ESTIMATING THE THROUGHPUT OF AN EXPONENTIAL CONWIP ASSEMBLY SYSTEM

IZAK DUENYAS
Department of Industrial & Operations Engineering
University of Michigan
Ann Arbor, MI 48109-2117

WALLACE J. HOPP
Department of Industrial Engineering & Management Sciences
Northwestern University
Evanston, IL 60208

Technical Report 92-36
May 1992

ESTIMATING THE THROUGHPUT OF AN EXPONENTIAL CONWIP ASSEMBLY SYSTEM

IZAK DUENYAS

Department of Industrial and Operations Engineering

The University of Michigan

Ann Arbor, Michigan 48109

WALLACE J. HOPP

Department of Industrial Engineering and Management Sciences

Northwestern University

Evanston Illinois 60208

Abstract

We consider a production system consisting of several fabrication lines feeding an assembly station where both fabrication and assembly lines consist of multiple machine exponential workstations and the CONWIP (CONstant Work-In-Process) mechanism is used to regulate work releases. We model this system as an assembly-like queue and develop approximations for the throughput and average number of jobs in queue. These approximations use an estimate of the time that jobs from each line spend waiting for jobs from other lines before being assembled. We use our approximations to gain insight into the related problems of capacity allocation, bottleneck placement and WIP setting.

Keywords: Assembly-like queues, approximations, throughput, design issues.

1 Introduction

Assembly-like queues arise in a variety of practical situations, primarily in manufacturing systems but also in models of data flow through computer systems (Dennis [9]). Despite the enormous number of practical applications of these systems, little work has been done on these queues due to their analytical intractability. The vast majority of queueing network models (e.g., QNA, by Whitt [32]) do not handle assemblies. Those that do consider assemblies model the lines feeding assembly as single machines, an assumption that severely limits their applicability to most manufacturing systems (Ammar [1], Bhat [4], Bonomi [6], Hopp and Simon [16], Lipper and Sengupta [18]). Analysis of more realistic systems has been limited to simulation studies. For instance, Baker et al. ([2], [3]) have used simulation to analyze the behavior of assembly systems and allocate work optimally in these systems. While valuable, simulation can be tedious for optimization purposes (e.g., for setting optimal WIP levels or lead times).

An issue that complicates the modeling of assembly systems is that assembly-like queues are unstable unless some feedback mechanism is used to link releases to outputs (Harrison [14]). In manufacturing systems, the most common method for controlling releases is MRP. Recently, in the wake of the success of Japanese firms, pull systems, such as kanban, have gained popularity (Monden [19], Ohno [20]). However, while some analytic models of pull systems have been developed (e.g., Duenyas et al. [10], Bitran and Chang [5], Wang and Wang [31]), design and control of such systems remains as much an art as a science. The goal of this paper is to expand the analytic capability for treating pull systems by addressing the issue of assemblies.

A particular pull system that has the dual advantage of being more broadly applicable and analytically tractable than kanban is CONWIP (CONstant Work-In-Process) (Spearman et al. [28], [29]). In a CONWIP system the total amount of work is held constant by authorizing production of a new unit only when an output occurs. This can be accomplished with cards (kanbans), such that all jobs must have cards attached to them. Each time a

job is completed, its card is removed and sent to the front of the line to authorize the start of another job. Alternatively, electronic signals can be used in place of cards, so that the WIP level in the line is monitored and a new job is started whenever the WIP level falls below a specified level. Note that under the CONWIP mechanism only the first machine in a production line is governed by the pull mechanism; jobs are pushed between machines elsewhere in the line and all inter-machine buffers are assumed infinite.

In an assembly system operating under CONWIP, which is illustrated in Figure 1, completion of an assembly sends a card (electronic signal) to the front of each fabrication line authorizing the start of a new job. Because the number of cards (WIP levels) in each fabrication may differ, the jobs authorized to start at the time an assembly is completed may not be destined for the same assembly. However, because jobs are only started when assemblies are completed, CONWIP serves to pace the fabrication lines to assembly. Furthermore, because it prevents "WIP explosions," CONWIP achieves many of the benefits claimed for kanban (e.g., short cycle times and predictable behavior) (see Spearman and Zazanis [30]).

In this paper, we have chosen to focus on CONWIP because it is a practical release mechanism, facilitates analysis, and provides a first step toward rationalizing the design and analysis of pull assembly systems. Our results are directly applicable to the problems of the optimal allocation of capacity and WIP in a CONWIP system. The general qualitative insights may extend to assembly systems operating under other control policies (e.g., kanban). A comparative study of alternate control methods for assembly systems is an attractive area for further research, but is beyond the scope of this paper.

Since this paper represents a first-cut at the problem, we will frequently make the simplifying assumption that the processing times are exponentially distributed. In fact, the exponential distribution has been argued to be a good approximation in some real-world production systems (Solberg [25], Solberg and Nof [26]). However, in recognition of the fact that this exponential assumption is not valid in all production systems, we examine the effect of different (stochastically ordered) distributions on the throughput and optimal

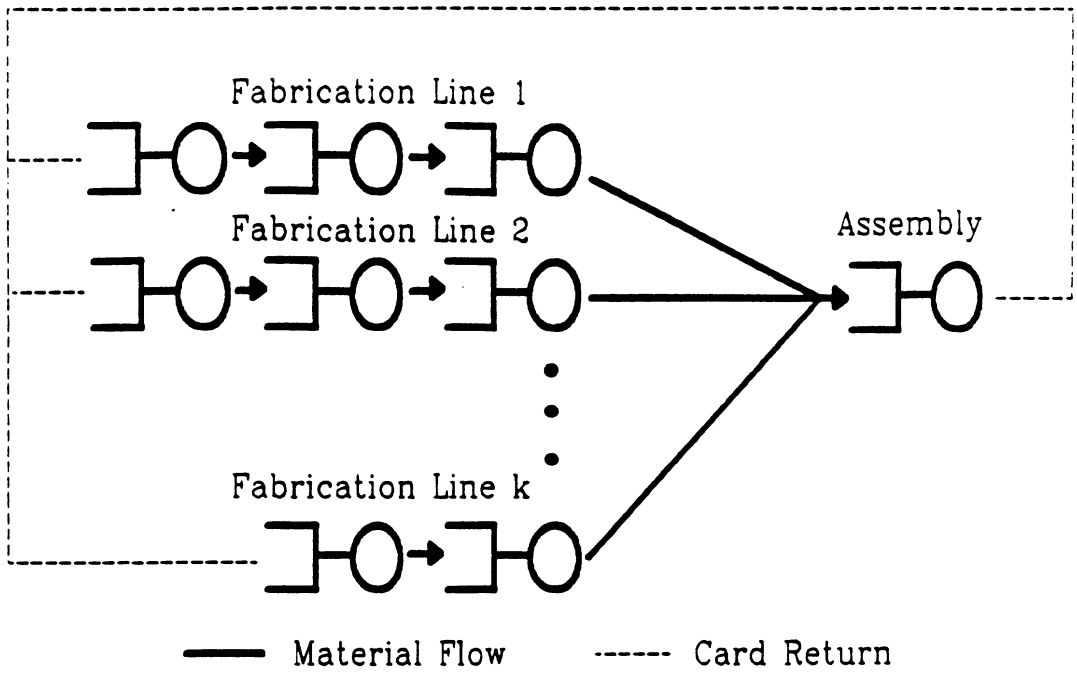

Figure 1: CONWIP Assembly System

card counts. These results provide insights for extending our analysis to non-exponential assembly systems.

The remainder of this paper is organized as follows. Section 2 introduces our notation and problem formulation. In Section 3, we derive an upper bound for throughput in an assembly system. We use this upper bound to derive an approximation for the throughput and show how this yields an approximation for the average number in queue at assembly and fabrication machines in Section 4. In Section 5, we test our approximations against several different test problems. In Section 6, we show how our throughput approximation can be used to address two design problems.

2 Problem Formulation

We consider k production lines with m_j exponential machines (servers) and n_j jobs (customers) at each line j as shown in Figure 1. At each line j, jobs start at machine $(j, 1)$ and after being served at machine (j, i) move to machine $(j, i+1)$. Service times at machine

(j, i) are independent and exponentially distributed with mean $\lambda_{(j,i)}^{-1}$. For the purposes of this paper, we will consider all jobs to be identical. In practice, however, we have used this type of single product model for a multi-product system in which all products share common routings and a distinct (mix-independent) bottleneck exists. In this case, we can "standardize" the products according to their processing times at the bottleneck (e.g., a job that requires twice the standard amount of processing time at the bottleneck is counted as two units of WIP). While this is clearly an approximation, our experience has shown it to be a useful one for rough-cut analysis.

After completing work at machine (j, m_j), a job joins the assembly queue. If there is at least one job from each line in the assembly queue, then the assembly operation begins. Assembly times are independent and exponentially distributed with mean λ_A^{-1}. An output occurs when service at the assembly is completed. Under the CONWIP protocol, this sends a signal to machines $(j, 1)$, $j = 1...k$, to add a new job to their queues. We begin observing the output process at time $t = 0$ and define N_t as the number of outputs until time t. We are interested in finding the throughput for the system, $\theta = \lim_{t\to\infty} N_t/t$. By Little's law, the problem is equivalent to finding the average cycle time (flowtime, round-trip time) for any one of the lines.

If $n_j = 1$ for each line j, the times between outputs are i.i.d. For this special, but not very realistic, case the problem is greatly simplified. We let $Y_{(j,i)}$ denote the (random) service time of a job at machine i in line j. Since every time a unit is completed at the assembly machine, work is begun on machine 1 of each line on a new unit and the assembly machine can begin work only when the jobs from each line have reached it, the expected cycle time is the sum of the assembly time plus the maximum expected time to go through each line. Hence, the expected cycle time can be expressed as

$$E[\max_j \sum_{i=1}^{m_j} Y_{(j,i)}] + \lambda_A^{-1} \tag{2.1}$$

and we are left only with the cumbersome, but tractable, task of calculating the expected value of the maximum of sums of exponentials.

For values of n_j greater than 1, the interoutput times are not i.i.d., nor are the cycle times. For those cases, one way of getting an exact solution for the throughput is by solving the underlying Markov chain. However, for any realistically sized problem, that approach is impractical due to the enormous number of states in the Markov chain. Computational difficulties will be particularly acute if the decision maker wishes to use the throughput expression in an algorithm for optimizing WIP levels. To provide a practical alternative, we develop an approximation for the throughput.

3 An Upper Bound for Throughput

Let $\{F/F_A/n\}$ denote the assembly system shown in Figure 1. Successive service times of machine i in line j are iid random variables with cumulative distribution function (cdf) F_{ji}, and F is a two-dimensional array containing the cdf's F_{ji}. Service times at the assembly are iid random variables with cdf F_A. There are n_j jobs at each line j, and n is an array containing the n_j values. Let $\theta\{F/F_A/n\}$ denote the steady-state average throughput of $\{F/F_A/n\}$. We use $\leq^{st}$ to mean "stochastically less than," and $\leq^{icx}$ as "increasing convex ordering" as defined in [22]. When we write $\leq^{st}$ ($\leq^{icx}$) we mean that the proposition holds under either ordering.

Proposition 1 *Suppose F and $\hat{F}$ are two arrays of cdf's such that either*

(a)

$$\begin{array}{lcll} F_A & = & \hat{F}_A & \\ F_{ji} & \leq^{st} (\leq^{icx}) & \hat{F}_{ji} & \textit{for } (j,i) = (j',i') \\ F_{ji} & = & \hat{F}_{ji} & \textit{for } (j,i) \neq (j',i') \end{array}$$

for a given (j',i'), or

(b)

$$F_A \leq^{st} (\leq^{icx}) \hat{F}_A$$

$$F_{ji} = \hat{F}_{ji} \quad \textit{for all } (j,i)$$

Then, $\theta\{F/F_A/n\} \geq \theta\{\hat{F}/\hat{F}_A/n\}$

Proof: The proof is similiar to that of Lemma 1 in Hopp and Simon [16] and is omitted.

Now, let $\{F_r/F_A/n_r\}$ denote a closed tandem queueing network that consists of machines $(r,1),...,(r,m_r)$ in sequence with the assembly machine at the end (where the assembly machine does not "assemble" but, instead, processes single jobs with processing times distributed as F_A. In this case, jobs start at machine $(r,1)$, move to $(r,i+1)$ after (r,i) and to the assembly machine after (r,m_r). After completing work at the assembly, jobs return to machine $(r,1)$. Let $\theta\{F_r/F_A/n_r\}$ denote the throughput of this system. Then, we have

Corollary 1

$$\theta\{F/F_A/n\} \leq \min_r \theta\{F_r/F_A/n_r\} \tag{3.2}$$

Proof: Fix r. Consider the system $\{\hat{F}/F_A/n\}$, where $\hat{F}$ is the same as F except for $\hat{F}[j',i'] = I$, $j' \neq r$, where I is the unit step function at zero. Clearly, since $F \leq^{st} I$ for any cdf F of a positive random variable, by Proposition 1 we have $\theta\{F/F_A/n\} \leq \theta\{\hat{F}/F_A/n\}$. We can apply Proposition 1 in this way repeatedly by replacing machines in all the lines except line r with machines that have processing time cdfs I. This leads to the system $\{F_r/F_A/n_r\}$, so $\theta\{F/F_A/n\} \leq \theta\{F_r/F_A/n_r\}$ and thus the corollary follows. □

Notice that neither Proposition 1 nor Corollary 1 require exponential processing times. We let $\{\lambda/\lambda_A/n\}$ represent the system in Figure 1 with exponential distributions for processing times where λ is an array containing the processing rates. Then from Corollary 1, we have

$$\theta\{\lambda/\lambda_A/n\} \leq \min_r \theta\{\lambda_r/\lambda_A/n_r\} \tag{3.3}$$

Let $U = \min_r \theta\{\lambda_r/\lambda_A/n_r\}$. By Corollary 1, U is an upper bound on $\theta\{\lambda/\lambda_A/n\}$. Since all processing times are exponential, it is straightforward to compute U by mean-value analysis [21].

In addition to providing the basis for an upper bound on the throughput of an exponential assembly system, Proposition 1 provides useful qualitative results. First, it states that speeding up any machine (in the sense of replacing its processing time distribution with a stochastically smaller distribution) causes throughput to increase. Second, the result for the $\leq^{icx}$ ordering implies that the throughput of the assembly system where all the machines have exponential distributions is a lower bound on the throughput of a system with machines that have IFR distributions and the same mean processing times [27].

These results could eliminate unnecessary simulations from consideration. For instance, suppose we have simulated the assembly system with a given machine assigned normal processing times. Since decreasing the mean of a normal distribution results in a stochastically smaller distribution, we know that the original simulated throughput is a lower bound on the throughput that will result if the mean processing time of the machine is reduced.

We next present an approximation for the throughput that utilizes the upper bound, U. Our approach will also lead us to an approximation of the average number in queue at assembly and at each fabrication machine.

4 Approximations for Throughput and Queue Lengths

A job in line r is delayed at assembly if, when it is that particular job's turn to be served by the assembly machine, there is not at least one job from all other lines. When this occurs, the job in line r has to wait for the other jobs to arrive. Our approximation is based upon estimating the expected waiting time that this job experiences. We start by developing our approximation for an assembly system with two lines, and then generalize it to more than two lines.

Consider an assembly system with two lines. Let W_1 be the amount of time that a job

from line 1 has to wait at assembly for a job from line 2. To calculate EW_1, we condition on the position of the jobs in line 2. Let N_i be the number of jobs in line 2, machine i. Then we have

$$EW_1 = \sum_{i=1}^{m_2+1} E[W_1|N_i > 0, \sum_{p=i+1}^{m_2+1} N_p = 0]P(N_i > 0, \sum_{p=i+1}^{m_2+1} N_p = 0) \tag{4.1}$$

For example, if $N_{m_2+1} > 0$, then this means that there is a job from line 2 already waiting at assembly, and hence the expected waiting time is 0. In general, we have

$$E[W_1|N_i > 0, \sum_{p=i+1}^{m_2+1} N_p = 0] = \sum_{p=i}^{m_2} \lambda_p^{-1} \tag{4.2}$$

The calculation of the probabilities in (4.1) is not as easy, however, since we do not know the distribution of jobs in the network. Hence, we approximate these by supposing that, while jobs in line 1 have to wait for jobs in line 2 for their assembly operation, jobs in line 2 are independent of jobs in line 1 and start their assembly operation regardless of whether or not there are jobs from line 1 in assembly. This makes line 2 a regular closed queueing network and we can easily calculate the probabilities in (4.1) using Buzen's coefficents as follows:

$$P[N_i > 0, \sum_{p=i+1}^{m_2+1} N_p = 0] = \frac{G(n_2, i) - G(n_2, i-1)}{G(n_2, m_2+1)} \tag{4.3}$$

where

$$G(n,m) = \sum_{(N_1,.....N_m)\in Z(n,m)} \prod_{i=1}^{m} \lambda_i^{-N_i}$$

$$Z(n,m) = \{(N_1, ..., N_m) : N_i \geq 0, \sum_{i=1}^{m} N_i = n\}$$

Since it is difficult to characterize the convolution of the distribution of the time a job from line 1 spends waiting for a job from line 2 and its processing time distribution, we make the further approximation that it takes an exponentially distributed time with mean $\lambda_A^{-1} + EW_1$. We let

$$\varphi_{Ai} = 1/(\lambda_A^{-1} + EW_i) \tag{4.4}$$

and describe the new network by $\{\lambda_1/\varphi_{A1}/n_1\}$. Notice, however, that just as jobs from line 1 wait for jobs from line 2, the reverse is true as well. Hence, to capture the effect that both lines have on each other, we propose starting with $\{\lambda_1/\lambda_A/n_1\}$, calculating EW_2 and using $\{\lambda_2/\varphi_{A2}/n_2\}$ to calculate EW_1 and continuing in this manner until the throughput converges. The final issue that we have to resolve here is the choice of line 1 and line 2. We let line 1 be the line that sets U, that is let $h = argmin_r \theta\{\lambda_r/\lambda_A/n_r\}$. We renumber line h as line 1 since that line is actually "closest" to the throughput of the network. We can now present a

Procedure for computing θ_{ap} (2 lines)

1. *Let $h = argmin_r\ \theta\{\lambda_r/\lambda_A/n_r\}$. Renumber line h as line 1. Let $\theta_1 = min\theta_r\{\lambda_r/\lambda_A/n_r\}$. Let $\varphi_{A1} = \lambda_A$.*

2. *Compute EW_2 using (4.1) and $\{\lambda_1/\varphi_{A1}/n_1\}$. Compute φ_{A2} using (4.4).*

3. *Compute EW_1 using (4.1) and $\{\lambda_2/\varphi_{A2}/n_2\}$. Compute φ_{A1} using (4.4).*

4. *$\theta_{ap} = \theta\{\lambda_1/\varphi_{A1}/n_1\}$. If $|\theta_{ap} - \theta_1| < \delta$ for some prespecified δ then stop. Else, let $\theta_1 = \theta_{ap}$, goto 2.*

Proposition : *The above procedure will converge to a finite value.*

Proof: It is enough to show that θ_{ap} decreases at each iteration of the procedure and that $\theta_{ap} > 0$. Denote by EW_2^i and EW_1^i, respectively the EW_1 and EW_2 value generated by the algorithm in the ith iteration. We start with $EW_1 = 0$ in the first iteration in step 1, and compute EW_2^1. Next, we compute EW_1^1 in step 3. We note that the higher EW_2^i, the lower EW_1^{i+1} generated in step 2, since the more we add to the processing time of the assembly station, the more likely it is that jobs will be in front of the assembly station and hence the less the expected waiting time for a job there. Using this argument recursively, $EW_1^{i+1} \geq EW_1^i$ and therefore θ_{ap} is decreasing in each step. To show that θ_{ap} does not decrease to 0, we note that $\theta_{ap} = 0$ implies $EW_1 = \infty$, but that would in turn imply

$EW_2 = 0$, which in the next step would generate a finite EW_1 value. Hence, since θ_{ap} is decreasing in each iteration and the final θ_{ap} is not 0, it is converging to a positive value. □.

We can use a similiar approach to derive an approximation for assembly systems with more than two lines. In this case, if there are k lines in the system, all k lines should have one job at the assembly machine for the assembly operation to begin. We again make the assumption that lines $2, \ldots, k$ are independent closed queueing networks to calculate EW_1. To illustrate the nature of the calculations involved, consider an example consisting of a 3-line assembly system with 2 machines in line 2 and 1 machine in line 3 feeding into the assembly machine. (Since we are calculating how long jobs from line 1 wait at assembly, the number of jobs in line 1 does not matter). We let N_{ji} denote the number of jobs in line j machine i, and let Y_{ji} denote the (random) processing time of a job in line j machine i. Then,

$$\begin{aligned}
E[W_1] = {} & E[W_1|N_{23} > 0, N_{32} > 0]P(N_{23} > 0, N_{32} > 0) + E[W_1|N_{22} > 0, N_{23} = 0, N_{32} > 0] \\
& P(N_{22} > 0, N_{23} = 0, N_{32} > 0) + E[W_1|N_{22} + N_{23} = 0, N_{32} > 0] \\
& P(N_{22} + N_{23} = 0, N_{32} > 0) + E[W_1|N_{23} > 0, N_{32} = 0]P(N_{23} > 0, N_{32} = 0) + \\
& E[W_1|N_{22} > 0, N_{23} = 0, N_{32} = 0]P(N_{22} > 0, N_{23} = 0, N_{32} = 0) + \\
& E[W_1|N_{22} + N_{23} = 0, N_{32} = 0]P(N_{22} + N_{23} = 0, N_{32} = 0)
\end{aligned} \tag{4.5}$$

Calculating the first five conditional expectations in (4.5) is straightforward and yields:

$$\begin{aligned}
& E[W_1|N_{23} > 0, N_{32} > 0] = 0 \\
& E[W_1|N_{22} > 0, N_{23} = 0, N_{32} > 0] = 1/\lambda_{22} \\
& E[W_1|N_{22} + N_{23} = 0, N_{32} > 0] = 1/\lambda_{21} + 1/\lambda_{22} \\
& E[W_1|N_{23} > 0, N_{32} = 0] = 1/\lambda_{31} \\
& E[W_1|N_{22} > 0, N_{23} = 0, N_{32} = 0] = \tfrac{\lambda_{22}+\lambda_{31}}{\lambda_{22}\lambda_{31}} - \tfrac{1}{\lambda_{22}+\lambda_{31}}
\end{aligned}$$

The fifth expectation term involves the maximum of two exponentials. The sixth term however, requires computing the expectation of the maximum of sums of exponentials; that

is

$$E[W_1|N_{22}+N_{23}=0,N_{32}=0]=E[max(Y_{21}+Y_{22},Y_{31})] \tag{4.6}$$

While (4.6) is not difficult to compute, the expectation of the maximum of sums of exponentials in a larger network becomes more complicated. In general, we would be faced with terms like $E[\max_{j\in\{2,\ldots,k\}}(\sum_{i=b_j}^{m_j} Y_{ji})]$, where b_j represents the location of the job closest to assembly in fabrication line j. If k and $m_j, j=2,\ldots,k$ are large, this could be very tedious. For this reason, we approximate these expectations by replacing sums of exponentials with an exponential that has the equivalent mean. For example, in (4.6), we replace $Y_{21}+Y_{22}$ by an exponentially distributed random variable Z with mean $(\lambda_{21}+\lambda_{22})/\lambda_{21}\lambda_{22}$. Calculating the probabilities in (4.5) is the same as in (4.3), but since there are $k-1$ closed queueing networks in this case, we calculate the probability for each one and multiply. We can now state the

Procedure for computing θ_{ap} (k lines)

1. *Let $h = argmin_r\ \theta\{\lambda_r/\lambda_A/n_r\}$. Renumber line h as line 1. Let $\theta_1 = \min_r \theta\{\lambda_r/\lambda_A/n_r\}$. Renumber the other lines from $2,\ldots,k$ arbitrarily. For $i=2,..,k$ let $\varphi_{Ai}=\lambda_A$.*

2. *For $i=2,\ldots,k$ compute EW_i using $\{\lambda_r/\varphi_{Ar}/n_r\}$ $\quad r=1,\ldots,k, r\neq i$.*

3. *Update φ_{Ai} for $i=2,\ldots,k$ using (4.4).*

4. *Compute EW_1 using $\{\lambda_r/\varphi_{Ar}/n_r\}$ $\quad r=2,\ldots,k$*

5. *Update φ_{A1} using (4.4)*

6. *$\theta_{ap}=\theta\{\lambda_1/\varphi_{A1}/n_1\}$. If $|\theta_{ap}-\theta_1|<\delta$ then stop. Else, $\theta_1=\theta_{ap}$. Goto 2.*

We can use θ_{ap} to get an approximation for the cycle time in each line. Letting c_i denote the cycle time of a job in Line 1, by using Little's Law, we have

$$c_i=\frac{n_i}{\theta_{ap}} \tag{4.7}$$

We can also use the above procedure to estimate the mean number of jobs at each station. To do this, note that we are approximating each line r by the closed queueing network $\{\lambda_r/\varphi_{Ar}/n_r\}$ generated in the last iteration of the above algorithm. Hence, we can obtain approximate values for the mean number of jobs in each line using standard results for closed queueing networks and the closed queueing networks generated in the last iteration of the procedure.

The procedure we have outlined above can easily be generalized to problems with multiple servers. In this case, we have k production lines with m_j stations. Each station (j, i) has $z_{(j,i)}$ identical machines. Since we can still calculate the steady-state distributions of jobs in closed queueing networks with multiple servers [7], we can use a procedure entirely analogous to that above to treat the multi-machine workstation case.

5 Computational Results

The real test of any approximation is how well it works over a range of cases. To test the method described above, we generated a variety of problems with 2 and 3 lines and compared the throughput of the system from simulation, θ_s, with our approximation results. The cases summarized in Table 1 are representative of the range of scenarios that could be observed in practice. These include cases with balanced and unbalanced fabrication lines, fast and slow assembly operations, and single and multiple machines. For each case, we examine the accuracy of the throughput approximation for a variety of WIP allocations.

In Example 1, we consider a perfectly balanced system with two lines with 4 workstations in each line. The workstations and the assembly operation have single machines with mean processing times of 2. We made use of a MOR-DS [8] program to simulate this system for different WIP allocations. For each case, we simulated the system for 6000 time units 20 times and we recorded the throughput each time. The average of these values gave us θ_s. Each simulation run (20 values) lasted about 10 minutes on a 386 machine, while the computation involved in our approximation took negligible time (less than 1 second for each

Example	Number of Fabrication Lines	Location of Bottleneck	Single or Multiple Machine Workstations
1	2	Balanced	Single
2-4	2	Assembly	Single
5-9	2	Fabrication	Single
10	2	Fabrication	Multiple
11	3	Fabrication	Single

Table 1: Description of Examples

of the examples considered.) The simulation results and our approximation are reported in Table 2. We report both the value of θ_{ap} computed to an accuracy of 0.001 and also the value of θ_1, the value of the approximation after a single iteration. As shown in Table 1, the value of θ_1 and the value of θ_{ap} were very close in all the cases; in fact it rarely took more than 4 iterations to obtain an accuracy level of 0.001. As observed in Table 1, the approximation θ_{ap} behaved very well and the accuracy was within 2%.

We also tested our approximation of mean number of jobs at each station. Since all fabrication machines are identical, the mean queue length at each fabrication machine is the same. We denote the value of the mean number in front of assembly and all fabrication machines, obtained by simulation, as n_s^a and n_s^f, and similarly those obtained by approximation as n_{ap}^a and n_{ap}^f. Note that it is more difficult to estimate the mean number in front of assembly since a small error in estimating the mean number of jobs in each of the fabrication machines will lead to a large error in estimating mean number in front of assembly. Hence, it is not surprising that the queue length approximation performed better for fabrication than assembly. Despite this, the results were very good as it can be seen in Table 1.

In Examples 2 through 4, we considered systems where the assembly machine was the

n_1	n_2	θ_s	θ_{ap}	%err	θ_1	n_s^a	n_{ap}^a	n_s^f	n_{ap}^f
2	2	0.144	0.140	-2.7	0.142	0.97	1.00	0.26	0.25
3	3	0.189	0.187	-1.0	0.190	0.68	0.71	0.58	0.57
4	4	0.226	0.222	-1.8	0.228	1.31	1.29	0.67	0.67
5	5	0.254	0.252	-0.8	0.258	1.57	1.58	0.86	0.86
6	6	0.277	0.275	-0.8	0.282	1.87	1.86	1.03	1.03
10	10	0.338	0.338	0.0	0.344	2.91	2.97	1.77	1.76
12	12	0.357	0.358	0.2	0.363	3.63	3.52	2.09	2.12
2	6	0.166	0.164	-1.2	0.164	0.40	0.43	0.40	0.39

Table 2: Results for Example 1

bottleneck. For the same fabrication lines, we varied the mean processing time at the assembly machine to test the effect on the approximation. In each of the examples, we considered a system with 2 lines. Each line had 3 stations and each workstation had a single machine with mean processing times of 2. In Example 2, 3 and 4 the assembly machine had a mean processing time of 3, 4, and 6, respectively. The simulation estimates and our approximations for the various WIP allocations are shown in Tables 3, 4 and 5. The throughput approximation erred both high and low but consistently gave results within 4% of the simulation value. The queue length approximations also erred high and low. Both approximations became better as the assembly machine became a more distinct bottleneck, and in Example 4, there was barely any difference between the approximated and simulated throughputs. The value of the approximation after 1 iteration was almost the same as the value of the approximation to an accuracy of 0.001.

In Examples 5 through 7, we considered systems where the assembly machine was faster than any of the other machines. That is, we varied the processing time at assembly in the opposite direction of Examples 2 through 4. The fabrication machines were the same as in Examples 2 through 4. The mean processing times at assembly were 1.5, 0.9, and 0.1

n_1	n_2	θ_s	θ_{ap}	%err	θ_1	n_s^a	n_{ap}^a	n_s^f	n_{ap}^f
2	2	0.156	0.153	-1.9	0.154	0.90	0.90	0.37	0.37
3	3	0.198	0.197	-0.5	0.198	1.39	1.36	0.54	0.55
5	5	0.251	0.252	0.3	0.253	2.37	2.37	0.88	0.88
2	4	0.175	0.169	-3.4	0.170	0.72	0.77	0.43	0.41
3	5	0.215	0.211	-1.9	0.211	1.17	1.21	0.61	0.60
4	6	0.243	0.240	-1.2	0.241	1.61	1.69	0.80	0.77
7	8	0.291	0.287	-1.4	0.287	3.37	3.43	1.21	1.19

Table 3: Results for Example 2

n_1	n_2	θ_s	θ_{ap}	%err	θ_1	n_s^a	n_{ap}^a	n_s^f	n_{ap}^f
2	2	0.138	0.139	0.7	0.139	0.95	1.00	0.35	0.33
3	3	0.174	0.176	1.1	0.176	1.59	1.58	0.47	0.47
5	5	0.215	0.217	0.9	0.217	2.91	2.89	0.70	0.70
2	4	0.154	0.152	-1.3	0.152	0.38	0.36	0.54	0.55
3	5	0.186	0.186	0.0	0.186	1.45	1.47	0.52	0.51
4	6	0.243	0.240	-1.2	0.241	1.61	1.69	0.80	0.77

Table 4: Results for Example 3

n_1	n_2	θ_s	θ_{ap}	%err	θ_1	n_s^a	n_{ap}^a	n_s^f	n_{ap}^f
2	2	0.116	0.116	0.0	0.116	1.20	1.19	0.27	0.27
3	3	0.140	0.140	0.0	0.141	1.95	1.91	0.35	0.36
5	5	0.161	0.161	0.0	0.161	3.61	3.62	0.46	0.46
2	4	0.125	0.124	-0.8	0.124	1.13	1.20	0.29	0.27
3	5	0.148	0.145	-2.0	0.145	1.84	1.87	0.39	0.38
4	6	0.157	0.156	-0.6	0.156	2.70	2.71	0.43	0.43
7	8	0.163	0.162	-0.7	0.162	5.53	5.52	0.49	0.49

Table 5: Results for Example 4

respectively for Examples 5, 6 and 7. The approximation had a slightly harder time in these cases, but the difference between the approximation and the simulation results for throughput were still within 4%. Approximations of queue lengths were also somewhat worse than those in Examples 2 through 4. We can explain this degradation as follows. The amount of time that a job spends at assembly has two components, the (known) processing time and the (estimated) waiting time. The reason that the approximations seem to get worse as the mean processing time at assembly decreases seems to be due to the fact that the estimated part becomes a larger portion of the total time, and hence the accuracy of the approximations slightly worsens.

In order to observe the effects of having unbalanced lines, and lines with different numbers of machines on our approximation, we took the system in Example 7 and decreased the number of machines in line 2. In Example 8, line 1 had 3 machines and line 2 had 2 machines with mean processing time of 2. The mean processing time at the assembly machine was 0.1. Example 9 was the same as Example 8 except that line 2 had only 1 machine with mean processing time of 2. The results are displayed in Tables 9 and 10. Unbalancing the lines resulted in the throughput approximation working very well with the highest error less than 3%. The queue length approximation also worked well in the unbalanced case.

n_1	n_2	θ_s	θ_{ap}	%err	θ_1	n_s^a	n_{ap}^a	n_s^f	n_{ap}^f
2	2	0.179	0.175	-2.2	0.181	0.70	0.72	0.43	0.43
3	3	0.233	0.229	-1.7	0.237	1.07	1.00	0.64	0.67
5	5	0.300	0.300	0.0	0.310	1.53	1.52	1.16	1.16
2	4	0.206	0.199	-3.4	0.201	0.44	0.51	0.52	0.50
3	5	0.257	0.249	-3.1	0.253	0.67	0.75	0.78	0.75
4	6	0.290	0.286	-1.4	0.290	1.00	0.99	1.00	1.00
7	8	0.349	0.350	0.3	0.357	1.83	1.74	1.72	1.75

Table 6: Results for Example 5

n_1	n_2	θ_s	θ_{ap}	%err	θ_1	n_s^a	n_{ap}^a	n_s^f	n_{ap}^f
2	2	0.190	0.186	-2.1	0.193	0.59	0.64	0.47	0.45
2	3	0.212	0.203	-4.2	0.207	0.39	0.50	0.54	0.50
3	3	0.244	0.242	-0.8	0.253	0.82	0.85	0.73	0.72
5	5	0.320	0.315	-1.6	0.329	1.26	1.20	1.25	1.27
3	5	0.269	0.265	-1.5	0.269	0.48	0.56	0.84	0.81
3	4	0.264	0.257	-2.6	0.263	0.62	0.56	0.79	0.81
5	7	0.332	0.330	-0.6	0.336	0.87	0.82	1.38	1.39

Table 7: Results for Example 6

n_1	n_2	θ_s	θ_{ap}	%err	θ_1	n_s^a	n_{ap}^a	n_s^f	n_{ap}^f
2	2	0.205	0.198	-3.4	0.212	0.45	0.52	0.52	0.49
3	3	0.262	0.258	-1.5	0.275	0.64	0.65	0.79	0.78
5	5	0.322	0.330	2.4	0.348	0.93	0.83	1.36	1.39
2	4	0.237	0.228	-3.8	0.233	0.14	0.21	0.62	0.60
3	5	0.283	0.282	-0.4	0.289	0.26	0.30	0.91	0.90
4	6	0.316	0.319	0.9	0.325	0.38	0.35	1.21	1.22
7	8	0.365	0.378	3.6	0.385	1.91	1.94	1.70	1.69

Table 8: Results for Example 7

n_1	n_2	θ_s	θ_{ap}	%err	θ_1	n_s^a	n_{ap}^a	n_s^f	n_{ap}^f
2	2	0.227	0.221	-2.6	0.226	0.23	0.31	0.59	0.56
3	3	0.280	0.280	0.0	0.286	0.35	0.34	0.88	0.88
5	5	0.339	0.348	2.7	0.352	0.49	0.32	1.50	1.56
2	4	0.242	0.239	-1.3	0.240	0.05	0.12	0.65	0.63
3	5	0.290	0.292	0.7	0.293	0.13	0.15	0.96	0.95
4	6	0.326	0.328	0.6	0.328	0.16	0.16	1.28	1.28
7	8	0.378	0.385	1.8	0.386	0.47	0.21	2.18	2.26

Table 9: Results for Example 8

n_1	n_2	θ_s	θ_{ap}	%err	θ_1	n_s^a	n_{ap}^a	n_s^f	n_{ap}^f
2	2	0.242	0.238	-1.7	0.240	0.09	0.13	0.64	0.62
3	3	0.293	0.293	0.0	0.294	0.12	0.13	0.96	0.96
5	5	0.352	0.354	0.5	0.355	0.14	0.11	1.62	1.63
2	4	0.248	0.245	-1.2	0.245	0.03	0.05	0.66	0.65
3	5	0.296	0.296	0.0	0.297	0.04	0.06	0.99	0.99
4	6	0.329	0.331	0.6	0.331	0.06	0.07	1.31	1.31
7	8	0.388	0.388	0.0	0.388	0.16	0.09	2.28	2.30

Table 10: Results for Example 9

We also examined the performance of the throughput approximation for systems with multiple machines and more than 2 lines. Example 10 represents a multi-machine system with unbalanced fabrication lines and an assembly machine faster than the fabrication bottleneck. In this example, line 1 has 5 stations and line 2 has 4 stations. (We use the word "stations" here because multiple machines exist.) The processing times (number of machines) for line 1 are: 1.7 (1), 3 (2), 5 (3), 2 (1), 1.5 (1) and for line 2: 6 (3), 5(2), 2 (1), 1.4 (1). The assembly station consists of a single machine with mean processing time of 2. The results in Table 11 show that, despite the introduction of multiple machines, the throughput approximation was still consistently within 4% of simulation. Interestingly, the approximation gave better results after only one iteration and got worse with more iterations in this case.

Next, we considered an example with 3 lines. This example had unbalanced fabrication lines and the (not very pronounced) bottleneck in fabrication. For this example, line 1 had 4 machines with mean processing times 3.43, 2.87, 3.74, 2.77, line 2 had 3 machines with mean processing times 2.83, 3.05, 2.14 and line 3 had 3 machines with processing times 2.52, 1.03 and 3.58. The assembly machine had a mean processing time of 3.66. The results for Example 11, given in Table 12, show that despite the fact that we had 3 lines,

n_1	n_2	θ_s	θ_{ap}	%err	θ_1
5	4	0.202	0.196	-3.0	0.199
6	4	0.206	0.202	-1.9	0.203
3	3	0.149	0.143	-4.0	0.148
4	3	0.161	0.156	-3.1	0.158
4	5	0.206	0.198	-3.8	0.203
3	4	0.164	0.158	-3.6	0.161
4	4	0.190	0.184	-3.2	0.190
5	6	0.238	0.230	-3.4	0.237

Table 11: Results for Example 10

n_1	n_2	n_3	θ_s	θ_{ap}	%err	θ_1
3	4	5	0.129	0.125	-3.1	0.125
4	4	4	0.146	0.142	-2.7	0.144
3	2	2	0.110	0.109	-0.9	0.113
5	3	4	0.151	0.150	-0.7	0.154
3	3	3	0.124	0.119	-4.0	0.123
2	7	4	0.100	0.099	-1.0	0.099
4	5	3	0.142	0.144	1.4	0.145
5	5	5	0.162	0.160	-1.2	0.161

Table 12: Results for Example 11

the throughput approximation worked very well and the results were still within 4% of the simulation values.

These eleven examples are representative of our experience that our approximations behave well for multi machine and single machine systems. In all cases, the maximum error of the throughput approximation was about 4%. The queue length approximations, particularly for the mean number of jobs at assembly, were somewhat worse than this but clearly respectable. Whenever a capacity or WIP imbalance introduces a distinct bottleneck, the approximations seem to work extremely well. However, in balanced cases, the approximations also worked well. As a final accuracy check we computed the standard deviation of the simulation means. In every case we tried, the throughput approximation was easily within two standard deviations of the mean.

6 Design Considerations

The primary uses of the throughput approximation developed above would be to help design new production lines and reconfigure existing lines. The two major controls available to the

decision-maker are: (1) the capacities of the work stations which are determined by the rate at which they produce and the number of machines at each station and (2) the WIP levels (card counts) in each fabrication line. By using the approximation developed here, the decision-maker could make fast rough-cut comparisons of the impact of different capacity and/or WIP placement strategies. By quickly ruling out many combinations, the decision-maker could use simulation for detailed consideration of only a few promising alternatives. We illustrate below how our approximation can be used in these problems.

Capacity Allocation

A major control available to a decision-maker in many manufacturing systems is the capacities of work stations. This problem has been addressed for serial lines in many papers and we refer the reader to Baker et al. [3] for a literature review. One way to approach the capacity allocation problem is to maximize throughput subject to a budget constraint. If we assume that the cost of capacity at fabrication station ij (assembly station A) is a function of mean processing time, which we denote by $c_{ij}(1/\lambda_{ij})$ ($c_A(1/\lambda_A)$), then we can use our approximation as the basis for a formulation of the capacity allocation problem.

One reasonable formulation would try to maximize throughput subject to a budgetary constraint, where the decision variables are the capacities at each station *and* the WIP levels. This implies a nested algorithm, which chooses feasible capacity configurations and then optimizes the WIP levels for that configuration. In order for this procedure to make sense, we must define the WIP setting problem precisely. We will address this question separately below, after considering the issue of where the bottleneck should be placed.

Bottleneck Placement

In the general case, where capacity costs different amounts at different workstations, we cannot say much about the placement of the bottleneck other than that it depends on the capacity costs. However, in situations where capacity costs are uniform, which might occur where capacity additions are made by adding workers, we can make more precise observations. Under these conditions, we can address the question of whether it is

preferable to place the bottleneck in fabrication or assembly.

We begin by using our approximation to show that in a simple example consisting of an assembly machine fed by two single machine fabrication lines, exchanging the bottleneck at assembly with a faster fabrication machine causes throughput to increase. To do this, we define processing times p_1, p_2 and p_3 such that $p_1 \leq p_2 \leq p_3$. We start with the line configured such that the machine with processing time p_1 is in the first fabrication line, the machine with processing time p_2 is in the second fabrication line, and the machine with processing time p_3 is at assembly. We suppose that the first fabrication line has n_1 jobs and the second fabrication line has n_2 jobs.

Without loss of generality, we assume that the throughput of the closed queueing network consisting of n_2 jobs, and processing times p_2 and p_3 is less than the throughput of the closed queueing network consisting of n_1 jobs, and processing times p_1 and p_3. We let $G(n_1, p_1, p_2)$ denote Buzen's coefficient for a closed queueing network with 2 machines, n_1 jobs and processing times p_1 and p_2. In this case, our approximation would approximate the assembly system by a closed queueing network with processing times p_2 and $p_3 + p_1^{n_1+1}/G(n_1, p_1, p_3 + p_2^{n_2+1}/G(n_2, p_2, p_3))$.

Now, if we consider the assembly system where the machines with processing times of p_2 and p_3 are interchanged, our approximation would approximate this system by a closed queueing network with 2 machines, n_1 jobs and processing times p_3 and $p_2 + p_1^{n_1+1}/G(n_1, p_1, p_2 + p_3^{n_2+1}/G(n_2, p_2, p_3))$.

To show that the throughput is higher in the second case, we first note that $p_2 + p_3^{n_2+1}/G(n_2, p_2, p_3) = p_3 + p_2^{n_2+1}/G(n_2, p_2, p_3)$. Hence, if we let $\delta = p_1^{n_1+1}/G(n_1, p_1, p_3 + p_2^{n_2+1}/G(n_2, p_2, p_3))$, we are comparing the throughput of two closed queueing networks one with processing times p_2 and $p_3 + \delta$ and the other with processing times p_3 and $p_2 + \delta$. It is straightforward to show by induction that the throughput of the closed queueing network with processing times p_3 and $p_2 + \delta$ is higher.

Because we are using an approximation, the above result does not prove that exchanging

n_1	n_2	θ_1	θ_2	%dif
2	2	0.128	0.138	7.8
3	3	0.167	0.179	7.2
3	4	0.176	0.197	11.9
2	3	0.139	0.161	15.8
2	4	0.141	0.172	21.9

Table 13: Bottleneck in Assembly vs. Fabrication

the bottleneck at assembly with a faster fabrication machine increases throughput. However, by using simulation we verified that this result does seem to hold for simple systems, such as that considered above, and for more complicated systems. In fact, the increase in throughput can be substantial, as we illustrate in the following example.

Consider a two-line assembly system with five machines in each line. The machines in the first line have mean processing times of 1.7, 2.1, 1.3, 0.6 and 2.5 and the machines in the second line have mean processing times of 0.8, 1.4, 3.0, 1.0, and 1.1. The assembly machine has a processing time of 3.5. We also consider the system identical to this one except that the assembly machine is switched with the machine which has a processing time of 0.8. We denote the throughput of the first system by θ_1 and the throughput of the second system by θ_2. In Table 13, we display the throughput for the two systems under different WIP levels. Notice that the difference in throughput between the two systems is as high as 21.9%.

Setting WIP levels

An important problem in the control of a pull manufacturing system is setting work-in-process (WIP) levels (card counts). The card counts affect the rate at which goods are produced in a pull system. Early pull systems set card counts by trial and error ([13], [24]). Recent research has sought quantitative models for assisting in the WIP setting process. Bitran and Chang [5] developed a mathematical programming approach to determine the

number of cards for a deterministic kanban system. Wang and Wang [31] used a Markov process approach to compute card counts for a kanban system with exponential machines. Duenyas, Hopp and Spearman [10] developed a method for setting quota and card counts for a single fabrication line operating under the CONWIP protocol.

For the purposes of WIP setting, we will assume that the firm can sell all it can make (i.e., its master production schedule is always filled). This assumption will be realistic in many cases where demand is large relative to plant capacity. It may also be valid where the firm has made a strategic decision to limit capacity (e.g., by controlling the number of shifts or machine capacities) below the level of demand.

Under the sell-all-you-can-make assumption, we state an objective function by assuming that every unit produced generates a revenue of p, and a periodic unit holding cost for WIP, h_i, is associated with each line i. The periodic profit as a function of the card count vector, n, is denoted by $\pi(n)$ and can be written as

$$\pi(n) = p\theta\{\lambda/\lambda_A/n\} - \sum_{i=1}^{k} h_i n_i. \tag{6.1}$$

The optimal card count can be found by searching over different values of n.

While this approach is mathematically tractable, it may not be realistic. The reason is that frequently actual holding costs are a less important disincentive to high WIP than are the long, uncompetitive cycle times produced by high WIP levels. If the firm has long cycle times, then the probability of an order being cancelled or changed increases, which translates into decreased revenue or increased costs to the firm. One could address this issue by introducing a non-linear penalty for holding costs, but then the question would arise as to how to set this penalty function.

Hence, it may make more sense to set up the problem in terms of maximizing throughput subject to an overall cycle time constraint. Notice that this need not imply that the cycle times for different lines have to be the same. If, for example, the cost of material in a certain line is much higher than the cost of material in the other lines, it may be more desirable to have a shorter cycle time for that line. Our problem can now be stated as maximizing the

throughput subject to the constraint that average cycle times for line i are not to exceed d_i. We must choose the card count n_i so as to achieve this. We let $c_i\{F/F_A/n\}$ denote the cycle time for line i in system $\{F/F_A/n\}$. The following proposition enables us to restrict the set of WIP values we must consider.

Proposition 2 *Suppose n and $\hat{n}$ are two arrays such that for a given i',*

$$n_i > \hat{n}_i \quad \textit{for } i = i'$$

$$n_i = \hat{n}_i \quad \textit{otherwise}$$

Then $\theta\{F/F_A/n\} \geq \theta\{F/F_A/\hat{n}\}$ and for $j = 1, \ldots, k$, $j \neq i'$,

$$c_j\{F/F_A/n\} \leq c_j\{F/F_A/\hat{n}\}$$

Proof: The proof is similiar to the proof of Proposition 1. □

Proposition 2 indicates that if the cycle time for line j is greater than d_j, then decreasing the number of jobs in other lines (lines $i = 1, \ldots, k, i \neq j$), will not make c_j feasible. We now present an

Algorithm for computing optimal card counts

1. *For $i = 1, \ldots, k$, let n_i be the highest value such that*

$$c_i\{\lambda_i/\lambda_A/n_i\} \leq d_i$$

2. *Compute θ_{ap} using the previously given procedure and $c_i, i = 1, \ldots, k$ using (4.7).*

3. *If for $i = 1, \ldots, k, c_i \leq d_i$ then stop, the current n is optimal. Else, for each i such that $c_i > d_i$ let $n_i = n_i - 1$. Go to step 2.*

We use Proposition 2 in Step 3 of the algorithm. Suppose c_i is infeasible, then by Proposition 2, decreasing the number of jobs in other lines will only make c_i more infeasible. We can not increase the number of jobs in other lines, since that will make the cycle times

of those lines infeasible, hence the only solution is to decrease n_i. Since throughput is increasing in the number of jobs in any line, by Proposition 2, the first feasible solution that the algorithm finds will also necessarily be optimal.

One question that we would like to answer is whether the optimal card counts that we obtain from the above card count setting algorithm are of any value to us if the processing time distributions are not exponential. The following result shows that if we solve the card count setting problem with exponential processing times, then the results that we obtain will be a lower bound for a system consisting of stations which have processing times that are stochastically less than the exponential.

Proposition 3 *Let $\{F/F_A/n\}$ denote an assembly system as in Figure 1. Let $N = (N_1, N_2, ..., N_k)$ be a vector of optimal card counts obtained for this system. Let $\{F'/F'_A/n\}$ denote another assembly system such that this system has the same number of lines and machines as $\{F/F_A/n\}$, but $F'_{ji} \leq^{st} F_{ji}$ (or $F'_{ji} \leq^{icx} F_{ji}$) for all j and i and $F'_A \leq^{st} F_A$. Let N' be the vector of optimal card counts for $\{F'/F'_A/n\}$. Then, for all $j = 1, ..., k$, $N'_j \geq N_j$.*

Proof: It follows from Proposition 1 that $\theta\{F/F_A/n\} \leq \theta\{F'/F'_A/n\}$ for all n. Hence, $\theta\{F/F_A/N\} \leq \theta\{F'/F'_A/N\}$. By Little's law, N will be feasible for $\{F'/F'_A/N\}$. To show that $N'_j \geq N_j, j = 1..., k$, suppose not, i.e., suppose that for some line $\hat{j}$, $N'_{\hat{j}} < N_{\hat{j}}$. Let N'' be defined such that

$$N''_j = N'_j \quad \text{for all } j \neq \hat{j}$$

$$N''_j = N_j \quad \text{for } j = \hat{j}$$

Then, $\theta\{F/F_A/N''\} \geq \theta\{F/F_A/N'\}$ since for all $j = 1, ..., k; N''_j \geq N'_j$. Since N' is optimal and hence feasible, and N'' is the same as N' except for the line $\hat{j}$, using Proposition 2, we find that for all $j = 1, ..., k, j \neq \hat{j}$, we have $c_j\{F'/F'_A/N''\} \leq c_j\{F'/F'_A/N'\} \leq d_i$. Similarly, since $N''_{\hat{j}} = N_{\hat{j}}$, and $N''_j \geq N_j$ for all other j, we have $c_{\hat{j}}\{F'/F'_A/N''\} \leq c_{\hat{j}}\{F'/F'_A/N\} \leq d_i$. Thus, N'' is feasible and results in higher throughput than N', but this is a contradiction since it was assumed that N' was the optimal solution. Hence the result has been proven. □

Proposition 3 can be used along with the procedure developed above to decrease the number of simulations that have to be performed for setting card counts for systems with general processing time distributions. For example, suppose we have managed to find the optimal card count distribution for a system with normal processing times (e.g., by simulation). Then, since decreasing the mean of the normal results in a stochastically smaller distribution, Proposition 3 implies that the counts in this optimal vector represents a lower bound on the optimal counts that will result if any processing time mean is decreased. Also, Proposition 3 implies that the optimal card count for systems where all the processing times are exponential is a lower bound on systems where all the processing times are IFR with the same mean. To see this, we note, as in Spearman [27], that IFR implies NBUE (new better than used in expectation) and that if $F_{ji}(F'_{ji})$ is NBUE (exponential) with mean λ_{ji}^{-1} then $F'_{ji} \geq^{icx} F_{ji}$.

7 Conclusions and Further Research

In this paper, we derived approximations of the throughput and average number of jobs in queue for an assembly-like queueing system. These approximations can be used as the basis of a decision support system which, in conjunction with simulation, aids the user in the configuration of fabrication/assembly lines. We demonstrated how the throughput approximation can be utilized in decisions concerning capacity and WIP levels.

In the course of our analysis, we have made several qualitative observations about the behavior of assembly systems. Briefly, these can be summarized as follows.

1. Throughput is a non-decreasing function of machine speed. Specifically, if processing time distributions are replaced by stochastically smaller distributions, throughput goes up. The throughput of an assembly system with exponential processing times is a lower bound on the throughput of an assembly system with IFR processing times with the same mean processing times.

2. A bottleneck at assembly limits throughput more than an equivalent bottleneck in fabrication. Exchanging (in terms of capacity, not functionality) the bottleneck at assembly with a faster fabrication machine increases throughput.

3. In the problem of allocating card counts to fabrication lines to achieve maximum throughput subject to cycle time constraints, faster machines (i.e., in the sense of having stochastically smaller processing times) allow greater WIP and therefore higher throughput. The optimal card count for an assembly system with exponential processing times is a lower bound for an assembly system with IFR processing times with the same mean processing times.

In conclusion, this paper provides a potentially useful first cut at developing analytical modeling support for assembly systems, but there is much work to be done. Further research should include characterizing cycle time variance and the variance of the cumulative output until a fixed time t of assembly-like queueing systems (e.g., in the vein of [11]). These results would provide the basis for more general production quota and WIP setting models. Further work should also address assembly-like queues with more general processing time distributions. While the exponential distribution is a good approximation for systems with high variability, it may not be a good characterization when this assumption is violated, for instance in the case of automatic machinery. In that case, a model with deterministic processing times and random failures may be more appropriate, as in [10].

Acknowledgement

We would like to thank the associate editor and two referees for many insightful comments that greatly improved the paper. This work was supported in part by the National Science Foundation under Grant No: DDM-8905638 and by a Horace H. Rackham School Grant.

Bibliography

[1] M.H. Ammar. 1980. "Modelling and Analysis of unreliable manufacturing assembly networks with finite storage." MIT Laboratory for Information and Decision Sciences, Report LIDS-TH-1004.

[2] K.R. Baker, S.G. Powell and D.F. Pyke. 1990. "Buffered and Unbuffered Assembly Systems with Variable Processing Times." Working Paper No. 246, The Amos Tuck School of Business Administration, Dartmouth College, Hanover, NH 03755.

[3] K.R. Baker, S.G. Powell and D.F. Pyke. 1990. "Optimal Allocation of Work in Assembly Systems." Working Paper No. 258, The Amos Tuck School of Business Administration, Dartmouth College, Hanover, NH 03755.

[4] U.N. Bhat. 1986. "Finite capacity assembly-like queues." *Queueing Systems: Theory and Applications* **1** 15, 85.

[5] G.R. Bitran and L. Chang. 1987. "A mathematical programming approach to a deterministic kanban system." *Management Science* **33**, 427.

[6] F. Bonomi. 1987. "An approximate analysis for a class of assembly-like queues." *Queueing Systems: Theory and Applications* **1**, 289.

[7] J.P. Buzen. 1973. "Computational Algorithms for Closed Queueing Networks with Exponential Servers." *Communications of the ACM* **16**, 527.

[8] G.L. Curry, B.L. Deuermeyer, and R.M. Feldman. 1989. *Discrete Simulation: Fundamentals and Microcmputer Support.* Holden-Day, Oakland.

[9] J.B. Dennis. 1980. "Data flow super computers." *IEEE Computer* **13**, 48.

[10] I. Duenyas, W.J. Hopp, and M.L. Spearman. 1991. "Characterizing the Output Process of a CONWIP Line with Deterministic Processing and Random Outages." Technical Report 91-11, Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL 60208.

[11] I. Duenyas and W.J. Hopp. 1990. "Estimating Variance of Output from Cyclic Exponential Queueing Systems." *Queueing Systems* **7**, 337.

[12] GINO, General Interactive Optimizer, is a trademark of LINDO Systems Inc., P.O. Box 148231, Chicago, IL.

[13] R.W. Hall. 1983. *Zero Inventories.* Dow Jones-Irwin, Homewood, IL.

[14] J.M. Harrison. 1973. "Assembly-like Queues." *Journal of Applied Probability* **10**, 354.

[15] F. Hillier, and R. Boling. 1966. "The Effects of Some Design Factors on the Efficiency of Production Lines with Variable Operation Times." *The Journal of Industrial Engineering,* **17**, 651.

[16] W.J. Hopp and J.T. Simon. 1989. "Bounds and Heuristics for Assembly-Like Queues." *Queueing Systems: Theory and Applications* **4**, 137.

[17] W.J. Hopp, M.L. Spearman, and I. Duenyas. "Economic Production Quotas for Pull Manufacturing Systems." to appear in *IIE Transactions.*

[18] E.H. Lipper and B. Sengupta. 1986. "Assembly-Like queues with finite capacity: bounds, asymptotics and approximations." *Queueing Systems: Theory and Applications* **1**, 67.

[19] Y. Monden. 1983. *Toyota Production System.* Industrial Engineering and Management Press.

[20] T. Ohno. 1988. *Toyota Production System: Beyond Large Scale Production.* Productivity Press, Cambridge, Mass. (original Japanese 1978)

[21] M. Reiser and S. Lavenberg. 1980. "Mean Value Analysis of Closed MultiChain Queueing Networks." *J.ACM* **27**, 313.

[22] S.M. Ross. (1983). *Stochastic Processes*, Wiley, New York.

[23] R. Schassberger and H. Daduna. 1983. "The Time for a Round Trip in a Cycle of Exponential Queues." *J.ACM* **30**, 146.

[24] R.J. Schonberger. 1986. *World Class Manufacturing: The Lessons of Simplicity Applied.* The Free Press, New York.

[25] J.J. Solberg. 1981. "Capacity planning with a stochastic workflow model." *AIEE Transactions* **13**, 116.

[26] J.J. Solberg, S.Y. Nof. 1980. "Analysis of flow control in alternative manufacturing configurations." *Journal of Dynamic Systems Measurement and Control* **102**, 141.

[27] M.L. Spearman. 1991. "An Analytic Congestion Model for Closed Production Systems with IFR Processing Times." *Management Science* **37** 1015.

[28] M.L. Spearman, W.J. Hopp and D.L. Woodruff. 1989. "A Hierarchical Control Architecture for Constant Work-In-Process (CONWIP) Production Systems." *Journal of Manufacturing and Operations Management* **2**, 147.

[29] M.L. Spearman, D.L. Woodruff and W.J. Hopp. 1990. "CONWIP: a pull alternative to kanban." *International Journal of Production Research* **28**, 879.

[30] M.L. Spearman, M.A. Zazanis. "Push and Pull Production Systems: Issues and Comparisons." to appear in *Operations Research.*

[31] H. Wang and H.B. Wang. 1990. "Determining the number of kanbans: a step toward non-stock-production." *International Journal of Production Research,* **28**, 2101.

[32] W. Whitt. 1983. "The Queueing Network Analyzer" and "Performance of the Queueing Network Analyzer." *Bell Systems Technical Journal* **62**, 2799-2843.