



Analyzing closed Kanban-controlled assembly systems by iterative aggregation–disaggregation

Jishnu Hazra^a, Paul J. Schweitzer^{b,*}, Abraham Seidmann^b

^aIndian Institute of Management, Bangalore, 560076, India

^bW.E. Simon Graduate School of Business Administration, University of Rochester, Rochester, NY 14627, USA

Abstract

In recent years, many companies have begun using WIP-limiting strategies such as CONWIP, Kanban or drum-buffer-rope to control parts flows in complex production systems. This paper analyzes assembly systems with a tree structure, random processing times and a constant WIP control system. We present heuristic version of the exact aggregation–disaggregation theory for finite Markov chains to evaluate the performance of these closed Kanban-controlled assembly systems. Because the approximation is theory based, it provides a framework for further model development, with some possible extensions described in the paper. The approximation has the novel feature of doing *simultaneous multiple partitions of the state space*, in such a way that the associated aggregate transition rates are *mutually consistent*. The methodology is a novel approach towards extending aggregation ideas to fork-and-join queueing networks, and it provides several useful operational and analytical insights. Numerical comparisons with simulation show that the proposed approximation computes accurate estimates of the plant throughput. It provides a fast way to assess the performance and economic impact of changes in the total WIP level (or the number of Kanbans), or in the part routes, on the throughput rate of the assembly system. © 1999 Elsevier Science Ltd. All rights reserved.

Scope and purpose

We develop a new methodology for rapid performance analysis of assembly systems with a tree structure, random processing times, and a constant work in process Kanban control system. Extensive evaluations show that this methodology computes accurate estimates of the plant throughput. It provides a fast way for managers to assess the performance and economic impact of changes in the total WIP level (or the number of Kanbans) and in the part routes, on the overall throughput rate of the assembly system.

Keywords: Assembly systems; Kanban-controlled; Aggregation; Disaggregation

* Corresponding author.

E-mail address: schweitzer@ssb.rochester.edu (P.J. Schweitzer)

1. Introduction

Many manufacturing systems require more than one distinct input part. These machines cannot start processing until all input parts from the preceding stages are available at their input buffer. This results in synchronization delays, which make such queueing systems difficult to analyze. The multi-stage assembly system is represented by a directed tree as discussed in [1]. We use the generic term part for components and subassemblies. The part in a buffer of a downstream machine is an assembly of parts from the buffers of upstream machines. We model the detailed parts flows in the tree-structured assembly system as a continuous-time Markov chain (CTMC). We begin, therefore, with the aggregation–disaggregation (A/D) approach to solving CTMCs with large state spaces and then show how to use it for the rapid approximation of complex assembly queueing systems. We expect our results to become an important part of the process analysis activities undertaken by assembly system designers and managers. The need for our research arises from the observation that even complex queueing network models cannot capture the part mating delays, and simulation models of such systems require special effort due to the particular part flows involved. Our results show that the proposed approximation computes accurate estimates of the assembly plant throughput.

1.1. Exact aggregation–disaggregation

We consider an ergodic continuous-time Markov chain with finite state space S and transition rate matrix $\{r_{ij} | i, j \in S, i \neq j\}$. The equilibrium probabilities $[\pi_i]_{i \in S}$ (where $\pi_i = \text{Pr}[\text{in state } i]$), are the solution to the $|S|$ Kolmogoroff equations:

$$\pi_i \sum_{j \in S/i} r_{ij} = \sum_{j \in S/i} \pi_j r_{ji}; \quad i \in S \text{ (discard one)}, \quad (1.1a)$$

$$\sum_{i \in S} \pi_i = 1. \quad (1.1b)$$

Often $|S|$ is too large to permit direct solution of Eq. (1.1) either by an exact (pivoting) algorithm or even by successive approximations.

An alternative numerical approach is Takahashi's aggregation–disaggregation (A/D) procedure [2–4], in which the state space S is *partitioned* into J groups of “similar” states:

$$S = S(1) + S(2) + \dots + S(\alpha) + \dots + S(J). \quad (1.2)$$

Attention is focused upon the *aggregate equilibrium probabilities*,

$$\bar{\pi}_\alpha = \text{Pr}[\text{in group } S(\alpha)] = \sum_{i \in S(\alpha)} \pi_i, \quad 1 \leq \alpha \leq J \quad (1.3)$$

and upon the *aggregate transition rates*,

$$R_{\alpha\beta} = \sum_{i \in S(\alpha)} \left[\frac{\pi_i}{\sum_{k \in S(\alpha)} \pi_k} \right] \sum_{j \in S(\beta)} r_{ij}, \quad 1 \leq \alpha, \beta \leq J, \quad \alpha \neq \beta \quad (1.4)$$

with the interpretation

$$R_{\alpha\beta} dt + o(dt) = \Pr[\text{in } S(\beta) \text{ at time } t + dt | \text{in } S(\alpha) \text{ at time } t], \alpha \neq \beta.$$

$[R_{\alpha\beta}]$ inherits the non-negativity and ergodicity properties from $[r_{ij}]$.

The A/D approach replaces Eq. (1.1) with the equivalent quadruple set of equations consisting of Eq. (1.4); the *aggregation equations*:

$$\bar{\pi}_\alpha \sum_{\substack{\beta=1 \\ \beta \neq \alpha}}^J R_{\alpha\beta} = \sum_{\substack{\beta=1 \\ \beta \neq \alpha}}^J \bar{\pi}_\beta R_{\beta\alpha}, \quad 1 \leq \alpha \leq J \text{ (discard one)} \tag{1.5a}$$

$$\sum_{\alpha=1}^J \bar{\pi}_\alpha = 1; \tag{1.5b}$$

the *disaggregation equations* for group $S(\alpha)$, $1 \leq \alpha \leq J$:

$$\pi_i \sum_{j \in S(i)} r_{ij} = \sum_{j \in S(\alpha)/i} \pi_j r_{ji} + \sum_{\substack{\beta=1 \\ \beta \neq \alpha}}^J \bar{\pi}_\beta D_{\beta i}, \quad i \in S(\alpha) \tag{1.6}$$

and the *disaggregation rate matrix*

$$D_{\beta i} = \sum_{j \in S(\beta)} \left[\frac{\pi_j}{\sum_{k \in S(\beta)} \pi_k} \right] r_{ji}, \tag{1.7}$$

with the interpretation

$$D_{\beta i} dt + o(dt) = \Pr[\text{in } i \text{ at time } t + dt | \text{in } S(\beta) \text{ at time } t].$$

Formally, Eq. (1.5) is the set of Kolmogoroff equations for an aggregated J -state CTMC, even though the aggregated process is not Markovian. The quadruple of Eqs. (1.4)–(1.7) is just sufficient to determine the quadruple of unknowns $\{\pi_i, \bar{\pi}_\alpha, R_{\alpha\beta}, D_{\beta i}\}$. The usual iterative A/D algorithm proceeds by alternation between solving Eq. (1.5) for $\{\bar{\pi}_\alpha\}$ and solving Eq. (1.6) for $\{\pi_i\}$. Much smaller sets of linear equations are solved at each step. Upon convergence, Eq. (1.3) provides a numerical check.

The A/D framework, however, suffers from two drawbacks. First, since the size of the problem now is *even larger* than $|S|$, it is still impractical for problems with enormous state spaces. Second, if the system performance characteristics, such as throughputs, depend only on $\{\bar{\pi}_\alpha\}$, it seems cumbersome to have to deal with both $\{\pi_i\}$ and $\{\bar{\pi}_\alpha\}$ in order to compute just the latter.

Ideally one wants a *purely aggregated version* of the problem, one which involves just the $\{\bar{\pi}_\alpha\}$ and the $\{R_{\alpha\beta}\}$. This would overcome both drawbacks. But a glance at the quadruple shows that this is impossible, because the $\{\bar{\pi}_\alpha\}$ depend upon the $\{R_{\alpha\beta}\}$, which in turn depend on the $\{\pi_i\}$. The exact $\{\bar{\pi}_\alpha\}$ cannot be computed, therefore, without already knowing $\{\pi_i\}$.

1.2. Heuristic aggregation

The pursuit of a purely aggregated version, involving just $\{\bar{\pi}_\alpha\}$ and $\{R_{\alpha\beta}\}$, may however, be undertaken as an *approximation*. One estimates $\{R_{\alpha\beta}\}$ and then solves Eq. (1.5) to obtain an estimate of $\{\bar{\pi}_\alpha\}$. As the estimate of $\{R_{\alpha\beta}\}$ improves, so does the estimate of $\{\bar{\pi}_\alpha\}$.

A whole nested family of procedures can be used to estimate $\{R_{\alpha\beta}\}$, with increasing accuracy accompanied by increasing computational effort. At one extreme is solving the full quadruple set of equations with perfect accuracy. At the other extreme is a one-pass procedure, in which a *single* estimate of the conditional probabilities $[\pi_i / \sum_{k \in S(\alpha)} \pi_k]$ is inserted into Eq. (1.4) to obtain a *single* estimate of the $\{R_{\alpha\beta}\}$. In between are various heuristic estimates of $\{R_{\alpha\beta}\}$ that may involve intermediate models, typically iterative and more coarsely aggregated. These heuristics exploit structural features of the problem at hand, and no general recipe can be given for their construction.

1.3. Multiple partitions

This paper describes one such intermediate procedure as applied to closed Kanban-controlled assembly systems. It has the novel feature of doing *multiple partitions* of the state space, and it generates estimates of $\{R_{\alpha\beta}\}$ by the requirement that the various partitions generate *mutually consistent* estimates of the aggregate transition rates. This leads to a *fixed-point problem* for the $\{R_{\alpha\beta}\}$, which itself is solved by iteration. We show that the procedure works, and discuss how it is capable of even higher accuracy by additional refinement of the procedure for estimating the $\{R_{\alpha\beta}\}$.

The plan of the paper is as follows. Section 2 describes the Kanban system and our notation. Section 3 sets up a single partition of the state space S . Section 4 sets up the multiple partitions and the resulting fixed-point problem. Section 5 describes the computational procedure and results, and Section 6 offers further development of the model. Section 7 contains our summary and conclusions.

2. Closed Kanban-controlled assembly system

In this paper we study an assembly system operating under CONWIP (Constant Work In Process) control. The goal is to estimate the throughput and the expected local buffer occupancy for a given topology and a given number of Kanbans. The Kanban system dynamics can be understood from Fig. 1. The assembly system is a directed tree with M machines numbered $1, 2, \dots, M$ (here, $M = 12$). Machine f has exponentially distributed assembly times with mean assembly time $1/\mu_f$, $1 \leq f \leq M$. There is one *root* node r (here, $r = 1$), a set L of two or more *leaf* nodes (here, $L = \{7, 8, 9, 10, 11, 12\}$), and a set INT of *intermediary* nodes (here, $\text{INT} = \{2, 3, 4, 5, 6\}$).

New components are loaded at the leaf nodes, processed or assembled at each machine, and forwarded to the machine's output buffer. For instance, machine 8 takes parts from the input buffer (whose queue length is $n_{1,8}$) and, after completion, forwards them to its output buffer (whose queue length is $n_{8,5}$). That is, $n_{1,8}$ will decrease by one, and $n_{8,5}$ will increase by one. Leaf nodes start the production process of the basic platform (e.g., a computer's motherboard or a car's chassis.) These nodes do not do assembly. Note that machine 5 takes one part from each of its two input buffers (whose queue lengths are $n_{8,5}$ and $n_{9,5}$) and after assembly forwards them to its output buffer (whose queue length is $n_{5,2}$). Both $n_{8,5}$ and $n_{9,5}$ will decrease by one, and $n_{5,2}$ will increase by one. The root node, machine 1, combines subassemblies coming from nodes 2 and 3. When the root node completes assembly, a fully-assembled part exits the system, and a Kanban (authorization to

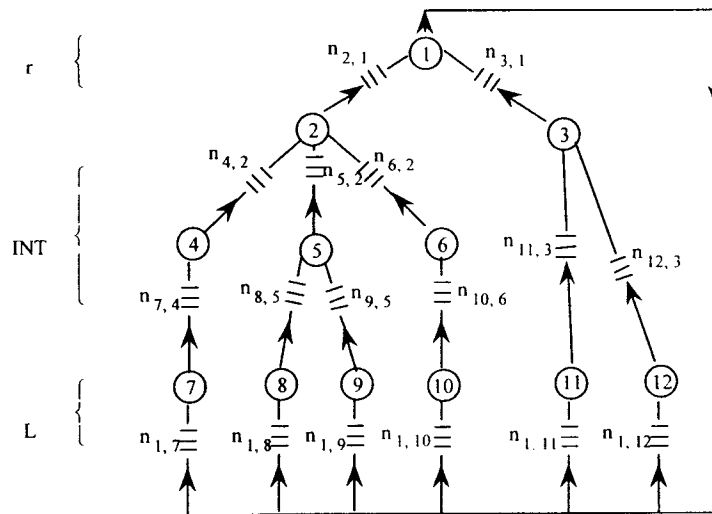


Fig. 1. Part flows and buffer structure in a 12-machine assembly system.

produce) as well as raw material is instantaneously and simultaneously delivered to all the leaf nodes. That is, $n_{2,1}$ and $n_{3,1}$ will both decrease by one, while $n_{1,7}$, $n_{1,8}$, $n_{1,9}$, $n_{1,10}$, $n_{1,11}$, and $n_{1,12}$ will all increase by one.

In this notation, n_{ij} denotes the number of parts in the buffer located downstream from i and upstream from j . n_{ij} includes both parts on queue and parts in the midst of service at j . There is sufficient buffer space that blocking never occurs. Starvation, however, can occur; for example, machine 5 cannot begin assembly unless it has at least one part in each of its input buffers, namely, when $n_{8,5} \geq 1$ and $n_{9,5} \geq 1$.

Aside from the feedback flow of Kanbans out of the root node, the topology is a directed tree. In general, the tree is unbalanced. Every machine has at least one input buffer and, aside from the root node, exactly one output buffer. Technically, this is a closed queueing network with one fork and several joins [5].

The Kanban control ensures that balanced and coordinated flows of raw materials occur at all the leaf nodes. In open systems, this is often accomplished by MRP systems. A special case of the tree system shown here is the linear assembly line, with *one* leaf node, which has been analyzed as a CONWIP (constant work-in-process) system.

For extensive discussion of models of closed Kanban systems, including CONWIP, see [1,6,7]. Other recent discussions are given in [5,8–11,14]. There is an extensive literature on open assembly systems, but not on closed ones.

A convenient modeling viewpoint is that parts *never disappear*; they are merely mated together. For example a Kanban will simultaneously bring 6 components to the 6 leaf nodes. Four of these six components will subsequently enter the root node from machine 2, while two others will subsequently enter the root node from machine 3. The completed part consists of the 6 assembled components. After exiting the root node, the completed part is “disassembled” into its 6 components, and these 6 components are delivered back to the 6 leaf nodes. The total number of components never changes.

The assembly system is started with all buffers empty except for the leaf nodes, each of which have their input buffers filled with N components. There will then always be enough components to assemble exactly N finished goods.

We let m_f denote the number of parts going upstream on a path from machine f to a leaf node. A convenient consequence of the above assumptions is *path independence*; m_f is independent of the choice of path between f and a leaf node. In Fig. 1, this means:

$$\begin{aligned}
 m_5 &= n_{8,5} + n_{1,8} = n_{9,5} + n_{1,9} \text{ (equality of two paths)} \\
 m_1 &= n_{2,1} + n_{5,2} + n_{8,5} + n_{1,8} \\
 &= n_{2,1} + n_{6,2} + n_{10,6} + n_{1,10} \\
 &= n_{3,1} + n_{11,3} + n_{1,11} \\
 &= \text{etc. (equality of 6 paths)} \\
 &= N \text{ (Note that } m_r = N\text{)}.
 \end{aligned} \tag{2.1}$$

A convenient way of writing the property of path independence is as the recursion

$$m_{S(f)} = m_f + n_{fS(f)} \quad f \neq r, \text{ where } S(f) \text{ denotes the successor to machine } f.$$

The boundary values are $m_f = n_{rf}$ if f is a leaf. For Fig. 1, by taking $f = 8$ or 9 ,

$$m_5 = m_8 + n_{8,5} = m_9 + n_{9,5}.$$

Thus the two paths between machine 5 and the two leaves 8 and 9 have the same number of parts.

Path independence is lost if any machine requires more than one part from any of its input buffers, or if initial conditions have unequal numbers of parts on the different paths. For simplicity our analysis employs these convenient conditions, but it could be undertaken without them.

The network topology is characterized by

M = number of nodes,

r = root node,

INT = set of intermediate nodes, neither root nor leaf,

L = set of leaf nodes or nodes without predecessors (leaf nodes have only one input buffer),

$S(f)$ = unique successor (downstream) node to node f , $f \neq r$ [e.g., $S(8) = 5$],

$P(f)$ = non-empty set of predecessor (upstream) nodes to node f , $f \notin L$ [e.g., $P(1) = \{2, 3\}$].

This topological information, plus N and the $\{\mu_i\}$, constitute the full set of parameters for the problem. The *full state space* S consists of the set of buffer occupancy numbers $\underline{n} = \{n_{ij} \mid 1 \leq i \leq M, j \in S(i)\}$, restricted by the equality of parts on each path and by having N parts on each leaf-to-root path.

The dynamics can be described as a continuous-time Markov chain (CTMC) with states $\underline{n} = \{n_{ij}\}$ and state space S . The transition rate for machine i ($i \in \text{INT}$) to complete an assembly is

$$\begin{cases} \mu_i & \text{if } n_{ji} \geq 1 \text{ for all } j \in P(i), \\ 0 & \text{otherwise.} \end{cases}$$

If such a transition occurs, the output buffer level $n_{iS(i)}$ increases by one while all input buffer levels $\{n_{ji} \mid j \in P(i)\}$ decrease by one.

The equilibrium probabilities of this CTMC are denoted by $\pi(n)$, $n \in S$. Since realistic problems can easily have 10^7 states, it is hopeless to try to calculate them directly from Eq. (1.1). Furthermore, they are not of central interest. The main performance measures of interest are

$$\theta_f = \text{throughput of node } f, \quad 1 \leq f \leq M,$$

calculated from

$$\theta_f = \mu_f \Pr [\text{all input buffers for node } f \text{ are non-empty;}] \tag{2.2}$$

$$U_f = \text{utilization of node } f, \quad 1 \leq f \leq M,$$

calculated from

$$U_f = \theta_f / \mu_f; \tag{2.3}$$

$$E[n_{ij}] = \text{mean occupancy level at the output buffer of node } i; \quad 1 \leq i \leq M, \quad i \neq r, \quad j = S(i)$$

$$E[n_{rj}] = \text{mean occupancy level at the input buffer to each leaf node } j, \quad j \in L.$$

These are calculated from

$$E[n_{ij}] = \sum_n n_{ij} \pi(n). \tag{2.4}$$

The problem can be made dimensionless by measuring the service rates $\{\mu_f\}$ and the throughputs $\{\theta_f\}$, as multiples of μ_r or of $\min_g \mu_g$.

The following properties, Eqs. (2.5) and (2.6), will hold for the exact model and should hold, at least as approximate equalities, for any approximation:

$$\theta_1 = \theta_2 = \dots = \theta_M. \tag{2.5}$$

All machines must have the same throughput, and this cannot exceed $\min_{1 \leq f \leq M} \mu_f$. Due to starvation, the throughput can be appreciably less than $\min_f \mu_f$.

$$\text{The path-independence condition of mean buffer levels upstream of any machine [e.g., Eq. (2.1)].} \tag{2.6}$$

A more systematic way of writing the path-independence condition is as the recursion

$$Em_{S(f)} = Em_f + En_{fS(f)}, \quad \text{all } f \neq r \tag{2.7}$$

plus the boundary values $Em_r = N$ and $Em_f = En_{rf}$ (if f is a leaf).

3. The partition associated with machine f

3.1. The aggregate states

Pick any non-leaf machine f with two or more input buffers. For simplicity the description is given as in Fig. 2 for the case in which f has 2 input buffers and two predecessors, labeled g and h .

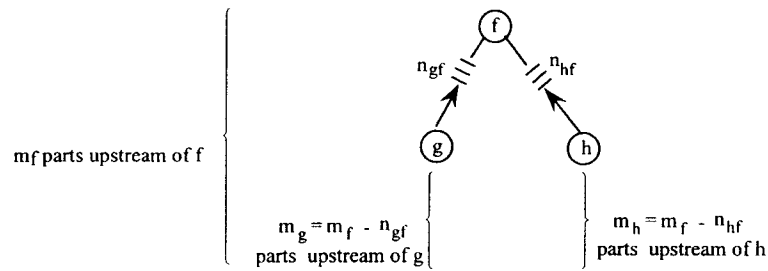


Fig. 2. Buffer content feeding an assembly mode.

[The analysis is simpler if f has one input buffer, and more cumbersome if f has 3 or more input buffers. The framework is the same in all cases.]

We generate a partition based on the triple (m_f, n_{gf}, n_{hf}) , of non-negative integers, where

$$0 \leq \max[n_{gf}, n_{hf}] \leq m_f \leq N.$$

That is, α in Eq. (1.2) is this triple

$$\alpha = (m_f, n_{gf}, n_{hf}),$$

and $S(\alpha)$ is the set of states $\underline{n} \in S$ with the given values of m_f, n_{gf}, n_{hf} . All other state information is discarded. The aggregate probability [see Eq. (1.3)] that this set of states occurs is denoted by $\bar{\pi}^{(f)}(m_f, n_{gf}, n_{hf})$, with the superscript indicating this as a partition based on machine f . $S(\alpha)$ is distinguished from $S(f)$ because α is a triple while f is an integer between 1 and M .

It is clear that any accurate analysis of machine f must contain at least information about *both* of its input buffer levels n_{gf} and n_{hf} . Carrying some additional upstream information via m_f is useful because it supplies $m_g = m_f - n_{gf}$ and hence some partial information about how likely g is to complete a service, thereby increasing n_{gf} by one.

3.2. Aggregate transition rates

According to Eq. (1.5), for the purpose of computing aggregate transition rates, the changes in the triple $\alpha = (m_f, n_{gf}, n_{hf})$ behave as a CTMC. The four possible transitions for the triple are as follows for $f \in \text{INT}$:

(a) *Machine f has a service completion*

Next triple $\beta: m_f, n_{gf}, n_{hf}$ all decrease by one. The aggregate transition rate " $R_{\alpha\beta}$ " is $\mu_f I(n_{gf} \geq 1 \text{ and } n_{hf} \geq 1)$ and is known. Here $I(E)$ is the indicator function, 1 if E holds and 0 otherwise.

(b) *Machine g has a service completion*

Next triple $\beta: n_{gf}$ increases by one, m_f and n_{hf} are unchanged.

Case 1: g is a leaf. The aggregate transition rate " $R_{\alpha\beta}$ " is $\mu_g I(m_f - n_{gf} \geq 1)$ and is known.

Case 2: g is not a leaf. The aggregate rate " $R_{\alpha\beta}$ " is unknown and will have to be estimated.

Denote the exact value by $\bar{\mu}^{(g)}(m_f, n_{gf}, n_{hf})$. The expression $\bar{\mu}^{(g)}(m_f, n_{gf}, n_{hf})$ is to be interpreted as the effective service rate of g as seen by f when in the aggregate state $\alpha = (m_f, n_{gf}, n_{hf})$. The rate is given by Eq. (1.4) as

$$\bar{\mu}^{(g)}(m_f, n_{gf}, n_{hf}) = \mu_g \Pr [\text{all of } g\text{'s input buffers are non-empty} | m_f, n_{gf}, n_{hf}]$$

In both cases, the rate is zero unless $m_g = m_f - n_{gf} \geq 1$.

(c) *Machine h has a service completion*

Analogous to (b): if h is not a leaf, there is an unknown effective service rate $\bar{\mu}^{(h)}(m_f, n_{gf}, n_{hf})$ for transition to a new state where n_{hf} increases by one.

(d) *Kanban arrives simultaneously at all the leaf nodes*

The next triple β has m_f increased by one, with n_{gf} and n_{hf} unchanged. The aggregate rate “ $R_{\alpha\beta}$ ” is unknown and must be estimated. We denote the exact aggregate transition rate “ $R_{\alpha\beta}$ ” as $\bar{\gamma}^{(f)}(m_f, n_{gf}, n_{hf})$. It is to be interpreted as the arrival rate of Kanbans as seen by f while in aggregate state $\alpha = (m_f, n_{gf}, n_{hf})$.

The rate is given by Eq. (1.4) as

$$\bar{\gamma}^{(f)}(m_f, n_{gf}, n_{hf}) = \mu_r \Pr[\text{all input buffers to root node are non-empty} | m_f, n_{gf}, n_{hf}].$$

Of course this rate vanishes if $f \neq r$ and $m_f = N$, because the root node is starved for parts coming from machine f .

If $f \in \text{INT}$, the Kanban arrival rate $\bar{\gamma}^{(f)}(m_f, n_{gf}, n_{hf})$ is unknown and must be estimated. If $f = r$, the Kanban arrival rate is known exactly:

$$\bar{\gamma}^{(r)}(m_r = N, n_{gr}, n_{hr}) = \mu_r I(n_{gr} \geq 1 \text{ and } n_{hr} \geq 1).$$

The aggregate CTMC associated with a non-leaf machine f has as its states the triples (m_f, n_{gf}, n_{hf}) with $0 \leq \max[n_{gf}, n_{hf}] \leq m_f \leq N$, and it involves some unknown aggregate transition rates $\bar{\mu}^{(g)}$, $\bar{\mu}^{(h)}$ and $\bar{\gamma}^{(f)}$. If these rates were known exactly, solution of this aggregate problem (1.5) would give exact values for the aggregate equilibrium probabilities:

$$\bar{\pi}^{(f)}(m_f, n_{gf}, n_{hf}) = \Pr[\text{in a state with given values of } m_f, n_{gf}, n_{hf}].$$

3.3. Approximate aggregate transition rates

The aggregate transition rates are unknown and must be estimated. We denote estimates with a^* , namely $\bar{\mu}^{(g)*}$, $\bar{\gamma}^{(f)*}$, $\bar{\pi}^{(f)*}$, θ_f^* , $\Pr[\]^*$, En_{gf}^* , and En_{hf}^* . We use approximations of the form (3.1) and (3.5) below. Specifics about $\bar{\mu}^{(g)*}$ and $\bar{\gamma}^{(f)*}$ are given in Section 4.1.

The key approximation for $\bar{\mu}^{(g)}$ is

$$\bar{\mu}^{(g)}(m_f, n_{gf}, n_{hf}) \cong \bar{\mu}^{(g)*}(m_f - n_{gf}); \quad g \in \text{INT} \tag{3.1}$$

i.e., the effective service rate at g depends only on m_g . If g is a leaf node, Eq. (3.1) is replaced by the exact result:

$$\bar{\mu}^{(g)}(m_f, n_{gf}, n_{hf}) = \mu_g I(m_f - n_{gf} \geq 1), \quad g \in L.$$

We know $\bar{\mu}^{(g)}(m_f, n_{gf}, n_{hf}) = 0$ if $m_g = 0$, and we therefore require that

$$\bar{\mu}^{(g)*}(m_g) = 0 \quad \text{if } m_g = 0. \quad (3.2)$$

In addition, since

$$\bar{\mu}^{(g)}(m_f, n_{gf}, n_{hf}) = \mu_g \Pr[g \text{ busy} | m_f, n_{gf}, n_{hf}], \quad (3.3)$$

we know that $\bar{\mu}^{(g)}(m_f, n_{gf}, n_{hf}) \leq \mu_g$, and we therefore require that

$$\bar{\mu}^{(g)*}(m_g) \leq \mu_g, \quad m_g = 1, 2, \dots, N. \quad (3.4)$$

The key approximation for $\bar{\gamma}^{(f)}$ is

$$\begin{aligned} \bar{\gamma}^{(f)}(m_f, n_{gf}, n_{hf}) &\cong \bar{\gamma}^{(f)*}(m_f) \quad f \neq r, \\ \bar{\gamma}^{(f)}(m_f, n_{gf}, n_{hf}) &= \mu_r I(n_{gr} \geq 1 \text{ and } n_{hr} \geq 1) \quad f = r, \end{aligned} \quad (3.5)$$

i.e., the effective Kanban arrival rate seen by $f \neq r$ depends only on m_f .

Since $\bar{\gamma}^{(f)}(n_f, n_{gf}, n_{hf}) = 0$ when $f \neq r$ and $m_f = N$ (the root machine is starved for parts from machine f), we therefore require that

$$\bar{\gamma}^{(f)*}(N) = 0, \quad f \neq r. \quad (3.6)$$

Since

$$\bar{\gamma}^{(f)}(m_f, n_{gf}, n_{hf}) = \mu_r \Pr[r \text{ busy} | m_f, n_{gf}, n_{hf}] \quad (3.7)$$

(The Kanban arrival rate = the completion rate at the root), we know that $\bar{\gamma}^{(f)}(m_f, n_{gf}, n_{hf}) \leq \mu_r$, and we therefore require that

$$\bar{\gamma}^{(f)*}(m_f) \leq \mu_r. \quad (3.8)$$

3.4. Special cases for f

If machine f has only one input buffer, then the partition is on the pair (m_f, n_{gf}) . The analysis is simpler because transitions of type (c) are absent.

If f is the root machine r , the analysis is somewhat simpler because $m_r = N$, so only n_{gr} and n_{hr} enter the CTMC. In this case, transitions (a) and (d) coincide and have rate $\mu_r I(n_{gr} \geq 1 \text{ and } n_{hr} \geq 1)$ to a new triple where both n_{gr} and n_{hr} have decreased by one.

3.5. Model outputs

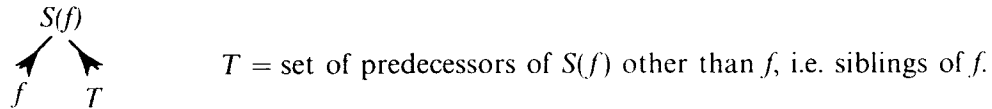
Given these estimated transition rates $\{\bar{\mu}^{(g)*}, \bar{\mu}^{(h)*}, \bar{\gamma}^{(f)*}\}$, we can solve the aggregate CTMC (1.5) and obtain estimates $\bar{\pi}^{(f)*}(m_f, n_{gf}, n_{hf})$ of the exact aggregate equilibrium probabilities $\bar{\pi}^{(f)}(m_f, n_{gf}, n_{hf})$ [= “ $\bar{\pi}_x$ ”].

3.6. Byproducts of partition at f

As by products, we can calculate estimates of throughputs and mean queue lengths from Eqs. (3.9) and (3.10):

$$\begin{aligned} \theta_f^* &= \text{estimated throughput at } f \\ &= \mu_f \Pr[n_{gf} \geq 1 \text{ and } n_{hf} \geq 1]^* \quad [\text{cf. Eq. (2.2)}] \\ &= \mu_f \sum_{m_f=1}^N \sum_{n_{gf}=1}^{m_f} \sum_{n_{hf}=1}^{m_f} \pi^{(f)*}(m_f, n_{gf}, n_{hf}) \quad f \in L. \end{aligned} \tag{3.9a}$$

If f is a leaf node, there is no aggregate model such as in Fig. 2 to use for computing θ_f , as in Eq. (3.9a). However, f is part of the aggregate model



associated with $S(f)$ and we may use

$$\theta_f^* = \theta_{S(f)}^*, \quad f \in L. \tag{3.9b}$$

En_{gf}^* = estimated queue length at output buffer of g

$$= \sum_{m_f=0}^N \sum_{n_{gf}=0}^{m_f} \sum_{n_{hf}=0}^{m_f} n_{gf} \bar{\pi}^{(f)*}(m_f, n_{gf}, n_{hf}). \tag{3.10}$$

4. Mutually consistent rates

4.1. Estimating the aggregate rates

At this point we have constructed a partition and an aggregate model for *one* non-leaf machine f , with predecessors g and h , with model inputs $\bar{\mu}^{(g)*}$, $\bar{\mu}^{(h)*}$ and $\bar{\gamma}^{(f)*}$, and with model outputs $\bar{\pi}^{(f)*}(n_f, n_{gf}, n_{hf})$, θ_f^* , En_{gf}^* , and En_{hf}^* . Now consider performing a similar partition on *every* non-leaf machine. We must provide the model-inputs in a *self-consistent* way. This will lead to a joint fixed-point problem [Eqs. (4.1)–(4.3) below] for the full set of unknown rates,

$$\{\bar{\mu}^{(f)*}(m_f), 1 \leq m_f \leq N\} + \{\bar{\gamma}^{(f)*}(m_f), 0 \leq m_f \leq N - 1\} \quad \text{for all } f \in \text{INT}.$$

The number of unknowns is $2N|\text{INT}| = O(NM)$, and the storage requirements are modest for realistic problems.

4.2. Estimating the effective service rates

The effective service rates are estimated as

$$\begin{aligned} \bar{\mu}^{(f)*}(m_f) &= \text{service completion rate at machine } f, \text{ given that there are } m_f \text{ jobs upstream of } f \\ &= \mu_f \Pr[n_{gf} \geq 1 \text{ and } n_{hf} \geq 1 | m_f]^*, \end{aligned}$$

so

$$\begin{aligned} \bar{\mu}^{(f)*}(m_f) &= 0, \quad m_f = 0 \\ \bar{\mu}^{(f)*}(m_f) &= \frac{\mu_f \sum_{n_{gf}=1}^{m_f} \sum_{n_{hf}=1}^{m_f} \bar{\pi}^{(f)*}(m_f, n_{gf}, n_{hf})}{\sum_{n_{gf}=0}^{m_f} \sum_{n_{hf}=0}^{m_f} \bar{\pi}^{(f)*}(m_f, n_{gf}, n_{hf})}, \quad 1 \leq m_f \leq N \\ f \in \text{INT}, P(f) &= \{g, h\}. \end{aligned} \quad (4.1)$$

This automatically satisfies Eqs. (3.2) and (3.4).

4.3. Estimating the Kanban arrival rate

The Kanban arrival rate, $\bar{\gamma}^{(g)*}(m_g)$, is estimated by the consistency requirement that g and its successor $f = S(g)$ “see” the same expected Kanban arrival rate, under the same conditions, namely the same m_g :

$$\begin{aligned} \bar{\gamma}^{(g)*}(m_g) &= E[\bar{\gamma}^{(f)*}(m_f) | m_g], \quad 0 \leq m_g \leq N, f = S(g) \\ &= \begin{cases} 0 & m_g = N \\ \sum_{m_r=m_g}^N \bar{\gamma}^{(f)*}(m_f) \text{Pr}[m_f | m_g]^* & f \neq r, g \in \text{INT}, 0 \leq m_g \leq N - 1. \end{cases} \end{aligned}$$

The last expression is

$$\begin{aligned} \bar{\gamma}^{(g)*}(m_g) &= 0, \quad m_g = N, \\ \bar{\gamma}^{(g)*}(m_g) &= \frac{\sum \sum \sum \bar{\gamma}^{(f)*}(m_f) \bar{\pi}^{(f)*}(m_f, n_{gf}, n_{hf})}{\sum \sum \sum \bar{\pi}^{(f)*}(m_f, n_{gf}, n_{hf})}, \quad 0 \leq m_g \leq N - 1, \text{ where } f = S(g) \neq r. \end{aligned} \quad (4.2)$$

The triple sums are taken over all triples (m_f, n_{gf}, n_{hf}) which satisfy $m_g \leq m_f \leq N$, $0 \leq n_{gf} \leq m_f$, $0 \leq n_{hf} \leq m_f$, $m_f - n_{gf} = m_g$.

When $f = r$ and $g \in P(r)$, Eq. (4.2) is replaced by

$$\begin{aligned} \bar{\gamma}^{(g)*}(m_g) &= \mu_r \text{Pr}[\text{machine } r \text{ is busy} | m_g]^* \\ &= \mu_r \text{Pr}[n_{gr} \geq 1 \text{ and } n_{hr} \geq 1 | m_g]^* \\ &= \begin{cases} 0 & m_g = N, \\ \frac{\mu_r \sum_{k=1}^{N-m_g} \bar{\pi}^{(r)*}(m_r = N, n_{gr} = N - m_g, n_{hr} = k)}{\sum_{k=0}^{N-m_g} \bar{\pi}^{(r)*}(m_r = N, n_{gr} = N - m_g, n_{hr} = k)} & 0 \leq m_g \leq N - 1. \end{cases} \end{aligned} \quad (4.3)$$

Note that Eq. (4.3) used the exact expression $\bar{\gamma}^{(r)}(m_r, n_{gr}, n_{hr}) = \mu_r I(n_{gr} \geq 1 \text{ and } n_{hr} \geq 1)$ to evaluate $\bar{\gamma}^{(g)*}$. Eqs. (4.2) and (4.3) automatically satisfy Eq. (3.6). Eq. (4.3) satisfies (3.8). Induction on the nodes, proceeding upstream from the root, then shows that Eq. (4.2) satisfies (3.8).

4.4. Overall fixed-point problem

Eqs. (4.1)–(4.3) for all $f \in \text{INT} + \{r\}$ provide a set of simultaneous relationships among the unknowns

$$\{\bar{\mu}^{(f)*}(m_f), 1 \leq m_f \leq N, f \in \text{INT}\} + \{\bar{\gamma}^{(f)*}(m_f), 0 \leq m_f \leq N - 1, f \in \text{INT}\}.$$

They constitute a *fixed-point problem* for these unknowns. As is the case for other nodal decomposition schemes, this fixed-point set can be solved by successive approximations:

- Initialization: $\bar{\mu}^{(f)*}(m_f) = \mu_f I(m_f \geq 1), f \in \text{INT} + \{r\}, 0 \leq m_f \leq N,$
 $\bar{\gamma}^{(f)*}(m_f) = \mu_r I(m_f < N), f \in \text{INT} + \{r\}, 0 \leq m_f \leq N.$
- Loop step: pick next machine $f \in \text{INT} + \{r\}$ and generate partition associated with f
 retrieve the aggregate rates $\bar{\gamma}^{(f)*}$ and $\{\bar{\mu}^{(g)*}, g \in P(f)\}$
 solve aggregate CTMC for $\bar{\pi}^{(f)*}$
 revise $\bar{\mu}^{(f)*}$ [if $f \in \text{INT}$] by Eq. (4.1)
 revise $\bar{\gamma}^{(g)*}$ [if $g \in P(f)$ such that $g \notin L$] by Eq. (4.2) or Eq. (4.3)
 revise θ_f^* by Eq. (3.9)
 revise $\text{En}_{g_f^*}$ for all $g \in P(f)$ by Eq. (3.10)
- Termination test: if revised values have not converged, return to loop step.

This algorithm typically converges in a few score iterations.

The most time-consuming part is the repeated solution of the aggregate CTMCs. We did not explore acceleration by exploiting the skip-free behavior of m_f , i.e., a generalized birth and death process. The aggregate CTMCs have $O(N^{t+1})$ states where $t = \max_f |P(f)|$. Even with small t , this puts a significant restriction on N .

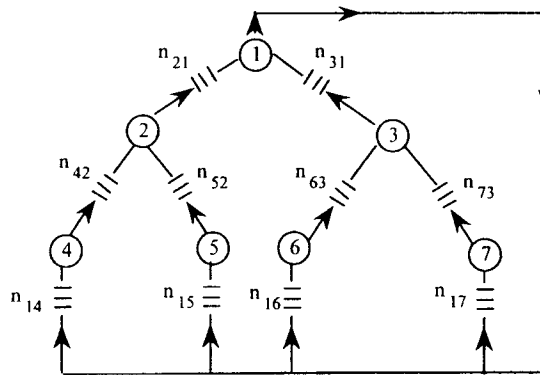
When one moves onward from the current choice of f to the next, the $\bar{\pi}^{(f)*}$ can be discarded. Only the $\{\bar{\mu}^{(f)*}\}$ and $\{\bar{\gamma}^{(f)*}\}$ need be carried forward, so the algorithm's storage requirements are only $O(MN)$. Storing the $\bar{\pi}^{(f)*}$ has the merit of providing a good starting point for the iterative solution of this aggregate CTMC when one returns to it, however, thus reducing computer time.

The scheme merits the name iterative aggregation–disaggregation because it alternates between computing the aggregate parameters $\bar{\mu}^{(f)*}$ and $\bar{\gamma}^{(f)*}$, and solving the detailed Kolmogoroff equations for one partition, $\{\pi^{(f)*}(m_f, n_{gf}, n_{hf})\}$. Strictly speaking, both of these steps are at the aggregate level, but the former is more highly aggregated than the latter.

Upon termination, we immediately have the performance measures $\{\theta_f^*\}$ and $\{\text{En}_{g_f^*}\}$. Other performance measures can be computed from $\bar{\pi}^{(f)*}$.

5. Computational results

We start by presenting our results for a 7-machine assembly system with binary tree topology (Fig. 3). It was evaluated both by simulation and by the aggregation–disaggregation algorithm for populations $N = 6, 9, 12$. Table 1 displays the initial numerical results. The principal observations are that the algorithm successfully computes throughput to within 1–2%. It has much lower accuracy (up to 30% error) for predicting expected queue lengths. From a practical system-design perspective, it is important to note that the absolute error in the mean queue length hardly exceeds

Fig. 3. The experimental seven-machine assembly system. Every $\mu_i = 1$.Table 1
Numerical results for test case

N	6			9			12		
	A/D	Sim	%error ^a	A/D	Sim	%error ^a	A/D	Sim	%error ^a
θ^c	0.658	0.664	- 1.0	0.750	0.750	0.0	0.804	0.801	0.4
$En_{21} = En_{31}^b$	2.220	2.360	- 6.0	3.270	3.490	- 3.5	4.320	4.630	- 7.0
$En_{42} = En_{52} = En_{63} = En_{73}^b$	2.600	2.140	22.0	3.900	3.210	22.0	5.350	4.270	23.0
$En_{14} = En_{15} = En_{16} = En_{17}^b$	1.180	1.500	- 22.0	1.820	2.300	- 20.0	2.430	3.080	- 21.0

Note: θ = throughput, En_{ij} = expected occupancy in output buffer of machine i .

^a %error = 100(algorithm - simulation)/simulation.

^b Mean queue lengths are given as average of values at these buffers, which should be identical.

^c θ = average of the θ_f , which should be identical.

one part. We attribute this inaccuracy to the inability of the aggregate model to describe correlated behavior among machines. This defect is common to most single-node decomposition methods [12]. This may not be a major practical issue as here we deal with closed queueing systems where buffer occupancies are also controlled through the size of the population.

The assembly system in Fig. 3 has four loops of Kanbans (and parts). We expect, therefore, four relationships of the form $En_{21} + En_{42} + En_{14} = N$. These held within 1%. The solution violated Eq. (2.5), but the variation among the θ_f^* was only about 1%. The average of the θ_f^* seems to be a good estimate of θ . Similarly, the solution violated Eq. (2.7), but the variation among different paths was small as well. Section 6.2 sketches one way to force equalities in (2.5) and (2.7).

We find that the assembly system throughputs are only 2/3 as large as the ideal value, $\min_f\{\mu_i\} = 1$, for $N = 6$. That happens because of inherent delays among mating parts as evidenced by the increased buffer occupancies in front of those machines. The results in Table 1 show that the system throughput increases in a concave fashion to 4/5 of the ideal value as we double the number of Kanbans ($N = 12$).

Table 2
Input data for the seven-machine network in Fig. 1

Data set	μ_1	μ_2	μ_3	μ_4	μ_5	μ_6	μ_7
1	1	1	1	1	1	1	1
2	1	0.8	0.8	0.8	0.8	0.8	0.8
3	1	0.8	1	1	1	0.8	0.8
4	1	1	1	0.8	0.8	0.8	0.8

Table 3
Input data for the six-machine network in Fig. 4

Data set	μ_1	μ_2	μ_3	μ_4	μ_5	μ_6
1	1	1	1	1	1	1
2	1	1	0.8	1	1	1
3	1	1	0.8	1	1	0.8
4	1	1	1	0.8	0.8	1

Table 4
Input data for the five-machine network in Fig. 5

Data set	μ_1	μ_2	μ_3	μ_4	μ_5
1	1	1	1	1	1
2	1	0.8	0.8	1	1
3	1	1	1	0.8	0.8

Table 5
Input data for the five-machine network in Fig. 6

Data set	μ_1	μ_2	μ_3	μ_4	μ_5
1	1	1	0.8	0.8	0.8
2	1	1	0.8	1	0.8
3	1	1	0.8	1	1
4	1	1	1	1	1

We conducted extensive computations to better understand the accuracy performance of the algorithm. The results reported here are based on four different network topologies (Figs. 1 and 4–6). For each topology, three or four different data sets are used with three different values of N , the number of Kanbans ($N = 6, 9$ and 12). Kanban flows from the root node (always numbered 1 in the diagram) to the leaf nodes. The data sets are given by Tables 2–5. Our results in Tables 6–19

Table 6
Results for the seven-machine network and Data set 2 in Table 2

	N = 6			N = 9			N = 12		
	Algo	Sim	% Error	Algo	Sim	% Error	Algo	Sim	% Error
θ	0.540	0.558	- 3.23	0.613	0.626	- 2.08	0.651	0.666	- 2.25
$E(B_{2,1})$	1.977	2.048	- 3.47	2.803	2.954	- 5.11	3.602	3.807	- 5.38
$E(B_{3,1})$	1.977	1.968	0.46	2.803	2.735	2.49	3.602	3.501	2.88
$E(B_{4,2})$	2.751	2.347	17.21	4.181	3.570	17.11	5.661	4.850	16.72
$E(B_{5,2})$	2.751	2.258	21.83	4.181	3.421	22.22	5.661	4.595	23.20
$E(B_{6,3})$	2.751	2.338	17.66	4.181	3.576	16.92	5.661	4.860	16.48
$E(B_{7,3})$	2.751	2.386	15.30	4.181	3.721	12.36	5.661	5.050	12.10
$E(B_{1,4})$	1.275	1.606	- 20.61	2.008	2.477	- 18.93	2.727	3.344	- 18.45
$E(B_{1,5})$	1.275	1.694	- 24.73	2.008	2.625	- 23.50	2.727	3.598	- 24.21
$E(B_{1,6})$	1.275	1.695	- 24.78	2.008	2.689	- 25.33	2.727	3.639	- 25.06
$E(B_{1,7})$	1.275	1.647	- 22.59	2.008	2.544	- 21.07	2.727	3.449	- 20.93

Table 7
Results for the seven-machine network and Data set 3 in Table 2

	N = 6			N = 9			N = 12		
	Algo	Sim	% Error	Algo	Sim	% Error	Algo	Sim	% Error
θ	0.587	0.600	- 2.17	0.662	0.670	- 1.19	0.707	0.708	- 0.14
$E(B_{2,1})$	2.235	2.373	- 5.82	3.239	3.544	- 8.61	4.108	4.735	- 13.24
$E(B_{3,1})$	1.934	1.941	- 0.36	2.760	2.713	1.73	3.628	3.409	6.42
$E(B_{4,2})$	2.804	2.445	14.68	4.363	3.808	14.57	6.091	5.218	16.73
$E(B_{5,2})$	2.804	2.389	17.37	4.363	3.727	17.06	6.091	5.149	18.29
$E(B_{6,3})$	2.544	2.023	25.75	3.753	2.946	27.39	4.849	3.822	26.87
$E(B_{7,3})$	2.544	2.100	21.14	3.753	3.103	20.95	4.849	4.079	18.88
$E(B_{1,4})$	0.962	1.182	- 18.61	1.402	1.648	- 14.93	1.800	2.047	- 12.07
$E(B_{1,5})$	0.962	1.238	- 22.29	1.402	1.729	- 18.91	1.800	2.117	- 14.97
$E(B_{1,6})$	1.523	2.037	- 25.23	2.489	3.341	- 25.50	3.526	4.768	- 26.05
$E(B_{1,7})$	1.523	1.959	- 22.26	2.489	3.185	- 21.85	3.526	4.512	- 21.85

show that the algorithm computes the system throughput with an error of 5% or less in all cases. That error was not affected by the value of N . Most of the estimates for the buffer occupancy were accurate within 30%, or better and within roughly 1 job in absolute value. Estimation errors for buffer occupancies were greater for upstream machines than for downstream machines. Again, the magnitude of that estimation error was not necessarily correlated with N . In all cases, increasing N results in a concave increase in the system throughput θ . The effect of changes in N on θ depends of the topology of the assembly system, as can be seen by comparing the two five-machine topologies studied.

The results of our comparative study are presented in Tables 6-19.

Table 8
Results for the seven-machine network and Data set 4 in Table 2

	<i>N</i> = 6			<i>N</i> = 9			<i>N</i> = 12		
	Algo	Sim	% Error	Algo	Sim	% Error	Algo	Sim	% Error
θ	0.575	0.592	− 2.87	0.650	0.663	− 1.96	0.694	0.703	− 1.28
$E(B_{2,1})$	2.063	2.133	− 3.28	2.956	3.072	− 3.78	3.808	3.925	− 2.98
$E(B_{3,1})$	2.063	2.074	− 0.53	2.956	2.919	1.27	3.808	3.707	2.72
$E(B_{4,2})$	2.479	2.005	23.64	3.656	2.953	23.81	4.799	3.829	25.33
$E(B_{5,2})$	2.479	1.914	29.52	3.656	2.741	33.38	4.799	3.552	35.11
$E(B_{6,3})$	2.479	1.960	26.48	3.656	2.861	27.79	4.799	3.698	29.77
$E(B_{7,3})$	2.479	2.031	22.06	3.656	3.021	21.02	4.799	3.995	20.13
$E(B_{1,4})$	1.460	1.862	− 21.59	2.391	2.975	− 19.63	3.397	4.246	− 20.00
$E(B_{1,5})$	1.460	1.953	− 25.24	2.391	3.187	− 24.98	3.397	4.523	− 24.89
$E(B_{1,6})$	1.460	1.966	− 25.74	2.391	3.220	− 25.75	3.397	4.596	− 26.09
$E(B_{1,-})$	1.460	1.895	− 22.96	2.391	3.061	− 21.89	3.397	4.298	− 20.96

Table 9
Results for the six-machine network of Fig. 4 and Data set 1 in Table 3

	<i>N</i> = 6			<i>N</i> = 9			<i>N</i> = 12		
	Algo	Sim	% Error	Algo	Sim	% Error	Algo	Sim	% Error
θ	0.691	0.683	1.17	0.775	0.766	1.17	0.825	0.813	1.48
$E(B_{2,1})$	1.910	2.216	− 13.81	2.842	3.309	− 14.11	3.822	4.414	− 13.41
$E(B_{3,1})$	2.539	2.751	− 7.71	3.664	4.040	− 9.31	4.699	5.371	− 12.51
$E(B_{4,2})$	2.784	2.235	24.56	4.161	3.357	23.95	5.552	4.453	24.68
$E(B_{5,2})$	2.784	2.175	28.00	4.161	3.263	27.52	5.552	4.353	27.54
$E(B_{6,3})$	1.803	1.639	10.01	2.719	2.489	9.24	3.636	3.319	9.55
$E(B_{1,4})$	1.304	1.549	− 15.82	1.996	2.335	− 14.52	2.624	3.134	− 16.27
$E(B_{1,5})$	1.304	1.609	− 18.96	1.996	2.428	− 17.79	2.624	3.234	− 18.86
$E(B_{1,6})$	1.642	1.611	1.92	2.604	2.470	5.43	3.656	3.311	10.42

6. Further refinements

We sketch several ways that the accuracy of the model can be improved with increased computational effort. This demonstrates the flexibility of the A/D framework for developing whole families of approximations.

6.1. More-detailed partitions

The current partitioning (m_f, n_{g_f}, n_{h_f}) on f and its immediate predecessors, g and h , does not provide information either upstream of g and h or downstream of f . A finer partition would employ

Table 10
Results for the six-machine network of Fig. 4 and Data set 2 in Table 3

	N = 6			N = 9			N = 12		
	Algo	Sim	% Error	Algo	Sim	% Error	Algo	Sim	% Error
θ	0.654	0.647	1.08	0.728	0.719	1.25	0.768	0.755	1.72
$E(B_{21})$	2.335	2.579	- 9.46	3.695	4.131	- 10.55	5.195	5.942	- 12.57
$E(B_{31})$	2.006	2.154	- 6.87	2.650	2.822	- 6.09	3.133	3.248	- 3.54
$E(B_{42})$	2.535	2.022	25.37	3.661	2.881	27.07	4.718	3.579	31.82
$E(B_{52})$	2.535	1.969	28.75	3.661	2.798	30.84	4.718	3.490	35.19
$E(B_{63})$	2.498	2.401	4.04	4.121	4.102	0.46	5.888	6.176	- 4.66
$E(B_{14})$	1.130	1.399	- 19.23	1.643	1.989	- 17.40	2.087	2.479	- 15.81
$E(B_{15})$	1.130	1.452	- 22.18	1.643	2.072	- 20.70	2.087	2.568	- 18.73
$E(B_{16})$	1.483	1.445	2.63	2.220	2.076	6.94	2.971	2.576	15.33

Table 11
Results for the six-machine network of Fig. 4 and Data set 3 in Table 3

	N = 6			N = 9			N = 12		
	Algo	Sim	% Error	Algo	Sim	% Error	Algo	Sim	% Error
θ	0.623	0.615	1.30	0.689	0.681	1.17	0.722	0.714	1.12
$E(B_{21})$	2.638	2.892	- 8.78	4.292	4.721	- 9.09	6.139	6.823	- 10.02
$E(B_{31})$	1.587	1.743	- 8.95	1.956	2.202	- 11.17	2.232	2.497	- 10.61
$E(B_{42})$	2.352	1.844	27.55	3.303	2.541	29.99	4.136	3.060	35.16
$E(B_{52})$	2.352	1.796	30.96	3.303	2.455	34.54	4.136	3.003	37.73
$E(B_{63})$	2.270	2.125	6.82	3.506	3.389	3.45	4.769	4.737	0.68
$E(B_{14})$	1.009	1.265	- 20.24	1.405	1.738	- 19.16	1.725	2.117	- 18.52
$E(B_{15})$	1.009	1.313	- 23.15	1.405	1.824	- 22.97	1.725	2.174	- 20.65
$E(B_{16})$	2.133	2.133	0.00	3.534	3.409	3.67	4.999	4.766	4.89

Table 12
Results for the six-machine network of Fig. 4 and Data set 4 in Table 3

	N = 6			N = 9			N = 12		
	Algo	Sim	% Error	Algo	Sim	% Error	Algo	Sim	% Error
θ	0.625	0.627	- 0.32	0.691	0.696	- 0.72	0.729	0.731	- 0.27
$E(B_{21})$	1.372	1.631	- 15.88	1.799	2.158	- 16.64	2.179	2.548	- 14.48
$E(B_{31})$	3.113	3.236	- 3.80	4.866	5.067	- 3.97	6.713	7.217	- 6.98
$E(B_{42})$	2.817	2.232	26.21	4.158	3.347	24.23	5.304	4.398	20.60
$E(B_{52})$	2.817	2.134	32.01	4.158	3.105	33.91	5.304	4.045	31.12
$E(B_{63})$	1.525	1.397	9.16	2.148	1.980	8.48	2.748	2.418	13.65
$E(B_{14})$	1.811	2.138	- 15.29	3.042	3.495	- 12.96	4.516	5.054	- 10.65
$E(B_{15})$	1.811	2.236	- 19.01	3.042	3.737	- 18.60	4.516	5.408	- 16.49
$E(B_{16})$	1.340	1.366	- 1.90	1.964	1.953	0.56	2.522	2.366	6.59

Table 13
Results for the five-machine network of Fig. 5 and Data set 1 in Table 4

	<i>N</i> = 6			<i>N</i> = 9			<i>N</i> = 12		
	Algo	Sim	% Error	Algo	Sim	% Error	Algo	Sim	% Error
θ	0.733	0.701	4.56	0.816	0.781	4.48	0.867	0.826	4.96
$E(B_{21})$	2.189	2.670	− 18.01	3.153	3.943	− 20.04	4.060	5.230	− 22.37
$E(B_{31})$	2.189	2.534	− 13.61	3.153	3.754	− 16.01	4.060	5.017	− 19.08
$E(B_{42})$	1.968	1.679	17.21	2.897	2.569	12.77	3.753	3.439	9.13
$E(B_{53})$	1.968	1.737	13.30	2.897	2.620	10.57	3.753	3.461	8.44
$E(B_{14})$	1.829	1.651	10.78	2.938	2.488	18.09	4.179	3.331	25.46
$E(B_{15})$	1.829	1.730	5.72	2.938	2.627	11.84	4.179	3.523	18.62

Table 14
Results for the five-machine network of Fig. 5 and Data set 2 in Table 4

	<i>N</i> = 6			<i>N</i> = 9			<i>N</i> = 12		
	Algo	Sim	% Error	Algo	Sim	% Error	Algo	Sim	% Error
θ	0.650	0.632	2.85	0.715	0.697	2.58	0.754	0.731	3.15
$E(B_{21})$	2.125	2.448	− 13.19	3.052	3.531	− 13.57	3.923	4.542	− 13.63
$E(B_{31})$	2.125	2.309	− 7.97	3.052	3.325	− 8.21	3.923	4.284	− 8.43
$E(B_{42})$	2.424	2.215	9.44	3.866	3.653	5.83	5.407	5.234	3.31
$E(B_{53})$	2.424	2.297	5.53	3.866	3.736	3.48	5.407	5.348	1.10
$E(B_{14})$	1.437	1.337	7.48	2.071	1.816	14.04	2.662	2.225	19.64
$E(B_{15})$	1.437	1.394	3.08	2.071	1.939	6.81	2.662	2.368	12.42

Table 15
Results for the five-machine network of Fig. 5 and Data set 3 in Table 4

	<i>N</i> = 6			<i>N</i> = 9			<i>N</i> = 12		
	Algo	Sim	% Error	Algo	Sim	% Error	Algo	Sim	% Error
θ	0.650	0.631	3.01	0.710	0.698	1.72	0.744	0.732	1.64
$E(B_{21})$	2.004	2.453	− 18.30	2.828	3.572	− 20.83	3.577	4.652	− 23.11
$E(B_{31})$	2.004	2.292	− 12.57	2.828	3.224	− 12.28	3.577	4.117	− 13.12
$E(B_{42})$	1.646	1.358	21.21	2.254	1.896	18.88	2.701	2.329	15.97
$E(B_{53})$	1.646	1.410	16.74	2.254	1.978	13.95	2.701	2.393	12.87
$E(B_{14})$	2.337	2.189	6.76	3.906	3.533	10.56	5.708	5.019	13.73
$E(B_{15})$	2.337	2.298	1.70	3.906	3.798	2.84	5.708	5.491	3.95

Table 16
Results for the five-machine network of Fig. 6 and Data set 1 in Table 5

	N = 6			N = 9			N = 12		
	Algo	Sim	% Error	Algo	Sim	% Error	Algo	Sim	% Error
θ	0.609	0.624	- 2.40	0.672	0.688	- 2.33	0.706	0.722	- 2.22
$E(B_{21})$	1.574	1.645	- 4.32	2.327	2.332	- 0.21	3.184	3.012	5.71
$E(B_{31})$	3.613	3.652	- 1.07	5.047	5.176	- 2.49	6.091	6.482	- 6.03
$E(B_{42})$	2.719	2.244	21.17	3.935	3.301	19.21	4.945	4.270	15.81
$E(B_{52})$	2.719	2.118	28.38	3.935	3.011	30.69	4.945	3.793	30.37
$E(B_{13})$	2.387	2.348	1.66	3.952	3.824	3.35	5.913	5.518	7.16
$E(B_{14})$	1.709	2.112	- 19.08	2.740	3.367	- 18.62	3.870	4.718	- 17.97
$E(B_{15})$	1.709	2.238	- 23.64	2.740	3.657	- 25.08	3.870	5.196	- 25.52

Table 17
Results for the five-machine network of Fig. 6 and Data set 2 in Table 5

	N = 6			N = 9			N = 12		
	Algo	Sim	% Error	Algo	Sim	% Error	Algo	Sim	% Error
θ	0.635	0.645	- 1.55	0.700	0.707	- 0.99	0.733	0.736	- 0.41
$E(B_{21})$	1.756	1.841	- 4.62	2.682	2.739	- 2.08	3.738	3.656	2.24
$E(B_{31})$	3.377	3.447	- 2.03	4.556	4.743	- 3.94	5.412	5.799	- 6.67
$E(B_{42})$	3.146	2.777	13.29	4.748	4.363	8.82	6.112	6.061	0.84
$E(B_{52})$	2.361	1.750	34.91	3.296	2.301	43.24	4.322	2.666	62.12
$E(B_{13})$	2.623	2.553	2.74	4.444	4.257	4.39	6.591	6.201	6.29
$E(B_{14})$	1.103	1.383	- 20.25	1.575	1.898	- 17.02	2.142	2.283	- 6.18
$E(B_{15})$	1.881	2.410	- 21.95	3.020	3.961	- 23.76	3.936	5.678	- 30.68

Table 18
Results for the five-machine network of Fig. 6 and Data set 3 in Table 5

	N = 6			N = 9			N = 12		
	Algo	Sim	% Error	Algo	Sim	% Error	Algo	Sim	% Error
θ	0.673	0.679	- 0.88	0.742	0.744	- 0.27	0.782	0.773	1.16
$E(B_{21})$	2.090	2.224	- 6.03	3.438	3.659	- 6.04	4.810	5.470	- 12.07
$E(B_{31})$	3.001	3.107	- 3.41	3.697	3.817	- 3.14	4.320	4.107	5.19
$E(B_{42})$	2.680	2.238	19.75	3.810	3.168	20.27	4.962	3.893	27.46
$E(B_{52})$	2.680	2.150	24.65	3.810	3.015	26.37	4.962	3.677	34.95
$E(B_{13})$	2.999	2.893	3.66	5.303	5.183	2.32	7.690	7.893	- 2.57
$E(B_{14})$	1.232	1.538	- 19.90	1.746	2.173	- 19.65	2.225	2.637	- 15.62
$E(B_{15})$	1.232	1.626	- 24.23	1.746	2.326	- 24.94	2.225	2.853	- 22.01

Table 19
Results for the five-machine network of Fig. 6 and Data Set 4 in Table 5

	<i>N</i> = 6			<i>N</i> = 9			<i>N</i> = 12		
	Algo	Sim	% Error	Algo	Sim	% Error	Algo	Sim	% Error
θ	0.698	0.710	– 1.69	0.778	0.789	– 1.39	0.820	0.834	– 1.68
$E(B_{2,1})$	1.744	1.888	– 7.63	2.625	2.825	– 7.08	3.566	3.788	– 5.86
$E(B_{3,1})$	4.094	4.121	– 0.66	6.044	6.170	– 2.04	7.898	8.178	– 3.42
$E(B_{1,3})$	1.906	1.879	1.44	2.955	2.830	4.42	4.101	3.822	7.30
$E(B_{4,2})$	2.881	2.420	19.05	4.289	3.642	17.76	5.681	4.810	18.11
$E(B_{5,2})$	2.881	2.336	23.33	4.289	3.461	23.92	5.681	4.623	22.89
$E(B_{1,4})$	1.379	1.691	– 18.45	2.091	2.533	– 17.45	2.750	3.402	– 19.17
$E(B_{1,5})$	1.379	1.778	– 22.44	2.091	2.714	– 22.96	2.750	3.589	– 23.38

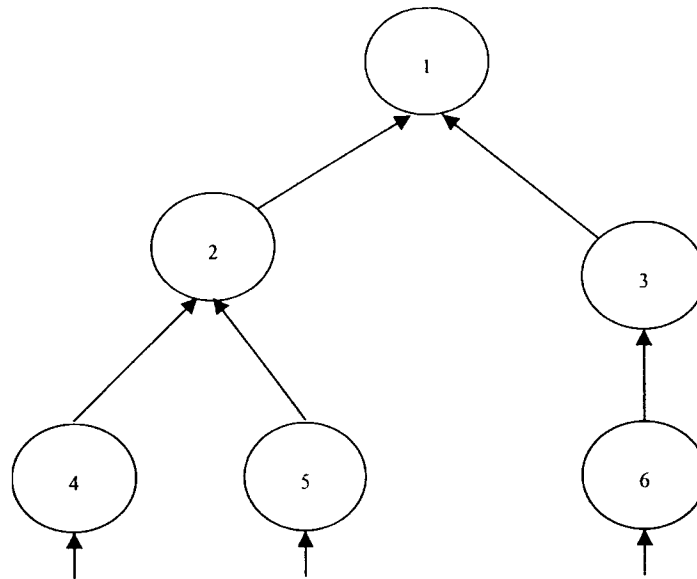


Fig. 4. A six-machine network model. (Machines 4–6 are leaf nodes).

additional state variables to capture the additional information. As the partition gets finer, the model becomes more accurate.

6.2. Adding tunable parameters

Estimates (4.1)–(4.3) of the aggregate transition rates can be augmented with additional parameters in order to meet additional conditions on the system. This is illustrated by one approach to ensure that Eqs. (2.5) and (2.7) hold:

Expression (4.2) for the estimated Kanban arrival rate $\bar{\gamma}^{(f)*}$ as seen by machine f , for $f \in \text{INT}$, can be modified by incorporating an additional parameter a_f . This is justified by the recognition that

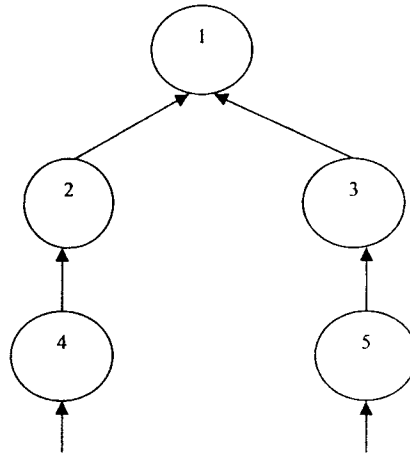


Fig. 5. A five-machine network model. (Machines 4 and 5 are leaf nodes).

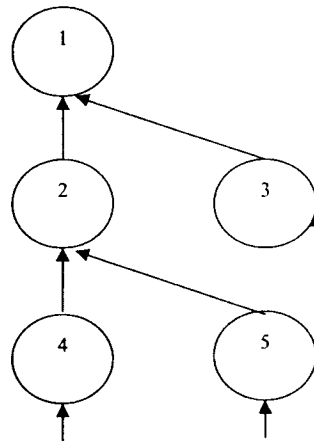


Fig. 6. A six-machine network model. (Machines 3–5 are leaf nodes).

the approximation Eq. (3.5) is not exact. The new parameter helps determine $\bar{\gamma}^{(f)*}$ and hence θ_f , and it should be chosen so that $\theta_f = \theta_r$. Satisfaction of this condition for all $f \neq r$ is Eq. (2.5).

Similarly, expression (4.1) for the effective service rate $\bar{\mu}^{(f)*}$ at machine $f \in \text{INT}$ can be modified by employing an additional parameter b_f . This is justified by the recognition that approximation (3.1) is not exact. This new parameter helps determine $\bar{\mu}^{(f)*}$ and hence $\text{En}_{fS(f)}$, and it should be chosen so that

$$\text{En}_{fS(f)} = \text{Em}_{S(f)} - \text{Em}_f.$$

Satisfaction of this condition for all $f \in \text{INT}$ will ensure that the path-independence condition (2.7) holds.

6.3. Partitions with multiple machines

Simulation shows the existence of a “floating bottleneck”, where completion of an assembly at a slow machine triggers a quick succession of other assembly operations downstream, with this cascade stopping when the platoon of parts reaches a new downstream bottleneck. The aggregate model of f fails to capture this dynamic correlated behavior of several machines, which may be the source of the large errors in computing mean queue lengths, a common failing of “single-node” decompositions. The appropriate remedy is a partition that simultaneously encompasses several assembly machines. This, unfortunately, leads to an explosion of the number of the states. A coarser partition of the Cartesian-product state space seems appropriate. Computational results for blocking networks [13] suggest that there are diminishing marginal benefits from incorporating more than a few machines in the aggregate model.

6.4. Hybrid schemes

Hazra [1.7] describes a different aggregation approach for evaluating assembly networks, and surveys other available techniques. Each technique has its own strengths and weaknesses. An open issue is how to construct *hybrid* approaches that combine the benefits of several pre-existing schemes. For example, Hazra’s [1] scheme has only 3% error in computing mean queue lengths, while the present A/D scheme has more accurate throughputs. Could the aggregate service rates from one scheme be used as inputs to the other? The goal would be to achieve the advantages of *both*: only 1% error in throughput and only 3% error in mean queue length.

6.5. Triangular decomposition of the fixed-point problem

There is structure in the fixed-point equations that has not yet been exploited. The influence diagram for the network in Fig. 1 is given by Fig. 7.

Fig. 7 shows the interactions in the fixed-point system, and mimics the physical topology. At machine f , the service rate information $\bar{\mu}^{(f)}$ is passed downstream to the successor while the Kanban arrival rate information $\bar{\gamma}^{(f)}$ is passed upstream to all the predecessors. For example, the model at machine 2 receives $\bar{\mu}^{(4)}$ information from the machine-4 model, $\bar{\mu}^{(5)}$ information from the machine-5

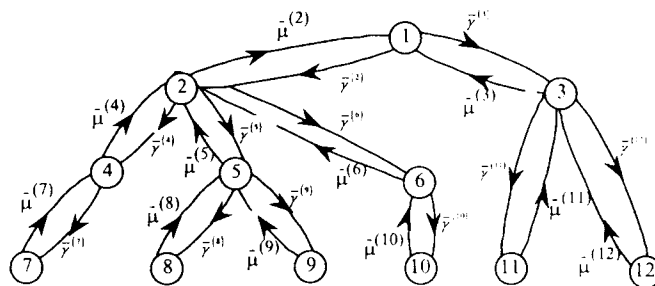


Fig. 7. Influence diagram for the assembly network in Fig. 1. Note: for $f \in L = \{7, 8, 9, 10, 11, 12\}$, $\bar{\mu}^{(f)}$ are already known exactly while $\bar{\gamma}^{(f)}$ are not needed.

model, and $\bar{\gamma}_i^{(2)}$ information from the machine-1 model. The model at machine 2 then generates $\bar{\gamma}_i^{(4)}$ information, which it sends to the machine-4 model, $\bar{\gamma}_i^{(5)}$ information, which it sends to the machine-5 model, and $\bar{\mu}^{(2)}$ information, which it sends to the machine-1 model. Each machine model interacts with all its immediate neighbors, and with no one else, and these information flows are always bidirectional.

The resulting equations are *block triangular*. For example, the Kanban arrival information $\bar{\gamma}_i^{(2)}$ passed to machine 2 contains all the information needed to compute the behavior at the subtree consisting of machines 2 and 4–10. In particular $\bar{\mu}^{(4)}$ and $\bar{\mu}^{(5)}$ will be implicit functions of $\bar{\gamma}_i^{(2)}$, so that the entire subtree can be collapsed into a *single fictitious machine 2[#]*. This provides the justification for an aggregation approach whereby the subtrees are merged sequentially, similar to the merging in [7]. The critical feature in any such merging scheme, of course, is making accurate estimates of the queue-dependent service rate $\bar{\mu}^{(2\#)}$ of the fictitious machine 2[#]. Hazra's results show that this approach is highly promising.

7. Summary

The performance evaluation of a closed Kanban-controlled assembly system is accomplished by a new heuristic version of the exact aggregation–disaggregation theory for finite continuous-time Markov chains. It provides accurate estimates of the throughput. Because the approximation is theory based, it provides a framework for further model development, with some possible extensions described in the paper. The approximation has the novel feature of doing *simultaneous multiple partitions of the state space*, in such a way that the associated aggregate transition rates are *mutually consistent*. The methodology is a novel approach towards extending aggregation ideas to fork-and-join queueing networks. It provides fast and accurate throughput estimates for complex production systems, and insights into the behavior of these systems. We use the results of this methodology to show how the throughput of an assembly system varies with changes in the parts flows, in the total number of Kanbans, or in the processing rate at the individual stations. The speed at which we obtain these results make it a practical tool for the analysis of key design trade-offs for complex manufacturing systems. Anticipated use of the model includes balancing of routes, speeds and topologies.

Acknowledgements

Parts of this paper were completed while P. Schweitzer was a Lady Davis Fellow at Hebrew University in Jerusalem, Israel. He thanks the University and Lady Davis Trust for many courtesies extended.

References

- [1] Hazra J, Seidmann A. Performance evaluation of closed tree-structured assembly systems. IIE Transactions 1996; 28:591–9.

- [2] Takahashi Y. A lumping method for numerical calculations of stationary distributions of Markov chains. Research Report No. B-18, Dept. of Information Sciences, Tokyo Institute of Technology, Tokyo, Japan, 1975.
- [3] Schweitzer PJ. Aggregation methods for large Markov chains. In: Iazeolla G, Courtois PJ, Hordijk A, editors. Mathematical computer performance and reliability. Amsterdam: North-Holland, 1984:275–86.
- [4] Schweitzer PJ. A Survey of aggregation–disaggregation in large Markov chains. In: Stewart WJ, editor. Numerical solution of Markov chains, New York: Marcel Dekker, 1991:63–88.
- [5] Dallery Y, Liu Z, Towsley D. Equivalence, reversibility, Symmetry and concavity properties in fork-join queueing networks with blocking. *Journal of Association for Computing Machinery* 1994;41:903–42.
- [6] Viswanadham N, Narahari Y. Performance modeling of automated manufacturing systems. Englewood Cliffs, NJ: Prentice-Hall, 1992. Hoop MJ, Spearman ML. Factory physics. foundation of manufacturing management. Homewood, IL: Irwin, 1996.
- [7] Hazra J. Performance evaluation of Kanban-controlled assembly networks. PhD Thesis, William E. Simon Graduate School of Business Administration, Rochester, NY: University of Rochester, 1995.
- [8] Gershwin SB. Manufacturing systems engineering. Englewood Cliffs, NJ: Prentice-Hall, 1994. Buzacott JA, Shanthikumar JG. Stochastic models of manufacturing systems. S. Englewood Cliffs, NJ: Nahmias, Prentice-Hall, 1993. Production and operations analysis. Homewood, IL: Irwin, 1997.
- [9] Chandrasekhar Rao P, Suri R. Performance analysis of an assembly station with input from multiple fabrication lines. Working paper Dec. 1996, Department of Industrial Engineering, University of Wisconsin-Madison, Madison, WI.
- [10] Chandrasekhar Rao P, Suri R. Approximate queueing network models for closed fabrication assembly systems – Part I: Single level systems. *Production and Operations Management* 1998;3:244–75.
- [11] Baynat B, Bruno, Dallery Y. Approximate techniques for general closed queueing networks with subnetworks. *EJOR* 1992; 69(2): 250–64.
- [12] Schweitzer PJ. Aggregative modelling of queueing networks. In: Bonatti M, editor. Teletraffic science for new cost-effective systems, networks and services. Amsterdam: Elsevier, North-Holland, 1989:1522–8.
- [13] Takahashi Y. Aggregate approximation for acyclic queueing networks with communication blocking. In: Perros HG, Altiock T, editors. Queueing networks with blocking. Amsterdam: Elsevier, 1989:33–46.
- [14] Dimascolo M, Frein Y, Dallery Y. An analytical method for performance evaluation of Kanban controlled production systems. *Operations Research* 1996;44(1):50–65.

Jishnu Hazra received his B.Tech (Mech. Engg.) from IIT, Delhi and M.S. and Ph.D. from the University of Rochester in Operations Management. His research interests are in Supply Chain Management and JIT Manufacturing. He has conducted Executive Development Programs on Supply Chain Management for TATA-IBM executives and other companies. He is also involved in consulting and research projects in the areas of R&D Project Planning and Logistics.

Paul J. Schweitzer is Professor of Business Administration at the William E. Simon Graduate School of Business Administration of the University of Rochester. His interests include military operations research, performance models of manufacturing and computer systems, Markovian decision processes (MDPs), management science and information technology. He has done seminal work on value-iteration for MDPs, perturbation of Markov chains, mean value analysis of queueing networks including flexible manufacturing systems, and aggregation techniques. He is the author of over 100 research papers and reports.

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. Professor Seidmann is the author of over 75 research articles which appear in many of the leading scientific journals and is a Department Editor on Interdisciplinary Management Research and Applications in Management Science. His current research and consulting activities include Information Systems, Business Process Reengineering, Project Management and Optimal Resource Allocation, Strategic Manufacturing Systems, Information Economics, Healthcare Management, Stochastic Processes and Performance Modeling for Capacity Planning of Assembly Systems.