulation results for the other two kanban configurations, we can see which NLP model provides better kanban configuration for given throughput requirement. The simulation results shows that the NLP model for the system without DPD provides a better kanban configuration in that the throughput obtained by simulation is closer to the throughput requirement.

CASE 2:

Now, the service rates are given such that the type 1 product line has a higher capacity than the type 2 product line and there are some bottleneck machines. The service rates of the system without DPD are $\mu_1 = 1$, $\mu_2 = 0.9$, $\mu_3 = 0.6$, $\mu_4 = 0.5$, $\mu_5 = 0.4$, $\mu_6 = 0.5$. The maximum throughput that the system can obtain is 0.9. The maximum throughput for the type 1 product is 0.5 and the maximum throughput for the type 2 product is 0.4. The second

kanban configuration, (K_1,K_2,K_3)= (4,5,2), was obtained with throughput constraints, $TH_2 \geq$ 0.48 and $TH_3 \geq 0.27$. The third kanban configuration, (K_1,K_2,K_3)= (5,4,4), was obtained with throughput constraints, $TH_2 \geq 0.45$ and $TH_3 \geq 0.36$. Table 19 shows the results. The bounds still have a wide gap. Again, the upper bounds with the objective functions, Max TH_1 and Max TH_2, are provide the best bounds. Also, the simulation results with Erlang service time (k = 2 phases) and the bounds of Max TH_1 for the first set of throughput requirements and Erlang service time (k = 3 phases) and the bounds of Max TH_1 for the other two sets of throughput requirements are closest among all bounds.

The same service rates and throughput constraints are given for the CONWIP systems with DPD for evaluation. This example shows an advantage of implementing DPD. Although the maximum throughput of the systems remains the same, the bottleneck machine of type 2 product machine is one of the two parallel machines in the system and the maximum

Table 19. Performance bounds for unbalanced CONWIP system without DPD

Method	(K_1,K_2,K_3)								
	(3,2,2)			(4,5,2)			(5,4,4)		
	TH_1	TH_2	TH_3	TH_1	TH_2	TH_3	TH_1	TH_2	TH_3
Simulation (Exp)	0.582	0.332	0.253	0.659	0.430	0.229	0.715	0.414	0.299
Simulation (Erl. K=2)	0.664	0.375	0.289	0.742	0.474	0.267	0.792	0.457	0.335
Simulation (Erl. K=3)	0.706	0.396	0.309	0.779	0.488	0.291	0.828	0.476	0.352
Simulation (Erl. K=4)	0.734	0.411	0.323	0.801	0.493	0.307	0.847	0.484	0.362
Max TH_1	0.678	0.375	0.303	0.783	0.484	0.299	0.818	0.448	0.370
Min TH_1	0.371	0.205	0.166	0.460	0.307	0.154	0.507	0.280	0.227
Max TH_2	0.594	0.375	0.219	0.674	0.484	0.190	0.720	0.461	0.258
Max TH_3	0.612	0.310	0.303	0.738	0.435	0.303	0.673	0.302	0.370

throughput of type 2 is now 0.5, the same as for the type 1 product. Comparing results of systems with the first kanban configuration, the system with DPD performs better than the system without DPD when there is the same number of kanbans (Table 20).The result shows that the gap is still quite wide and the simulation result with Erlang service time distribution (k = 4) and the bounds of Max TH_1 are closest among all bounds. Again, the NLP model of the CONWIP system without DPD provides a better bounds than the CONWIP system with DPD.

CASE 3:

The models are examined to see the effect of priority. Service rates are given that the service rates for common processes are equal to 1. Also, the service rates for the first type of product are equal to 0.7 and the service rates for the second type of product are equal to 0.4.

Table 20. Performance bounds for unbalanced CONWIP system with DPD

Method	(K_1,K_2,K_3)								
	(5,1,1)			(4,2,1)			(8,1,1)		
	TH_1	TH_2	TH_3	TH_1	TH_2	TH_3	TH_1	TH_2	TH_3
Simulation (Exp)	0.641	0.395	0.288	0.600	0.387	0.213	0.729	0.389	0.340
Simulation (Erl. K=2)	0.711	0.384	0.327	0.663	0.430	0.234	0.799	0.419	0.380
Simulation (Erl. K=3)	0.751	0.400	0.350	0.698	0.449	0.249	0.833	0.433	0.400
Simulation (Erl. K=4)	0.776	0.410	0.366	0.721	0.462	0.258	0.854	0.441	0.413
Max TH_1	0.777	0.409	0.368	0.750	0.467	0.283	0.819	0.427	0.392
Min TH_1	0.332	0.166	0.166	0.325	0.195	0.130	0.375	0.188	0.187
Max TH_2	0.776	0.441	0.335	0.703	0.500	0.203	0.819	0.452	0.366
Max TH_3	0.776	0.360	0.416	0.750	0.354	0.396	0.818	0.386	0.432

In the comparison, priority is first given to the higher capacity, 0.7, product so that the higher capacity product becomes type 1 product. Then, in a second sub-case, priority is give to the lower capacity, 0.4, product so that the lower capacity product becomes type 1 product.

The models are examined with the kanban configuration found by the NLP to maximize system throughput with a given total number of kanbans in the system. The service rates are 1 for the common process machines, 0.7 for the type 1 product machines and 0.4 for the type 2 product machines. The NLP suggests the kanban configuration of (K_1,K_2,K_3) = (4,3,3) for the system without DPD and (K_1,K_2,K_3) = (8,1,1) for the systems with DPD (Table 21). Then, the service rates are interchanged between products so that the service rate of type 1 product machines are equal to 0.4 and those for the type 2 product machines are 0.7. Although the gaps are still wide, the NLP solution provides good information as to how the performance of systems would change (Table 22). First, the NLP solution suggests to change

Table 21.Performance bounds for high capacity/high priority CONWIP system

Method	W/O DPD			W/ DPD		
	(K_1,K_2,K_3) = (4,3,3)			(K_1,K_2,K_3) = (8,1,1)		
	TH_1	TH_2	TH_3	TH_1	TH_2	TH_3
Simulation (Exp)	0.730	0.480	0.250	0.785	0.509	0.276
Simulation (Erl. K=2)	0.819	0.541	0.278	0.860	0.555	0.305
Simulation (Erl. K=3)	0.865	0.573	0.292	0.899	0.577	0.321
Simulation (Erl. K=4)	0.894	0.593	0.301	0.921	0.590	0.331
Max TH_1	0.800	0.520	0.270	0.889	0.567	0.322
Min TH_1	0.485	0.309	0.176	0.408	0.259	0.149
Max TH_2	0.766	0.525	0.241	0.885	0.610	0.275
Max TH_3	0.684	0.384	0.300	0.885	0.532	0.353

Table 22. Performance bounds for high capacity/low priority CONWIP system

Method	W/O DPD						W/DPD		
	$(K_1,K_2,K_3) = (4,2,4)$			$(K_1,K_2,K_3) = (4,3,3)$			$(K_1,K_2,K_3) = (8,1,1)$		
	TH_1	TH_2	TH_3	TH_1	TH_2	TH_3	TH_1	TH_2	TH_3
Simulation (Exp)	0.731	0.250	0.480	0.723	0.290	0.433	0.784	0.324	0.459
Simulation (Erl. K=2)	0.817	0.277	0.540	0.812	0.322	0.490	0.859	0.345	0.513
Simulation (Erl. K=3)	0.862	0.291	0.571	0.859	0.337	0.522	0.897	0.355	0.542
Simulation (Erl. K=4)	0.891	0.301	0.590	0.889	0.347	0.541	0.920	0.361	0.559
Max TH_1	0.800	0.240	0.560	0.800	0.300	0.500	0.889	0.336	0.552
Min TH_1	0.492	0.137	0.355	0.485	0.176	0.309	0.408	0.149	0.258
Max TH_2	0.730	0.266	0.463	0.660	0.300	0.360	0.885	0.370	0.514
Max TH_3	0.800	0.240	0.560	0.780	0.255	0.638	0.885	0.304	0.581

the kanban configuration for the system without DPD to $(K_1,K_2,K_3) = (4,2,4)$. Although the upper bounds remain the same, the lower bounds of the new configuration are higher. The simulation result also confirms that the new configuration generates higher throughput. Also, for the system with DPD, the NLP solution suggests that the throughput of type 1 products increases and the throughput of type 2 products decreases. Although the results are not presented, when the system is examined with other total numbers of kanbans in the system, 15 and 20, the results are consistent. Some of the kanban configurations of the NLP solution are changed when the service rates of two lines are interchanged. The system throughput of the new kanban configuration is higher than the throughput with old kanban configuration under the new service rates.

6.2 Comparison of CONWIP Systems using Heuristic Algorithm

The CONWIP systems are compared on the basis of the results of the heuristic algorithm. There are three characteristics that we would like to see in the study. First, the algorithm must be able to find solutions within a reasonable amount of time and effort for applicability in real word. The number of simulation runs is used to determine the efficiency of simulation because running simulations takes the most time in the heuristic. Second, implementing delayed product differentiation should be beneficial in that the total number of kanbans required to obtain a required throughput is reduced. Third, the heuristic algorithm should be usable in systems with other service time distributions, specifically Erlang service time distributions.

The service rates are the same as in the unbalanced CONWIP system example, CASE 2, presented in previous section. Two sets of throughput requirements, ($TH_2 \geq 0.45$, $TH_3 \geq 0.36$) and ($TH_2 \geq 0.475$, $TH_3 \geq 0.38$), are examined. The throughput requirements are equivalent to 90% and 95%, respectively of the maximum capacities of each product of the CONWIP systems. The results show that the system with DPD requires a smaller total number of kanbans (Table 23). The system requires 7 and 15 fewer kanbans, respectively, for the given throughput requirement. These are equal to 27% and 37% less, respectively, than the number of kanbans required for the system without DPD. The kanban configurations are obtained by performing from 12 to 24 simulation runs. It seems that the heuristic can find good kanban configurations with reasonable effort.

Cycle times of CONWIP loops are also shown in Table 23. The cycle time of the common process loop is increased after delaying product differentiation. On the other hand, the cycle times of the type 1 and type 2 product loops are decreased. This is expected because

Table 23. Heuristic algorithm result for systems with Exponential service time

Throughput Requirement	$TH_2 \geq 0.45, TH_3 \geq 0.36$		$TH_2 \geq 0.475, TH_3 \geq 0.38$	
	W/O DPD	W/ DPD	W/O DPD	W/ DPD
Initial solution (K_1, K_2, K_3)	(5,4,4)	(8,1,1)	(9,5,5)	(16,2,1)
Heuristic solution (K_1^*, K_2^*, K_3^*)	(10,6,10)	(16,2,1)	(20,8,13)	(22,3,1)
Total number of kanbans	26	19	41	26
TH	0.812	0.824	0.855	0.857
TH_2	0.452	0.451	0.475	0.476
TH_3	0.360	0.373	0.380	0.381
Cycle time (common process loop)	12.32	19.42	23.39	25.67
Cycle time (type 1 product loop)	13.27	4.44	16.84	6.30
Cycle time (type 2 product loop)	27.78	2.68	34.21	2.62
Number of simulation run	17	19	24	12

the number of machines in the common process loop is increased and the numbers of machines are decreased in the other two loops. Comparing the total cycle time of each product, i.e., the sum of the cycle time of the common process loop and the cycle time of a type specific loop, the total cycle time for the type 1 product is decreased. The total cycle time for the type 2 product is greatly decreased. The results also show that the cycle times of the product specific loops are greatly decreased.

The heuristic algorithm is examined for the CONWIP systems with Erlang service time (k = 4). The service rates and the throughput requirements remain the same as in CASE 2. The results show that the system with DPD requires fewer kanbans than the system without DPD (Table 24). The total numbers of kanbans required for DPD system have 1 and 5 fewer kanbans, respectively, equivalent to 8% and 31% reduction. Comparing with the results in Table 23, the systems having Erlang service time require from 42% to 60%,

Table 24. Heuristic algorithm result for systems with Erlang service time (k = 4)

Throughput Requirement	$TH_2 \geq 0.45$, $TH_3 \geq 0.36$		$TH_2 \geq 0.475$, $TH_3 \geq 0.38$	
	W/O DPD	W/ DPD	W/O DPD	W/ DPD
Initial solution (K_1, K_2, K_3)	(5,4,4)	(8,1,1)	(9,5,5)	(16,2,1)
Heuristic solution (K_1^*, K_2^*, K_3^*)	(6,3,3)	(8,2,1)	(7,4,5)	(9,2,1)
Total number of kanbans	12	11	16	12
TH	0.836	0.847	0.867	0.858
TH_2	0.469	0.481	0.485	0.483
TH_3	0.367	0.376	0.381	0.385
Cycle time (common process loop)	7.18	9.32	8.07	10.36
Cycle time (type 1 product loop)	6.40	4.16	8.25	4.14
Cycle time (type 2 product loop)	8.17	2.66	13.12	2.60
Number of simulation run	8	6	12	12

respectively, fewer kanbans than the one having exponential service time. In the examples in previous section, when the objective was to maximize the total throughput, the NLP solution allocated only one kanban in each product specific loop. However, to satisfy a given throughput requirement, the product specific loops may require more than one kanban. The throughput requirement for the type 1 product is greater than type 2 product. Although the NLP solution for the first throughput requirement ($TH_2 \geq 0.45$, $TH_3 \geq 0.36$) assigns only one kanban to the type 1 product CONWIP loop, the solution for the second throughput requirement ($TH_2 \geq 0.475$, $TH_3 \geq 0.38$) suggests two kanbans. The heuristic results shows that two kanbans are required for the type 1 CONWIP loop with DPD for both the throughput requirements. Comparing the cycle times for the example of Erlang service rate, the results also show similar changes after DPD as with the example of exponential service rate. The total cycle times for both the type 1 product loop and the type 2 product loop are decreased.

Also, the cycle time for the common process loop increases and cycle times for the product specific loops greatly decrease.

The reduction in cycle time of product specific loops can be a great benefit of DPD when there is a change in demand rates or product mix. Especially, adding new kanbans for the product specific loops will take effect much faster in the system with DPD.

CHAPTER 7 FUTURE RESEARCH AND CONCLUSION

We began by developing nonlinear programming models for queueing network models of simple pull systems. A system of multiple CONWIP loops was selected for the analysis of delayed product differentiation. All results indicate that delaying product differentiation reduces the number of kanbans required to meet throughput targets and that the system performance improves by implementation of DPD. There still are some issues that need additional attention.

The infeasibility of NLP model of the single-stage kanban system with random supply and demand is noteworthy in that it means this kanban system cannot reach steady state under certain conditions. It would not reach steady state as long as the distributions of supply and demand are independent and the rates are the same. The following argument shows why the infeasibility occurs. Assume the times between arrivals of demand and arrivals of supplies are exponentially distributed and let $N_1(t)$ = number of supply arrivals up to time t ~ Poisson(λt) and $N_2(t)$ = number of demand arrivals up to time t ~ Poisson(λt), $\mathrm{E}[N_1(t) - N_2(t)] = 0$ and $\mathrm{Var}[N_1(t) - N_2(t)] = \lambda t$. In other words, the expected numbers of supply and demand arrivals are the same but the variation keeps increasing as time goes on. Let $N_C(t)$ = the number of finished parts having left the system up to time t. It must be that $N_C(t) \le N_1(t)$ and $N_C(t) \le N_2(t)$. Assuming that all the kanbans are at buffer B_2 in the beginning, the supply buffer population, X_1 and demand buffer population can be written as

$$X_1(t) = \max\left(0,\ N_1(t) - N_C(t) - X_3(t) - X_4(t)\right)$$

and

$$X_5(t) = \max\left(0,\ N_2(t) - N_C(t) - X_2(t) - X_3(t) - X_3(t) + K\right) = \max(0,\ N_2(t) - N_C(t)).$$

Since $N_C(t) \le N_1(t)$ and $N_C(t) \le N_2(t)$, the supply and demand buffer populations must satisfy

$$X_1(t) \ge N_1(t) - N_2(t) - X_3(t) - X_4(t)$$

and

$$X_5(t) \ge N_2(t) - N_1(t).$$

Because the variance $\mathrm{Var}\left[N_1(t) - N_2(t)\right]$ is increasing as t increases, at least one of the supply buffer population $X_1(t)$ and demand buffer population $X_5(t)$ also increases. Therefore, there can not be steady state values for the supply and demand buffer populations. For other distributions, there also can not be steady state if the variance increases as time increases. This must also be true in CONWIP systems. In a pull type system, the system may not steadily work if supply and demand do not arrives with some degree of simultaneity. The kanban system can be compared with set of gears in a machine. A machine will run smoothly only if all gears move together. The infeasibility of the modularized kanban system suggests that the whole system must be analyzed together. This makes NLP modeling very hard because the model size rapidly increases as the number of fork/join station increases. Furthermore, it suggests that a kanban system is not appropriate if the supply and demand rate are the same and these processes are independent.

A nonlinear model for three serial machine kanban system was designed. Because of the infeasibility of modularized kanban loop, all three kanban loops are modeled together in a single NLP model. Assuming infinite supply and demand, the model now provides

throughput bounds of the system in the steady state. It confirms that all kanban loops in a system need to be analyzed together.

The two serial machine CONWIP system models with exponential service time and Erlang service time were developed for the framework for analyzing systems with and without delayed product differentiation in which the inventory of the systems is controlled by multiple CONWIP loops. Although it was expected that better performance bounds could be obtained with Erlang service time, the upper bounds of the model quickly increased to the maximum capacity of the system. On the other hand, when the system is balanced, the model provides exact performance bounds for the exponential service time. Also, when the system is not balanced, the bounds show the performance change when the number of kanbans is changed. The lower bounds of the Erlang model provide information about the performance change. Because the upper bounds quickly approach the maximum capacity of the system, the kanban configuration obtained by Erlang model requires a very small number of kanbans in the system. It requires a maximum of two kanbans for any throughput requirement in the two serial machine example with Erlang service time. The Erlang service time NLP model must be improved so that the upper bounds are more accurate.

A system with multiple CONWIP loops was selected for the inventory control policy. Not only it is easier to model but also the CONWIP policy is easy to control in real production systems because there are fewer types of kanbans. The models can be generalized so that systems with more machines can be easily analyzed. Under the multiple CONWIP loop systems, the model is easy to generalize because only the location of fork/join stations and addition parallel machines make the models differ from one another. The modeling methodology presented in this study can easily be used to analyze different types of systems.

Specially, it is very easy to model a factory with flow lines producing several different products. Furthermore, the results show that the throughput of exponential service time systems without DPD is very close to the throughput of Erlang service time systems with DPD. This result emphasizes the importance of a smaller variance of service time for better performance.

The heuristic algorithm shows that DPD can improve throughput while requiring less inventory in the system. The heuristic algorithm finds a proper kanban configuration using simulation and NLP models. It is also able to find kanban configurations for systems with other types of service time distribution. The idea of the heuristic is to find a minimum kanban configuration using the NLP model and a maximum kanban configuration using simulation. Then, it finds a kanban configuration requiring fewer kanbans in a systematic way. The algorithm can be directly applied for systems having more variety of products. Although it would increase the number of simulation runs, the heuristic algorithm also can find kanban configurations without using the NLP model by setting lower bounds at 1. The heuristic algorithm is similar to the branch and bound algorithm in that it finds a feasible solution then searches for other feasible solutions with less total number of kanbans. A true branch and bound algorithm may better be used to search for an global optimal kanban configuration if the search space is small.

The implementation of DPD can shorten the time required to adapt to demand changes. The heuristic algorithm developed in this study can help to find a new configuration that satisfies the changed demand. Also, DPD can increase the maximum throughput of products when the new part of common process machines is the bottleneck process. Although, in the examples in the study, the first machine of the product specific CONWIP loops

becomes the new common machine, any machine in the product specific CONWIP loops could be common process machine. Also, the new common process machines do not necessarily have to be the last machine in the common process loop. In other words, it is possible that a system can be improved by changing any product specific process to a common process.

The model developed in this study can be applied where manufacturing facilities have a flow line of common operations in upstream processes and separate flow lines of product specific operations in downstream processes. It also can be applied in the supply chain consisting of one part supplier and multiple manufacturing facilities producing different finished products using the parts. The manufacturing processes of supplier can be designed as the common process CONWIP loop and the processes of each facility can be designed as a product specific loop. Furthermore, the same modeling methodology can be extended for modeling supply chains having more CONWIP loops and fork/join stations.

Some additional topics can be considered in the future. Models for different service time distributions can be developed. Although the heuristic algorithm works well in the example of our study, more complex systems would substantially increase the time required for finding a kanban configuration. The number of possible combinations of kanban configurations can be much bigger in real systems. By having a model with precise bounds for other distributions, the effort for finding the best kanban configuration can be greatly reduced. The improvement of bounds still is an issue for the exponential model. Although kanban systems can not be analyzed as a set of independent modules, there may be better ways to model them.

This research developed analytical models for kanban systems and CONWIP systems.

The models can find throughput bounds for performance evaluation, a kanban configuration to maximize performance with a given number of kanbans, and a minimum required kanban configuration for a given throughput requirement. A heuristic algorithm provides a method for finding the best configuration for a given throughput requirement. Although the methodology is examined using small examples, the methodology can be applied to real world problems. We also show that the systems with DPD can perform better with less inventory in the system. Using the methods developed in this study, one can analyze how much an existing system can be improved by implementing delayed product differentiation. It also is very useful as a tool for figuring out how to reallocate kanbans when demand changes. Moreover, the methodology can be used in other types of CONWIP systems.

APPENDICES

APPENDIX 1. LINGO MODEL NOTATIONS

Table A1. Notations in LINGO model

NLP model	LINGO model
K_p	NK_*p*
λ	L
ν	N
μ_i	M*i*
β_i	B*i*
γ_{ij}	G*ij*
δ_{ijk}	D*ijk*
ε_{ijkl}	E*ijkl*
w_i	W*i*
z_{ij}	Z*ij*
α_{ijk}	A*ijk*

APPENDIX 2. THREE MACHINE KANBAN LINGO MODEL

```
MODEL:

M1 = 1;
M2 = 1;
M3 = 1;

MAX = M1*W1;

N_K1 = 10;
N_K2 = 10;
N_K3 = 10;

@GIN(N_K1);
@GIN(N_K2);
@GIN(N_K3);

N_K1 + N_K2 + N_K3 = N_K;

! CONSTRAINT(2): X AND Y;
B1 <= N_K1*W1;
B2 <= N_K1*W2;
B3 <= N_K2*W3;
B4 <= N_K2*W4;
B5 <= N_K2*W5;
B6 <= N_K3*W6;
B7 <= N_K3*W7;

! CONSTRAINT(3):CONSTANT SYSTEM POPULATION;
B1 + B2       = N_K1;
B3 + B4 + B5 = N_K2;
B6 + B7       = N_K3;

! CONSTRAINT(4):;
B1     + G21   = N_K1*W1;
B2     + G12   = N_K1*W2;
G13            = N_K1*W3;
G14    + G24   = N_K1*W4;
G17    + G27   = N_K1*W7;
G13            = N_K1*Z13;
D146   + D246 = N_K1*Z46;
D157   + D257 = N_K1*Z57;
E1246 + D246 = N_K1*A246;
E1257 + D257 = N_K1*A257;

G31   + G41    + G51   = N_K2*W1;
B4    + G34    + G54   = N_K2*W4;
G37   + G47    + G57   = N_K2*W7;
G31   + D513   + D413 = N_K2*Z13;
D346 + G46             = N_K2*Z46;
D357 + D457   + G57   = N_K2*Z57;
        D426           = N_K2*A246;
        E4257 + D527 = N_K2*A257;
```

```
G61  + G71   = N_K3*W1;
G64  + G74   = N_K3*W4;
B7   + G76   = N_K3*W7;
D613 + D713  = N_K3*Z13;
G64  + D746  = N_K3*Z46;
       G75   = N_K3*Z57;
D624 + E7246 = N_K3*A246;
       E725  = N_K3*A257;

! CONSTRAINT(5): MEAN WAITING TIME;
B1 - W1*(1+B1) <= 0;
B4 - W4*(1+B4) <= 0;
B7 - W7*(1+B7) <= 0;

! CONSTRAINT(6);
W1 <= 1;  W4 <= 1;  W7 <= 1;

! CONSTRAINT(7);
W2 + W3 <= 1;  W5 + W6 <= 1;

! CONSTRAINT(8);
W1 <= B1;  W4 <= B4;  W7 <= B7;

! CONSTRAINT(9);
Z13 <= G13;   Z13 <= G31;
Z46 <= G46;   Z46 <= G64;
Z57 <= G57;   Z57 <= G75;

! CONSTRAINT(10);
A246 <= D246;  A246 <= D426;   A246 <= D624;
A257 <= D257;  A257 <= D527;   A257 <= D725;

! CONSTRAINT(11);
W1 >= Z13;  W4 >= Z46;  W7 >= Z57;

! CONSTRAINT (12);
Z46 >= A246; Z57 >= A257;

! CONSTRAINT(13);
G13 <= B1; G14 <= B1; G17 <= B1;
G41 <= B4; G46 <= B4; G47 <= B4;
G71 <= B7; G72 <= B7; G74 <= B7; G75 <= B7;

! CONSTRAINT(14);
G14 >= D146;  G17 >= D157;
G24 >= D246;  G27 >= D257;
G34 >= D346;  G37 >= D357;
G41 >= D413;  G46 >= D426;  G47 >= D457;
G51 >= D513;  G57 >= D527;
G61 >= D613;  G64 >= D624;
G71 >= D713;  G72 >= D725;
G72 >= D726;  G74 >= D746;

D146 >= E1246;  D157 >= E1257;
```

```
D457 >= E4257;
D726 >= E7246;  D746 >= E7246;

! CONSTRAINT(16-21);
W1 + W2 - Z12 = 1;
B1 - G12 = N_K1*(W1-Z12);
B2 - G21 = N_K1*(W2-Z12);
G31 = B3;
Z13 = W3;
Z12 + Z13 <= W1;
G12 + G21 = N_K1*Z12; G12 >= Z12; G21 >= Z12;
G12 >= E1246 + E1257;

W6 + W7 - Z67 = 1;
B6 - G67 = N_K3*(W6-Z67);
B7 - G76 = N_K3*(W7-Z67);
G57 = B5;
Z57 = W5;
Z57 + Z67 <= W7;
G67 + G76 = N_K1*Z67;  G67 >= Z67; G76 >= Z67;

Z13 + A246 + Z257 <= 1;
Z46 + Z57 <= 1;

G46 + D457 <= B4;
G41 + D426 <= B4;

! CONSTRAINT: STATIONARY FIRST MOMENTS;
!(1); M1*W1 - M1*Z13 - M2*A246 - M3*A257 = 0;
!(3); M1*W1 - M2*Z46 - M3*Z57 = 0;
!(4); M1*W1 - M2*W4 = 0;
!(5); M1*W1 - M3*W7 = 0;

! CONSTRAINT: STATIONARY SECOND MOMENTS;
!(1,1); M2*E1246 + M3*E1257 + M1*W1 - M1*B1 + M1*G13 - M1*Z13 = 0;
!(2,2); M1*G21 + M1*W1 - M1*Z13 - M2*D246 - M3*D257 = 0;
!(3,3); M2*D346 + M3*D357 - M1*G31 + M1*Z13 = 0;
!(4,4); M3*E4257 + M1*D413 - M2*B4 + M2*D426 + M2*W4 - M2*A246 = 0;
!(5,5); M2*G54 - M3*G57 + M3*Z57 = 0;
!(6,6); M3*G67 - M2*G64 + M2*Z46 = 0;
!(7,7); M2*D746 - M3*B7 + M2*Z46 + M3*G75 = 0;

!(1,2); M2*D246 + M3*D257 - M2*A246 - M3*A257 - M1*G21 = 0;
!(1,3); M2*D146 - M2*E1246 + M3*D157 - M3*E1257 - M1*G13 = 0;
!(2,4); M1*G41 - M1*D413 - M2*D426 - M3*E4257 + M3*D257 - M2*G24 + M2*D246
- M3*A257 = 0;
!(2,5); M1*G51 - M1*D513 - M3*D527 - M3*D257 + M3*A257 + M2*G24 - M2*D246
= 0;
!(2,6); M1*G61 - M1*D613 - M2*D246 - M2*D624 + M2*A246 + M3*G27 - M3*D257
= 0;
!(3,4); M3*D457 - M3*E4257 + M2*G46 - M2*D426 - M2*Z46 + M2*A246 - M1*D413
+ M1*G31 - M1*Z13 - M2*G34 =0;
!(3,5); -M1*D513 + M3*G57 - M3*D527 - M3*D357 - M3*Z57 + M3*A257 + M2*G34
- M2*D346 = 0;
```

```
!(3,6); -M1*D613 + M2*G64 - M2*D624 - M2*D346 - M2*Z46 + M2*A246 + M3*G37
- M3*D357 = 0;
!(4,5); M1*D513 + M3*D527 - M3*A257 - M2*G54 + M2*B4 - M2*G46 - M2*W4 +
M2*Z46 - M3*D457 = 0;
!(4,6); M1*D613 - M2*G64 + M2*D624 - M2*G46 +M2*Z46 - M2*A246 + M3*G47 -
M3*D457 = 0;
!(5,7); M2*G74 - M2*D746 - M3*G75 =0;
!(6,7); M3*B7 - M3*G75 - M3*G67 - M3*W7 + M3*Z57 - M2*D746 + M2*G64 -
M2*Z46 = 0;
!(1,4); M2*D426 - M1*G41 + M1*D413 + M3*E4257 + M3*E1257 + M3*A257 +
M1*G13 - M2*G14 + M2*E1246 = 0;
!(1,5); -M1*G51 + M1*D513 + M3*D527 - M3*A257 + M2*G14 - M2*D146 - M3*D157
= 0;
!(1,6); -M1*G61 + M1*D613 + M2*D624 - M2*A246 + M3*G17 - M3*D157 - M2*D146
= 0;
!(1,7); M3*D725 - M1*G71 + M1*D713 + M2*E7246 + M2*A246 + M2*D146 - M3*G17
+ M3*D157 = 0;
!(2,7); M1*G71 - M1*D713 - M3*D725 - M2*E7246 + M2*D246 - M2*A246 + M3*G27
- M3*D257 = 0;
!(3,7); M3*G75 - M3*D725 - M1*D713 + M2*D746 - M2*E7246 + M2*D346 + M2*Z46
- M2*A246 - M3*G37 + M3*D357 = 0;
!(4,7); M3*D725 + M1*D713 - M2*G72 + M2*D726 - M2*D746 + M2*E7246 + M2*G46
- M2*Z46 + M2*A246 - M3*G47 + M3*D457 = 0;

END
```

APPENDIX 3. TWO MACHINE CONWIP WITH EXPONENTIAL SERVICE TIME LINGO MODEL

```
Model:

M1 = 1.0;
M2 = 0.9;

Min = NK1;

@GIN(NK1);

! CONSTRAINT(2): X AND Y;
B1 <= NK1*W1;    B2 <= NK1*W2;

! CONSTRAINT(3): CONSTANT SYSTEM POPULATION;
B1 + B2 = NK1;

! CONSTRAINT(4):  ;
G12 + B2 = NK1*W2;
G21 + B1 = NK1*W1;

! CONSTRAINT(5): MEAN WAITING TIME;
B1 - W1*(1+B1) <= 0;  B2 - W2*(1+B2) <= 0;

! CONSTRAINT(6),(7);
W1 <= 1;  W2 <= 1;
W1 + W2 >= 1;

! CONSTRAINT(8);
W1 <= B1;  W2 <= B2;

! CONSTRAINT(13);
G12 <= B1;  G21 <= B2;

! CONSTRAINT(15);
G12 <= (NK1-1)*W1;  G21 <= (NK1-1)*W2;

!First Moment Constraint
!(1); M2*W2 - M1*W1 = 0;
!(2); !M1*W1 - M2*W2 = 0;

!Second Moment Constraint
!(1,1); M2*G12 - M1*B1 + M2*W2 = 0;
!(2,2); M1*G21 - M2*B2 + M1*W1 = 0;

!Second Cross Moment Constraint
(1,2); !M2*B2 - M2*G12 - M2*W2 - M1*G21 + M1*B1 - M1*W1 = 0;

END
```

APPENDIX 4. TWO MACHINE CONWIP WITH ERLANG SERVICE TIME LINGO MODEL

```
Model:

M1 = 1.0;
M2 = 0.9;

MIN = NK1;

@GIN(NK1);

! CONSTRAINT(2): X AND Y;
B1a <= NK1*W1a;   B2a <= NK1*W2a;

! CONSTRAINT(3): CONSTANT SYSTEM POPULATION;
B1a + B1b + B1c + B1d + B2a + B2b + B2c + B2d = NK1;

! CONSTRAINT(4):  ;
G1a2a + G1b2a + G1c2a + G1d2a <= NK1*W2a;
G1a2b + G1b2b + G1c2b + G1d2b + G2a2b <= NK1*W2b;
G1a2c + G1b2c + G1c2c + G1d2c + G2a2c <= NK1*W2c;
G1a2d + G1b2d + G1c2d + G1d2d + G2a2d <= NK1*W2d;
G2a1a + G2b1a + G2c1a + G2d1a <= NK1*W1a;
G2a1b + G2b1b + C2c1b + G2d1b + G1a1b <= NK1*W1b;
G2a1c + G2b1c + G2c1c + G2d1c + G1a1c <= NK1*W1c;
G2a1d + G2b1d + G2c1d + G2d1d + G1a1d <= NK1*W1d;

D1a2a2b + D1b2a2b + D1c2a2b + D1d2a2b + G2a2b <= NK1*Z2a2b;
D1a2a2c + D1b2a2c + D1c2a2c + D1d2a2c + G2a2c <= NK1*Z2a2c;
D1a2a2d + D1b2a2d + D1c2a2d + D1d2a2d + G2a2d <= NK1*Z2a2d;
D2a1a1b + D2b1a1b + D2c1a1b + D2d1a1b + G1a1b <= NK1*Z1a1b;
D2a1a1c + D2b1a1c + D2c1a1c + D2d1a1c + G1a1c <= NK1*Z1a1c;
D2a1a1d + D2b1a1d + D2c1a1d + D2d1a1d + G1a1d <= NK1*Z1a1d;

! CONSTRAINT(5): MEAN WAITING TIME;
4*B1a - W1a*(1+4*B1a) <= 0;  4*B2a - W2a*(1+4*B2a) <= 0;

! CONSTRAINT(6),(7);     !W1a >= W1b;
W1a + W1b + W1c + W1d - Z1a1b - Z1a1c - Z1a1d <= 1;
W2a + W2b + W2c + W2d - Z2a2b - Z2a2c - Z2a2d <= 1;

W1a <= 1;  W1b <= 1;  W1c <= 1;  W1d <= 1;
W2a <= 1;  W2b <= 1;  W2c <= 1;  W2d <= 1;

W1a + W1b + W1c + W1d + W2a + W2b + W2c + W2d >= 1;

! CONSTRAINT(8);
W1a <= B1a;  W2a <= B2a;

! CONSTRAINT(9);
Z1a1b <= G1a1b;  Z1a1c <= G1a1c;  Z1a1d <= G1a1d;
Z2a2b <= G2a2b;  Z2a2c <= G2a2c;  Z2a2d <= G2a2d;
```

```
! CONSTRAINT(10);
!A2b47 <= D2b47;

! CONSTRAINT(11);
W1a >= Z1a1b;  W1b >= Z1a1b;  W1a >= Z1a1c;  W1c >= Z1a1c;
W1a >= Z1a1d;  W1d >= Z1a1d;  W2a >= Z2a2b;  W2b >= Z2a2b;
W2a >= Z2a2c;  W2c >= Z2a2c;  W2a >= Z2a2d;  W2d >= Z2a2d;

! CONSTRAINT(12);
!Z2b4 >= A2b47;

! CONSTRAINT(13);
G1a1b <= B1a;  G1a1c <= B1a;  G1a1d <= B1a;  G1a2a <= B1a;
G1a2b <= B1a;  G1a2c <= B1a;  G1a2d <= B1a;  G1b2a <= B1b;
G1b2b <= B1b;  G1b2c <= B1b;  G1b2d <= B1b;  G1c2a <= B1c;
G1c2b <= B1c;  G1c2c <= B1c;  G1c2d <= B1c;  G1d2a <= B1d;
G1d2b <= B1d;  G1d2c <= B1d;  G1d2d <= B1d;  G2a2b <= B2a;
G2a2c <= B2a;  G2a2d <= B2a;  G2a1a <= B2a;  G2a1b <= B2a;
G2a1c <= B2a;  G2a1d <= B2a;  G2b1a <= B2b;  G2b1b <= B2b;
G2b1c <= B2b;  G2b1d <= B2b;  G2c1a <= B2c;  G2c1b <= B2c;
G2c1c <= B2c;  G2c1d <= B2c;  G2d1a <= B2d;  G2d1b <= B2d;
G2d1c <= B2d;  G2d1d <= B2d;

! CONSTRAINT(14);
D1a2a2b <= G1a2a;  D1a2a2b <= G1a2b;  D1a2a2c <= G1a2a;  D1a2a2c <= G1a2c;
D1a2a2d <= G1a2a;  D1a2a2d <= G1a2d;  D1b2a2b <= G1b2a;  D1b2a2b <= G1b2b;
D1c2a2b <= G1c2a;  D1c2a2b <= G1c2b;  D1c2a2c <= G1c2a;  D1c2a2c <= G1c2c;
D1c2a2d <= G1c2a;  D1c2a2d <= G1c2d;  D1d2a2b <= G1d2a;  D1d2a2b <= G1d2b;
D1d2a2c <= G1d2a;  D1d2a2c <= G1d2c;  D1d2a2d <= G1d2a;  D1d2a2d <= G1d2d;
D2a1a1b <= G2a1a;  D2a1a1b <= G2a1b;  D2a1a1c <= G2a1a;  D2a1a1c <= G2a1c;
D2a1a1d <= G2a1a;  D2a1a1d <= G2a1d;  D2b1a1b <= G2b1a;  D2b1a1b <= G2b1b;
D2b1a1c <= G2b1a;  D2b1a1c <= G2b1c;  D2b1a1d <= G2b1a;  D2b1a1d <= G2b1d;
D2c1a1b <= G2c1a;  D2c1a1b <= G2c1b;  D2c1a1c <= G2c1a;  D2c1a1c <= G2c1c;
D2c1a1d <= G2c1a;  D2c1a1d <= G2c1d;  D2d1a1b <= G2d1a;  D2d1a1b <= G2d1b;
D2d1a1c <= G2d1a;  D2d1a1c <= G2d1c;  D2d1a1d <= G2d1a;  D2d1a1d <= G2d1d;

! CONSTRAINT(15);
G1a1b <= (NK1-1)*W1a;  G1a1c <= (NK1-1)*W1a;  G1a1d <= (NK1-1)*W1a;
G1a2a <= (NK1-1)*W1a;  G1a2b <= (NK1-1)*W1a;
G1a2c <= (NK1-1)*W1a;  G1a2d <= (NK1-1)*W1a;
G1b2a <= (NK1-1)*W1b;  G1b2b <= (NK1-1)*W1b;
G1b2c <= (NK1-1)*W1b;  G1b2d <= (NK1-1)*W1b;
G1c2a <= (NK1-1)*W1c;  G1c2b <= (NK1-1)*W1c;
G1c2c <= (NK1-1)*W1c;  G1c2d <= (NK1-1)*W1c;
G1d2a <= (NK1-1)*W1d;  G1d2b <= (NK1-1)*W1d;
G1d2c <= (NK1-1)*W1d;  G1d2d <= (NK1-1)*W1d;

G2a2b <= (NK1-1)*W2a;  G2a2c <= (NK1-1)*W2a;  G2a2d <= (NK1-1)*W2a;
G2a1a <= (NK1-1)*W2a;  G2a1b <= (NK1-1)*W2a;
G2a1c <= (NK1-1)*W2a;  G2a1d <= (NK1-1)*W2a;
G2b1a <= (NK1-1)*W2b;  G2b1b <= (NK1-1)*W2b;
G2b1c <= (NK1-1)*W2b;  G2b1d <= (NK1-1)*W2b;
G2c1a <= (NK1-1)*W2c;  G2c1b <= (NK1-1)*W2c;
G2c1c <= (NK1-1)*W2c;  G2c1d <= (NK1-1)*W2c;
```

```
G2d1a <= (NK1-1)*W2d;  G2d1b <= (NK1-1)*W2d;
G2d1c <= (NK1-1)*W2d;  G2d1d <= (NK1-1)*W2d;

! CONSTRAINT(19);
W1 = W1a + W1b + W1c + W1d - Z1a1b - Z1a1c - Z1a1d;
W2 = W2a + W2b + W2c + W2d - Z2a2b - Z2a2c - Z2a2d;
B1 = B1a + B1b + B1c + B1d;
B2 = B2a + B2b + B2c + B2d;

!First Moment Constraint;
!(1a); M2*W2d - M1*W1a + M1*Z1a1b + M1*Z1a1c + M1*Z1a1d = 0;
!(1b); W2d = W1b;    !M1*W1a - M1*Z1a1b - M1*Z1a1c - M1*Z1a1d - M1*W1b =
0;
!(1c); W1b = W1c;
!(1d); W1c = W1d;
!(2a); M1*W1d - M2*W2a + M2*Z2a2b + M2*Z2a2c + M2*Z2a2d = 0;
!(2b); W1d = W2b;    !M2*W2a - M2*Z2a2b - M2*Z2a2c - M2*Z2a2d - M2*W2b =
0;
!(2c); W2b = W2c;
!(2d); W2c = W2d;

!Second Moment Constraint;
!(1a,1a); M2*G1a2d - M1*B1a + M1*G1a1b + M1*G1a1c + M1*G1a1d + M2*W2d = 0;
!(1b,1b); B1b = W1b;
!(1c,1c); B1c = W1c;
!(1d,1d); B1d = W1d;
!(2a,2a); M1*G2a1d - M2*B2a + M2*G2a2b + M2*G2a2c + M2*G2a2d + M1*W1d = 0;
!(2b,2b); B2b = W2b;
!(2c,2c); B2c = W2c;
!(2d,2d); B2d = W2d;

!Second Cross Moment Constraint;
!(1a,1b); M2*G1b2d + M1*B1a - M1*G1a1b - M1*G1a1c - M1*G1a1d - M1*W1a +
M1*Z1a1b + M1*Z1a1c + M1*Z1a1d - M1*G1a1b = 0;
         !M2*G1b2d + M2*G1a2d - M1*G1a1b = 0;
!(1a,1c); M2*G1c2d + M1*G1a1b - M1*G1a1c = 0;
!(1a,1d); M2*G1d2d + M1*G1a1c - M1*G1a1d = 0;
!(1a,2a); M2*G2a2d - M1*G2a1a + M1*D2a1a1b + M1*D2a1a1c + M1*D2a1a1d +
M1*G1a1d - M2*G1a2a + M2*D1a2a2b + M2*D1a2a2c + M2*D1a2a2d = 0;
!(1a,2b);          - M1*G2b1a + M1*D2b1a1b + M1*D2b1a1c + M1*D2b1a1d +
M2*G1a2a - M2*D1a2a2b - M2*D1a2a2c - M2*D1a2a2d - M2*G1a2b = 0;
!(1a,2c);          - M1*G2c1a + M1*D2c1a1b + M1*D2c1a1c + M1*D2c1a1d +
M2*G1a2b - M2*G1a2c = 0;
!(1a,2d); M2*B2d - M2*G1a2d - M2*W2d - M1*G2d1a + M1*D2d1a1b + M1*D2d1a1c
+ M1*D2d1a1d + M2*G1a2c = 0;
!(1b,1c); !B1b - W1b = 0;
!(1b,1d);
!(1b,2a); M1*G2a1a - M1*D2a1a1b - M1*D2a1a1c - M1*D2a1a1d - M1*G2a1b -
M2*G1b2a + M2*D1b2a2b + M2*D1b2a2c + M2*D1b2a2d = 0;
!(1b,2b); M1*G2b1a - M1*D2b1a1b - M1*D2b1a1c - M1*D2b1a1d - M1*G2b1b +
M2*G1b2a - M2*D1b2a2b - M2*D1b2a2c - M2*D1b2a2d - M2*G1b2b = 0;
!(1b,2c); M1*G2c1a - M1*D2c1a1b - M1*D2c1a1c - M1*D2c1a1d - M1*G2c1b +
M2*G1b2b - M2*G1b2c = 0;
!(1b,2d); M1*G2d1a - M1*D2d1a1b - M1*D2d1a1c - M1*D2d1a1d - M1*G2d1b +
M2*G1b2c - M2*G1b2d  = 0;
```

```
!(1c,1d); !B1c - W1c = 0;
!(1c,2a); M1*G2a1b - M1*G2a1c - M2*G1c2a + M2*D1c2a2b + M2*D1c2a2c +
M2*D1c2a2d = 0;
!(1c,2b); M1*G2b1b - M1*G2b1c + M2*G1c2a - M2*D1c2a2b - M2*D1c2a2c -
M2*D1c2a2d - M2*G1c2b = 0;
!(1c,2c); M1*G2c1b - M1*G2c1c + M2*G1c2b - M2*G1c2c = 0;
!(1c,2d); M1*G2d1b - M1*G2d1c + M2*G1c2c - M2*G1c2d  = 0;
!(1d,2a); M1*G2a1c - M1*G2a1d + M1*B1d - M1*W1d - M2*G1d2a + M2*D1d2a2b +
M2*D1d2a2c + M2*D1d2a2d = 0;
!(1d,2b); M1*G2b1c - M1*G2b1d + M2*G1d2a - M2*D1d2a2b - M2*D1d2a2c -
M2*D1d2a2d - M2*G1d2b = 0;
!(1d,2c); M1*G2c1c - M1*G2c1d + M2*G1d2b - M2*G1d2c = 0;
!(1d,2d); M1*G2d1c - M1*G2d1d + M2*G1d2c - M2*G1d2d = 0;
!(2a,2b); M1*G2b1d + M2*B2a - M2*G2a2b - M2*G2a2c - M2*G2a2d - M2*W2a +
M2*Z2a2b + M2*Z2a2c + M2*Z2a2d - M2*G2a2b = 0;
          !M1*G2b1d + M1*G2a1d - M2*G2a2b = 0;
!(2a,2c); M1*G2c1d + M2*G2a2b - M2*G2a2c = 0;
!(2a,2d); M1*G2d1d + M2*G2a2c - M2*G2a2d = 0;
!(2b,2c); !B2b - W2b = 0;
!(2b,2d);
!(2c,2d); !B2c - W2c = 0;

END
```

APPENDIX 5. CONWIP WITHOUT DPD LINGO MODEL

```
Model:

M1 = 1.0;
M2 = 0.9;
M3 = 0.6;
M4 = 0.5;
M5 = 0.4;
M6 = 0.5;

MAX = M1*W1;

NK1 = 10;
NK2  = 5;
NK3  = 5;

@GIN(NK1);
@GIN(NK2);
@GIN(NK3);

! CONSTRAINT(2): X AND Y;
B1 <= NK1*W1;      B2 <= NK1*W2;      B3 <= NK1*W3;
B4 <= NK2*W4;      B5 <= NK2*W5;      B6 <= NK2*W6;
B7 <= NK3*W7;      B8 <= NK3*W8;      B9 <= NK3*W9;

! CONSTRAINT(3): CONSTANT SYSTEM POPULATION;
B1 + B2 + B3 = NK1;
B4 + B5 + B6 = NK2;
B7 + B8 + B9 = NK3;

! CONSTRAINT(4):  ;
B1     + G21    + G31    = NK1*W1;          G12    + B2     + G32    = NK1*W2;
         G24             <= NK1*W4;                   G24             <= NK1*W4;
G15    + G25    + G35    = NK1*W5;          G16    + G26    + G36    = NK1*W6;
         G27             <= NK2*W7;          G18    + G28    + G38    = NK1*W8;

G19    + G29    + G39    = NK1*W9;
G41    + G51    + G61    = NK2*W1;          G42    + G52    + G62    = NK2*W2;
                  G63    <= NK2*W3;
G45    + B5     + G65    = NK2*W5;          G46    + G56    + B6     = NK2*W6;
G48    + G58    + G68    = NK2*W8;          G49    + G59    + G69    = NK2*W9;
G71    + G81    + G91    = NK3*W1;          G72    + G82    + G92    = NK3*W2;
                  G93    <= NK3*W3;
G75    + G85    + G95    = NK3*W5;          G76    + G86    + G96    = NK3*W6;
G78    + B8     + G98    = NK3*W8;          G79    + G89    + B9     = NK3*W9;


D124   + G24             = NK1*Z24;         D127   + G27             = NK1*Z27;
D136   + D236   + G36    = NK1*Z36;         D139   + D239   + G39    = NK1*Z39;
G42    + D524   + D624   = NK2*Z24;                  D527   + D627  <= NK2*Z27;
         D536   + G63    = NK2*Z36;                  D539   + D639   = NK2*Z39;
D724   + D824   + D924   = NK3*Z24;         G72    + D827   + D927   = NK3*Z27;
```

```
        D836  + D936  = NK3*Z36;                    D839  + G93   = NK3*Z39;

E1247 + D247           = NK1*A247;
        E5247 + E6247 <= NK2*A247;
D724  + E8247 + E9247  = NK3*A247;

! CONSTRAINT(5): MEAN WAITING TIME;
B1 - W1*(1+B1) <= 0;    B2 - W2*(1+B2) <= 0;
B5 - W5*(1+B5) <= 0;    B6 - W6*(1+B6) <= 0;
B8 - W8*(1+B8) <= 0;    B9 - W9*(1+B9) <= 0;

! CONSTRAINT(6),(7);
W1 <= 1;  W2 <= 1;  W5 <= 1;  W6 <=1;  W8 <= 1;  W9 <=1;
W3 + W4 <= 1;  W3 + W7 <= 1;
W1 + W2 + W3 >= 1;  W4 + W5 + W6 >= 1;
W7 + W8 + W9 >= 1;

! CONSTRAINT(8);
W1 <= B1;  W2 <= B2;  W3 <= B3;  W4 <= B4;  W5 <= B5;
W6 <= B6;  W7 <= B7;  W8 <= B8;  W9 <= B9;

! CONSTRAINT(9);
Z24 <= G24;  Z24 <= G42;  Z36 <= G36;  Z36 <= G63;
Z27 <= G27;  Z27 <= G72;  Z39 <= G39;  Z39 <= G93;

! CONSTRAINT(10);
A247 <= D247;  A247 <= D724;

! CONSTRAINT(11);
W2 >= Z24;  W4 >= Z24;  W2 >= Z27;  W7 >= Z27;
W3 >= Z36;  W6 >= Z36;  W3 >= Z39;  W9 >= Z39;

! CONSTRAINT(12);
Z24 >= A247;  Z27 >= A247;

! CONSTRAINT(13);
B1 >= G12;  B1 >= G15;  B1 >= G16;  B1 >= G18;
B1 >= G19;  B2 >= G21;  B2 >= G24;  B2 >= G25;
B2 >= G26;  B2 >= G27;  B2 >= G28;  B2 >= G29;
B3 >= G31;  B3 >= G32;  B3 >= G35;  B3 >= G36;
B3 >= G38;  B3 >= G39;  B4 >= G41;  B4 >= G42;
B4 >= G45;  B4 >= G46;  B4 >= G49;  B5 >= G51;
B5 >= G52;  B5 >= G56;  B6 >= G61;  B6 >= G62;
B6 >= G63;  B6 >= G65;  B6 >= G69;  B7 >= G71;
B7 >= G72;  B7 >= G78;  B7 >= G79;  B8 >= G81;
B8 >= G82;  B8 >= G89;  B9 >= G91;  B9 >= G92;
B9 >= G93;  B9 >= G98;
B4 >= G48;  B5 >= G58;  B5 >= G59;  B6 >= G68;
B7 >= G75;  B7 >= G76;  B8 >= G85;  B8 >= G86;
B9 >= G95;  B9 >= G96;

! CONSTRAINT(14);
G12 >= D124;  G12 >= D127;  G16 >= D136;  G19 >= D139;  G26 >= D236;
G29 >= D239;  G24 >= D247;  G27 >= D247;  G52 >= D524;  G52 >= D527;
G56 >= D536;  G62 >= D624;  G62 >= D627;  G63 >= D639;  G69 >= D639;
```

```
G72 >= D724;  G82 >= D824;  G82 >= D827;  G89 >= D839;  G92 >= D924;
G92 >= D927;  G93 >= D936;
G59 >= D539;  G86 >= D836;  G96 >= D936;

D124 >= E1247;  D127 >= E1247;  D524 >= E5247;
D527 >= E5247;  D624 >= E6247;
D627 >= E6247;  D824 >= E8247;  D827 >= E8247;
D924 >= E9247;  D927 >= E9247;

! CONSTRAINT(15);
!B1 + G21 <= NK1*W1;  !G12 + B2 <= NK1*W2;  !B1 + G31 <= NK1*W1;  !B2 +
G32 <= NK1*W2;
!G45 + B5 <= NK2*W5;  !G46 + B6 <= NK2*W6;  !G56 + B6 <= NK2*W6;  !B5 +
G65 <= NK2*W5;
!G78 + B8 <= NK3*W8;  !G79 + B9 <= NK3*W9;  !G89 + B9 <= NK3*W9;  !B8 +
G98 <= NK3*W8;

! CONSTRAINT(16);
G21 <= (NK1-1)*W2;  G12 <= (NK1-1)*W1;  G31 <= (NK1-1)*W3;  G32 <= (NK1-
1)*W3;
G45 <= (NK2-1)*W4;  G46 <= (NK2-1)*W4;  G56 <= (NK2-1)*W5;  G65 <= (NK2-
1)*W6;
G78 <= (NK3-1)*W7;  G79 <= (NK3-1)*W7;  G89 <= (NK3-1)*W8;  G98 <= (NK3-
1)*W9;

! CONSTRAINT: STATIONARY FIRST MOMENT;
!(1); M2*Z24 + M2*Z27 - M2*A247 + M4*Z36 + M6*Z39 - M1*W1 = 0;
!(2); M1*W1 - M2*W2 = 0;
!(3);
!(4); M4*W6 - M4*Z36 - M2*Z24 = 0;
!(5); M4*W6 - M3*W5 = 0;
!(6);
!(7); M6*W9 - M6*Z39 - M2*Z27 + M2*A247 = 0;
!(8); M6*Z39 + M2*Z27 - M2*A247 - M5*W8 = 0;
!(9);

! CONSTRAINT: STATIONARY SECOND MOMENT;
!(1,1); M2*D124 + M2*D127 - M2*E1247 + M4*D136 + M6*D139 - M1*B1 + M1*W1 =
0;
!(2,2); M1*G21 + M1*W1 - M2*B2 = 0;
!(3,3); M2*G32 - M4*G36 - M6*G39 + M4*Z36 + M6*Z39 = 0;
!(4,4); M4*G46 - M2*G42 + M2*Z24 = 0;
!(5,5); M2*D524 + M4*D536 - M3*B5 + M3*W5 = 0;
!(6,6); M3*G65 - M4*B6 + M4*W6 = 0;
!(7,7); M6*G79 + M6*W9 - M6*Z39 - M2*G72 + M2*D724 = 0;
!(8,8); M2*D827 - M2*E8247 + M6*D839 - M5*B8 + M5*W8 = 0;
!(9,9); M5*G98 - M6*B9 + M6*W9 = 0;

! CONSTRAINT: STATIONARY CROSS MOMENT (BETWEEN DIFF. PRODUCT BUFFERS);
!(1,2) ; M4*D236 + M6*D239 + M2*G24 + M2*G27 - M2*D247 - M2*Z24 - M2*Z27 +
M2*A247 - M1*G21 + M1*B1 - M1*W1 - M2*G12 = 0;
!(1,3) ; M4*G36 + M6*G39 - M4*D136 - M6*D139 - M4*Z36 - M6*Z39 - M1*G31 +
M2*G12 - M2*D127 - M2*D124 + M2*E1247 = 0;
!(1,4) ; M2*G42 - M2*D124 - M2*Z24 - M1*G41 + M4*G16 - M4*D136 = 0;
```

```
!(1,5) ; M2*D527 - M2*E5247 + M6*D539 + M2*D524 + M4*D536 + M2*D124 +
M4*D136 + M2*Z24 + M4*Z36 - M1*G51 - M3*G15 = 0;
!(1,6) ; M2*D624 + M2*D627 - M2*E6247 + M6*D639 + M4*G63 - M4*D136 -
M4*Z36 - M1*G61 + M3*G15 - M4*G16 + M4*D136 = 0;
!(1,7) ; M2*G72 - M2*D127 + M2*E1247 - M2*Z27 + M2*A247 - M1*G71 + M6*G19
- M6*D139 = 0;
!(1,8) ; M2*D824+ M4*D836 + M2*D827 - M2*E8247 + M6*D839 + M2*D127 -
M2*E1247 + M6*D139 + M2*Z27 - M2*A247 + M6*Z39 - M1*G81 - M5*G18 = 0;
!(1,9) ; M2*D924 + M2*D927 - M2*E9247 + M4*D936 + M6*G93 - M6*D139 -
M6*Z39 - M1*G91 + M5*G18 - M6*G19 + M6*D139 = 0;
!(2,3) ; M1*G31 - M2*G32 + M2*B2 - M2*G24 - M2*G27 + M2*D247 - M2*W2 +
M2*Z24 + M2*Z27 - M2*A247 - M4*D236 - M6*D239 = 0;
!(2,4) ; M1*G41 - M2*G42 - M2*G24 + M2*Z24 + M4*G26 - M4*D236 = 0;
!(2,5) ; M1*G51 - M2*G52 + M2*G24 - M2*Z24 + M4*D236 - M3*G25 =0;
!(2,6); M1*G61 - M2*G62 + M3*G25 - M4*G26 =0;
!(2,7) ; M1*G71 - M2*G72 - M2*G27 + M2*D247 + M2*Z27 - M2*A247 + M6*G29 -
M6*D239 = 0;
!(2,8) ; M1*G81 - M2*G82 + M2*G27 - M2*D247 - M2*Z27 + M2*A247 + M6*D239 -
M5*G28 = 0;
!(2,9); M1*G91 - M2*G92 + M5*G28 - M6*G29 =0;
!(3,4);
!(3,5) ; M2*G52 - M2*D524 - M2*D527 + M2*E5247 - M6*D539 - M4*D536 +
M4*G36 - M4*Z36 - M3*G35 = 0;
!(3,6) ; M2*G62 - M2*D624 - M2*D627 + M2*E6247 - M6*D639 - M4*G63 - M4*G36
+ M4*Z36 + M3*G35 = 0;
!(3,7);
!(3,8) ; M2*G82 - M2*D824 - M2*D827 + M2*E8247 - M4*D836 - M6*D839 +
M6*G39 - M6*Z39 - M5*G38 = 0;
!(3,9) ; M2*G92 - M2*D924 - M2*D927 + M2*E9247 - M4*D936 - M6*G93 - M6*G39
+ M6*Z39 + M5*G38 = 0;
!(4,5) ; M4*G56 - M4*D536 - M2*D524 + M2*G42 - M2*Z24 - M3*G45 = 0;
!(4,6) ; M4*B6 - M4*G63 - M4*G46 - M4*W6 + M4*Z36 - M2*D624 + M3*G45 = 0;
!(4,7); M4*G76 - M2*D724 + M6*G49 = 0;
!(4,8); M4*G86 - M4*D836 - M2*D824 - M5*G48 =0;
!(4,9); M4*G96 - M4*D936 - M2*D924 + M5*G48 - M6*G49 =0;
!(5,6) ; M2*D624 + M4*G63 - M4*Z36 - M3*G65 + M3*B5 - M3*W5 - M4*G56 = 0;
!(5,7); M2*D724 - M3*G75 + M6*G59 - M6*D539 - M2*D527 + M2*E5247 =0;
!(5,8); M2*D824 + M4*D836 - M3*G85 + M2*D527 - M2*E5247 + M6*D539 - M5*G58
=0;
!(5,9); M2*D924 + M4*D936 - M3*G95 + M5*G58 - M6*G59 =0;
!(6,7); M3*G75 - M4*G76 + M6*G69 - M6*D639 - M2*D627 + M2*E6247 =0;
!(6,8); M3*G85 - M4*G86 + M2*D627 - M2*E6247 + M6*D639 - M5*G68=0;
!(6,9); M3*G95 - M4*G96 + M5*G68 - M6*G69 =0;
!(7,8) ; M6*G89 - M6*D839 - M2*D827 + M2*E8247 + M2*G72 - M2*D724 - M2*Z27
+ M2*A247 - M5*G78 = 0;
!(7,9) ; M6*B9 - M6*G93 - M6*G79 - M6*W9 + M6*Z39 - M2*D927 + M2*E9247 +
M5*G78 = 0;
!(8,9) ; M2*D927 - M2*E9247 + M6*G93 - M6*Z39 - M5*G98 + M5*B8 - M5*W8 -
M6*G89 = 0;
END
```

APPENDIX 6. CONWIP WITH DPD LINGO MODEL

```
Model:

M1 = 1.0;
M2 = 0.9;
M3 = 0.6;
M4 = 0.4;
M5 = 0.5;
M6 = 0.5;

MAX = M1*W1;

NK1 =  8;
NK2 = 1;
NK3 = 1;

@GIN(NK1);
@GIN(NK2);
@GIN(NK3);

! CONSTRAINT(2): X AND Y;
B1 <= NK1*W1;      B2 <= NK1*W2;      B3 <= NK1*W3;
B4 <= NK1*W4;      B5 <= NK1*W5;      B6 <= NK1*W6;
B7 <= NK2*W7;      B8 <= NK2*W8;      B9 <= NK3*W9;
B0 <= NK3*W0;

! CONSTRAINT(3): CONSTANT SYSTEM POPULATION;
B1 + B2 + B3 + B4 + B5 + B6 = NK1;
B7 + B8 = NK2;
B9 + B0 = NK3;

! CONSTRAINT(4):  ;
B1  + G21 + G31 + G41 + G51 + G61 = NK1*W1;
G12 + B2  + G32 + G42 + G52 + G62 = NK1*W2;
G14 + G24 + G34 + B4  + G54 + G64 = NK1*W4;
G15 + G25 + G35 + G45 + B5  + G65 = NK1*W5;
G18 + G28 + G38 + G48 + G58 + G68 = NK1*W8;
G10 + G20 + G30 + G40 + G50 + G60 = NK1*W0;
G71 + G81 = NK2*W1;  G72 + G82 = NK2*W2;
G74 + G84 = NK2*W4;  G75 + G85 = NK2*W5;
      G86 = NK2*W6;  G78 + B8  = NK2*W8;
G70 + G80 = NK2*W0;  G91 + G01 = NK3*W1;
G92 + G02 = NK3*W2;  G94 + G04 = NK3*W4;
G95 + G05 = NK3*W5;        G06 = NK3*W6;
G98 + G08 = NK3*W8;  G90 + B0  = NK3*W0;

D124 + G24  + D324 + G42 + D524 + D624 = NK1*Z24;
D134 + D234 + G34  + G43 + D534 + D634 = NK1*Z34;
D135 + D235 + G35  + G435+ D534 + D635 = NK1*Z35;
D147 + D247 + D347 + G47 + D547 = NK1*Z47;
D149 + D249 + D349 + G49 + D549 = NK1*Z49;
D157 + D257 + D357 + D457 + G57 = NK1*Z57;
```

```
D159 + D259 + D359 + D459 + G59 = NK1*Z59;
D168 + D268 + D368 + D468 + D568 + G68 = NK1*Z68;
D160 + D260 + D360 + D460 + D560 + G60 = NK1*Z60;
D724 + D824 = NK2*Z24;    D734 + D834 = NK2*Z34;
D735 + D835 = NK2*Z35;     G74 + D847 = NK2*Z47;
G75  + D857 = NK2*Z57;           D860 = NK2*Z60;
       G86 = NK2*Z68;    D924 + D024 = NK3*Z24;
D934 + D034 = NK3*Z34;    D935 + D035 = NK3*Z35;
D947 + D047 = NK3*Z47;    G94  + D049 = NK3*Z49;
D957 + D057 = NK3*Z57;    G95  + D059 = NK3*Z59;
D068 = NK3*Z68;                  G06 = NK3*Z60;

E1479 + E2479 + E3479 + D479 + E5479 = NK1*A479;
E1579 + E2579 + E3579 + E4579 + D579 = NK1*A579;
E7245 + E8245 = NK2*A245;  !D749 + E8479 = NK2*A479;
E9245 + E0245 = NK3*A245;  D947 + E0479 = NK3*A479;
D957 + E0579 = NK3*A579;

! CONSTRAINT(5): MEAN WAITING TIME;
B1 - W1*(1+B1) <= 0;     B2 - W2*(1+B2) <= 0;
!B4 - W4*(1+B4) <= 0;    !B5 - W5*(1+B5) <= 0;
B8 - W8*(1+B8) <= 0;     B0 - W0*(1+B0) <= 0;

! CONSTRAINT(6),(7);
W1 <= 1;  W2 <= 1;  W3 <= 1;  W4 <= 1;  W5 <= 1;
W6 <= 1;  W7 <= 1;  W8 <= 1;  W9 <= 1;  W0 <= 1;
W6 + W7 <= 1;  W6 + W9 <= 1;  W3 <= W4;  W3 <= W5;
W1 + W2 + W3 + W4 + W5 + W6 >= 1;
W7 + W8 >= 1;  W9 + W0 >= 1;

! CONSTRAINT(8);
W1 <= B1;  W2 <= B2;  W3 <= B3;  W4  = B4;  W5  = B5;
W6 <= B6;  W7 <= B7;  W8 <= B8;  W9 <= B9;  W0 <= B0;

! CONSTRAINT(9);
Z24 <= G24;  Z24 <= G42;  Z34 <= G34;  Z34 <= G43;  Z35 <= G35;
Z35 <= G53;  Z47 <= G47;  Z47 <= G74;  Z49 <= G49;  Z49 <= G94;
Z57 <= G57;  Z57 <= G75;  Z59 <= G59;  Z59 <= G95;  Z68 <= G68;
Z68 <= G86;  Z60 <= G60;  Z60 <= G06;

! CONSTRAINT(10);
A245 <= D245;  A245 <= D425;  A245 <= D524;  A347 <= D347;
A347 <= D734;  A349 <= D349;  A349 <= D934;  A357 <= D357;
A357 <= D735;  A359 <= D359;  A359 <= D935;  A479 <= D479;
A479 <= D947;  A579 <= D579;  A579 <= D957;

! CONSTRAINT(11);
W2 >= Z24;  W4 >= Z24;  W3 >= Z34;  W4 >= Z34;  W3 >= Z35;
W5 >= Z35;  W4 >= Z47;  W7 >= Z47;  W4 >= Z49;  W9 >= Z49;
W5 >= Z57;  W7 >= Z57;  W5 >= Z59;  W9 >= Z59;  W6 >= Z68;
W8 >= Z68;  W6 >= Z60;  W0 >= Z60;

! CONSTRAINT(12);
Z24 >= A245;  Z34 >= A347;  Z47 >= A347;  Z34 >= A349;
Z49 >= A349;  Z35 >= A357;  Z57 >= A357;  Z35 >= A359;
```

```
Z59 >= A359;  Z47 >= A479;  Z49 >= A479;  Z57 >= A579;
Z59 >= A579;

! CONSTRAINT(13);
B1 >= G12;  B1 >= G14;  B1 >= G15;  B1 >= G18;  B1 >= G10;  B2 >= G21;
B2 >= G24;  B2 >= G25;  B3 >= G31;  B3 >= G32;  B3 >= G34;  B3 >= G35;
B3 >= G38;  B3 >= G30;  B4 >= G41;  B4 >= G42;  B4 >= G43;  B4 >= G45;
B4 >= G47;  B4 >= G48;  B4 >= G49;  B4 >= G40;  B5 >= G51;  B5 >= G52;
B5 >= G53;  B5 >= G54;  B5 >= G57;  B5 >= G58;  B5 >= G59;  B5 >= G50;
B6 >= G61;  B6 >= G62;  B6 >= G64;  B6 >= G65;  B6 >= G68;  B6 >= G60;
B7 >= G71;  B7 >= G72;  B7 >= G74;  B7 >= G75;  B7 >= G78;  B8 >= G81;
B8 >= G82;  B8 >= G84;  B8 >= G85;  B8 >= G86;  B9 >= G91;  B9 >= G92;
B9 >= G94;  B9 >= G95;  B9 >= G90;  B0 >= G01;  B0 >= G02;  B0 >= G04;
B0 >= G05;  B0 >= G06;
B2 >= G20;  B2 >= G28;  B7 >= G70;  B8 >= G80;  B9 >= G98;  B0 >= G08;

! CONSTRAINT(14);
G12 >= D124;  G14 >= D124;  G14 >= D134;  G15 >= D135;  G14 >= D147;
G14 >= D149;  G15 >= D157;  G15 >= D159;  G18 >= D168;  G10 >= D160;
G24 >= D234;  G25 >= D235;  G24 >= D245;  G25 >= D245;  G24 >= D247;
G24 >= D249;  G25 >= D257;  G34 >= D347;  G34 >= D349;  G35 >= D359;
G38 >= D368;  G30 >= D360;  G45 >= D457;  G47 >= D457;  G45 >= D459;
G49 >= D459;  G48 >= D468;  G40 >= D460;  G47 >= D479;  G49 >= D479;
G54 >= D547;  G57 >= D547;  G54 >= D549;  G59 >= D549;  G58 >= D568;
G50 >= D560;  G57 >= D579;  G59 >= D579;  G62 >= D624;  G64 >= D624;
G64 >= D634;  G65 >= D635;  G72 >= D724;  G74 >= D724;  G74 >= D734;
G75 >= D735;  G82 >= D824;  G84 >= D824;  G84 >= D834;  G85 >= D835;
G84 >= D847;  G84 >= D849;  G85 >= D857;  G85 >= D859;  G86 >= D860;
G92 >= D924;  G94 >= D924;  G94 >= D934;  G95 >= D935;  G94 >= D947;
G95 >= D957;  G02 >= D024;  G04 >= D024;  G04 >= D034;  G04 >= D047;
G04 >= D049;  G05 >= D057;  G05 >= D059;  G06 >= D068;  G28 >= D268;
G20 >= D260;  G32 >= D324;  G34 >= D324;  G42 >= D425;  G45 >= D425;
G43 >= D435;  G45 >= D435;  G52 >= D524;  G54 >= D524;  G53 >= D534;
G54 >= D534;  G80 >= D860;  G08 >= D068;

D124 >= E1245;  D147 >= E1479;  D149 >= E1479;  D157 >= E1579;
D159 >= E1579;  D247 >= E2479;  D249 >= E2479;  D257 >= E2579;
D259 >= E2579;  D347 >= E3479;  D349 >= E3479;  D357 >= E3579;
D359 >= E3579;  D457 >= E4579;  D459 >= E4579;  D547 >= E5479;
D549 >= E5479;  D624 >= E6245;  D724 >= E7245;  D824 >= E8245;
D847 >= E8479;  D849 >= E8479;  D857 >= E8579;  D859 >= E8579;
D924 >= E9245;  D024 >= E0245;  D034 >= E0347;  D035 >= E0357;
D047 >= E0479;  D049 >= E0479;  D057 >= E0579;  D059 >= E0579;
D324 >= E3245;

! CONSTRAINT(16);
G12 <= (NK1-1)*W1;  G14 <= (NK1-1)*W1;
G15 <= (NK1-1)*W1;  G21 <= (NK1-1)*W2;
G24 <= (NK1-1)*W2;  G25 <= (NK1-1)*W2;
G31 <= (NK1-1)*W3;  G32 <= (NK1-1)*W3;
G34 <= (NK1-1)*W3;  G35 <= (NK1-1)*W3;
G41 <= (NK1-1)*W4;  G42 <= (NK1-1)*W4;
G43 <= (NK1-1)*W4;  G45 <= (NK1-1)*W4;
G51 <= (NK1-1)*W5;  G52 <= (NK1-1)*W5;
G53 <= (NK1-1)*W5;  G54 <= (NK1-1)*W5;
```

```
G61 <= (NK1-1)*W6;  G62 <= (NK1-1)*W6;
G64 <= (NK1-1)*W6;  G65 <= (NK1-1)*W6;
G78 <= (NK2-1)*W7;  G90 <= (NK3-1)*W9;
G52 <= (NK1-1)*W5;

D124 <= (NK1-2)*W1;  D134 <= (NK1-2)*W1;
D135 <= (NK1-2)*W1;  D234 <= (NK1-2)*W2;
D235 <= (NK1-2)*W2;  D245 <= (NK1-2)*W2;
D624 <= (NK1-2)*W6;  D634 <= (NK1-2)*W6;
D635 <= (NK1-2)*W6;  D324 <= (NK1-2)*W3;
D425 <= (NK1-2)*W4;  D435 <= (NK1-2)*W4;
D524 <= (NK1-2)*W5;  D534 <= (NK1-2)*W5;

E1245 <= (NK1-3)*W1;  E3245 <= (NK1-3)*W3;
E6245 <= (NK1-3)*W6;

! CONSTRAINT: STATIONARY FIRST MOMENT;
!(1); M3*Z47 + M3*Z49 - M3*A479 + M4*Z57 + M4*Z59 - M4*A579 + M5*Z68 +
M6*Z60 - M1*W1 = 0;
!(2); M1*W1 - M2*W2 = 0;
!(3); M2*A245 - M3*Z34 - M4*Z35 = 0;
!(4); M2*W2 - M2*Z24 - M3*W4 + M3*Z34 = 0;
!(5); M2*Z24 - M2*A245 - M4*W5 + M4*Z35 = 0;
!(6); M3*W4 + M4*W5 - M1*W1 = 0;
!(7); M5*W8 - M5*Z68 - M3*Z47 - M4*Z57 = 0;
!(8);
!(9); M6*W0 - M6*Z60 - M3*Z49 + M3*A479 - M4*Z59 + M4*A579 = 0;
!(0);

! CONSTRAINT: STATIONARY SECOND MOMENT;
!(1,1); M3*D147 + M3*D149 - M3*E1479 + M4*D157 + M4*D159 - M4*E1579 +
M5*D168 + M6*D160 - M1*B1 + M1*W1 = 0;
!(2,2); M1*G21 - M2*B2 + M2*W2 = 0;
!(3,3); M2*E3245 - M3*G34 - M4*G35 + M2*A245 = 0;
!(4,4); - M3*B4 + M3*G43 + M3*W4 - M3*Z34 = 0;
!(5,5); - M4*B5 + M4*G53 + M4*W5 - M4*Z35 = 0;
!(6,6); M3*G64 + M4*G65 - M5*G68 - M6*G60 + M5*Z68 + M6*Z60 = 0;
!(7,7); M5*G78 - M3*G74 - M4*G75 + M3*Z47 + M4*Z57 = 0;
!(8,8); M3*D847 + M4*D857 - M5*B8 + M5*G86 + M5*W8 - M5*Z68 = 0;
!(9,9); M6*G90 - M3*G94 + M3*D947 - M4*G95 + M4*D957 + M6*W0 - M6*Z60 = 0;
!(0,0); M3*D049 - M3*E0479 + M4*D059 - M4*E0579 - M6*B0 + M6*G06 + M6*W0 -
M6*Z60 = 0;

!(1,2) ; M3*D247 + M3*D249 - M3*E2479 + M4*D257 + M4*D259 - M4*E2579 +
M5*D268 + M6*D260 - M1*G21 + M1*B1 - M1*W1 - M2*G12 = 0;
!(1,3) ; M5*D368 - M6*D360 + M3*D347 + M3*D349 - M3*E3479 + M4*D357 +
M4*D359 - M4*E3579 - M3*A347 - M3*A349 + M3*P3479 - M4*A357 - M4*A359 +
M4*P3579 - M1*G31 + M2*E1245 - M3*D134 - M4*D135 = 0;
!(1,4) ; M4*D457 + M4*D459 - M4*E4579 + M5*D468 + M6*D460 + M3*G47 +
M3*G49 - M3*D479 - M3*Z47 + M3*A347 - M3*Z49 + M3*A349 + M3*A479 -
M3*P3479 - M1*G41 + M2*G12 - M2*D124 - M3*G14 + M3*D134 = 0;
!(1,5) ; M3*D547 + M3*D549 - M3*E5479 + M5*D568 + M6*D560 + M4*G57 +
M4*G59 - M4*D579 - M4*Z57 + M4*A357 - M4*Z59 + M4*A359 + M4*A579 -
M4*P3579 - M1*G51 + M2*D124 - M2*E1245 - M4*G15 + M4*D135 = 0;
```

```
!(1,6) ; M5*G68 + M6*G60 - M5*D168 - M6*D160 - M5*Z68 - M6*Z60 - M1*G61 +
M3*G14 - M3*D147 - M3*D149 + M3*E1479 + M4*G15 - M4*D157 - M4*D159 +
M4*E1579 = 0;
!(1,7) ; M3*G74 + M4*G75 - M3*D147 - M3*D157 - M3*Z47 - M4*Z57 - M1*G71 +
M5*G18 - M5*D168 = 0;
!(1,8) ; M3*D849 - M3*E8479 + M4*D859 - M4*E8579 + M5*G86 + M6*D860 +
M3*D847 + M4*D857 + M3*D147 + M4*D157 + M3*Z47 + M4*Z57 - M1*G81 - M5*G18
+ M5*D168 = 0;
!(1,9) ; M3*G94 + M4*G95 - M3*D149 + M3*E1479 - M4*D159 + M4*E1579 -
M3*Z49 + M3*A479 - M4*Z59 + M4*A579 - M1*G91 + M6*G10 - M6*D160 = 0;
!(1,0) ; M3*D047 + M4*D057 + M5*D068 + M6*G06 + M3*D049 - M3*E0479 +
M4*D059 - M4*E0579 + M3*D149 - M3*E1479 + M4*D159 - M4*E1579 + M3*Z49 -
M3*A479 + M4*Z59 - M4*A579 - M1*G01 - M6*G10 + M6*D160 = 0;
!(2,3); M1*G31 - M2*G32 + M2*D245 - M2*A245 - M3*D234 - M4*D235 = 0;
!(2,4) ; M1*G41 - M2*G42 + M2*B2 - M2*G24 - M2*W2 + M2*Z24 - M3*G24 +
M3*D234 = 0;
!(2,5) ; M1*G51 - M2*G52 + M2*G24 - M2*D245 - M2*Z24 + M2*A245 - M4*G25 +
M4*D235 = 0;
!(2,6); M1*G61 - M2*G62 + M3*G24 - M3*D247 - M3*D249 + M3*E2479 + M4*G25 -
M4*D257 - M4*D259 + M4*E2579 - M5*D268 - M6*D260 = 0;
!(2,7); M1*G71 - M2*G72 + M5*G28 - M5*D268 - M3*D247 - M4*D257 = 0;
!(2,8); M1*G81 - M2*G82 - M5*G28 + M5*D268  + M3*D247 + M4*D257= 0;
!(2,9); M1*G91 - M2*G92 + M6*G20 - M6*D260 - M3*D249 + M3*E2479 - M4*D259
+ M4*E2579 = 0;
!(2,0); M1*G01 - M2*G02 - M6*G20 + M6*D260 + M3*D249 - M3*E2479 + M4*D259
- M4*E2579 = 0;
!(3,4);M2*D425 - M3*G43 - M3*D435 + M2*G32 - M2*D324 = 0;
!(3,5);M2*D524 - M3*D534 - M3*G53 + M2*D324 - M2*E3245 = 0;
!(3,6) ; M2*E6245 - M3*D634 - M4*D635 + M3*G34 - M3*D347 - M3*D349 +
M3*E3479 + M4*G35 - M4*D357 - M4*D359 + M4*E3579 - M3*Z34 + M3*A347 +
M3*A349 - M3*P3479 - M4*Z35 + M4*A357 + M4*A359 - M4*P3579 - M5*D368 -
M6*D360 = 0;
!(3,7) ; M2*E7245 - M3*D734 - M4*D735 - M3*D347 - M4*D357 + M3*A347 +
M4*A357 + M5*G38 - M5*D368 = 0;
!(3,8) ; M2*E8245 - M3*D834 - M4*D835 + M3*D347 + M4*D357 - M3*A347 -
M4*A357  - M5*G38 + M5*D368 = 0;
!(3,9) ; M2*E9245 - M3*D934 - M4*D935 - M3*D349 + M3*E3479 - M4*D359 +
M4*E3579 + M3*A349 - M3*P3479 + M4*A359 - M4*P3579 + M6*G30 - M6*D360 = 0;
!(3,0) ; M2*E0245 - M3*D034 - M4*D035 + M3*D349 - M3*E3479 + M4*D359 -
M4*E3579 - M3*A349 + M3*P3479 - M4*A359 + M4*P3579 - M6*G30 + M6*D360 = 0;
!(4,5);M2*G52 - M2*D524 - M3*G54 + M3*D534 + M2*G42 - M2*D425 - M4*G45 +
M4*D435 = 0;
!(4,6) ; M2*G62 - M2*D624 - M3*G64 + M3*D634 + M3*B4 - M3*G47 - M3*G49 +
M3*D479 - M3*W4 + M3*Z47 + M3*Z49 - M3*A479 + M3*Z34 - M3*A347 - M3*A349 +
M3*P3479 + M4*G45 - M4*D457 - M4*D459 + M4*E4579 - M5*D468 - M6*D460 = 0;
!(4,7) ; M2*G72 - M2*D724 - M3*G74 + M3*D734 - M3*G47 + M3*Z47 - M3*A347 +
M5*G48 - M5*D468 - M4*D457 = 0;
!(4,8) ; M2*G82 - M2*D824 - M3*G84 + M3*D834 + M3*G47 - M3*Z47 + M3*A347 -
M5*G48 + M5*D468 + M4*D457 = 0;
!(4,9) ; M2*G92 - M2*D924 - M3*G94 + M3*D934 - M3*G49 + M3*D479 + M3*Z49 -
M3*A349 - M3*A479 + M3*P3479 + M6*G40 - M6*D460 - M4*D459 + M4*E4579 = 0;
!(4,0) ; M2*G02 - M2*D024 - M3*G04 + M3*D034 + M3*G49 - M3*D479 - M3*Z49 +
M3*A349 + M3*A479 - M3*P3479 + M6*G40 - M6*D460 + M4*D459 - M4*E4579 = 0;
```

```
!(5,6) ; M2*D624 - M2*E6245 - M4*G65 + M4*D635 + M4*B5 - M4*G57 - M4*G59 +
M4*D579 - M4*W5 + M4*Z57 + M4*Z59 - M4*A579 + M4*Z35 - M4*A357 - M4*A359 +
M4*P3579 + M3*G54 - M3*D547 - M3*D549 + M3*E5479 - M5*D568 - M6*D560 = 0;
!(5,7) ; M2*D724 - M2*E7245 - M4*G75 + M4*D735 - M4*G57 + M4*Z57 - M4*A357
+ M5*G58 - M5*D568 - M3*D547 = 0;
!(5,8) ; M2*D824 - M2*E8245 - M4*G85 + M4*D835 + M4*G57 - M4*Z57 + M4*A357
- M5*G58 + M5*D568 + M3*D547 = 0;
!(5,9) ; M2*D924 - M2*E9245 - M4*G95 + M4*D935 - M4*G59 + M4*D579 + M4*Z59
- M4*A359 - M4*A579 + M4*P3579 + M6*G50 - M6*D560 - M3*D549 + M3*E5479 =
0;
!(5,0) ; M2*D024 - M2*E0245 - M4*G05 + M4*D035 + M4*G59 - M4*D579 - M4*Z59
+ M4*A359 + M4*A579 - M4*P3579 - M6*G50 + M6*D560 + M3*D549 - M3*E5479 =
0;
!(6,7);
!(6,8);M3*G84 - M3*D847 - M3*D849 + M3*E8479 + M4*G85 - M4*D857 - M4*D859
+ M4*E8579 - M5*G86 - M6*D860 = 0;
!(6,9);
!(6,0);M3*G04 - M3*D047 - M3*D049 + M3*E0479 + M4*G05 - M4*D057 - M4*D059
+ M4*E0579 - M5*D068 - M6*G06 = 0;
!(7,8) ; M5*B8 - M5*G86 - M5*G78 - M5*W8 + M5*Z68 - M3*D847 - M4*D857 +
M3*G74 + M4*G75 - M3*Z47 - M4*Z57 = 0;
!(7,9);M5*G98 - M3*D947 - M4*D957 + M6*G70 = 0;
!(7,0);M5*G08 - M5*D068 - M3*D047 - M4*D057 - M6*G70 = 0;
!(8,9);M3*D947 + M4*D957 - M5*G98 + M6*G80 - M6*D860 - M3*D849 + M3*E8479
- M4*D859 + M4*E8579 = 0;
!(8,0);M3*D047 + M4*D057 - M5*G08 + M5*D068 - M6*G80 + M6*D860 + M3*D849 -
M3*E8479 + M4*D859 - M4*E8579 = 0;
!(9,0) ; M6*B0 - M6*G06 - M6*G90 - M6*W0 + M6*Z60 - M3*D049 + M3*E0479 -
M4*D059 + M4*E0579 + M3*G94 - M3*D947 + M4*G95 - M4*D957 - M3*Z49 +
M3*A479 - M4*Z59 + M4*A579 = 0;
END
```

REFERENCES

Anderson, W., 1950, Marketing efficiency and the principle of postponement, *Cost and Profit Outlook*, September, p. 3.

Aviv, Y. and Federgruen, A., 2001, Design for postponement: a comprehensive characterization of its benefits under unknown demand distributions, *Operations Research*, Vol. 49, No. 4, pp. 578-598.

Buzacott, J. A. and Shanthikumar, J. G., 1993, *Stochastic Models of Manufacturing Systems*, Prentice Hall, Englewood Cliffs, N.J.

Baynat, B., and Dallery, Y, 1996, A product-form approximation method for general closed queueing networks with several classes of customers, *Performance Evaluation*, Vol. 24, pp. 165-188.

Baynat, B., Dallery, Y., Di Mascolo, M., and Frein, Y., 2001, A multi-class approximation techniques for the analysis of kanban-like control systems, *International Journal of Production Research*, Vol. 39, No. 2, pp. 307-328.

Bonvik, A. M., Couch, C. E., and Gershwin, S. B., 1997, A comparison of production-line control mechanisms, *International Journal of Production Research*, Vol. 35, No. 3, pp. 789-804.

Duenyas, I., 1994, A simple release policy for networks of queues with controllable inputs, *Operations Research*, Vol. 42, No. 6, pp. 1162-1171.

Federgruen, A. 1993, *Recent advances in production and distribution management, Perspectives in Operations Management*, Kluwer academic publisher, Norwell, MA.

Garg, A. and Tang, C. S., 1997, On postponement strategies for product families with

multiple points of differentiation, *IIE Transactions*, Vol. 29, pp. 641-650.

Gupta, S. and Krishan, V., 1998, Product family-based assembly design methodology, *IIE Transactions*, Vol. 30, No. 10, pp. 933-945.

He, D. and Babayan, A., 2002, Scheduling manufacturing systems for delayed product differentiation in agile manufacturing, *International Journal of Production Research*, Vol. 40, No. 11, pp. 2461-2481.

He, D., Kusiak, A., and Tseng, T., 1998, Delayed product differentiation: a design and manufacturing perspective, *Computer-Aided Design*, Vol. 30, No. 2, pp. 105-113

Herer, Y. T. and Masin, M., 1997, Mathematical programming formulation of CONWIP based production lines; and relationship to MRP, *International Journal of Production Research*, Vol. 35, No. 4, pp. 1067-1076 .

Hopp, W. J. and Spearman, M. L., 2000, Factory Physics, McGraw-Hill Higher Education.

Hopp, W. J. and Roof, M. L., 1998, Setting WIP levels with statistical throughput control (STC) in CONWIP production lines, *International Journal of Production Research*, Vol. 36, No. 4, 867-882.

Kelton, D. K., Sadowski, R. P., and Sadowski, D. A., 2001, *Simulation with Arena*, 2nd Edition, McGraw-Hill, Boston, MA.

Kim, H., and Ryan, S. M., 2002, A new analytical evaluation model for kanban controlled systems with fork/join synchronization stations, *2002 IIE Annual Conference Proceedings*, Orlando, FL.

Kim, H., and Ryan, S. M., 2003, Analysis of delayed product differentiation under a CONWIP policy, *2003 IIE Annual Conference Proceedings*, Portland, OR, In preparation.

Krishnamurthy, A., Suri. R., and Vernon, M., 2001, Analytical Performance Analysis of Kanban Systems Using a New Approximation for Fork/Join Stations, *2001 IIE Annual Conference Proceedings*, Dallas, TX.

Krishnamurthy, A., Suri, R. and Vernon, M. K., 2000, Push Can Perform Better than Pull for Flexible Manufacturing Systems with Multiple Products, *Proc. Industrial Engineering Research Conference*, Cleveland, OH.

Kumar, S. and Kumar, P. R., 1994, Performance Bounds for Queueing Networks and Scheduling Policies, *IEEE Transactions on Automatic Control*, Vol. 39, No. 8, pp. 1600-1611.

Lee, H. L. 1996, Effective Inventory and Service Management Through Product and Process Redesign, *Operations Research*, Vol. 44, No.v1, pp. 151-159.

Lee, H. L. and Tang, C. S., 1997, Modeling the Costs and Benefits of Delayed Product Differentiation, *Management Science*, Vol. 43, No. 1, pp. 40-53.

Lee, H. L. and Tang, C. S., 1998, Variability Reduction Through Operations Reversal, *Management Science*, Vol. 44, No. 2, pp. 162-172.

LINDO Systems Inc., 1996, *Solver suite: LINDO, LINGO, and What's Best!*, LINDO Systems Inc., Chicago, IL.

Ma, S., Wang, W. and Liu, L., 2002, Commonality and postponement in multistage assembly systems, *European Journal of Operations Research*, Vol. 142, pp. 523-538.

Monden, Y., 1983, *Toyota Production System: Practical approach to management*, Industrial Engineering and Management Press, Norcross, GA.

Muckstadt, J. A. and Tayur, S. R., 1995, A comparison of alternative kanban control mechanisms. I. Background and structural results, *IIE Transactions*, Vol. 27, pp.

140-150.

Muckstadt, J. A. and Tayur, S. R., 1995, A comparison of alternative kanban control mechanisms. II. Experimental results, *IIE Transactions*, Vol. 27, pp. 140-150.

Ryan, S. M. and Choobineh, F. F., 2002, Total WIP and WIP mix for a CONWIP controlled job shop, *IIE Transactions*, Accepted.

Ryan, S. M., Baynat, B. and Choobineh, F. F., 2000, Determining inventory levels in a CONWIP controlled job shop, *IIE Transactions*, Vol. 32, 105-114.

Schraner, E. and Hausman, W. H., 1997, Optimal production operations sequencing, *IIE Transactions*, Vol. 29, pp. 651-669.

Schweitzer, P. J., Seidmann, A. and Shalev-Oren, S. (1986), The correction terms in approximate mean value analysis, *Operations Research Letters*, Vol. 4, No. 5, 197-200.

Silver, E. A. and Peterson, R., 1985, *Design systems for inventory management and production planning*, Wiley, New York, NY.

Signorelli, S., Heskett, J. L., 1989, *Benetton (A)*, Harvard Business School.

Spearman, M., Woodruff, D. and Hopp, W., 1990, CONWIP: a pull alternative to kanban, *International Journal of Production Research*, Vol. 28, No. 5, pp. 879-894.

Spearman, M., and Zazanis, M. 1992, Push and pull production systems: issues and comparisons, *Operations Research*, Vol. 40, pp. 521-532.

Suri, R., 1998, *Quick response manufacturing: a companywide approach to reducing lead time*, Productivity Press, Portland, OR.

Suri, R. and Hilderbrant, R. R., 1984, Modelling Flexible Manufacturing Systems Using Mean-Value Analysis, *Journal of Manufacturing Systems*, Vol. 3, No. 1, pp. 27-38.

Swaminathan, J. M. and Tayur, S. R., 1999, Managing design of assembly sequences for product line differentiation, *IIE Transactions*, Vol. 31, pp. 1015-1026.

Swaminathan, J. M. and Tayur, S. R., 1998, Managing broader product lines through delayed differentiation using vanilla boxes, *Management Science*, Vol. 44 No. 2, pp. S162-S172.

Tayur, S. R., 1992, Properties of serial kanban systems, Queueing Systems, Vol. 12, pp. 297-318.

Van Hoek, R. I., 2001, The rediscovery of postponement; a literature review and directions for research, *Journal of Operations Management*, Vol. 19, No. 2, pp. 161-184.