PERGAMON

Computers & Operations Research 29 (2002) 103–121

computers & operations research

www.elsevier.com/locate/dsw

Buffer allocation in flow-shop-type production systems with general arrival and service patterns

Ming-Guang Huang, Pao-Long Chang*, Ying-Chyi Chou

Institute of Business and Management, National Chiao Tung University, 4F 114 Section 1 Chung Hsiao West Road, Taipei, Taiwan

Received 1 October 1999; received in revised form 1 May 2000

Abstract

This study investigates the buffer allocation strategy of a flow-shop-type production system that possesses a given total amount of buffers and finite buffer capacity for each workstation as well as general interarrival and service times in order to optimize such system performances as minimizing work-in-process, cycle time and blocking probability, maximizing throughput, or their combinations. In theory, the buffer allocation problem is in itself a difficult NP-hard combinatorial optimization problem, it is made even more difficult by the fact that the objective function is not obtainable in closed form for interrelating the integer decision variables (i.e., buffer sizes) and the performance measures of the system. Therefore, the purpose of this paper is to present an effective design methodology for buffer allocation in the production system. Our design methodology uses a dynamic programming process along with the embedded approximate analytic procedure for computing system performance measures under a certain allocation strategy. Numerical experiments show that our design methodology can quickly and quite precisely seek out the optimal or sub-optimal allocation strategy for most production system patterns.

Scope and purpose

Buffer allocation is an important, yet intriguingly difficult issue in physical layout and location planning for production systems with finite floor space. Adequate allocation and placement of available buffers among workstations could help to reduce work-in-process, alleviate production system's congestion and even blocking, and smooth products manufacturing flow. In view of the problem complexity, we focus on flow-shop-type production systems with general arrival and service patterns as well as finite buffer capacity. The flow-shop-type lines, which usually involve with product-based layout, play an important role in mass production type of manufacturing process organization such as transfer line, batch flow line, etc. The purpose of this paper is to present a design methodology with heuristic search and imbedded analytic algorithm of system performances for obtaining the optimal or sub-optimal buffer allocation strategy. Successful use of

* Corresponding author. Tel.: + 886-2-23146515; fax: + 886-2-23610656.
E-mail address: paolong@cc.nctu.edu.tw (P.-L. Chang).

this design methodology would improve the production efficiency and effectiveness of flow-shop-type production systems. © 2001 Elsevier Science Ltd. All rights reserved.

Keywords: Buffer allocation; Open tandem general queueing networks; Blocking; Dynamic programming

1. Introduction

Buffer allocation is one of the most important, yet intriguingly difficult problems in the design of production systems since it must be coupled with an effective way of computing system performance measures and a discrete optimization procedure for allocating buffers to the queues of each workstation in the system in order to achieve such system performances as minimizing WIP, cycle time and blocking probability, maximizing throughput, or their combinations. In theory, the buffer allocation problem is in itself a difficult NP-hard combinatorial optimization problem, it is made even more difficult by the fact that the objective function is not obtainable in closed form for interrelating the integer decision variables (i.e. buffer sizes) and the performance measures of the system. In view of the vast difficulty involved in solving our described problem, we present an effective design methodology for buffer allocation in a flow-shop-type production system that possesses a given amount of buffers and finite buffer capacity for each workstation as well as general interarrival and service times.

In recent years, there have been substantial efforts devoted to the design of buffer in production system. A summary of these studies is as follows. Reiman [1] analyzed the allocation strategy under light traffic density. Venkat [2] solved the buffer allocation strategy assuming that queuing networks posed exponential distributed service time and Poisson arrival process. Conway et al. [3] demonstrated how to employ the simulation experiments to study the buffer allocation problems. Hillier et al. [4] has performed rather comprehensive studies to characterize the optimal buffer allocation pattern, and numerical results for lines with up to nine workstations were provided. So [5] carried out the allocation strategy with minimal work-in-process through the use of enumeration method.

Some researchers have taken advantage of the heuristic search method. Kubat and Sumita [6], and Jafari and Shantikumar [7] used dynamic programming to determine buffer allocation of tandem production system with $M/M/1/K$ queues. Hillier and So [8], Smith and Daskalaki [9], and Smith and Chikhale [10] applied other heuristic search techniques to find buffer allocation of series/general assembly/transfer lines with $M/M/1/K$ and $M/E_k/1/K$ queues. Hillier and So found that buffers increase progressively in downstream workstations in a balanced facility when both throughput and work-in-process are jointly considered. McClain and Moodie [11] also agreed to their conclusion. Besides, they further found that whether production lines are balanced or unbalanced under parts successively arriving, as a general rule, turned up the incremental attribute. Singh and Smith [12] have exploited Powell's unconstrained optimization procedure (Powell [13]) along with the embedded expansion method (Kerbache and Smith [14,15]) to capture buffer allocation for series, merge and split, and general configurations with $M/M/C/K$ queues.

In the course of searching, it is necessary to have an analytic approach for evaluating system performances under a given buffer allocation strategy. Generally speaking, the approximate procedure can be classified into two categories. One is according to the notion of analyzing successive pairs of adjacent queues as suggested by Brandwajn and Jow [16], and Gershwin [17]. The other is employed in this paper and summarized as follows.

(1) The queueing network with blocking is decomposed into individual queues with modified service and arrival process. The service process is revised to reflect the additional delay that a part might undergo due to blocking.
(2) The queue buffer is also modified for one particular blocking mechanism.
(3) Each queue is then analyzed in isolation.

Most of the approximate algorithms proposed for exponential service times such as those by Shanthikumar [18], Altiok [19], Boxma and Konheim [20], Perrors and Altiok [21], and Perros and Snyder [22]. Altiok [23] considered tandem configurations with general service times, but he assumed that the arrival process to each decomposed queue is a Poisson process. Kerbache and Smith [15] analyzed open finite queueing networks with general service times assuming that the arrival process is renewal. The methods presented by Harrison and Pich [24] and Shi [25] are a QNA version; i.e., two-moment approximation (see Whitt [26,27]), for handling queueing networks with blocking. Shi extended from Bronshtein [28] and assumed that wherever blocking occurs, arriving customers simply get lost.

In this paper, we present an approximate algorithm for analyzing the performance of open tandem queueing networks with blocking and general interarrival and service times. This algorithm is an extension of the algorithms proposed by Altiok [19,29], Perrors and Altiok [21], and Perrors and Snyder [22], where the arrival process to each decomposed queue was assumed to be Poisson. However, because of the blocking effect, this assumption is generally not true. Jun and Perrors [30] reported on an approximation, which is abbreviated as JP approach, that each arrival process is assumed to be renewal and approximated by a Coxian distribution (see Altiok [29], and Yao and Buzacott [31]). Also, the parameters of these distributions are approximated iteratively. Their method had been validated by solving performance measures of open tandem queueing network with finite buffer, general service times and Poisson arrival process. In this paper, we revise the JP approach to deal with exogenous renewal arrival process. To this end, our algorithm primarily differs from the JP approach in that the arrival process at each finite queue is assumed to be a Coxian distribution in the three basic steps of the JP approach.

The rest of the paper is organized as follows. First, in Section 2, we outline the underlying nature of the flow-shop-type production system under study and summarize the approximate algorithm for solving performance measures of the system. Then, in Section 3, we describe the search procedure of forward recursive dynamic programming algorithm for finding out the buffer allocation strategy in detail. Some numerical cases are presented in Section 3.2, where we compare the results of the search procedure with the output obtained from exhaustive enumeration. Brief concluding remarks are presented in Section 4.

2. The approximate algorithm for performance evaluation

2.1. The tandem queueing network model

First of all, we outline our system patterns as shown in Fig. 1. The fundamental properties of the flow-shop-type production system are assumed as follows.

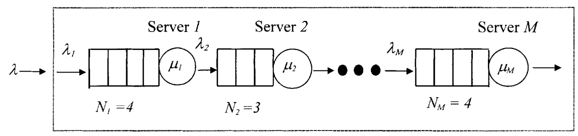

Fig. 1. The model of a flow-shop-type production system.

(1) The system in our model consists of a number of queues arranged in tandem. A part gets into the system via the first queue, flows through and is processed by all queues in turn, and finally departs from the system by way of the last queue.
(2) The total factory area that can be used to set up the buffer capacities is fixed and finite. As a result, buffer capacity (including the one in service) of all queues in the system are restricted in size, and thus the blocking phenomenon exists.
(3) Parts arrive the system from outside according to a renewal process, that is, the interarrival times of parts are independently and identically distributed random variables with an arbitrary probability distribution. In addition, we also assume that the arrival process to each queue is renewal. Parts arriving during the period that the buffer capacity of the first queue is full are lost.
(4) Each queue has a single server with general service time distribution. Moreover, the service times and interarrival times are mutually independent. In addition, service disciplines are on a first-come-first-served basis for all queues.
(5) Once the operation of a part is finished at the ith queue, the part goes directly to the $(i+1)$th queue, where it enters into service immediately if the server is free; otherwise, it joins the queue if there is a place in the buffer capacity. If there is no place in the queue of the $(i+1)$th queue, the part has to wait at the ith queue, which therefore becomes blocked. During this time, the ith queue remains idle and it cannot serve any other parts that might be waiting in its queue. Obviously, blocking of the last queue is not feasible in this model.

2.2. The approximate algorithm

As mentioned, we revise the JP approach to evaluate the performances of the system described in Section 2.1. The primary characteristics of our algorithm are summarized as follows.

(1) The network decomposition is exploited. That is, the system of tandem queues is decomposed into individual queues and then each queue is analyzed in isolation. Finally, one can obtain performance measures of the whole system by an integrated process.
(2) The Coxian distribution with two phases (denoted by Cox-2) is used to approximate general service time and interarrival time distribution whose squared coefficient of variation (c^2) is greater than 0.5. That is, a part in queue i first goes through service (arrival) phase 1 with mean $1/\xi_{i1}$ $(1/\bar{\lambda}_{i1})$; after completing this service (arrival), it leaves (arrives at) the service facility with probability $1-\alpha_i$ $(1-\bar{\gamma}_i)$, while with probability α_i $(\bar{\gamma}_i)$, it receives an additional phase 2 service (arrival) with mean $1/\xi_{i2}$ $(1/\bar{\lambda}_{i2})$.
(3) The parameters of the service and the arrival processes are computed and amended approximately using an iterative scheme.

First, we define the following notations to facilitate description of the analytic procedure.

M	total number of queues
$\bar{\Lambda}(\bar{\lambda}_1, \bar{\lambda}_2, \bar{\gamma})$	parameters of the *Cox*-2 exogenous interarrival time distribution to the system
$\bar{\Lambda}_i(\bar{\lambda}_{i1}, \bar{\lambda}_{i2}, \bar{\gamma}_i)$	parameters of the *Cox*-2 effective interarrival time distribution at queue i
$\Lambda_i(\lambda_{i1}, \lambda_{i2}, \gamma_i)$	parameters of the *Cox*-2 overall interarrival time distribution at queue i
N_i	buffer capacity of queue i (including the one in service)
$\Xi_i(\xi_{i1}, \xi_{i2}, \alpha_i)$	parameters of the *Cox*-2 original service time distribution at queue i
$\Omega_i(\omega_{i1}, \omega_{i2}, \beta_i)$	parameters of the *Cox*-2 effective service time distribution at queue i (including the blocking delay)
$p_i(n)$	probability that there are n parts in queue i (including the one in service) at any time
$p_i(n, j, k)$	probability that there are n parts in queue i (including the one in service), the arrival process is in phase j ($j = 1, 2$), and the service process is in phase k ($k = 1, 2$) at any time
$q_i(n)$	probability that an arriving part finds n parts in queue i (including the one in service)
$q_i(n, j)$	probability that the server is in phase j ($j = 1, 2$) and there are n parts in queue i (including the one in service) immediately before an arrival
$q_i(n, j, k)$	probability that the arrival process is in phase j ($j = 1, 2$), the service process is in phase k ($k = 1, 2$), and there are n parts in queue i (including the one in service) immediately before an arrival
$r_i(n)$	probability that a departing part leaves n parts in queue i
$w_{i,j}$	conditional probability that the ith server is at the jth phase of service, given that queue i has N_i parts, i.e., queue i is full
π_i	conditional probability that upon service completion at the queue i, the queue $i+1$ is full, i.e., it contains N_{i+1} parts including the one in service
$S_i^*(s)$	Laplace transform of the pdf $(s_i(t))$ of the original service time at queue i
$B_i^*(s)$	Laplace transform of the pdf $(b_i(t))$ of the effective service time at queue i (including the blocking delay)
$D_i^*(s)$	Laplace transform of the pdf $(d_i(t))$ of the interdeparture time from queue i
$A_i^*(s)$	Laplace transform of the pdf $(a_i(t))$ of the effective interarrival time at queue i in starting step
$E_i^*(s)$	Laplace transform of the pdf $(e_i(t))$ of the effective interarrival time at queue i in backward adjustment step and forward adjustment step
$O_i^*(s)$	Laplace transform of the pdf $(o_i(t))$ of the overall interarrival time at queue i

Next, we will describe the approximate algorithm in detail. The analytic procedure of algorithm consists of four basic steps, i.e., preparation step, starting step, backward adjustment step, and forward adjustment step.

Preparation step

(1) Get the first three moments $m_{i1}^s, m_{i2}^s, m_{i3}^s; i = 1, 2, \dots, M$ of the original service time for each queue, and then convert the computed results into the parameters $\Xi_i(\xi_{i1}, \xi_{i2}, \alpha_i)$; $i = 1, 2, \dots, M$ of *Cox*-2 distribution (refer to Appendix A).

(2) Get the first three moments m_1^a, m_2^a, m_3^a of the interarrival time to the first queue.
(3) Except the first queue, add one fictitious buffer to every queue so as to accommodate the blocked part, i.e., $N_i + 1$; $i = 2, 3, \ldots, M$.

Starting step
(1) The first three moments m_{i1}^a, m_{i2}^a, m_{i3}^a of the effective interarrival time for all queues are m_1^a, m_2^a, m_3^a.
(2) Because of the network structure, the last queue could not get blocked. Thus, we start with the Mth queue and proceed backward to queue 1. $B_M^*(s) = S_M^*(s)$ and $\Omega_M(\omega_{M1}, \omega_{M2}, \beta_M) = \Xi_M(\xi_{M1}, \xi_{M2}, \alpha_M)$ evidently make sense for queue M. So, we can analyze Mth queue as a Cox-2/Cox-2/1/N_M + 1 queue and compute the queue-length distribution $p_M(n), p_M(n, j, k)$ and $q_M(n), q_M(n, j)$ and $q_M(n, j, k)$ (refer to Appendix A). As for the other queues, because they may get blocked, it is necessary to add probable blocking delay time to the original service time. Laplace transform of the effective service time is then obtained as follows (see Jun and Perros [30])

$$B_i^*(s) = (1 - \pi_{i+1})S_i^*(s) + \pi_{i+1}\left[S_i^*(s)*\left(w_{i+1,1}B_{i+1}^*(s) + w_{i+1,\,2}\frac{\omega_{i+1,2}}{s + \omega_{i+1,2}}\right)\right],$$
$$i = M-1, M-2, \ldots, 1, \tag{1}$$

where

$$S_i^*(s) = (1 - \alpha_i)\frac{\xi_{i1}}{s + \xi_{i1}} + \alpha_i\frac{\xi_{i1}}{s + \xi_{i1}}\frac{\xi_{i2}}{s + \xi_{i2}}, \tag{2}$$

$$B_{i+1}^*(s) = (1 - \beta_{i+1})\frac{\omega_{i+1,1}}{s + \omega_{i+1,1}} + \beta_{i+1}\frac{\omega_{i+1,1}}{s + \omega_{i+1,1}}\frac{\omega_{i+1,2}}{s + \omega_{i+1,2}}. \tag{3}$$

The quantities π_i and $w_{i+1,j}$ are calculated approximately from queue-length distribution of the ith queue. In particular, the quantity $w_{i+1,j}$ can be obtained as follows:

$$w_{i+1,j} = q_{i+1}(N_{i+1}, j)/q_{i+1}(N_{i+1}); \quad j = 1, 2. \tag{4}$$

The probability π_i can be solved using Little's formula on the fictitious position, i.e., $N_{i+1} + 1$ in $(i+1)$th queue as follows:

$$(1/m_1^a)\pi_i\left[w_{i+1,1}\left(\frac{1}{\omega_{i+1,1}} + \frac{\beta_{i+1}}{\omega_{i+1,1}}\right) + w_{i+1,2}\frac{1}{\omega_{i+1,2}}\right] = p_{i+1}(N_{i+1} + 1). \tag{5}$$

The quantity within bracket in the above expression is the remaining service time of the $(i+1)$th server at the instance when the ith server gets blocked. Therefore, the first three moments of the effective service time are computed as follows:

$$m_{i1}^b = -\lim_{s \to 0}\frac{\mathrm{d}B_i^*(s)}{\mathrm{d}s}, \quad m_{i2}^b = \lim_{s \to 0}\frac{\mathrm{d}^2B_i^*(s)}{\mathrm{d}s^2}, \quad m_{i3}^b = -\lim_{s \to 0}\frac{\mathrm{d}^3B_i^*(s)}{\mathrm{d}s^3}. \tag{6}$$

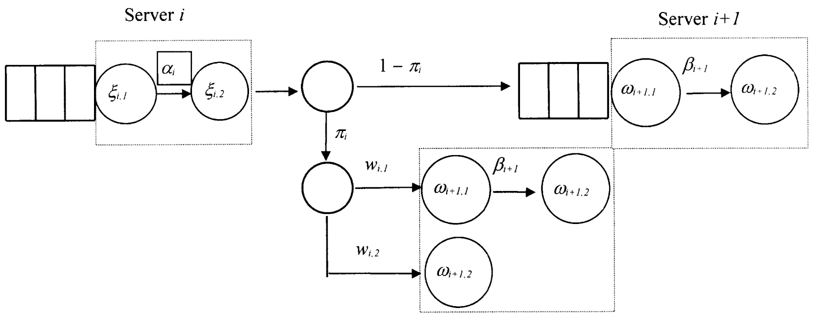

Fig. 2. The blocking and effective service time.

The computed results are then converted into the parameters $\Omega_i(\omega_{i1}, \omega_{i2}, \beta_i)$; $i = M-1, M-2, \dots, 1$ of *Cox*-2 distribution (refer to Appendix A). Fig. 2 shows the concept of blocking and effective service time.

(3) To guarantee a throughput $1/m_1^a$, we need to know in advance the first three moments $m_{i1}^o, m_{i2}^o, m_{i3}^o$ of $o_i(t)$ rather than the first three moments $m_{i1}^a, m_{i2}^a, m_{i3}^a$ of $a_i(t)$.
Through the relationship between $A_i^*(s)$ and $O_i^*(s)$ (see Kuehn [31]):

$$A_i^*(s) = \frac{(1-q_i(N_i+1))O_i^*(s)}{1-q_i(N_i+1)O_i^*(s)}, \tag{7}$$

$m_{i1}^o, m_{i2}^o, m_{i3}^o$ can be expressed as follows

$$\begin{aligned} m_{i1}^o &= (1-q_i(N_i+1))m_{i1}^a, \\ m_{i2}^o &= (1-q_i(N_i+1))(m_{i2}^a - 2{*}q_i(N_i+1)(m_{i1}^a)^2), \\ m_{i3}^o &= (1-q_i(N_i+1))\times(m_{i3}^a - 6{*}q_i(N_i+1)m_{i2}^a m_{i1}^a + 6q_i(N_i+1)^2(m_{i1}^a)^3), \\ & i = M, M-1, \dots, 2. \end{aligned} \tag{8}$$

Note that the three moments $m_{i1}^o, m_{i2}^o, m_{i3}^o$ are a function of $q_i(N_i+1)$, which is unknown. This can be obtained by solving the fixed-point problem, $1/m_{i1}^o = (1/m_{i1}^a)/(1-q_i(N_i+1))$. As for the first queue, the $m_{11}^o, m_{12}^o, m_{13}^o$ are set equal to $m_{11}^a, m_{12}^a, m_{13}^a$. Then the computed results are converted into the parameters $\Lambda_i(\lambda_{i1}, \lambda_{i2}, \gamma_i)$ of *Cox*-2 distribution (refer to Appendix A).

(4) Working backwards and following the same procedure as in Mth queue, we can analyze queue i; $i = M-1, \dots, 2$, in isolation as a *Cox*-2/*Cox*-2/1/N_i+1 queue, and the first queue is analyzed as a *Cox*-2/*Cox*-2/1/N_1 with $\Lambda_i(\lambda_{i1}, \lambda_{i2}, \gamma_i)$ and $\Omega_i(\omega_{i1}, \omega_{i2}, \beta_i)$ in order to obtain the queue-length distribution $p_i(n)$, $p_i(n, j, k)$ and $q_i(n)$, $q_i(n, j)$ and $q_i(n, j, k)$.

Backward adjustment step

(1) In this step, the analysis of the network proceeds from queue 1 to queue M. The purpose of this step is to revise the parameters of the arrival process, so that the effective service time is kept constant.
(2) Because the effective and overall interarrival time does not change at the first queue, we do not analyze queue 1 in backward adjustment step.

(3) Let us consider queue i; $i = 2, 3, \dots, M$. In order to characterize its arrival process, we must know $d_{i-1}(t)$, the pdf of the interdeparture time from the upstream queue. Assuming that successive effective services are independent of each other, we can obtain its Laplace transform as follows (see Jun and Perros [30]):

$$D_i^*(s) = \delta_{i1} O_i^*(s) B_i^*(s) + \delta_{i2} \frac{\lambda_{i2}}{s + \lambda_{i2}} B_i^*(s) + (1 - \delta_{i1} - \delta_{i2}) B_i^*(s), \quad i = 1, \dots, M - 1, \tag{9}$$

where

$$O_i^*(s) = (1 - \gamma_i) \frac{\lambda_{i1}}{s + \lambda_{i1}} + \gamma_i \frac{\lambda_{i1}}{s + \lambda_{i1}} \frac{\lambda_{i2}}{s + \lambda_{i2}}, \tag{10}$$

where δ_{ij} is the probability that an arrival is in phase j of the arrival process and a departing part leaves queue i empty, $j = 1, 2$.

We can use following equations to calculate δ_{ij}:

$$\delta_{i1} = \frac{1}{G_i} [(1 - \beta_i) \omega_{i1} p_i(1,\, 1,\, 1) + \omega_{i2} p_i(1,\, 1,\, 2)],$$

$$\delta_{i2} = \frac{1}{G_i} [(1 - \beta_i) \omega_{i1} p_i(1,\, 2,\, 1) + \omega_{i2} p_i(1,\, 2,\, 2)],$$

$$G_i = (1 - \beta_i) \omega_{i1} \sum_{n=1}^{N_i+1} \sum_{k=1}^{2} p_i(n,\, k,\, 1) + \omega_{i2} \sum_{n=1}^{N_i+1} \sum_{k=1}^{2} p_i(n,\, k,\, 2). \tag{11}$$

$D_i^*(s)$ is in fact equal to $E_{i+1}^*(s)$, the Laplace transform of the effective interarrival time at queue $i + 1$. From $E_{i+1}^*(s)$, we can obtain the first three moments of $e_{i+1}(t)$: $m_{i+1,1}^e$, $m_{i+1,2}^e$, $m_{i+1,3}^e$ via the transformation similar to (6). However, we need to know the overall arrival process $o_{i+1}(t)$. Taking the same procedure as (8), we can obtain the first three moments of $o_{i+1}(t)$: $m_{i+1,1}^o$, $m_{i+1,2}^o$, $m_{i+1,3}^o$. Once again, the computed results are converted into the parameters $\Lambda_i(\lambda_{i1},\, \lambda_{i2},\, \gamma_i)$ of Cox-2 distribution (refer to Appendix A).

(4) Working forward, we can analyze each queue i; $i = 2, 3, \dots, M$, in isolation as a Cox-2/Cox-2/1/$N_i + 1$ queue with $\Lambda_i(\lambda_{i1},\, \lambda_{i2},\, \gamma_i)$ and $\Omega_i(\omega_{i1},\, \omega_{i2},\, \beta_i)$ in order to obtain the queue-length distribution $p_i(n)$.

Forward adjustment step

(1) The step also proceeds forward from queue M to queue 1 as the starting step. The objective in this step is to re-calculate the effective service time at each queue.
(2) Queue M is not analyzed in this step since it cannot be blocked. The analysis, therefore, proceeds from queue $M - 1$ to queue 1.
(3) Except for replacing $a_i(t)$ obtained in starting step with $e_i(t)$ obtained in backward adjustment step, other analytic procedures are identical to the starting step.
(4) Working backwards, we can analyze queue i; $i = M - 1, M - 2, \dots, 2$, in isolation as a Cox-2/Cox-2/1/$N_i + 1$ queue with $\Lambda_i(\lambda_{i1},\, \lambda_{i2},\, \gamma_i)$ and $\Omega_i(\omega_{i1},\, \omega_{i2},\, \beta_i)$ in order to obtain the queue-length distribution $p_i(n)$. The first queue is analyzed as a Cox-2/Cox-2/1/N_1.

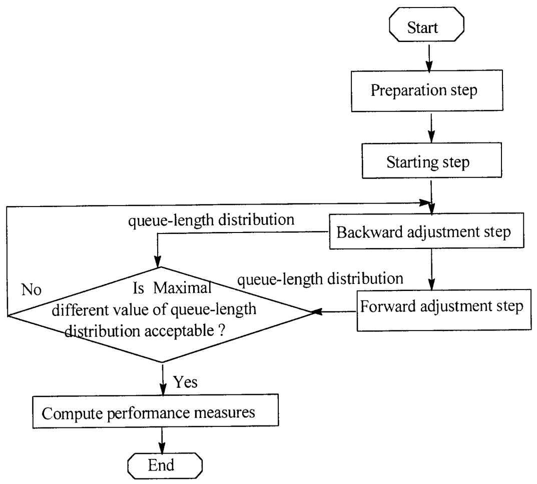
Start
Preparation step
Starting step
Backward adjustment step
queue-length distribution
Forward adjustment step
queue-length distribution
Is Maximal different value of queue-length distribution acceptable ?
No
Yes
Compute performance measures
End

Fig. 3. The analytic procedure of the approximate algorithm.

In addition, note that each queue in all steps is analyzed numerically using an iterative scheme, which is developed by Yao and Buzacott [32] to solve the queue-length distribution $p_i(n)$ and $p_i(n, i, j)$ of $Cox\text{-}2/Cox\text{-}2/c/N$ queue.

Then, as shown in Fig. 3, the analytical process progresses repeatedly between forward adjustment step and backward adjustment step until the maximal difference of queue-length distribution obtained from both steps is less than a default value. It should be noted that after the procedure stops, the actual probability $p_i(N_i)$, $i = 2, 3, \ldots, M$, is obtained as the sum of $p_i(N_i)$ and $p_i(N_i + 1)$. Then, we can resolve some of the performance measures through the steady-state queue-length distribution and Little's formula.

Finally, by integrating the measures of individual queues, we can calculate the performance measures of the whole production system such as WIP, cycle time, throughput, and so on as follows:

$$\text{WIP} = \sum_{i=1}^{M} L_i = \sum_{i=1}^{M} \sum_{k=0}^{N_i} k p_i(k), \tag{12}$$

$$\text{Cycle time} = \sum_{i=1}^{M} W_i = \sum_{i=1}^{M} \left(\frac{L_i}{(1/m_{i1}^e)} \right), \tag{13}$$

$$\text{Throughput} = (1/m_1^a)(1 - p_1(N_1)). \tag{14}$$

The complete approximate algorithm is summarized in Appendix A.

2.3. Numerical examples

In this section, the approximate procedure discussed in Section 2.2 was employed to analyze various tandem configurations. To examine the level of precision, the approximate results were

Table 1
$(1/m_1^q,\ c_a^2) = (2.5,\ 2)$, $1/m_{i1}^s = (4,\ 3,\ 4)$, $c_{s,i}^2 = (2,\ 2,\ 2)$, $N_i = (4,\ 5,\ 3)$

	Approximate	Simulation	Relative err.
L_1	1.570	1.620	0.0309
L_2	2.409	2.482	0.0294
L_3	0.962	0.977	0.0154
Throughput	2.062	2.026	– 0.0178

Table 2
$(1/m_1^q, c_a^2) = (3, 2)$, $1/m_{i1}^s = (4, 3, 4)$, $c_{s,i}^2 = (2, 2, 2)$, $N_i = (4, 5, 3)$

	Approximate	Simulation	Relative err.
L_1	1.941	2.043	0.0499
L_2	2.830	2.891	0.0211
	1.062	1.043	– 0.0182
Throughput	2.254	2.145	– 0.0508

Table 3
$(1/m_1^q,\ c_a^2) = (3.5,\ 2)$, $1/m_{i1}^s = (4,\ 3,\ 4)$, $c_{s,i}^2 = (2,\ 2,\ 2)$, $N_i = (4,\ 5,\ 3)$

	Approximate	Simulation	Relative err.
	2.250	2.353	0.0438
	3.129	3.243	0.0352
	1.124	1.125	0.0009
Throughput	2.391	2.263	– 0.0566

Table 4
$(1/m_1^q, c_a^2) = (2, 2)$, $1/m_{i1}^s = (4, 3, 4, 3, 4)$, $c_{s,i}^2 = (2, 2, 2, 2, 2)$, $N_i = (4,\ 5,\ 3,\ 5,\ 4)$

	Approximate	Simulation	Relative err.
	1.190	1.250	0.0480
	1.991	2.129	0.0648
	0.984	1.037	0.0511
	1.661	1.730	0.0399
	0.883	0.868	– 0.0184
Throughput	1.779	1.747	– 0.0183

compared with the simulation results obtained from six numerical experiments. Simulation models were created and simulation outputs of system-length and throughput were obtained by using the PROMODEL package under the same system conditions and inputs.

Tables 1–3, Tables 4–5, and Table 6 give results for three-queue, five-queue, and seven-queue networks, respectively. Each table gives the approximate results, the simulation results, and the observed relative error. Due to space considerations, we only give the mean system-length L_i for each queue i, and throughput. From these comparisons, the average relative errors of the proposed algorithm were found to be about 4% for throughput, and not exceeding 8% for mean system-length. Moreover, the approximations still give an acceptable result as the number of queues or arrival rate increases. As for cycle time, since system-time is highly correlated with system-length according to Little's formula, it implies that their scale of errors would be pretty close. From this, we are confident that the algorithm presented in this paper should give good approximate when it is used to solve the $\sum GI/G/1/N$ queueing network in tandem.

3. Herhuric search process for buffer allocation

3.1. Heuristic search process

We primarily employed the forward recursive dynamic programming algorithm to obtain the buffer allocation strategy. Dynamic programming typically solves the problem in stages, with each

Table 5
$(1/m_1^q, c_a^2) = (2.5, 2)$, $1/m_{i1}^s = (4, 3, 4, 3, 4)$, $c_{s,i}^2 = (2, 2, 2, 2, 2)$, $N_i = (4, 5, 3, 5, 4)$

	Approximate	Simulation	Relative err.
	1.649	1.750	0.0577
	2.655	2.787	0.0474
	1.238	1.283	0.0351
	2.087	2.116	0.0137
	1.051	1.036	− 0.0145
Throughput	2.024	1.956	− 0.0348

Table 6
$(1/m_1^q, c_a^2) = (2.5, 2)$, $1/m_{i1}^s = (4, 3, 4, 3, 4, 3, 4)$, $c_{s,i}^2 = (2, 2, 2, 2, 2, 2, 2)$, $N_i = (4, 5, 6, 3, 6, 5, 4)$

	Approximate	Simulation	Relative err.
	1.589	1.719	0.0756
	2.442	2.653	0.0795
	2.454	2.661	0.0778
	1.602	1.648	0.0279
	1.789	1.848	0.0319
	2.181	2.118	− 0.0298
	1.073	1.032	− 0.0397
Throughput	2.0508	1.9643	− 0.0440

stage involving exactly one optimizing variable. The computations at the different stages are linked through recursive computations in a manner that yields a feasible optimal solution to the entire problem.

To facilitate exposition, we define the following additional notations:

K	total number of buffers that can be allocated
$\bar{N}_j(X) = \{N_1, N_2, \ldots, N_j\}$	the certain buffer allocation strategy for the first j workstations, and $X = \sum_{k=1}^{j} N_k$
$F_l(\bar{N}_j(X))$	the measure of lth kind of performance generated by the first j workstations that can be system throughput, WIP, and cycle time, when the allocation strategy is $\bar{N}_j(X)$
$f_{lj}(N_j)$	the measure of lth kind of performance generated by the jth workstations that can be throughput, WIP, and cycle time, when it is assigned N_j buffer capacities
$Z_j(X,\ \bar{N}_j(X))$	the multi-objective non-linear function for the first j workstations $Z_j(F_1(\bar{N}_j(X), F_2(\bar{N}_j(X), \ldots, F_p(\bar{N}_j(X))$
$g_j(N_j)$	the formulation of performance for the jth workstation when it is assigned N_j buffer capacities $g_j(f_{1j}(N_j), f_{2j}(N_j), \ldots, f_{pj}(N_j))$

Let the number of workstations represent stage, and the number of buffers, which have already been allocated, represent state. The type of mathematical programming problem can be written as follows:

$$\begin{aligned} Z_M^*(K,\ \bar{N}_M^*(K)) = \text{Extremize} \quad & Z_M(K,\ \bar{N}_M(K)) \\ \text{subject to:} \quad & \sum_{i=1}^{M} N_i = K, \\ & N_i \geqslant 0, \quad i = 1, 2, \ldots, M. \end{aligned} \tag{15}$$

In addition, the following formula summarizes the nature of the forwardly recursive dynamic programming algorithm:

$$\begin{pmatrix}\text{best performance}\\ \text{for the stages}\\ 1,2,\ldots,i+1 \text{ given}\\ \text{state } x_{i+1}\end{pmatrix} = \begin{matrix}\text{Max } or \text{ Min}\\ \text{over all feasible}\\ \text{alternatives of}\\ \text{stage } i+1\\ \text{given } x_{i+1}\end{matrix} \left\{ \begin{pmatrix}\text{performance of}\\ \text{the feasible}\\ \text{alternative for}\\ \text{stage } i+1\end{pmatrix} + \begin{pmatrix}\text{best performance}\\ \text{for stage } 1,2,\ldots,i\\ \text{given its state } x_i\end{pmatrix} \right\}, \tag{16}$$

where x_{i+1} is the amount of buffers allocated to stage $1, 2, \ldots, i+1$, x_i the amount of buffers allocated to stage $1, 2, \ldots, i$; x_{i+1} the amount of buffers allocated to given alternative of stage $i+1$.

Taking an example by minimization, we illustrate derivative process of the optimal solution as follows.

(1) The first two workstations

$$Z_2^*(x_2, N_2^*(x_2)) = \mathrm{Min}[g_1(x_1) + g_2(x_2 - x_1)], \quad x_2 = 0, 1, \ldots, K, \quad x_1 = 0, 1, 2, \ldots, x_2. \tag{17}$$

(2) The 3rd workstation to $(M-1)$th workstation

$$Z_i^*(x_i, \bar{N}_i^*(x_i)) = \mathrm{Min}[Z_{i-1}^*(x_{i-1}, \bar{N}_{i-1}^*(x_{i-1})) + g_i(x_i - x_{i-1})],$$
$$x_i = 0, 1, \ldots, K, \quad x_{i-1} = 0, 1, \ldots, x_i, \quad i = 3, 4, \ldots, M-1. \tag{18}$$

(3) The Mth workstation.
Since all available buffers have to be allocated exhaustively and disposed of entirely, so

$$Z_M^*(K, \bar{N}_M^*(K)) = \mathrm{Min}[Z_{M-1}^*(x_{M-1}, \bar{N}_{M-1}^*(x_{M-1})) + g_M(K - x_{M-1})], \quad x_{M-1} = 0, 1, \ldots, K. \tag{19}$$

Following the above dynamic programming process (as shown in Fig. 4), we could derive the optimal or sub-optimal buffer allocation strategy.

3.2. Numerical experiments

Sets of experiments were performed so as to validate the integration of the approximate algorithm and DP and to illustrate the efficacy of the design methodology. To this end, the results produced by our heuristic algorithm were compared with the results obtained by the enumerative method. Both methods were based on the approximate algorithm in which we have already justified in Section 2.

For simplification and demonstration purpose, the objective function was set to be

$$Z_M^*(K, \bar{N}_M^*(K)) = \text{Maximize } Z_M(K, \bar{N}_M(K)), \tag{20}$$

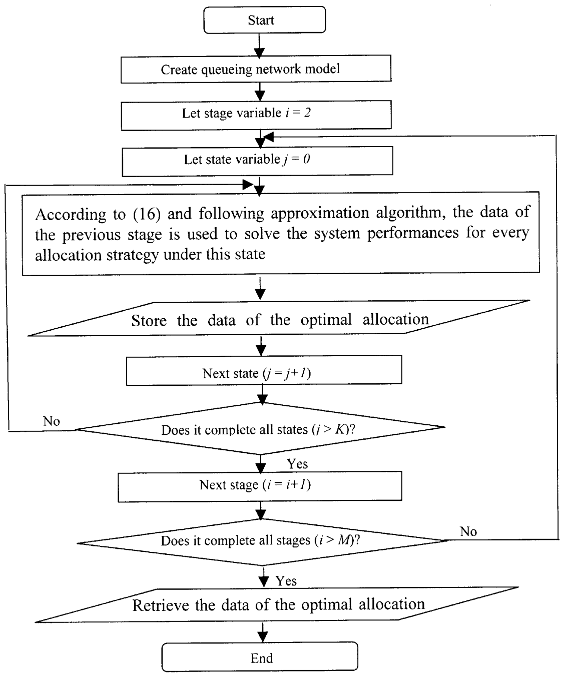
Start
Create queueing network model
Let stage variable $i = 2$
Let state variable $j = 0$
According to (16) and following approximation algorithm, the data of the previous stage is used to solve the system performances for every allocation strategy under this state
Store the data of the optimal allocation
Next state ($j = j+1$)
No
Does it complete all states ($j > K$)?
Yes
Next stage ($i = i+1$)
Does it complete all stages ($i > M$)?
No
Yes
Retrieve the data of the optimal allocation
End

Fig. 4. The search process of dynamic programming along with the embedded approximate analytic procedure of system performance.

where $Z_M(K,\ \bar{N}_M(K)) = F_{\text{throughput}}(\bar{N}_M(K))$ for the six examples described in Section 2.3. The search results of the optimal allocation strategy by heuristic algorithm and enumerative method were shown in Table 7. Though there were some discrepancy between the two strategies obtained, it is clear that the differences of system throughput under the two strategies were trivial for the most part. In our view, sacrificing slight precision is worthwhile saving much search time. In particular, it is more evident when the total available buffer capacity or number of workstations gets larger.

Nevertheless, the methodology might be further improved through a refined technique to enhance solution quality, if necessary. Among other things, a partial reason behind the difference between heuristic algorithm and enumerative method is the dependence of arrival processes between workstations such that in the search process of DP, the feasible optimal solution of the previous stage is not necessarily included in the optimal solution of the rear stage. To reduce the effect of dependence, we can reserve more than one of the optimal and sub-optimal solutions in the previous stage so as to increase the possibility that the real optimal solution is included in the rear stage. Table 7 shows the results and the improvement of throughput when we reserved two solutions in these experiments. Of course, this is a tradeoff between solution quality and computational efficiency (search time). We suggest that the manufacturing system should adopt the methods of enhancing solution quality when the space, which is available for allocation, is not sufficient and is difficult for enlarging, and blocking is very likely to happen in the manufacturing line. In the meantime, a more effective buffer allocation strategy might help to reduce the incidence of

Table 7
A comparison between allocation strategy and throughput obtained by heuristic algorithm and enumerative method, respectively

System pattern	Total buffer capacity	Heuristic search algorithm				Enumerative method	
		Reverse 1 solution		Reserve 2 solution		Alloc. str. $(N_1^*, \ldots, N_M^*)$	Throu.
		Alloc. str. $(N_1^*, \ldots, N_M^*)$	Throu.	Alloc. str. $(N_1^*, \ldots, N_M^*)$	Throu.		
$(1/m_1^a, c_a^2) = (2.5, 2)$, $1/m_{i1}^s = (4, 3, 4)$, $c_{s,i}^2 = (2, 2, 2)$	9	(5, 2, 2)	2.0802	(4, 4, 1)	2.0857	(5, 3, 1)	2.0895
$1/m_1^a, c_a^2) = (3, 2)$, $1/m_{i1}^s = (4, 3, 4)$, $c_{s,i}^2 = (2, 2, 2)$	10	(5, 3, 2)	2.3011	(4, 4, 2)	2.3072	(4, 4, 2)	2.3072
$(1/m_1^a, c_a^2) = (3.5, 2)$, $1/m_{i1}^s = (4, 3, 4)$, $c_{s,i}^2 = (2, 2, 2)$	12	(4, 4, 4)	2.5039	(4, 4, 4)	2.5039	(4, 4, 4)	2.5039
$(1/m_1^a, c_a^2) = (2, 2)$, $1/m_{i1}^s = (4, 3, 4, 3, 4)$, $c_{s,i}^2 = (2, 2, 2, 2, 2)$	13	(6, 2, 1, 2, 2)	1.7802	(6, 2, 2, 1, 2)	1.7877	(5, 3, 2, 2, 1)	1.8059
$(1/m_1^a, c_a^2) = (2.5, 2)$, $1/m_{i1}^s = (4, 3, 4, 3, 4)$, $c_{s,i}^2 = (2, 2, 2, 2, 2)$	15	(4, 2, 1, 4, 4)	1.9595	(5, 1, 2, 6, 1)	1.9805	(3, 4, 4, 3, 1)	2.0350
$(1/m_1^a, c_a^2) = (2.5, 2)$, $1/m_{i1}^s = (4, 3, 4, 3, 4, 3, 4)$, $c_{s,i}^2 = (2, 2, 2, 2, 2, 2, 2)$	16	(4, 1, 1, 2, 2, 5, 1)	1.6773	(3, 2, 1, 3, 5, 1, 1)	1.6934	(4, 3, 2, 2, 2, 2, 1)	1.7626

blocking. On the other hand, one should pursue computational efficiency rather solution quality if the space is sufficient.

4. Conclusion remarks

In this paper, we study the buffer allocation strategy of the flow-shop-type production system that possesses a given total amount of the buffers and finite buffer capacity for each workstation as well as general interarrival and service times. We hope to attain such system performances as minimizing work-in-process, cycle time and blocking probability, maximizing throughput, or their combinations according to the allocation strategy thus obtained. To this end, we present a design methodology that performs the buffer allocation using a dynamic programming process along with the embedded approximate analytic procedure for computing performance measures under a certain buffer allocation strategy. Although our methodology cannot overcome completely the dependent problem, the numerical experiments demonstrate the effectiveness of this design methodology. As a result, we are confident that using the design methodology presented by our paper could determine the buffer allocation quickly and quite precisely.

For future research, our developed heuristic algorithm could be generalized to cope with buffer allocation strategy under arbitrary configuration; i.e., job-shop-type production system (see Altiok and Perros [33]). Other extension involves treating buffer allocation strategy under multi-server queues with tandem or arbitrary network configuration.

Appendix A

A.1. Coxian-2 distribution (adapted from Jun and Perros [30])

Consider a Coxian distribution with two phases, denoted by *Cox*-2, and having parameters (μ_1, μ_2, θ). Its Laplace transform is as follows:

$$f^*(s) = (1-\theta)\frac{\mu_1}{s+\mu_1} + \theta\frac{\mu_1}{s+\mu_1}\frac{\mu_2}{s+\mu_2}. \tag{A.1}$$

The first moments of a *Cox*-2, m_1, m_2, and m_3, can be derived by following equations:

$$m_1 = \frac{1}{\mu_1} + \frac{\theta}{\mu_2}, \quad m_2 = 2\left(\frac{1}{\mu_1^2} + \frac{\theta}{\mu_1\mu_2} + \frac{\theta}{\mu_2^2}\right), \quad m_3 = 6\left(\frac{1}{\mu_1^3} + \frac{\theta}{\mu_1^2\mu_2} + \frac{\theta}{\mu_1\mu_2^2} + \frac{\theta}{\mu_2^3}\right). \tag{A.2}$$

For a general distribution with a squared coefficient of variation $c^2 > 0.5$, given its first three moments m_1, m_2, and m_3, the *Cox*-2 parameters can be determined through the following equations according to different conditions:

(1) $c^2 > 1.0$ and $m_3/m_1^3 > 1.5(c^2+1)^2$;

$$u = (6m_1m_2 - 2m_3)/(3m_2^2 - 2m_1m_3),$$
$$v = (12m_1^2 - 6m_2)/(3m_2^2 - 2m_1m_3),$$
$$\mu_1, \mu_2 = \tfrac{1}{2}[u \pm (u^2 - 4v)^{1/2}], \quad \theta = \mu_2(m_1 - 1/\mu_1). \tag{A.3}$$

(2) $c^2 > 1.0$ and $m_3/m_1^3 \leqslant 1.5(c^2+1)^2$;

$$\mu_1 = 2/m_1, \quad \mu_2 = 1/(m_1 c^2), \quad \theta = 1/(2c^2). \tag{A.4}$$

(3) $0.5 < c^2 \leqslant 1.0$;

$$\mu_1 = 1/(m_1 c^2), \quad \mu_2 = 2/(m_1), \quad \theta = 2(1-c^2). \tag{A.5}$$

A.2. Arrival-epoch probabilities of a $C_2/C_2/1/N$ queue (adapted from Jun and Perros [30])

The probability $q(n)$ and the probability $q(n, j)$ can be obtained as follows:

$$q(n,1) = q(n,1,1) + q(n,2,1), \quad q(n,2) = q(n,1,2) + q(n,2,2),$$

$$q(n) = q(n,1) + q(n,2). \tag{A.6}$$

Consider a $C_2/C_2/1/N$ queueing system with arrival *Cox*-2 parameters $ar = (a_1, a_2, \alpha)$, the probability $q(n, j, k)$ can be expressed in terms of the time-average probabilities $p(n, j, k)$ as follows

$$q(n,1,k) = (1-\alpha)a_1 p(n,1,k)/H, \quad k = 1,2,$$

$$q(n,2,k) = a_2 p(n,2,k)/H, \quad k = 1, 2, \tag{A.7}$$

where

$$H = (1-\alpha)a_1 \sum_{n=0}^{N} \sum_{k=1}^{2} p(n,1,k) + a_2 \sum_{n=0}^{N} \sum_{k=1}^{2} p(n,2,k). \tag{A.8}$$

Approximate algorithm for tandem queueing network model

Preparation step
0. Get $m_{i1}^s, m_{i2}^s, m_{i3}^s; i = 1, 2, \ldots, M$, and obtain $S_i^*(s)$ and $\Xi_i(\xi_{i1}, \xi_{i2}, \alpha_i)$; $i = 1, 2, \ldots, M$.
1. Get m_1^a, m_2^a, m_3^a.
2. Except for queue 1, add one fictitious buffer to each queue.

Starting step
0. Get $m_{i1}^a, m_{i2}^a, m_{i3}^a$; $i = 1, 2, \ldots, M$.
1. Set $i = M$.
2. Obtain $B_i^*(s)$ and $\Omega_i(\omega_{i1}, \omega_{i2}, \beta_i)$.
3. Obtain $O_i^*(s)$ and $m_{i1}^o, m_{i2}^o, m_{i3}^o$, and compute $\Lambda_i(\lambda_{i1}, \lambda_{i2}, \gamma_i)$.
4. Analyze queue i as a *Cox*-2/*Cox*-2/1/$N_i + 1$ queue and compute

$$p_i^{(1)}(n),\ p_i^{(1)}(n,j,k); \quad n = 0, 1, \ldots, N_i + 1; j = 1, 2; k = 1, 2,$$

$$q_i^{(1)}(n),\ q_i^{(1)}(n,j),\ q_i^{(1)}(n,j,k); \quad n = 0, 1, \ldots, N_i + 1; j = 1, 2; k = 1, 2,\ w_{i,j},\ j = 1, 2, \text{ and } \pi_{i-1}.$$

5. Set $i = i - 1$.
6. If $i = 1$, analyze queue 1 as a *Cox*-2/*Cox*-2/1/N_1 queue, compute $p_1^{(1)}(n)$; $n = 0, 1, \ldots, N_1$, and go to backward adjustment step. Otherwise, go to 2.

Backward adjustment step

0. Set $i = 1$.
1. Obtain $D_i^*(s)$, $E_{i+1}^*(s)$, and $O_{i+1}^*(s)$.
2. Calculate $m_{i+1,1}^o$, $m_{i+1,2}^o$, $m_{i+1,3}^o$ and obtain $\Lambda_{i+1}(\lambda_{i+1,1}, \lambda_{i+1,2}, \gamma_{i+1})$.
3. Keep $\Omega_{i+1}(\omega_{i+1,1}, \omega_{i+1,2}, \beta_{i+1})$ constant.
4. Analyze queue i as a $Cox\text{-}2/Cox\text{-}2/1/N_{i+1} + 1$ queue and compute

$$p_{i+1}^{(2)}(n),\ p_{i+1}^{(2)}(n, j, k);\quad n = 0, 1, \ldots, N_{i+1} + 1; j = 1, 2; k = 1, 2,$$

and

$$q_{i+1}^{(2)}(n),\ q_{i+1}^{2)}(n, j), q_{i+1}^{(2)}(n, j, k);\quad n = 0, 1, \ldots, N_{i+1} + 1; j = 1, 2; k = 1, 2.$$

5. If $i = M - 1$, go to forward adjustment step. Otherwise, set $i = i + 1$ and go to 1.

Forward adjustment step

0. Use the same procedure as in the starting step, except replace $a_i(t)$ with $e_i(t)$.
1. Analyze queue i as a $Cox\text{-}2/Cox\text{-}2/1/N_i + 1$ queue and compute

$$p_i^{(3)}(n),\ p_i^{(3)}(n, j, k);\quad n = 0, 1, \ldots, N_i + 1; j = 1, 2; k = 1, 2,$$

and

$$q_i^{(3)}(n),\ q_i^{(3)}(n, j),\ q_i^{(3)}(n, j, k);\quad n = 0, 1, \ldots, N_i + 1; j = 1, 2; k = 1, 2.$$

For queue 1, analyze it as a $Cox\text{-}2/Cox\text{-}2/1/N_1$ queue, compute $p_1^{(3)}(n)$; $n = 0, 1, \ldots, N_1$.

2. Test for convergence. If

$$\operatorname*{Max}_n |p_i^{(2)}(n) - p_i^{(3)}(n)| < \varepsilon,\quad i = 1, 2, \ldots, M, \tag{A.9}$$

then go to 3. Otherwise, go back to backward adjustment step.

3. The actual probability $p_i(N_i)$, $i \geqslant 2$, is obtained by summing $p_i(N_i)$ and $p_i(N_i + 1)$.
4. Calculate L_i and W_i; $i = 1, 2, \ldots, M$
5. Obtain WIP, Cycle time, and Throughput.
6. Stop.

References

[1] Reiman M. The optimal buffer allocation problem in light traffic. Proceeding of the 26th IEEE Conference Decision and Control, Los Angeles, CA, December 9–11, 1987. p. 1409–1503.

[2] Venkat A. The optimal buffer allocation problem. IEEE Transactions on Information Theory 1989;35:721–5.

[3] Conway RW, Maxwell WL, McClain JO, Thomas LJ. The role of work-in-process inventories in series production lines. Operations Research 1988;36:229–41.

[4] Hillier FS, So KC, Boling RW. Toward characterizing the optimal allocation of storage space in production line system with variable operation times. Management Science 1993;39:126–33.

[5] So KC. Optimal buffer allocation strategy for minimizing work-in-process inventory in unpaced production lines IIE. Transactions 1997;29:81–8.

[6] Kubat P, Sumita U. Buffer and backup machines in automatic transfer lines. International Journal of Production Research 1985;23:1259–80.

[7] Jafari MA, Shantikumar JG. Determination of optimal buffer storage capacities and optimal allocation in multistage automatic lines. IIE Transactions 1989;21:130–5.
[8] Hillier FS, So KC. The effect of the coefficient of variation of operation times on the allocation of storage space in production line system. *IIE* Transactions 1991;23:198–206.
[9] MacGregor Smith J, Daskalaki S. Buffer space allocation in automated assembly lines. Operations Research 1988;36:343–58.
[10] MacGregor Smith J, Chikhale N. Buffer allocation for a class of nonlinear stochastic knapsack problems. Annual Operational Research 1995;58:323–60.
[11] McClain JO, Moodie DR. A comment on buffer space allocation in automated assembly lines. Operations Research 1991;39:857–60.
[12] Singh A, Smith JM. Buffer allocation for an integer nonlinear network design problem. Computers and Operations Research 1997;24:453–72.
[13] Powell MJ. An efficient method for finding the minimum of a function of several variables without calculating derivatives. Computer Journal 1964;7:155–62.
[14] Kerbache L, MacGregor Smith J. The generalized expansion method for open finite queueing networks. European Journal of Operational Research 1987;32:448–61.
[15] Kerbache L, MacGregor Smith J. Asymptotic behavior of the expansion method for open finite queueing networks. Computers and Operations Research 1988;15:157–69.
[16] Brandwajn A, Jow YL. An approximation method for tandem queues with blocking. Operations Research 1988;36:73–83.
[17] Gershwin SB. An efficient decomposition method for the approximate evaluation of tandem queues with finite storage space and blocking. Operations Research 1987;35:291–305.
[18] Shanthikumar JG. Open queueing network models of dynamic job shops. International Journal of Production Research 1981;19:255–66.
[19] Altiok T. Approximate analysis of exponential tandem queue with blocking. European Journal of Operational Research 1982;11:390–8.
[20] Boxma O, Konheim A. Approximate analysis of exponential queueing system with blocking. Acta Informatica 1981;15:19–66.
[21] Perros HG, Altiok T. Approximate analysis of open networks of queues with blocking: tandem configuration. IEEE Transactions Software Engineering 1986;SE-12:450–61.
[22] Perros HG, Snyder P. Computationally efficient approximation algorithm for analyzing open queueing networks with blocking. Performance Evaluation Journal 1988/1989;9:217–224.
[23] Altiok T. Approximate analysis of queue in series with phase-type service time and blocking. Operations Research 1989;37:601–10.
[24] Harrison JM, Pich MT. Two-moment analysis of open queueing networks with general workstation capabilities. Operations Research 1996;44:936–50.
[25] Shi L. Approximate analysis for queueing networks with finite capacity and customer loss. European Journal of Operational Research 1995;85:178–91.
[26] Whitt W. The queueing network analyzer. Bell System Technical Journal 1983;62:2779–815.
[27] Whitt W. Variability functions for parametric-decomposition approximations of queueing networks. Management Science 1995;41:1704–15.
[28] Bronshtein O, Gertsbakh IB. An open exponential queueing network with limited waiting spaces and losses: a method of approximation analysis. Performance Evaluation 1984;4:31–43.
[29] Altiok T. On the phase-type approximation of general distributions. AIIE Transactions 1985;17:110–6.
[30] Jun KP, Perros HG. An approximate analysis of open tandem queueing networks with blocking and general service time. European Journal of Operational Research 1989;46:123–35.
[31] Kuehn PJ. Approximation analysis of general networks by decomposition. IEEE Transactions on Communication 1979;27:113–26.
[32] Yao DD, Buzacott JA. Queueing models for a flexible machining station, Part II: the method of Coxian phases. European Journal of Operational Research 1985;19:241–52.
[33] Altiok T, Perrors HG. Approximate analysis of arbitrary configurations of queueing networks with blocking, Annals of OR 1987;9:481–509.

Ming-Guan Huang is a lecturer at the Department of Industrial Engineering and Management, Tahwa Institute of Technology. He received a BS in Computer Science and Information Engineering and an MS in Industrial Engineering from National Chiao Tung University. He is currently a Ph.D. candidate at the institute of Business and Management, National Chiao Tung University. His major research interests are Production Planning and Management, Manufacturing System Evaluation, Quantitative Methods, and Information Management.

Pao-Long Chang is a professor at the Institute of Business and Management, National Chiao Tung University, Taiwan. He received a BS in Mathematics from Fu-Jen Catholic University in Taiwan, and MA in Mathematics from State University of New York at Albany and a Ph.D. in Mathematics from the University of Washington, in USA. His previous articles have appeared in the *Journal of the Operations Research Society, Japan, Journal of the Operational Research Society, Journal of Environmental Management, Computers and Operations Research, International Journal of Production Economics, Technovation, International Journal of Technology Management and IEEE Transactions on Engineering Management*. His current research interests are in the areas of Operations Research and Technology Management.

Ying-Chyi Chou is a Ph.D. candidate at the Institute of Business and Management, National Chiao Tung University. She received a BS in Mathematics from National Tsing Hua University and an MS in Statistics from National Chung University. Her major research interests are in the areas of Statistics, Production Planning and Management, and Inventory Management