Helber, Stefan; Kellenbrink, Carolin; Südbeck, Insa

Article — Published Version

Evaluation of stochastic flow lines with provisioning of auxiliary material

OR Spectrum

Suggested Citation: Helber, Stefan; Kellenbrink, Carolin; Südbeck, Insa (2023) : Evaluation of stochastic flow lines with provisioning of auxiliary material, OR Spectrum, ISSN 1436-6304, Springer Berlin Heidelberg, Berlin/Heidelberg, Vol. 46, Iss. 3, pp. 669-708, https://doi.org/10.1007/s00291-023-00737-9

This Version is available at:
https://hdl.handle.net/10419/317050

OR Spectrum (2024) 46:669–708
https://doi.org/10.1007/s00291-023-00737-9

ORIGINAL ARTICLE

Evaluation of stochastic flow lines with provisioning of auxiliary material

Stefan Helber[1] · **Carolin Kellenbrink**[1] · **Insa Südbeck**[1]

Received: 18 December 2022 / Accepted: 18 October 2023 / Published online: 4 December 2023

Abstract

Flow lines are often used to perform assembly operations in multi-stage processes. During these assembly operations, components that are relatively small, compared to the work pieces travelling down the flow line, are mounted to the work pieces at a given stage. Those components, or more generally, any kind of auxiliary material, are provisioned to the corresponding production stage in a repetitive but not necessarily deterministic manner using a certain delivery frequency, each time filling the local storage up to a predetermined order-up-to level. Just like random processing times, machine failures, and repairs, the randomness of the provisioning process can impact the long-term throughput of such a flow line. In this paper, we develop a fast and accurate analytical performance evaluation method to estimate the long-term throughput of a Markovian flow line of this type for the practically important case of limited buffer capacities between the production stages. We first give an exact characterization of a two-machine line of that type and show how to determine system state probabilities and aggregate performance measures. Furthermore, we show how to use this two-machine model as the building block of an approximate decomposition approach for longer flow lines. As opposed to previous decomposition approaches, even the state space of the two-machine lines can become so large that an exact solution of the Markov chains can become impractical. We hence show how to set up, train, and use an artificial neural network to replace the Markov chain solver embedded in the decomposition approach, which then leads to an accurate and extremely fast flow line evaluation tool. The proposed methodology is evaluated by a comparison with simulation results and used to characterize the structural patterns describing the behaviour of flow lines of this type. The method can be used to systematically consider the combined impact of the delivery frequency and the local order-up-to levels for the auxiliary material when designing a flow line of this type.

✉ Stefan Helber
stefan.helber@prod.uni-hannover.de

[1] Institut für Produktionswirtschaft, Leibniz University Hannover, Königsworther Platz 1, Hannover D-30167, Germany

Keywords Flow line evaluation · Auxiliary material · Decomposition · Markov chain · Artificial neural network · Automatically guided vehicles

1 Introduction

In this paper, we consider a model of a stochastic flow line with buffers of limited size between adjacent machines at the respective production stages. As in previously published papers, we assume random processing times, random times to failure and random repair times, see (Papadopoulos et al. 2019) for a recent overview. In such systems, a main design question is to choose the size of buffers between adjacent machines for given machine parameters.

The additional feature characterizing this paper is a further random process directed at the provisioning of auxiliary material at the production stages. Examples of this auxiliary material are components that are mounted on the work pieces as part of the assembly operation performed at the respective production stage. In practice, one can find storage facilities such as racks, containers, or shelves next to the machines in which the auxiliary material is temporarily stored. As the auxiliary material is being depleted during the assembly operation, new material has to be provisioned in a regular manner.

One way to achieve this is to operate with a so-called milk-run system in which a vehicle travels along the stations of the line to provision those stations with a common delivery frequency. Another option is to operate with vehicles that head during each provisioning trip from the auxiliary material warehouse to only a single station at the line. In this type of system, the delivery frequency can be station-specific, which can be advantageous, e.g. if at some stations very voluminous types of auxiliary material are delivered rather frequently to save storage space along the line. Such a material provisioning system can, for example, be realized with automatically guided vehicles (AGVs). From the perspective of an individual station, the arrival process of those vehicles can then appear as random, in a manner similar to the arrival process of work pieces to a machine in an automated flexible manufacturing system.

Upon arrival of such a vehicle, the local storage of the auxiliary material gets refilled to a predetermined order-up-to level. This leads to further important design questions beyond the distribution of buffers. On the one hand, it is not attractive to have oversized local storage systems, in particular as they require valuable floor space. On the other hand, overly frequent replenishment operations place a heavy burden on the transportation system used to bring the auxiliary material to the flow line, which can also be costly in terms of required vehicles and furthermore lead to congestion in the internal transportation system.

Analysing such a system with complex interactions between main product movements down the stochastic flow line and auxiliary material provisioning at the stations is a non-trivial task, in particular if random processing, failure, repair, and material provisioning times together drive the performance of the flow line. For this reason, we propose a Markovian model of such a flow line by assuming

that all those random times follow individual exponential distributions. In this attempt, we encounter the typical problem of the explosion of the state space. For this reason, we use a decomposition method as proposed by Choong and Gershwin (1987) to numerically analyse the system. The building block of the decomposition is a two-machine model which we need to solve frequently, quickly, and accurately. In previously proposed decomposition methods for stochastic flow lines, it typically turned out to be unproblematic to solve the two-machine model, i.e. to quickly and exactly determine the steady-state probabilities and, subsequently, performance measures such as long-term throughput, blocking and starving probabilities, since the state space of the two-machine models was typically rather small.

In our case, however, this is no longer true. The size of the state space of the two-machine model can become so large that analysing the two-machine system becomes a problem which requires special attention. We use a Gauss–Seidel (GS) approach to determine the steady-state probabilities of the two-machine lines numerically. As a methodological alternative, we also trained an artificial neural network (ANN) based on numerically determined performance data for two-machine lines and then used this neural network to predict the relevant performance measures and state probabilities inside of the decomposition approach for longer flow lines with more than two machines, to show how a hybrid between established flow line decomposition approaches and artificial neural networks can be constructed in the case of extremely large two-machine system state spaces. This approach is in principle applicable to other system configurations as well.

Our paper contributes to the research on manufacturing systems in multiple ways. First, we add the modelling component of exponentially distributed inter-replenishment times of the auxiliary material to the Markovian model of unreliable flow lines with random processing times, times to failure, and repair times. For the case of the two-machine model, we give a detailed description of the resulting continuous time Markov chain (CTMC) model and show how to determine steady-state probabilities and performance measures numerically, in spite of the relatively large state space of the two-machine model. We furthermore show to which extent those values can be predicted by a neural network. Second, we show how the decomposition for longer lines proposed in Choong and Gershwin (1987) can be adapted to deal with the new flow line feature. Third, we use the models to study the delicate interactions of the different sources of randomness and design parameters such as buffer sizes, order-up-to levels, and delivery frequencies on the performance on the flow line. This helps to develop intuition about the conditions under which the provisioning process for the auxiliary material has a major impact on the line.

The remainder of this paper is organized as follows: In Sect. 2, we review the relevant literature. A technical description of the studied system as well as a formal model is given in Sect. 3. The core of the paper is Sect. 4 in which we describe several methods used to determine performance measures for flow lines of the studied type. We start with the two-machine model and then move on to the analysis of longer lines via a decomposition approach. Numerical results for the application of those methods are then presented in Sect. 5. We conclude with Sect. 6 and give hints for further research.

2 Review of the literature

Over the decades, a rich body of literature has emerged that addresses flow line design and operation problems from many different angles. A first and tremendously important decision area addresses the line balancing problem of allocating tasks to stations of the line, see (Becker and Scholl 2006; Scholl and Becker 2006; Boysen et al. 2022). More recently, the organization of material supply for the stations has gained attention, in particular for mixed-model assembly lines and often via tow trains, see, for example, Emde and Boysen (2012); Emde et al. (2012); Boysen and Emde (2014) for surveys. The integration of the so-called "supermarkets" holding the material as well as Kanban number optimization for the auxiliary material is studied, for example, by Nourmohammadi et al. (2019); Faccio et al. (2013); Delice et al. (2023). The decisions treated in these works essentially shape the system studied in this paper, which in turn focuses on a probabilistic perspective. For this reason, the relevant literature for this work concerns stochastic flow lines systems in general, decomposition approaches for analysing these systems, and the supply of auxiliary material. Dallery and Gershwin (1992); Papadopoulos and Heavey (1996), and Papadopoulos et al. (2009) provide general overviews of the analysis and design of flow lines. Additionally, Hudson et al. (2015) provides an overview of unbalanced flow lines. We can find overviews of the analysis of two-machine systems in Li et al. (2006) and Papadopoulos et al. (2019).

The exact analysis of flow lines with CTMC is limited to very small systems. However, we can use the developed methods even to evaluate longer lines within approximate approaches such as aggregation and decomposition. In aggregation, virtual machines iteratively model the aggregated behaviour of the subsequent or previous system. Li and Meerkov (2009) develop several extensions for this approach. In the decomposition method introduced by Gershwin (1987), one two-machine-one-buffer system is introduced for each buffer of the flow line. The parameters of the virtual machines are updated iteratively until the flow through all substitute systems is identical. Choong and Gershwin (1987) extended the approach to exponential processing, failure, and repair rates. Dallery et al. (1988) developed the Dallery-David-Xie (DDX) algorithm to solve the decomposition equations efficiently. Burman (1995) developed the accelerated DDX, which is faster and more robust than the DDX.

Several authors developed decomposition approaches for different assumptions on the flow line systems, e.g. Tempelmeier and Bürger (2001), and Helber (2005). Helber (1998), Manitz (2015), and Tancrez (2020) examine flow lines with assembly/disassembly operations. Helber (1999) additionally considers rework loops within these systems. The work of Li (2005) considers a decomposition approach for flow lines with rework loops and scrapping. Sachs et al. (2022) developed a decomposition approach for flow lines with spare parts needed for a machine's repair process. To our knowledge, there is no decomposition approach for evaluating flow lines with auxiliary material.

There is only a limited number of publications on flow lines with the provisioning of auxiliary material. Some articles consider flow lines with a synchronous

flow of work pieces, e.g. Bukchin and Meller (2005); Alnahhal and Noche (2015), and Baller et al. (2020). However, we consider systems with an asynchronous flow of work pieces. Hence, the processing is independent of each other on the different stages of the flow line. Yan et al. (2010) consider the problem of assigning drivers for the supply of line-side buffers to avoid material shortages. The authors develop a simulation-based algorithm. Chang et al. (2013) analyse a manufacturing and material handling system. They develop an integrated model for evaluating the throughput, work in progress, driver utilization, and material supply. Weiss et al. (2017) use a sample-based approach in combination with a rule-based local search procedure to optimize the buffer allocation within a flow line where the first machine needs a limited material. Based on this model, the authors evaluate different order policies for the supply of the first machine. Mindlina and Tempelmeier (2022) analyse flow lines with a milk-run supply of material. In such systems, a transport vehicle visits all stations in a deterministic time interval. The authors develop a mixed integer model for the integrated evaluation and optimization of the flow line and material supply. Südbeck et al. (2023) evaluate milk-run-supplied flow lines with a recurrent neural network. They optimize the buffers and material supply with different gradient and local search approaches.

3 Problem and model description

Figure 1 depicts a schematic representation of the flow line model considered in this paper. Squares indicate machines and circles represent buffers. The model is based on the following assumptions: The work pieces travel through the system from the left to the right and leave the system again when the process on the last machine has been completed. Upstream of the first machine, there is an infinite supply of work pieces to be processed. Likewise, there is an infinite space downstream of the last machine. The first machine can hence never be starved and the last machine can never be blocked. The system can therefore be analysed in isolation from its surroundings.

Processing times at machine i are exponentially distributed with rate μ_i. The buffers between adjacent machines i and $i+1$ can hold up to C_i work pieces. We assume *blocking after service* (BAS), i.e. if upon process completion at machine i the buffer

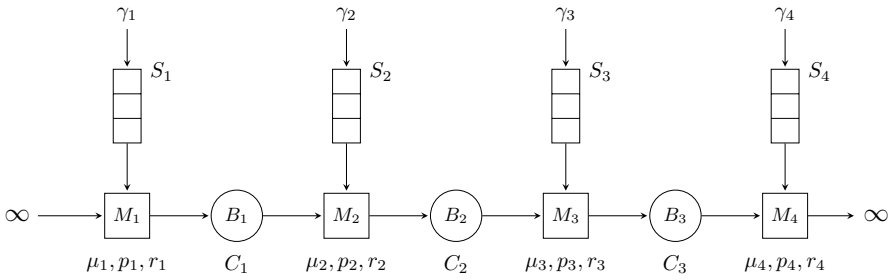


Fig. 1 Example of a four-machine flow line

downstream of machine i is full, the processed work piece remains on machine i which is then blocked.

While a machine i operates, it can fail, i.e. we assume operation-dependent failures (ODF). In other words, a machine that is blocked or starved cannot fail. Times to failure as well as repair times are exponentially distributed with rate p_i and r_i, respectively.

For each operation on a work piece, the machine requires one unit of the auxiliary material, e.g. a component to be mounted to the work piece. This is not a limitation of generality, since for example a set of four screws may be defined as one unit. The auxiliary material is brought to machine i with exponentially distributed inter-arrival times with rate γ_i. Upon arrival of the transport of the auxiliary material for machine i, the local level is raised up to the order-up-to level S_i. This could be due to an individual AGV driving to an individual station (as opposed to a tow train visiting several adjacent stations in a single trip) exchanging a partially empty transportation box or rack for auxiliary material by a full one. (In a practical setting, we typically find $S_i \gg 1$ and hence $\gamma_i \ll \mu_i$.)

The state of the system at any given moment in time can be described by specifying for each machine i the following elements:

- The number $n_i = 0, ..., C_i + 2$ of work pieces already processed by machine i, but not yet by machine $i + 1$, so that for $n_i > 0$ we have one such work piece on machine $i + 1$ and the remaining $n_i - 1$ work pieces are waiting in buffer between machine i and $i + 1$ or on machine i itself if the buffer is full,
- the state α_i, with $\alpha_i = 1$ indicating that machine i is operational and $\alpha_i = 0$ that it is down, and
- the level $l_i = 0, ..., S_i$ of the auxiliary material, with $l_i = 0$ indicating that machine i cannot operate as it lacks the auxiliary material.

In order for machine i to operate, we must simultaneously have $(n_{i-1} > 0) \wedge (n_i < C_i + 2) \wedge (\alpha_i = 1) \wedge (l_i > 0)$, i.e. the machine is neither starving nor blocked, operational and there is at least one unit of auxiliary material available. The state of an I-stage flow line can hence be described as the following tuple:

$$s = (n_1, n_2, ..., n_{I-1}, \alpha_1, \alpha_2, ..., \alpha_I, l_1, l_2, ..., l_I) \tag{1}$$

Not all combinations of values for the elements of the tuple constitute a feasible state. For example, due to the assumption of operation-dependent failures, we know that if machine i is down, i.e. $\alpha_i = 0$, then we cannot possibly have $n_{i-1} = 0$ (machine is starved) or $n_i = C_i + 2$ (machine is blocked) or $l_i = 0$ (machine lack the auxiliary material).

If we denote with

$$\mathscr{P}(s) = \mathscr{P}(n_1, n_2, ..., n_{I-1}, \alpha_1, \alpha_2, ..., \alpha_I, l_1, l_2, ..., l_I) \tag{2}$$

the probability of being in state s, then we can define the throughput of the system as determined via the last machine I, i.e. $\alpha_I = 1$, $n_{I-1} > 0$, and $l_I > 0$, as follows:

$$TP = \sum_{n_1=0}^{C_1+2} \cdots \sum_{n_{I-2}=0}^{C_{I-2}+2} \sum_{n_{I-1}=1}^{C_{I-1}+2} \sum_{\alpha_1=0}^{1} \cdots \sum_{\alpha_{I-1}=0}^{1} \sum_{l_1=0}^{S_1} \cdots \sum_{l_{I-1}=0}^{S_{I-1}} \sum_{l_I=1}^{S_I} \mathscr{P}(n_1, n_2, \cdots, n_{I-1}, \alpha_1, \alpha_2, \cdots, \alpha_I = 1, l_1, l_2, \cdots, l_I) \cdot \mu_I \tag{3}$$

Other performance measures can, in principle, be defined similarly. Note, however, this is merely of conceptional interest as the state space is extremely large, so that it is practically impossible to determine steady-state probabilities $\mathscr{P}(s)$ in Eq. (2) for a line with more than two machines.

4 Solution approaches for two-machine models and for longer lines

4.1 Two-machine model: transitions and performance measures

For a flow line with only two machines, denoted as the upstream (u) and the downstream (d) machines, the state can be given as the following five-dimensional tupel:

$$s = (n, \alpha_u, \alpha_d, l_u, l_d) \tag{4}$$

To determine the steady-state probabilities $\mathscr{P}(s)$ for such a two-machine model, we have to find a solution to the set of transition equations describing the dynamics of the underlying continuous-time Markov chain and the additional normalization condition with $N = C + 2$:

$$\sum_{n=0}^{N} \sum_{\alpha_u=0}^{1} \sum_{\alpha_d=0}^{1} \sum_{l_u=0}^{S_u} \sum_{l_d=0}^{S_d} \mathscr{P}(n, \alpha_u, \alpha_d, l_u, l_d) = 1 \tag{5}$$

As a notational tool, we define an indicator function $\mathbb{1}_{(\mathscr{L})}$ operating on a logical proposition $\mathscr{L}$ as follows:

$$\mathbb{1}_{(\mathscr{L})} = \begin{cases} 1, & \text{if } \mathscr{L} \text{ is true} \\ 0, & \text{if } \mathscr{L} \text{ is false.} \end{cases} \tag{6}$$

With the help of this indicator function, we can now state the balance equations for the two-machine system being in steady state for $n = 0, ..., N = C + 2$, $\alpha_u \in \{0, 1\}$, $\alpha_d \in \{0, 1\}$, $l_u = 0, \cdots, S_u$, and $l_d = 0, \cdots, S_d$:

$$
\begin{aligned}
&\mathscr{P}(n,\alpha_u,\alpha_d,l_u,l_d)\cdot\Big((\mu_u+p_u)\cdot\mathbb{1}_{(n<N,\,\alpha_u=1,l_u>0)}+(\mu_d+p_d)\cdot\mathbb{1}_{(n>0,\,\alpha_d=1,l_d>0)}\\
&\quad+r_u\cdot\mathbb{1}_{(\alpha_u=0)}+r_d\cdot\mathbb{1}_{(\alpha_d=0)}+\gamma_u\cdot\mathbb{1}_{(l_u<S_u)}+\gamma_d\cdot\mathbb{1}_{(l_d<S_d)}\Big)\\
&=\mathscr{P}(n-1,\alpha_u,\alpha_d,l_u+1,l_d)\cdot\mu_u\cdot\mathbb{1}_{(n>0,\,\alpha_u=1,l_u<S_u)}\\
&\quad+\mathscr{P}(n+1,\alpha_u,\alpha_d,l_u,l_d+1)\cdot\mu_d\cdot\mathbb{1}_{(n<N,\,\alpha_d=1,l_d<S_d)}\\
&\quad+\mathscr{P}(n,\alpha_u+1,\alpha_d,l_u,l_d)\cdot p_u\cdot\mathbb{1}_{(n<N,\,\alpha_u=0,l_u>0)}\\
&\quad+\mathscr{P}(n,\alpha_u,\alpha_d+1,l_u,l_d)\cdot p_d\cdot\mathbb{1}_{(n>0,\,\alpha_d=0,l_d>0)}\\
&\quad+\mathscr{P}(n,\alpha_u-1,\alpha_d,l_u,l_d)\cdot r_u\cdot\mathbb{1}_{(\alpha_u=1)}\\
&\quad+\mathscr{P}(n,\alpha_u,\alpha_d-1,l_u,l_d)\cdot r_d\cdot\mathbb{1}_{(\alpha_d=1)}\\
&\quad+\sum_{l'_u=0}^{S_u-1}\mathscr{P}(n,\alpha_u,\alpha_d,l'_u,l_d)\cdot\gamma_u\cdot\mathbb{1}_{(l_u=S_u)}\\
&\quad+\sum_{l'_d=0}^{S_d-1}\mathscr{P}(n,\alpha_u,\alpha_d,l_u,l'_d)\cdot\gamma_d\cdot\mathbb{1}_{(l_d=S_d)}
\end{aligned}
\tag{7}
$$

For small buffer sizes C and order-up-to levels S_u and S_d, this linear system of equations of the unknown state probabilities (including the normalization constraint) can be solved directly, for example, using MATLAB. However, even for moderate buffer sizes of $C = 10$ and order-up-to levels of $S_u = S_d = 20$, we already have $(10+3)\cdot 2\cdot 2\cdot(20+1)\cdot(20+1) = 22,932$ different system states and setting up the complete generator matrix Q quickly becomes impractical. Fortunately, we can exploit the fact that it is a sparse matrix in which most entries are 0. We hence employ a tailored implementation of GS which takes advantage of the fact that the generator matrix of the CTMC defined by Eq. (7) is extremely sparse to solve the two-machine model, i.e. to determine its steady-state probabilities.

Given the values of the steady-state probabilities $\mathscr{P}(n,\alpha_u,\alpha_d,l_u,l_d)$, we can compute the throughput of the system via, for example, the upstream machine as

$$
TP=\mu_u\cdot\sum_{n=0}^{N-1}\sum_{\alpha_d=0}^{1}\sum_{l_u=1}^{S_u}\sum_{l_d=0}^{S_d}\mathscr{P}(n,\alpha_u=1,\alpha_d,l_u,l_d) \tag{8}
$$

and the average buffer level as

$$
BL=\sum_{n=0}^{N}\sum_{\alpha_u=0}^{1}\sum_{\alpha_d=0}^{1}\sum_{l_u=0}^{S_u}\sum_{l_d=0}^{S_d}n\cdot\mathscr{P}(n,\alpha_u,\alpha_d,l_u,l_d). \tag{9}
$$

This two-machine model can, on the one hand, already be used for an exact analysis of the behaviour of a short line of the type under consideration. However, it can also serve as a building block in a decomposition approach suitable to analyse longer lines.

Over the course of that decomposition, we will need three further identities related to the two-machine models. The first identity stems from the fact that, in

the long run, for each failure, say of the upstream machine, there must be a repair of that upstream machine:

$$\begin{aligned} & p_u \cdot \mathscr{P}[\alpha_u = 1 \wedge n < N \wedge l_u > 0] \\ = \quad & r_u \cdot \mathscr{P}[\alpha_u = 0 \wedge n < N \wedge l_u > 0] \end{aligned} \tag{10}$$

For the virtual upstream machine to be able to fail, it must be up, not be blocked, and it must not lack the auxiliary material. On the other hand, a machine being down implies that it is not blocked and does not lack its auxiliary material.

A further identity, the two-machine *flow rate–idle time* (FRIT) equation, addresses the two-machine line throughput from the perspective of either the up- or the downstream machine. The key idea is that the throughput of the upstream machine

$$TP = \mu_u \cdot \frac{r_u}{r_u + p_u} \cdot (1 - \mathscr{P}[n = N \vee l_u = 0]) \tag{11}$$

$$= \mu_u \cdot e_u \cdot (1 - \mathscr{P}[n = N \vee l_u = 0]) \tag{12}$$

is the product of its isolated processing rate μ_u, the efficiency (or "availability") $e_u = \frac{r_u}{r_u + p_u}$ of that machine, and the probability that the machine is neither blocked nor lacking its auxiliary material. This identity can be solved for the latter probability as follows:

$$1 - \mathscr{P}[n = N \vee l_u = 0] = \frac{TP}{\mu_u \cdot e_u} \tag{13}$$

From an alternative formulation of Eq. (8)

$$TP = \mu_u \cdot \mathscr{P}[\alpha_u = 1 \wedge n < N \wedge l_u > 0], \tag{14}$$

together with Eq. (10), a further identity results:

$$\mathscr{P}[\alpha_u = 0 \wedge n < N \wedge l_u > 0] = TP \frac{p_u}{\mu_u \cdot r_u} \tag{15}$$

Analogous identities hold for the downstream machine, i.e.

$$1 - \mathscr{P}[n = 0 \vee l_d = 0] = \frac{TP}{\mu_d \cdot e_d} \tag{16}$$

and

$$\mathscr{P}[\alpha_d = 0 \wedge n > 0 \wedge l_d > 0] = TP \frac{p_d}{\mu_d \cdot r_d}. \tag{17}$$

Those four identities (13), (15), (16), and (17) will all be used in the derivation of the decomposition equations.

4.2 Decomposition approach for longer lines

4.2.1 Basic idea of the decomposition approach

As shown in Fig. 2, the basic idea of the decomposition approach as presented by Choong and Gershwin (1987) is to introduce as many virtual two-machine lines as the original system has buffers. The goal is to break down the flow line into coupled subsystems that are each in itself easier to solve. In Fig. 2, the virtual upstream machine $M_u(2)$ of the second virtual line, for example, serves as an aggregate representation of the part of the original line that is upstream of buffer B_2 of that original line and drives the flow of work pieces into that buffer.

In our exposition below, we follow the convention to use subscripts to denote parameters or performance measures related to the original line and indices in parentheses as we denote virtual lines or their machines. So r_3 denotes the repair rate of machine 3 of the original line, whereas $r_u(3)$ is the repair rate of the upstream machine of virtual two-machine line 3.

The parameter types and the general behaviour of the virtual machines equal those of the machines in the original line. Selected parameters, i.e. the buffer sizes C, order-up-to levels S, and auxiliary material replenishment rates γ of the virtual machines are even assumed to be identical to those of the corresponding machines in the original line. As in Gershwin's original approach, we assume, however, that

- processing rates $\mu_u(i)$ and $\mu_d(i)$,
- failure rates $p_u(i)$ and $p_d(i)$, and
- repair rates $r_u(i)$ and $r_d(i)$

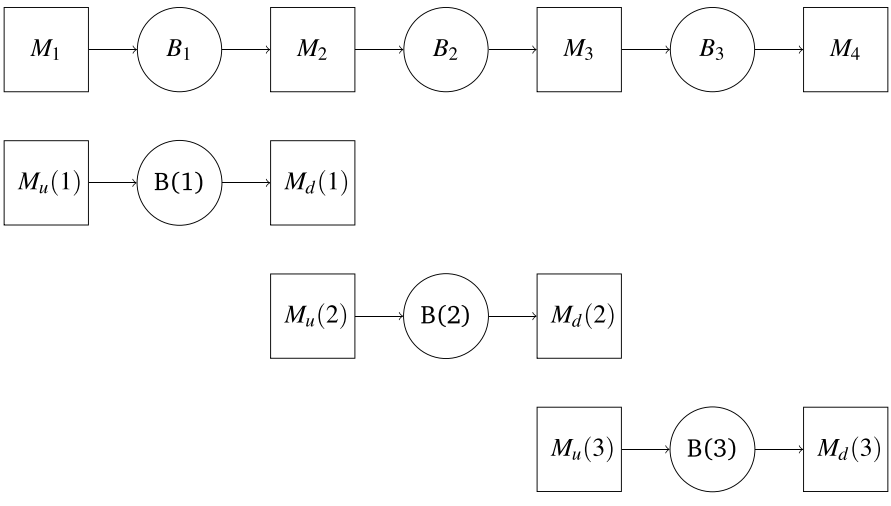


Fig. 2 Flow line with four machines and corresponding decomposition

of virtual up- and downstream machines of virtual line $L(i)$ have to be determined in such a coupled (!) way that the performance measures of the virtual two-machine lines (to be determined as explained in Sect. 4.1) can be used to approximate the performance measures of the original line. As we have $I-1$ virtual lines for a flow line with I machines, each with six unknown parameters as given above, we need a total of $6 \cdot (I-1)$ equations to determine those parameters.

In the original line, we clearly observe *conservation of flow*, i.e. the throughput through all machines is identical in the long run. If a decomposition of a longer line into a set of coupled two-machine lines works as desired, then the throughput of all the virtual lines must also be identical. Therefore, we postulate the following relationship for a flow line with I machines:

$$TP_1 = TP_2 = \cdots = TP_I = TP_u(1) = TP_d(1) = \cdots = TP_d(I-1) \tag{18}$$

As it is not necessary to distinct between $TP_u(i)$ and $TP_d(i)$, we denote $TP(i)$ as the throughput of the virtual line $L(i)$. We use this equality condition not only in the derivation of equations to determine the virtual machine parameters, but also as a termination criterion of the iterative numerical algorithm to solve those equations.

4.2.2 Decomposition equations

As suggested by Choong and Gershwin (1987), we have to derive the following three types of decomposition equations:

1. *Flow Rate–Idle Time* (FRIT) equations deal with the effects of blocking and starving on the throughput. They are also used to propagate the effects of a lack of auxiliary material specific for the flow line model treated in this paper and serve to determine processing rates $\mu_u(i)$ and $\mu_d(i)$ for virtual machines.
2. *Resumption-of-Flow* (ROF) equations model the aggregated repair processes and are used to determine repair rates $r_u(i)$ and $r_d(i)$ of virtual machines.
3. *Interruption-of-Flow* (IOF) equations characterize the aggregated failure processes. They serve to determine failure rates $p_u(i)$ and $p_d(i)$ of virtual machines.

The FRIT equation for a machine i of the original line with availability $e_i = \frac{r_i}{r_i+p_i}$

$$TP_i = \mu_i \cdot e_i \cdot (1 - \mathscr{P}[n_{i-1} = 0 \vee n_i = N_i \vee l_i = 0)] \tag{19}$$

$$\begin{aligned} \approx \mu_i \cdot e_i \cdot (1 &- \mathscr{P}[n_{i-1} = 0 \vee l_i = 0] \\ &- \mathscr{P}[n_i = N_i \vee l_i = 0] \\ &+ \mathscr{P}[l_i = 0]) \end{aligned} \tag{20}$$

is an approximation as it neglects the (typically small) probability of a machine being starved and blocked simultaneously.

We can reformulate this equation to

$$\begin{aligned}\frac{TP_i}{\mu_i \cdot e_i} = (1 - \mathscr{P}[n_{i-1} = 0 \vee l_i = 0]) \\ +(1 - \mathscr{P}[n_i = N_i \vee l_i = 0]) \\ -(1 - \mathscr{P}[l_i = 0]).\end{aligned} \tag{21}$$

We now combine the FRIT for machine i of the original line with the two-machine FRIT equations (13) as well as (16) and set the auxiliary material levels $l_i = l_u(i) = l_d(i-1)$ to find

$$\frac{TP_i}{\mu_i \cdot e_i} = \frac{TP(i-1)}{\mu_d(i-1) \cdot e_d(i-1)} + \frac{TP(i)}{\mu_u(i) \cdot e_u(i)} - (1 - \mathscr{P}[l_i = 0]) \tag{22}$$

If the decomposition approach works as intended, we can use the identities $TP(i-1) = TP(i) = TP(i+1) = TP_i$ to further find

$$\frac{1}{\mu_i \cdot e_i} + \frac{1 - \mathscr{P}[l_i = 0]}{TP(i)} = \frac{1}{\mu_d(i-1) \cdot e_d(i-1)} + \frac{1}{\mu_u(i) \cdot e_u(i)} \tag{23}$$

We obtain an estimate for $\mathscr{P}[l_i = 0]$ of the original line, which we cannot observe, by averaging, i.e. as $0.5 \cdot (\mathscr{P}[i-1;l_d = 0] + \mathscr{P}[i;l_u = 0])$, over the corresponding probabilities stemming from the analysis of the corresponding two-machine lines $L(i-1)$ and $L(i)$, respectively. The probability $\mathscr{P}[l_i = 0]$ of machine i in the original line is a new component of the decomposition that does not exist in the original decomposition as suggested by Choong and Gershwin (1987).

The final version of the new decomposition equations reads as follows:

$$\mu_u(i) = \frac{1}{K_1} \cdot \frac{r_u(i) + p_u(i)}{r_u(i)} \tag{24}$$

with

$$K_1 = \frac{1 - 0.5 \cdot (\mathscr{P}[i-1;l_d = 0] + \mathscr{P}[i;l_u = 0])}{\mathrm{TP}(i-1)} + \frac{1}{e_i \cdot \mu_i} - \frac{1}{e_d(i-1) \cdot \mu_d(i-1)}. \tag{25}$$

Analogously, we obtain the following equation for the downstream machine

$$\mu_d(i) = \frac{1}{K_2} \cdot \frac{r_d(i) + p_d(i)}{r_d(i)} \tag{26}$$

with

$$K_2 = \frac{1 - 0.5 \cdot (\mathscr{P}[i;l_d = 0] + \mathscr{P}[i+1;l_u = 0])}{TP(i+1)} + \frac{1}{e_{i+1} \cdot \mu_{i+1}} - \frac{1}{e_u(i+1) \cdot \mu_u(i+1)} \tag{27}$$

The derivation of both the ROF equations and the IOF equations is documented in the electronic appendix for the sake of completeness as it exhibits some non-obvious

elements that are specific for the case of flow lines with auxiliary material. The final ROF equations for up- and downstream machines are

$$r_u(i) = r_i + K_3 \cdot \frac{\mu_u(i) \cdot r_u(i)}{p_u(i)}, \tag{28}$$

$$r_d(i) = r_{i+1} + K_4 \cdot \frac{\mu_d(i) \cdot r_d(i)}{p_d(i)}, \tag{29}$$

with

$$K_3 = \frac{\mathscr{P}[i-1; n=0, \alpha_u=0, \alpha_d=1, l_u \geq 1, l_d \geq 0](r_u(i-1) - r_i)}{TP(i-1)}, \tag{30}$$

$$K_4 = \frac{\mathscr{P}[i+1; n=N_i), \alpha_u=1, \alpha_d=0, l_u \geq 0, l_d \geq 1](r_d(i+1) - r_{i+1})}{TP(i+1)}. \tag{31}$$

Eventually, the IOF equations are

$$p_u(i) = p_i + K_5 \cdot \mu_u(i) \tag{32}$$

$$p_d(i) = p_{i+1} + K_6 \cdot \mu_d(i) \tag{33}$$

with

$$\begin{aligned} K_5 = & \frac{\mu_d(i-1) \cdot \mathscr{P}[i-1; n=1, \alpha_u=0, \alpha_d=1, l_u \geq 0, l_d \geq 1]}{TP(i-1)} \\ & + \frac{p_u(i-1) \cdot \mathscr{P}[i-1; n=0, \alpha_u=1, \alpha_d=1, l_u \geq 1, l_d \geq 1]}{TP(i-1)} \end{aligned} \tag{34}$$

$$\begin{aligned} K_6 = & \frac{\mu_u(i+1) \cdot \mathscr{P}[i+1; n=N(i+1)-1, \alpha_u=1, \alpha_d=0, l_u \geq 1, l_d \geq 0]}{TP(i+1)} \\ & + \frac{p_d(i+1) \cdot \mathscr{P}[i+1; n=N(i+1), \alpha_u=1, \alpha_d=1, l_u \geq 1, l_d \geq 1]}{TP(i+1)}. \end{aligned} \tag{35}$$

We can solve the decomposition equations (24), (28), and (32) for the desired values of the virtual machine parameters of upstream machines

$$\mu_u(i) = \frac{p_i + r_i}{K_1 r_i + K_3 - K_5} \tag{36}$$

$$r_u(i) = \frac{K_1 p_i r_i + K_3 p_i + K_5 r_i}{K_1 p_i - K_3 + K_5} \tag{37}$$

$$p_u(i) = \frac{K_1 p_i r_i + K_3 p_i + K_5 r_i}{K_1 r_i + K_3 - K_5} \tag{38}$$

and (26), (29), and (33) of downstream machines:

$$\mu_d(i) = \frac{p_{i+1} + r_{i+1}}{K_2 r_{i+1} + K_4 - K_6} \tag{39}$$

$$r_d(i) = \frac{K_2 p_{i+1} r_{i+1} + K_4 p_{i+1} + K_6 r_{i+1}}{K_2 p_{i+1} - K_4 + K_6} \tag{40}$$

$$p_d(i) = \frac{K_2 p_{i+1} r_{i+1} + K_4 p_{i+1} + K_6 r_{i+1}}{K_2 r_{i+1} + K_4 - K_6} \tag{41}$$

Note that in these decomposition equations, the terms K_1, K_3 and K_5 connect a virtual line $L(i)$ to its neighbours $L(i-1)$ and K_2, K_4 and K_6 connect a virtual line $L(i)$ to its neighbours $L(i+1)$, i.e. their respective parameters and performance measures. It is through this connection that iterative parameter updates in the solution process are propagated through the line.

The standard procedure to numerically solve the decomposition equations is to use an iterative procedure that updates the parameters of the virtual two-machine lines ($\mu_u(i)$, $r_u(i)$, $p_u(i)$, and $\mu_d(i)$, $r_d(i)$, $p_d(i)$, respectively) until the throughput of all virtual lines is identical and taken as an estimate of the throughput of the original line, see the pseudo-code description in Section D.

To determine the values of K_1 to K_6 of the virtual two-machine lines analysed in the course of this approach, the throughput as well as the eight steady-state probabilities shown in Table 1 need to be computed. To refer to these parameters, we introduce a shortened notation. For example, the probability $\mathscr{P}[n = 0 \wedge \alpha_u = 1 \wedge \alpha_d = 1 \wedge l_u > 0 \wedge l_d > 0]$ for system i is abbreviated as $\mathscr{P}[i;0,(n,n),11]$. The value before the semicolon gives the number of the analysed

Table 1 Throughput and selected state probabilities of a two-machine line needed in the decomposition approach

Parameter	Short notation	Mean	Min	Max
$TP(i)$ [TU^{-1}]	–	0.6771	0.5100	0.7916
$\mathscr{P}[i;l_u = 0]$	–	0.1201	0.0013	0.3875
$\mathscr{P}[i;l_d = 0]$	–	0.1047	0.0023	0.3104
$\mathscr{P}[i;n = 0, \alpha_u = 0, \alpha_d = 1, l_u \geq 1, l_d \geq 0]$	$\mathscr{P}[i;0,(n,a),01]$	0.0047	5.34×10^{-7}	0.0189
$\mathscr{P}[i;n = 0, \alpha_u = 1, \alpha_d = 1, l_u \geq 1, l_d \geq 1]$	$\mathscr{P}[i;0,(n,n),11]$	0.0124	1.71×10^{-7}	0.0780
$\mathscr{P}[i;n = 1, \alpha_u = 0, \alpha_d = 1, l_u \geq 0, l_d \geq 1]$	$\mathscr{P}[i;1,(a,n),01]$	0.0005	5.92×10^{-8}	0.0017
$\mathscr{P}[i;n = N(i)-1, \alpha_u = 1, \alpha_d = 0, l_u \geq 1, l_d \geq 0]$	$\mathscr{P}[i;N(i)-1,(n,a),10]$	0.0015	2.23×10^{-6}	0.0023
$\mathscr{P}[i;n = N(i), \alpha_u = 1, \alpha_d = 0, l_u \geq 0, l_d \geq 1]$	$\mathscr{P}[i;N(i),(a,n),10]$	0.0227	2.65×10^{-5}	0.0344
$\mathscr{P}[i;n = N(i), \alpha_u = 1, \alpha_d = 1, l_u \geq 1, l_d \geq 1]$	$\mathscr{P}[i;N(i),(n,n),11]$	0.1218	3.33×10^{-5}	0.2240

system. The first entry after the semicolon represents the current number of work pieces n_i in the system already processed by the virtual upstream machine of line i, but not the virtual downstream machine. The information on the material supply of the upstream and the downstream machine follows. No restriction on the material is indicated by "a", whereas "n" means that the machine is not starved of auxiliary material, i.e. $l_i > 0$. The last two numbers indicate whether the machines are operational ("1") or not ("0").

To give an idea of the scale of those parameters, Table 1 additionally shows mean, minimal, and maximal values for the throughput and the relevant probabilities computed with GS. The values are determined over 576 instances of unbalanced small two-machine lines $\mathrm{TML}_{\mathrm{unbal}}^{\mathrm{small}}$. The exact features of this data set are introduced in Chapter 5.1.1, but the basic behaviour can be observed in general, so that we do not go into the test design here. The throughput as well as the examined probabilities vary remarkably. However, some states are unlikely in all cases, e.g. the probability that the buffer is nearly full, the first machine is not starving for material and working while the second failed ($\mathscr{P}[i;N(i)-1,(n,a),10]$) with a maximal probability of less than 1%. Especially the small minimal values show that an accurate prediction of the parameters is important, since an error of one millionth, for example, can double the assumed probability.

4.3 Predicting throughput and selected state probabilities via an artificial neural network (ANN)

Especially for flow lines with large buffer sizes and high order-up-to levels, the state space of the CTMC describing the behaviour of a two-machine line can become huge. In this case, evaluating the two-machine systems takes a relatively long time, even when the Gauss–Seidel (GS) method is used to determine steady-state probabilities. This can make any decomposition approach impractical in which two-machine lines need to be analysed very often and leads to the question of a surrogate model to evaluate the two-machine systems. It is conceivable to use a discrete-event simulation to that end. However, in order to get very precise estimates of rather small state probabilities (see Table 1), extremely long simulations are necessary, which renders this approach impractical. As an alternative and to get a fast and precise evaluation tool, we hence propose to use an Artificial Neural Network (ANN) to predict the throughput and the eight state probabilities of the two-machine lines needed for the decomposition.

Based on training data, ANNs can learn an underlying function from a data set. As shown in Fig. 3, our data set consists of the parameters of two-machine lines as inputs, and the corresponding throughput and the eight state probabilities as outputs. An ANN consists of nodes that process the information, arranged in layers, cf., for example, Goodfellow et al. (2017). In our ANN, the input information is first processed with different nodes. In later layers, the network splits to predict the nine output values separately. As shown in Table 1, the outputs have very different dimensions. For this reason, we use individual nodes for each output. Another option would be to train nine ANN instead of one that splits for the different outputs. This

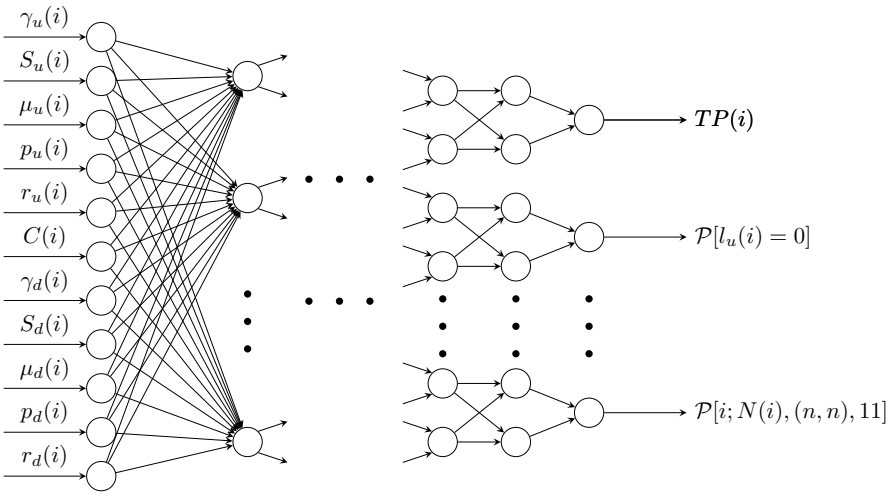


Fig. 3 ANN for the evaluation of two-machine lines

would imply calling nine ANN during each iteration of the decomposition approach, which would increase computational time.

Operating with an ANN to analyse the two-machine model is potentially attractive as the ANN needs to be trained only once and can then be a both extremely fast and relatively accurate evaluation tool.

5 Numerical results

5.1 Evaluation of two-machine lines

5.1.1 Test design

Our analysis starts with a focus on two-machine lines to examine the accuracy of determining the throughput and the corresponding probabilities using the three different evaluation methods, i.e. the discrete-event simulation, the GS method to numerically solve the CTMC, and our ANN. The simulation and the GS method are implemented in C++ and the ANN in Python, using TensorFlow Keras and scikit-learn. The simulation is terminated as the 95 % confidence interval of the simulated throughput reaches a half width of 0.005. In other words, our throughput estimates stemming from the simulation can be considered to be extremely precise. The iterative GS method terminates when the maximum change over all state probabilities is $1{\times}10^{-7}$ from one iteration to the next. We compute all results on an Intel Cascade Lake Xeon Gold 6230N CPU of the Leibniz University Hannover cluster system with a 2.3 GHz processor and 20 MB of cache. For information on the training of the ANN, we refer to Appendix E.

We define two machine types: A *standard machine* S with an effective processing rate $\mu_S^{\text{eff}} = 1$ per time unit (TU)$^{-1}$ and a *bottleneck machine* B with $\mu_B^{\text{eff}} = 0.8$ TU^{-1}. Both machine types have a failure rate $p_i = 0.005$ TU^{-1} and an efficiency $e_i = 0.95$, leading to a repair rate $r_i = \frac{e_i \cdot p_i}{1-e_i} = 0.095$ TU^{-1}. The (raw or original) processing rate $\mu_i = \frac{\mu_i^{\text{eff}}}{e_i}$ is then 1.05 TU^{-1} for the standard machine and 0.84 TU^{-1} for the bottleneck machine. For the analysis of balanced lines, two standard machines are combined. For unbalanced lines, the order of the different machines does not influence the structural behaviour substantially. Therefore, we only examine the case with machine 1 as the standard machine and machine 2 as the bottleneck machine.

Both for balanced and unbalanced lines, we define test sets with a structured variety of parameters concerning the buffers and the auxiliary material, as given in Table 2. Thereby, we distinguish between instances with a smaller and with a larger number of states. For both types of instances, buffer sizes C of 20, 40, 60, or 80 are considered.

Not all combinations of material arrival rates γ_i and order-up-to levels S_i lead to reasonable instances. To eliminate uninteresting borderline cases in which too low combined values of γ_i and S_i essentially starve the entire line, we introduced a quantity denoted as *material ratio* $mr_i = \gamma_i \cdot S_i$. It describes the maximum possible arrival rate of units of auxiliary material. This material ratio obviously is an upper limit of the throughput of that station i and, via the conservation of flow, for the entire line. The material ratio mr_i takes the values 1, 2, 3, and 4 TU^{-1}. For a material ratio mr_i of 1 TU^{-1}, there is, on average, at most one piece of auxiliary material available per unit of time, making it not unlikely that material is missing. For a material ratio of 4 TU^{-1}, the probability of a material shortage is much lower. Auxiliary material arrival rates γ_i of 0.5, 0.65, and 0.8 TU^{-1} define the instances in which auxiliary material arrives very frequently. This results in small order-up-to levels S_i between 2 and 8 and a maximal number of states of $(80+3) \cdot (8+1) \cdot (8+1) \cdot 2 \cdot 2 = 26,892$ for a *small* (!) CTMC for a two-machine line. The larger instances (with a larger state space of the CTMC) stem from auxiliary material arrival rates γ_i assuming much smaller values of 0.05, 0.1, and 0.15 TU^{-1}. These rather infrequent arrivals lead to substantially larger order-up-to levels between 7 and 80. Due to these high levels of S_i, the maximum number of states increases to $(80+3) \cdot (80+1) \cdot (80+1) \cdot 2 \cdot 2 = 2,178,252$.

By applying a full factorial design with all combinations of material supply parameters for both machines, we are considering $4 \cdot 4^2 \cdot 3^2 = 576$ instances in each

Table 2 Definition of small and large test instances

	small instances	large instances
C	[20, 40, 60, 80]	
$mr_i \; \forall \; i = 1,2$	[1, 2, 3, 4]	
$\gamma_i \; \forall \; i = 1,2$	[0.5, 0.65, 0.8]	[0.05, 0.10, 0.15]
$S_i = \lceil \frac{mr_i}{\gamma_i} \rceil \; \forall \; i = 1,2$	2 .. 8	7 .. 80
number of states	828 .. 26,892	5,888 .. 2,178,252

of the four test cases for two-machine lines (TMLs): small balanced $\text{TML}_{\text{bal}}^{\text{small}}$, small unbalanced $\text{TML}_{\text{unbal}}^{\text{small}}$, large balanced $\text{TML}_{\text{bal}}^{\text{large}}$, and large unbalanced $\text{TML}_{\text{unbal}}^{\text{large}}$.

5.1.2 Performance analysis

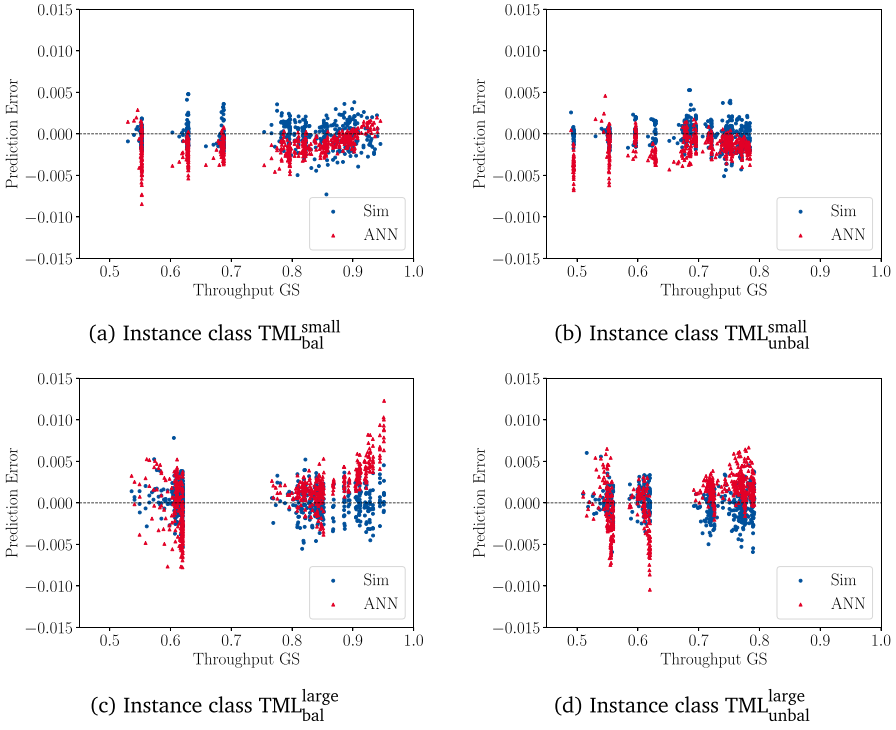
We use the GS method, our self-programmed discrete-event simulation, and our ANN for each of the four instance classes to determine or predict the throughput as well as the eight state probabilities presented in Table 1 that are needed in the decomposition approach. As the iterative numerical GS method can achieve an arbitrary degree of accuracy, we use its results as a reference which we consider to be exact and to validate our simulation, which will be our reference for longer lines. As the general behaviour remains the same for all instance classes, we exemplarily show the throughput prediction error between GS and simulation results as well as between GS and the ANN results for the instances in $\text{TML}_{\text{unbal}}^{\text{large}}$ in Table 3. In addition to the mean absolute error (MAE), we present the minimal and maximal errors over all 576 instances.

The simulation and the ANN both exhibit a very high solution quality. With a mean MAE over all nine parameters of 0.0007 and 0.0008, respectively, they are both suitable to predict the parameters of the two-machine flow line. A closer look shows that the simulation has a slightly better solution quality concerning the throughput, whereas the remaining parameters are very similar for both approaches.

The scatter plots in Fig. 4 show the prediction error of the throughput obtained with simulation and with ANN compared to GS for all instance classes. The x-axis gives the throughput obtained with GS. Each dot represents the error for one instance. As expected, the throughput is lower for the unbalanced lines, i.e. the dots are shifted to the left compared to balanced lines. We can observe that several instances have the same performance, especially for low throughputs. This is because that one parameter, for example, a low material ratio $mr_i = 1\ \text{TU}^{-1}$ for one machine, restricts the whole line so much that the remaining parameters do not

Table 3 Simulation and ANN error for the instance set $\text{TML}_{\text{unbal}}^{\text{large}}$

Parameter	GS - simulation			GS - ANN		
	MAE	$\min_{j \in \mathcal{J}} er_j$	$\max_{j \in \mathcal{J}} er_j$	MAE	$\min_{j \in \mathcal{J}} er_j$	$\max_{j \in \mathcal{J}} er_j$
$TP(i)$ [in TU^{-1}]	0.0013	− 0.0060	0.0059	0.0023	− 0.0105	0.0067
$\mathscr{P}[i;l_u = 0]$	0.0013	− 0.0057	0.0087	0.0013	− 0.0126	0.0073
$\mathscr{P}[i;l_d = 0]$	0.0010	− 0.0047	0.0048	0.0011	− 0.0068	0.0047
$\mathscr{P}[i;0,(n,a),01]$	0.0003	− 0.0020	0.0017	0.0002	− 0.0009	0.0034
$\mathscr{P}[i;0,(n,n),11]$	0.0002	− 0.0015	0.0028	0.0004	− 0.0030	0.0052
$\mathscr{P}[i;1,(a,n),01]$	2.78×10^{-5}	− 0.0002	0.0003	2.05×10^{-5}	− 0.0001	0.0002
$\mathscr{P}[i;N(i)-1,(n,a),10]$	4.28×10^{-5}	− 0.0003	0.0003	3.68×10^{-5}	− 0.0002	0.0001
$\mathscr{P}[i;N(i),(a,n),10]$	0.0008	− 0.0042	0.0036	0.0007	− 0.0025	0.0011
$\mathscr{P}[i;N(i),(n,n),11]$	0.0012	− 0.0072	0.0052	0.0010	− 0.0075	0.0068
Mean	0.0007			0.0008		



(a) Instance class TML_{bal}^{small} (b) Instance class TML_{unbal}^{small}

(c) Instance class TML_{bal}^{large} (d) Instance class TML_{unbal}^{large}

Fig. 4 Scatter plots of errors GS - Simulation and GS - ANN

influence the throughput. Overall, the small prediction errors show that the simulation and the ANN are reasonable procedures for predicting the throughput of a two-machine flow line.

Table 4 presents the computational times for all three procedures. The GS method is relatively fast, compared to the simulation, for the instances with a relatively small number of states, with an average computational time of only 0.1727 and 0.1338 s, respectively. For the instances with a larger number of states, the computational time increases to a mean value of more than 4 s and a maximal value of 76 s. Thereby, the unbalanced systems are solved faster than the balanced systems. The generator

Table 4 Computational times for the two-machine lines in seconds

	GS			Simulation			ANN		
	Mean	Min	Max	Mean	Min	Max	Mean	Min	Max
TML_{bal}^{small}	0.1727	0.0164	1.1192	0.4705	0.1352	1.2891	0.0002	0.0000	0.0011
TML_{unbal}^{small}	0.1338	0.0151	0.7462	0.4613	0.1268	1.5586	0.0001	0.0000	0.0011
TML_{bal}^{large}	4.6914	0.0451	76.0777	0.4755	0.0952	1.3143	0.0001	0.0000	0.0011
TML_{unbal}^{large}	4.0153	0.0522	47.0646	0.4274	0.0899	1.2145	0.0001	0.0000	0.0012

matrix size of the CTMC is irrelevant for the simulation and the ANN, so we do not observe significant differences between small and large instances. The range between minimal and maximal values is comparatively narrow. However, our self-programmed simulation is *faster* than the GS method for large instances. The ANN dominates in all instances with respect to computation times by some orders of magnitude.

For the usage within the decomposition approach and as opposed to the GS method, the ANN seems to be extremely promising due to its short computational times. Via the GS method, however, one can create extremely precise training data for the ANN to "learn" the throughput *and* the probability functions required for the decomposition. These state probabilities listed in Table 1 can be very small and hence very hard to learn from simulation data, which is why the GS method has to be used to generate training data.

5.1.3 Flow line behaviour

Figure 5 presents results on the interaction between repair times and order-up-to levels with respect to the throughput for two-machine lines that are balanced as they operate with stochastically identical machines. The results were obtained via the GS method. We consider two different cases. In both cases, the failure rates are $p_i = 0.005\ \text{TU}^{-1}$. Furthermore, the buffer size C equals 20 in both cases, and the replenishment rate γ_i is chosen to be $0.1\ \text{TU}^{-1}$. In the first (solid blue line) case, the processing rates are $\mu_1 = \mu_2 = 1.05\ \text{TU}^{-1}$. Both machines have the same efficiency of $e_1 = e_2 = 0.95$ with $e_i = \frac{r_i}{r_i + p_i}$ for $i \in \{1, 2\}$, which results in repair rates of $r_1 = r_2 = 0.095\ \text{TU}^{-1}$. In the second (dashed red line) case, we choose the processing rates $\mu_1 = \mu_2 = 1.20\ \text{TU}^{-1}$ combined with an efficiency of $e_1 = e_2 = 0.83$ and hence repair rates of about $r_1 = r_2 = 0.02441\ \text{TU}^{-1}$. In both cases, the isolated processing rates $\mu_i e_i = \mu_i \frac{r_i}{r_i + p_i}$ are approximately $1\ \text{TU}^{-1}$.

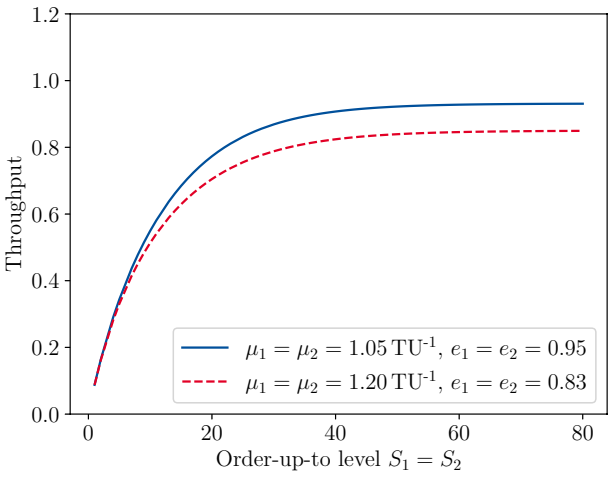


Fig. 5 Throughput depending on the order-up-to level for different machine types

For both lines, the throughput initially rises with a higher order-up-to level $S_1 = S_2$ as material shortages are avoided. For an order-up-to level of 40 units or higher, i.e. a material ratio $mr_i = \gamma_i \cdot S_i \geq 0.1 \cdot 40\,\mathrm{TU}^{-1} = 4\ \ \mathrm{TU}^{-1}$ for machines $i \in \{1, 2\}$, the material availability has almost no influence on the throughput of the line. However, we can observe that for a given order-up-to level, the solid blue line case for machines with a higher efficiency e_i has a higher throughput than the dashed red case one, although both lines have the same effective processing rate $\mu_i e_i$. In the second (broken red line) case, the lower efficiency e_i is due to lower repair rates, i.e. longer repair times. Hence, more blocking and starving occurs, such that a further increase of the throughput can only be achieved via larger buffer sizes C, but not via higher order-up-to levels S_1 and S_2.

In Fig. 6, we analyse the impact of the auxiliary material arrival rate γ_i on the throughput for different order-up-to levels. The analysis is based on a balanced two-machine line with processing rates $\mu_1 = \mu_2 = 1.05\ \mathrm{TU}^{-1}$, efficiencies $e_1 = e_2 = 0.95$, and failure rates $p_1 = p_2 = 0.005\ \mathrm{TU}^{-1}$. The buffer size C equals 20. The solid blue line case in Fig. 6 equals the solid blue case line in Fig. 5 with material arrival rates $\gamma_1 = \gamma_2 = 0.1\,\mathrm{TU}^{-1}$. In the dashed and dotted green case, the material arrival rate $\gamma_1 = \gamma_2$ is increased to $0.15\ \mathrm{TU}^{-1}$. In this case, the throughput is higher for given order-up-to levels as the material arrives more frequently. In the dashed red case with $\gamma_1 = \gamma_2 = 0.05\ \mathrm{TU}^{-1}$, a remarkably lower throughput can be observed, showing again the high relevance and interaction of the material parameters γ_i and S_i. In particular, the figure illustrates how a higher order-up-to level S_i can compensate for a lower material arrival rate γ_i. For example, to obtain a throughput of at least $0.8\ \mathrm{TU}^{-1}$, an order-up-to level $S_1 = S_2 = 15$ is sufficient for $\gamma_1 = \gamma_2 = 0.15\ \mathrm{TU}^{-1}$. In contrast, the order-up-to level needs to be increased to $S_1 = S_2 = 22$ for $\gamma_1 = \gamma_2 = 0.1\ \mathrm{TU}^{-1}$ to obtain the same throughput.

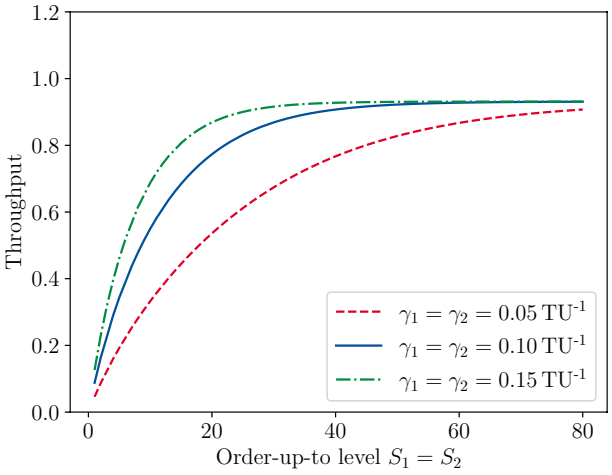


Fig. 6 Throughput depending on the order-up-to level for different inter-arrival rates

5.2 Evaluation of longer lines using the decomposition approach

5.2.1 Test design

To analyse the behaviour of longer flow lines with auxiliary material and to evaluate the decomposition approach's performance, we analysed each line of a large numerical test bed via a discrete-event simulation to create reference values. Then we tried to analyse each such line with two variants of our decomposition algorithm. Like the simulation, the two variants of the decomposition approach were implemented in C++. We distinguish the following variants of the decomposition algorithm: In Dec^{GS}, the two-machine lines are solved practically exactly using the GS method. In Dec^{ANN}, the parameters of the two-machine lines are obtained using the ANN which itself was implemented in Python and is called from the C++ program. Additionally, it would be possible to compute the parameters of the two-machine lines using the simulation. As our former results have shown, the simulation has an comparable accuracy comparable to that of the ANN, but is much slower. For this reason, this approach is not promising.

We consider flow lines with 4, 6, and 10 machines. To show the approach's behaviour in extreme cases and deliberately bring it to its numerical limits in terms of computation times and accuracy, we also present results for 20 and 30 machines. We therefore have five different cases with respect to the line length. For balanced lines, we assume all machines to be of the standard machine type, as introduced in Sect. 5.1.1. For the buffer sizes C_i, the replenishment rates γ_i, and the material ratio mr_i, we again use the values in Table 2. As we now apply the same parameters for all machines of one instance, $4 \cdot 4 \cdot 3 = 48$ instances are considered for each of the five different cases of the line lengths. Therefore, in total $48 \cdot 5 = 240$ instances each are considered in the instance classes of balanced longer lines (LL) with a small number of states $\text{LL}_{\text{bal}}^{\text{small}}$ (stemming from high material arrival rates $\gamma_i \in \{0.5, 0.65, 0.8\}$ in Table 2) and with a large number of states $\text{LL}_{\text{bal}}^{\text{large}}$ (stemming from low material arrival rates $\gamma_i \in \{0.05, 0.1, 0.15\}$ in Table 2).

For the analysis of unbalanced lines, each instance contains exactly one bottleneck machine B, again as defined in Sect. 5.1.1. Preliminary results showed that the position of a clear bottleneck does not influence the system's throughput strongly, so we do not present results for different machines being the bottleneck. Instead, the machine in front of the middle buffer is always the bottleneck. To assure that the bottleneck machine does not slow down the whole system too much, we increase the buffers C_B directly before and behind the bottleneck machine to $C_B = C_S \cdot C^{\text{factor}}$ with $C^{\text{factor}} \in \{1, 1.5, 2.0\}$ and $C_S \in \{20, 40\}$, i.e. we assume that some effort has been made to mitigate the effect of the machine being relatively slow. The material arrival rates $\gamma_i \in \{0.5, 0.65, 0.8\}$ for small instances and $\gamma_i \in \{0.05, 0.1, 0.15\}$ for large instances are the same for all machines. However, the bottleneck machine should also not starve for the auxiliary material. To assure this, the bottleneck machine's material ratio mr_B depends on the standard machine's material ratio mr_S. It is computed as $mr_B = mr_S \cdot mr^{\text{factor}}$ with $mr^{\text{factor}} \in \{1, 1.5, 2.0\}$ and $mr_S \in \{1, 2\}$.

Considering all combinations of mr_S, C_S, mr^{factor}, C^{factor}, and γ_i, we analyse $2 \cdot 2 \cdot 3 \cdot 3 \cdot 3 = 108$ instances for a given length of the flow line, i.e. $108 \cdot 5 = 540$

instances for unbalanced small (LL_{unbal}^{small}) and large (LL_{unbal}^{large}) instances for longer lines with 4, 6, 10, 20, and 30 machines. In the following, we refer to the instance set for a specific length of the flow line by adding the number of machines to the class' name, e.g. LL_{unbal}^{large}-30 for longer unbalanced lines with 30 machines and a large number of states.

5.2.2 Performance analysis

The iterative Algorithm 1 to numerically solve the decomposition equations as documented in Appendix D is essentially a variant of a fixed-point iteration, see (Dallery et al. 1988; Burman 1995). It is supposed to terminate if the throughput differences over the different two-machine lines are below a certain small limit ϵ. The usually observed behaviour of this type of flow line decomposition algorithm is that this limit is reached within 10–20 iterations, which usually takes fractions of a second on current computers. Further iterations then lead to smaller deviations in the throughput results over the different two-machine lines. In our case, however, the situation is more complex for two reasons:

1. Low material arrival rates γ_i typically require large order-up-to levels S_i, which creates CTMC of the two-machine lines with large state spaces and (if the two-machine lines are analysed via the GS method) relatively large computation times, as indicated in Tables 2 and 4. This effect can eliminate the entire speed advantage of the decomposition, relative to a performance evaluation via a discrete-event simulation.
2. For rather long lines (in our case, lines with 20 or 30 stations), convergence issues of Algorithm 1 can sometimes occur as the termination criterion of extremely close throughput estimates from the two-machine line solutions is not met. Instead, from iteration to iteration, throughput estimates begin to oscillate. We conjecture that this is a numerical artefact stemming from our combination of numerical methods.

In our numerical study, we wanted to figure out under which conditions these effects limit the applicability of the entire approach. For this reason, two additional abortion criteria were defined in our decomposition approach, an extreme time limit (TL) of 10 h and an iteration limit (IL) of 1000 iterations. In all cases for lines with 4, 6, and 10 machines, we observed the desired abortion due to the regular abortion criterion of a sufficient degree of accordance over the throughput estimates of the different virtual two-machine lines. However, for cases with 20 or 30 lines, we observed some cases with a termination due to the iteration limit of 1000 iterations or 10 h of computation time. Details are reported in Table 5. The instances with 30 stations led to more convergence issues than those with 20 stations. In addition, the lines with infrequent deliveries, high order-up-to levels, and hence large state spaces of the decomposition's two-machine lines were also more difficult to solve. Finally, the rather rare convergence issues occurred more frequently when the GS method was uses to solve the two-machine lines.

Table 5 Termination by iteration limit (IL) and time limit (TL)

$\|\mathcal{A}\|$	instances	LL^{small}				LL^{large}			
		Dec^{GS}		Dec^{ANN}		Dec^{GS}		Dec^{ANN}	
		IL	TL	IL	TL	IL	TL	IL	TL
20	bal (48)	–	–	–	–	–	6	–	–
	unbal (108)	–	–	–	–	7	–	–	–
30	bal (48)	–	–	–	–	–	14	–	–
	unbal (108)	14	–	8	–	40	7	19	–

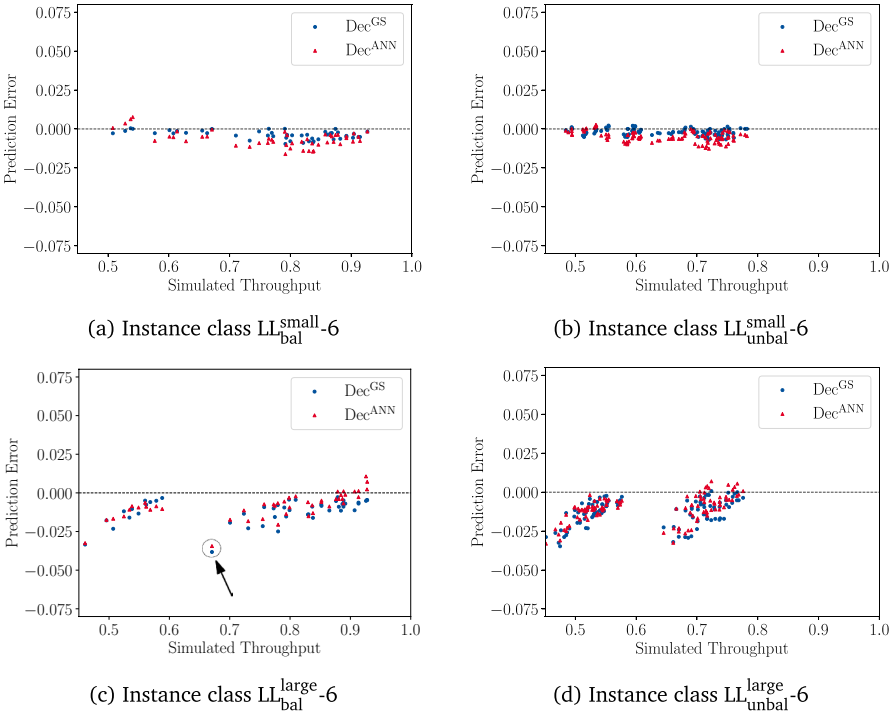
All instances in which the iteration or time limit has been reached for at least one approach are discarded from the further analysis, as we now turn to the results with respect to the accuracy of the approach in the case of a normal termination.

The mean average percentage error (MAPE) of throughput estimates determined via the approaches presented in this paper relative to the simulation results is given in Table 6. As we can consider the GS method applied to the two-machine models to be practically exact, as opposed to the ANN being an approximation itself which is used within the further approximation of the decomposition approach, we expected the Dec^{GS} to be more accurate than the Dec^{ANN}. For the cases with the small state spaces, we find the expected behaviour, but interestingly we do not seem to lose a lot of accuracy by evaluating the two-machine lines with an ANN as opposed to using the GS method. It is interesting that for the systems with the large state spaces, the Dec^{ANN} is even slightly *more* accurate than the Dec^{GS} decomposition. However, in all cases the differences between the throughput accuracy estimations that can be attributed to using either the GS method or the ANN seem to be negligible. This is an important result as it indicates that a suitably trained ANN can indeed be used within a larger decomposition approach!

To look more closely at the decomposition approach's performance, we present the error depending on the simulated throughput in Fig. 7. The behaviour is comparable for the different numbers of machines, so we exemplarily refer to results for lines with 6 machines. Again, we can observe that the prediction error is very small for both approaches. For instances with a small number of states, Dec^{GS} is even nearly meeting the zero line. Both methods tend to underestimate the throughput

Table 6 MAPE in per cent for longer lines

$\|\mathcal{A}\|$	LL^{small}_{bal}		LL^{large}_{bal}		LL^{small}_{unbal}		LL^{large}_{unbal}	
	Dec^{GS}	Dec^{ANN}	Dec^{GS}	Dec^{ANN}	Dec^{GS}	Dec^{ANN}	Dec^{GS}	Dec^{ANN}
4	0.2391	0.5630	0.9917	0.8535	0.1996	0.5059	1.1081	0.9069
6	0.4836	0.9342	1.6809	1.4044	0.3456	0.7605	2.1993	1.9339
10	0.9366	1.4428	2.6352	2.2985	0.6183	1.0705	3.6537	3.3108
20	1.6150	2.1138	3.9104	3.5837	1.0846	1.5485	5.5520	5.1415
30	1.9167	2.4079	4.5339	4.2831	1.3099	1.8887	6.4476	6.0781



(a) Instance class LL^{small}_{bal}-6

(b) Instance class LL^{small}_{unbal}-6

(c) Instance class LL^{large}_{bal}-6

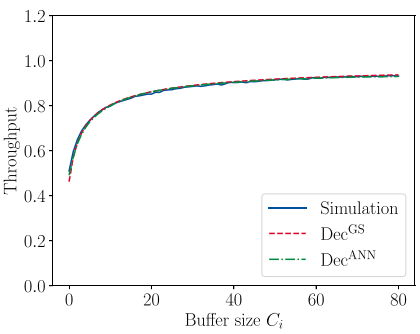
(d) Instance class LL^{large}_{unbal}-6

Fig. 7 Scatter plots of errors Dec^{GS} vs. Simulation and Dec^{ANN} vs. Simulation

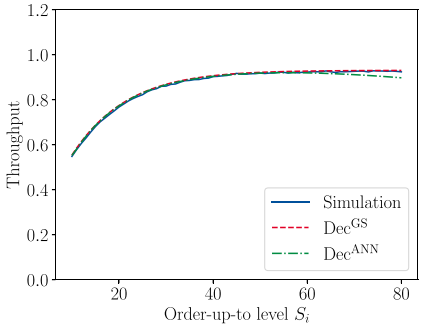
as they mainly lead to negative errors. This is an effect of the decomposition from Sect. 4.2.2 being an approximation.

In Fig. 7c, two adjacent data points are distinguished. They show the throughput estimation errors relative to the simulation result for a particular flow line. Apparently, the differences between the Dec^{GS} and Dec^{ANN} results are relatively small and smaller than the differences to the "true" values stemming from the simulation. We see similar patterns in the other graphs as well. This is also an important result as it leads to an immediate conclusion: It does not seem to be worthwhile to first start with the iterations using the ANN and then, in order to achieve higher accuracy, use the GS method to obtain the final degree of accuracy. It is sufficient to use the GS method to generate training data for the ANN and then use the ANN within the decomposition to quickly and accurately evaluate two-machine lines!

To show the effect of different instance parameters on the performance of the approaches, Fig. 8 gives the throughput quality depending on the buffer size and the order-up-to level, respectively, as determined via the discrete-event simulation and the two decomposition methods Dec^{GS} and Dec^{ANN}. The analysis is based on a balanced 4-machine line with $\gamma_i = 0.1\ TU^{-1}$. For the case of varying buffers, we use an order-up-to level $S_i = 40$, and for the case of varying order-up-to levels, we apply buffer sizes $C_i = 40$. The results obtained with simulation, Dec^{GS}, and Dec^{ANN} are very close in all cases. The effect of varying buffer size or order-up-to level is much higher than that of the different solution procedures. For very high



(a) Throughput depending on buffer size



(b) Throughput depending on order-up-to level

Fig. 8 Throughput of a 4-machine line for the three approaches

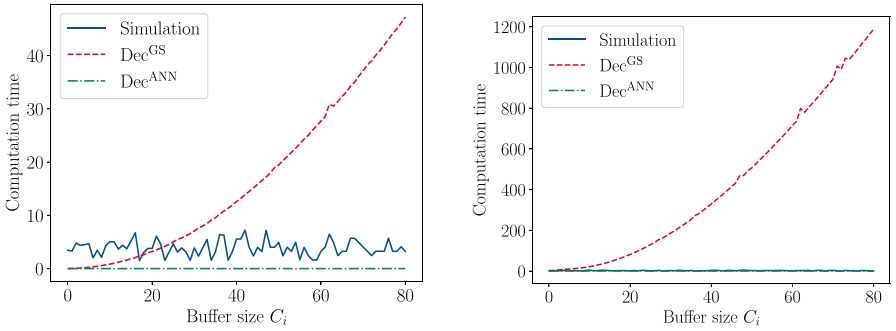
order-up-to levels ($S_i > 60$) the performance of Dec^{ANN} decreases. In these cases, we have a material ratio of $mr_i > 0.1 \cdot 60$ TU$^{-1} = 6$ TU^{-1} which is far above the values of mr_i in the training data set. Therefore, we expect decreasing accuracy of the ANN for these cases. In a practical setting, such high order-up-to levels would not be applied. In summary, the results show an excellent performance of the decomposition approach, independent of which method evaluates the two-machine lines.

The computational times for all approaches are presented in Table 7. Dec^{GS} is much faster for unbalanced lines than for balanced lines. This is in accordance with the typical behaviour of this type of algorithm. However, even though the Dec^{GS} appears to yield useful results, it is *substantially slower* than our (specifically tailored and hence very fast) discrete-event simulation. The comparison between Dec^{ANN} and our discrete-event simulation with respect to speed advantages has to be evaluated with care as the latter terminated only after it had reached a high accuracy with a half width of the simulated throughput confidence interval of 0.005, see Sect. 5.1.1. A less accurate simulation would be substantially faster. The element which eventually makes our approach useful is the combination of the decomposition for the longer lines with the ANN approach to determine results for the virtual two-machine lines. It is *this combination* which yields an extremely fast and still sufficiently accurate evaluation tool. This is particularly true for the lines with up to 10 machines where convergence is achieved quickly and accurate results can be determined in much less time than when using simulation.

For the example of a balanced line with 4 machines, we show in Fig. 9 the influence of the buffer size on the computational time. On the left-hand side, we present the behaviour for small instances with auxiliary material arrival rate $\gamma_i = 0.65$ TU^{-1} and order-up-to level $S_i = 6$. Whereas Dec^{ANN} is the fastest approach, the computational time of the simulation varies a bit but still is on quite a low level. However, the computational time needed for Dec^{GS} shows an substantial growth depending on the buffer size and the number of states, respectively. This gets even clearer for large instances with auxiliary material arrival rate $\gamma_i = 0.1$ TU^{-1} and order-up-to level

Table 7 Computational times for the evaluation of longer lines in seconds

$\lvert\mathcal{A}\rvert$	LL_{bal}^{small}			LL_{bal}^{large}			LL_{unbal}^{small}			LL_{unbal}^{large}		
	Sim	Dec^{GS}	Dec^{ANN}	Sim	Dec^{GS}	Dec^{ANN}	Sim	Dec^{GS}	Dec^{ANN}	Sim	Dec^{GS}	Dec^{ANN}
4	3.12	17.16	0.01	2.57	579.65	0.01	3.18	6.10	0.01	2.69	169.30	0.01
6	6.37	47.27	0.02	5.17	1548.20	0.03	5.53	12.33	0.03	4.35	550.34	0.04
10	15.02	141.15	0.09	12.58	4454.22	0.11	16.42	31.30	0.09	11.95	1457.87	0.15
20	59.43	524.02	0.54	39.33	9074.98	0.71	49.86	105.86	0.71	32.97	4929.44	1.34
30	123.49	986.25	1.56	75.96	11022.26	2.16	117.83	148.16	1.22	69.03	5914.79	2.41



(a) Small state space (b) large state space

Fig. 9 Influence of the buffer size on the computational time

Table 8 Average number of iterations for the evaluation of longer lines

$\lvert\mathcal{A}\rvert$	LL_{bal}^{small}		LL_{bal}^{large}		LL_{unbal}^{small}		LL_{unbal}^{large}	
	Dec^{GS}	Dec^{ANN}	Dec^{GS}	Dec^{ANN}	Dec^{GS}	Dec^{ANN}	Dec^{GS}	Dec^{ANN}
4	10.23	10.60	12.42	12.77	10.44	10.61	13.46	13.73
6	16.71	16.96	20.65	20.71	17.46	17.41	22.74	25.71
10	33.48	32.83	40.79	41.15	39.05	36.52	55.19	59.06
20	91.98	90.48	113.43	121.31	138.33	121.59	228.43	221.96
30	167.87	170.27	210.82	239.71	151.11	136.27	276.67	263.66

$S_i = 40$ on the right-hand side. For a buffer size $C_i = 80$, the evaluation of the corresponding flow lines takes up to 20 min.

We can observe that the average number of iterations is comparable between Dec^{GS} and Dec^{ANN}, see Table 8. The number of machines, i.e. the length of the line, has the primary effect. Compared to the small instances, the number of iterations of large instances rises only slightly. While for small instances the number of iterations is lower for unbalanced lines, for large instances the unbalanced lines require more iterations, again indicating numerical difficulties if the lines become very long.

To sum up, our results show that the performance of the two variants Dec^{GS} and Dec^{ANN} of the decomposition approach presented in this paper is comparable with respect to accuracy. However, Dec^{ANN} overwhelmingly dominates Dec^{GS} with respect to speed, without losing to much accuracy due to using the ANN.

5.2.3 Flow line behaviour

For the analysis of the flow line behaviour, we analyse 6-machine lines. If not stated otherwise, we use inter-arrival rates $\gamma = 0.1$ TU^{-1}, order-up-to levels $S_i = 40$, buffer sizes $C_i = 40$, and standard and bottleneck machines as introduced in Sect. 5.1.1.

Figure 10 presents the influence of the order-up-to level S_i on the throughput for different inter-arrival rates for a balanced line. It is evident that, especially

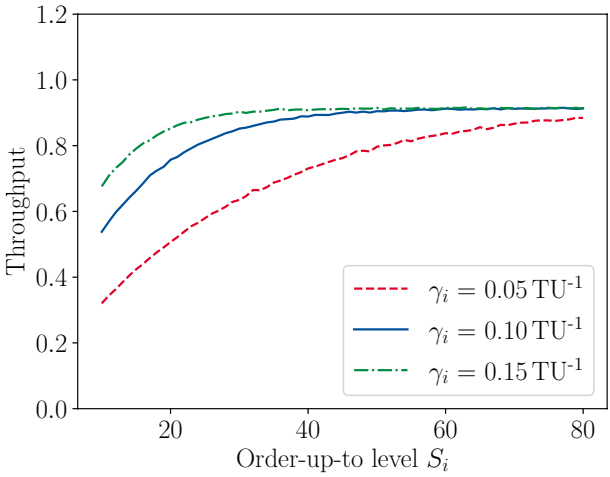


Fig. 10 Throughput depending on the order-up-to level S_i for balanced 6-machine lines

for low inter-arrival rates, the order-up-to level significantly influences the throughput. For higher inter-arrival rates, the order-up-to level is less significant. However, the figure shows that higher order-up-to levels can compensate for lower inter-arrival rates. For example, to obtain a throughput of 0.8 TU^{-1} with an inter-arrival rate of 0.1 TU^{-1}, an order-up-to level $S_i = 24$ is needed. With an inter-arrival rate of 0.15 TU^{-1}, an order-up-to level $S_i = 16$ is sufficient. In both cases, two cases, the resulting material ratio $mr_i = \gamma_i \cdot S_i$ is approximately 2.4 TU^{-1}, which is substantially larger than the throughput of the line of 0.8 TU^{-1}, indicating the need for relatively large auxiliary material storage facilities next to the stations to cope with the numerous aspects of stochastic fluctuations in such a line.

We finally turn to the case of lines with a bottleneck. In Fig. 11, we refer to an unbalanced 6-machine line to observe the influence of the buffer factor C^{factor}, which determines the buffer size directly in front and directly behind the bottleneck machine. Increasing the *bottleneck* buffers has nearly no influence in the case of very small and very large buffers. Only in the range between 5 and 20 buffers can the throughput be perceptibly increased by higher values of C^{factor}. Still, the effect of increasing the bottleneck buffers according to C^{factor} is minor compared to the influence of increasing the standard buffer size C_i of all machines.

The effect of varying the order-up-to levels S_i for different material factors mr^{factor} is given in Fig. 12, again for the case of a line with a bottleneck machine. We observe that the global order-up-to level S_i has a higher effect on the throughput than the material ratio of the bottleneck machine mr^{factor}. There is nearly no difference between values of 1.5 and 2.0, whereas lowering the order-up-to level of the bottleneck machine to that of the remaining machines, e.g. setting mr^{factor} to 1.0, lowers the throughput.

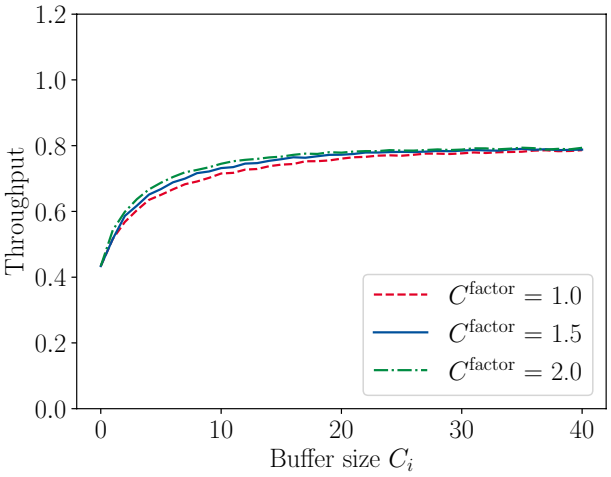


Fig. 11 Throughput depending on the buffer size C_i for unbalanced 6-machine lines

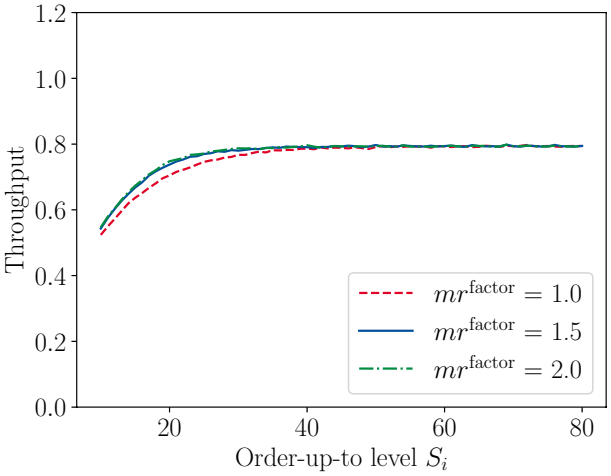


Fig. 12 Throughput depending on the order-up-to level S_i for unbalanced 6-machine lines

6 Conclusion, managerial implications, and further research

We presented a new continuous-time Markov chain model of a flow line with independent and stochastic provisioning of auxiliary material. A decomposition approach led to a numerical algorithm to determine the throughput of such a line. Due to the large state space, solving the two-machine lines turned out to be extremely time-consuming. However, we were able to show how a suitably trained artificial neural network can be used within this decomposition algorithm to evaluate the two-machine lines, such that the complete approach can be used to analyse longer lines quickly and with a high degree of accuracy. The observed

system behaviour of these flow lines completely agrees with our theoretical reasoning.

The managerial implications are twofold: With respect to the system design, we observed the need to have relatively large order-up-to levels for the auxiliary material next to the stations, as material ratios are required that are substantially larger than the throughput of the system. This problem can only be reduced by making the material provisioning more regular, ideally deterministic. For this reason, a deterministic milk-run supply may lead to substantially lower storage requirements for the auxiliary material.

The other managerial implication is method-oriented. Our work showed how it is possible to combine established evaluation methods based on the two-machine flow line decomposition approach with machine learning methods. Appropriately combined, these two technologies can be used to evaluate systems quickly and precisely that can hardly be evaluated if the state space of the two-machine lines becomes too large. We therefore propose an evaluation method which paves the way for integrated formal optimization methods to decide about buffer allocation, order-up-to levels, and auxiliary material delivery frequencies simultaneously.

This last remark points to future research. With this new flow line model and the evaluation method, systematic optimization can be considered as well. One practically interesting problem would then be to optimize the above-mentioned variables of buffer sizes, order-up-to levels, and material provisioning frequencies simultaneously and to assess the benefit from such an integrated optimization. An alternative approach would be to first ignore the auxiliary material altogether and optimize the buffer allocation in isolation. In a further step, and then given the target throughput of the system, appropriate combinations of order-up-to levels and auxiliary material delivery frequencies could be determined.

Funding Open Access funding enabled and organized by Projekt DEAL.

Appendix

A. Definition of virtual machine states

The states $\alpha_u(i) \in \{0, 1\}$ of virtual upstream and $\alpha_d(i) \in \{0, 1\}$ of virtual downstream machine are defined exactly as in Choong and Gershwin (1987). Machine $M_u(i)$ is up if M_i is up and not starved due to $M_u(i-1)$ being down and buffer $i-1$ being empty:

$$\begin{aligned}\{\alpha_u(i,t)=1\} \text{ iff } \{&\alpha_i(t)=1\wedge\\ &\{n(i-1,t)>0\vee\\ &n(i-1,t)=0\wedge\alpha_u(i-1,t)=1\}\}\end{aligned} \tag{42}$$

The crucial element here is the recursive nature of this definition. Machine $M_u(i)$ is down if M_i is down or starved due to $M_u(i-1)$ being down and buffer $i-1$ empty:

$$\{\alpha_u(i,t)=0\} \text{ iff } \{\alpha_i(t)=0\vee n(i-1,t)=0\wedge\alpha_u(i-1,t)=0\} \tag{43}$$

The definition of virtual downstream machine states is analogous.

B. Derivation of the resumption-of-flow equations

The probability of a repair of the virtual upstream machine of line $L(i)$ is the conditional probability of seeing its state change for "0" (i.e. "down") at time t to "1" (i.e. "up") at time $t+\delta t$. It can be determined by inserting the definition of the virtual machine being down and then decomposing on the conditioning event:

$$\begin{aligned}r_u(i)\delta t =&\mathscr{P}[\alpha_u(i,t+\delta t)=1|\alpha_u(i,t)=0]\\ =&\mathscr{P}[\alpha_u(i,t+\delta t)=1|\alpha_i(t)=0\vee\\ &\qquad\alpha_u(i-1,t)=0\wedge n(i-1,t)=0]\\ =&A\cdot B+C\cdot D=A\cdot(1-D)+C\cdot D\\ =&A+(C-A)\cdot D\end{aligned}$$

Terms A to D denote the following conditional probabilities:

$$\begin{aligned}A=&\mathscr{P}[\alpha_u(i,t+\delta t)=1|\alpha_i(t)=0]\\ B=&\mathscr{P}[\alpha_i(t)=0|\alpha_u(i,t)=0]\\ C=&\mathscr{P}[\alpha_u(i,t+\delta t)=1|\alpha_u(i-1,t)=0\wedge n(i-1,t)=0]\\ D=&\mathscr{P}[\alpha_u(i-1,t)=0\wedge n(i-1,t)=0|\alpha_u(i,t)=0]\end{aligned}$$

Terms A and C denote repairs of machine i of the original line and virtual upstream machine $M_u(i-1)$ of line $L(i)$, respectively:

$$\begin{aligned}A=&\mathscr{P}[\alpha_u(i,t+\delta t)=1|\alpha_i(t)=0]\\ =&r_i\cdot\delta t\end{aligned}$$

$$\begin{aligned}C=&\mathscr{P}[\alpha_u(i,t+\delta t)=1|\alpha_u(i-1,t)=0\wedge n(i-1,t)=0]\\ =&r_u(i-1)\cdot\delta t\end{aligned}$$

It remains to determine term D. To this end, several implications are used:

- Condition $\alpha_u(i,t)=0$ implies conditions $n(i,t)<N(i)$ and $l(i,t)>0$ as a machine cannot fail while being blocked or starved for main products or auxiliary material.

- Conditions $\alpha_u(i-1,t)=0 \wedge n(i-1,t)=0$ imply condition $\alpha_u(i,t)=0$ by definition of virtual machine states.
- Conditions $\alpha_u(i-1,t)=0 \wedge n(i-1,t)=0$ imply conditions $\alpha_d(i-1,t)=1$ because the starved downstream machine cannot fail and $l_u(i-1,t)>0$ because the upstream machine can only fail if it is not starving.

The definition of conditional probability and identity (15) together with the implications introduced above yield:

$$\begin{aligned}
D =& \mathscr{P}[\alpha_u(i-1,t)=0 \wedge n(i-1,t)=0 | \alpha_u(i,t)=0] \\
=& \frac{\mathscr{P}[\alpha_u(i-1,t)=0 \wedge n(i-1,t)=0 \wedge \alpha_u(i,t)=0]}{\mathscr{P}[\alpha_u(i,t)=0]} \\
=& \frac{\mathscr{P}[\alpha_u(i-1,t)=0 \wedge n(i-1,t)=0 \wedge \alpha_d(i-1,t)=1]}{\mathscr{P}[\alpha_u(i,t)=0 \wedge n(i,t)<N(i) \wedge l_u(i,t)\geq 1]} \\
=& \frac{\mathscr{P}[\alpha_u(i-1,t)=0 \wedge l_u(i-1)>0 \wedge n(i-1,t)=0 \wedge \alpha_d(i-1,t)=1]}{\mathscr{P}[\alpha_u(i,t)=0 \wedge n(i,t)<N(i) \wedge l_u(i,t)\geq 1]} \\
=& \frac{\mathscr{P}[i-1;0,(n,a),01]}{TP(i)\cdot p_u(i)} \cdot \mu_u(i)\cdot r_u(i)
\end{aligned}$$

We use the shorthand notation for state probabilities of virtual two-machine lines introduced in the fourth entry of Table 1 for the last equality. Bringing everything together, we find:

$$\begin{aligned}
r_u(i)\delta t &= A + (C-A)\cdot D \\
&= r_i\cdot\delta t + \Big((r_u(i-1)-r_i)\cdot\delta t\cdot\mathscr{P}[i-1;0,(n,a),01]]\Big)\frac{\mu_u(i)\cdot r_u(i)}{TP(i)\cdot p_u(i)}
\end{aligned}$$

Dividing by δt, using conservation of flow $TP(i-1)=TP(i)$, and rearranging the terms, we eventually have the following form

$$r_u(i) = r_i + K_3\cdot\frac{\mu_u(i)\cdot r_u(i)}{p_u(i)}$$

with

$$K_3 = \frac{\mathscr{P}[i-1;0,(n,a),01](r_u(i-1)-r_i)}{TP(i-1)}.$$

For the downstream machine, analogous arguments and index transformation lead to

$$r_d(i) = r_{i+1} + K_4\cdot\frac{\mu_d(i)\cdot r_d(i)}{p_d(i)}$$

with

$$K_4 = \frac{\mathscr{P}[i+1;N(i+1),(a,n),10](r_d(i+1)-r_{i+1})}{TP(i+1)}.$$

C. Derivation of the interruption-of-flow equations

We present the derivation of the IOF equations for the virtual upstream machine $M_u(i)$, again using the definition of virtual machine states. The first step is to break the conditional probability of a failure apart by considering the mutually exclusive ways how at time $t+\delta t$ the machine can be down and adding the respective probabilities:

$$\begin{aligned}
p_u(i)\delta t &= \mathscr{P}[\alpha_u(i,t+\delta t)=0 | \alpha_u(i,t)=1 \wedge n(i,t)<N(i) \wedge l(i,t)>0] \\
&= \mathscr{P}[\alpha_i(t+\delta t)=0 | \alpha_u(i,t)=1 \wedge n(i,t)<N(i) \wedge l(i,t)>0] \\
&\quad + \mathscr{P}[\alpha_u(i-1,t+\delta t)=0 \wedge n(i-1,t+\delta t)=0 | \\
&\qquad \alpha_u(i,t)=1 \wedge n(i,t)<N(i) \wedge l(i,t)>0] \\
&= p_i \cdot \delta t + \mathscr{P}[\alpha_u(i-1,t+\delta t)=0 \wedge n(i-1,t+\delta t)=0 | \\
&\qquad \alpha_u(i,t)=1 \wedge n(i,t)<N(i) \wedge l(i,t)>0]
\end{aligned}$$

A further decomposition on the conditioning event for the second summand yields

$$\begin{aligned}
&\mathscr{P}[\alpha_u(i-1,t+\delta t)=0 \wedge n(i-1,t+\delta t)=0 | \\
&\qquad \alpha_u(i,t)=1 \wedge n(i,t)<N(i) \wedge l(i,t)>0]
\end{aligned} \tag{44}$$

$$\begin{aligned}
=&\mathscr{P}[\alpha_u(i-1,t+\delta t)=0 \wedge n(i-1,t+\delta t)=0 | \\
&\alpha_u(i,t)=1 \wedge n(i,t)<N(i) \wedge l(i,t)>0 \wedge \\
&\quad \{n(i-1,t)>1 \vee \\
&\quad n(i-1,t)=1 \wedge \alpha_u(i-1,t)=1 \vee \\
&\quad n(i-1,t)=1 \wedge \alpha_u(i-1,t)=0 \vee \\
&\quad n(i-1,t)=0 \wedge \alpha_u(i-1,t)=1 \wedge l(i-1,t)>0 \vee \\
&\quad n(i-1,t)=0 \wedge \alpha_u(i-1,t)=1 \wedge l(i-1,t)=0\}] \\
=&A \cdot B + C \cdot D + E \cdot F + G \cdot H + I \cdot J
\end{aligned} \tag{45}$$

with terms A to J to be analysed below based on a decomposition on five mutually exclusive and collectively exhaustive events in Eq. (45).

We first note that more than one event required during an interval of length δt for transitions A and C related to the first two conditioning events in Eq. (45):

$$\begin{aligned}
A =&\mathscr{P}[\alpha_u(i-1,t+\delta t)=0 \wedge n(i-1,t+\delta t)=0 | \\
&\qquad \alpha_u(i,t)=1 \wedge n(i,t)<N(i) \wedge l(i,t)>0 \wedge \{n(i-1,t)>1\}] \\
=&\left(\mu_d(i-1)\delta t\right)^2 + ... + o(\delta t) = o(\delta t)
\end{aligned}$$

Since A is irrelevant given $\lim_{\delta t \to 0} \frac{o(\delta t)}{\delta t} = 0$, there is no need to determine B.

In a similar way, we establish that C is of order $o(\delta t)$:

$$\begin{aligned} C =& \mathscr{P}[\alpha_u(i-1,t+\delta t)=0 \wedge n(i-1,t+\delta t)=0| \\ & \alpha_u(i,t)=1 \wedge n(i,t)<N(i) \wedge l(i,t)>0 \wedge \\ & \{n(i-1,t)=1 \wedge \alpha_u(i-1,t)=1\}] \\ =& \mu_d(i-1)\delta t \cdot p_u(i-1)\delta t + ... + o(\delta t) = o(\delta t) \end{aligned}$$

Since C is hence also irrelevant, there is no need to determine D.

For the transition leading to the conditional probability related to the third conditioning event in Eq. (45), the last work piece available at the downstream machine $M_d(i-1)$ needs to be completed

$$\begin{aligned} E =& \mathscr{P}[\alpha_u(i-1,t+\delta t)=0 \wedge n(i-1,t+\delta t)=0| \\ & \alpha_u(i,t)=1 \wedge n(i,t)<N(i) \wedge l(i,t)>0 \wedge \\ & \alpha_u(i-1,t)=0 \wedge n(i-1,t)=1] \\ =& \mu_d(i-1)\delta t \end{aligned}$$

and likewise for the fourth conditioning event in Eq. (45), a failure of the virtual upstream machine $M_u(i-1)$ is required:

$$\begin{aligned} G =& \mathscr{P}[\alpha_u(i-1,t+\delta t)=0 \wedge n(i-1,t+\delta t)=0| \\ & \alpha_u(i,t)=1 \wedge n(i,t)<N(i) \wedge l(i,t)>0 \wedge \\ & \alpha_u(i-1,t)=1 \wedge n(i-1,t)=0 \wedge l(i-1,t)>0] \\ =& p_u(i-1)\delta t \end{aligned}$$

For the fifth and final conditioning event in Eq. (45), as machine M_{i-1} is lacking auxiliary material, i.e. we have $l(i-1,t)=0$, we cannot have a failure and hence have

$$\begin{aligned} I =& \mathscr{P}[\alpha_u(i-1,t+\delta t)=0 \wedge n(i-1,t+\delta t)=0| \\ & \alpha_u(i,t)=1 \wedge n(i,t)<N(i) \wedge l(i,t)>0 \wedge \\ & \alpha_u(i-1,t)=1 \wedge n(i-1,t)=0 \wedge l(i-1,t)=0] \\ =& 0. \end{aligned}$$

Note that $\alpha_u(i,t)=1$ implies $\alpha_i(t)=1$, which together with $n(i,t)<N(i)$ is the definition of $\alpha_d(i-1,t)=1$. We furthermore postulated $l_i(t)=l_u(i,t)=l_d(i-1,t)$ before and therefore find

$$\begin{aligned} F =& \mathscr{P}[\alpha_u(i,t)=1 \wedge n(i,t)<N(i) \wedge l(i,t)>0 \wedge \alpha_u(i-1,t)=0 \wedge n(i-1,t)=1| \\ & \alpha_u(i,t)=1 \wedge n(i,t)<N(i) \wedge l(i,t)>0] \\ =& \frac{\mathscr{P}[\alpha_u(i,t)=1 \wedge n(i,t)<N(i) \wedge l(i,t)>0 \wedge \alpha_u(i-1,t)=0 \wedge n(i-1,t)=1]}{\mathscr{P}[\alpha_u(i,t)=1 \wedge n(i,t)<N(i) \wedge l(i,t)>0]} \\ =& \frac{\mathscr{P}[n(i-1,t)=1 \wedge \alpha_u(i-1,t)=0 \wedge \alpha_d(i-1,t)=1 \wedge l_d(i-1,t)>0]}{\mathscr{P}[\alpha_u(i,t)=1 \wedge n(i,t)<N(i) \wedge l(i,t)>0]} \\ =& \mathscr{P}[i-1;1,(a,n),01] \cdot \frac{\mu_u(i)}{TP(i)} \end{aligned}$$

Note that we used the standard two-machine FRIT for which we already established $\mathrm{PR}(i) = \mu_u(i) \cdot \mathscr{P}[\alpha_u(i,t) = 1 \wedge n(i,t) < N(i)]$ to reformulate the above denominator! In a similar way, we find

$$\begin{aligned} H =& \mathscr{P}[\alpha_u(i,t) = 1 \wedge n(i,t) < N(i) \wedge l(i,t) > 0 \wedge \alpha_u(i-1,t) = 1 \wedge n(i-1,t) = 0 \wedge l(i-1,t) > 0 | \\ & \alpha_u(i,t) = 1 \wedge n(i,t) < N(i) \wedge l(i,t) > 0] \\ =& \frac{\mathscr{P}[\alpha_u(i,t) = 1 \wedge n(i,t) < N(i) \wedge l(i,t) > 0 \wedge \alpha_u(i-1,t) = 1 \wedge n(i-1,t) = 0 \wedge l(i-1,t) > 0]}{\mathscr{P}[\alpha_u(i,t) = 1 \wedge n(i,t) < N(i) \wedge l(i,t) > 0]} \\ =& \frac{\mathscr{P}[n(i-1,t) = 0 \wedge \alpha_u(i-1,t) = 1 \wedge \alpha_d(i-1,t) = 1 \wedge l_u(i-1,t) > 0 \wedge l_d(i-1,t) > 0]}{\mathscr{P}[\alpha_u(i,t) = 1 \wedge n(i,t) < N(i) \wedge l(i,t) > 0]} \\ =& \mathscr{P}[i-1;0,(n,n),11] \frac{\mu_u(i)}{TP(i)} \end{aligned}$$

Putting everything together, we find

$$p_u(i)\delta t = p_i \delta t + o(\delta t) \cdot B + o(\delta t) \cdot D + E \cdot F + G \cdot H + 0 \cdot J$$

After dividing by δt and taking limits for $\delta t \to 0$, we find

$$\begin{aligned} p_u(i) =& p_i + \lim_{\delta t \to 0} \frac{o(\delta t)}{\delta t} \cdot B + \lim_{\delta t \to 0} \frac{o(\delta t)}{\delta t} \cdot D \\ & + \mu_d(i-1) \cdot \mathscr{P}[i-1;1,(a,n),01] \cdot \frac{\mu_u(i)}{TP(i)} + p_u(i-1) \cdot \mathscr{P}[i-1;0,(n,n),11] \frac{\mu_u(i)}{TP(i)} \\ =& p_i + \Big(\mu_d(i-1) \cdot \mathscr{P}[i-1;1,(a,n),01] + p_u(i-1) \cdot \mathscr{P}[i-1;0,(n,n),11] \Big) \cdot \frac{\mu_u(i)}{TP(i)}. \end{aligned}$$

We finally have our two IOFs

$$\begin{aligned} p_u(i) =& p_i + K_5 \cdot \mu_u(i) \\ p_d(i) =& p_{i+1} + K_6 \cdot \mu_d(i) \end{aligned}$$

with

$$\begin{aligned} K_5 &= \frac{\mu_d(i-1) \cdot \mathscr{P}[i-1;1,(a,n),01] + p_u(i-1) \cdot \mathscr{P}[i-1;0,(n,n),11]}{TP(i-1)} \\ K_6 &= \frac{\mu_u(i+1) \cdot \mathscr{P}[i+1;N(i+1)-1,(n,a),10] + p_d(i+1) \cdot \mathscr{P}[i+1;N(i+1),(n,n),11]}{TP(i+1)}. \end{aligned}$$

D. Iterative solution of the decomposition equations

Algorithm 1 is used to solve the decomposition equations numerically.

Initialize $\varepsilon := 0.00001$;
for *all virtual lines* $i = 1,2,...,I-1$ **do**
  Initialize $p_u(i) := p_i,\ r_u(i) := r_i,\ \mu_u(i) := \mu_i, \lambda_u(i) := \lambda_i, S_u(i) := S_i$ and $p_d(i) := p_{i+1},\ r_d(i) := r_{i+1}, \mu_d(i) := \mu_{i+1}, \lambda_d(i) = \lambda_{i+1}, S_d(i) = S_{i+1}, C(i) = C_i$;
  Determine steady-state probabilities $\mathscr{P}(i;n,\alpha_u,\alpha_d,l_u,l_d)$ as well as throughput $TP(i)$;
end
while $|TP(1) - TP(I-1)| > \varepsilon$ **do**
  for $i = 2,3,...,I-1$ **do**
    Forward pass:
    Solve Equations (36) to (38) for line i to determine new values for rates $\mu_u(i), r_u(i)$, and $p_u(i)$;
    Update steady-state probabilities $\mathscr{P}(i;n,\alpha_u,\alpha_d,l_u,l_d)$ and throughput $TP(i)$;
  end
  for $i = I-2, I-3,...,1$ **do**
    Backward pass:
    Solve Equations (39) to (41) for line i to determine new values for rates $\mu_d(i), r_d(i)$, and $p_d(i)$;
    Update steady-state probabilities $\mathscr{P}(i;n,\alpha_u,\alpha_d,l_u,l_d)$ and throughput $TP(i)$.
  end
end

Algorithm 1 Decomposition algorithm

E. Training of the ANN for two-machine lines

We trained one ANN for small instance sets (ANN^{small}) of both balanced and unbalanced two-machine lines and one further ANN for large instance sets (ANN^{large}), again with balanced and unbalanced two-machine lines. To not be limited to the specific values from our tests sets as shown in Table 2, we created training and validation data sets with broader ranges between the lower bound (LB) and the upper bound (UB), see Table 9. The grey values are not inputs to the ANN, but are needed for generating the data set. We drew each of the parameters μ_i, p_i, e_i, C, γ_i and mr_i from the specified range according to a uniform distribution and calculate r_i and S_i. To systematically cover the parameter space, we used orthogonal Latin hypercube sampling, see (Owen 1992; Tang 1993).

We evaluated the flow lines with GS to compute the throughput and the state probabilities. For each ANN, the ANN^{small} and the ANN^{large}, we generated about 100,000 samples for the training data set and another 10,000 samples for a validation data set. We terminated the training process after 100 epochs.

In the training process of the ANN, we minimized the mean squared error (MSE) of the outputs compared to the values obtained by GS. The sum of the MSE of all individual outputs defines the training error. Since the different output values have different dimensions, we calibrated the inputs and outputs to values between zero and one. This way, we assured that all outputs are equally important during the training process.

Table 9 Parameter ranges of the training data sets for the small and large instance class

Parameter	Small		Large	
	LB	UB	LB	UB
μ_i	0.8	1.2	0.8	1.2
p_i	0.001	0.05	0.001	0.05
r_i	0.009	4.95	0.009	4.95
e_i	0.9	0.99	0.9	0.99
C	0	80	0	80
γ_i	0.5	0.8	0.05	0.15
S_i	2	8	7	80
mr_i	1	4	1	4

The results show that the training and validation errors of the calibrated data are very close to each other. This indicates that the ANN neither over- nor underfits the training data set. Table 10 breaks down the training and validation error for the back-transformed prediction of each output. We observe small MSE for all output values in training and validation for both ANN.

Table 10 Training and validation error for all outputs after 100 epochs

Parameter	$\text{MSE}^{\text{small}}_{\text{Train}}$	$\text{MSE}^{\text{small}}_{\text{Val}}$	$\text{MSE}^{\text{large}}_{\text{Train}}$	$\text{MSE}^{\text{large}}_{\text{Val}}$
$TP(i)$ [TU^{-1}]	3.76×10^{-6}	3.98×10^{-6}	3.81×10^{-6}	3.92×10^{-6}
$\mathscr{P}[i;l_u = 0]$	4.81×10^{-7}	5.23×10^{-7}	2.06×10^{-6}	2.21×10^{-6}
$\mathscr{P}[i;l_d = 0]$	7.76×10^{-7}	8.70×10^{-7}	7.71×10^{-7}	8.29×10^{-7}
$\mathscr{P}[i;0,(n,a),01]$	2.52×10^{-7}	2.94×10^{-7}	2.29×10^{-7}	2.49×10^{-7}
$\mathscr{P}[i;0,(n,n),11]$	6.36×10^{-7}	7.74×10^{-7}	7.62×10^{-7}	8.27×10^{-7}
$\mathscr{P}[i;1,(a,n),01]$	2.08×10^{-7}	2.67×10^{-7}	2.37×10^{-7}	2.48×10^{-7}
$\mathscr{P}[i;N(i)-1,(n,a),10]$	6.46×10^{-7}	6.81×10^{-7}	9.21×10^{-7}	9.79×10^{-7}
$\mathscr{P}[i;N(i),(a,n),10]$	8.54×10^{-10}	9.94×10^{-10}	1.15×10^{-9}	1.23×10^{-9}
$\mathscr{P}[i;N(i),(n,n),11]$	1.04×10^{-9}	1.12×10^{-9}	1.94×10^{-9}	1.93×10^{-9}
Sum	6.76×10^{-6}	7.39×10^{-6}	8.79×10^{-6}	9.27×10^{-6}

References

Alnahhal M, Noche B (2015) Dynamic material flow control in mixed model assembly lines. Comput Ind Eng 85:110–119

Baller R, Hage S, Fontaine P, Spinler S (2020) The assembly line feeding problem: an extended formulation with multiple line feeding policies and a case study. Int J Prod Econ 222:107489. https://doi.org/10.1016/j.ijpe.2019.09.010

Becker C, Scholl A (2006) A survey on problems and methods in generalized assembly line balancing. Eur J Oper Res 168(3):694–715

Boysen N, Emde S (2014) Scheduling the part supply of mixed-model assembly lines in line-integrated supermarkets. Eur J Oper Res 239(3):820–829

Boysen N, Schulze P, Scholl A (2022) Assembly line balancing: what happened in the last fifteen years? Eur J Oper Res 301(3):797–814

Bukchin Y, Meller RD (2005) A space allocation algorithm for assembly line components. IIE Trans 37(1):51–61. https://doi.org/10.1080/07408170590516854

Burman MH (1995) New results in flow line analysis. Ph. D. thesis, Massachusetts Institute of Technology

Chang Q, Pan C, Xiao G, Biller S (2013) Integrated modeling of automotive assembly line with material handling. J Manuf Sci Eng 135(1):011018. https://doi.org/10.1115/1.4023365

Choong YF, Gershwin SB (1987) A decomposition method for the approximate evaluation of capacitated transfer lines with unreliable machines and random processing times. IIE Trans 19(2):150–159

Dallery Y, David R, Xi XL (1988) An efficient algorithm for analysis of transfer lines with unreliable machines and finite buffers. IIE Trans 20(3):280–283. https://doi.org/10.1080/07408178808966181

Dallery Y, Gershwin SB (1992) Manufacturing flow line systems: a review of models and analytical results. Queueing Syst 12(1–2):3–94

Delice Y, Aydoğan EK, Himmetoğlu S, Özcan U (2023) Integrated mixed-model assembly line balancing and parts feeding with supermarkets. CIRP J Manuf Sci Technol 41:1–18

Emde S, Boysen N (2012) Optimally routing and scheduling tow trains for JIT-supply of mixed-model assembly lines. Eur J Oper Res 217(2):287–299

Emde S, Fliedner M, Boysen N (2012) Optimally loading tow trains for just-in-time supply of mixed-model assembly lines. IIE Trans 44(2):121–135

Faccio M, Gamberi M, Persona A (2013) Kanban number optimisation in a supermarket warehouse feeding a mixed-model assembly system. Int J Prod Res 51(10):2997–3017

Gershwin SB (1987) An efficient decomposition method for the approximate evaluation of tandem queues with finite storage space and blocking. Oper Res 35(2):291–305

Goodfellow I, Bengio Y, Courville A (2017) Deep learning. Adaptive computation and machine learning series. MIT Press, Cambridge

Helber S (1998) Decomposition of unreliable assembly/disassembly networks with limited buffer capacity and random processing times. Eur J Oper Res 109(1):24–42. https://doi.org/10.1016/S0377-2217(97)00166-5

Helber S (1999) Performance analysis of flow lines with non-linear flow of material. Lecture notes in economics and mathematical systems. Springer

Helber S (2005) Analysis of flow lines with cox-2-distributed processing times and limited buffer capacity. OR Spectr 27:221–242

Hudson S, McNamara T, Shaaban S (2015) Unbalanced lines: where are we now? Int J Prod Res 53(6):1895–1911

Li J (2005) Overlapping decomposition: a system-theoretic method for modeling and analysis of complex manufacturing systems. IEEE Trans Autom Sci Eng 2(1):40–53

Li J, Blumenfeld DE, Alden JM (2006) Comparisons of two-machine line models in throughput analysis. Int J Prod Res 44(7):1375–1398

Li J, Meerkov SM (2009) Production systems engineering. Springer, Boston

Manitz M (2015) Analysis of assembly/disassembly queueing networks with blocking after service and general service times. Ann Oper Res 226(1):417–441. https://doi.org/10.1007/s10479-014-1639-x

Mindlina J, Tempelmeier H (2022) Performance analysis and optimisation of stochastic flow lines with limited material supply. Int J Prod Res 60(17):5293–5306

Nourmohammadi A, Eskandari H, Fathi M (2019) Design of stochastic assembly lines considering line balancing and part feeding with supermarkets. Eng Optim 51(1):63–83

Owen AB (1992) Orthogonal arrays for computer experiments, integration and visualization. Stat Sin 2(2):439–452

Papadopoulos CT, Li J, O'Kelly ME (2019) A classification and review of timed Markov models of manufacturing systems. Comput Ind Eng 128:219–244

Papadopoulos CT, O'Kelly ME, Vidalis MJ, Spinellis D (2009) Analysis and design of discrete part production lines. Springer, Berlin

Papadopoulos H, Heavey C (1996) Queueing theory in manufacturing systems analysis and design: a classification of models for production and transfer lines. Eur J Oper Res 92(1):1–27

Sachs FE, Helber S, Kiesmüller G (2022) Evaluation of unreliable flow lines with limited buffer capacities and spare part provisioning. Eur J Oper Res 302(2):544–559

Scholl A, Becker C (2006) State-of-the-art exact and heuristic solution procedures for simple assembly line balancing. Eur J Oper Res 168(3):666–693

Südbeck I, Mindlina J, Schnabel A, Helber S (2023) Using recurrent neural networks for the performance analysis and optimization of stochastic milkrun-supplied flow lines

Tancrez JS (2020) A decomposition method for assembly/disassembly systems with blocking and general distributiions. Flex Serv Manuf J 32:272–296. https://doi.org/10.1007/s10696-019-09332-z

Tang B (1993) Orthogonal array-based Latin hypercubes. Biometrika 88(424):1392–1397. https://doi.org/10.2307/2291282

Tempelmeier H, Bürger M (2001) Performance evaluation of unbalanced flow lines with general distributed processing times, failures and imperfect production. IIE Trans 33(4):293–302

Weiss S, Matta A, Stolletz R (2017) Optimization of buffer allocations in flow lines with limited supply. IISE Trans 50(3):191–202. https://doi.org/10.1080/24725854.2017.1328751

Yan CB, Zhao Q, Huang N, Xiao G, Li J (2010) Formulation and a simulation-based algorithm for line-side buffer assignment problem in systems of general assembly line with material handling. IEEE Trans Autom Sci Eng 7(4):902–920. https://doi.org/10.1109/TASE.2010.2046892