European Journal of Operational Research 37 (1988) 214–226
North-Holland

Theory and Methodology

Approximate analysis of production systems

M.B.M. de KOSTER *
Department of Industrial Engineering, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, Netherlands

Abstract: In this paper complex production systems are studied where a single product is manufactured and where each production unit stores its output in at most one buffer and receives its input from at most one buffer. The production units and the buffers may be connected nearly arbitrarily. The buffers are supposed to be of finite capacity and the goods flow is continuous. For such networks it is possible to estimate the throughput by applying repeated aggregation over the production units. The approximation appears to be best when the network shows some resemblance with a flow line.

Keywords: Queues, networks, manufacturing industries, design

1. Introduction

In estimating the performance of production systems it is often useful to model the system as a network of linked finite queues. Recently approximation techniques have been developed to analyse such networks.

Most papers deal with very specific networks like flow lines, for example. In a flow line all products (actually of the same type) have the same routing over the machines. Each machine has its own queue in front of it. Perros and Altiok [18] discuss a method to approximate the marginal probability distribution of the number of items in each queue and the line throughput. Their method performs well for not too unbalanced lines. Similar results have been achieved by Altiok [1], Pollock et al. [20], Takahashi et al. [22], Hillier and Boling [15], Choong and Gershwin [7] and Brandwajn and Jow [6].

* Research supported by the Netherlands Organization for the Advancement of Pure Research (ZWO)

Received April 1987; revised October 1987

In all these papers, except [15] and [20] for reasons of computational complexity, service times are supposed to be exponentially distributed. In Glassey and Hong [14] and De Koster [10] the throughput of flow lines with a continuous product flow is approximated. The machines have production rates instead of service times and are subject to failure.

Other special networks for which approximation methods exist are so-called split and merge configurations, that is one machine is linked with a number of parallel machines and each machine has its own queue in front of it. Examples are Boxma and Konheim [5] and Altiok and Perros [3]. The blocking mechanisms studied in these papers differ from each other.

Combining the approximation techniques for flow lines with those of split and merge configurations Boxma and Konheim [5], Altiok and Perros [2] and Perros and Snyder [19] developed algorithms for the approximate analysis of arbitrary networks of finite queues. In [5] however, no numerical results are given for such combinations. The numerical results of [2] and [19] are good.

Also closed queueing networks have been approximated. Suri and Diehl [21] discuss a closed network of finite queues in series, of which however the first queue has a capacity larger than the number of products in the system. They give approximations for the line throughput and the blocking probabilities. Onvural and Perros [17] obtain equivalencies between closed queueing networks with blocking with respect to buffer capacities and number of customers in the network.

De Koster [12] studies flow lines with a continuous goods flow where the machines are controlled by the total work-in-process downstream. A particular production unit operates only if the total work-in-process downstream of that unit does not exceed a certain fixed level. This model covers also the closed queueing model. The production lines in the paper are approximated by ordinary flow lines with about the same throughput, probabilities of full and empty buffers and mean total work-in-process.

In this paper we consider open networks consisting of combinations of split and merge configurations with finite intermediate buffers. The layout of the networks may be nearly arbitrarily, however, no loops are allowed. A more precise description is given in Section 2. An example of such a network is sketched in Figure 1.

In Figure 1 a machine is denoted by capital M and a buffer by capital B.

The major difference with the networks studied in [2] or [19] is that the product flow is continuous and the machines have production rates instead of service times. Furthermore in the network of Figure 1 machines may share buffers, whereas in [2] and [19] each machine has its own buffer in front of it. This means that we have to do with a global restriction holding for several machines, rather than a local restriction for each machine. The difference is, that when several machines obtain form a single buffer and that buffer is (nearly) empty, then it has to be decided which machines are allowed to operate on the products. A similar situation arises when several machines supply a single buffer. In general the installation of such a common buffer preceding several machines, improves the performance of the system. If products are allocated to a single machine and that machine fails, then the system stops, whereas if the products are not allocated beforehand, they can be processed by other machines.

The approximation of the throughput of a network as in Figure 1 is the result of the reduction of the network to a two-stage system with one intermediate buffer. The modelling of the machines with service rates and a continuous product flow implies that we encounter no extra computational difficulties in the evaluation of two-stage systems, when the intermediate buffers have relatively large capacities. Furthermore we suppose that the machines in Figure 1 have an arbitrary (but finite) number of states, each with its own production rate and each with an exponential duration. The transitions between states follow an irreducible Markov process. These assumptions allow for the approximate solution of complex networks, with a large number of machines and buffers, within a reasonable time. We treat a case with nine machines and four buffers.

In the reduction of a network as sketched in Figure 1 to a two-stage line we repeatedly use two different aggregation steps, which will be clarified in Section 3. The one step is the aggregation of two sets of parallel machines in series, with a single intermediate buffer. The other step is the aggregation of two machines and an intermediate

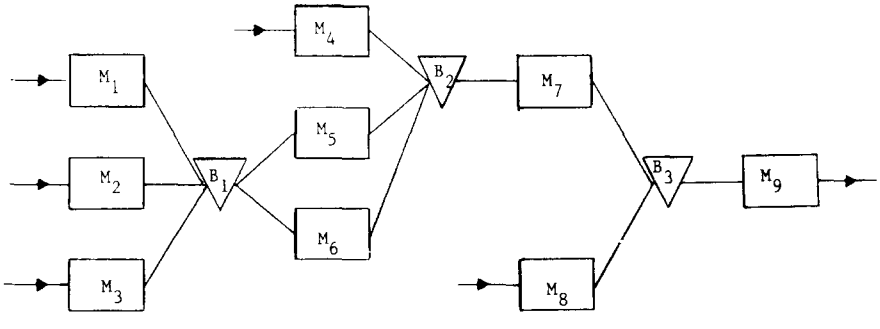

Figure 1. Example of the type of network studied in the paper

buffer. In both steps the resulting aggregate machine is a machine with two states.

In Section 2 the allowed production systems will be described and in Section 4 numerical results presented. Finally, in Section 5 some conclusions are drawn.

2. The production systems

The production systems studied here consist of N machines connected arbitrarily with intermediate buffers. However, we assume that there is only one type of product flowing through the machines. This one product should be interpreted as some aggregate product. If in reality there are many products then this can be achieved by expressing the products in each buffer in 'work' for the succeeding machines. If the average machine speeds are known ('products' per unit of time), then the amount of work can again be expressed in amount of product.

No directed cycles are allowed in the network. Furthermore each machine receives input from at most one buffer and stores its output in at most one buffer. Each network of this type without bypasses is allowed, however sometimes networks with bypasses can also be solved. This is the case when also the aggregate machines, obtained by aggregation over two (sets of parallel) machines with a single intermediate buffer, receive input from and store their output in exactly one buffer. An example of an allowed network has been given in Figure 1 in Section 1. Note that in this type of network, assembly and disassembly of different products and reprocessing of failed products is not allowed. In De Koster [13] attention to this kind of network is paid.

If several machines are obtaining from a single buffer, or supplying a single buffer, then a (relative) priority is given to the machines (or, more accurately, to the arcs connecting the machines with the buffer). This priority rule determines the machine speeds in case the buffer is empty and the machines obtain from the buffer, or in case the buffer is full and the machines supply the buffer. Several priority rules are possible. The priority rule chosen in this paper is explained in the sequel.

The full list of assumptions the networks have to satisfy is as follows.

(a) The directed graph representing the structure of the network doesnot contain directed cycles.

(b) Each machine supplies at most one buffer and obtains from at most one buffer. Machines that have no upstream buffer are called *source* machines, machines that have no downstream buffer are called *sink* machines. There is a directed path from a source machine to a sink machine.

(c) With each arc connecting a machine and a buffer there is associated a relative priority of the connection. A *bypass* is a set of two different paths from a buffer B_1 to another buffer B_2. For every bypass, if the priority of the first arc of the one path is greater than the priority of the first arc of the other path at B_1, then the same has to hold at B_2 for the last arc of both paths.

(d) Bypasses are allowed only insofar that repeated aggregation steps over two (sets of parallel) machines in series with a single intermediate buffer, must result in a correct network. That is, in the resulting network each (aggregate) machine must supply at most one buffer and obtain from at most one buffer.

In the networks studied in this paper the production is supposed to be continuous. M_i is able to work at speeds $\nu_1^i, \ldots, \nu_{N_i}^i$ (not necessarily different). M_i works at speed ν_r^i during an exponentially distributed interval (parameter λ_r^i). After such a period with speed ν_r^i a transition takes place with probability p_{rs}^i to a state with speed ν_s^i. The matrix $P^i = (p_{rs}^i)$ is an irreducible Markov matrix. Note that by choosing some ν_r^i 's equal, it is possible to generate arbitrary phase-type distributions for the intervals that a machine is working at a certain speed.

We suppose that the machines in the system not preceded by a buffer are never starved, that is, they always have items to work on. In a similar way the machines not succeeded by a buffer are never blocked by lack of storage capacity for finished items. Since buffers are assumed to be of finite capacity it may occur that machines are forced down or slowed down by full or empty buffers. Suppose buffer B_l is full and let it be preceded by machines $M_{n_1}, \ldots, M_{n_r}$ and succeeded by machines $M_{m_1}, \ldots, M_{m_s}$. If the total net speed of all machines $M_{n_1}, \ldots, M_{n_r}$ is greater than the total net speed of machines $M_{m_1}, \ldots, M_{m_s}$ at that moment, then machines $M_{n_1}, \ldots, M_{n_r}$ will have to

slow down till their total speed equals the total speed of machines $M_{m_1}, \ldots, M_{m_s}$. The net speed reduction of the machines $M_{n_1}, \ldots, M_{n_r}$ due to blocking is allocated to these machines in order of decreasing priority. (The highest priority machine obtains the greatest speed.) The priority of M_{n_i} is greater than M_{n_j} iff $n_i < n_j$. For example, let the net total speed of machines $M_{m_1}, \ldots, M_{m_s}$ be w_2, the net total speed of machines $M_{n_1}, \ldots, M_{n_r}$ be $w_1 > w_2$ and the instantaneous speed of M_{n_i} be $v(n_i)$. Then the speed of M_{n_i}, $w(n_i)$, becomes

$$\min\left\{v(n_i),\ w_2 - \sum_{n_j < n_i} w(n_j)\right\}.$$

In the same way machines $M_{m_1}, \ldots, M_{m_s}$ may be slowed down, if B_l is empty and $w_2 > w_1$. This slowing down is also in order of decreasing priority.

Since all machines are connected to each other via buffers the effects of machines slowing down other machines through full or empty buffers may propagate through the network.

It is possible to use other priority rules, but for many priority rules the determination of the machine speeds in case of slowing down by full or empty buffers is not a simple task. In fact certain priority rules may lead to unsolvable systems of equations or systems with more than one solution for the machine speeds. The main reason for using this particular priority rule is that the system of equations for the machine speeds is always solvable and has a unique solution. This is the reason for requiring network condition (c). This fact is demonstrated in the appendix. In the appendix also an efficient algorithm is given to determine the machine speeds in the network at change points, based on dynamic programming.

Priority rules, as the one mentioned earlier, make the network reversible. This means that if the direction of the product flow through all buffers is reversed then the throughput remains equal. Even stronger, for a network controlled in this way it holds that the output process equals the input process of the same network with a reversed product flow. The same holds for the input process compared with the output process of the reversed network. This is a consequence of the fact that if a machine is slowed down by an empty buffer, then it is slowed down in exactly the same way (by the priority rules) by a full buffer in the reversed network and the fact that a buffer content X_i of the i-th buffer corresponds to a buffer content $K_i - X_i$ in the reversed network (provided this condition is also satisfied initially). For a more formal proof see Corten and De Koster [8]. For more information about the reversibility of production systems see Muth [16], Yamazaki et al. [24], or Ammar and Gershwin [4]. Reversibility as such is not needed for the approximation algorithm, but in particular cases it may be convenient to describe the input behaviour of the reversed network rather than the output behaviour of the original network.

The capacity of buffer B_i is K_i and the total throughput of the system (the average production per unit of time of all sink machines) is denoted by $v(K)$, where K is the vector of the buffer capacities.

3. The approximation steps

As indicated in the introduction, in approximating a production system as in Figure 1 we only use two sorts of approximation steps, sketched in Figures 2 and 3. The step sketched in Figure 2 is the aggregation of two sets of parallel machines in series. The step sketched in Figure 3 is the aggregation of two machines and an intermediate buffer. Although strictly speaking each line as sketched in Figure 2 can be represented as a line as in Figure 3 we distinguish between these lines since we will apply the approximation step of figure 3 only in

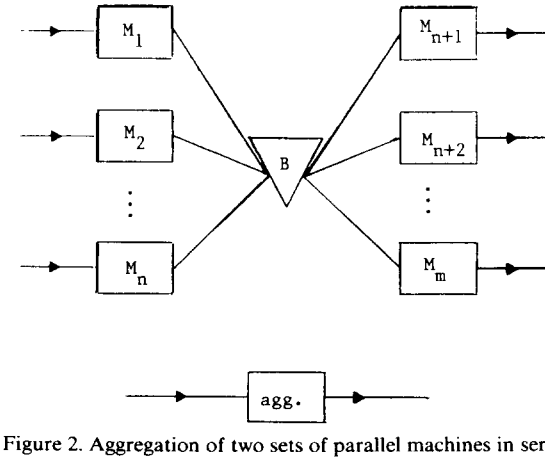

Figure 2. Aggregation of two sets of parallel machines in series

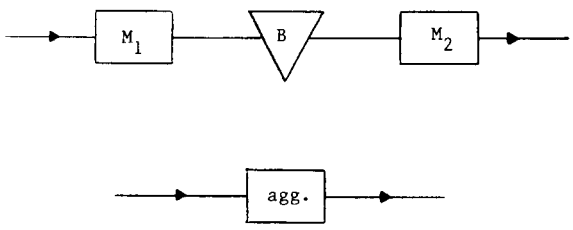

Figure 3. Approximation of a two-stage line

two special cases. These cases are: both the first and the second machine in Figure 3 have exactly two states and two speeds and the case where the first machine has four states and two speeds (each corresponding with two states) and the second machine has exactly one state. In the step of Figure 2 we additionally use an intermediate step. In this intermediate step, which is described in De Koster [9], the line of Figure 2 is first approximated by another two-stage line, however, this whole line has only four states (the line in Figure 2 may have many more states). Actually this intermediate line is of the first type as dealt with in Figure 3. Only after applying this intermediate step we approximate the resulting two-stage line by a two-state aggregate machine. The reason for applying this extra step is that it allows an approximation of very general networks. If the system of Figure 2 consists of many machines, then the system can be in many states and the calculation of the parameters of the aggregate machine may give computational difficulties. The intermediate step does not give such difficulties. Although in the examples given in Section 4 the state space is not so large that the parameters of the aggregate machine could not be solved, still this intermediate step is used to make the approximation method generally applicable.

The two approximation steps will be clarified in Subsections 3.1 and 3.2 respectively.

3.1. *Approximation of parallel machines in series*

Suppose we have n parallel machines connected with m parallel machines with an intermediate buffer of capacity K as sketched in Figure 4.

For reasons of convenience we suppose the parallel machines have only two states, namely an up-state (with speed ν_i for the first level machines $M1_i$) and a down-state with speed zero. The failure rate for $M1_i$ is denoted by λ_i, the repair rate by μ_i. The parameters of the second-level machines $M2_i$ are denoted by ω_i for the production rate, η_i for the failure rate and ρ_i for the repair rate. Both sets of parallel machines are now represented as one production unit (PU). If all speeds of the first-level machines are different then the first level production unit ($= PU_1$) has 2^n states. Such a state can be represented by an n-tuple $(\delta_1, \ldots, \delta_n)$ where δ_j is shorthand for 1, if machine j is up and 0, if machine j is down. The production rate of PU_1 corresponding with this state is $\sum_{j=1}^{n} \nu_j \delta_j$ and the duration of an interval with this production rate is exponentially distributed with parameter $\sum_{j=1}^{n} [\lambda_j \delta_j + \mu_j(1 - \delta_j)]$. If the states of PU_1 are ordered in decreasing lexicographic order then the element p_{12}, for instance, in the transition matrix $P = (p_{ij})$ between these states, equals $\lambda_n/(\lambda_1 + \cdots + \lambda_n)$. Note that in general $p_{ij} \geqslant 0$, $p_{ii} = 0$ and $\sum_{j=1}^{n} p_{ij} = 1$. The parameters of PU_1 and PU_2 can be derived similarly if the first level and second level machines have more than two states.

The production rate of PU_1 will sometimes be greater than the rate of PU_2 and sometimes be smaller (otherwise the buffer has no function). Therefore the buffer content will sometimes increase and sometimes decrease. In De Koster [9] it is shown that the throughput (and also the distribution of the buffer content and many other line characteristics) of a two-stage line is determined almost completely by the buffer fluctuations. Actually only the first and second moments of buffer increase and buffer decrease and the expected length of an increase and a decrease period suffice for an adequate approximation. The precise cause of those fluctuations is not important. This fact is

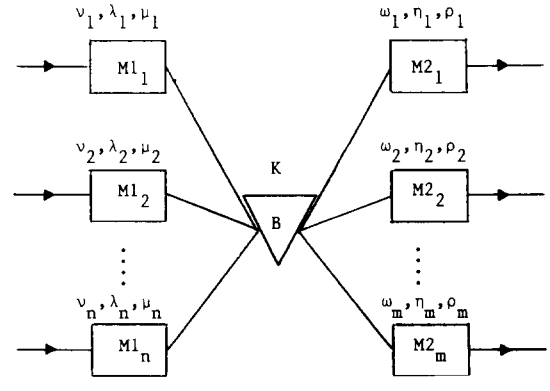

Figure 4. n parallel machines linked with m parallel machines

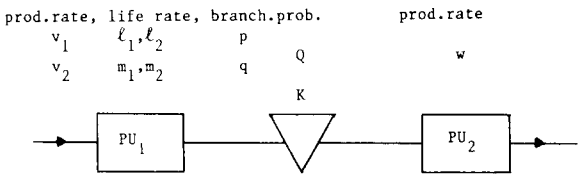

Figure 5. Two-stage line approximating the line of Figure 4

now used to approximate the line of Figure 4 by the two-stage line sketched in Figure 5. The moments of the buffer fluctuations can be calculated relatively easy for n,m not too large (calculation of the line throughput and buffer content distribution is computationally much harder). For the line of Figure 4 we only have to solve some systems of $O(2^{n+m})$ linear equations of the type $Ax = b$, for some vectors b.

PU_2 in the line of Figure 5 is completely reliable with production rate w. PU_1 has only two different speeds, $v_1 > w$ and $v_2 < w$. Both the v_1-interval and the v_2-interval of PU_1 have two-stage Coxian distributions, as sketched in Figure 6, with branching probability p and q respectively. Each stage in the Coxian distribution of Figure 6 has exponential duration.

The transitions between the four machine states (v_1, v_1, v_2, v_2) of PU_1 in Figure 5 are determined by the transition matrix Q which has the following form

$$Q = \begin{bmatrix} 0 & p & 1-p & 0 \\ 0 & 0 & 1 & 0 \\ 1-q & 0 & 0 & q \\ 1 & 0 & 0 & 0 \end{bmatrix}.$$

The 9 parameters in the line of Figure 5 are now chosen so that the first and second moment of the buffer increase (T) and the buffer decrease (S) and the expected length of such an increase (L) and a decrease period (M) of the lines of Figures 4 and 5 are equal.

How this choice has to be made is explained in [9]. In the determination of the 9 parameters we have only 6 binding equations. Therefore some freedom is left. An extra degree of freedom is used by choosing w (the production rate of PU_2) equal to the net average production rate of PU_2 in the line of Figure 4.

Having chosen the parameters of the line of Figure 5 in this way it can be proven that the net average production rates of PU_1 of the lines of Figures 4 and 5 are equal. Furthermore $v(0) = v_{ap}(0)$ and $v(\infty) = v_{ap}(\infty)$, where $v(K)$ is the throughput of the line of Figure 4 with buffer capacity K and $v_{ap}(K)$ is the throughput of the approximating line of Figure 5.

The aim of the approximation is that the resulting line of Figure 5 is easy to analyse. The line has only 4 states and hence the throughput, buffer content distribution, blocking and starvation probabilities, and various other input and output parameters can be calculated by the method described by Wijngaard [23] in short time with standard subroutines.

3.2. Aggregation over two subsequent PU's

The second approximation step needed in the approximation of the system is the aggregation over two subsequent PU's and an intermediate buffer of capacity K. There are two types of two-stage lines that need to be aggregated. They are denoted by 'line a' and 'line b', respectively. Line a is sketched in Figure 5 and line b is sketched in Figure 7.

The aggregation for both line a and line b is similar. They are approximated by a single PU with two states and two different speeds ω_1 and ω_2 $(\omega_1 > \omega_2 \geq 0)$. A speed ω_i-interval is exponentially distributed with parameter ρ_i. If we want to approximate the input behaviour of line a or b, then ω_2 equals the lowest possible input speed of the line. (The lowest possible input speed of line a is v_2, and $\min\{\nu_2, \Phi_1, \Phi_2\}$ for line b.) ω_1 equals the conditional expected input speed of line a or b

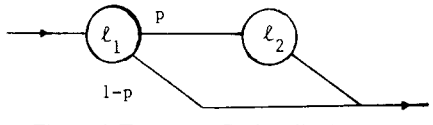

Figure 6. Two-stage Coxian distribution

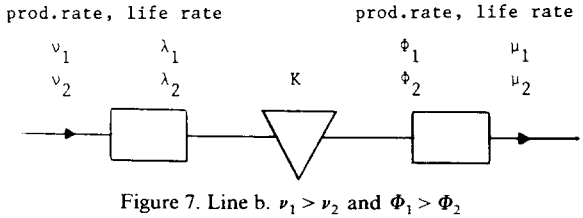

Figure 7. Line b. $\nu_1 > \nu_2$ and $\Phi_1 > \Phi_2$

given that this speed is greater than ω_2. If we want to approximate the output behaviour, then ω_1 and ω_2 are chosen similarly. $1/\rho_2$ is the average duration of an interval of speed ω_2 and $1/\rho_1$ is the average duration of an interval of speed greater than ω_2. These four parameters can be obtained both for line a and line b similarly as in [10] or [11]. Furthermore it is easy to calculate the average buffer content of lines a and b. The CPU time needed for the calculation of these parameters depends on the buffer size and on the required precision but is even for large buffer size less than a second on a Burroughs B7900 computer when using standard subroutines of the NAG-library.

3.3. *Approximation of flow lines*

By repeatedly applying the approximating steps described in Subsections 3.1 and 3.2 it is possible to reduce each allowed network to a flow line where each PU may have more than two states. It is now possible by again repeatedly applying step 1 and step 2 (of Subsections 3.1 and 3.2, respectively) to reduce this line to a two-stage line with an intermediate buffer of finite capacity. How the subsequent approximation steps can be carried out precisely will become clear in the next section.

The approximation of the flow line can be carried out in different ways. We may approximate the line from left to right or vice versa. In total there are $(N-1)!$ different aggregation sequences possible for an N-stage flow line.

If the flow line consists of more than three PU's the best results are achieved when the capacities of the intermediate buffers in this reduction are adapted. The approximation by adapting the buffer contents is an improved version of the algorithm described in [10]. It is described in [11]. If the resulting flow line consists of exactly three PU's we can either adapt the buffer capacities as in [11], or not. For the networks of this paper the method of [11] should be followed if there is a high negative correlation between the buffer contents, especially if it occurs often that the first buffer is empty and the second full. In the examples of the next section for all resulting flow lines which have three PU's, the buffer capacities are not adapted. For the first example given in Section 4, the buffer capacities are adapted, in concordance with [11].

4. Numerical results

In this section the approximation technique will be tested on four different production systems for varying buffer capacities. The systems have an increasing complexity. The first system is a flow line, consisting of four machines and three buffers, the second and third system consist of six machines and three buffers, the fourth system consists of nine machines and four buffers. Except for the first system all machines involved have exactly two states: an up-state with speed ν_i and failure rate λ_i, and a down-state with speed 0 and repair rate μ_i for machine M_i. In the first system M_1 has five states, M_2 two, M_3 four and M_4 has also four states. The lay-out of this system and the machine parameters (ν_j^i, λ_j^i and transition matrix P^i, for M_i) are given in Figure 8. The net average machine speeds are also given. For the systems 2, 3 and 4 the net average machine speed equals $\nu\mu/(\lambda+\mu)$, for the machines of the first system the expression for the net machine speed is somewhat more complex. State 5 of machine M_1 in the first system can be interpreted as a state in which the machine is switched over, since after a stay in each other state $i \neq 5$, the transition probability $p_{i5} = 1$. The first system is rather unbalanced, machine M_4 is a bottleneck, with a net average production rate of 0.8457. The second and third system are also unbalanced, the fourth system is rather balanced.

All systems were simulated on the B7900 using 10 runs of 500 000 units of time per run. Especially for the largest system this took considerable amounts of CPU time.

To approximate the system of Figure 8 by a two-stage line, the system was approximated from left to right. Approximation from right to left gives similar results. First machines M_1 and M_2 were aggregated, using step 1 and step 2, the parameters of the aggregate PU being determined by the input behaviour. This results in a three-stage line $PU_1 - B - M_3 - B_3 - M_4$, where PU_1 is the aggregate PU, with two states, B is a buffer with adapted capacity $K_2 + (K_1 - E\,X_1)$ and M_3, M_4 and B_3 are machines and buffer from the original system. X_1 is the content of the first buffer in the line $M_1 - B_1 - M_2$ and E is the expectation operator. $K_1 - \mathrm{E}\,\mathrm{X}_1$ is the average storage space left in B_1 as seen by PU_1 (which represents the input behaviour of $M_1 - B_1 - M_2$). A discussion on buffer

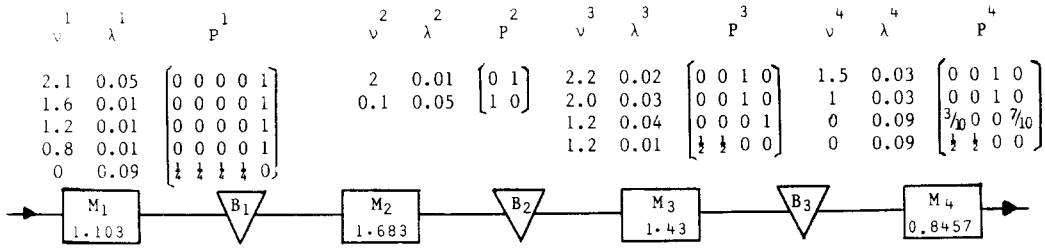

Figure 8. Production system 1

capacity adaptation can be found in [11]. PU_1 and M_3 are now approximated with respect to output behaviour, using steps 1 and 2, by PU_2. PU_2 has two states. The throughput of the system is now approximated by the throughput of the line PU_2–B_3–M_4. In order to obtain the throughput of PU_2–B_3–M_4 we again applied step 1 and step 2, although we could also have obtained the throughput of PU_2–B_3–M_4 immediately, since the line has only 8 ($= 2 \cdot 4$) machine states. This extra approximation, using step 1, is always applied when the two-stage line to be aggregated does not fit immediately in one of the models of Figures 5 or 7. For various values of the buffer capacities the results are given in Table 1. In Table 1 the approximate throughput is denoted by $v_{ap}(K)$. Relative errors (in %) are also given.

The results of Table 1 are acceptable. The lay-out of the second system is given in Figure 9. As in Figure 8 the machine parameters (ν, λ, μ) and the net average machine speeds of the machines are indicated. In order to approximate system 2 by a flow line we first aggregate machines M_2 and M_3, using step 2, the parameters of the aggregate PU being determined by the output behaviour. The result is a three-stage line PU_1–B_2–M_4–B_3–PU_2, where PU_1 has four states and PU_2 has three states (namely the number of 'up' machines). The resulting flow line is now approximated from the left and from the right for various buffer sizes using steps 1 and 2. For the approximation from the left first steps 1 and 2 are applied subsequently on PU_1–B_2–M_4. Denote the resulting aggregate PU by PU_3. In order to obtain the throughput of PU_3–B_3–PU_2 we again applied step 1 and step 2. In Table 2 the results are given.

Table 1
Approximations of system 1

K_1	K_2	K_3	$v(K)$	$v_{ap}(K_3)$	Rel. err. (%)
5	5	5	0.6355	0.6437	−1.29
10	5	10	0.6821	0.6895	−1.08
15	10	15	0.7291	0.7250	0.56
15	20	20	0.7610	0.7492	1.55
15	30	35	0.7917	0.7875	0.53
∞	∞	∞	0.8457		

From Table 2 we see that the approximation both from the left and from the right performs well. Relative errors are in all cases below 1%.

The third example network is sketched in Figure 10. Machine M_1 is perfect with speed 0.9. Here the bottleneck of the line is constituted by M_6 with also an average speed of 0.9, but with a greater variance in speed than M_1.

Production system 3 can be approximated by a flow line two-stage by first aggregating over M_3, M_4, B_2 and M_5 using steps 1 and 2. The resulting aggregate PU is denoted by PU_1. Depending on whether we want to approximate this resulting flow line from the left or from the right we may

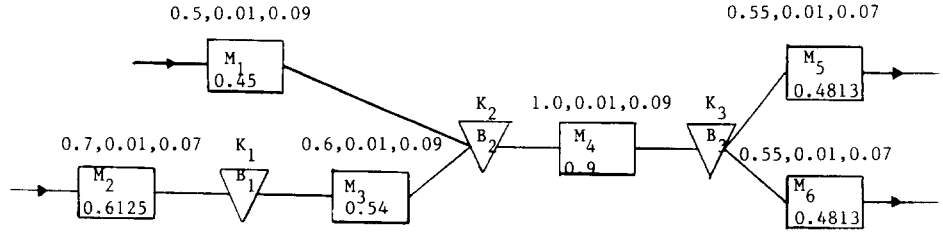

Figure 9. Production system 2

Table 2
Approximations of system 2

K_1	K_2	K_3	$v(K)$	Left		Right	
				$v_{ap}(K_3)$	rel. err. (%)	$v_{ap}(K_2)$	rel. err.
4	6	6	0.7983	0.8006	−0.29	0.8028	−0.56
5	10	10	0.8246	0.8262	−0.19	0.8278	−0.39
10	10	10	0.8311	0.8331	−0.24	0.8330	−0.23
10	15	20	0.8558	0.8575	−0.20	0.8566	−0.09
10	20	15	0.8551	0.8565	−0.16	0.8568	−0.20
∞	∞	∞	0.9				

Table 3
Approximations of system 3

K_1	K_2	K_3	$v(K)$	Left		Right	
				$v_{ap}(K_3)$	rel. err.	$v_{ap}(K_1)$	rel. err.
5	3	5	0.8261	0.8199	0.75	0.8648	−4.68
10	6	10	0.8530	0.8444	1.01	0.8770	−2.81
10	10	10	0.8570	0.8366	2.38	0.8777	−2.42
15	10	10	0.8612	0.8419	2.24	0.8806	−2.25
10	10	15	0.8635	0.8462	2.00	0.8798	−1.89
∞	∞	∞	0.9				

we may use the output or the input behaviour, respectively, in step 2. The result is a flow line M_1–B_1–PU_2–B_3–M_6, where PU_2 has four states, arising from the combination of PU_1 and M_2. The results for the approximation of the flow line both from the left and from the right are given, for various buffer sizes, in Table 3.

It appears from Table 3 that the approximations perform less for this system than for systems 1 and 2. The reasons for this are not clear yet but it has appeared that the more resemblance there is between the layout of the system and a flow line, the better the performance of the approximation. The bypass in this network may have complicated effects on the performance of M_1 and M_6.

The last production system studied in this section is sketched in Figure 11.

Production system 4 can be approximated by a two-stage flow line PU_1–B_4–M_9, where PU_1 has four states and arises from the combined aggregation of the left upper branch and the left lower branch of the system. The left upper branch is approximated from the left to the right since we are interested in its output behaviour. (However, approximation from the right to the left gives similar results.) The results are given in Table 4.

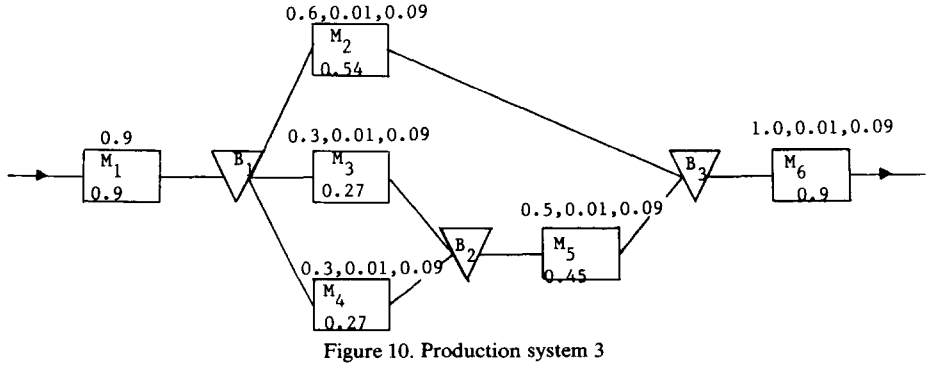

Figure 10. Production system 3

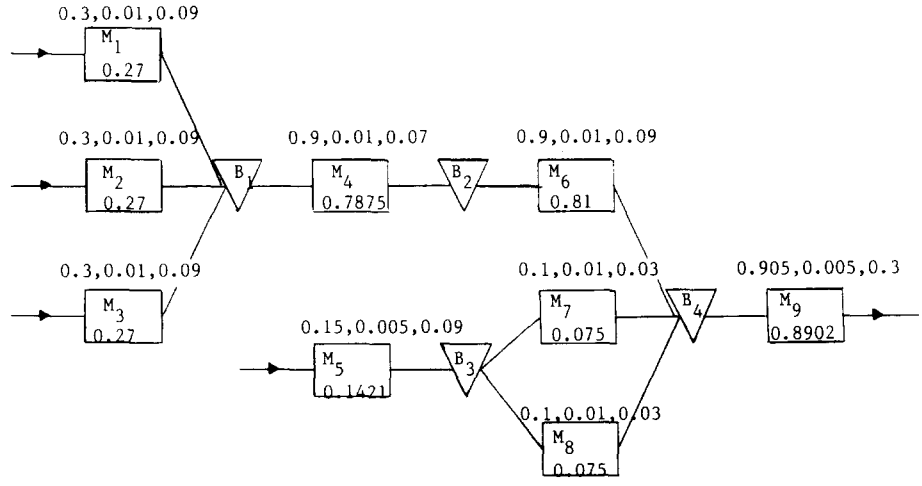

Figure 11. Production system 4

The results of Table 4 are acceptable but also not very good. The approximation appears to perform best for systems which show some resemblance with flow lines (systems 1 and 2). The intermediate approximation step was always applied in the aggregation of a line as in Figure 4, although the number of states involved in the example networks is not so large that this was really necessary, strictly spoken. However this extra approximation is very accurate, when used isolated, and probably has little influence on the results. In production systems 1 and 2 this step apparently has no influence. What seems to have more influence on the quality of the approximation is the used priority rule. The approximation method is insensitive to the used priority in allocating the speed reduction to the machines in case of slow down. For instance if the left lower branch in production system 4 is given priority over the left upper branch if B_4 is full, then the throughput of the system would probably decrease, but the approximate throughput would be the same as in Table 4. Also if other priority rules were used the approximation method would yield the same results. In production systems 1 and 2 the priority rule has hardly any influence. It is not clear therefore whether the poor performance of the approximation method for systems 3 and 4, is due to the fact that systems 3 and 4 have complicated structure (with a bypass and independent input streams, respectively) or to the insensitivity to the used priority. This is a topic for future study.

Table 4
Approximations of system 4

K_1	K_2	K_3	K_4	$v(K)$	$v_{ap}(K_4)$	rel. err.
5	3	3	5	0.7733	0.8067	−3.88
10	6	6	10	0.8078	0.8343	−3.28
10	10	10	10	0.8147	0.8426	−3.42
15	10	10	10	0.8176	0.8461	−3.49
10	10	10	15	0.8265	0.8473	−2.52
15	6	6	15	0.8225	0.8433	−2.33
20	20	10	15	0.8397	0.8598	−2.39
∞	∞	∞	∞	0.8902		

5. Conclusions

In this paper networks of machines and buffers are approximated for the system throughput. The approximation algorithm consists of the combined algorithms for the approximation of flow lines consisting of two-state machines (see [10] and [11]) and the approximation of two multi-state machines with intermediate buffer. The approximation performs well for systems which resemble flow lines. For more complex systems the method works, but needs to be refined in order to obtain better results. The approximate throughput is independent of the priority used in allocating the speed reduction to the machines in case of slow down. Some future research should be devoted to networks of this type, but also to networks where assembly and disassembly of different products takes place, or where loops occur, due to the reprocessing of failed products.

Appendix

In this appendix we will show, for networks of machines as in the paper, connected with intermediate finite buffers, a continuous product flow, and with priority rules for machines sharing the same buffer as defined in Section 2, that the machine speeds due to influences from the network can be determined at all points in time and that they are unique. An efficient method will be given to determine the speeds in the network (based on dynamic programming). It is then easy to see that in the reversed network the speeds are the same for all possible realisations of the process generating the stand alone machine speeds and furthermore a buffer content X_i corresponds to a buffer content $Y_i = K_i - X_i$ in the reversed network, provided this is so initially.

From these facts it is then straightforward to prove that for instance the troughput of a network and its reverse are the same.

In order to determine the speeds of the machines of an allowed network we define *source buffers* as buffers that are never empty. These buffers precede source machines. Similarly *sink buffers* are buffers that are never full. Sink buffer succeed sink machines. Note that in this way each machine has exactly one upstream and one downstream buffer. We define the following additional variables. With each buffer b_1 that has downstream machines there is associated an output rate restriction $o(b_1)$, which is the maximum possible output rate of products from this buffer. If b_1 is non-empty then $o(b_1) = \infty$. With each buffer b_2 that has upstream machines there is associated an input rate restriction $i(b_2)$, which is the maximum possible input rate into this buffer. If b_2 is non-full then $i(b_2) = \infty$.

We suppose that the network is connected. If not we can treat each connected component individually.

The machine speeds only have to be determined at change points, points in time where either a buffer becomes full or empty, or starts filling after having been empty, or starts empying after having been full. In between change points the machine speeds remain constant. The algorithm locates a machine of which the speed can be calculated and removes this machine with every connecting arc from the network. Non-connected buffers are also removed and this procedure is repeated till all machine speeds have been determined. At each removal one or more networks remain which are of the same type as the original network. In the algorithm, Q denotes the set of buffers, v_i the instantaneous machine speed of M_i and w_i the speed of M_i as influenced by the network. pred_i is the buffer preceding M_i and succ_i is the buffer succeeding M_i.

Conditions (a) and (c) of Section 2 guarantee that the algorithm always takes a new machine and a new buffer while going downstream or upstream through the network. The finiteness of the network guarantees that always a machine can be found of which the speed can be determined. If the remaining network is disconnected then the components are treated one by one.

For production system 3 of Section 4 the algorithm gives the following results, in case B_1 and B_3 are full and B_2 is empty. Denote the source buffer (preceding M_1) by B_0, the sink buffer (succeeding M_6) by B_4.

Initialization:

$$o(B_0) = i(B_4) = o(B_1) = i(B_2) = o(B_3) = \infty,$$

$$i(B_1) = o(B_2) = i(B_3) = 0.$$

Machine speed determination: $w_6 = 1$; $i(B_3) = 1$; $w_2 = 0.6$; $i(B_1) = 0.6$; $i(B_3) = 0.4$; $w_3 = 0.3$; $o(B_2) = 0.3$; $i(B_1) = 0.9$; $w_4 = 0.3$; $o(B_2) = 0.6$; $i(B_1) = 1.2$; now we have two components left, namely B_0–M_1–B_1 and B_2–M_5–B_3. $w_1 = 0.9$; $i(B_1) = 0.3$; $w_5 = 0.4$.

References

[1] Altiok, T., "Approximate analysis of exponential tandem queues with blocking", *European Journal of Operational Research* 11 (1982) 390–398.

[2] Altiok, T., and Perros, H.G., "Approximate analysis of arbitrary configurations of open networks with blocking", *Annals of Operations Research* 9 (1987) 481–509.

[3] Altiok, T., and Perros H.G., "Open networks of queues with blocking: Split and merge configurations", *IIE Transactions*, 18 (1986) 251–261.

[4] Ammar, M.H., and Gershwin, S.B., "Equivalence relations in queueing models of manufacturing" *Proceedings of the Nineteenth IEEE Conference on Decision and Control*, 1981.

[5] Boxma, O.J., and Konheim, A.G., "Approximate analysis of exponential queueing systems with blocking", *Acta Informatica* 15 (1981) 19–66.

[6] Brandwajn, A., and Yow, Y.L., "An approximation method for tandem queues with blocking", Amdahl Corp., Tech. Rep., 1985.

Algorithm
initialization for all full buffers b: $i(b) := 0$;
for all empty buffers b: $o(b) := 0$;
while $Q \neq \emptyset$
do found: = false;
choose source buffer a;
while not found
do while a is not a sinkbuffer **and not** found
do go downstream along the arc with the highest priority to machine i and further downstream to buffer b;
if b is not full
then found: = true;
$w_i := \min\{v_i, o(a)\}$
else $a := b$
fi
od;
if a has no other incoming arcs with a higher priority than the arc along which a was entered **and not** found
then found: = true;
$w_i := \min\{v_i, i(a), o(\text{pred}_i)\}$
fi;
while a not a source buffer **and not** found
do go upstream along the arc with the highest priority to machine i and further to buffer b
if b is not empty
then found: = true;
$w_i := \min\{v_i, i(a)\}$
else a: $= b$
fi
od;
if a has no other outgoing arcs with a higher priority than the arc along which a was entered **and not** found
then found: = true;
$w_i := \min\{v_i, o(a), i(\text{succ}_i)\}$
fi;
od;
remove i from the network with all its arcs and non-connected buffers;
if i obtained from an empty buffer b
then $o(b) := o(b) - w_i$ **fi**;
if i supplied an emtpy buffer b
then $o(b) := o(b) + w_i$ **fi**;
if i obtained from a full buffer b
then $i(b) := i(b) + w_i$ **fi**;
if i supplied a full buffer b
then $i(b) := i(b) - w_i$
fi
od

[7] Choong, Y.F., and S.B. Gershwin, "A decomposition method for the approximate evaluation of capacitated transfer lines with unreliable machines and random processing times", *IIE Transactions* 19 (1987) 150–159.

[8] Corten, Maurice and Koster, René de, "On the reversibility of manufacturing networks", manuscript in preparation.

[9] De Koster, M.B.M., "Capacity analysis of two-stage production lines with many products", *Engineering Costs and Production Economics* 12 (1987) 175–186.

[10] De Koster, M.B.M., "Estimation of line efficiency by aggregation", *International Journal of Production Research* 25 (1987) 615–626.

[11] De Koster, M.B.M., "An improved algorithm to approximate the behaviour of flow lines", to appear in *International Journal of Production Research*.

[12] De Koster, M.B.M., "Approximations of flow lines with integrally controlled buffers", *IIE Transactions*, to be published.

[13] De Koster, M.B.M., "Approximation of assembly-disassembly systems", report BDK/ORS/87/02, Eindhoven University of Technology, 1987.

[14] Glassey, C.R., and Hong, Y., "The analysis of behavior of an unreliable N-stage automatic transfer line with $(N-1)$ inter-stage buffer storages", Dept. of IE and OR Tech. Rep. ORC-86-10, University of California, Berkeley, 1986.

[15] Hillier, F.S., and Boling, R., "Finite queues in series with exponential or Erlang service times — A numerical approach", *Operations Research* 15 (1967) 286–303.

[16] Muth, E.J., "The reversibility property of production lines", *Management Science* 25 (1979) 152–158.

[17] Onvural, R.O., and Perros, H.G., "Some exact results on

closed queueing networks with blocking", report North Carolina State University, 1986.

[18] Perros, H.G., and Altiok, T., "Approximate analysis of open networks with blocking: Tandem configurations", *IEEE Transactions on Software Engineering* 12 (1986) 450–461.

[19] Perros, H.G., and Snyder, P.M., "A computationally efficient algorithm for analyzing open queueing networks with blocking", report North Carolina State University, 1986.

[20] Pollock, S.M., Birge, J.R., and Alden, J.M., "Approximation analysis for open tandem queues with blocking: Exponential and general service distributions", University of Michigan, Ann Arbor, 1986.

[21] Suri, R., and Diehl, G.W., "A variable buffer-size model and its use in analyzing queueing networks with blocking", *Management Science* 32 (1986) 206–224.

[22] Takahashi, Y., Miyahara, H., and Hasegawa, T., "An approximation method for open restricted queueing networks", *Operations Research* 28, (1980) 594–602.

[23] Wijngaard, J., "The effect of interstage buffer storage on the output of two unreliable production units in series, with different production rates", *AIIE Transactions* 11 (1979) 42–47.

[24] Yamazaki, G., Kawashima, T., and Sakasegawa, H., "Reversibility of tandem blocking queueing systems", *Management Science* 31 (1985) 78–83.