

Performance evaluation of closed tree-structured assembly systems

JISHNU HAZRA¹ and ABRAHAM SEIDMANN²

¹Indian Institute of Management, Bangalore, 5600 76, India

²William E. Simon Graduate School of Business Administration, University of Rochester, Rochester, NY 14627, USA

Received September 1993 and accepted August 1995

Assembly systems are very common in manufacturing environments. In such systems, certain machines require more than one part type before they can start processing. Our paper presents a practical stochastic model for evaluating the performance of closed-loop assembly systems. This evaluation is difficult because of synchronization delays and the large state space required to capture the interaction of the flow of parts among the various subassemblies. We present an approximate, but computationally efficient, aggregation/disaggregation algorithm to compute the system throughput and mean queue lengths for both the matched and unmatched WIP inventory components. The aggregation computes the conditional system throughput as a function of the number of jobs in the system, whereas the mean queue lengths are obtained by disaggregation. Application of our algorithm reveals several salient operational insights, such as the increase in queue length towards the downstream machines or the concavity of the assembly system throughput with respect to the total number of kanbans.

1. Introduction

Most modeling of manufacturing systems assumes that each machine requires only one input, yet in practice many systems contain machines that require more than one distinct input. These machines cannot start processing until all inputs from the preceding stage are available; this results in a synchronization delay, which makes the system difficult to analyze. Simulation has been the primary tool for analyzing such complex assembly systems (Baker *et al.*, 1993), but it is time-consuming, and the output requires further statistical analysis. In this paper we present a computationally efficient analytical model for studying the performance of closed assembly systems in which some machines carry out assembly operations.

We analyze a production system that has an assembly-like network structure, which is common in the automobile and electronics industries. It is well known that assembly-like queueing systems with infinite buffer capacity are unstable unless job-loading at the first stage is linked to the output rate at the last stage (Harrison, 1973). We use a practical production control scheme that may be considered a combination of push and pull mechanisms. The final production stage sends a production authorization card to the first stage (the pull component), and thereafter jobs flow from the first to the last stage as if they are in a push system (Spearman *et al.*, 1990). Performance analyses of such systems have been mentioned as important future research issues in Dallery and Gershwin (1992) and Suri *et al.* (1993). Because a

large state space is needed to represent such systems, exact analysis is not computationally feasible. We therefore present a new approach for approximately computing two major performance measures: mean throughput and the mean queue lengths at each work station. Our analysis identifies and measures the special dynamics of work-in-progress (WIP) flow in assembly networks. Conceptually we show that WIP has two parts between an assembly station and the preceding stages: the first part is the unmatched components that are waiting for a corresponding piece in another input buffer; the second is a complete kit (matched components) awaiting assembly.

Earlier studies of two-stage systems include Ammar and Gershwin (1990), Bhat (1986), Bonomi (1986), Hopp and Simon (1989), Lipper and Sengupta (1986), Liu and Perros (1991a, b) and Rao and Suri (1994). Even less has been written on multi-stage assembly systems (DiMascolo *et al.*, 1991; Gershwin, 1991; Liu and Buzacott, 1989); these papers analyze open systems with finite buffer capacity. Duenyas and Hopp (1993) have considered a closed system that has n lines feeding a single assembly machine with work releases controlled by a CONWIP mechanism similar to ours. For a recent survey of serial and assembly flowlines, see, for example, Dallery and Gershwin (1992) and Gershwin (1994).

Our paper is the first to present a computationally efficient methodology for analyzing the performance of closed queueing networks with a tree-like topology and with any number of assembly machines. The number of jobs in the assembly network is an independent variable,

and the production output is dependent. In Section 2 we describe our model and discuss our aggregation/disaggregation approach. In Section 3 we test our approximations and show how they can be used to improve the system's performance. Finally, Section 4 contains our conclusions and suggests directions for further research.

2. Analysis of the assembly system

2.1. Notation

A multi-stage assembly system can be represented by a connected directed tree. Nodes represent machines, and rectangles represent buffers (here we shall use the term nodes and machines interchangeably). The orientation of an arc is such that its tail emanates from the upstream machine and its head points into the downstream machine. Each buffer is connected to exactly two machines: an upstream machine and a downstream machine. Machine i is the *successor* of j if there is an arc oriented from node j to node i ; it belongs to set $S(j)$. Once a part completes processing on machine j , it is processed on machine $S(j)$. Each machine has exactly one successor machine (except the root node, which has no successor machine). Similarly, machine j is the *predecessor* of machine i if there is an arc oriented from node j to node i ; machine j belongs to set $P(i)$. A machine can have more than one predecessor. Machines represented by leaf nodes have no predecessors; the set of all leaf nodes is represented by L .

The buffer between downstream machine i and upstream machine j is denoted by $B(j, i)$. If i is a leaf node then $B(0, i)$ will represent its input buffer. In assembly systems the buffer can be conceptually decomposed into two queues: *unmatched* and *matched*. The unmatched part of the queue consists of parts waiting for the corresponding pieces. Once all the corresponding pieces are available, the parts join the matched queue of kits ready for assembly. In Fig. 1, each buffer is represented by its matched and unmatched queues (except for the buffer of the leaf nodes).

The number of parts waiting in buffer $B(j, i)$ plus the part being processed on machine i (if the machine is not idle) will be represented by the random variable B_{ji} ; its value is given by b_{ji} . If machine j is the predecessor of machine i , then the number of parts waiting in the unmatched queue of buffer $B(j, i)$ is represented by the random variable B_{ji}^u , and the number of completed kits in the matched queue is represented by the random variable B_{ji}^m (it includes the one being processed on machine i). The following equation will therefore always hold: $B_{ji} = B_{ji}^u + B_{ji}^m$. If we compute the expected value of any two of these three random variables, the third can be calculated from the above equation. In this paper we thus compute only $E(B_{ji})$ and $E(B_{ji}^m)$.

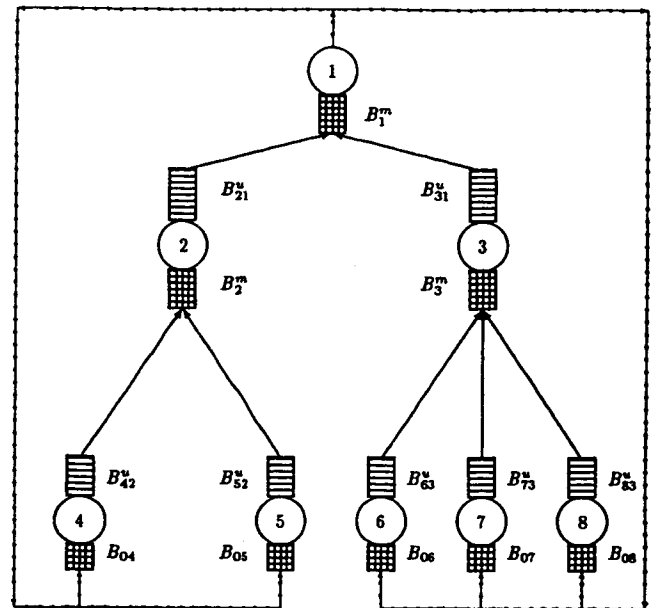


Fig. 1. An eight-machine assembly network: B_{ji}^u , number of parts waiting in the unmatched queue of buffer $B(j, i)$; B_{ji}^m , number of parts in the matched queue of buffer $B(j, i)$; node 1 is the root and nodes 4 to 8 are leaf nodes.

For each machine i , we define the *siblings* of i as all the machines that have the same successor; they belong to set $B(i)$. The root node has no siblings; it is also the final-stage machine. The *loops* in the system are completed by *information* arcs from the root node to each of the leaf nodes; these send work releases (see item 2 in Section 2.2). There are thus as many loops as there are leaf nodes. A *chain* $c(i_1, i_l)$ is a sequence of *nodes* connected by *buffers* from node i_1 back to leaf node i_l . Suppose such a sequence is represented by $i_1, i_2, i_3, \dots, i_{l-1}, i_l$; then $i_1 = S(i_2)$, $i_2 = S(i_3)$, and finally $i_{l-1} = S(i_l)$. The set of buffers in this chain will be $B(i_2, i_1), B(i_3, i_2), \dots, B(i_l, i_{l-1}), B(0, i_l)$. In Fig. 1, node 1 is the root node, and nodes 4, 5, 6, 7, and 8 are the leaf nodes. Chains $c(1, 4)$ and $c(3, 7)$ have the node arc sequences 1-2-4 and 3-7, respectively. The information feedback arcs connect node 1 to each of the leaf nodes.

2.2. Operational assumptions

The model used to describe the part and information flows is based on typical operational assumptions (Duenyas and Hopp, 1993; Spearman *et al.*, 1990).

1. The processing times at machine i are independent and exponentially distributed with mean μ_i^{-1} .
2. When the root node finishes processing (or assembling) a part, the part leaves the system and a new part is added to each one of the input buffers of the leaf nodes instantaneously.
3. Machine i can operate only when all its input buffers

have at least one part. It takes one part from each input buffer and assembles them to form a new part. A machine is starved when any input buffer is empty.

4. The total number of parts in every chain $c(i_1, i_l)$, where i_1 is the root node and $i_l \in L$, is equal to the number of parts, N .

5. The buffer capacity of all machines is at least N ; thus they are never blocked.

6. Machine i produces a new part by assembling parts from each of its input buffers $B(P(i), i)$. The new part moves into its output buffer; this buffer also serves as the input buffer of $S(i)$, the next machine downstream.

We use the generic term *part* for components and subassemblies and final assemblies. It is understood that a part in a buffer of a downstream machine is an assembly of parts from the buffers of upstream machines. For example, in Fig. 1, a part in buffer $B(2, 1)$ is formed by combining parts from buffers $B(4, 2)$ and $B(5, 2)$.

2.3. Analytical approach

We model the assembly system as a continuous-time Markov chain (CTMC). If $B(t)$ represents the state space at time t , then the stochastic process $\{B(t); t \geq 0\}$ is the underlying CTMC. There is more than one way to represent $B(t)$; the obvious method is buffer occupancy, and the machine state when buffer occupancy is zero. The buffer occupancy of the leaf nodes is not needed in the state representation. This CTMC can be analyzed exactly, but an exact solution is impracticable because even for a small system the state space becomes very large. For example, the number of states for the network in Fig. 1, with N parts circulating in each loop, is equal to $(N+1)^3(N+2)^3(2N+3)/24$; for $N=20$ there are more than 176 million states.

Our algorithm calculates throughput and mean queue lengths and introduces an extension of the aggregation/disaggregation technique developed earlier by Chandy *et al.* (1975). During the aggregation step we sequentially shrink the size of the network by replacing each subnetwork with a single node; this method renders a complex system simple. At the end of the aggregation step we compute the throughput of the root node. During the disaggregation we compute the throughput of all other nodes and the mean queue lengths.

In Hazra and Seidmann (1995) we prove that any network that satisfies the assumptions of Section 2.2 has an equivalent two-stage assembly network. The second stage of this equivalent network consists of a root node with the same processing rate as that in the original network. The first stage consists of as many nodes as there are predecessors of the root node in the original network. These nodes have state-dependent pro-

cessing rates. These processing rates cannot be computed exactly without first solving for the steady-state probabilities of the original network. We thus develop an approximation for computing these processing rates.

2.4. Approximations for throughput and queue lengths

The approximation proceeds by replacing a subnetwork with a single node, which is termed a *complex* node and is denoted by a prime sign. The subnetwork is a two-stage assembly system with n parts circulating in each loop; this network will be represented by TSAS(n, μ_i, S), where $S = D \cup F$, $D = \{\mu_j : j \in P(i) \cap L\}$, and $F = \{\mu_j : j \in P(i) \text{ and } j \notin L\}$. The processing rate of the first-stage nodes in this subnetwork is represented by sets D and F ; the latter set comprises the complex nodes. The processing rate of the second-stage machine is given by μ_i .

2.4.1. Estimation of throughput

To simplify our exposition we follow a specific application example of the algorithm. Consider the network in Fig. 1. We first replace the subnetwork consisting of nodes 3, 6, 7 and 8, TSAS($n, \mu_3, \{\mu_6, \mu_7, \mu_8\}$), with a single node, 3'. We then compute the throughput of this subnetwork for $n = 1, \dots, N$, where n is the number of parts circulating in each loop. The stationary probabilities of the state of the system of the subnetwork are calculated by solving its global balance equations. The state of the system is represented by (b_{63}, b_{73}, b_{83}) , where b_{ji} is the number of parts in buffer $B(j, i)$. $\pi_3^n(s)$ is the steady-state probability of state s with n parts circulating in each loop (the subscript 3 is the assembly node of the subnetwork under consideration). The balance equation is given by

$$\begin{aligned} & [\mu_3 I(b_{63} > 0, b_{73} > 0, b_{83} > 0) + \mu_6 I(b_{63} < n) + \\ & \mu_7 I(b_{73} < n) + \mu_8 I(b_{83} < n)] \pi_3^n(b_{63}, b_{73}, b_{83}) \\ & = \mu_3 I(b_{63} < n, b_{73} < n, b_{83} < n) \pi_3^n(b_{63} + 1, b_{73} + 1, b_{83} + 1) \\ & + \mu_6 I(b_{63} > 0) \pi_3^n(b_{63} - 1, b_{73}, b_{83}) + \\ & \mu_7 I(b_{73} > 0) \pi_3^n(b_{63}, b_{73} - 1, b_{83}) + \\ & \mu_8 I(b_{83} > 0) \pi_3^n(b_{63}, b_{73}, b_{83} - 1), \quad 0 \leq b_{63}, b_{73}, b_{83} \leq n. \quad (1) \end{aligned}$$

$I(c)$ is the indicator function and has a value of 1 if condition c is true, 0 otherwise. Equation (1) and the normalizing equation are used to compute the steady-state probabilities. The throughput of this system is given by $\theta(n) = \mu_3 \sum_{n_1=1}^n \sum_{n_2=1}^n \pi_3^n(n_1, n_2)$. The processing rate at the complex node 3' is then given by $\mu_{3'}(n) = \theta(n)$, which is state-dependent and depends on the number of parts in its input buffer. This could be interpreted as an instantaneous departure rate from the subnetwork, conditional on the number of parts in it. We also approximate the following conditional probabilities

for the original network by

$$\Pr(B_{63} = b_{63}/B_{63} + B_{06} = n) = \sum_{n_2=0}^n \pi_3^n(b_{63}, n_2),$$

$$b_{63} = 0, \dots, n. \quad (2)$$

Similar equations can be obtained for buffers $B(7, 3)$ and $B(8, 3)$. The following relationship also holds true:

$$\Pr(B_{06} = b_{06}/B_{63} + B_{06} = n) =$$

$$\Pr(B_{63} = n - b_{06}/B_{63} + B_{06} = n), \quad b_{06} = 0, \dots, n. \quad (3)$$

Equations (2) and (3) hold for $n = 1, \dots, N$. The aggregation of nodes 2, 4 and 5 into node 2' is done in a similar fashion. The entire network is thus reduced to three nodes, 1, 2', and 3'. We can now solve the system given by TSAS($N, \mu_1, \{\mu_{2'}, \mu_{3'}\}$). The number of parts circulating in each loop is N because the number of parts in the input buffer of node 2' (3') represents the sum of parts in buffers $B(4, 2)$ and $B(0, 4)$ or $B(5, 2)$ and $B(0, 5)$ ($B(6, 3)$ and $B(0, 6)$ or $B(7, 3)$ and $B(0, 7)$). The following condition is always true: $B_{42} + B_{04} = B_{52} + B_{05}$ ($B_{63} + B_{06} = B_{73} + B_{07}$). We approximate the throughput of the original network by the throughput of this subnetwork. If $\pi_1^N(b_{21}, b_{31})$ is the steady-state probability of state (b_{21}, b_{31}) , where b_{21} is the number of parts in buffer $B(2', 1)$ and b_{31} is the number of parts in $B(3', 1)$, then the balance equations are given by

$$[\mu_1 I(b_{21} > 0, b_{31} > 0) + \mu_{2'}(N - b_{21})I(b_{21} < N)$$

$$+ \mu_{3'}(N - b_{31})I(b_{31} < N)]\pi_1^N(b_{21}, b_{31}) =$$

$$\mu_1 I(b_{21} < N, b_{31} < N)\pi_1^N(b_{21} + 1, b_{31} + 1) +$$

$$\mu_{2'}(N - b_{21} + 1)I(b_{21} > 0)\pi_1^N(b_{21} - 1, b_{31}) +$$

$$\mu_{3'}(N - b_{31} + 1)I(b_{31} > 0)\pi_1^N(b_{21}, b_{31} - 1),$$

$$0 \leq b_{21}, b_{31} \leq N. \quad (4)$$

In (4), $\mu_{i'}(\cdot)$, $i' = 2', 3'$, indicates that the processing rates of these nodes are dependent on the numbers of parts in their input buffers. We approximate the conditional probabilities of the original system by

$$\Pr(B_{31} = b_{31}/B_{31} + B_{63} + B_{06} = N) = \sum_{i=0}^N \pi_1^N(i, b_{31}),$$

$$b_{31} = 0, \dots, N. \quad (5)$$

A similar equation can be obtained for buffer $B(2, 1)$. Since $\Pr(B_{31} + B_{63} + B_{06} = N) = 1$, we thus have

$$\Pr(B_{31} = b_{31}) = \Pr(B_{31} = b_{31}/B_{31} + B_{63} + B_{06} = N),$$

$$b_{31} = 0, \dots, N. \quad (6)$$

Moreover,

$$\Pr(B_{31} = b_{31}) = \Pr(B_{63} + B_{06} = N - b_{31}),$$

$$b_{31} = 0, \dots, N. \quad (7)$$

Using these estimated probabilities, we can compute the marginal queue length distribution and thus the mean queue length. For example, the marginal queue length distribution of B_{63} is given by

$$\Pr(B_{63} = b_{63}) = \sum_{n=b_{63}}^N \Pr(B_{63} = b_{63}/B_{63} + B_{06} = n) \times$$

$$\times \Pr(B_{63} + B_{06} = n), \quad b_{63} = 0, \dots, N. \quad (8)$$

The first term on the right-hand side of (8) is obtained from (2), and the second term is obtained from (7). In the next section we show how to compute the marginal queue length distribution of a system with more than three stages.

2.4.2. The unmatched and matched queue length distributions

Consider chain $c(1, l)$ composed of a series of nodes 1, 2, 3, ..., l , where node 1 is the root and node l is the leaf, such that $S(2) = 1$, $S(3) = 2, \dots, S(l) = l - 1$. During the aggregation step the following probability distributions are estimated:

$$\Pr(B_{j+1,j} = b / \sum_{k=j}^l B_{k+1,k} = n) \quad b = 0, \dots, n \quad n = 1, \dots, N,$$

$$\forall j = 1, \dots, l - 1. \quad (9)$$

In the above equation we define $l + 1 = 0$; the probabilities will be zero for $b > n$. Equation (9) is the conditional queue length distribution, given the total WIP in chain $c(j, l)$. From these distributions the marginal queue length distribution can be calculated. As discussed earlier, $\Pr(B_{21} = b / \sum_{k=1}^l B_{k+1,k} = N) = \Pr(B_{21} = b)$ and $\Pr(B_{21} = b) = \Pr(\sum_{k=2}^l B_{k+1,k} = N - b)$. We can therefore calculate the marginal probability distribution of the number of parts in buffer $B(2, 1)$. By unconditioning (9) we get the probability distribution of buffer $B(3, 2)$:

$$\Pr(B_{32} = b) = \sum_{n=b}^N \Pr(B_{32} = b / \sum_{k=2}^l B_{k+1,k} = n) \times$$

$$\Pr(\sum_{k=2}^l B_{k+1,k} = n). \quad (10)$$

For any other buffer $B(j+1, j)$ we must compute the distribution of $\Pr(\sum_{k=j}^l B_{k+1,k} = n)$, $j = 3, \dots, l$, which can be done recursively with the following equation,

starting from $j = 3$:

$$\Pr\left(\sum_{k=j}^l B_{k+1,k} = n\right) = \sum_{b=0}^{N-n} \Pr(B_{jj-1} = b / \sum_{k=j-1}^l B_{k+1,k} = n+b) \\ \times \Pr\left(\sum_{k=j-1}^l B_{k+1,k} = n+b\right). \quad (11)$$

The marginal probabilities of queue lengths can be calculated by using the conditional probabilities obtained during the aggregation phase.

The matched queue length distribution of machine i (assuming it has two predecessors, j and k) is obtained by observing the distribution of the minimal total input buffer of i , as given by (12) and (13); the unmatched queue length distribution can be similarly obtained.

$$\Pr(B_i^m = b / \sum_{B(j_1,j_2) \in T(c(i,l))} B_{j_1 j_2} = n) = \\ \sum_{b_2=b+1}^n \pi_i^n(b, b_2) + \sum_{b_1=b+1}^n \pi_i^n(b_1, b) \\ + \pi_i^n(b, b), \quad b \leq n; \quad (12)$$

$$\Pr(B_i^m = b) = \sum_{n=b}^N \Pr(B_i^m = b / \sum_{B(j_1,j_2) \in T(c(i,l))} B_{j_1 j_2} = n) \times \\ \times \Pr\left(\sum_{B(j_1,j_2) \in T(c(i,l))} B_{j_1 j_2} = n\right); \quad (13)$$

where

$$\pi_i^n(b_1, b_2) = \Pr(B_{ji} = b_1, B_{ki} = b_2 / \sum_{(j_1,j_2) \in T(c(i,l))} B_{j_1 j_2} = n).$$

The first two terms on the right-hand side of (12) will equal 0 when $b = n$. $T(c(i, l))$ is the set of all buffers in chain $c(i, l)$, and l is a leaf node such that $i - j$ or $i - k$ is part of the node-arc-node sequence (recall that node i has two predecessors, j and k). The throughput of the root node is computed in the final step of the aggregation process, where we consider the subnetwork consisting of the root node and its predecessors. The throughput of any other node can also be approximated. The throughput of node i is simply given by

$$\mu_i \{1 - \Pr(B_i^m = 0)\}. \quad (14)$$

2.5. Algorithm

Our algorithm of major steps in computing the performance measure is formally stated below. We assume that node 1 is the root node. $\Lambda_i(n)$ is the processing rate of node i when there are n jobs in its input buffer. For a simple node, i , the processing rate of the node is independent of the number of jobs in the input buffer

and is equal to μ_i for $1 \leq n \leq N$.

Algorithm

- Step 0.** Initialize: Input network R and processing rates μ_i , $i = 1, \dots, \mathcal{M}$.
- Step 1.** Assign $\Lambda_i(n) = \mu_i$, $n = 1, \dots, N$, $i = 1, \dots, \mathcal{M}$. (Aggregation step).
- Step 2.** Select a leaf node i such that $B(i) = \phi$; if $B(i) \neq \phi$, then $B(i) \subset L(R)$.
- Step 3.** If $S(i) = \text{root node}$ then go to 6, else go to 4.
- Step 4.** Assign $j = S(i)$. Compute the throughput $\theta(n)$ and the conditional probabilities (equation (9)) for queue lengths for the network described by TSAS($n, \mu_j, \{\Lambda_i\} \cup D$), $n = 1, \dots, N$, where $D = \{\Lambda_t : t \in B(i)\}$. Assign $\Lambda_j(n) = \theta(n)$.
- Step 5.** $R = R / \{i\} \cup \{t : t \in B(i)\}$ (these nodes are removed from the network). Go to step 2.
- Step 6.** Compute the throughput and the conditional probabilities for the network described by TSAS($N, \mu_1, \{\Lambda_i\} \cup D$), where $D = \{\Lambda_t : t \in B(i)\}$. Assign $\theta_1 = \theta(N)$.
- Step 7.** (Disaggregation step) Compute the marginal queue length distribution and the matched queue length distribution (equations (9), (11)–(13)). Compute the throughput for all nodes $i \neq 1$ (using equation (14)). Assign $\theta_{\text{avg}} = \sum_{i=1}^{\mathcal{M}} \theta_i / \mathcal{M}$. Stop.

The algorithm can solve reasonably large problems. For the network shown in Fig. 2, we have solved problems, using a PC, with $N = 40$; there are more than 2 billion states in this network and an exact solution methodology is computationally intractable. In each aggregation step the algorithm solves a two-stage assembly network. In this network there is a single assembly machine fed by k inputs, where k is the number of predecessors of the assembly node in the original network. When $k = 1$ we just need to solve an $M/M/1/n$ queueing system where the arrival rates could be state-dependent. Closed-form solutions exist for this class of problem. If $k > 1$, we must solve the subnetwork numerically. The convergence of problems with small values of k is very fast owing to the sparsity of the transition rate matrix of the CTMC. For larger values of k , however, solving the subnetwork could be computationally burdensome, and one may use heuristic methods similar to those used in Lipper and Sengupta (1986) and Rao and Suri (1994). We have used an exact numerical procedure to solve each subnetwork.

3. Computational observations

In this section we present some numerical results comparing the proposed approximation with simulation. The simulation program was coded in SIMAN, and the

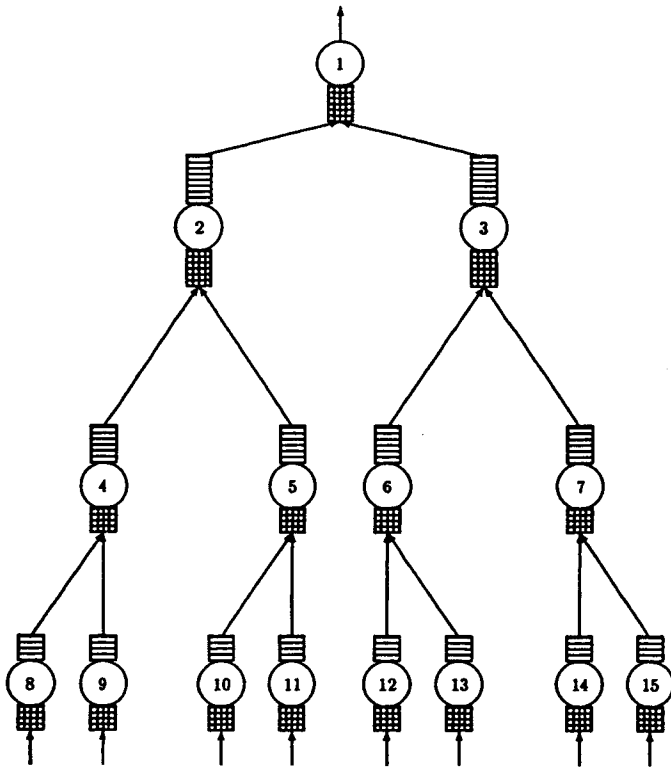


Fig. 2. A 15-machine assembly network.

algorithm was coded in Pascal. Both programs were run on a 33 MHz 486 PC. The run time for the algorithm for most problems reported here was a few seconds to a minute; it increases quadratically with N because we need to store the conditional probabilities (9). For each experiment we ran the simulation ten times, producing

Table 1. Data set 1: processing rates of machines in Fig. 1

μ_1	μ_2	μ_3	μ_4	μ_5	μ_6	μ_7	μ_8
x	y	5	z	5	5	5	5

50 000 units per run. The estimate of the actual performance measure is the mean of these ten runs. We define the percentage error in predicting the performance measure ϕ as $100(\phi_{\text{algorithm}} - \phi_{\text{simulation}})/\phi_{\text{simulation}}$. For brevity, we present results for two data sets with two different network topologies: data set 1 (Table 1) refers to Fig. 1, and data set 2 (Table 3) pertains to Fig. 2. In these tables we define θ as the expected throughput of the system (parts per unit time), and $E(B_{ji})$ is the expected number of parts in buffer $B(j, i)$. $E(B_i^m)$ is the expected length of the matched queue at the input buffer of machine i ; this represents the number of assembled kits. The unmatched queue length $E(B_{ji}^u)$ can be easily calculated from the following equation: $E(B_{ji}^u) = E(B_{ji}) - E(B_i^m)$.

In the examples reported in Tables 1–4 we observed that θ , the system throughput, is predicted with an accuracy of 96% or better. The estimation error tends to decline as N increases. In addition, systems with pronounced bottlenecks generated smaller estimation errors than homogeneous systems wherein the processing rates of all machines were identical. The estimation error for WIP queue lengths is 12% or better, and the typical error is less than 5%. The error in estimating the expected queue length tends to increase slightly as the number of stages increases, but it is independent of N and whether or not the system is homogeneous.

Table 2. Accuracy analyses for data set 1

N (x, y, z)...	12 (3,5,5)			12 (5,3,5)			12 (5,5,3)		
	Algo.	Sim.	% error	Algo.	Sim.	% error	Algo.	Sim.	% error
θ	2.954	2.955	-0.03	2.977	2.989	-0.40	2.983	2.996	-0.43
$E(B_{21})$	8.575	8.625	-0.58	1.525	1.515	0.66	1.542	1.547	0.32
$E(B_{31})$	8.187	8.297	-1.33	8.074	8.191	-1.43	8.052	8.143	-1.12
$E(B_{42})$	1.976	1.985	-0.45	9.038	9.066	-0.31	1.465	1.487	-1.48
$E(B_{52})$	1.976	1.960	0.82	9.038	9.061	-0.25	9.006	9.013	0.01
$E(B_{63})$	2.353	2.329	1.03	2.420	2.381	1.64	2.433	2.432	0.00
$E(B_{73})$	2.353	2.312	1.77	2.420	2.374	1.94	2.433	2.399	1.42
$E(B_{83})$	2.353	2.317	1.55	2.420	2.380	1.68	2.433	2.415	0.75
$E(B_{04})$	1.448	1.390	4.17	1.437	1.419	1.27	8.993	8.966	0.30
$E(B_{05})$	1.448	1.415	2.33	1.437	1.424	0.91	1.452	1.441	0.76
$E(B_{06})$	1.461	1.374	6.33	1.506	1.429	5.39	1.515	1.425	6.32
$E(B_{07})$	1.461	1.391	5.03	1.506	1.436	4.87	1.515	1.458	3.91
$E(B_{08})$	1.461	1.386	5.41	1.506	1.429	5.39	1.515	1.441	5.14
$E(B_1^m)$	7.285	7.579	-3.88	1.401	1.434	-2.30	1.411	1.443	-2.22
$E(B_2^m)$	1.373	1.367	0.44	8.427	8.455	-0.33	1.434	1.458	-1.65
$E(B_3^m)$	1.352	1.347	0.37	1.391	1.383	0.58	1.398	1.403	-0.36

Table 3. Data set 2: processing rates of machines in Fig. 2

μ_1	μ_2	μ_3	μ_4	μ_5	μ_6	μ_7	μ_8	μ_9	μ_{10}	μ_{11}	μ_{12}	μ_{13}	μ_{14}	μ_{15}
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5

Table 4. Accuracy analyses for data set 2

N	10			20			40		
	Algo.	Sim.	% error	Algo.	Sim.	% error	Algo.	Sim.	% error
θ	3.275	3.411	-3.99	3.982	4.062	-1.97	4.439	4.496	-1.27
$E(B_{2,1})$	3.070	2.931	4.74	5.977	5.972	0.08	11.80	11.18	5.55
$E(B_{3,1})$	3.070	2.944	4.28	5.977	5.781	3.39	11.80	11.34	4.06
$E(B_{4,2})$	2.777	2.786	-0.32	5.491	5.477	0.26	10.92	11.03	-1.00
$E(B_{5,2})$	2.777	2.823	-1.63	5.491	5.538	-0.85	10.92	11.28	-3.19
$E(B_{6,3})$	2.777	2.783	-0.22	5.491	5.378	2.10	10.92	11.09	-1.53
$E(B_{7,3})$	2.777	2.808	-1.10	5.491	5.516	-0.45	10.92	11.10	-1.62
$E(B_{8,4})$	2.391	2.515	-4.93	4.855	4.872	-0.35	9.713	9.950	-2.38
$E(B_{9,4})$	2.391	2.532	-5.57	4.855	5.041	-3.69	9.713	10.22	-4.96
$E(B_{10,5})$	2.391	2.486	-3.82	4.855	4.916	-1.24	9.713	10.16	-4.40
$E(B_{11,5})$	2.391	2.515	-4.93	4.855	4.990	-2.71	9.713	10.06	-3.45
$E(B_{12,6})$	2.391	2.532	-5.57	4.855	5.200	-6.63	9.713	9.810	-0.99
$E(B_{13,6})$	2.391	2.501	-4.40	4.855	5.166	-6.02	9.713	10.30	-5.70
$E(B_{14,7})$	2.391	2.509	-4.70	4.855	5.066	-4.17	9.713	10.07	-3.55
$E(B_{15,7})$	2.391	2.498	-4.28	4.855	5.074	-4.32	9.713	10.17	-4.49
$E(B_{0,8})$	1.763	1.768	-0.28	3.677	3.679	-0.05	7.565	7.830	-3.38
$E(B_{0,9})$	1.763	1.760	0.28	3.677	3.510	4.76	7.565	7.560	0.07
$E(B_{0,10})$	1.763	1.760	0.28	3.677	3.575	2.85	7.565	7.369	2.66
$E(B_{0,11})$	1.763	1.731	1.85	3.677	3.500	5.06	7.565	7.478	1.16
$E(B_{0,12})$	1.763	1.740	1.32	3.677	3.643	0.93	7.565	7.751	-2.40
$E(B_{0,13})$	1.763	1.772	-0.51	3.677	3.675	0.05	7.565	7.264	4.14
$E(B_{0,14})$	1.763	1.738	1.44	3.677	3.637	1.10	7.565	7.483	1.10
$E(B_{0,15})$	1.763	1.750	0.74	3.677	3.628	1.35	7.565	7.381	2.49
$E(B_1^m)$	1.512	1.692	-10.6	3.119	3.522	-11.4	6.363	6.480	-1.81
$E(B_2^m)$	1.573	1.735	-9.34	3.244	3.520	-7.84	6.607	7.223	-8.53
$E(B_4^m)$	1.666	1.784	-6.61	3.425	3.560	-3.79	6.871	7.229	-4.95

The sum of parts in every chain $c(1, i_l)$, where 1 is the root node and $i_l \in L$ is equal to N . For example, in Table 2 the expected number of parts in chain $c(1, 4)$ for the case when machine 1 is a bottleneck is equal to $E(B_{2,1}) + E(B_{4,2}) + E(B_{0,4}) = 8.625 + 1.985 + 11.390 = 12$. The expected number of unmatched parts waiting at the output buffer of machine 4 when machine 1 is a bottleneck is given by $E(B_{4,2}^m) = E(B_{4,2}) - E(B_2^m) = 1.985 - 1.367 = 0.618$.

Several interesting operational insights are obtained using our algorithm. The concavity of θ with respect to N can be observed in all cases. For instance, in Table 4, when N increases from 10 to 20 the throughput increases by 19%, and when N increases from 20 to 40 the throughput increases by only 10.6%. The formal proof for concavity is presented in Hazra and Seidmann (1995). Assuming the revenue generation rate to be proportional to θ , and the inventory holding cost to be proportional to the expected WIP in the system, then one

can use our algorithm to compute the profit-maximizing value of N efficiently.

Because assembly machines have more than one predecessor they can be simultaneously overloaded by some input streams and starved by others. This happens when the output buffer of one predecessor machine has accumulated several (unmatched) parts and the input buffer (matched queue) of the assembly machine is empty. In fact, every assembly is associated with a 'zero cross product' at each level of its unmatched queues. For example, in Fig. 1 there are three machines feeding assembly machine 3. At any time at most two of these machines can have a queue of unmatched parts, so at least one queue should have zero inventory. Otherwise the kit assembly is formed and transferred to the matched input queue of machine 3. Our algorithm measures the long-run effect of these synchronization and queueing delays through the expected value of the queue lengths of B_{ji}^m and B_i^m .

Proper identification of bottleneck machines is crucial for making the decisions about allocation of marginal capacity. Such identification can be difficult because in many industrial settings the mean processing rates of machines can vary greatly and thereby produce unexpected bottlenecks. Our algorithm is useful as a bottleneck analysis tool. We define a bottleneck as the machine with the highest utilization. If the utilization is equal for all machines, the machine with the longest queue defines the bottleneck. The utilization of machine i is given by θ/μ_i . In Table 2, for the first set of data, the utilization of machine 1 is 98.5% (2.955/3), and the utilization of all other machines is 59.1% (2.955/5). In this particular case it is not difficult to identify the bottleneck as machines 1, 2 and 4 for the first, second and third cases respectively. On the other hand, consider Table 3, which presents a homogeneous network with 15 machines of equal processing rates (5 parts per unit time). The utilization of all machines is equal. Clearly, in this homogeneous assembly system the overall expected queue length of the downstream machine is longer. For $N = 10$, $E(B_{08}) = 1.768$, $E(B_{84}) = 2.515$, $E(B_{42}) = 2.786$, and $E(B_{21}) = 2.931$. Our analysis shows that most of this increase in queue length is caused by a higher proportion of unmatched components.

Table 1 shows the results of experiments using machines 1, 2, and 4 as the slowest machine. For example, when machine 4 is the slowest machine there is a significant build-up of queue at the input buffer of machine 4 ($E(B_{04}) = 8.966$); this delay is caused by a capacity shortage. Interestingly, our algorithm also points to significant queue build-ups in two other 'remote' locations; these are queues of unmatched parts in the output buffers of machines 3 and 5 ($E(B_{31}^u) = 8.143 - 1.443 = 6.70$, $E(B_{52}^u) = 9.013 - 1.458 = 7.555$). These queues are the result of the 'long-distance correlation' typical of tightly coupled flow controls in closed assembly systems, which ensures that no single machine can get stochastically much ahead of all others. Similar observations can be made when either machine 1 or machine 2 is the slowest. Hence, when managing complex assembly systems one cannot use merely the queue length as an indicator of a local bottleneck. Proper diagnosis requires a performance evaluation algorithm similar to ours in order to decouple analytically the 'problems' from the 'symptoms', as illustrated above.

Comparison of the three cases in Tables 3 and 4 shows that the system throughput varies only slightly as the bottleneck is moved. This evidence is in contrast with the result in Duenyas and Hopp (1993), which showed that a bottleneck at the downstream assembly machine limits throughput more than an equivalent bottleneck in an upstream non-assembly machine. This difference may be attributable to differing network structures.

4. Conclusions

We have proposed an efficient approximate method to compute the performance measures of a closed manufacturing system with several assembly machines. It provides the basis for exploring the economic tradeoffs between the total number of jobs in a system and the leadtime performance. It can also be used in determining the minimum assembly capacity required to meet a specific leadtime target, or to study the impact of network topology options and assembly sequences on the production capacity of a plant. The results from a large number of samples, a few of which were presented here, indicate that the proposed approximation is acceptable. The error in predicting throughput decreases with N , the number of kanbans, and the throughput predicted by the algorithm slightly underestimates the throughput obtained from simulation. The error in queue lengths is somewhat higher, yet we feel that it is still acceptable.

Using the proposed approximation and the analysis leading to it we show that the WIP in assembly systems should be conceptually partitioned into two parts: unmatched and matched components; the mean queue lengths for these components provide important insights about capacity allocation. A long unmatched queue indicates a large synchronization delay, whereas a long matched queue implies a large queueing delay. A station with small production capacity will have a long queueing delay, but it will increase the synchronization delay at other stations. Hence, in assembly networks one must take into account both sorts of delay when allocating capacity. Further studies could extend our model to incorporate parallel assembly machines and multiple part types with batch production.

5. Acknowledgements

We thank Professor Paul Schweitzer from the University of Rochester and Professor Izak Duenyas and Professor Stephen Pollock from the University of Michigan for many useful suggestions. We also thank the two anonymous referees and the Departmental Editor, Professor Michael Caramanis, for their valuable feedback.

References

- Ammar, M.H. and Gershwin, S.B. (1990) Equivalence relations in queueing models of fork/join networks. *Performance Evaluation*, **10**, 233–245.
- Baker, K.R., Powell, S. and Pyke, D. (1993) Optimal allocation of work in assembly systems. *Management Science*, **39**, 101–106.
- Bhat, U.N. (1986) Finite capacity assembly like queues. *Queueing Systems*, **1**, 85–101.
- Bonomi, F. (1986) An approximate analysis for a class of assembly like queues. *Queueing Systems*, **1**, 289–309.

- Chandy, K.M., Herzog, U. and Woo, L. (1975) Parametric analysis of queueing networks. *IBM Journal of Research and Development*, **19**, 36-42.
- Dallery, Y. and Gershwin, S.B. (1992) Manufacturing flow line systems: a review of models and analytical results. *Queueing Systems*, **12**, 3-94.
- DiMascolo, M., David, R. and Dallery, Y. (1991) Modeling and analysis of assembly systems with unreliable machines and finite buffers. *IIE Transactions*, **23**, 4, 315-330.
- Duenyas, I. and Hopp, W.J. (1993) Estimating the throughput of a cyclic exponential assembly system. *Queueing Systems*, **14**, 137-157.
- Gershwin, S.B. (1991) Assembly/disassembly systems: an efficient decomposition algorithm for tree structured networks. *IIE Transactions*, **23**, 302-314.
- Gershwin, S.B. (1994) *Manufacturing Systems Engineering*, Prentice Hall, Englewood Cliffs, NJ.
- Harrison, J.M. (1973) Assembly-like queues. *Journal of Applied Probability*, **10**, 354-367.
- Hazra, J. and Seidmann, A. (1995) Performance evaluation and structural properties of closed assembly networks. Working Paper, Simon School, University of Rochester, Rochester, NY.
- Hopp, W.J. and Simon, J.T. (1989) Bounds and heuristics for assembly like queues. *Queueing Systems*, **4**, 137-156.
- Lipper, E.H. and Sengupta, B. (1986) Assembly like queues with finite capacity: bounds, asymptotics and approximations. *Queueing Systems*, **1**, 67-83.
- Liu, Y.C. and Perros, H.G. (1991a) A decomposition procedure for the analysis of a closed fork/join queueing system. *IEEE Transactions on Computer*, **40**, 365-370.
- Liu, Y.C. and Perros, H.G. (1991b) Approximate analysis of a closed fork/join model. *European Journal of Operational Research*, **53**, 382-392.
- Liu, X.G. and Buzacott, J.A. (1989) Approximate models of assembly systems with finite inventory banks. *European Journal of Operations Research*, **45**, 143-154.
- Rao, P.C. and Suri, R. (1994) Approximate queueing network models for closed fabrication/assembly systems. Working Paper, Depart-

ment of Industrial Engineering, University of Wisconsin, Madison WI.

Spearman, M.L., Hopp, W.J. and Woodruff, D.L. (1990) CONWIP: a pull alternative to kanban. *International Journal of Production Research*, **28**, 879-894.

Suri, R., Sanders, J.L. and Kamath, M. (1993) Performance evaluation of production networks, in *Handbooks in OR & MS*, Graves, S.C. et al. (eds), Elsevier Science Publishers, vol. 4, chapter 5.

Biographies

Jishnu Hazra is a Professor of Operations Management at the Indian Institute of Management in Bangalore, India. He received a B.Tech. in Mechanical Engineering from the Indian Institute of Technology and a Ph.D. from the William E. Simon Graduate School of Business Administration of the University of Rochester. His main research interest is in modeling manufacturing systems and operations management.

Abraham Seidmann is the Xerox Professor and Areas Coordinator of Computers and Information Systems, Management Science and Operations Management at the William E. Simon Graduate School of Business Administration at the University of Rochester, Rochester, NY. Professor Seidmann is the author of numerous research articles and is a Department Editor on Interdisciplinary Management Research and Applications in *Management Science*. He is also an Associate or Area Editor for *IIE Transactions*, *International Journal of Flexible Manufacturing Systems*, *Production Planning and Controls*, *Journal of Intelligent Manufacturing*, and *Production and Operations Management*. His current research and consulting activities include Strategic Manufacturing Systems, Business Process Reengineering, Information Economics, Stochastic and Performance Modelling. Professor Seidmann has consulted with many leading industrial corporations and presented research or executive seminars on four continents.

