European Journal of Operational Research 197 (2009) 572–580

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Production, Manufacturing and Logistics

Exact analysis of a two-workstation one-buffer flow line with parallel unreliable machines

Diamantidis C. Alexandros, Papadopoulos T. Chrissoleon *

Department of Economics, Aristotle University of Thessaloniki, Law School Building, Second Floor, GR-541 24, Thessaloniki, Greece

ARTICLE INFO

Article history:
Received 11 January 2008
Accepted 3 July 2008
Available online 15 July 2008

Keywords:
Flow/production lines
Unreliable parallel-machine workstations
Performance evaluation
Markovian analysis

ABSTRACT

This paper examines a model of a serial flow line with two workstations and an intermediate buffer. Each workstation consists of multiple unreliable parallel machines which are not necessarily identical, viz., the processing times, failure times and repair times of the parallel machines at each workstation are assumed to be exponentially distributed with non-identical mean rates. The system under consideration is solved via exact Markovian analysis. More specifically, a recursive algorithm that generates the transition matrix for any value of the intermediate buffer capacity is developed and all possible transition equations are derived and solved analytically. Once the transition equations are solved the performance measures of the model under consideration can be easily evaluated. This model may be used as a decomposition block for solving larger flow lines with parallel unreliable machines at each workstation.

1. Introduction and literature review

Markovian analysis can be used for modelling production/flow lines as queuing networks, when this is feasible, (Papadopoulos et al., 1989, 1990 and Heavey et al., 1993; among others). The main disadvantage of the Markovian analysis is that it is applicable only to small systems with up to six or seven workstations in series. This limitation is due to the huge state space which results when the number of workstations as well as the capacities of the intermediate buffers increase.

There is a vast literature on the analysis of flow lines where each workstation consists of a single-machine and the flow of material is linear (Helber, 1999; Papadopoulos et al., 1993).

There is a relatively scarce literature on the analysis of flow lines with multiple parallel-machine workstations.

Forestier (1980) examined production/flow lines with each workstation consisting of two parallel machines.

Dubois and Forestier (1982) examined analogous systems via Markovian analysis.

Van Dijk and Van der Wal (1989) obtained lower and upper bounds for finite multi-server exponential tandem queues.

Buzacott and Shanthikumar (1993), in their excellent book, analyzed finite buffer tandem queuing models with multiple machines per workstation and exponential processing times.

Magazine and Stecke (1996) considered small flow lines with two and three workstations consisting of reliable parallel machines.

Ancelin and Semery (1987) described methods in which each parallel-machine workstation was replaced by an equivalent single-machine workstation.

Vidalis and Papadopoulos (2000) developed a recursive algorithm for generating the transition matrices of multi-station multi-server exponential reliable queueing networks.

Chen and Lan (2001) proposed a mathematical model of a two stage production system with unreliable machines and finite working hour capacity.

Lan and Chen (2003) provided a mathematical model to reach the optimal multistage production with probabilistic demand, finite resource capacity at each stage, related costs, as well as sales price and unreliable machines.

Kalir and Arzi (1998) presented a model for the determination of the profit maximizing configuration of workstations each one consisting of parallel unreliable machines along a flexible production line with infinite buffers. The existence of infinite buffers does not allow the blocking phenomenon to occur.

* Corresponding author. Tel.: +30 2310997466.
E-mail addresses: adiama@econ.auth.gr (D.C. Alexandros), hpap@econ.auth.gr (P.T. Chrissoleon).

0377-2217/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2008.07.004

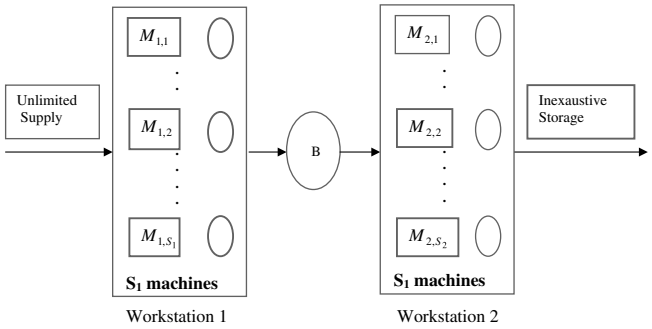

Fig. 1. A system with two workstations with parallel unreliable machines (servers) and an intermediate buffer.

In this paper, a system with two serial workstations, where each workstation consists of multiple *unreliable* parallel machines and an intermediate buffer of finite capacity as depicted in Fig. 1 is examined using Markovian analysis. This model may be used as a decomposition block of a decomposition method to evaluate the performance measures of large production systems with *unreliable* multiple parallel-machine workstations. For an introduction to decomposition the reader is addressed to Gershwin (1987), who was the first who applied this technique to analyze large production lines. Dallery et al. (1988) improved the convergence of the algorithm proposed by Gershwin (1987). In Diamantidis et al. (2007), decomposition was applied directly to systems consisting of parallel reliable machines at each workstation without using replacement techniques for evaluating the throughput of large flow lines. They presented numerical results for the estimation of throughput of large production lines (with up to 1000 workstations).

In Diamantidis et al. (2007), a decomposition block similar to the one considered in this work was used with the difference that all parallel machines were assumed to be reliable. This difference constitutes the contribution of this work.

This paper is organized as follows: Section 2 describes the model and gives the steps of the solution method. Section 3 presents the algorithm for generating the transition matrix as well as an example-application of the algorithm. In Section 4, numerical results are provided. Finally, Section 5 concludes the paper and gives a direction for further research.

2. The model and the solution method

A flow line consisting of two parallel-machine workstations, M_1 and M_2 and an intermediate buffer, B, capable of producing only one part type of products like the one depicted in Fig. 1 is analyzed using Markovian analysis.

Workstation M_1 consists of S_1 different unreliable parallel machines $M_{1,i}$ $i = 1,\ldots,S_1$, and likewise, workstation M_2 consists of S_2 different unreliable parallel machines $M_{2,j}$, $j = 1,\ldots,S_2$.

The S_i, $i = 1,2$ parallel machines at workstation M_i, $i = 1,2$ have exponentially distributed service times, times to failure and times to repair with means $\frac{1}{\mu_{i,j}}$, $\frac{1}{p_{i,j}}$ and $\frac{1}{r_{i,j}}$, $i = 1,2$ and $j = 1,2,\ldots,S_i$, respectively.

It is also assumed that when any one of the S_1 parallel machines at workstation M_1 completes a part, that part is placed into the buffer B downstream of the workstation, provided the buffer is not full.

The S_1 parallel machines at workstation 1 cannot fail when they are blocked, while the S_2 parallel machines at workstation 2 cannot fail when they are starved. Therefore, all machines at both workstations fail only while processing workpieces (Operation Dependent Failures (ODFs)).

The state of each one of the S_i, $i = 1,2$ parallel machines at workstation M_i, $i = 1,2$ is denoted by $a_{i,j} \in \{0,1\}$, $i = 1,2$ and $j = 1,2,\ldots,S_i$.

If $a_{i,j} = 1$, $i = 1,2$, $j = 1,\ldots,S_i$, machine $M_{i,j}$ is operational (*up state*). If $a_{i,j} = 0$, machine $M_{i,j}$ is under repair (*down state*).

For the system depicted in Fig. 1, it is assumed that an unlimited supply of workpieces is available upstream of workstation M_1 (i.e., workstation M_1 is never starved), and that an inexhaustive storage area is present downstream of workstation M_2 (i.e., workstation M_2 is never blocked). The size or capacity of the intermediate buffer is denoted by B. Because each parallel-machine has its own work area with capacity one (which is denoted by a circle beside each parallel-machine), there is a difference between the total storage capacity of the system and the buffer capacity. The total storage capacity of the system presented in Fig. 1, is the physical storage of buffer B plus the work areas of all parallel machines at both workstations M_1 and M_2. Therefore, the total storage capacity of the system, C, is $C = S_1 + S_2 + B$.

A part is assumed to be stored at the work area of each one of the parallel machines of workstation M_1 if it has been processed by the corresponding parallel-machine but finds the buffer containing B parts and S_2 parts currently occupy the work areas of all parallel machines of workstation M_2. However, if a part in the work area of one of the S_1 parallel machines at workstation M_1, is still being processed or waiting for the corresponding machine to be repaired, then this part is not assumed to be stored in the total storage capacity of the system as it is not yet available to proceed to workstation M_2.

Workstation M_1 can be either *partially* or *fully blocked*. More specifically, if the current inventory of parts of the system (including those on the machines) equals $S_2 + B + i$, $i = 1,\ldots,S_1 - 1$ then, only the first i machines at workstation M_1 are blocked and the remaining $S_1 - i$ machines are available for processing. In this case, workstation M_1 is partially blocked.

If the total storage capacity of the system equals $S_1 + S_2 + B$, then, all S_1 machines of the first workstation are blocked and workstation 1 is said to be fully blocked.

If the total storage capacity of the system equals 0 then all machines at workstation 2 are starved. If the total storage capacity of the system equals i, $i = 1,2,\ldots,S_2 - 1$ then the first i machines at workstation 2 are busy and the rest machines are starved.

Due to the exponentially distributed service times, failure times and repair times, during the time interval $[t, t + dt]$ only one event can occur at each workstation. Thus during the time interval $[t, t + dt]$ only one machine among the S_1 machines of workstation M_1 can produce

a part, fail or be repaired, respectively, or only one machine among the S_2 machines of workstation M_2 can remove a part from the intermediate buffer, fail or be repaired, respectively.

The state of the system at time t is represented by the vector:

$$S(t) = (n(t), a_{1,1}(t), a_{1,2}(t), \ldots, a_{1,S_1}(t), a_{2,1}(t), a_{2,2}(t), \ldots, a_{2,S_2}(t)),$$

where $n(t)$ is the total number of units in the system at time t (with a maximum capacity C).

Since the total number of units in the system at time t, $n(t)$, varies from 0 to $C = S_1 + S_2 + B$ and the state of the system is represented by the vector $S(t)$, it is straightforward that the total number of states of the system, N, is

$$N = 2^{S_1+S_2} \cdot (C+1). \quad (1)$$

To analyze the model of Fig. 1 using Markovian analysis, in order to evaluate the throughput of the system, the following steps should be followed:

Step 1: Derivation of the transition equations (internal, lower boundary and upper boundary) of the Markovian model and development of a recursive algorithm for generating the transition matrix for any value C of the total storage capacity of the system ($C = S_1 + S_2 + B$).

Step 2: Numerical computation of the transition probabilities and the various performance measures of the system, such as throughput.

3. The algorithm for generating the transition matrix

From Eq. (1) the dimension of the transition matrix for the system depicted in Fig. 1 is $N \times N$. Before starting the development of the recursive algorithm for generating the transition matrix for any value C of the total storage capacity, it is necessary to enumerate (order) the states of the system in a specific way in order to facilitate the transitions among the various states of the system.

For each value n of the total number of units in the system ($n = 0, \ldots, C$) there are $2^{S_1+S_2}$ states. The assignment of a number to each state can be done using the following pseudo code:

```
k=1
For n=0 to C do
 For α1,1 = 0 to 1 do
  For α1,2 = 0 to 1 do
      .
       .
        .
         For α1,S1 = 0 to 1 do
          For α2,1 = 0 to 1 do
            .
             .
              . For α2,S2 = 0 to 1 do
                    Assign number k to state S(t)
                    k=k+1
                end for
            end for
          end for
  end for
 end for
end for
```

Let k be the number that corresponds to the state from which we want to calculate the transition probabilities.

The proposed algorithm for generating the transition matrix for any value C of the total storage capacity of the system consists of the following three steps:

***Step 1*: Creation of transition probabilities from internal states:**

States $S = (n, a_{1,1}, a_{1,2}, \ldots, a_{1,S_1}, a_{2,1}, a_{2,2}, \ldots, a_{2,S_2})$, with $n = S_2, \ldots, S_2 + B$ are referred to as internal states.

All possible transitions can be divided in three categories:

(i) Transitions from states $(n, a_{1,1}, a_{1,2}, \ldots, a_{1,S_1}, a_{2,1}, a_{2,2}, \ldots, a_{2,S_2})$ to states $(n, a_{1,1}, a_{1,2}, \ldots, a_{1,S_1}, a_{2,1}, a_{2,2}, \ldots, a_{2,S_2})$. The transitions are done according to the following rules which may be written in a pseudo code:

For $n = S_2$ to $S_2 + B$ do
  For $g = 2$ to $S_1 + S_2 + 1$ do
    If the gth element of vector S is equal to 1 then move to the state that corresponds to number $k - 2^{S_1+S_2-g+1}$ with transition probability equal to the failure rate of the corresponding machine, the state of which is located to the gth position of vector S.
    If the gth element of vector S is equal to 0 then move to the state that corresponds to number $k + 2^{S_1+S_2-g+1}$ with transition probability equal to the repair rate of the corresponding machine, the state of which is located to the gth position of vector S.
  End for
End for

(ii) Transitions from states $(n, a_{1,1}, a_{1,2}, \ldots, a_{1,S_1}, a_{2,1}, a_{2,2}, \ldots, a_{2,S_2})$ to states $(n+1, a_{1,1}, a_{1,2}, \ldots, a_{1,S_1}, a_{2,1}, a_{2,2}, \ldots, a_{2,S_2})$. Such a transition is possible if there is at least one $a_{1,i} = 1$, $i = 1, \ldots, S_1$. The transitions are done according to the following rules which may be written in a pseudo code:

For $n = S_2$ to $S_2 + B$ do
If there is only one $a_{1,i}$, $i = 1, \ldots, S_1$ that is equal to 1 then move to the state that corresponds to number $k + 2^{S_1+S_2}$ with transition probability that is equal to the processing rate of the machine for which the corresponding state $a_{1,i}$, $i = 1, \ldots, S_1$ is equal to 1.
If there is more than one $a_{1,i}$, $i = 1, \ldots, S_1$ that is equal to 1 then move to the state that corresponds to number $k + 2^{S_1+S_2}$ with transition probability that is equal to the sum of the processing rates of the machines for which the corresponding states $a_{1,i}$, $i = 1, \ldots, S_1$ are equal to 1.
End For

(iii) Transitions from states $(n, a_{1,1}, a_{1,2}, \ldots, a_{1,S_1}, a_{2,1}, a_{2,2}, \ldots, a_{2,S_2})$ to states $(n-1, a_{1,1}, a_{1,2}, \ldots, a_{1,S_1}, a_{2,1}, a_{2,2}, \ldots, a_{2,S_2})$. Such a transition is possible if there is at least one $a_{2,i} = 1$, $i = 1, \ldots, S_2$. The transitions are done according to the following rules which may be written in a pseudo code:

For $n = S_2$ to $S_2 + B$ do
If there is only one $a_{2,i}$, $i = 1, \ldots, S_2$ that is equal to 1 then move to the state that corresponds to number $k - 2^{S_1+S_2}$ with transition probability that is equal to the processing rate of the machine for which the corresponding state $a_{2,i}$, $i = 1, \ldots, S_2$ is equal to 1.
If there are more than one $a_{2,i}$, $i = 1, \ldots, S_2$ that are equal to 1 then move to the state that corresponds to number $k - 2^{S_1+S_2}$ with transition probability that is equal to the sum of the processing rates of the machines for which the corresponding states $a_{2,i}$, $i = 1, \ldots, S_2$ are equal to 1.
End For

Finally, the probability that during the time interval $[t, t + \mathrm{d}t]$ the state of the system does not change can be found by subtracting the sum of all other transition probabilities from 1.

***Step 2*: Creation of transition probabilities from lower boundary states:**
States $S = (n, a_{1,1}, a_{1,2}, \ldots, a_{1,S_1}, a_{2,1}, a_{2,2}, \ldots, a_{2,S_2})$, with $n = 0, \ldots, S_2 - 1$ are referred to as lower boundary states.
All possible transitions can be divided in three categories:

(i) Transitions from states $(n, a_{1,1}, a_{1,2}, \ldots, a_{1,S_1}, a_{2,1}, a_{2,2}, \ldots, a_{2,S_2})$ to states $(n, a_{1,1}, a_{1,2}, \ldots, a_{1,S_1}, a_{2,1}, a_{2,2}, \ldots, a_{2,S_2})$. The transitions are done according to the following rules which may be written in a pseudo code:

For $n = 0$ to $S_2 - 1$ do
For $g = 2$ to $S_1 + S_2 + 1$ do
If the gth element of vector S is equal to 1 and $g < S_1 + n + 2$ then move to the state that corresponds to number $k - 2^{S_1+S_2-g+1}$ with transition probability that is equal to the failure rate of the corresponding machine, the state of which is located to the gth position of vector S.
If the gth element of vector S is equal to 1 and $g \geqslant S_1 + n + 2$ then there is no possible transition.
If the gth element of vector S is equal to 0 then move to state that corresponds to number $k + 2^{S_1+S_2-g+1}$ with transition probability that is equal to the repair rate of the corresponding machine, the state of which is located to the gth position of vector S.
End for
End for

(ii) Transitions from states $(n, a_{1,1}, a_{1,2}, \ldots, a_{1,S_1}, a_{2,1}, a_{2,2}, \ldots, a_{2,S_2})$ to states $(n+1, a_{1,1}, a_{1,2}, \ldots, a_{1,S_1}, a_{2,1}, a_{2,2}, \ldots, a_{2,S_2})$. Such a transition is possible if there is at least one $a_{1,i} = 1$, $i = 1, \ldots, S_1$. The transitions are done according to the following rules which may be written in a pseudo code:

For $n = 0$ to $S_2 - 1$ do
If there is only one $a_{1,i}$, $i = 1, \ldots, S_1$ that is equal to 1 then move to the state that corresponds to number $k + 2^{S_1+S_2}$ with transition probability that is equal to the processing rate of the machine for which the corresponding state $a_{1,i}$, $i = 1, \ldots, S_1$is equal to 1.
If there are more than one $a_{1,i}$, $i = 1, \ldots, S_1$ that are equal to 1 then move to the state that corresponds to number $k + 2^{S_1+S_2}$ with transition probability that is equal to the sum of the processing rates of the machines for which the corresponding states $a_{1,i}$, $i = 1, \ldots, S_1$ are equal to 1.
End For

(iii) Transitions from states $(n, a_{1,1}, a_{1,2}, \ldots, a_{1,S_1}, a_{2,1}, a_{2,2}, \ldots, a_{2,S_2})$ to states $(n-1, a_{1,1}, a_{1,2}, \ldots, a_{1,S_1}, a_{2,1}, a_{2,2}, \ldots, a_{2,S_2})$. Such a transition is possible if there is at least one $a_{2,i} = 1$, $i = 1, \ldots, S_2$ and $n > 0$. The transitions are done according to the following rules which may be written in a pseudo code:

For $n = 1$ to $S_2 - 1$ do
If there is only one $a_{2,i}$, $i = 1, \ldots S_2$ that is equal to 1 and its position, i, in vector S is such that $i < S_1 + n + 2$ then move to the state that corresponds to number $k - 2^{S_1+S_2}$ with transition probability that is equal to the processing rate of the machine for which the corresponding state $a_{2,i}$ $i = 1, \ldots, S_2$ is equal to 1. If its position i in vector S is such that $i \geqslant S_1 + n + 2$ then there is no possible transition.

If there are more than one $a_{2,i}$, $i = 1,\ldots,S_2$ that are equal to 1 and their positions, i, in vector S are such that $i < S_1 + n + 2$ then move to the state that corresponds to number $k - 2^{S_1+S_2}$ with transition probability that is equal to the sum of the processing rates of the machines for which the corresponding states $a_{2,i}$, $i = 1,\ldots,S_2$ are equal to 1. If their positions, i, in state vector S are such that $i \geqslant S_1 + n + 2$ then there is no possible transition.
End For

Finally, the probability that during the time interval $[t, t + dt]$ the state of the system does not change can be found by subtracting the sum of all other transition probabilities from 1.

***Step 3*: Creation of transition probabilities from upper boundary states:**

States $S = (n, a_{1,1}, a_{1,2}, \ldots, a_{1,S_1}, a_{2,1}, a_{2,2}, \ldots, a_{2,S_2})$, with $n = S_2 + B+1, \ldots, S_2 + B + S_1$ are referred to as upper boundary states. All possible transitions can be divided in three categories:

(i) Transitions from states $(n, a_{1,1}, a_{1,2}, \ldots, a_{1,S_1}, a_{2,1}, a_{2,2}, \ldots, a_{2,S_2})$ to states $(n, a_{1,1}, a_{1,2}, \ldots, a_{1,S_1}, a_{2,1}, a_{2,2}, \ldots, a_{2,S_2})$. The transitions are done according to the following rules which may be written in a pseudo code:

For $n = S_2 + B + 1$ to $S_2 + B + S_1$ do
For $g = 2$ to $S_1 + S_2 + 1$ do
If the gth element of vector S is equal to 1 and $g > n - S_2 - B + 1$ then move to the state that corresponds to number $k - 2^{S_1+S_2-g+1}$ with transition probability that is equal to the failure rate of the corresponding machine, the state of which is located to the gth position of vector S.
If the gth element of vector S is equal to 1 and $g \leqslant n - S_2 - B + 1$ then there is no possible transition.
If the gth element of vector S is equal to 0 then move to the state that corresponds to number $k + 2^{S_1+S_2-g+1}$ with transition probability that is equal to the repair rate of the corresponding machine, the state of which is located to the gth position of vector S.
End for
End for

(ii) Transitions from states $(n, a_{1,1}, a_{1,2}, \ldots, a_{1,S_1}, a_{2,1}, a_{2,2}, \ldots, a_{2,S_2})$ to states $(n + 1, a_{1,1}, a_{1,2}, \ldots, a_{1,S_1}, a_{2,1}, a_{2,2}, \ldots, a_{2,S_2})$. Such a transition is possible if there is at least one $a_{1,i} = 1$, $i = 1,\ldots,S_1$ and $n < S_1 + S_2 + B$. The transitions are done according to the following rules which may be written in a pseudo code:

For $n = S_2 + B + 1$ to $S_2 + B + S_1 - 1$ do
If there is only one $a_{1,i}$, $i = 1,\ldots,S_1$ that is equal to 1 and its position, i, at vector S is such that $i > n - S_2 - B + 1$ then move to the state that corresponds to number $k + 2^{S_1+S_2}$ with transition probability that is equal to the processing rate of the machine for which the corresponding state $a_{1,i}$, $i = 1,\ldots,S_1$ is equal to 1. If its position i in vector S is such that $i \leqslant n - S_2 - B + 1$ then there is no possible transition.
If there are more than one $a_{1,i}$, $i = 1,\ldots,S_1$ that are equal to 1 and their positions, i in vector S are such that $i > n - S_2 - B + 1$ then move to the state that corresponds to number $k + 2^{S_1+S_2}$ with transition probability that is equal to the sum of the processing rates of the machines for which the corresponding states $a_{1,i}$, $i = 1,\ldots,S_1$ are equal to 1. If their positions, i, in state vector S are such that $i \leqslant n - S_2 - B + 1$ then there is no possible transition.
End For

(iii) Transitions from states $(n, a_{1,1}, a_{1,2}, \ldots, a_{1,S_1}, a_{2,1}, a_{2,2}, \ldots, a_{2,S_2})$ to states $(n - 1, a_{1,1}, a_{1,2}, \ldots, a_{1,S_1}, a_{2,1}, a_{2,2}, \ldots, a_{2,S_2})$. Such a transition is possible if there is at least one $a_{2,i} = 1$, $i = 1,\ldots,S_2$. The transitions are done according to the following rules which may be written in a pseudo code:

For $n = S_2 + B + 1$ to $S_2 + B + S_1$ do
If there is only one $a_{2,i}$, $i = 1,\ldots S_2$ that is equal to 1 then move to the state that corresponds to number $k - 2^{S_1+S_2}$ with transition probability that is equal to the processing rate of the machine for which the corresponding state $a_{2,i}$, $i = 1,\ldots,S_2$ is equal to 1.
If there are more than one $a_{2,i}$, $i = 1,\ldots,S_2$ that are equal to 1 then move to the state that corresponds to number $k - 2^{S_1+S_2}$ with transition probability that is equal to the sum of the processing rates of the machines for which the corresponding states $a_{2,i}$, $i = 1,\ldots,S_2$ are equal to 1.
End For

Finally the probability that during the time interval $[t, t + dt]$ the state of the system does not change can be found by subtracting the sum of all other transition probabilities from 1.

3.1. Application of the algorithm: an example

An example is given to show application of the algorithm to the case: $S_1 = S_2 = 2$, $B = 2$, $r_{i,j} = 0.1, p_{i,j} = 0.01$, $i,j = 1,2$.
In this case, the total number of units in the system, n, varies from 0 to $C = 6$ and the total number of states equals $2^4(6 + 1) = 112$ states.
For each step of the algorithm a sample example that shows how the respective transition probabilities from a certain state are derived is given.

3.1.1. Application of Step 1

All states with $n = 2,3,4$ are the internal states.

(i) The transition probabilities from state $(2,0,1,0,1)$ to states $(2,a_{1,1},a_{1,2},a_{2,1},a_{2,2})$ are derived.
According to the ordering described in Section 3 number $k = 38$ is assigned to state $(2,0,1,0,1)$.Since $a_{1,1} = 0$ is located at the second position of the state vector there is a transition to state numbered $k + 2^{4-2+1} = 38 + 8 = 46$ which is state $(2,1,1,0,1)$ with transition probability $r_{1,1}$.Since $a_{1,2} = 1$ is located at the third position of the state vector there is a transition to state numbered $k - 2^{4-3+1} = 38 - 4 = 34$ which is state $(2,0,0,0,1)$ with transition probability $p_{1,2}$. Since $a_{2,1} = 0$ is located at the fourth position of the state vector there is a transition to state numbered $k + 2^{4-4+1} = 38 + 2 = 40$ which is state $(2,0,1,1,1)$ with transition probability $r_{2,1}$. Finally, since $a_{2,2} = 1$ is located at the fifth position of the state vector there is a transition to state numbered $k - 2^{4-5+1} = 38 - 1 = 37$ which is state $(2,0,1,0,0)$ with transition probability $p_{2,2}$.
(ii) The derivation of the transition probability from state $(2,0,1,0,1)$ to state $(3,0,1,0,1)$ is shown.
Because $a_{1,2} = 1$ such a transition is feasible. According to Step 1 (ii), there is a transition to state numbered $k + 2^4 = 38 + 16 = 54$ which is state $(3,0,1,0,1)$ with transition probability $\mu_{1,2}$.
(iii) The transition probability from state $(2,0,1,0,1)$ to state $(1,0,1,0,1)$ is presented.
Because $a_{2,2} = 1$ such a transition is feasible. According to Step 1 (iii), there is a transition to state numbered $k - 2^4 = 38 - 16 = 22$ which is state $(1,0,1,0,1)$ with transition probability $\mu_{2,2}$.
The transition probability from state $(2,0,1,0,1)$ to state $(2,0,1,0,1)$ is equal to $(1 - r_{1,1} - p_{1,2} - r_{2,1} - p_{2,2} - \mu_{1,2} - \mu_{2,2})$.

3.1.2. Application of Step 2

All states with $n = 0,1$ are the lower boundary states.

(i) The transition probabilities from state $(0,0,1,1,1)$ to states $(0,a_{1,1},a_{1,2},a_{2,1},a_{2,2})$ are given.
According to the ordering described in Section 3 number $k = 8$ is assigned to state $(0,0,1,1,1)$.
Since $a_{1,1} = 0$ is located at the second position of the state vector there is a transition to state numbered $k + 2^{4-2+1} = 8 + 8 = 16$ which is state $(0,1,1,1,1)$ with transition probability $r_{1,1}$.
Since $a_{1,2} = 1$ is located at the third position of the state vector there is a transition to state numbered $k - 2^{4-3+1} = 8 - 4 = 4$ which is state $(0,0,0,1,1)$ with transition probability $p_{1,2}$.
Since $a_{2,1} = a_{2,2} = 1$ are located at the fourth and fifth positions $(j = 4)$ and $(j = 5)$, respectively, of the state vector and $j \geqslant S_1 + n + 2 = 2 + 0 + 2 = 4$ there is no transition to other states with $n = 0$.
(ii) The transition probability from state $(0,0,1,1,1)$ to state $(1,0,1,1,1)$ is given.
Since $a_{1,2} = 1$ such a transition is feasible. According to Step 2 (ii), there is a transition to state numbered $k + 2^4 = 8 + 16 = 24$ which is state $(1,0,1,1,1)$ with transition probability $\mu_{1,2}$.
(iii) According to Step 2 (iii), since $n = 0$ there is no feasible transition to any other state $(n - 1,a_{1,1},a_{1,2},a_{2,1},a_{2,2})$.

Finally, the transition probability from state $(0,0,1,1,1)$ to state $(0,0,1,1,1)$ is equal to $(1 - r_{1,1} - p_{1,2} - \mu_{1,2})$.

3.1.3. Application of Step 3

All states with $n = 5,6$ are the upper boundary states.

(i) The transition probabilities from state $(6,1,1,0,1)$ to states $(6,a_{1,1},a_{1,2},a_{2,1},a_{2,2})$ are presented.
According to the ordering described in Section 3 number $k = 110$ is assigned to state $(6,1,1,0,1)$.
Since $a_{1,1} = 1$ is located at the second position $(j = 2)$ of the state vector and $j \leqslant n - S_2 - B + 1$ there is no feasible transition.
Since $a_{1,2} = 1$ is located at the third position $(j = 3)$ of the state vector and $j \leqslant n - S_2 - B + 1$ there is no feasible transition.
Since $a_{2,1} = 0$ is located at the fourth position of the state vector there is a transition to state numbered $k + 2^{4-4+1} = 110 + 2 = 112$ which is state $(6,1,1,1,1)$ with transition probability $r_{2,1}$.
Since $a_{2,2} = 1$ is located at the fifth position of the state vector there is a transition to state numbered $k - 2^{4-5+1} = 110 - 1 = 109$ which is state $(6,1,1,0,0)$ with transition probability $p_{2,2}$.
(ii) Because $n = 6$ according to Step 3(iii), there is no feasible transition from this state to any other state with $(n + 1,a_{1,1},a_{1,2},a_{2,1},a_{2,2})$.
(iii) Since $a_{2,2} = 1$ there is a transition to state numbered $k - 2^4 = 110 - 16 = 94$ which is state $(5,1,1,0,1)$ with transition probability $\mu_{2,2}$.

Finally, the transition probability from state $(6,1,1,0,1)$ to state $(6,1,1,0,1)$ is equal to $(1 - r_{2,1} - p_{2,2} - \mu_{2,2})$.

4. Numerical results

The transition equations for any value, C, of the total storage capacity of a system with two-workstation and one-buffer (depicted in Fig. 1) were solved and the steady state probabilities were computed using the LU decomposition technique. A C++ program that generates the transition matrix and solves numerically the transition equations computing all transition probabilities was developed.

The CPU time of the numerical solution for each case was also obtained.

The second column in all tables gives the throughput (mean production rate) of the system obtained by the proposed algorithm ($PR_{algorithm}$). The last column provides the CPU time (in seconds) taken by the proposed algorithm.

For the experiments presented in Table 1 it is assumed that $S_1 = 2, S_2 = 1$, the failure and repair rates of all machines at both workstations are equal to 0.01 and 0.1, respectively, while the size of the intermediate buffer varies from 0 to 10 with step 1.

Table 2 presents numerical results for systems where $S_i = 2$, $i = 1,2$, the buffer size is equal to 2, the failure and repair rates of all machines at both workstations are equal to 0.01 and 0.1, respectively, the processing rates of all machines at both workstations except machine $M_{1,2}$ are equal to 1 and the processing rate of machine $M_{1,2}$ varies from 0.1 to 1 with step 0.1.

Table 1
Throughput of a system with $S_1 = 2$, $S_2 = 1$, $\mu_{1,j} = 1$, $r_{1,j} = 0.1$, $p_{1,j} = 0.01$, $j = 1,2$, $\mu_{2,1} = 1$, $r_{2,1} = 0.1$, $p_{2,1} = 0.01$

Buffer size (B)	Throughput obtained from the algorithm ($PR_{algorithm}$)	CPU time in seconds
0	0.7848	0.141
1	0.8206	0.187
2	0.8374	0.219
3	0.8458	0.219
4	0.8502	0.234
5	0.8526	0.234
6	0.8540	0.219
7	0.8549	0.219
8	0.8554	0.234
9	0.8558	0.250
10	0.8561	0.250

Table 3 presents numerical results for systems where $S_i = 2$, $i = 1,2$, the buffer size is equal to 2, the failure and repair rates of all machines at both workstations are equal to 0.01 and 0.1, respectively, the processing rates of all machines at both workstations except machine $M_{2,2}$ are equal to 1 and the processing rate of machine $M_{2,2}$ varies from 0.1 to 1 with step 0.1.

In Table 4, numerical results are presented for systems where $S_i = 3$, $i = 1,2$, the buffer size is equal to 5, the processing rates of all machines at both workstations are equal to 1, the failure and repair rates of all machines at both workstations except machine $M_{1,2}$ are equal to 0.01 and 0.1, respectively, and the repair rate of machine $M_{1,2}$ varies from 0.01 to 0.1 with step 0.01.

Table 2
Throughput of a system with $S_i = 2$, $i = 1,2$, $B = 2$, $\mu_{2,j} = 1$, $j = 1,2$, $r_{i,j} = 0.1$, $p_{i,j} = 0.01$, $i,j = 1,2$, $\mu_{1,1} = 1$

Processing rate $\mu_{1,2}$	Throughput obtained from the algorithm ($PR_{algorithm}$)	CPU time in seconds
0.1	0.9572	0.235
0.2	1.0320	0.250
0.3	1.1031	0.250
0.4	1.1699	0.250
0.5	1.2323	0.250
0.6	1.2901	0.235
0.7	1.3433	0.234
0.8	1.3919	0.250
0.9	1.4360	0.234
1	1.4758	0.250

Table 3
Throughput of a system with $S_i = 2$, $i = 1,2$, $B = 2$, $\mu_{1,j} = 1$, $j = 1,2$, $r_{i,j} = 0.1$, $p_{i,j} = 0.01$, $i,j = 1,2$, $\mu_{2,1} = 1$

Processing rate $\mu_{2,2}$	Throughput obtained from the algorithm ($PR_{algorithm}$)	CPU Time in seconds
0.1	0.9690	0.235
0.2	1.0443	0.235
0.3	1.1151	0.234
0.4	1.1811	0.250
0.5	1.2422	0.234
0.6	1.2983	0.250
0.7	1.3495	0.250
0.8	1.3960	0.234
0.9	1.4380	0.234
1	1.4758	0.234

Table 4
Throughput of a system with $S_i = 3$, $i = 1,2$, $B = 5$, $\mu_{i,j} = 1$, $i = 1,2$, $j = 1,2,3$ $r_{2,j} = 0.1$, $j = 1,2,3$, $p_{i,j} = 0.01$, $i = 1,2$, $j = 1,2,3$, $r_{1,1} = r_{1,3} = 0.1$

Repair rate $r_{1,2}$	Throughput obtained from the algorithm ($PR_{algorithm}$)	CPU time in seconds
0.01	2.1275	31.578
0.02	2.2340	30.938
0.03	2.2854	30.985
0.04	2.3157	31.031
0.05	2.3356	31.110
0.06	2.3496	31.203
0.07	2.3601	32.468
0.08	2.3682	31.750
0.09	2.3746	30.797
0.1	2.3799	31.703

Table 5
Throughput of a system with $S_i = 3$, $i = 1,2$, $B = 5$, $\mu_{i,j} = 1$, $i = 1,2$, $j = 1,2,\ 3$, $r_{i,j} = 0.1$, $i = 1,2$, $j = 1,2,3$, $p_{1,j} = 0.01$, $j = 1,2,3$, $p_{2,1} = p_{2,2} = 0.01$

Failure rate $p_{2,3}$	Throughput obtained from the algorithm ($PR_{algorithm}$)	CPU time in seconds
0.001	2.4188	31.578
0.002	2.4142	30.938
0.003	2.4097	30.985
0.004	2.4053	31.031
0.005	2.4009	31.110
0.006	2.3966	31.203
0.007	2.3923	32.468
0.008	2.3881	31.750
0.009	2.3840	30.797
0.01	2.3799	31.703

Table 6
Throughput of a system with $S_i = 3$, $i = 1,2$, $B = 10,\ldots,100$, $\mu_{i,j} = 1$, $r_{i,j} = 0.1$, $p_{i,j} = 0.01$, $i = 1,2$, $j = 1,2,3$

Buffer size (B)	Throughput obtained from the algorithm ($PR_{algorithm}$)	CPU time in seconds
10	2.4607	14.485
20	2.5355	58.1250
30	2.5747	152.453
40	2.5998	320.140
50	2.6175	613.437
60	2.6306	1089.91
70	2.6408	1710.000
80	2.6490	3003.130
90	2.6556	5389.390
100	2.6612	9166.800

Table 5 gives numerical results for systems where $S_i = 3$, $i = 1,2$, the buffer size is equal to 5, the processing rates of all machines at both workstations are equal to 1, the failure and repair rates of all machines at both workstations except machine $M_{2,3}$ are equal to 0.01 and 0.1, respectively, and the failure rate of machine $M_{2,3}$ varies from 0.001 to 0.01 with step 0.001.

Finally, Table 6 presents numerical results for systems where $S_i = 3$, $i = 1,2$, the processing, failure and repair rates of all machines at both workstations are equal to 1, 0.01 and 0.1, respectively, while the size of the intermediate buffer varies from 10 to 100 with step 10.

The reader may see that the behaviour of the system is good and predictable. More specifically considering the numerical results given in Tables 1 and 6 it can be easily seen that while the buffer size increases the throughput of the system increases as well. The numerical results of Tables 2 and 3 indicate that while the processing rate of a machine increases the throughput also increases. The numerical results of Table 4 (Table 5) indicate that while the repair (failure) rates of a machine increase, respectively, the throughput of the system increases (decreases), respectively.

5. Conclusions and further research

A Markov process model with two serial workstations, where each workstation consists of multiple *unreliable* not identical parallel machines and an intermediate buffer of finite capacity as depicted in Fig. 1 was analyzed in this work. The processing times, failure times and repair times of all machines at both workstations were assumed to be exponentially distributed.

The transition equations of the Markovian model were derived and analytically solved and a procedure that generates the transition matrix for any value, C, of the total storage capacity was developed. The steady state probabilities were computed using the LU decomposition technique. The algorithm was programmed and implemented in the C++ programming language.

Diamantidis et al. (2007) examined a similar model with two workstations and one intermediate buffer with each workstation consisting of parallel reliable machines. The contribution of this study is that each parallel-machine at each workstation is considered to be unreliable with its own specific processing time, failure time and repair time whereas in the model examined by Diamantidis et al. (2007) all machines at each workstation were assumed to be identical and perfectly reliable.

This study may be extended by using the proposed model as a building block in a decomposition method to evaluate performance measures of large production/flow lines with parallel non-identical unreliable machines at each workstation.

References

Ancelin, B., Semery, A., 1987. Calcul de la productivité d'une ligne intégrée de fabrication: CALIF, une méthode analytique industrielle. RAIRO APII 21 (3), 209–238.
Buzacott, J.A., Shanthikumar, J.G., 1993. Stochastic Models of Manufacturing Systems. Prentice-Hall, Englewood Cliffs, New Jersey.
Chen, M.-S., Lan, C.-H., 2001. Two-stage production with unreliable machine and finite working hour capacity. International Journal of Information and Management Sciences 12 (2), 11–24.
Dallery, Y., David, R., Xie, X., 1988. An efficient algorithm for the analysis of transfer lines with unreliable machines and finite buffers. IIE Transactions 20 (3), 280–283.
Diamantidis, A.C., Papadopoulos, C.T., Heavey, C., 2007. Approximate Analysis of serial flow lines with multiple parallel-machine stations. IIE Transactions 39 (4), 361–375.
Dubois, D., Forestier, J.P., 1982. Productivité et en-cours moyens d'un ensemble de deux machines separées par une zone de stockage. RAIRO Automatique 16 (2), 105–132.
Forestier, J.P., 1980. Modélisation stochastique et comportement asymptotique d'un système automatise de production. RAIRO Automatique 14 (2), 127–143.
Gershwin, S.B., 1987. An efficient decomposition algorithm for the approximate evaluation of tandem queues with finite storage space and blocking. Operations Research 35, 291–305.
Heavey, C., Papadopoulos, H.T., Browne, J., 1993. The throughput rate of multistation unreliable production lines. European Journal of Operational Research 68, 69–89.

Helber, S., 1999. Performance Analysis of Flow-Lines with Nonlinear Flow of Material. Lecture Notes in Economics and Mathematical Systems, vol. 473. Springer-Verlag.
Kalir, A., Arzi, Y., 1998. Optimal design of flexible production lines with unreliable machines and infinite buffers. IIE Transactions (Institute of Industrial Engineers) 30 (4), 391–399.
Lan, C.-H., Chen, M.-S., 2003. Multistage production with probabilistic demand and finite resource capacity. Transactions of the Canadian Society for Mechanical Engineering 27 (1-2), 77–91.
Magazine, M.J., Stecke, K.E., 1996. Throughput for production lines with serial workstations and parallel service facilities. Performance Evaluation 25, 211–232.
Papadopoulos, H.T., Heavey, C., O'Kelly, M.E.J., 1989. The throughput rate of multistation reliable production lines with interstation buffers: (I) Exponential case. Computers in Industry 13, 229–244.
Papadopoulos, H.T., Heavey, C., O'Kelly, M.E.J., 1990. Throughput rate of multistation reliable production lines with interstation buffers: (II) Erlang case. Computers in Industry 13, 317–337.
Papadopoulos, H.T., Heavey, C., Browne, J., 1993. Queueing Theory in Manufacturing Systems Analysis and Design. Chapman & Hall, London.
Van Dijk, N.M., Van der Wal, J., 1989. Simple bounds and monotonicity results for finite multi-server exponential tandem queues. Queuing Systems 4, 1–16.
Vidalis, M.I., Papadopoulos, H.T., 2000. A recursive algorithm for generating the transition matrices of multistation multiserver exponential reliable queuing networks. Computers and Operations Research 28, 853–883.