In this study, the buffer allocation problem (BAP) in homogeneous, asymptotically reliable serial production lines is considered. A known aggregation method, given by Lim, Meerkov, and Top (1990), for the performance evaluation (i.e., estimation of throughput) of this type of production lines when the buffer allocation is known, is used as an evaluative method in conjunction with a newly developed dynamic programming (DP) algorithm for the BAP. The proposed algorithm is applied to production lines where the number of machines is varying from four up to a hundred machines. The proposed algorithm is fast because it reduces the volume of computations by rejecting allocations that do not lead to maximization of the line’s throughput. Numerical results are also given for large production lines.

1. Introduction and literature review

The analysis of production or flow lines and, more generally, of manufacturing systems has been the object of numerous studies. Usually, production lines are modeled as serial queuing networks and analyzed either analytically as Markovian models or via approximate decomposition methods and simulation.

The literature on the modeling of production lines is vast and a large amount of research over the years has been devoted to this area. One of the first most complete analyses of such systems was published in 1962 by Sevastyanov [20]. The large amount of research allows us to review only the most directly relevant studies here.

For a systematic classification of the relevant works on the stochastic modeling of production lines, the interested reader is referred to the books by Buzacott and Shanthikumar [3], Papadopoulos et al. [17], Gershwin [7], Altiok [2], and Helber [10], and the review papers by Dallery and Gershwin [5] and Papadopoulos and Heavey [16], among others.

Hillier and Boling [11] and Heavey et al. [9], analyzed serial production lines using Markovian models, whereas Gershwin [6] and Dallery et al. [4] utilized decomposition approaches to evaluate the performance of the same type of production lines. The former method is practically applicable only to short lines due to the curse of the dimensionality
problem arising in the Markovian models. The latter method can be applied to large production lines at the expense of the accuracy of the numerical results. Altiok [2], among others, studied the phase-type distributions and their use to approximate any general distribution of service or interarrival times in the stochastic modeling of production lines.

Meerkov and his colleagues have developed performability, a quite interesting theory and method of analysis of production lines. Lim et al. [14] presented an asymptotic analysis technique for a model of a serial production line, gave estimates of its accuracy, and formulated a convergence theorem for their solution algorithm.

One of the most interesting questions that designers face in a serial production line is the buffer allocation problem (BAP), that is, how much buffer storage to allow and where to place it within the line. This is an important question because buffers can have a great impact on the efficiency of the production line. Buffers compensate for the blocking and the starving of the line’s stations. However, buffer storage is expensive both due to its direct cost and due to the increase of the work-in-process (WIP) inventories. In [17], both evaluative and generative (optimization) models are given for modeling the various types of manufacturing systems.

The BAP has been solved via various techniques such as simulated annealing [21, 22], genetic algorithms [15], cross entropy search method [1], gradient techniques [8], heuristics [18], and other search methods (tabu search, among others).

One of the optimization methods for solving the BAP is the dynamic programming (DP) method. Several authors have employed the DP method for solving the BAP (see Jafari and Shanthikumar [12], Kubat and Sumita [13], and Yamashita and Altiok [23], among others). However, this method was employed in the case of a production line with synchronous transfer as defined in [17], among others, where the steady-state throughput can be approximated in a closed recursive form. Another classification of the research work relevant to the BAP is based on whether the lines under study are balanced or unbalanced. Powell [19] provided a literature review according to this scheme.

In this paper, a DP algorithm for the BAP is developed. This algorithm makes use of an aggregation procedure to approximate various performance measures of the production line developed by Lim et al. [14].

The remainder of this paper is structured as follows. Section 2 presents the problem and the model. Section 3 presents the proposed DP algorithm with an application example. Section 4 gives numerical results for several production lines, both short and large, and Section 5 concludes the paper and proposes some areas for further research. Finally, the appendix gives the aggregation method for the approximation of throughput for a given buffer allocation, as taken by Lim et al. [14].

2. The model and the problem definition

The main assumptions of our model are the same as those made by Lim et al. [14], as follows.

1. A serial production line (see Figure 2.1) consists of \( M \) machines \( M_i, i = 1, \ldots, M \), and \( M - 1 \) buffers \( B_i \) of finite capacity at least one (this assumption is dictated by
A production line with $M$ machines and $M - 1$ buffers.

(A.1), as otherwise function $Q(a, N)$ is not defined). Assuming that each buffer stores at least 1 unit and $N$ is the total buffer capacity that can be allocated, it is obvious that for a production line consisting of $M$ machines and $M - 1$ buffers, the maximum buffer capacity that each buffer can store is up to $N - M + 2$ units.

(2) An inexhaustive supply of workpieces is available upstream machine $M_1$, and an unlimited storage area is present downstream machine $M_M$. Thus the first machine is never starved and the last machine is never blocked.

(3) All machines have equal and constant service times. Time is scaled so that this machine cycle takes one time unit. Thus processing times are assumed to be deterministic and identical for all machines and are taken as the time unit.

(4) Machine $M_i$, being not blocked and not starved, takes part in production during any time slot with probability $\delta_i$ and fails to do so with probability $1 - \delta_i$ (production lines in which machines have this property are called homogeneous by Lim et al. [14], that is, a homogeneous line is characterized by machines with one parameter only, $\delta_i$, instead of the two parameters $p_i$ and $r_i$ usually used in the literature, where $p_i$ and $r_i$ denote, respectively, the failure rate and the repair rate of machine $M_i$). It is assumed that blocked and/or starved machines do not fail.

(5) It is assumed that the production lines under consideration are homogeneous, asymptotically reliable, that is, $\delta_i = 1 - \epsilon \Lambda_i$, where $0 < \epsilon \ll 1$ and $\Lambda_i$, $i = 1,\ldots,M$, is independent of $\epsilon$. The $\Lambda_i$’s are known as the loss parameters, as defined by Lim et al. [14].

Following the lines of Lim et al. [14], denote by $L_i$ the cumulative losses in the $i$th operation (jobs/h) and by $AX_i$ the actual throughput, then $\delta_i = 1 - L_i/AX_i$, and since $\delta_i = 1 - \epsilon \Lambda_i$, where $0 < \epsilon \ll 1$, it is obvious that $L_i \ll AX_i$. Thus the loss parameter $\Lambda_i$ refers to the fraction of the cumulative losses in the $i$th operation divided by the actual throughput of machine $M_i$.

Major decisions in designing production lines involve the workload allocation and the BAPs with respect to an objective function such as profit maximization or throughput maximization for a given total buffer capacity. In this study, the latter has been chosen to deal with; namely, our objective is to find the optimal buffer allocation for a given total buffer capacity in order to maximize the average production rate of the production line. The above problem may be expressed mathematically as follows.

Find the optimal vector $B = (N_1,\ldots,N_{M-1})$ that maximizes $\{X_M\}$ given that $\sum_{j=1}^{M-1} N_j = N$, where $X_M$ is the mean production rate (throughput) of a production line consisting of $M$ machines and $M - 1$ buffers. $N$ is the total buffer capacity and $N_i$ is the capacity of buffer $i$, $i = 1,\ldots,M - 1$. 
3. The dynamic programming algorithm

Before expressing the DP algorithm mathematically, we introduce the following symbols.

$Z_i$ is the buffer capacity that the DP algorithm allocates to buffer $i$ and to all buffers upstream buffer $i$. Therefore $Z_i = N_1 + \cdots + N_i$, for all $i \geq 2$. It is obvious that $Z_1 = N_1$ and that $Z_{M-1} = N_1 + \cdots + N_{M-1} = N$.

$\Lambda^f_j(N_{j-1})$ is the value of the aggregated loss parameter $\Lambda^f_j$, defined in the appendix, when buffer space $N_{j-1}$ is allocated to buffer $j-1$, where $N_{j-1} = 1, 2, \ldots, N - M + 2, j = 2, \ldots, M$. Equation (A.14) in the appendix indicates that parameter $\Lambda^f_j(N_{j-1})$ is used in order to calculate parameter $\Lambda^f_j$. From the various values of parameter $\Lambda^f_j$, that one with the maximum value is selected.

$f_1(Z_1)$ is the set of feasible values of parameter $\Lambda^f_2(N_1), N_1 = 1, \ldots, N - M + 2$, that are used to calculate $f_2(Z_2)$.

The DP algorithm consists of four steps which are summarized below.

Step 1. Calculate the forward pass loss parameters $\Lambda^f_i(N_{i-1}), i = 2, \ldots, M$, $N_{i-1} = 1, \ldots, N - M + 2$, by using expression (A.9) of Lim et al. [14].

Step 2. The fundamental recursion equation. Execute the following recursive equations:

$$f_j(Z_j) = \min_{\text{feasible values } N_j, f_{j-1}(Z_{j-1})} \{ \Lambda^f_{j+1}(N_j) + f_{j-1}(Z_{j-1}) \},$$

$$j = 2, \ldots, M - 1, \quad N_j = 1, \ldots, N - M + 2,$$

$$Z_j = j, \ldots, N - M + 1, \quad Z_{M-1} = N,$$

$$Z_{j-1} = Z_j - N_j. \quad (3.1)$$

Step 3. Termination of the algorithm. The algorithm terminates when the value of $f_{M-1}(Z_{M-1})$ is calculated.

Step 4. Determination of the optimal buffer allocation. The optimal buffer allocation is given by vector $(T_1, T_2, \ldots, T_{M-1})$, with its elements obtained as follows.

1. Allocate $T_{M-1}$ units of buffer space to buffer $B_{M-1}$, where $T_{M-1}$ is the value for which $f_{M-1}(N)$ is obtained.
2. Allocate $T_{M-2}$ units of buffer space to buffer $B_{M-2}$, where $T_{M-2}$ is the value for which $f_{M-2}(N - T_{M-1})$ is obtained.
3. Allocate $T_{M-3}$ units of buffer space to buffer $B_{M-3}$, where $T_{M-3}$ is the value for which $f_{M-3}(N - T_{M-1} - T_{M-2})$ is obtained, and so forth.
4. Allocate $T_2$ and $T_1$ units of buffer capacity to buffers $B_2$ and $B_1$, respectively, where $T_2$ and $T_1$ are the values for which $f_2(N - (T_{M-1} + T_{M-2} + \cdots + T_3))$ is obtained.

3.1. Application of the algorithm: an example. In this section, an application of the algorithm is given for a production line with 4 machines and loss parameters $\Lambda_1 = 3.4$, $\Lambda_2 = 2.1$, $\Lambda_3 = 4.3$, and $\Lambda_4 = 1.1$, respectively. The total buffer capacity that is to be allocated is 10 units.
### Table 3.1. Terms $\Lambda_f^j(N_{j-1})$, $j = 2, 3, 4$, calculated as described in Step 1.

<table>
<thead>
<tr>
<th>Buffer 1</th>
<th>Buffer 2</th>
<th>Buffer 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_1$</td>
<td>$\Lambda_2^f(N_1)$</td>
<td>$N_2$</td>
</tr>
<tr>
<td>1</td>
<td>5.5</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>4.2021</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3.8008</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>3.6215</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>3.5285</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>3.4765</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>3.4463</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>3.4283</td>
<td>8</td>
</tr>
</tbody>
</table>

**Step 1.** In this step, we calculate the forward pass loss parameters $\Lambda_f^j(N_{j-1})$, $i = 2, \ldots, 4$, $N_{i-1} = 1, \ldots, 8$. These parameters are presented in Table 3.1 and will be used throughout the algorithm.

**Step 2.** In DP, computations are carried out in stages by breaking down the problem into subproblems. Each subproblem is then considered separately with the objective of reducing the volume of computations. However, since the subproblems are interdependent, a procedure must be devised to link the computations in a manner that guarantees that a feasible solution for each stage is also feasible for the entire problem.

\[ j = 2, \quad Z_2 = 2, \ldots, 9. \]  

For $Z_2 = 2 = N_1 + N_2$, the only feasible values of $N_1$ and $N_2$ are $\{N_1 = 1 \text{ and } N_2 = 1\}$. As $f_1(Z_1)$, the value of parameter $\Lambda_2^f(1)$ is used and

\[ f_2(Z_2 = 2) = \min \{ \Lambda_3^f(1) + \Lambda_2^f(1) \} = \min \{9.8 + 5.5\} = 15.3 \]  

(3.3)

corresponding to $T_1 = 1$ and $T_2 = 1$.

For $Z_2 = 3 = N_1 + N_2$, the feasible values of $N_1$ and $N_2$ are $\{N_1 = 1 \text{ and } N_2 = 2\}$ or $\{N_1 = 2 \text{ and } N_2 = 1\}$. As $f_1(Z_1)$, the values of parameters $\Lambda_2^f(1)$, $\Lambda_2^f(2)$ are used and

\[ f_2(Z_2 = 3) = \min \{ \Lambda_3^f(1) + \Lambda_2^f(2), \Lambda_3^f(2) + \Lambda_2^f(1) \} \]

\[ = \min \{9.8 + 4.2021, 7.3874 + 5.5\} \]

\[ = \min \{14.0021, 12.8874\} = 12.8874 \]

(3.4)

corresponding to $T_1 = N_1 = 1$ and $T_2 = N_2 = 2$.

For $Z_2 = 4 = N_1 + N_2$, the feasible values of $N_1$ and $N_2$ are $\{N_1 = 1 \text{ and } N_2 = 3\}$, $\{N_1 = 2 \text{ and } N_2 = 2\}$, or $\{N_1 = 3 \text{ and } N_2 = 1\}$. As $f_1(Z_1)$, the values of parameters $\Lambda_2^f(1)$, $\Lambda_2^f(2)$, $\Lambda_2^f(3)$ are used and
A DP algorithm for the buffer allocation problem

\[ f_2(Z_2 = 4) = \min \{ \Lambda_1^f(1) + \Lambda_3^f(3), \Lambda_1^f(2) + \Lambda_2^f(2), \Lambda_3^f(3) + \Lambda_2^f(1) \} \]
\[ = \min \{ 9.8 + 3.8008, 7.3874 + 4.2021, 6.5986 + 5.5 \} \]  \hspace{1cm} (3.5)
\[ = \min \{ 13.6008, 11.5895, 12.0986 \} = 11.5895 \]
corresponding to \( T_1 = N_1 = 2 \) and \( T_2 = N_2 = 2 \).

For \( Z_2 = 5 = N_1 + N_2 \), the feasible values of \( N_1 \) and \( N_2 \) are \( \{ N_1 = 1 \quad \text{and} \quad N_2 = 4 \}, \{ N_1 = 2 \quad \text{and} \quad N_2 = 3 \}, \{ N_1 = 3 \quad \text{and} \quad N_2 = 2 \}, \) or \( \{ N_1 = 4 \quad \text{and} \quad N_2 = 1 \}. \) As \( f_1(Z_1) \), the values of parameters \( \Lambda_1^f(1), \Lambda_2^f(2), \Lambda_3^f(3), \Lambda_2^f(4) \) are used and

\[ f_2(Z_2 = 5) = \min \{ \Lambda_1^f(1) + \Lambda_3^f(4), \Lambda_1^f(2) + \Lambda_3^f(3), \Lambda_1^f(3) + \Lambda_2^f(2), \Lambda_3^f(4) + \Lambda_2^f(1) \} \]
\[ = \min \{ 9.8 + 3.6215, 7.3874 + 3.8008, 6.5986 + 4.2021, 6.2158 + 5.5 \} \]  \hspace{1cm} (3.6)
\[ = \min \{ 13.4215, 11.1882, 10.8007, 11.7158 \} = 10.8007 \]
corresponding to \( T_1 = N_1 = 2 \) and \( T_2 = N_2 = 3 \).

For \( Z_2 = 6 = N_1 + N_2 \), the feasible values of \( N_1 \) and \( N_2 \) are \( \{ N_1 = 1 \quad \text{and} \quad N_2 = 5 \}, \{ N_1 = 2 \quad \text{and} \quad N_2 = 4 \}, \{ N_1 = 3 \quad \text{and} \quad N_2 = 3 \}, \{ N_1 = 4 \quad \text{and} \quad N_2 = 2 \}, \) or \( \{ N_1 = 5 \quad \text{and} \quad N_2 = 1 \}. \) As \( f_1(Z_1) \), the values of parameters \( \Lambda_1^f(1), \Lambda_2^f(2), \Lambda_3^f(3), \Lambda_2^f(4), \Lambda_2^f(5) \) are used and

\[ f_2(Z_2 = 6) = \min \{ \Lambda_1^f(1) + \Lambda_3^f(5), \Lambda_1^f(2) + \Lambda_3^f(4), \Lambda_1^f(3) + \Lambda_2^f(2), \Lambda_3^f(4) + \Lambda_2^f(1) \} \]
\[ = \min \{ 9.8 + 3.5285, 7.3874 + 3.6215, 3.8008 + 6.5986, 6.2158 + 4.2021, 5.9952 + 5.5 \} \]  \hspace{1cm} (3.7)
\[ = \min \{ 13.3285, 10.0089, 10.3994, 10.4179, 11.4952 \} \]
\[ = 10.3994 \]
corresponding to \( T_1 = N_1 = 3 \) and \( T_2 = N_2 = 3 \).

For \( Z_2 = 7 = N_1 + N_2 \), the feasible values of \( N_1 \) and \( N_2 \) are \( \{ N_1 = 1 \quad \text{and} \quad N_2 = 6 \}, \{ N_1 = 2 \quad \text{and} \quad N_2 = 5 \}, \{ N_1 = 3 \quad \text{and} \quad N_2 = 4 \}, \{ N_1 = 4 \quad \text{and} \quad N_2 = 3 \}, \{ N_1 = 5 \quad \text{and} \quad N_2 = 2 \}, \) or \( \{ N_1 = 6 \quad \text{and} \quad N_2 = 1 \}. \) As \( f_1(Z_1) \), the values of parameters \( \Lambda_1^f(1), \Lambda_2^f(2), \Lambda_3^f(3), \Lambda_2^f(4), \Lambda_2^f(5), \Lambda_2^f(6) \) are used and

\[ f_2(Z_2 = 7) = \min \{ \Lambda_1^f(1) + \Lambda_3^f(6), \Lambda_1^f(2) + \Lambda_3^f(5), \Lambda_1^f(3) + \Lambda_2^f(4), \Lambda_3^f(4) + \Lambda_2^f(3), \Lambda_3^f(5) + \Lambda_2^f(2), \Lambda_3^f(6) + \Lambda_2^f(1) \} \]
\[ = \min \{ 9.8 + 3.4765, 7.3874 + 3.5285, 6.5986 + 3.6215, 6.2158 + 3.8008, 5.9952 + 4.2021, 5.8553 + 5.5 \} \]  \hspace{1cm} (3.8)
\[ = \min \{ 13.2765, 10.9159, 10.2201, 10.0166, 10.1973, 11.3553 \} \]
\[ = 10.0166 \]
corresponding to \( T_1 = N_1 = 3 \) and \( T_2 = N_2 = 4 \).
For $Z_2 = 8 = N_1 + N_2$, the feasible values of $N_1$ and $N_2$ are \{N_1 = 1 and N_2 = 7\}, \{N_1 = 2 and N_2 = 6\}, \{N_1 = 3 and N_2 = 5\}, \{N_1 = 4 and N_2 = 4\}, \{N_1 = 5 and N_2 = 3\}, \{N_1 = 6 and N_2 = 2\}, or \{N_1 = 7 and N_2 = 1\}. As $f_1(Z_1)$, the values of parameters $\Lambda_1^f(1), \Lambda_2^f(2), \Lambda_3^f(3), \Lambda_4^f(4), \Lambda_5^f(5), \Lambda_6^f(6), \Lambda_7^f(7)$ are used and

$$f_2(Z_2 = 8) = \min \{\Lambda_1^f(1) + \Lambda_2^f(7), \Lambda_2^f(2) + \Lambda_3^f(6), \Lambda_3^f(3) + \Lambda_4^f(5), \Lambda_4^f(4) + \Lambda_5^f(4), \Lambda_5^f(5) + \Lambda_6^f(3), \Lambda_6^f(6) + \Lambda_7^f(2), \Lambda_7^f(7) + \Lambda_1^f(1)\}$$


$$= \min \{13.2463, 10.8639, 10.1271, 9.8373, 9.796, 10.0574, 11.26104\}$$

$$= 9.796$$

(corresponding to $T_1 = N_1 = 3$ and $T_2 = N_2 = 5$).

For $Z_2 = 9 = N_1 + N_2$, the feasible values for $N_1$ and $N_2$ are \{N_1 = 1 and N_2 = 8\}, \{N_1 = 2 and N_2 = 7\}, \{N_1 = 3 and N_2 = 6\}, \{N_1 = 4 and N_2 = 5\}, \{N_1 = 5 and N_2 = 4\}, \{N_1 = 6 and N_2 = 3\}, \{N_1 = 7 and N_2 = 2\}, or \{N_1 = 8 and N_2 = 1\}. As $f_1(Z_1)$, the values of parameters $\Lambda_1^f(1), \Lambda_2^f(2), \Lambda_3^f(3), \Lambda_4^f(4), \Lambda_5^f(5), \Lambda_6^f(6), \Lambda_7^f(7), \Lambda_8^f(8)$ are used.

Therefore

$$f_2(Z_2 = 9) = \min \{\Lambda_1^f(1) + \Lambda_2^f(8), \Lambda_2^f(2) + \Lambda_3^f(7), \Lambda_3^f(3) + \Lambda_4^f(6), \Lambda_4^f(4) + \Lambda_5^f(5), \Lambda_5^f(5) + \Lambda_6^f(4), \Lambda_6^f(6) + \Lambda_7^f(3), \Lambda_7^f(7) + \Lambda_1^f(2), \Lambda_1^f(8) + \Lambda_2^f(1)\}$$


$$= \min \{13.2283, 10.8337, 10.0751, 9.7443, 9.6167, 9.6561, 9.96314, 11.1948\}$$

$$= 9.6167$$

(3.10)

(corresponding to $T_1 = N_1 = 4$ and $T_2 = N_2 = 5$;)

$$i = 3, \quad Z_3 = 10. \quad (3.11)$$

For $N_3 = 1, \ldots, N - M + 2 = 1, \ldots, 8$, it is straightforward that the feasible values of $f_2(Z_2)$ and $N_3$ are \{N_3 = 1 and $f_2(Z_2 = 9)$\}, \{N_3 = 2 and $f_2(Z_2 = 8)$\}, \{N_3 = 3 and $f_2(Z_2 = 7)$\}, \{N_3 = 4 and $f_2(Z_2 = 6)$\}, \{N_3 = 5 and $f_2(Z_2 = 5)$\}, \{N_3 = 6 and $f_2(Z_2 = 4)$\}, \{N_3 = 7 and $f_2(Z_2 = 3)$\}, or \{N_3 = 8 and $f_2(Z_2 = 2)$\}.

Therefore

$$f_3(Z_3 = 10) = \min \{\Lambda_4^f(1) + f_2(Z_2 = 9), \Lambda_4^f(2) + f_2(Z_2 = 8), \Lambda_4^f(3) + f_2(Z_2 = 7), \Lambda_4^f(4) + f_2(Z_2 = 6), \Lambda_4^f(5) + f_2(Z_2 = 5), \Lambda_4^f(6) + f_2(Z_2 = 4), \Lambda_4^f(7) + f_2(Z_2 = 3), \Lambda_4^f(8) + f_2(Z_2 = 2)\}.$$
corresponding to $T_3 = N_3 = 2$ and $f_2(Z_2 = 8)$.

**Step 3.** The algorithm terminates as $f_3(Z_3)$ has been calculated.

**Step 4.** From the previous calculations, notice that $f_2(Z_2 = N - T_3 = 10 - 2 = 8)$ was obtained for $T_1 = 3$ and $T_2 = 5$.

Therefore the optimal buffer allocation is given by the vector $(T_1, T_2, T_3) = (3, 5, 2)$.

### 4. Numerical results

In this section, numerical results are presented showing buffer allocations obtained using the proposed DP algorithm for four, five, six stations, and large production lines with up to a hundred stations (the latter are given in **Section 4.1**). For the short lines, by the enumeration method, all possible buffer allocations of a given total buffer capacity were tested and the optimal buffer allocation, namely, that one giving the maximum throughput, was obtained. The DP algorithm was implemented in PASCAL and in a very slow old PC486 system. Tables 4.1, 4.2, and 4.3 present the buffer allocations, for given total buffer capacities, obtained by the DP algorithm for short lines with four, five, and six stations, respectively.

**Comment.** We have applied enumeration for all total buffer capacities given in **Table 4.1**, that is, for 4 to 10 units of total buffer capacities. Comparing the results from the enumeration method with those obtained by the DP algorithm, we have found that in all cases the results were identical. Also notice that the run time is very small and lies between 0.04 and 0.19 seconds even in a slow old PC486 system. In Tables 4.2 and 4.3, for the cases where the results from the enumeration method differ from those obtained by the proposed algorithm, a percentage error has been introduced. The error has been calculated using the following formula:

$$\text{Error} = \left| \frac{X_{M,\text{enum}} - X_{M,\text{DP}}}{X_{M,\text{enum}}} \right| \times 100\%, \tag{4.1}$$

where $X_{M,\text{enum}}$ and $X_{M,\text{DP}}$ denote the throughput of the buffer configuration obtained by enumeration and the proposed DP algorithm, respectively.

**4.1. Numerical results for large production lines.** In this section, numerical results are presented, showing buffer allocations obtained using the proposed DP algorithm in production lines with many stations $M$, $10 \leq M \leq 100$. Tables 4.4, 4.5, 4.6, and 4.7 present the buffer allocations obtained by the DP algorithm for given total buffer capacities, for large production lines with ten, fifty, eighty, and one hundred stations, respectively.
Table 4.1. Application of the DP algorithm in a four-station production line with $\Lambda_1 = 3.4$, $\Lambda_2 = 2.1$, $\Lambda_3 = 4.3$, and $\Lambda_4 = 1.1$.

<table>
<thead>
<tr>
<th>$N$</th>
<th>Buffer 1</th>
<th>Buffer 2</th>
<th>Buffer 3</th>
<th>$X_4$ ($\epsilon = 0.01$)</th>
<th>Run time</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>3</td>
<td>5</td>
<td>2</td>
<td>0.9522</td>
<td>0.19 s</td>
</tr>
<tr>
<td>9</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>0.9510</td>
<td>0.17 s</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>0.9487</td>
<td>0.11 s</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>0.9452</td>
<td>0.09 s</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>0.9396</td>
<td>0.08 s</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>0.9328</td>
<td>0.06 s</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>0.9182</td>
<td>0.04 s</td>
</tr>
</tbody>
</table>

Table 4.2. Application of the DP algorithm in a five-station production line with $\Lambda_1 = 3.4$, $\Lambda_2 = 2.1$, $\Lambda_3 = 4.3$, $\Lambda_4 = 1.1$, and $\Lambda_5 = 5.5$.

<table>
<thead>
<tr>
<th>$N$</th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
<th>B4</th>
<th>$X_{5,\text{enum}}$</th>
<th>$X_{5,\text{DP}}$</th>
<th>Error</th>
<th>Run time</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>0.9359</td>
<td>0.9358</td>
<td>0.0106%</td>
<td>0.09 s</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>0.9321</td>
<td>0.9320</td>
<td>0.0107%</td>
<td>0.08 s</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>0.9269</td>
<td>0.9201</td>
<td>0.7278%</td>
<td>0.06 s</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>0.9164</td>
<td>0.9114</td>
<td>0.5370%</td>
<td>0.05 s</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>0.9009</td>
<td>0.9001</td>
<td>0.0872%</td>
<td>0.05 s</td>
</tr>
</tbody>
</table>

Table 4.3. Application of the DP algorithm in a six-station production line with $\Lambda_1 = 3.4$, $\Lambda_2 = 2.1$, $\Lambda_3 = 4.3$, $\Lambda_4 = 1.1$, $\Lambda_5 = 2.4$, and $\Lambda_6 = 3.7$.

<table>
<thead>
<tr>
<th>$N$</th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
<th>B4</th>
<th>B5</th>
<th>$X_{6,\text{enum}}$</th>
<th>$X_{6,\text{DP}}$</th>
<th>Error</th>
<th>Run time</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>0.9338</td>
<td>0.9338</td>
<td>0%</td>
<td>1.89 s</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>0.9237</td>
<td>0.9217</td>
<td>0.212%</td>
<td>0.91 s</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>0.9139</td>
<td>0.9096</td>
<td>0.471%</td>
<td>0.73 s</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>0.8969</td>
<td>0.8907</td>
<td>0.686%</td>
<td>0.49 s</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>0.8715</td>
<td>0.8589</td>
<td>1.441%</td>
<td>0.30 s</td>
</tr>
</tbody>
</table>

Table 4.4. Application of the DP algorithm in a ten-station production line with $M = 10$, $N = 20$. The production rate for this specific buffer allocation is 0.9382.

<table>
<thead>
<tr>
<th>$\Lambda_1$</th>
<th>$\Lambda_2$</th>
<th>$\Lambda_3$</th>
<th>$\Lambda_4$</th>
<th>$\Lambda_5$</th>
<th>$\Lambda_6$</th>
<th>$\Lambda_7$</th>
<th>$\Lambda_8$</th>
<th>$\Lambda_9$</th>
<th>$\Lambda_{10}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.4</td>
<td>2.1</td>
<td>4.3</td>
<td>1.1</td>
<td>2.4</td>
<td>3.7</td>
<td>1.5</td>
<td>2.3</td>
<td>2.6</td>
<td>1.4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>B1</th>
<th>B2</th>
<th>B3</th>
<th>B4</th>
<th>B5</th>
<th>B6</th>
<th>B7</th>
<th>B8</th>
<th>B9</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>0.3 s</td>
</tr>
</tbody>
</table>
Table 4.5. Application of the DP algorithm in a fifty-station production line with $M = 50$, $N = 90$. The production rate for this specific buffer allocation is 0.8651.

<table>
<thead>
<tr>
<th>B1</th>
<th>B2</th>
<th>B3</th>
<th>B4</th>
<th>B5</th>
<th>B6</th>
<th>B7</th>
<th>B8</th>
<th>B9</th>
<th>B10</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>B11 B12 B13 B14 B15 B16 B17 B18 B19 B20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>B31 B32 B33 B34 B35 B36 B37 B38 B39 B40</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>B41 B42 B43 B44 B45 B46 B47 B48 B49</td>
<td>Time</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>27 s</td>
</tr>
</tbody>
</table>

Table 4.6. Application of the algorithm in an eighty-station production line with $M = 80$, $N = 200$. The run time is 6 minutes and 0.44 second in a PC486. The production rate for this specific buffer allocation is 0.8810.

<table>
<thead>
<tr>
<th>B1</th>
<th>B2</th>
<th>B3</th>
<th>B4</th>
<th>B5</th>
<th>B6</th>
<th>B7</th>
<th>B8</th>
<th>B9</th>
<th>B10</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>8</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>B11 B12 B13 B14 B15 B16 B17 B18 B19 B20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>B31 B32 B33 B34 B35 B36 B37 B38 B39 B40</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>B41 B42 B43 B44 B45 B46 B47 B48 B49 B50</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>B51 B52 B53 B54 B55 B56 B57 B58 B59 B60</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>B61 B62 B63 B64 B65 B66 B67 B68 B69 B70</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>B71 B72 B73 B74 B75 B76 B77 B78 B79</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

Unfortunately, we cannot compare these results with those obtained from enumeration because it is impossible to use enumeration in large production lines (because of the huge number of states that should be examined).
Table 4.7. Application of the DP algorithm in a hundred-station production line with $M = 100$, $N = 400$. The run time is 24 minutes and 6.81 seconds in a PC486. The production rate for this specific buffer allocation is 0.9112.

<table>
<thead>
<tr>
<th></th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
<th>B4</th>
<th>B5</th>
<th>B6</th>
<th>B7</th>
<th>B8</th>
<th>B9</th>
<th>B10</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>14</td>
<td>5</td>
<td>7</td>
<td>8</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>B11</td>
<td>B12</td>
<td>B13</td>
<td>B14</td>
<td>B15</td>
<td>B16</td>
<td>B17</td>
<td>B18</td>
<td>B19</td>
<td>B20</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>B31</td>
<td>B32</td>
<td>B33</td>
<td>B34</td>
<td>B35</td>
<td>B36</td>
<td>B37</td>
<td>B38</td>
<td>B39</td>
<td>B40</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>B41</td>
<td>B42</td>
<td>B43</td>
<td>B44</td>
<td>B45</td>
<td>B46</td>
<td>B47</td>
<td>B48</td>
<td>B49</td>
<td>B50</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>B51</td>
<td>B52</td>
<td>B53</td>
<td>B54</td>
<td>B55</td>
<td>B56</td>
<td>B57</td>
<td>B58</td>
<td>B59</td>
<td>B60</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>B61</td>
<td>B62</td>
<td>B63</td>
<td>B64</td>
<td>B65</td>
<td>B66</td>
<td>B67</td>
<td>B68</td>
<td>B69</td>
<td>B70</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>B71</td>
<td>B72</td>
<td>B73</td>
<td>B74</td>
<td>B75</td>
<td>B76</td>
<td>B77</td>
<td>B78</td>
<td>B79</td>
<td>B80</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>B81</td>
<td>B82</td>
<td>B83</td>
<td>B84</td>
<td>B85</td>
<td>B86</td>
<td>B87</td>
<td>B88</td>
<td>B89</td>
<td>B90</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>B91</td>
<td>B92</td>
<td>B93</td>
<td>B94</td>
<td>B95</td>
<td>B96</td>
<td>B97</td>
<td>B98</td>
<td>B99</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

5. Conclusions and further research

In this study, we present a dynamic programming algorithm that solves the buffer allocation problem (BAP) of $N$ units of total buffer capacity in a homogeneous asymptotically reliable serial production line consisting of $M$ machines and $M-1$ buffers. The main conclusions are as follows.

1. The proposed dynamic programming algorithm for short (with $M < 10$ stations) production lines found, in almost all cases, the optimal solution for the BAP. In the cases where the algorithm did not give the optimal solution, it gave a near-optimal solution.

2. The algorithm is quite fast and in all cases where we applied it, we did not encounter any bugs and the algorithm always converged to a solution. The run time in all cases was quite small.

3. The DP algorithm can be applied in large production lines to effectively (rapidly and accurately) find a near-optimal solution to the BAP. Even in large systems, the proposed algorithm worked quite effectively.
A further investigation to improve the accuracy of the proposed algorithm might include the effect the backward pass search might have on the accuracy of the numerical results. However, an application of this backward pass to a few short lines showed no further improvement in the optimal buffer allocation. Another point that needs further investigation in order to improve the accuracy of the proposed algorithm might be the appropriate use of the values of the loss parameters, \( \Lambda_i \). In our numerical results, we used values analogous to those used by Lim et al. [14] for short lines.

An extension of the work presented in this paper would be the study of a production line where the general topology would be quite different from the topology of the model considered in this study. That is, a production line which consists of identical machines in parallel order at each workstation. Another optimization problem, apart from the buffer allocation, would be the server allocation as well as the workload allocation problem or, even better, the simultaneous optimization of the parameters of these design problems taken in various combinations.

Appendix

**Throughput approximation of homogeneous asymptotically reliable production lines for a given buffer allocation (taken from Lim et al. [14])**

The method for the performance evaluation of a homogeneous asymptotically reliable serial production line presented here is introduced and explicitly analyzed in [14].

Define the function

\[
Q(a, N) = \frac{1-a}{1-aN}, \quad a \in \mathbb{R}^+, \quad N \in [1, \infty]. \quad (A.1)
\]

**Two-machine lines.** A two-machine, one-buffer production line in steady state is equivalent to a single aggregated machine characterized by

\[
\delta_{\text{aggregation}} = 1 - \left[ \Lambda_2 + \Lambda_1 Q\left( \frac{\Lambda_2}{\Lambda_1}, N \right) \right] \varepsilon. \quad (A.2)
\]

Thus, the loss parameter of the equivalent aggregated machine is

\[
\Lambda_{\text{aggregation}} = \Lambda_2 + \Lambda_1 Q\left( \frac{\Lambda_2}{\Lambda_1}, N \right), \quad (A.3)
\]

where \( \Lambda_1 \) and \( \Lambda_2 \) are the loss parameters of the first and second machines, respectively. It has been proved that the mean production rate \( X_2 \) of a two-machine, one-buffer production line is given by

\[
X_2 = 1 - \left[ \Lambda_2 + \Lambda_1 Q\left( \frac{\Lambda_2}{\Lambda_1}, N \right) \right] \varepsilon + O(\varepsilon^2) \quad (A.4)
\]

\[
= 1 - \left[ \Lambda_1 + \Lambda_2 Q\left( \frac{\Lambda_1}{\Lambda_2}, N \right) \right] \varepsilon + O(\varepsilon^2).
\]
It is straightforward that
\[
\Lambda_{\text{aggregation}} = \Lambda_1 + \Lambda_2 Q\left(\frac{\Lambda_1}{\Lambda_2}, N\right).
\] (A.5)

Equations (A.3) and (A.5) show that
\[
\Lambda_{\text{aggregation}} = \Lambda_1 + \Lambda_2 Q\left(\frac{\Lambda_1}{\Lambda_2}, N\right) = \Lambda_2 + \Lambda_1 Q\left(\frac{\Lambda_2}{\Lambda_1}, N\right).
\] (A.6)

The above process can be generalized for the case of a homogeneous asymptotically reliable serial production line consisting of \(M\) machines and \(M - 1\) buffers. Firstly, the first two machines \(M_1\) and \(M_2\) are combined into an aggregated machine with the loss parameter \(\Lambda_{2f}\) defined by (A.3), that is,
\[
\Lambda_{2f} = \Lambda_2 + \Lambda_1 Q\left(\frac{\Lambda_2}{\Lambda_1}, N_1\right).
\] (A.7)

The superscript “\(f\)” indicates that during the aggregation, we move forward (from machine \(M_1\) to machine \(M_M\)). The aggregated machine, characterized by \(\Lambda_{2f}\), is now combined with the third machine defined by the loss parameter \(\Lambda_3\). The new aggregated machine is characterized by the loss parameter
\[
\Lambda_{3f} = \Lambda_3 + \Lambda_{2f} Q\left(\frac{\Lambda_3}{\Lambda_{2f}}, N_2\right).
\] (A.8)

At the \(i\)th step of this multistage aggregation process, one may obtain
\[
\Lambda_{if} = \Lambda_i + \Lambda_{i-1f} Q\left(\frac{\Lambda_i}{\Lambda_{i-1f}}, N_{i-1}\right),
\] (A.9)

and at the final step,
\[
\Lambda_{Mf} = \Lambda_M + \Lambda_{M-1f} Q\left(\frac{\Lambda_M}{\Lambda_{M-1f}}, N_{M-1}\right).
\] (A.10)

The estimate of the throughput obtained as a result of this aggregation is
\[
X_{Mf} = 1 - \left[\Lambda_M + \Lambda_{M-1f} Q\left(\frac{\Lambda_M}{\Lambda_{M-1f}}, N_{M-1}\right)\right] \varepsilon.
\] (A.11)

Because there is no proof that \(X_{Mf}\) is close to the real throughput of a production line with \(M\) machines and \(M - 1\) buffers, another set of iterations, this time directed backward instead of forward, should be supplemented. This scheme is called backward aggregation.
A DP algorithm for the buffer allocation problem

and aggregates the line moving from the last machine $M_M$ to the first machine $M_1$. Thus

$$\Lambda_{M-1}^b = \Lambda_M + \Lambda_M Q \left( \frac{\Lambda_{M-1}^f}{\Lambda_M}, N_{M-1} \right),$$

$$\Lambda_{M-2}^b = \Lambda_{M-2} + \Lambda_{M-1}^b Q \left( \frac{\Lambda_{M-2}^f}{\Lambda_{M-1}}, N_{M-2} \right),$$

$$\Lambda_j^b = \Lambda_j + \Lambda_{j+1}^b Q \left( \frac{\Lambda_j^f}{\Lambda_{j+1}^b}, N_j \right),$$

and, at the final step,

$$\Lambda_1^b = \Lambda_1 + \Lambda_2^b Q \left( \frac{\Lambda_1}{\Lambda_2^b}, N_1 \right).$$

By repeating the process and constructing a new forward aggregation based on this backward aggregation, and so on, the following iterative algorithm is obtained:

$$\Lambda_i^f(s+1) = \Lambda_i + \Lambda_{i-1}^f(s+1) Q \left( \frac{\Lambda_i^b(s)}{\Lambda_{i-1}^f(s+1)}, N_{i-1} \right), \quad i = 2, \ldots, M,$$

$$\Lambda_j^b(s+1) = \Lambda_j + \Lambda_{j+1}^b(s+1) Q \left( \frac{\Lambda_j^f(s+1)}{\Lambda_{j+1}^b(s+1)}, N_j \right), \quad j = 1, \ldots, M-1,$$

$$s = 0, 1, \ldots, \Lambda_i^b(0) = \Lambda_i, \quad \Lambda_i^f(s) = \Lambda_1, \quad \Lambda_M^b(s) = \Lambda_M, \quad \forall s.$$}

Procedure (A.14) generates the following two sequences of throughput estimates:

$$X_M^f(s) = 1 - \Lambda_M^f(s) \varepsilon,$$

$$X_M^b(s) = 1 - \Lambda_M^b(s) \varepsilon.$$}

References


