PERFORMANCE EVALUATION OF A SHARED ECHELON BUFFER POLICY IN SERIAL PRODUCTION LINES WITH EXPONENTIAL PROCESSING TIMES

by

GEORGIOS THEODOSIOU

Diploma in Electrical and Computer Engineering,

Aristotle University of Thessaloniki, 2013

A THESIS

submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

DEPARTMENT OF MECHANICAL ENGINEERING

SCHOOL OF ENGINEERING

UNIVERSITY OF THESSALY

2016

Approved by Members of Advisory Committee:

A. Supervisor:	George Liberopoulos, Professor, Department of Mechanical Engineering, University of Thessaly
B. Member:	Dimitris Pandelis, Associate Professor, Department of Mechanical Engineering, University of Thessaly
C. Member:	Georgios Saharidis, Assistant Professor, Department of Mechanical Engineering, University of Thessaly

Σύνοψη

Θεωρούμε μια γραμμή παραγωγής που αποτελείται από N μηχανές σε σειρά με $N-1$ ενδιάμεσους πεπερασμένους αποθηκευτικούς χώρους, τοποθετημένους μεταξύ ζευγών διαδοχικών μηχανών. Τα προϊόντα ξεκινούν την επεξεργασία τους από την πρώτη μηχανή, επισκέπτονται κάθε μηχανή με συγκεκριμένη σειρά και εγκαταλείπουν το σύστημα από την N-στή μηχανή. Οι χρόνοι επεξεργασίας των μηχανών θεωρούνται εκθετικά κατανεμημένες τυχαίες μεταβλητές. Κατά τον παραδοσιακό τρόπο λειτουργίας μιας τέτοιας γραμμής, μια μηχανή επιτρέπεται να επεξεργάζεται ένα προϊόν αν υπάρχει διαθέσιμος χώρος ενδιάμεσα αυτής και της επόμενης μηχανής. Αναφερόμαστε σε αυτό τον τρόπο λειτουργίας ως πολιτική εγκατεστημένων αποθηκευτικών χώρων (installation buffer policy). Σε αυτή την εργασία, θα διερευνήσουμε μια πολιτική που στοχεύει στην αύξηση της χρήσης των χώρων αυτών επιτρέποντας σε μια μηχανή να αποθηκεύει τα προϊόντα που παράγει στο συνολικό αποθηκευτικό χώρο μεταξύ αυτής και της τελευταίας μηχανής. Ο συνολικός χώρος στα κατάντη μιας μηχανής αναφέρεται ως κλιμακωτός αποθηκευτικός χώρος (echelon buffer) και αντιστοιχεί στο σύνολο των εγκατεστημένων χώρων στα κατάντη αυτής της μηχανής· η προκύπτουσα πολιτική αναφέρεται ως πολιτική κλιμακωτών αποθηκευτικών χώρων. Από την οπτική των αποθηκευτικών χώρων, κάθε ενδιάμεσος χώρος είναι κοινόχρηστος από όλες τις μηχανές στα ανάντη της γραμμής. Η πολιτική κλιμακωτών αποθηκευτικών χώρων χρησιμοποιεί καθολικά δεδομένα αφού επιτρέπει σε κάθε μηχανή να επεξεργάζεται προϊόντα βάσει του επιπέδου αποθέματος παραγωγής σε εξέλιξη (WIP) ολόκληρου του τμήματος της γραμμής στα κατάντη αυτής της μηχανής. Αντίθετα, η πολιτική εγκατεστημένων αποθηκευτικών χώρων χρησιμοποιεί μόνο τοπικά δεδομένα αφού επιτρέπει σε κάθε μηχανή να επεξεργάζεται προϊόντα βασισμένη στο επίπεδο αποθέματος σε εξέλιξη του τοπικού εγκατστημένου χώρου που έπεται αμέσως μετά την μηχανή. Για να αξιολογήσουμε την απόδοση της γραμμής στο πλαίσιο της πολιτικής κλιμακωτών αποθηκευτικών χώρων, αναπτύξαμε μια προσεγγιστική μέθοδο αποσύνθεσης του αρχικού συστήματος με N μηχανές και $N-1$ κλιμακωτούς αποθηκευτικούς χώρους σε υποσυστήματα 2 μηχανών, όπου κάθε υποσύστημα έχει ένα ενδιάμεσο πεπερασμένο αποθηκευτικό χώρο που αντιπροσωπεύει ένα κλιμακωτό αποθεματικό χώρο στην αρχική γραμμή. Για την περίπτωση που οι μηχανές έχουν εκθετικά κατανεμημένους χρόνους επεξεργασίας (ανάλογο συνεχούς χρόνου του μοντέλου αξιοπιστίας Bernoulli), μοντελοποιούμε κάθε υποσύστημα 2-μηχανών σαν μια δισδιάστατη Μαρκοβιανή αλυσίδα που μπορεί να λυθεί αριθμητικά. Οι παράμετροι των υποσυστημάτων των 2 μηχανών προσδιορίζονται από τις σχέσεις ροής των προϊόντων μέσω των κλιμακωτών αποθηκευτικών χώρων στο αρχικό σύστημα. Για την επίλυση των σχέσεων αυτών σχεδιάσαμε έναν επαναληπτικό αλγόριθμο. Η αριθμητική εφαρμογή δείχνει ότι η μέθοδος αυτή είναι υψηλής ακριβείας και υπολογιστικά αποδοτική.

Λέξεις κλειδιά: γραμμή παραγωγής; κλιμακωτός αποθηκευτικός χώρος; αξιολόγηση απόδοσης; αποσύνθεση; προσομοίωση.

Abstract

We consider a production line consisting of N machines in series with $N-1$ finite intermediate buffers located between consecutive machine pairs. The parts begin their processing in the first machine, visit each machine in the line in a fixed order and leave the system from the Nth machine. The processing times of the machines are assumed to be exponentially distributed random variables. In the traditional way of operating such a line, a machine is allowed to process a part if there is space in the single intermediate buffer between it and the next machine. We refer to this way of operation as installation buffer (IB) policy. In this thesis, we investigate a policy aimed at increasing the utilization of buffers by allowing a machine to store the parts that it produces in the total buffer space between it and the last machine. The total buffer downstream of a machine is referred to as echelon buffer and corresponds to the ensemble of the installation buffers downstream of this machine; the resulting policy is referred to as echelon buffer (EB) policy. From the point of view of buffers, under the EB policy, each intermediate buffer is shared by all its upstream machines. The EB policy uses global information because it enables each machine to process parts based on the WIP level of the entire part of the line downstream of this machine. On the contrary, the IB policy uses only local information because it enables each machine to process parts based on the WIP level of the local installation buffer immediately following this machine. To evaluate the performance of the line under the EB policy, we develop an approximation method that decomposes the original system with N machines and $N-1$ echelon buffers into $N-1$ 2-machine subsystems, where each subsystem has an intermediate finite buffer representing one of the echelon buffers in the original line. For the case where the machines have exponentially distributed processing times (continuous time analog of the Bernoulli reliability model), we model each 2-machine subsystem as a 2D Markov chain that can be solved numerically. The parameters of the 2-machine subsystems are determined by relationships among the flows of parts through the echelon buffers in the original system. An iterative algorithm is developed to solve these relationships. Numerical experimentation shows that this method is computationally efficient and highly accurate.

Keywords: production line; echelon buffer; performance evaluation; decomposition; simulation.

Table of Contents

List of Figures and Tables

Chapter 1 - Introduction

Production lines are series of work centers connected linearly, each one consisting of one or more identical machines in parallel. Every part visits each workstation in a fixed order, starting at the first machine, and leaving after the last machine. They are of economic importance since they are used in high volume manufacturing, particularly automobile production where they make engine blocks, transmission cables, cylinders, connecting rods etc. Their capital costs range from hundreds of thousands dollars to tens of millions of dollars. Stochastic production lines are subject to disturbances arising from variations in processing times and unpredictable failures of the workstations involved. This can cause the machines to stop producing and can lower the throughput of the line. In order to mitigate the effect of such disturbances, it is customary to install buffers between the machines so that parts flow from machine to buffer to machine and so on until they exit the line. Other ways of increasing throughput are to raise the processing rates of the machines starting with the slowest one, or to reduce the variance of processing times causing congestion in the line. However, these techniques require good handling at the machine level and possibly investing in new equipment. Inserting a buffer between two machines speeds up the line by decoupling the operation of the machines, as long as this buffer is neither full nor empty, hence limiting the propagation of processing time delays. We refer to such a buffer as installation buffer because it locally stores parts that are produced by its upstream machine (installation). We also refer to the resulting operating policy as installation buffer (IB) policy. An example of a production line with 4 machines denoted by $\mathrm{M}_n, n=1\ldots4$ and 3 intermediate installation buffers denoted by $B_n^{'}, n=1,2,3$ is shown in Figure 1. The capacities of these buffers are denoted by $K_n^{'}, n=1,2,3$.

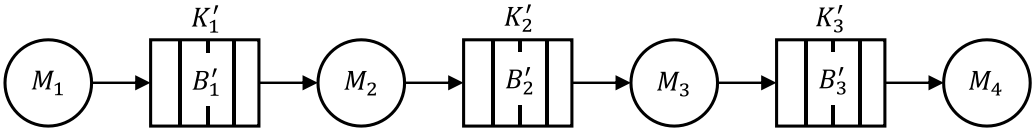

Figure 1.Serial production line with finite intermediate installation buffers operated under an IB policy.

Although higher buffer capacity increases the production rate, it is not beneficial to install as many buffers as possible. In buffer optimization, the buffer capacity and the buffer distribution constitute important decision variables in the optimization of queueing networks and production systems in particular. Even if the total capacity has been optimized, storing parts locally in the installation buffers

between the machines does not take full advantage of this capacity. Apart from that, inserting buffers between machines comes at a cost of additional capital investment, floor space, and work-in-process (WIP) inventory. Depending on the industry, such a cost can be quite high. When the cost of buffer space is significant, it may be worthwhile to consider increasing the utilization of the existing buffers before setting out to increase total buffer capacity. In practice, it is not unusual for a line manager to route parts produced by a machine to buffers other than the machine's designated installation buffer, if that buffer is full. Such an action effectively increases the utilization of the buffers. The question is, can this be done systematically, and if so, what are the gains and losses in performance?

1.1 How the policy works

If the physical layout of the production line is the traditional layout of machines in series with intermediate buffers, as shown in Figure 1, then under the EB policy, a part produced by machine M_n will be physically stored in the first non-full installation buffer downstream of M_n, as shown in Figure 2. In this case, each installation buffer is shared by all its upstream machines. Under the classical IB policy, on the other hand, each installation buffer is used only by the machine that directly precedes it, as is shown in Figure 1.

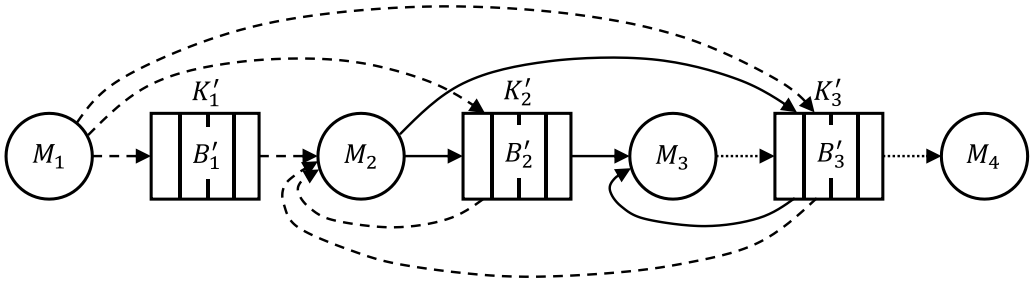

Figure 2.Serial production line with finite intermediate installation buffers operated under an EB policy.

If the physical layout of the production line is one where there is a common storage area on the side of the line (or in the interior of the line, if the line is U-shaped or L-shaped), then under the EB policy, this area is divided into compartments that play the same role as the intermediate installation buffers in the traditional serial layout. In this case, the flow of parts is identical to that in the serial layout shown in Figure 2, except that the buffers (compartments) are adjacent, as is shown in Figure 3(a).

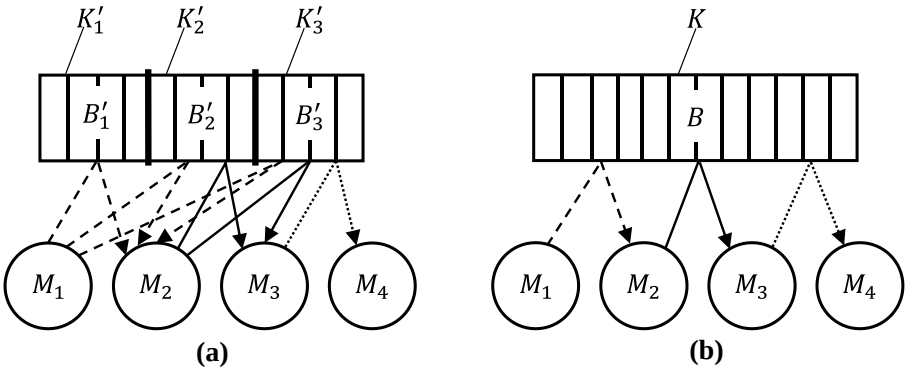

Figure 3.Serial production line with a common storage area divided into (a) several compartments or (b) a single compartment, operated under an EB policy.

To further clarify how the EB policy works, in both Figures 2 and 3(a), the first machine, M_1, can store parts in installation buffers, $B_1^{'}, B_2^{'}$ or $B_3^{'}$ with $B_1^{'}$ having the highest priority and $B_3^{'}$ the lowest. Similarly, machine M_2 can store parts in buffers $B_2^{'}$ or $B_3^{'}$, whereas machine M_3 can store parts only in buffer $B_3^{'}$. Clearly, if the capacities of buffers $B_1^{'}, B_2^{'}$ are zero while the capacity of $B_3^{'}$ is positive (i.e., if $K_1^{'} = K_2^{'} = 0$ and $K_3^{'} = K > 0$), then any machine will be allowed to store the parts that it produces in a single common buffer, as is shown in Figure 3(b). In this case, the EB policy reduces to a WIP-cap policy where every machine operates freely without ever being blocked, except for the first machine that is allowed to process a part only if there is space in the common buffer. This way of operation is effectively identical to the operation of a CONWIP system (Spearman et al. 1990). The only difference is that under CONWIP, the total WIP is constant because by definition it also includes the parts waiting to be processed on the first machine, whereas under the EB policy with $K_n^{'} = 0, n = 1, \ldots, N-2,$ and $K_{N-1}^{'} = K > 0$, the total WIP does not include the parts that are waiting to be processed on the first machine and hence is limited rather than being constant. For the purposes of this thesis, we will henceforth refer to an EB policy with $K_n^{'} = 0, n = 1, \ldots, N-2,$ and $K_{N-1}^{'} = K > 0$, as CONWIP.

1.2 Related literature

Most of the issues regarding flow line analysis that have been studied by the 1990's fall into one of three categories: (a) modeling aspects, (b) performance evaluation, and (c) optimization.

The major classes of manufacturing flow line models and their main features and properties such as blocking, processing times, failures and repairs, conservation of flow, rate-idle time and reversibility, including relationships among them, have been described by Dallery and Gershwin [1]. Comprehensive reviews and analyses of production lines can be also found in Gershwin [2]. Bernoulli machine reliability model is present in studying various aspects of production lines, e.g. production variability [3], transient behavior [4] and bottleneck identification [5]. This model allows simplicity in capturing the stochastic nature of machine processing times. Its continuous-time equivalent, the exponential processing time assumption, has also been used extensively in the literature [6][7].

The literature on production lines or tandem queues with finite inter-stage capacities contains various models of performance evaluation emphasizing exact or approximate computation of the steady-state performance measures of the system. The exact approaches are limited to two - or three - station lines. Exact models include Buzacott [8] and Gershwin and Berman [9], among others. These studies mainly focus on the solution of the steady-state flow-balance equations of the underlying Markov chains. Longer flow lines have been analyzed by using either numerical methods (e.g., the power method) or approximations. Various models of longer lines approximate the measures of performance by decomposing the system into a set of smaller systems (usually 2-machine, 1-buffer pseudo lines) that relate to each other within an iterative scheme [10][11]. Typically, in the first step of such methods, the performance of each 2-machine pseudo line is evaluated given the parameters of the two machines. In the second step, the parameters of the 2-machine pseudo line are determined by relationships among the flows of parts through the intermediate buffers of the original system. Good references where different models of 2-machine systems have been analyzed, include early and simple models [9] [12] [13] and more general models [14] [15] [16]. The majority of studies focuses on the impact of failures as well as buffer capacities on the measures of performance.

Most studies concern production lines operated under the traditional IB policy. Under this policy, the upstream (downstream) machine in each 2-machine pseudo line essentially represents the entire part of the line upstream (downstream) of the corresponding buffer and is only affected by the state in that part of the system; hence the decoupling effect of the buffer is clear. Under the EB policy however, the decoupling effect is more complex because parts produced by a machine may have to be temporarily stored in the installation buffer of another machine further downstream the line before returning upstream again for further processing. To address this complexity, special attention is required. The important concept of echelon stock was introduced by Clark [17]. In a multi-stage uncapacitated inventory system, a stage's echelon stock is the inventory position of the subsystem consisting of the stage itself and all its downstream stages. Axsäter and Rosling [18] show that for (Q,r)

rules, echelon-stock policies are in general superior to installation-stock policies whereby each stage follows a (Q,r) policy based on its local inventory position.

The idea of temporarily storing parts in shared buffers when the intermediate dedicated buffers following the machines are full, while often used in practice, has not been thoroughly investigated in the literature. Tempelmeier et al. [19][20] have made one of the first attempts to model a flexible manufacturing system (FMS) with some sharing of buffer space. The FMS consists of several workstations, where each workstation has one or more machines and a local finite buffer that is primarily used to store parts waiting to be processed by the machines; however, if needed it may also store parts that have finished their processing on the machines. More specifically, once a part has been processed at a workstation, it is normally stored in the buffer of the next workstation unless that buffer is full, in which case the part is stored in a common central buffer. If the central buffer is full too, the part is stored in the local buffer of the workstation that produced it if there is still space in it; otherwise, it stays on the machine blocking it from processing a new part. Parts are mounted onto pallets that come in a fixed number. To evaluate the performance of the system, they model the FMS as a closed queueing network (CQN) with blocking and solve it using numerical approximation techniques. Ferrari and Matta [21] present an approximate analytical method, based on decomposition techniques, that assesses the physical performance of small flow lines with both dedicated and shared buffer. Their aim of the analytical method is to capture the interdependent behaviour of the machines in the line due to the shared buffers. Their method deals with discrete and deterministic processing time, limited buffer capacity, and both time to repair (TTR) and time between failure (TBF) follow a geometric distribution. They assess the accuracy of the analytical solutions with respect to results provided by simulation.

A large number of researchers have addressed queueing network modeling of manufacturing systems. Papadopoulos and Heavey [22] provide a bibliography of material concerned with modeling of production and transfer lines using queueing networks. Queues in series with exponential processing times are decomposed by Hillier and Boling [23] and Altiok [24] among others. Hillier and Boling also developed expressions for the throughput of a serial production line of exponential machines with finite buffers by modeling them as continuous time Markov chains. Later, Hendricks [25] developed a technique to analytically describe the output process of a serial production line of N machines with exponential processing time distributions and finite buffer capacities. He used extensive exact results to examine the effects of line length, buffer capacity, and buffer placement on the inter-departure distribution, correlation structure, and variability of the output process of the production line. He used

results to determine the extent to which buffer allocation can be used to control the variability of the output process (and thereby the amount of work-in-process required to downstream subsystems). Springer [26] proposed an approximation for estimating the throughput rate and work-in-process inventory of finite-buffered exponential queues in series. He applied it to several sets of previously published test problems and found it performed well relative to existing models. In addition, he conducted a large simulation experiment to examine the robustness of the approximation under a wide range of parameter settings. Zhou and Lian [27] consider a 2-stage tandem network where each stage has a single exponential server. Customers arrive to the first stage following a Poisson process, and the waiting customers in the two stages share all or part of a common finite buffer. By constructing a Markov process, they derive the stationary probability distribution of the system and the sojourn time distribution. Their model, although limited to two servers, is somewhat similar to ours if one considers the external arrivals as being generated by a machine.

Finally, Koukoumialos and Liberopoulos [28] develop an analytical approximation method for the performance evaluation of a multi-stage production inventory system operated under an echelon kanban (EK) policy. The Kanban system is a message-passing mechanism that transfers the demand generated by the downstream stages to the upstream production stages in pull systems. Kanban-type production/inventory systems are often modeled as queueing networks in the literature. Consequently, most of the techniques that have been developed for the analysis of kanban-type production/inventory systems are based on methods for the performance evaluation of queueing networks. The connection between the EK policy and the EB policy becomes evident once the association between the number of available echelon kanbans and the number of available buffer spaces is made. In the EK system, each stage has an input buffer and is an open queueing network of machines with load-dependent continuous-time processing rates. The main production unit in this paper, on the other hand, is a machine with no input buffer. As a result of this difference, in the EK system, blockages happen at output buffers rather than on machines. Moreover, in the EK system, the analysis of each subsystem in isolation involves a product-form approximation technique for solving a CQN problem, whereas in the EB system, each subsystem is evaluated using exact continuous-time Markov chain analysis. As a result, the accuracy of the decomposition method is higher in the EB system than it is in the EK system.

1.3 Thesis outline

The remainder of this thesis is organized as follows. In Chapter 2, we present a continuous-time queuing network model of a production line controlled by an EB policy. In Chapter 3, we develop a

decomposition approximation method for evaluating the performance of the EB policy for the case where the machines have exponentially distributed processing times. The method that we develop is based on decomposing the original line with machines and echelon buffers into 2-machine pseudo lines where each pseudo line has an intermediate finite buffer representing one of the echelon buffers in the original line. The parameters of the 2-machine subsystems are determined by relationships among the flows of parts through the echelon buffers in the original system. An iterative algorithm is developed to solve these relationships. We present the analysis of each subsystem in isolation, and we develop the analysis for the entire system. In Chapter 4, we present numerical results on the performance of the decomposition method and on the effect of system parameters on performance. We evaluate the accuracy of this method by comparing it against simulation. Finally, we draw conclusions in Chapter 5.

Chapter 2 - Model of a serial production line with finite echelon buffers

We consider a serial production line consisting of N machines, denoted by $M_1, M_2, ..., M_N$, separated by $N-1$ infinite capacity buffers, denoted by $B_1^{''}, B_2^{''}, ..., B_{N-1}^{''}$. In what follows, we will make some necessary assumptions and then describe the model and how it works.

2.1 Assumptions of the analytical model

a) Parts flow from outside the system to M_1 to $B_1^{''}$ to M_2 to … to $B_{N-1}^{''}$ to M_N and exit the system.

b) Time is continuous.

c) The time to transfer parts from machines to remote installation buffers (if the nearby designated installation buffers are full) and back is negligible compared to the processing times on the machines.

d) $M_n, n = 1, ..., N$ produces a part with rate l_n unless it is starved or blocked. This implies that the processing time of a part on machine M_n is exponentially distributed with mean l_n^{-1}, variance l_n^{-2}, and relative standard deviation 1. Suppose T_{p_n} is the processing time of a part by machine M_n; then from cumulative distribution function:

$$P\{t_1 < T_{p_n} < t_2\} = \int_{t_1}^{t_2} l_n e^{-l_n t} = e^{-l_n t_1} - e^{-l_n t_2} \Rightarrow$$

$$P\{0 < T_{p_n} < \Delta t\} = 1 - e^{-l_n \Delta t} = l_n \Delta t + O(\Delta t^2) \underset{\Delta t \to 0}{=} l_n \Delta t.$$

e) The number of parts in buffer $B_n^{''}, n = 1, ..., N-1$, including the part in machine M_{n+1}, in period t, is denoted by $X_n^{''}(t)$ and is referred to as the stage WIP following machine M_n; hence, $B_n^{''}$ is referred to as the stage buffer following M_n.

f) When a part flows from machine M_n to buffer $B_n^{''}$, a token is generated and is placed in an associated finite buffer denoted by $B_n, n = 1, ..., N-1$. This token is removed from B_n and is attached onto the part when the part enters the last machine, M_N. The token is discarded when the part leaves M_N.

g) The number of tokens in buffer $B_n, n=1,...,N-1$, including the token that is attached onto the part in M_N, in period t, is denoted by $X_n(t)$ and is referred to as the echelon WIP downstream of M_n because it is equal to the sum of the stage WIP levels downstream of M_n, i.e.,

$$X_n(t) = \sum_{m=n}^{N-1} X_m''(t),\ n=1,\ldots,N-1; \quad (1)$$

hence, B_n is referred to as the echelon buffer following M_n.

h) The capacity of echelon buffer B_n is finite and is denoted by $K_n, n=1,...,N-1$; K_n is equal to the sum of the capacities of the physical installation buffer B_n' and all its downstream installation buffers in Figures 2 and 3(a), i.e.,

$$K_n = \sum_{m=n}^{N-1} K_m', n=1,...,N-1 \quad (2)$$

Alternatively, K_n' can be written in terms of K_n as follows:

$$K_n' = K_n - K_{n+1}, n=1,...,N-2 \quad (3)$$

$$K_{N-1}' = K_{N-1} \quad (4)$$

Expressions (3) and (4) imply that K_n is non-increasing in n, i.e., $K_n \geq K_{n+1}, n=1,...,N-2$ and $K_{N-1} > 0$.

i) Machine $M_n, n=1,...,N-1$, is blocked before service if $X_n(t) = K_n$. Machine M_N is never blocked.

j) Machine $M_n, n=2,...,N$ is starved if $X_{n-1}''(t) = 0$, or equivalently, if (i) $X_{n-1}(t) = X_n(t)$, for $n=2,...,N$ or (ii) $X_n(t) = 0$, for $n=N$. Machine M_1 is never starved.

2.2 Model structure and functioning

The serial production line with echelon buffers described above can be modeled as a continuous - time queueing network with exponentially distributed service times and blocking before service. We denote this network by L. Figure 4 shows such a network for $N=4$ machines. Machines $M_n, n=1,...,N-1$, behave as disassembly (split) servers because every time they produce a part, they also generate a token; the part moves to stage buffer B_n'' and the token moves to echelon buffer B_n. Machine M_N behaves as an assembly (merge) server because every time it is empty, it draws a part

from buffer B''_{N-1} (assuming one is available) and one token from each of the echelon buffers $B_1,...,B_{N-1}$. These tokens are assembled (merged) onto the part. When the part is processed, it leaves the line, and the tokens that were attached to it are discarded.

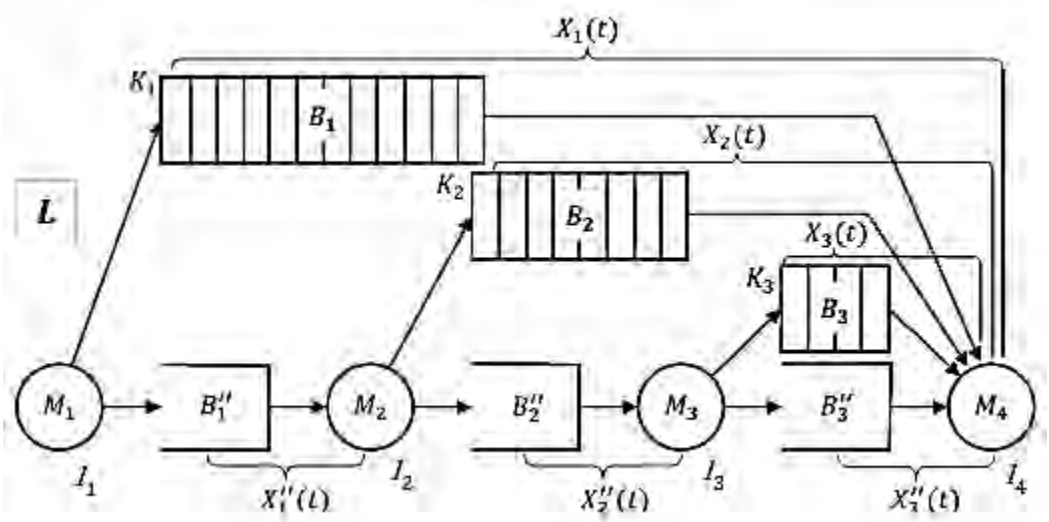

Figure 4. Queueing network model of a serial production line operated under an EB policy.

The exponential processing time assumption (d) is a simple assumption for capturing the randomness of machine processing times. As we will see later, the method that we develop in this paper for analyzing the system allows us to also deal with the more general case where each machine $M_n, n=2,\ldots,N,$ has load-dependent production rates, $l_n(x''_{n-1})$, where x''_{n-1} is the current value of $X''_{n-1}(t)$. Such a case can be used to model situations where the effective processing times are affected by the workload. The existence of such situations has been supported by empirical evidence.

The non-increasing echelon buffer capacity assumption (h) is redundant. It is stated to stress the fact that the behavior of a production line with $K_{n+1} \geq K_n$ for some n is identical to the behavior of the same line in which $K_{n+1} = K_n$. This assumption can be written equivalently in terms of parameters K'_n as follows: $K'_n \geq 0, n=1,...,N-2$, and $K'_{N-1} > 0$. In fact, if $K_{n+1} = K_n$ (equivalently, $K'_n = 0$), it is easy to see that echelon buffer B_{n+1} is obsolete and can be eliminated. With this in mind, note that the behavior of a line in which $K_n = K > 0, n=1,...,N-2$, (equivalently, $K'_n = 0, n=1,...,N-2$ and $K'_{N-1} = K > 0$), is equivalent to the behavior of the same line in which all echelon buffers except B_1 (equivalently all installation buffers except B'_{N-1}) have been eliminated, as is shown in Figure 3(b). The total WIP in such a line is limited by K, and therefore the line operates under a CONWIP policy,

as was mentioned earlier, with the reminder that in CONWIP, as was originally defined, the total WIP is constant because it includes parts waiting to be processed by M_1. Finally, note that although buffer $B_n^{''}$ has infinite capacity, the number of parts in it effectively is limited by K_n.

To further clarify the difference between the three types of buffers discussed thus far, note that installation buffers $B_n^{'}$ in Figures 2 and 3(a) are physical buffers that hold actual parts, whereas stage buffers $B_n^{''}$ in Figure 4 are functional buffers. More specifically, the entities in $B_n^{''}$ represent parts that have been produced by machine M_n and are physically stored in one of the installation buffers downstream of M_n. Finally, echelon buffers B_n are information buffers. The number of tokens in B_n represents the total number of parts residing in the physical installation buffers downstream of M_n. Hence, the number of empty spaces in B_n represents the total number of empty spaces in the physical installation buffers downstream of M_n.

In the following chapter we develop an approximation method for evaluating the performance of a production line operated under the EB policy described above. This method is based on (i) decomposing the long line (system) into many smaller parameterized lines (subsystems) that are easier to analyze and (ii) setting the parameters of these subsystems so the flow of parts through them mimics the flow of parts through the original system.

Chapter 3 - Decomposition of the production line model and parameter determination

Let us define the state of the queueing network model of a production line operated under an EB policy described in the previous chapter as the vector of echelon WIP levels $\mathbf{X}(t) = (X_1(t), X_1(t), ..., X_{N-1}(t))$. Clearly, $\{\mathbf{X}(t), t \geq 0\}$ is a continuous-time Markov chain. To find the number of states of this chain, we note that the echelon WIP levels satisfy $X_{n+1}(t) \leq X_n(t) \leq K_n, n = 1, ..., N-2$ and $0 \leq X_{N-1}(t) \leq K_{N-1}$. These inequalities can be written in terms of the stage WIP levels $X_n^{"}(t)$ as follows: $0 \leq X_n^{"}(t) \leq K_n - \sum_{m=n+1}^{N-1} X_m^{"}(t), n = 1, ..., N-2$ and $0 \leq X_{N-1}^{"}(t) \leq K_{N-1}$. Using these inequalities, we can express the total number of states of the Markov chain under the EB policy, denoted by NS^e, as follows:

$$NS^e = \sum_{x_{N-1}^{"}=0}^{K_{N-1}} \sum_{x_{N-2}^{"}=0}^{K_{N-2}-x_{N-1}^{"}} \cdots \sum_{x_n^{"}=0}^{K_n - \sum_{m=n+1}^{N-1} x_m^{"}} \cdots \sum_{x_2^{"}=0}^{K_2 - \sum_{m=3}^{N-1} x_m^{"}} \sum_{x_1^{"}=0}^{K_1 - \sum_{m=2}^{N-1} x_m^{"}} 1 \qquad (5)$$

This number can become very large even for problems of modest size and is certainly much larger that the corresponding number of states under the classical installation policy, denoted by NS^i given by

$$NS^i = \prod_{n=1}^{N-1} (K_n^{'} + 1) \qquad (6)$$

To get an idea of the relative magnitudes of NS^e and NS^i, consider a production line with $N = 7$ machines and installation buffer capacities $K_n^{'} = 5, n = 1, ..., 6$ corresponding to echelon buffer capacities $K_1 = 30, K_2 = 25, K_3 = 20, K_4 = 15, K_5 = 10, K_6 = 5$. From expressions (5) and (6), the number of states for this system under the EB and IB policies is $NS^e = 749.398$ and $NS^i = 46.656$, respectively.

Given the explosion in the number of states of the Markov chain model of a production line operated under an EB policy, in this chapter, we develop an approximation method for evaluating the performance of such a line. This method is based on decomposing the original system of N machines and $N-1$ echelon buffers into $(N-1)$ 2-machine subsystems that are easier to analyze. Each subsystem has an intermediate finite buffer that represents one of the echelon buffers in the original line. The ultimate goal is to set the parameters of each subsystem so that the behavior of its

intermediate buffer mimics as closely as possible the behavior of the corresponding echelon buffer in the original line.

3.1 Building block description and notation

Figure 5 shows the 3 subsystems that result from the decomposition of the 4-machine production line L shown in Figure 4. These subsystems are denoted by $L^n, n=1,...,N-1$. Subsystem $L^n, n=2,...,N-1$ represents the entire part of the original production line downstream of machine M_{n-1}, while subsystem L^1 represents the entire production line.

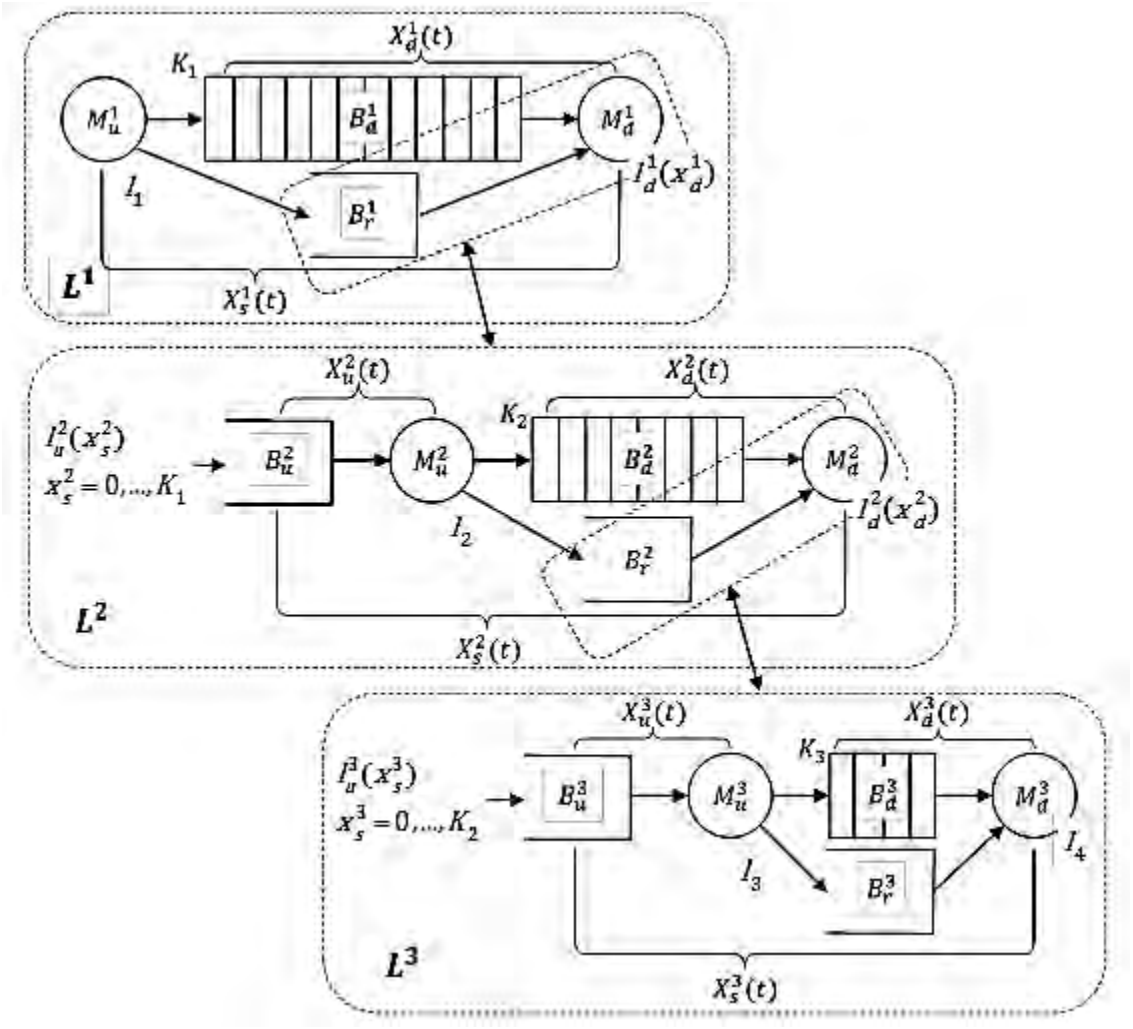

Figure 5. Decomposition of the production line with echelon buffers shown in Figure 4.

Each subsystem L_n has an upstream infinite buffer (except for L_1 which has none), an upstream machine, an intermediate finite buffer, and a downstream machine, denoted by B_u^n, M_u^n, B_d^n, and M_d^n, respectively. It also has another infinite buffer denoted by B_r^n that is parallel to B_d^n. The contents of B_r^n are always equal to the contents of B_d^n because both buffers are filled and depleted simultaneously. Hence, B_d^n could be eliminated from the model without changing its behavior. The number of parts in B_u^n, including the part in M_u^n, and the number of parts in B_d^n, including the part in M_d^n, in period t, are denoted by $X_u^n(t)$ and $X_d^n(t)$, respectively. The total number of parts in the entire subsystem is denoted by $X_s^n(t)$, i.e., $X_s^n(t) = X_u^n(t) + X_d^n(t)$. Machine M_u^n and buffer B_d^n represent machine M_n and echelon buffer B_n in the original N - machine line L, respectively; hence, M_u^n has production rate l_n and buffer B_d^n has capacity K_n including the space in M_d^n.

Machine M_d^n together with buffer B_r^n represents the entire part of the system downstream of M_n in the original line. It is therefore an aggregate representation of subsystem L_{n+1}, for $n = 1,..., N-2$. In other words, L_{n+1} is nested in L_n, $n = 1,..., N-1$. Because M_d^n represents an entire subsystem its behavior should be more complex than that of a simple machine with a single production rate. To capture this complexity, we assume that M_d^n has load-dependent production rates denoted by $l_d^n(x_d^n)$ where x_d^n is the current value of $X_d^n(t)$. In the last subsystem, L^{N-1}, M_d^{N-1} represents the last machine M_N in the original N - machine line; hence, M_d^{N-1} has production rate l_N.

Buffer B_u^n receives parts arriving from outside and represents stage buffer $B_{n-1}^{"}$ in the original line L. The latter buffer receives parts produced by machine M_{n-1}. Given that M_{n-1} may be blocked or starved, the behavior of the arrival process to $B_{n-1}^{"}$ should be more complex than the behavior of the production process of a simple machine with a single production rate. To capture this complexity, we assume that B_u^n receives parts with state-dependent arrival rate $l_u^n(x_s^n)$ where x_s^n is the current value of $X_s^n(t)$, i.e., the total WIP in subsystem L_n. The arrival process at buffer B_u^n has the property that $l_u^n(K_{n-1}) = 0$ because in the original line, if $X_{n-1}(t) = K_{n-1}$, M_{n-1} is blocked and hence the rate of a part arriving to $B_{n-1}^{"}$ becomes zero. This implies that the maximum value that x_s^n can take is K_{n-1}.

To evaluate the performance of the original production line L, we must address the following two problems: (i) How can we analyze each subsystem L^n in isolation given the state-dependent external arrival rates $l_u^n(x_s^n), x_s^n = 0,...,K_{n-1}$, (except for L^1 that has no external arrivals) and the load-dependent production rates of machine M_d^n, $l_d^n(x_d^n), x_d^n = 0,...,K_n$ (except for L^{N-1} where machine M_d^{N-1} has production rate l_N, and (ii) how can we determine the unknown rates $l_u^n(x_s^n)$, $x_s^n = 0,...,K_{n-1}$, $n = 2,...,N-1$ and $l_d^n(x_d^n), x_d^n = 0,...,K_n$, $n = 1,...,N-2$. We address these problems in Sections 3.2 and 3.3, respectively. Once these problems have been solved, the performance measures of the original system L can be obtained from the performance measures of subsystems $L_n, n = 1,...,N-1$.

3.2 Analysis of 2-machine subsystem L^n in isolation

In this section, we describe how to analyze each subsystem $L^n, n = 1,...,N-1$ in isolation. First, we concentrate on subsystems $L^n, n = 2,...,N-1$ that have external arrivals, and then we proceed with the simpler subsystem L^1 that has no external arrivals.

3.2.1 Analysis of subsystem $L^n, n = 2,...,N-1$

Figure 6 shows the queueing network model of subsystem $L^n, n = 2,...,N-1$, for the general case where M_u^n has load-dependent production rates $l_u(x_u^n)$. We consider this generalization to show that we can easily apply our analysis to the case where machine $M_n, n = 2,...,N-1$, in the original line has load dependent production rates, as was mentioned in Section 2.2.

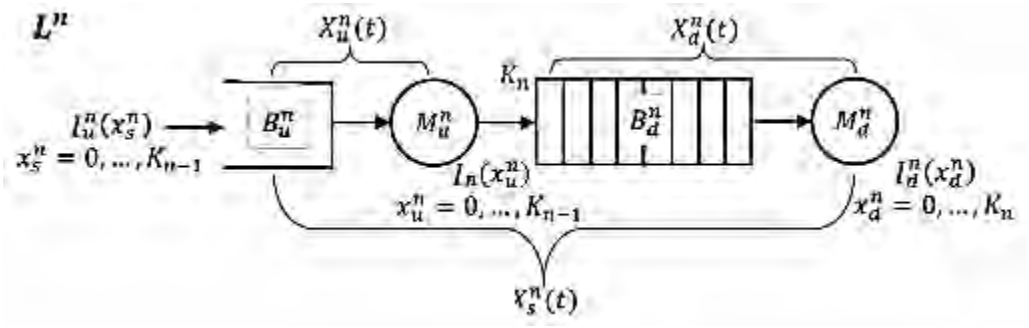

Figure 6.Queueing network model of subsystem $L^n, n = 2,...,N-1$.

If we define the state of each subsystem L^n as the vector of the WIP levels $\mathbf{X}^n(t) = (X_u^n(t), X_d^n(t))$, then $\{\mathbf{X}^n(t), t = 0,1,...\}$ is a 2-dimensional discrete-state, continuous-time Markov chain with transition probabilities that are functions of the load-dependent production rates $l_n(x_u^n), x_u^n = 0,..., K_{n-1}$, the state-dependent arrival rates $l_u^n(x_s^n), x_s^n = 0,..., K_{n-1}$, and the load-dependent production rates $l_d^n(x_d^n), x_d^n = 0,..., K_n$. The number of states of this chain is $(K_{n-1}+1)(K_n+1) - \frac{K_n(K_n+1)}{2}$. Figure 7 shows the state transition diagram of this chain for $K_{n-1} = 7$ and $K_n = 4$, indicating only the inter-state transitions but not the transition probabilities.

To find the stationary probabilities of $\{\mathbf{X}^n(t), t \geq 0\}$, denoted by $P^n(x_u^n, x_d^n)$, we must write the balance equations and the normalization equation and solve them. In what follows, we give the expressions for these equations, where, for notational simplicity, we dropped subscript/superscript n from rates $l_n(\bullet), l_u^n(\bullet), l_d^n(\bullet)$, and probabilities $P^n(\bullet,\bullet)$. We used i and j to denote states x_u^n and x_d^n, respectively. The form of the balance equations differs depending on whether the states of the Markov chain are in the middle, on the boundaries, or at the corners of the state transition diagram, as is indicated in Figure 7.

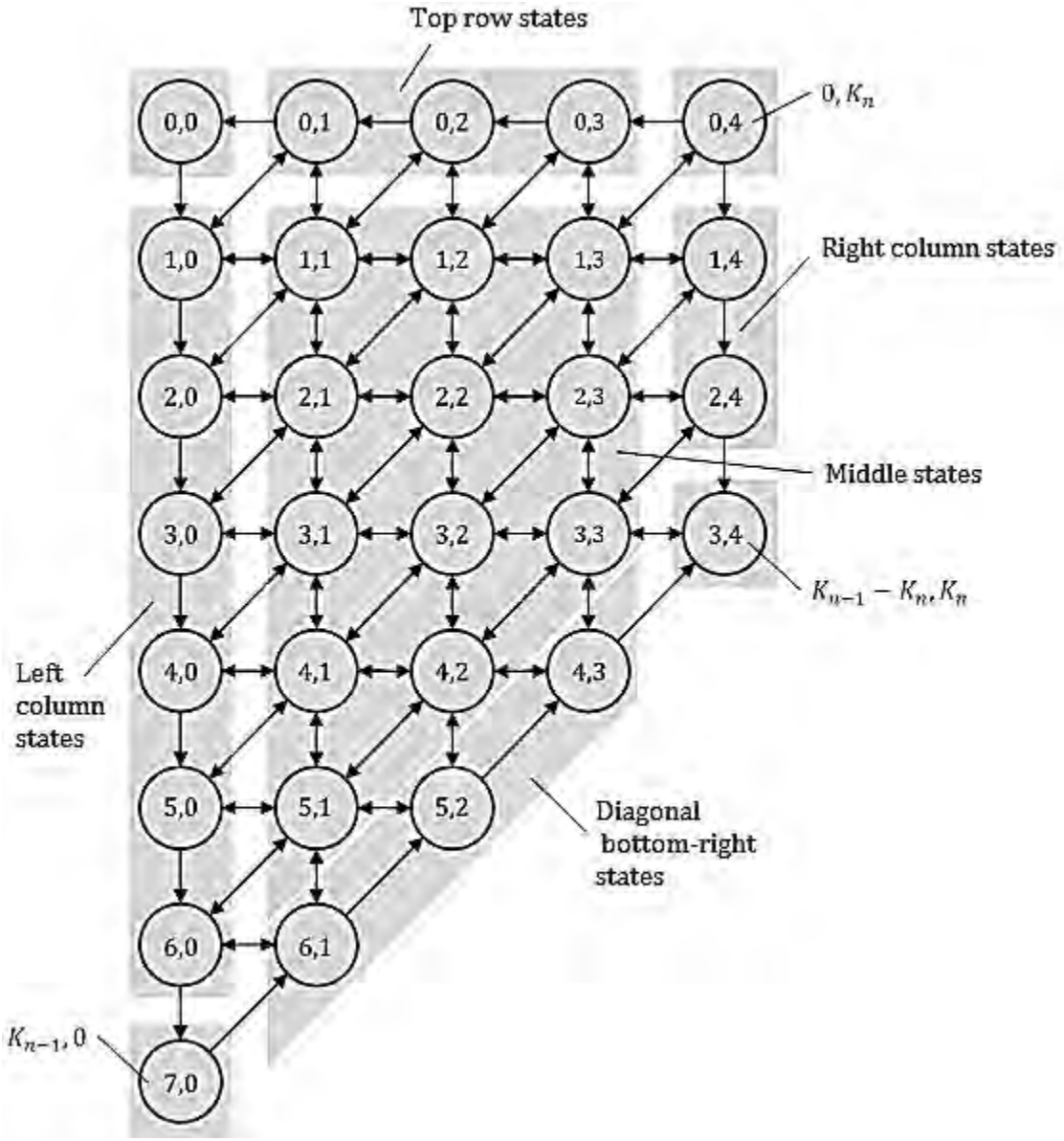

Figure 7. State transition diagram of $\mathbf{X}^n(t), n = 2,...,N-1$ for $K_{n-1} = 7$ and $K_n = 4$.

Balance equations

Middle States (i, j): For $j = 1,...,K_n - 1, i = 1,...,K_{n-1} - j - 1,$

$$P(i,j)[l_u(i+j) + l(i) + l_d(j)]$$
$$= P(i-1,j)l_u(i-1+j) + P(i+1,j-1)l(i+1) + P(i,j+1)l_d(j+1)$$

Top left corner state (0,0): $P(0,0)l_u(0) = P(0,1)l_d(1)$

Bottom left corner state (K_{n-1}, 0): $P(K_{n-1},0)l(K_{n-1}) = P(K_{n-1}-1,0)l_u(K_{n-1}-1)$

Top right corner state $(0, K_n)$: $P(0, K_n)[l_u(K_n) + l_d(K_n)] = P(1, K_n - 1)l(1)$

Bottom right corner state $(K_{n-1} - K_n, K_n)$: If $K_{n-1} > K_n$,

$$P(K_{n-1} - K_n, K_n)l_d(K_n)$$
$$= P(K_{n-1} - K_n - 1, K_n)l_u(K_{n-1} - 1) + P(K_{n-1} - K_n + 1, K_n - 1)l(K_{n-1} - K_n + 1)$$

Left column states $(i, 0)$: For $i = 1, ..., K_{n-1} - 1$

$$P(i, 0)[l_u(i) + l(i)] = P(i - 1, 0)l_u(i - 1) + P(i, 1)l_d(1)$$

Top row states $(0, j)$: For $j = 1, ..., K_n - 1$,

$$P(0, j)[l_u(j) + l_d(j)]$$
$$= P(1, j - 1)l(1) + P(0, j + 1)l_d(j + 1)$$

Right column states (i, K_n): For $i = 1, ..., K_{n-1} - K_n - 1$,

$$P(i, K_n)[l_u(i + K_n) + l_d(K_n)]$$
$$= P(i - 1, K_n)l_u(i - 1 + K_n) + P(i + 1, K_n - 1)l(i + 1)$$

Diagonal bottom-right states $(K_{n-1} - j, j)$: For $j = 1, ..., K_n - 1$,

$$P(K_{n-1} - j, j)[l(K_{n-1} - j) + l_d(j)] =$$
$$P(K_{n-1} - j - 1, j)l_u(K_{n-1} - 1) + P(K_{n-1} - j + 1, j - 1)l(K_{n-1} - j + 1)$$

Normalization equation

$$\sum_{j=0}^{K_n} \sum_{i=0}^{K_{n-1}-j} P(i, j) = 1.$$

Note that the transition rates at the extreme states are $l_u(K_{n-1})\Delta t = l(0)\Delta t = l_d(0)\Delta t = 0$; therefore, $1 - l_u(K_{n-1})\Delta t = 1 - l(0)\Delta t = 1 - l_d(0)\Delta t = 1$. We also let $\Delta t \to 0$ and eliminated terms of $O(\Delta t^n)$. These facts led to a simplified form of balance equations, as the total number of expressions required to describe them has been reduced. For example, the initial form of middle states balance equation is stated below:

$$
\begin{aligned}
&P(i,j)\{1-[l_u(i+j)\Delta t l(i)\Delta t l_d(j)\Delta t+(1-l_u(i+j)\Delta t)(1-l(i)\Delta t)(1-l_d(j)\Delta t)]\}\\
&=P(i-1,j)l_u(i-1+j)\Delta t(1-l(i-1)\Delta t)(1-l_d(j)\Delta t)\\
&+P(i-1,j+1)l_u(i+j)\Delta t(1-l(i-1)\Delta t)l_d(j+1)\Delta t\\
&+P(i+1,j)(1-l_u(i+j+1)\Delta t)l(i+1)\Delta t l_d(j)\Delta t\\
&+P(i,j-1)(1-l_u(i+j-1)\Delta t)l(i)\Delta t(1-l_d(j-1)\Delta t)\\
&+P(i+1,j-1)(1-l_u(i+j)\Delta t)(1-l_d(j-1)\Delta t)l(i+1)\Delta t\\
&+P(i,j+1)(1-l_u(i+j+1)\Delta t)(1-l_d(j+1)\Delta t)l_d(j+1)\Delta t
\end{aligned}
$$

The above system of equations is linear and has a unique solution. It can be solved using any numerical analysis scheme. In our numerical examples, we use the Gauss-Seidel method, where in each iteration we sequentially update the stationary probability of each state using the most recent values of the stationary probabilities of the other states involved. At the end of each iteration, we normalize all probabilities. We terminate the iterations when the maximum absolute percentage difference between two successive iterations is below a very small number ε. Once we have computed the stationary probabilities, we can use them to calculate the following performance measures of interest:

$l_{out}^n(x_d^n), x_d^n = 0,...,K_n$: internal state-dependent arrival rate of parts to buffer B_d^n.

$TH^n(x_s^n), x_s^n = 0,...,K_{n-1}$: conditional throughput of subsystem L^n

$\bar{X}_d^n$: average WIP level of buffer B_d^n.

FR^n: overflow rate of buffer B_u^n defined as the rate that X_u^n increases by one unit when $X_u^n \geq K_{n-1} - K_n$. FR^n approximates the rate that $X_{n-1}^{''}$ increases by one unit when $X_{n-1}^{''} \geq K_{n-1}^{'}$ in the original system L, and represents the rate at which parts produced by machine M_{n-1} are physically stored in an installation buffer downstream of $B_{n-1}^{'}$ because $B_{n-1}^{'}$ is full (hence, the term "overflow").

Note that in the above definitions, we have restored the original notation, namely, subscript/superscript n for the subsystem index, and x_u^n, x_d^n and x_s^n for the WIP levels i, j and $i+j$, respectively. The above performance measures are calculated as follows:

$$l_{out}^n(x_d^n)=\begin{cases}\dfrac{\sum_{x_u^n=0}^{K_{n-1}-x_d^n} P^n(x_u^n,x_d^n)l_n(x_u^n)}{\sum_{x_u^n=0}^{K_{n-1}-x_d^n} P^n(x_u^n,x_d^n)}, x_d^n=0,...,K_n-1,\\ 0, x_d^n=K_n\end{cases} \quad (7)$$

$$TH^n(x_s^n)=\frac{\sum_{x_u^n=(x_s^n-K_n)^+}^{x_s^n} P^n(x_u^n,x_s^n-x_u^n)l_d^n(x_s^n-x_u^n)}{\sum_{x_u^n=(x_s^n-K_n)^+}^{x_s^n} P^n(x_u^n,x_s^n-x_u^n)}, x_s^n=0,...,K_{n-1}, \quad (8)$$

$$\bar{X}_d^n=\sum_{x_d^n=0}^{K_n} x_d^n \sum_{x_u^n=0}^{K_{n-1}-x_d^n} P^n(x_u^n,x_d^n), \quad (9)$$

$$FR^n=\sum_{x_d^n=0}^{K_n}\sum_{x_u^n=(K_{n-1}-K_n)}^{K_{n-1}-x_d^n} P^n(x_u^n,x_d^n)l_u^n(x_u^n+x_d^n). \quad (10)$$

3.2.2 Analysis of subsystem L^1

The first subsystem of the decomposition L^1, shown at the top of Figure 5, differs from the other subsystems in that there is no input process to machine M_u^1; hence, it is simpler. Because M_u^1 represents machine M_1 in the original line L, it is never starved and continuously produces a part with rate l_1 unless it is blocked by buffer B_u^n when this buffer is full. If we define the state of L^1 as the WIP level $X_d^1(t)$ then clearly $\left\{X_d^1(t), t\geq 0\right\}$ is a continuous-time finite-state birth-death process, for which the stationary probabilities, denoted by $P^1(x_d^1)$ can be easily computed. The state transition diagram (excluding the self-transitions) of $X_d^1(t)$ is shown in Figure 8(a) or after elimination of $\Delta t\to 0$ and terms of $O(\Delta t^2)$ in 8(b). As previously, for notational simplicity, we dropped subscript/superscript "1" from rates $l_1, l_d^1(\bullet)$ and probabilities $P^1(x_d^1)$ and we used j to denote state x_d^1.

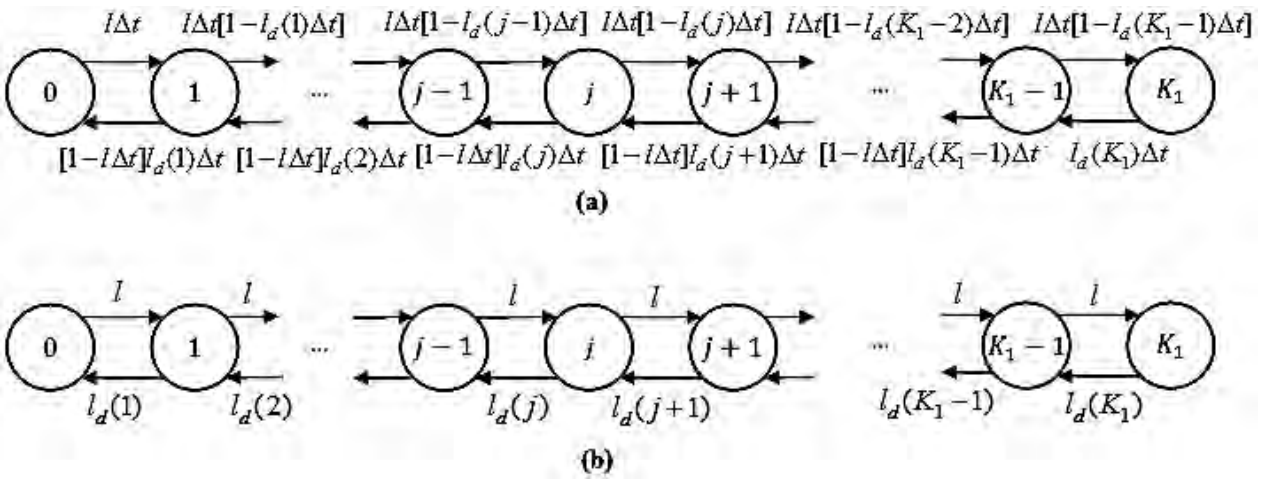

Figure 8.State transition diagram of $X_d^1(t)$.

To compute the stationary probabilities of $\left\{X_d^1(t), t \geq 0\right\}$ we define coefficients $C(j)$ which satisfy the following expression:

$$C(j) = \begin{cases} 1, \ j = 0 \\ \dfrac{l}{l_d(j)} C(j-1), \ j = 1,...,K_1. \end{cases}$$

The stationary probabilities are then given by:

$$P(j) = \frac{C(j)}{\sum_{i=0}^{K_1} C(i)}, \ j = 0,...,K_1.$$

Once we have computed the stationary probabilities, we can use them to calculate the average throughput of subsystem 1, denoted by TH^1, and the average WIP level in buffer 1, denoted by $\bar{X}_d^1$, where we have restored the original notation, namely, subscript/superscript "1" for the subsystem index and x_d^1 for the WIP level j. These two measures are calculated as follows:

$$TH^1 = l_1(1 - P^1(K_1)), \quad (11)$$

$$\bar{X}_d^1 = \sum_{x_d^1=0}^{K_1} x_d^1 P^1(x_d^1). \quad (12)$$

Finally, note that the internal state-dependent arrival probability of parts to buffer B_d^1, denoted by $l_{out}^1(x_d^1), x_d^1 = 0,...,K_1$, is simply given by

$$l_{out}^{1}(x_d^1)=\begin{cases} l_1, x_d^1=0,...,K_1-1, \\ 0, x_d^1=K_1. \end{cases}$$

3.3 Analysis of the original production line

The unknown parameters of each subsystem L_n are the state-dependent external arrival rates $l_u^n(x_s^n), x_s^n=0,...,K_{n-1}$ (except in L^1, where there are no external arrivals), and the load-dependent production rates $l_d^n(x_d^n), x_d^n=0,...,K_n$ of machine M_d (except in L^{n-1}, where M_d^{N-1} is identical to the last machine, M_N and therefore has known production rate l_N). To determine the values of these parameters we set up a system of equations that relate the flow of parts in subsystem L^n with the flow of parts in the neighboring subsystems L^{n+1} and L^{n-1}.More specifically, as we wrote earlier, M_d^n in subsystem $L_n, n=1,...,N-2$, is an aggregate representation of subsystem L^{n+1}. The load-dependent production rates of $M_d^n, l_d^n(x_d^n), x_d^n=0,...,K_n$, should therefore be equal to the conditional throughput of L^{n+1}. Just as M_d^n in L_n is an aggregate representation of $L^{n+1}, n=1,...,N-2$, so L_n is aggregately represented by machine as M_d^{n-1} in $L^{n-1}, n=1,...,N-2$. The external arrival process to buffer B_u^n in $L^n, l_u^n(x_s^n), x_s^n=0,...,K_{n-1}$, should therefore be equal to the internal state-dependent arrival process of parts from machine M_u^{n-1} to buffer B_d^{n-1} (as well as to the redundant buffer B_r^{n-1}) in L^{n-1}.

The above relationships can be written as follows:

$$l_d^n(x)=TH^{n+1}(x), x=0,...,K_n, n=1,...,N-2 \qquad (13)$$

$$l_u^n(x)=l_{out}^{n-1}(x), x=0,...,K_{n-1}, n=2,...,N-1 \qquad (14)$$

For each subsystem L^n, the conditional throughput $TH^n(x)$ and the internal state-dependent arrival rate $l_{out}^n(x)$ can be computed by analyzing the subsystem in isolation, given the values of the production rates $l_u^n(x), x=0,...,K_{n-1}$, and $l_d^n(x), x=0,...,K_n$, as was shown in Section 3.2. This means that $TH^{n+1}(x)$ in (13) is a function of $l_u^{n+1}(\bullet)$ and $l_d^{n+1}(\bullet)$ and $l_{out}^{n-1}(x)$ in (14) is a function of $l_u^{n-1}(\bullet)$ and $l_d^{n-1}(\bullet)$. Hence, the unknown parameters $l_u^n(x)$ and $l_d^n(x)$ in expressions (13) and (14) are the solution of a fixed-point problem. To determine their values, we use the following iterative algorithm.

Algorithm for analyzing the entire production line

Step 1 *Initialization.*

1.1. Set the unknown external arrival rates of each subsystem (except L^1 which receives no external arrivals) to some initial value. A reasonable initial value that we have used in our numerical experiments is the smallest production rate of all machines upstream of M_n, namely,

$$l_u^{n,init}(x) = \begin{cases} \min(l_m : m = 1,...,n-1), & x = 0,...,K_{n-1}-1, \\ 0, & x = K_{n-1}, \end{cases} \quad n = 2,...,N-1. \qquad (15)$$

1.2. Set the unknown production rates of machine M_d^n in each subsystem to some initial value. A reasonable initial value that we have used in our experiments is the smallest production probability of all machines downstream of M_n, namely,

$$l_d^{n,init}(x) = \begin{cases} 0, & x = 0, \\ \min(l_m : m = n+1,...,N), & x = 1,...,K_{n-1}, \end{cases} \quad n = 1,...,N-2. \qquad (16)$$

Step 2.*Main Iteration.*

Iterate backwards and forwards until the external and internal arrival rates converge, i.e., until $l_u^n(x) = l_{out}^{n-1}(x), x = 0,...,K_{n-1}, n = 3,...,N-1$. More specifically,

Set $n = N-1$.

While $n \geq 2$,

If $n = N-1$,

Given $l_u^{N-1}(x), x = 0,...,K_{N-2}$, **solve** subsystem L^{N-1} and **compute**

$TH^{N-1}(x), x = 1,...,K_{N-2}$, $\bar{X}_d^{N-1}$, FR^{N-1}, from (8)-(10), respectively, for $n = N-1$.

Set $l_d^{N-2}(x) = r_{out}^{N-1}(x), x = 1,...,K_{N-2}-1$; **set** $n = N-2$.

Else

Given $l_u^n(x), x = 0,...,K_{n-1}$ and $l_d^n(x), x = 0,...,K_n$ **solve** subsystem L^n and **compute**

$l_{out}^n(x), x = 0,...,K_n-1$, $TH^n(x), x = 1,...,K_{n-1}$, $\bar{X}_d^n$, FR^n, from (7)-(10), respectively.

If $l_{out}^n(x) \approx l_u^{n+1}(x), x = 0,...,K_n-1$,

Set $l_d^{n-1}(x) = TH^n(x), x = 1,...,K_n-1$; **set** $n = n-1$.

Else

Set $l_u^{n+1}(x) = l_{out}^n(x), x = 0,...,K_n-1$; **set** $n = n+1$.

Endif

Endif

Endwhile

Step 3.*Compute average system throughput and WIP.*

Given $l_d^1(x), x = 0,...,K_1$, solve subsystem L^1 and compute average throughput TH^1 from (11) and average

WIP level $\bar{X}_d^1$ from (12). These two values are the final estimates of the average throughput and total WIP of the system. Similarly, the most recent values of $\bar{X}_d^n, n = 2,...,N-1$, are the final estimates of

the average echelon WIP downstream of machine $M_n, n = 2,..., N-1$. Finally, the most recent values of $FR^n, n = 2,..., N-1$, are the final estimates of the overflow rates of $B'_{n-1}, n = 2,..., N-1$.

Note that the first time each subsystem $L^n, n = 2,..., N-1$, is solved using the method presented in Section 3.2, the stationary probabilities of Markov chain $\mathbf{X}^n(t)$ must be initialized. The simplest way to do this is to set them all equal and such that their sum is one. A more intelligent way is to set $P^n(x_u^n, x_d^n)$ equal to the normalized product of the approximate marginal stationary distributions of $X_u^n(t)$ and $X_d^n(t)$ in isolation. The approximate marginal distribution of $X_u^n(t)$ in isolation can be found by solving a 2-machine 1-buffer line (as a continuous-time finite-state birth-death process), where the upstream and downstream machines have production rates $l_{u,1}^{n,init}(x)$ and $l_{d,1}^{n-1,init}(x)$ given by (15) and (16), respectively. Similarly, the approximate marginal distribution of $X_d^n(t)$ in isolation can be found by solving a 2-machine 1-buffer line, where the upstream and machines have production probabilities $l_{u,1}^{n+1,init}(x)$ and $l_{d,1}^{n,init}(x)$ given by (15) and (16), respectively. From then on, each time subsystem $L^n, n = 2,..., N-1$, is solved again, the stationary probabilities from the previous time are used as initial values. Numerical experimentation has shown that this method results in significant gains in overall computational time. Finally, the criterion that we used to detect if $l_{out}^n(x) \approx l_u^{n+1}(x), x = 0,..., K_n - 1$ in step 2 of the above procedure is $\max_{x=0,...,K_n-1}\left\{\left|\frac{l_{out}^n(x) - l_u^{n+1}(x)}{l_u^{n+1}(x)}\right|\right\} < \varepsilon$ where ε is a very small number. In the following chapter we report on numerical experimentation with the decomposition method.

Chapter 4 - Numerical results on the performance of the decomposition method and the effect of system parameters on system performance

In this chapter, we evaluate the accuracy and efficiency of the decomposition method developed in Sections 3.2 and 3.3 by comparing it against simulation, for several instances of two numerical examples, also exploring the effect of system parameters on system performance. In all instances, we used the value of $\varepsilon = 0.0001$(convergence criterion) both in the procedure for analyzing each subsystem L^n in isolation, described in Section 3.2, and in the algorithm for analyzing the original system L, described in Section 3.3. To obtain the simulation results, for each instance, we executed 30 independent event-driven simulation runs of the production line over a maximum production of 200.000 parts. For each performance measure estimate that we compute, we report the sample mean and a 95% confidence interval over the 30 runs. Both the decomposition and simulation algorithms were written in Matlab R2011a and were run on a laptop with a Pentium® Dual-Core CPU @ 2.1 GHz.

4.1 Example 1: 5-machine line

In Example 1, we consider a production line consisting of $N = 5$ machines and 4 buffers. For this system, we evaluated 9 different instances (cases). Table 1 shows the input data for each case, namely, the production rates of the machines, $l_n, n = 1,\ldots,5$, the capacities of echelon buffers, $K_n, n = 1,\ldots,4$, and the resulting capacities of the installation buffers, $K_n^{'}, n = 1,\ldots,4$, computed from (3)-(4).

Table 1. Input data for Example 1.

#	l_1	l_2	l_3	l_4	l_5	K_1	K_2	K_3	K_4	K_1'	K_2'	K_3'	K_4'
1	6	6	6	6	6	20	15	10	5	5	5	5	5
2	6	6	6	6	6	40	30	20	10	10	10	10	10
3	6	6	6	6	6	60	45	30	15	15	15	15	15
4	6	6	4	6	6	20	15	10	5	5	5	5	5
5	6	6	4	6	6	40	30	20	10	10	10	10	10
6	6	6	6	6	6	40	40	40	40	0	0	0	40
7	6	6	4	6	6	40	40	40	40	0	0	0	40
8	8	8	8	8	8	40	30	20	10	10	10	10	10
9	9	9	6	9	9	20	15	10	5	5	5	5	5

Cases 1-3, 6, and 8 represent balanced lines where all machines have the same production rate. In all these cases, except case 8, the rate is 6; in case 8, it is 8. The difference between cases 1-3 is that the installation buffer capacities are 5, 10, and 15 units, respectively. The buffer capacities in case 8 are the same as those in case 2. In cases 4, 5, 7 and 9 all machines have the same production rate except M_3,

which has a smaller rate, representing a bottleneck machine. The buffer capacities in cases 4 and 5 are the same as those in cases 1 and 2, respectively. The buffer capacities in case 9 are the same as those is case 4. Finally, in cases 6 and 7, the capacities of all the installation buffers except the last one are zero; as a result, all echelon buffer capacities are the same. As was mentioned in Section 2.2, this corresponds to the case of a line operating under CONWIP. Tables 2 and 3 show the performance measure estimates obtained by decomposition and simulation, respectively. These measures are the average echelon WIP levels, denoted by $\bar{X}_n, n=1,\ldots,4$, the average line throughput, denoted by TH , the average overflow rate of buffer $B_n^{"}$, denoted by $FR_n, n=1,\ldots,3$ and the computation time, CPU , in seconds. For the simulation estimates, 95% confidence intervals are also shown. For the decomposition method, recall that the values of $\bar{X}_n$, TH , and FR_n are computed as the final values of $\bar{X}_d^n$, TH^1, and FR^{n+1} once the algorithm described in Section 3.3 converges. Note that FR_4 and more generally FR_{N-1} is zero because there is no overflow of parts for the last buffer $B_{N-1}^{"}$. Finally, Table 4 shows the percent difference between the decomposition and the simulation estimates, i.e. $\frac{estimate^{dec} - estimate^{sim}}{estimate^{sim}} \times 100$.

Table 2. Performance measure estimates for Example 1 obtained by decomposition.

#	$\bar{X}_1$	$\bar{X}_2$	$\bar{X}_3$	$\bar{X}_4$	TH	FR_1	FR_2	FR_3	$CPU(s)$
1	16,7188	11,2891	6,4386	2,2189	4,7405	2,5058	2,1836	1,8320	0,3525
2	33,3391	22,5563	12,9010	4,4989	5,3032	2,7737	2,4179	2,0230	2,8234
3	49,9530	33,8299	19,3720	6,7835	5,5186	2,8761	2,5079	2,0991	10,9487
4	18,2068	13,1128	3,5388	1,5062	3,8900	1,9693	3,4888	0,4649	0,3102
5	38,0069	27,9895	3,9838	1,9241	3,9970	1,9999	3,9705	0,0721	1,3662
6	31,9964	23,9965	16,0014	8,0058	5,4549	5,4544	5,4552	5,4544	6,3684
7	38,0003	36,0006	3,9994	1,9997	3,9999	3,9999	3,9999	3,9999	3,0302
8	33,3391	22,5563	12,9010	4,4989	7,0710	3,6983	3,2238	2,6973	2,8591
9	18,2068	13,1128	3,5388	1,5062	5,8350	2,9540	5,2332	0,6974	0,2852

Table 3. Performance measure estimates for Example 1 obtained by simulation.

#	$\bar{X}_1$	$\bar{X}_2$	$\bar{X}_3$	$\bar{X}_4$	TH	FR_1	FR_2	FR_3	$CPU(s)$
1	16,8639 ±0,0127	11,4144 ±0,0119	6,4878 ±0,0089	2,2136 ±0,0045	4,7546 ±0,0023	2,5297 ±0,0062	2,2393 ±0,0075	1,8769 ±0,0059	1459,6482
2	33,6387 ±0,0397	22,7848 ±0,0416	12,9726 ±0,0240	4,5045 ±0,0128	5,3102 ±0,0019	2,8030 ±0,0125	2,4812 ±0,0133	2,0511 ±0,0113	1422,5819
3	50,4385 ±0,1019	34,2058 ±0,0901	19,4570 ±0,0772	6,7871 ±0,0261	5,5226 ±0,0025	2,8999 ±0,0217	2,5859 ±0,0192	2,1188 ±0,0197	1410,9796
4	18,2466 ±0,0040	13,1445 ±0,0053	3,5674 ±0,0070	1,4992 ±0,0028	3,8904 ±0,0021	1,9777 ±0,0039	3,5050 ±0,0032	0,4622 ±0,0031	1334,7884
5	37,9977 ±0,0093	27,9782 ±0,0083	4,0273 ±0,0109	1,9232 ±0,0052	3,9971 ±0,0031	2,0012 ±0,0041	3,9700 ±0,0031	0,0739 ±0,0018	1350,6668

6	32,0191 ±0,0423	23,9655 ±0,0672	15,9346 ±0,0640	7,9794 ±0,0608	5,4541 ±0,0028	5,4549 ±0,0028	5,4547 ±0,0027	5,4545 ±0,0027	1361,0225
7	37,9942 ±0,0078	35,9859 ±0,0104	4,0317 ±0,0140	2,0008 ±0,0074	3,9995 ±0,0031	4,0003 ±0,0030	4,0002 ±0,0030	3,9996 ±0,0030	1351,3238
8	33,6387 ±0,0397	22,7848 ±0,0416	12,9726 ±0,0240	4,5045 ±0,0128	7,0803 ±0,0025	3,7374 ±0,0167	3,3083 ±0,0178	2,7348 ±0,0151	1438,8483
9	18,2466 ±0,0040	13,1445 ±0,0053	3,5674 ±0,0070	1,4992 ±0,0028	5,8356 ±0,0032	2,9666 ±0,0059	5,2575 ±0,0048	0,6933 ±0,0047	1376,0249

Table 4. Percent difference in performance measure estimates obtained by decomposition and simulation for Example 1.

#	$\bar{X}_1$	$\bar{X}_2$	$\bar{X}_3$	$\bar{X}_4$	FR_1	FR_2	FR_3	TH
1	-0,8604	-1,0977	-0,7583	0,2394	-0,9448	-2,4874	-2,3922	-0,2966
2	-0,8906	-1,0029	-0,5519	-0,1243	-1,0453	-2,5512	-1,3700	-0,1318
3	-0,9626	-1,0989	-0,4369	-0,0530	-0,8207	-3,0164	-0,9298	-0,0724
4	-0,2181	-0,2412	-0,8017	0,4669	-0,4247	-0,4622	0,5842	-0,0103
5	0,0242	0,0404	-1,0801	0,0468	-0,0650	0,0126	-2,4357	-0,0025
6	-0,0709	0,1294	0,4192	0,3309	-0,0092	0,0092	-0,0018	0,0147
7	0,0161	0,0408	-0,8012	-0,0550	-0,0100	-0,0075	0,0075	0,0100
8	-0,8906	-1,0029	-0,5519	-0,1243	-1,0462	-2,5542	-1,3712	-0,1314
9	-0,2181	-0,2412	-0,8017	0,4669	-0,4247	-0,4622	0,5914	-0,0103

4.1.1 Confidence in simulation results

The confidence intervals were calculated according to the formula $CI = \pm \frac{1,96 \cdot std(M)}{\sqrt{30}}$, where 1,96 is the critical value of normal distribution for confidence level 0,95 and $std(M)$ is the standard deviation of each estimate array of values that emerge during our 30 simulation runs. From the results in Table 3, we observe that in all cases, the confidence intervals of the throughput estimates obtained by simulation are very tight and are below 0,1% of these estimates. The confidence intervals of the average echelon WIP level estimates are looser but still remain below 0,8% of these estimates. Finally, the confidence intervals for the overflow rates remain well below 2,6% of these estimates.

4.1.2 Accuracy and computational efficiency of the decomposition method

From the results in Table 4, we make the following observations regarding the accuracy of the decomposition method with respect to simulation:

a. In all cases, the accuracy of the decomposition method is very high. More specifically, the absolute percent difference in the throughput estimate, average echelon WIP levels, and overflow rates does not exceed 0,3%, 1,1%, and 3,02%, respectively.
b. The accuracy of the decomposition method in estimating the average throughput is increasing in the echelon buffer capacities (compare cases 1-3). Most likely this happens because when the echelon

buffer capacities increase, the buffer-full and buffer-empty probabilities decrease. As a result, the decoupling effect of the buffers increases, improving the accuracy of the method.

c. The accuracy of the decomposition method is much higher for the lines with a bottleneck machine than for the balanced lines (compare cases 1, 2 vs. cases 4, 5, 9). Having a bottleneck machine in the line effectively separates the line into two segments, one upstream and the other downstream of the bottleneck machine. The bottleneck machine is almost never starved and hence almost always feeds the downstream segment independently of what is going on in the upstream segment. This decoupling effect again helps increase the accuracy of the decomposition method.

By comparing the last column of Tables 2 and 3, we make the following observations regarding the computational efficiency of the decomposition method compared to that of simulation:

a. The computational time using decomposition is 2 or even 3 orders of magnitude smaller than the corresponding time using simulation.

b. The computational time using decomposition is increasing in the echelon buffer capacities. This is because the larger the capacities, the larger the number of states of the Markov chain of the 2-machine subsystems L_n that need to be solved.

c. The computational time using decomposition is smaller for the lines with a bottleneck machine than it is for the balanced lines (compare cases 1, 2 and 4, 5, 9). Most likely this happens because of the decoupling effect discussed earlier.

4.1.3 Effect of system parameters on system performance

Finally, by comparing the performance measures between the different cases in Table 2 (and Table 3) we make the following observations regarding the effect of system parameters on system performance:

a. The average echelon WIP levels and line throughput are increasing in the echelon buffer capacities (compare cases 1-3). As the echelon buffer capacities increase, the average line throughput approaches the production rate of the slowest machine. This is also true when one compares cases 2 and 5 against 6 and 7.

b. The average echelon WIP level estimates are identical for cases 2, 8 and cases 4, 9. This possibly happens because in a system with 2 machines and 1 intermediate buffer, the average number of parts in the buffer is determined by $\frac{l_1}{l_2}$ ratio, where l_1 and l_2 are the processing rates of the 2 machines. In the above-mentioned cases these ratios are the same.

c. The overflow rate is decreasing in the echelon buffer capacities.

d. The line throughput and overflow rates are increasing in the production rate of the machines (compare cases 2, 8 and 4, 9).

e. Having a bottleneck machine in the line results in increasing the average echelon WIP levels upstream of the bottleneck and decreasing them downstream of the bottleneck (compare cases 1-2 vs. cases 4-5).

The above observations on the effect of the system parameters on system performance are expected.

4.2 Example 2: 10-machine line

In Example 2, we consider a production line consisting of $N=10$ machines and 9 buffers. For this system, we evaluated 6 different instances. Table 5 shows the input data for each instance. The rationale behind the choice of parameter values for the different instances is similar to that in Example 1 explained in the previous section.

Table 5. Input data for Example 2.

#	l_1	l_2	l_3	l_4	l_5	l_6	l_7	l_8	l_9	l_{10}	K_1	K_2	K_3	K_4	K_5	K_6	K_7	K_8	K_9	K'_1	K'_2	K'_3	K'_4	K'_5	K'_6	K'_7	K'_8	K'_9
1	6	6	6	6	6	6	6	6	6	6	45	40	35	30	25	20	15	10	5	5	5	5	5	5	5	5	5	5
2	6	6	6	6	6	6	6	6	6	6	90	80	70	60	50	40	30	20	10	10	10	10	10	10	10	10	10	10
3	6	6	6	6	6	4	6	6	6	6	45	40	35	30	25	20	15	10	5	5	5	5	5	5	5	5	5	5
4	6	6	6	6	6	4	6	6	6	6	90	80	70	60	50	40	30	20	10	10	10	10	10	10	10	10	10	10
5	6	6	6	6	6	6	6	6	6	6	45	45	45	45	45	45	45	45	45	0	0	0	0	0	0	0	0	45
6	8	8	8	8	8	8	8	8	8	8	45	40	35	30	25	20	15	10	5	5	5	5	5	5	5	5	5	5

Tables 6-8 are similar to Tables 2-4 and show the performance measure estimates obtained by simulation and decomposition, and the percent difference between these estimates. Briefly, cases 1, 2, 5, and 6 represent balanced lines, whereas cases 3 and 4 represent lines with a bottleneck machine. Also, all cases, except 5, represent lines where the echelon buffer capacities are incremented uniformly from the end to the beginning of the line. In case 5, all echelon buffer capacities are the same, implying that the capacities of all installation buffers except the last one are zero. This is equivalent to a CONWIP system.

Table 6. Performance measure estimates for Example 2 obtained by decomposition.

#	$\bar{X}_1$	$\bar{X}_2$	$\bar{X}_3$	$\bar{X}_4$	$\bar{X}_5$	$\bar{X}_6$	$\bar{X}_7$	$\bar{X}_8$	$\bar{X}_9$	TH
1	41,5466	35,8054	30,4288	25,2522	20,2291	15,3534	10,6485	6,1828	2,1676	4,6858
2	82,9800	71,6002	60,8894	50,5587	40,5272	30,7833	21,3735	12,4371	4,4093	5,2717
3	43,0632	37,8879	32,8357	27,8237	22,8342	7,9553	5,9373	3,8061	1,5674	3,9602
4	88,0001	77,9787	67,9772	57,9771	47,9772	8,0203	6,0185	4,0072	1,9300	3,9999

5	40,4977	35,9962	31,4965	26,9972	22,4990	18,0009	13,5030	8,9994	4,4984	5,0005
6	41,5466	35,8054	30,4288	25,2522	20,2291	15,3534	10,6485	6,1828	2,1676	6,2477

FR_1	FR_2	FR_3	FR_4	FR_5	FR_6	FR_7	FR_8	$CPU(s)$
2,5689	2,3549	2,2491	2,1751	2,1092	2,0352	1,9276	1,6861	10,0803
2,8545	2,6165	2,4984	2,4159	2,3428	2,2614	2,1428	1,8701	94,2226
2,0347	1,9489	1,9236	1,9118	3,8152	0,5135	0,5632	0,5980	6,0017
2,0026	1,9916	1,9909	1,9908	3,9994	0,0694	0,0707	0,0779	37,3216
5,0004	5,0004	5,0002	5,0000	4,9998	4,9996	4,9995	4,9995	49,2823
3,4253	3,1399	2,9988	2,9002	2,8122	2,7136	2,5701	2,2481	10,0212

Table 7. Performance measure estimates for Example 2 obtained by simulation.

#	$\bar{X}_1$	$\bar{X}_2$	$\bar{X}_3$	$\bar{X}_4$	$\bar{X}_5$	$\bar{X}_6$	$\bar{X}_7$	$\bar{X}_8$	$\bar{X}_9$	TH
1	41,8922	36,3079	31,0166	25,8617	20,7973	15,8128	10,9596	6,3123	2,1871	4,7155
	±0,0157	±0,0184	±0,0166	±0,0181	±0,0190	±0,0181	±0,0149	±0,0100	±0,0033	±0,0021
2	83,6240	72,5053	61,9496	51,6415	41,4645	31,5728	21,9008	12,6426	4,4362	5,2882
	±0,0665	±0,0730	±0,0733	±0,0777	±0,0694	±0,0637	±0,0548	±0,0346	±0,0091	±0,0024
3	43,1340	37,9898	32,9496	27,9364	22,9323	7,9920	5,9575	3,8169	1,5678	3,9627
	±0,0063	±0,0084	±0,0091	±0,0071	±0,0077	±0,0205	±0,0152	±0,0095	±0,0035	±0,0031
4	87,9670	77,9418	67,9460	57,9432	47,9475	8,0279	6,0087	4,0000	1,9288	4,0007
	±0,0080	±0,0102	±0,0084	±0,0073	±0,0081	±0,0245	±0,0199	±0,0150	±0,0072	±0,0032
5	40,4941	35,9942	31,4900	26,9996	22,5060	18,0039	13,4995	8,9987	4,5019	4,9985
	±0,0231	±0,0292	±0,0310	±0,0387	±0,0386	±0,0371	±0,0366	±0,0332	±0,0268	±0,0020
6	41,8922	36,3079	31,0166	25,8617	20,7973	15,8128	10,9596	6,3123	2,1871	6,2873
	±0,0157	±0,0184	±0,0166	±0,0181	±0,0190	±0,0181	±0,0149	±0,0100	±0,0033	±0,0028

FR_1	FR_2	FR_3	FR_4	FR_5	FR_6	FR_7	FR_8	$CPU(s)$
2,5457	2,3604	2,2851	2,2328	2,1998	2,1437	2,0523	1,7678	2970,2425
±0,0066	±0,0078	±0,0069	±0,0061	±0,0074	±0,0072	±0,0068	±0,0065	
2,8278	2,6254	2,5351	2,4976	2,4238	2,3701	2,2696	1,9438	3148,4971
±0,0153	±0,0155	±0,0144	±0,0185	±0,0120	±0,0142	±0,0141	±0,0138	
2,0282	1,9556	1,9389	1,9317	3,8273	0,5167	0,5698	0,6092	3102,1409
±0,0044	±0,0054	±0,0043	±0,0039	±0,0025	±0,0035	±0,0042	±0,0038	
2,0064	1,9909	1,9969	1,9904	3,9998	0,0701	0,0708	0,0775	3115,8593
±0,0048	±0,0052	±0,0044	±0,0056	±0,0033	±0,0018	±0,0016	±0,0025	
4,9995	4,9994	4,9993	4,9991	4,9990	4,9989	4,9988	4,9987	3186,9086
±0,0019	±0,0019	±0,0019	±0,0019	±0,0019	±0,0019	±0,0019	±0,0019	
3,3942	3,1472	3,0467	2,9771	2,9331	2,8583	2,7364	2,3570	3220,9396
±0,0088	±0,0104	±0,0092	±0,0082	±0,0098	±0,0096	±0,0091	±0,0087	

Table 8. Percent difference in performance measure estimates obtained by decomposition and simulation for Example 2.

#	$\bar{X}_1$	$\bar{X}_2$	$\bar{X}_3$	$\bar{X}_4$	$\bar{X}_5$	$\bar{X}_6$	$\bar{X}_7$	$\bar{X}_8$	$\bar{X}_9$	TH
1	-0,8250	-1,3840	-1,8951	-2,3568	-2,7321	-2,9052	-2,8386	-2,0516	-0,8916	-0,6298
2	-0,7701	-1,2483	-1,7114	-2,0968	-2,2605	-2,5006	-2,4077	-1,6255	-0,6064	-0,3120
3	-0,1641	-0,2682	-0,3457	-0,4034	-0,4278	-0,4592	-0,3391	-0,2830	-0,0255	-0,0631
4	0,0376	0,0473	0,0459	0,0585	0,0619	-0,0947	0,1631	0,1800	0,0622	-0,0200
5	0,0089	0,0056	0,0206	-0,0089	-0,0311	-0,0167	0,0259	0,0078	-0,0777	0,0400
6	-0,8250	-1,3840	-1,8951	-2,3568	-2,7321	-2,9052	-2,8386	-2,0516	-0,8916	-0,6298

FR_1	FR_2	FR_3	FR_4	FR_5	FR_6	FR_7	FR_8
0,9113	-0,2330	-1,5754	-2,5842	-4,1186	-5,0613	-6,0761	-4,6216
0,9442	-0,3390	-1,4477	-3,2711	-3,3419	-4,5863	-5,5869	-3,7915

0,3205	-0,3426	-0,7891	-1,0302	-0,3161	-0,6193	-1,1583	-1,8385
-0,1894	0,0352	-0,3005	0,0201	-0,0100	-0,9986	-0,1412	0,5161
0,0180	0,0200	0,0180	0,0180	0,0160	0,0140	0,0140	0,0160
0,9163	-0,2320	-1,5722	-2,5831	-4,1219	-5,0624	-6,0773	-4,6203

The observations on the results of Example 1 still hold for the results of Example 2. The only significant difference is that in Example 2, the computational time of the decomposition method is higher than it is in Example 1 but still lower than the corresponding time of simulation. This is natural because in Example 2, there are twice as many stages (machines) and – more importantly – the echelon buffer capacities are much higher.

Chapter 5 - Conclusions

In this thesis, we introduced the EB policy for controlling the flow of parts through a serial production line, and we developed a decomposition approximation method for evaluating its performance. Our numerical results show that this method is computationally efficient and highly accurate when compared to simulation. Given the promising results regarding the performance of the EB policy, a worthwhile direction for future research would be to generalize the decomposition method for more complicated machine behavior models than the exponential processing time assumption of Bernoulli model. Even under the continuous-time equivalent of Bernoulli machine, however, it would be useful to come up with a more efficient way to analyze the 2-machine subsystems in isolation in the decomposition method. A shortcoming of the EB policy is that it has increased material handling requirements compared to the IB policy. However, modern technology can handle such increased requirements at affordable costs. Finally, under the EB policy, parts are produced earlier by the first and the last machine of a production line than they do under the IB policy; hence, the average throughput of the line is higher. On the downside, parts spend more time in the line under the EB policy than they do under the IB policy as a result of the increased congestion induced by the former policy. From Little's law, this implies that the average WIP in the line is higher under the EB policy than it is under the IB policy. The question whether the benefit of the throughput increase under the EB policy outweighs the disadvantage of the WIP increase, also taking into account that less total buffer space may be needed under the EB policy than under the IB policy to achieve the same throughput level, can be a matter of further investigation.

Appendix

MATLAB Algorithms

DECOMPOSITION:

1. External Algorithm

```
% Decomposition-based approximate solution for an N-machine
% (N-1)-(echelon) buffer flow line with Exponential Machines
% Input:
% k = number of machines
% p = (array) probability that machine M_i produces one unit in one time
% unit, i = 1, ..., N
% N = (array) buffer size (including space in M_d) of buffer B_i,
% i = 1, ..., N-1;
% Output:
% Average throughput and buffer level vector

tic;

% Input parameters for testing

k = 5;                                  % number of machines
p = [8,8,8,8,8];                        % production rates
N = [10,10,10,10];

% Decomposition parameters
epsilon = 0.0001;               % convergence criterion

% Initialization
NE = fliplr(cumsum(fliplr(N))); % Echelon buffer sizes
NE1 = NE + 1;
BEDec = zeros(1,k-1);           % Average echelon buffer levels
FRDec = zeros(1,k-1);           % Frequency of installation buffer overflow

% Initialization
lu = zeros(k-1, NE1(1));
l = lu;
ld = lu;
Pmarg = lu;
PP = zeros (k-1, NE1(1), NE1(2));

count = zeros(1,k-1);

ld(1,1:NE1(1)) = [0, ones(1,NE(1))*min(p(2:k))];
Pmarg(1,1:NE1(1)) = InitPcont( ones(1,NE1(1))*p(1), ones(1,NE1(1))*min(p(2:k)),
NE(1));
for m=2:k-1
    lu(m,1:NE(m-1)) = ones(1,NE(m-1))*min(p(1:m-1));
    l(m,2:NE1(m-1)) = ones(1,NE(m-1))*p(m);
    ld(m,2:NE1(m)) = ones(1,NE(m))*min(p(m+1:k));
    Pmarg(m,1:NE1(m)) = InitPcont ( ones(1,NE1(m))* min(p(1:m)), ...
        ones(1,NE1(m))*min(p(m+1:k)), NE(m));
```

```
    SumPP = 0;
    for j = 1:NE1(m)
        for i = 1:NE1(m-1)-j+1
            PP(m,i,j) = Pmarg(m-1,i)*Pmarg(m,j);
            SumPP = SumPP + PP(m,i,j);
        end
    end
    PP(m,:,:) = PP(m,:,:)/SumPP;
end

i = k-1;

% Iterations
while i >= 2
    count(i) = count(i) + 1;
    dif = 0;
    [lout, rout, Bavout, FRout, Pout] = AlgLncontFR( lu(i,1:NE1(i-1)), ...
        l(i,1:NE1(i-1)), ld(i,1:NE1(i)), NE(i), NE(i-1), ...
        squeeze(PP(i,1:NE1(i-1),1:NE1(i))) );
    PP(i,1:NE1(i-1),1:NE1(i)) = Pout;
    BEDec(i) = Bavout;
    FRDec(i-1) = FRout;
    if i < k-1
        dif = max( abs(lout - lu(i+1,1:NE1(i))) / lu(i+1,1:NE1(i)) );
    end
    if dif < epsilon
        ld(i-1,1:NE1(i-1)) = rout;
        i = i-1;
    else
        lu(i+1,1:NE1(i)) = lout;
        i = i+1;
    end
end

[THEDec, Bavout] = AlgL1cont( p(1), ld(1,1:NE1(1)), NE(1) );
BEDec(1) = Bavout;

CPUDec = toc;
```

2. Internal function `InitPcont`

```
function P = InitPcont( l, ld, K )

% Analytical solution for a 2-machine 1-buffer flow line with Exponential
% machines

% parameters for testing the function
% K = 6;
% l = ones(1,K+1)*0.6;
% ld = ones(1,K+1)*0.4;

% Decomposition parameters
C = ones(1,K+1);           % C_i factors for the birth-death process

for i = 2:K+1
    C(i) = C(i-1)*l(i-1) / ld(i);
```

```
end
P = C/sum(C);
end
```

3. Internal function `AlgLncontFR`

```
function [lout, rout, Bavout, FRout, P] = AlgLncontFR( lu, l, ld, Kl, Kh, P )

% Parameters for independent testing of function
% Kl = 2;
% Kh = 4;
% lu = [ones(1,Kh)*0.4, 0];
% l = [0, ones(1,Kh)*0.5];
% ld = [0, ones(1,Kl)*0.6];
% numelements = (Kh+1)*(Kl+1)-(Kl+1)*Kl/2;     % Number of states
% P = ones(Kh+1, Kl+1)/numelements;             % Transition probability matrix

lout = zeros(1,Kl+1);
numlout = lout;
denlout = lout;
rout = zeros(1,Kh+1);
numrout = rout;
denrout = rout;
Bavout = 0;
FRout = 0;

% Decomposition parameters
epsilon = 0.0001;                               % convergence criterion
dif = epsilon + 1;                              % max difference

while dif > epsilon
    SumP = 0;
    Pold = P;

    % *** MIDDLE STATES ***
    for j = 2: Kl-1
        for i = 2: Kh-j+1
            num = P(i-1,j) * lu(i+j-2) + ...
                  P(i,j+1) * ld(j+1) + ...
                  P(i+1,j-1) * l(i+1);
            den = l(i) + ld(j) + lu(i+j-1);
            P(i,j) = num/den;
            SumP = SumP + P(i,j);

        end
    end
    if Kl >= 2
        for i = 2: Kh-Kl+1
            num = P(i-1,Kl) * lu(i+Kl-2) + ...
                P(i,Kl+1) * ld(Kl+1) + ...
                P(i+1,Kl-1) * l(i+1);
            den = lu(i+Kl-1) + l(i) + ld(Kl);
            P(i,Kl) = num/den;
            SumP = SumP + P(i,Kl);

        end
    end
```

```
% *** LEFT COLUMN ***
%if Kl >= 2
    for i = 2: Kh
        num = P(i-1,1) * lu(i-1) + P(i,2) * ld(2);
        den = lu(i) + l(i);
        P(i,1) = num/den;
        SumP = SumP + P(i,1);

    end
%end

% *** RIGHT COLUMN ***
for i = 2: Kh-Kl
    num = P(i-1,Kl+1) * lu(i+Kl-1) + ...
        P(i+1,Kl) * l(i+1);
    den = lu(i+Kl) + ld(Kl+1);
    P(i,Kl+1) = num/den;
    SumP = SumP + P(i,Kl+1);

end

% *** TOP ROW ***
for j = 2: Kl
    num = P(1,j+1) * ld(j+1) + P(2,j-1) * l(2);
    den = lu(j) + ld(j);
    P(1,j) = num/den;
    SumP = SumP + P(1,j);

end

% *** DIAGONAL BOTTOM RIGHT ***
for j = 2: Kl;
    i = Kh-j+2;
    num = P(i-1,j) * lu(i+j-2) + ...
          P(i+1,j-1) * l(i+1);
    den = l(i) + ld(j);
    P(i,j) = num/den;
    SumP = SumP + P(i,j);

end

% *** TOP LEFT CORNER ***
P(1,1) = P(1,2)* ld(2) / lu(1);
SumP = SumP + P(1,1);


% *** BOTTOM LEFT CORNER ***
P(Kh+1,1) = P(Kh,1) * lu(Kh) / l(Kh+1);
SumP = SumP + P(Kh+1,1);


% *** TOP RIGHT CORNER ***
P(1,Kl+1) = P(2,Kl) * l(2) / (lu(Kl+1) + ld(Kl+1));
SumP = SumP + P(1,Kl+1);
```

```
    % *** BOTTOM RIGHT CORNER ***
    if Kh > Kl
            P(Kh-Kl+1,Kl+1) = (P(Kh-Kl,Kl+1) * lu(Kh) + ...
            P(Kh-Kl+2,Kl) * l(Kh-Kl+2)) / ld(Kl+1);
        SumP = SumP + P(Kh-Kl+1,Kl+1);

    end
    P = P/SumP;
    dif = max(max(abs((P - Pold))./Pold));
end

for j = 1:Kl+1
    for i = 1:Kh-j+2
        numlout(j) = numlout(j) + P(i,j)*l(i);
        denlout(j) = denlout(j) + P(i,j);
        numrout(i+j-1) = numrout(i+j-1) + P(i,j)*ld(j);
        denrout(i+j-1) = denrout(i+j-1) + P(i,j);
        Bavout = Bavout + P(i,j)*(j-1);
    end
    for i = Kh - Kl + 1: Kh - j + 1
        FRout = FRout + P(i,j)*lu(i+j-1);
    end
end
lout = numlout./denlout; %internal state dependent arrival probabilities
lout(Kl+1) = 0;
rout = numrout./denrout; %conditional throughput r of subsystem Ln

end
```

4. Internal function `AlgLncontFR`

```
function [rout, Bavout] = AlgL1cont( l, ld, K )
% Analytical solution for a 2-machine 1-buffer flow line with Exponential
% machines

% parameters for testing the function
% K = 5;
% l = 0.6;
% ld = [0, ones(1,K)*0.5];

% Decomposition parameters
C = ones(1,K+1);
for i = 2:K+1
    C(i) = C(i-1)*l / ld(i);
end
P = C/sum(C);
rout = l * (1-P(K+1));
Bavout = sum((0:K).*P);

end
```

SIMULATION:

```
% SIMULATION SOLUTION
% Flow Line with Echelon Buffers and Exponential Machines

tic;

% Problem parameters

k = 5;                          % number of machines
p = 1./[9,9,6,9,9];             % production rates
N = [5,5,5,5];                  % buffer sizes

NE = fliplr(cumsum(fliplr(N))); % Echelon buffer sizes

% Simulation parameters
% Tsim = 400000;           % simulation horizon
MaxProd = 200000;        % number of parts produced before simulation ends
Reps = 30;               % Number of simulation runs
rng('default');          % random number generator reset
rng(1);                  % random number generator seed

% Initialization
rep = 0;                          % repetition index
THEarraySim = zeros(1,Reps);      % Av Throughput per sim run
BEmatSim = zeros(Reps,k-1);       % Av Bi vector per sim run
FRmatSim = zeros(Reps,k-1);       % Av Bi vector per sim run

% Repeat simulation runs
while rep < Reps
    BE = zeros(1,k-1);
    BEcum = BE;                 % BE*time in BE
    FR = zeros(1,k-1);          % Freq(i) = frequency of overflow of B_i

    t = 0;
    Prod = 0;
    % Simulate individual run
    tnext = exprnd(p);          % Generate random processing times for the machines

    %     while t < Tsim
    while Prod < MaxProd
        BEcur = BE;
        DFR = zeros(1,k-1);
        enable = zeros(1,k);    % Initiatially no production event is enabled
        tmin = 100000;
      % tmin is the minimum processing time among all enabled machines; initially
      it is set to a very large number.

      %Find next event and tmin
        if BE(1) < NE(1) % M1 not blocked
                  enable(1) = 1;  % If M1 is not blocked then it is enabled to
      process its part
            if tnext(1) < tmin
                tmin = tnext(1);
                tminidx = 1;    % tminidx is the machine that produces next
            end
        end
```

```
        for i=2:k-1
            if BE(i) < min(NE(i),BE(i-1))  % Mi not blocked or starved
                enable(i) = 1;
      % If Mi is neither blocked nor starved then it is enabled to process its part
                if tnext(i) < tmin
                    tmin = tnext(i);
                    tminidx = i;
                end
            end
        end

        if BE(k-1) > 0 % Mk not starved
            enable(k) = 1;
            if tnext(k) < tmin
                tmin = tnext(k);
                tminidx = k;
            end
        end

        % Update event times and state
        for i=1:k-1
            if enable(i) == 1
                if i == tminidx
                    if i < k-1
                        if BEcur(i) - BEcur(i+1) >= N(i)
                            DFR(i) = 1;
                        end
                    end
                    BEcur(i) = BEcur(i)+1;
                    tnext(i) = exprnd(p(i));
                else
                    tnext(i) = tnext(i) - tmin;
                end
            end
        end
        if enable(k) == 1
            if k == tminidx
                BEcur = BEcur - 1;
                Prod = Prod + 1;
                tnext(k) = exprnd(p(k));
            else
                tnext(k) = tnext(k) - tmin;
            end
        end

        %calculate frequency of buffer overflow
        FR = FR + DFR;

        BE = BEcur;
        BEcum = BEcum + BE*tmin;
        t = t + tmin;
    end
    rep = rep+1;
    THEarraySim(rep) = Prod/t; % Average throughput TH
    BEmatSim(rep,:) = BEcum/t; % Average WIP Bi, i=1,...,k-1
    FRmatSim(rep,:) = FR/t;    % Average frequency of buffer overflow
end
```

```
% Compute mean and CI of throughput and buffer levels
THESim = mean(THEarraySim);
THEstdSim = std(THEarraySim);
THECISim = [THESim - 1.96*THEstdSim/sqrt(Reps), THESim +
1.96*THEstdSim/sqrt(Reps)];

BESim = mean(BEmatSim,1);
BEstdSim = std(BEmatSim,1);
BECISim = [BESim - 1.96*BEstdSim/sqrt(Reps), BESim + 1.96*BEstdSim/sqrt(Reps)];

FRSim = mean(FRmatSim,1);
FRstdSim = std(FRmatSim,1);
FRCISim = [FRSim - 1.96*FRstdSim/sqrt(Reps), FRSim + 1.96*FRstdSim/sqrt(Reps)];

BEImatSim=[fliplr(diff(fliplr(BEmatSim),1,2)),BEmatSim(:,k-1)];
BEISim = mean(BEImatSim,1);
BEIstdSim = std(BEImatSim,1);
BEICISim = [BEISim - 1.96*BEIstdSim/sqrt(Reps); BEISim +1.96*BEIstdSim/sqrt(Reps)];

CPUSim = toc;
```

References

1. Dallery, Y. and Gershwin, S. B. (1992) Manufacturing flow line systems: a review of models and analytical results. *Queueing Systems Theory and Applications*, 12 (1), 3-94.
2. Gershwin, S. B. (1994) *Manufacturing Systems Engineering*, Prentice-Hall, Englewood Cliffs, NJ.
3. Li, J. and Meerkov, S. M. (2000) Production variability in manufacturing systems: Bernoulli reliability case. *Annals of Operations Research*, 93 (1), 299-324.
4. Meerkov, S. M. and Zhang, L. (2008) Transient behavior of serial production lines with Bernoulli machines. *IIE Transactions*, 40 (3), 297-312.
5. Biller, S., Li, J., and Meerkov, S. M. (2009) Bottlenecks in Bernoulli serial lines with rework. *IEEE Transactions on Automation Science and Engineering*, 7 (2), 208-217.
6. Altiok, T. (1997) *Performance Analysis of Manufacturing Systems*, Springer Series in Operations Research, Springer, New York, NY.
7. Li. J. and Meerkov, S. M. (2009) *Production Systems Engineering*, Springer, New York, NY.
8. Buzacott, J. A. (1972) The Effect of Station Breakdowns and Random Processing Times on the Capacity of Flow Lines with In-Process Storage. *AIIE Transactions*, 4 (4), 308-312.
9. Gershwin, S. B. and Berman, O. (1981) Analysis of transfer lines consisting of two unreliable machines with random processing times and finite storage buffers. *AIIE Transactions*, 13 (1), 2-11.
10. Gershwin, S. B. (1987) An efficient decomposition method for the approximate evaluation of tandem queues with finite storage space and blocking. *Operations Research*, 35 (2), 291-305.
11. Choong, Y. F., and Gershwin, S. B. (1987) A Decomposition Method for the Approximate Evaluation of Capacitated Transfer Lines with Unreliable Machines and Random Processing Times. *IIE Transactions*, 19 (2), 150-159.
12. Buzacott, J. A. (1967) Automatic transfer lines with buffer stocks. International Journal of Production Research, 5 (3), 183-200.
13. Dallery, Y., David, R., and Xie, X.-L. (1988) An efficient algorithm for analysis of transfer lines with unreliable machines and finite buffers. *IIE Transactions*, 20 (3), 280-283.
14. Levantesi, R., Matta, A., and Tolio. T. (2003). Performance evaluation of continuous production lines with machines having different processing times and multiple failure modes. *Performance Evaluation*, 51 (2-4), 247-268.
15. Tan, B. and Gershwin, S. B. (2009) Analysis of a general Markovian two-stage continuous-flow production system with a finite buffer. *International Journal of Production Economics*, 120 (2), 327-339.
16. Colledani, M. (2013) A Decomposition Method for the Analysis of Long Buffered Production Systems with Discrete General Markovian Machines *IFAC Proceedings Volumes*, 46 (9), 1644-1649.
17. Clark, A. J. (1958) A dynamic, single-item, multi-echelon inventory model. RM-2297, ASTIA Doc. No. AD 209538, Santa Monica California, The RAND Corporation.
18. Axsäter, S., and Rosling, K. (1993) Installation vs. echelon stock policies for multilevel inventory control. *Management Science*, 39 (10), 1274-1280.

19. Tempelmeier, H., Kuhn, H., and Tetzlaff, U. (1989) Performance evaluation of flexible manufacturing systems with blocking. *International Journal of Production Research*, 27 (11), 1963-1979.
20. Tempelmeier, H. and Kuhn, H. (1993) Flexible manufacturing Systems: *Decision Support for Design and Operation*, Wiley, New York, NY.
21. Ferrari, D., Matta, A. (2007) Analytical performance evaluation of small flow lines with shared buffer. *IFAC Proceedings Volumes*, 40 (6), 7–12.
22. Papadopoulos, H. T. and Heavey, C. (1996) Queueing theory in manufacturing systems analysis and design: a classification of models for production and transfer lines. *European Journal of Operational Research*, 92 (1), 1-27.
23. Hillier, F. S. and Boling, W. R. (1967) Finite Queues in Series with Exponential or Erlang Service Times - A Numerical Approach. *Operations Research*, 15 (2), 286-303.
24. Altiok, T. (1982) Approximate analysis of exponential tandem queues with blocking. *European Journal of Operational Research*, 11 (4), 390–398.
25. Hendricks, K. B. (1992) The output processes of serial production lines of exponential machines with finite buffers *Operations Research* 40 (6), 1139-1147.
26. Springer, M. C. (1994) A decomposition approximation for finite-buffered flow lines of exponential queues. *European Journal of Operational Research*, 74 (1), 95-110.
27. Zhou, W. and Lian, Z. (2011) A tandem network with a sharing buffer. *Applied Mathematical Modeling*, 35 (9), 4507-4515.
28. Koukoumialos, S. and Liberopoulos, G. (2005) An analytical method for the performance evaluation of echelon kanban control systems. *OR Spectrum*, 27 (2), 339-368.