Schmalenbachs Zeitschrift für betriebswirtschaftliche Forschung
https://doi.org/10.1007/s41471-024-00183-5

ORIGINAL ARTICLE

Using Recurrent Neural Networks for the Performance Analysis and Optimization of Stochastic Milkrun-Supplied Flow Lines

Insa Südbeck · Julia Mindlina · André Schnabel · Stefan Helber

Received: 22 May 2023 / Accepted: 4 April 2024

Abstract Long-term throughput, as a key performance indicator of a stochastic flow line, is affected by numerous parameters describing the features of the flow line, such as processing time and buffer size. Fast and accurate evaluation methods for a given set of values for those parameters are a prerequisite to systematically optimize such a flow line. In this paper, we consider the case of a flow line with random processing times, limited buffer capacities and so-called milkruns that supply the machines with material parts that are required to perform, e.g., assembly operations on workpieces. In such a system, shortages in the supply of material parts can limit the performance of the flow line. Up to now, there are no accurate analytical approaches to quantify the complex interactions in such milkrun-supplied flow lines for realistic problem sizes. We propose to use recurrent neural networks to determine the long-term throughput of such flow lines enabling us to evaluate production systems of flexible size. Our results show that the throughput can be determined accurately and quickly via recurrent neural networks. Furthermore, we use this new evaluation procedure as a building block to optimize this type of flow line using gradient and local search techniques.

Keywords Flow line · Milk run · Neural network · Performance Evaluation · Buffer allocation

MSC-Classification JEL classification · C45 · C61 · M11

Insa Südbeck · Julia Mindlina · André Schnabel · ✉ Stefan Helber
Institut für Produktionswirtschaft, Leibniz University Hannover, Königsworther Platz 1, 30167 Hannover, Germany
E-Mail: stefan.helber@prod.uni-hannover.de

Published online: 30 April 2024

1 Introduction

Flow lines are often used in the mass production of physical goods. In these lines, the processing times at different subsequent production stages, as seen by the workpieces, can vary randomly due to the random nature of the respective production processes, machine failures or other operational faults. To prevent such a disturbance in the flow of workpieces from propagating through the flow line and causing immediate blockage of upstream machines or starvation of downstream machines, subsequent production stages are often partially decoupled via buffers of limited capacity. This decoupling leads to the question of where to allocate buffer spaces to maximize the throughput of the line. Due to the complex interactions of blocking and starving in flow lines, the answer to this question depends on the entirety of the parameters describing the line. In practice, time-consuming discrete-event simulation models are often used to quantify the throughput of a particular flow line configuration in the design phase before often irreversible investment decisions in machines and buffers are made.

In this paper, we consider flow lines as they are often used for assembly operations of physical goods. It is common during different assembly operations for some kind of material parts to be assembled onto the main workpieces as they move downstream along the flow line. A certain number of these material parts is typically stored next to the respective machine or work station. This local material storage is periodically refilled by a train of cars circulating between central storage for those types of material parts, the so-called "supermarket", and the different work stations of the flow line. In such a system, an additional design question arises of how long the replenishment cycle should be and up to which level should the respective local storage of material parts be refilled upon the arrival of this train of cars. Due to the similarity with (former) daily deliveries of fresh milk to individual customer households, this system of material delivery is frequently called a "milkrun" in practice.

To create efficient production systems, it is essential to optimize both the flow line (in terms of buffer allocation) and the material component supply (in terms of the common milkrun replenishment cycle and the machine specific order-up-to levels) since material shortages impede the flow of workpieces through the flow line. However, quantifying the effect of such management decisions on the throughput of the line is methodologically challenging, in particular if results are required that are both accurate and can be obtained quickly within a systematic optimization process. This is particularly true for flow lines with a milkrun supply of material. Since analytical solution approaches are typically not suitable for modeling the complexity of these systems for realistic problem sizes, we propose Recurrent Neural Networks (RNNs) for the evaluation of stochastic flow lines with limited material supply. The main idea is to train such an RNN in advance of the optimization process based on a large data set of simulated flow line configurations. Thereby, the RNN allows for the performance evaluation of a flexible size of stages in the production system. During the optimization process, the RNN is then used as a both fast and accurate evaluation component for the numerous candidate configurations that are considered. The conceptual advantage of this approach is that it uses the flexibility of a discrete-event simulation of a manufacturing system and then delivers the computational speed of an analytic, closed-form solution for the throughput of the flow line as a function of its features, which are in turn subject to management decisions. This approach enables fast and systematic system optimization, which is typically not possible if a discrete-event simulation is used as the evaluation method within the optimization process. *Consequently, a trained RNN can act as a building block for an efficient decision support system.* However, setting up, training, and then using an RNN in the context of flow line analysis and optimization is not straight-forward. The main contributions of this paper are therefore as follows:

- We show how to set up an Artificial Neural Networks (ANN) to predict the throughput of a stochastic milkrun-supplied flow line. The special features of the selected RNN enable us to analyze, at least in principle, lines of *arbitrary* length.
- We demonstrate how the RNNs can be trained efficiently by creating training data in a systematic manner.
- We present numerical results that indicate the high precision of the throughput forecasts stemming from the RNN.
- We introduce two optimization approaches that combine our RNN with local and gradient search strategies and provide managerial insights into the optimal design of milkrun-supplied flow lines.

The remainder of this paper is organized as follows. In Sect. 2 we present a formal description of the production system and the resulting optimization problem and discuss the relevant literature. Sect. 3 provides a brief introduction to the architecture, operation, and training of the RNN, which quantifies the throughput function of the milkrun-supplied flow line. Numerical results related to the accuracy of the RNN are presented in Sect. 4. We introduce two optimization approaches and numerical results on their performance and the structure of the optimized flow line designs in Sect. 5. In Sect. 6, we draw our conclusions and present directions for further research.

2 Modeling and Optimization of Milkrun-Supplied Flow Lines

2.1 System Characteristics

We consider a basic model of a flow line with random processing times, finite buffer capacities between the machines, and limited material supply, as depicted in Fig. 1, for the case of a line with four machines M_1 to M_4 and three buffers B_1 to B_3. The key performance indicator of interest is the long-term throughput of the line in terms of work pieces per unit of time. To be able to analyze the system in isolation, we assume that in front of the first machine, there is an unlimited supply of raw work pieces. Likewise, we assume that downstream of the last machine, there is unlimited storage for completely processed work pieces. The first machine is, hence, never starved, and the last is never blocked.

For simplicity with respect to the exposition but without loss of generality pertaining to the methodology proposed in this paper, we assume that the processing

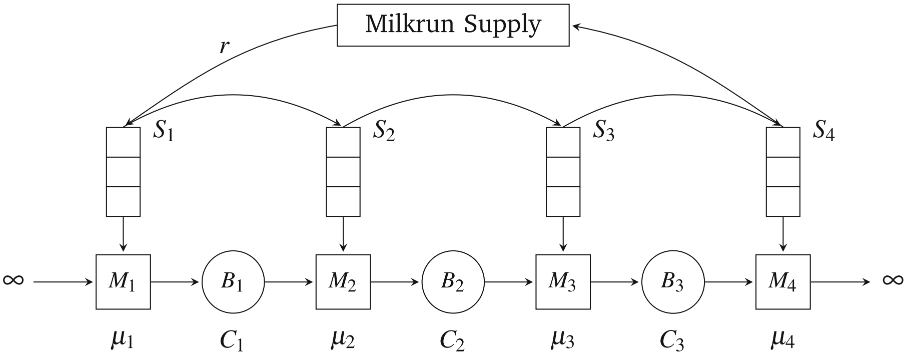


Fig. 1 Exemplary flow line with milkrun material part supply

times at the different machines i follow exponential distributions with rates μ_i. If a work piece has been processed by a machine and finds the downstream buffer full, it remains on that machine, which is then temporarily blocked, i.e., we assume blocking after service. Each processing task on a work piece consumes a unit of material parts that are stored next to the machines, e.g., because this unit is assembled onto the work piece. The local storage for material parts next to machine i is refilled to the machine-specific order-up-to level S_i, in constant and identical replenishment intervals of length r via a milkrun system. If a material part is not available in the line-side storage area, the work piece has to wait in the upstream buffer for the next material replenishment before its operation can start. The transportation times of the transport vehicle between machines are short relative to the length r of the replenishment cycle and are hence assumed to be zero.

Buffers of size C_i between machines i and $i+1$ dampen the propagation of blocking and starving in the line due to the random processing times. In addition to blocking and starving effects, material part shortages can impede the flow of work pieces through the flow line.

Due to the mutual interdependence between the flow line and material supply, integrated approaches are required to evaluate and optimize the performance of stochastic flow lines with limited material supply. In this way, a tailored material supply for the respective flow line configuration can be ensured.

The complex interactions in such a system can be demonstrated for the example in Fig. 1 with the parameter values in Table 1. The slowest machine in this flow line, machine M_3, can, on average, process 0.85 work pieces per time unit (TU), which constitutes the upper limit on the throughput of the line. However, the material

Table 1 Parameter values for the unbalanced four-machine line example

Parameter	Values
Processing rates $\mu_1, ..., \mu_4$ [TU^{-1}]	1.1, 0.9, 0.85, 0.95
Buffer sizes $C_1, ..., C_3$	all 2
Length r of the replenishment interval [TU]	60
Order-up-to levels $S_1, ..., S_4$	45, 54, 48, 51

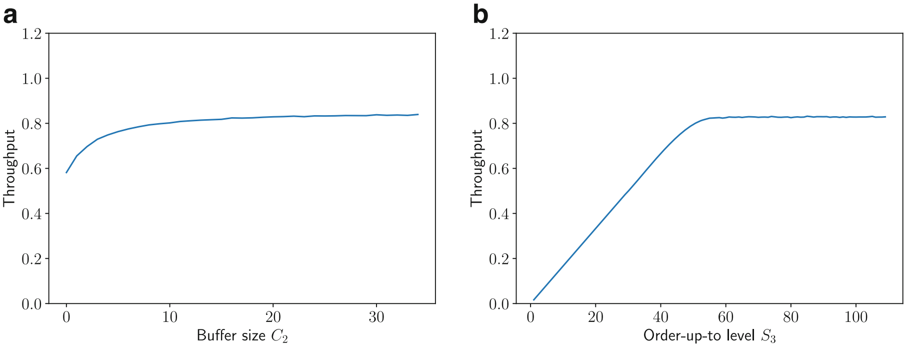


Fig. 2 Effect of an isolated parameter variation relative to a modified base case with buffer sizes $C_i = 20$ and order-up-to levels $S_i = 90$ for all machines i (discrete-event simulation results). **a** Variation in buffer size C_2, **b** Variation in order-up-to level S_3

supply at the first machine cannot exceed 45 material parts every 60 TU. This leads to a tighter upper bound on the throughput of 0.75 work pieces per time unit. However, the actual throughput of this line with small buffer sizes $C_i = 2$ and order-up-to levels S_i in Table 1, as determined via a simulation, is 0.6426/TU due to frequent blocking and starving because of the relatively small buffers and the relatively high variability of the exponentially distributed processing times.

For the processing rates μ_i and the length r of the replenishment interval in Table 1, both the buffer sizes C_i and the order-up-to levels S_i are insufficient to reach a throughput close to the processing rate $\mu_3 = 0.85\,\text{TU}^{-1}$ of the slowest machine M_3. In Fig. 2a and b, we therefore consider a new base case of a less “lean” line with buffers of size 20 and order-up-to levels of 90. The graphs show the effect of an isolated variation in the size C_2 of the second buffer (all other buffer sizes still being 20) or the order-up-to level S_3 at the third machine (all other order-up-to levels still being 90) on the throughput of the line. The graphs indicate that both C_2 and S_3 can be too small and hence limit the throughput of the line. They can also be too large and not have a further positive impact on the throughput of the line. The smoothness of the graphs of simulated throughput in Fig. 2a and b underlines the high degree of accuracy of our steady-state simulation results with a relative half-width of the 95% confidence interval not exceeding 0.5%.

2.2 Estimating the throughput via an artificial neural network

The throughput functions in Fig. 2a and b hold only for the given values of processing rates μ_i, buffer sizes C_i, length r of the replenishment cycle and order-up-to levels S_i. A series of time-consuming and highly precise discrete-event simulation runs were necessary to determine the shape of those figures, which renders a direct, simulation-based optimization impractical. This paper therefore proposes to use a neural network, i.e., a machine learning approach, to "learn" the (true but unknown) throughput function

$$TH = TH(\mu_1, ..., \mu_I, C_1, ..., C_{I-1}, S_1, ..., S_I, r) \tag{1}$$

for a high-dimensional parameter space via an RNN such that systematic flow line optimization becomes feasible using a previously trained neural network (instead of a simulation) as the evaluation component. Simulation results can be used to generate the data required to train the neural network. A suitably trained RNN hence results in a throughput estimate

$$\widehat{TH} = \widehat{TH}(\mu_1, ..., \mu_I, C_1, ..., C_{I-1}, S_1, ..., S_I, r). \tag{2}$$

For a given parameter constellation j of a flow line with parameters $\mu_{1,j}, ..., \mu_{I,j}, C_{1,j}, ..., C_{I-1,j}, S_{1,j}, ..., S_{I,j}, r_j$, our discrete-event simulation model provides an arbitrary exact value TH_j for the throughput of the line. For the same parameter constellation, our RNN provides a throughput estimate $\widehat{TH}_j$, thus leading to an estimation error

$$e_j = TH_j - \widehat{TH}_j. \tag{3}$$

The training of the RNN then aims to minimize those errors via a *loss function*, in our case, the mean squared error

$$\text{MSE} = \frac{\sum_{j=1}^{J} e_j^2}{J} = \frac{\sum_{j=1}^{J} (TH_j - \widehat{TH}_j)^2}{J} \tag{4}$$

by tuning the coefficients of the regression computation inside of the artificial neural network. The advantage of using a well-trained artificial neural network to obtain an estimate, in our case $\widehat{TH}$, is that the computations to determine $\widehat{TH}_j$ via the neural network are several orders of magnitude faster than a discrete-event simulation. This speed advantage makes systematic optimization, which would simply take too long if a discrete-event simulation were used to evaluate each tentatively considered flow line configuration, possible. However, to properly train such an artificial network, a sufficiently large number of observations is necessary prior to the optimization process, in which the network is then used as an evaluation component.

2.3 Optimization Problem

In the process of designing a flow line of this type, the optimization problem can be described as finding a combination of buffer sizes C_i, order-up-to levels S_i and length r of the milkrun replenishment cycle for given processing rates μ_i such that an exogenously given target throughput $TH^{\min}$ is achieved in the long run. This problem requires capital investment in buffers and local material part storage. Furthermore, we assume a cost (over the lifetime T of the system) that is proportional to the frequency with which the milkrun vehicles resupply the machines on the line.

Given the unit investment of buffers k_i^B for work pieces, unit investment for material storage k_i^M per unit of the auxiliary material, and unit cost k^R per delivery, we can state the problem to determine buffer capacities $\overline{C} = (C_1, C_2, ..., C_{I-1})$, order-up-to levels $\overline{S} = (S_1, S_2, ..., S_I)$ and the frequency $\frac{1}{r}$ of the common replenishment cycle of length r that minimize the required investment as follows:

$$\text{Minimize } f\left(\overline{C}, \overline{S}, \frac{1}{r}\right) = \sum_{i=1}^{I-1} k_i^B \cdot C_i + \sum_{i=1}^{I} k_i^M \cdot S_i + k^R \cdot \frac{T}{r} \tag{5}$$

s.t.

$$TH(\mu_1, ..., \mu_i, C_1, ..., C_{I-1}, S_1, ..., S_I, r) \geq TH^{\min} \tag{6}$$

$$C_i \geq 0, \quad i = 1, ..., I-1 \tag{7}$$

$$S_i \geq 1, \quad i = 1, ..., I \tag{8}$$

$$r > 0 \tag{9}$$

The decision variables of this problem are the buffer sizes C_i, the order-up-to levels S_i, and the length r of the replenishment interval. In the objective function (5), we aim to minimize the one-time investment in buffer space and material part storage plus the total (nondiscounted) cost of the replenishment operations over the lifetime T of the flow line.

Both the objective function and the main constraint (6) are nonlinear in the decision variables. Closed-form expressions of the throughput function (1) are not known in general. To facilitate a systematic flow line optimization, we propose to use deep learning methods to "learn" this throughput function $TH(\mu_1, ..., \mu_i, C_1, ..., C_{I-1}, S_1, ..., S_I, r)$ and then use gradient and local search methods to optimize the problem (5)–(9) in an integrated approach.

2.4 Literature Review

The modeling and optimization of stochastic flow lines with limited material supply via deep learning methods is related to three different literature streams. However, the connections between those streams are only partially developed.

Evaluation and optimization approaches for flow lines with random processing times and finite buffer capacities are discussed in a large number of publications. Surveys on the performance analysis of flow lines are provided by Dallery and Gershwin (1992); Papadopoulos and Heavey (1996); Li and Meerkov (2009) and in monographs by Buzacott and Shantikumar (1993); Gershwin (1994); Altiok (1997) and Papadopoulos et al. (2009). Markovian models, which are used for the exact analysis of small systems as e.g. two-machine lines, are presented by Dallery and Gershwin (1992); Li et al. (2006) and Papadopoulos et al. (2019).

Aggregation approaches and decomposition techniques are applied for an approximate analysis of long and complex flow lines. The initial decomposition approaches by Gershwin (1987); Dallery et al. (1988) and Buzacott and Shantikumar (1993) and the aggregation approach by Li et al. (2009) were extended for a variety of system configurations. However, there are no closed-form solutions for stochastic milkrun-supplied flow lines that can be used for the analysis of longer systems with decomposition.

The optimization of stochastic flow lines often addresses the buffer spaces between the production stages. A classification of the literature on buffer optimization approaches can be found in Demir et al. (2014) and Weiss et al. (2019). Examples of solution approaches are presented, e.g., by Gershwin and Schor (2000); Spinellis et al. (2000); Shi (2012) and Weiss and Stolletz (2015). To optimize milkrun-supplied flow lines, we aim to achieve simultaneous optimization of the number of buffers and the material supply configuration described by the material order-up-to levels and the milkrun replenishment cycle.

A large number of publications refer to the supply of material parts focussing on storage of parts, transport of parts and the part feeding policy. Though, only a restricted number of publications refers to integrated models for stochastic flow lines with limited material supply. Mindlina and Tempelmeier (2021) present approximative (mixed-integer) linear programming approaches for the evaluation and optimization of stochastic milkrun-supplied flow lines. Since these approaches face limitations in terms of problem size, we propose an evaluation approach based on a Recurrent Neural Network (RNN). Further publications with integrated approaches for stochastic milkrun-supplied flow lines imply other problem formulations. Yan et al. (2010) propose an approach to allocate a limited number of material transport vehicles to line-side buffers with the aim of preventing material shortages. Chang et al. (2013) develop a performance evaluation approach based on a max-plus linear system for flow lines with limited material supply and random machine failures. Ciemnoczolowski and Bozer (2013) present a closed-form model to approximate the material starving probability of work stations in a milkrun-supplied flow line. Further, Weiss et al. (2017) present a sample-based optimization approach to solve the buffer allocation problem of a stochastic flow line with limited material supply only at the first station.

The third literature stream refers to neural networks. We focus on publications that consider the application of neural networks to the evaluation and optimization of stochastic flow lines. In contrast to these publications, we apply a RNN for the high-dimensional parameter space of the underlying throughput function to enable optimization of the entire production system. To the best of our knowledge, there are no specific applications of RNNs to stochastic flow lines in the literature. A large number of publications incorporating RNNs can be found in papers on time series and demand prediction. However, we do not include these papers in our review to keep the focus on production systems. The foundational literature on RNNs is presented in Sect. 3. Altiparmak et al. (2002) apply an ANN together with simulated annealing to optimize the buffer sizes in a closed asynchronous assembly system. Tsadiras et al. (2013) develop a decision support system based on an ANN for the evaluation and a myopic algorithm for the optimization of the buffer allocation of reliable stochastic production lines. Jang et al. (2003) propose an ANN for the identification of variation patterns and the reduction of variability in an automotive assembly process. Li et al. (2016) apply a combination of grey model and ANN to predict the throughput of a multi-product production line with parallel machines and rework loops. Tan and Khayyati (2021) propose a machine learning framework for the implementation of data-driven control policies. A survey of recent deep learning algorithms, especially those with applications to smart manufacturing, is provided in Wang et al. (2018) and Arinez et al. (2020).

Consequently, the contribution of our article is the connection of the three literature streams in terms of applying an RNN to evaluate a complex stochastic system that cannot be modeled analytically since it implies mutual interdependence between the flow line and the milkrun material supply. Further, we use the approach for a holistic optimization of the system parameters with gradient and local search techniques. This paper is an extension with new numerical results of the working paper Südbeck et al. (2022) and represents some of the results of the German-language dissertation Südbeck (2023).

3 Performance Evaluation Using RNN

3.1 Configuration of the RNN for Predicting the Throughput

The goal of a neural network is to learn an underlying function from given data.[1] Therefore, the network needs a data set containing features as inputs and a label for each data point, which will be returned as the output. The algorithm learns to predict the labels of the data points. In our case, a data point describes a complete flow line configuration, and the label is the corresponding throughput of this line, as determined via the discrete-event simulation.

In general, ANNs consist of so-called neurons that receive, process, and transmit information. In a simple feedforward neural network, the neurons are arranged in layers and are connected by directed and weighted links to all neurons of the subsequent layer. Due to the fixed structure of ANNs it is only possible to apply them to flow lines with a fixed length. In recurrent artificial neural networks (RNNs) (Rumelhart et al. 1986), there are additional feedback loops to neurons of the same layer, so that a recurrence structure from a layer to itself is established. Such networks can be used to process sets of sequential data of variable size. In our approach, we have a sequence of production stages with different parameters per stage. Hence, it is possible to process flow lines with different lengths. The training data set consists of one matrix per flow line with the following structure:

$$X = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_I \end{bmatrix} = \begin{bmatrix} r & \mu_1 & S_1 & C_1 \\ r & \mu_2 & S_2 & C_2 \\ \vdots & \vdots & \vdots & \vdots \\ r & \mu_I & S_I & C_I \end{bmatrix} \tag{10}$$

This matrix comprises one vector per machine. The input vector for each machine consists of the milkrun cycle length r, the processing rate μ_i, the order-up-to level S_i and the buffer size C_i. Note that the entry C_I in equation (10) does *not* represent a physical buffer size as there is no buffer behind station I. That entry is a completely

[1] For more information about this section refer to Südbeck (2023, Chapt. 4, pp. 65).

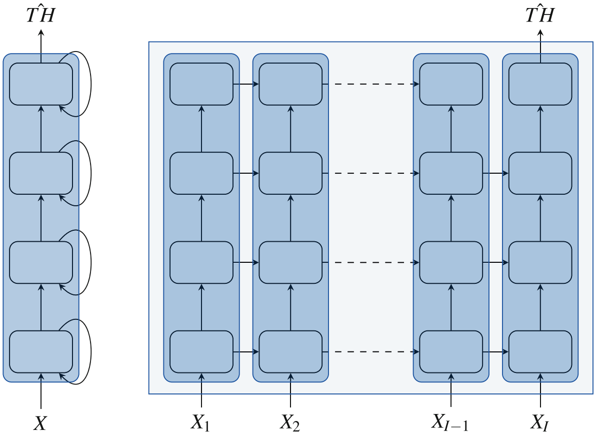


Fig. 3 An RNN (left) and its unfolded architecture (right) (cf. Géron 2019, p. 507)

artificial artefact used only to conserve the formal structure of that equation. The performance of the RNN improves if a large value, instead of no value, is selected for C_I. We chose $C_I = 100$ as in our numerical studies we never considered a larger buffer. The input of r for each machine is redundant for a milkrun supply, but we need this value to again obtain a unified structure of the inputs for all machines.

In our study, we use Gated Recurrent Units (GRUs) (Cho et al. 2014). A deep RNN consists of several layers with GRU cells. Fig. 3 shows such a four-layer RNN for predicting the throughput on the left side. The information is passed through subsequent layers (from bottom to top), and the neurons also receive information from themselves from the previous step. On the right side of Fig. 3, the RNN is unfolded. RNNs can unroll depending on the length of the given input vector of the explicit data point. Hence, they do not need fixed input lengths but can operate on vectors of different sizes. There is no restriction on the input length during training or when using the trained RNN for prediction, as long as it has the same structure for each element. In the input layer, the RNN can process the 4×1 parameter vector of a single machine and unrolls along with the length of the flow line, i.e., the number of rows in the input matrix. In Fig. 3, the information of one column is forwarded to the corresponding cells in the subsequent column. Additionally, external new information about the subsequent production stage is given to the network in the first layer of each column. The last layer provides the output of the throughput prediction.

To train a neural network on a given task, the weights between the neurons are adjusted according to the gradients of the loss function (4) to achieve a high prediction accuracy and, hence, to "learn". An efficient way to compute the gradients is backpropagation (Rumelhart et al. 1986).

Based on a random search, we configure the RNN with 2 layers each with 100 neurons. Within the learning process, we use a batch size of 500 and a learning rate of 0.001. Furthermore, we apply the Mean Squared Error (MSE) as a loss function and the Adam optimizer (Kingma and Ba 2015). During training, we use 90% of the data to adjust the network and 10% to validate the performance. For all other configurations, we use the default settings of *Keras* (Chollet et al. 2015; Abadi et al. 2015).

3.2 Generating Training Data with Orthogonal Latin Hypercube Sampling (OLHS)

The training of machine learning algorithms requires a large amount of data to learn the function relating the labels to the data point values, in our case, the throughput function. The training data set should cover the whole space of possible flow line specifications. Without an adequate training data set, the algorithm cannot learn the relationship. Hence, data generation is an important aspect. The data should be equally distributed within the parameter space. In previous studies, we observed that simple random sampling is insufficient for generating training data for the systems studied in this work. To cover the entire space to a sufficient degree would require an enormous amount of data, which is not practicable. Thus, we use Orthogonal Latin Hypercube Sampling (OLHS) to generate training data that systematically cover the entire space.

OLHS aims to achieve a roughly equal distribution of data points across the entire parameter space. Due to orthogonal arrays, the number of required data points increases as the number of dimensions within the parameter space increases. Each further stage of the flow line leads to three additional parameters μ_i, S_i, and C_i and, hence, three additional dimensions of the hypercube. As a result, the number of required data points to achieve an overall balanced data set (i.e., simulated flow line configurations with corresponding throughput labels) increases exponentially with the length of the flow line.

For this study, we created two separate data sets with OLHS and later merge them. The first consists of flow lines with four machines, and the second consists of flow lines with six machines. A flow line consisting of four machines has a total of $3 \cdot 4 = 12$ parameters. In our data generation process, all parameters are uniformly distributed within the given ranges. Therefore, we divide each dimension into two subsets and create hypercubes with all combinations of subsets. This process results in $2^{3 \cdot 4} = 4{,}096$ hypercubes. For the training process, we need a large amount of data. Hence, we decided to create at least 1 million data points, each representing one randomly created flow line with its corresponding throughput, as determined via simulation. In total, the number of training data points is the smallest multiple of the number of hypercubes that is larger than or equal to one million, which results in a data set with 1,003,520 data points. Hence, there are $\frac{1{,}003{,}520}{4{,}096} = 245$ samples in each hypercube.

We repeat this process for the second data set with flow lines consisting of six machines. These lines have a total of $3 \cdot 6 = 18$ parameters. Again, we divide each dimension into two subsets which results in 262,144 hypercubes. Due to the number of parameter dimensions and hypercubes, we duplicated the required number of data points to two million flow lines. Hence, this data set consists of 2,097,152 samples, with 8 data points located in each hypercube. Afterward, we merge the two data sets to obtain one training data set that consists of 3,100,672 flow lines. Due to the large number of hypercubes, including longer flow lines in the training data set was not

Table 2 Parameter ranges for flow line configurations

Parameter	LB	UB
Processing rate μ_i	0.8	1.2
Buffer size C_i	0	80
Milkrun cycle length r	30	120
Order-up-to level S_i	15	180
Material ratio $\frac{S_i}{r}$	0.5	1.5

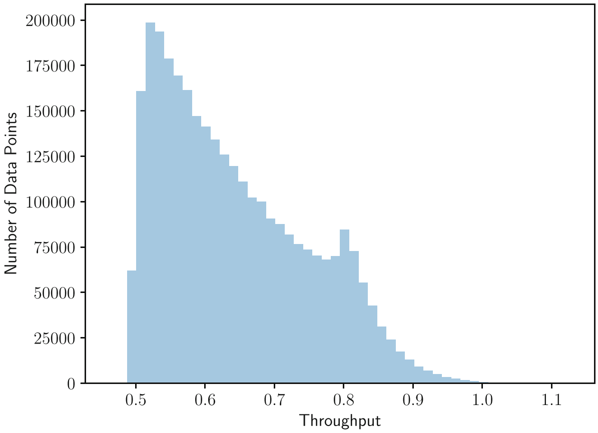
practicable. Nevertheless, with a data set consisting of two lengths of flow lines of four or six machines, we can achieve a promising accuracy with a trained recurrent neural network, even for longer lines, as our numerical results will demonstrate.

We create a separate validation data set with the same procedure which has a size of approx. 10% of the training data set.

Table 2 shows the parameter ranges of the training data set with the lower bound (LB) and the upper bound (UB). The order-up-to level S_i is calculated according to the material ratio and is dependent on the milkrun cycle length r. In this way, we ensure that the machines receive a moderate supply of material. The label for each data point, i.e., the throughput of the flow line, is computed with a discrete-event simulation model coded in the C programming language. Since there is no analytical model for the evaluation of the considered systems, we need a simulation model for generating training data.

Fig. 4 shows the distribution of the simulated throughput, i.e., the labels corresponding to the different data points (flow line configurations), within the training data set. Due to the random selection and combination of parameters of the flow lines within the training data set, the throughput is limited by at least one parameter in most of the cases. Therefore, 50% of the flow lines have a throughput of 0.55–0.72 products per time unit, with a mean of 0.64 products per time unit. Since the underlying training data are essential for the performance of the trained RNN, we assume that the prediction will perform well in areas with a high number of data points and poor for lower or higher throughputs. Specifically, accurate prediction of

Fig. 4 Distribution of the throughput within the training data set



throughputs below the minimum of 0.46 and above the maximum of 1.13 is probably not possible.

4 Numerical Study on the Accuracy of an RNN

After training the RNN for 50 epochs with about 3 million data points consisting of flow lines with 4 and 6 machines, we obtain an MSE on the training data set of $4.85 \cdot 10^{-6}$ and on the validation data set of $4.20 \cdot 10^{-6}$.[2] The training required approximately 29.76 minutes on the GPU partition of the cluster system of the Leibniz University of Hannover, Germany (Intel Xeon Gold 6230 CPU with 20 cores, 2.1 GHz and 27,5 MB Cache and NVIDIA Tesla V100 GPU with 2 GB memory).

Fig. 5 illustrates the training and validation loss. Since the two error measures do not differ significantly from each other, especially in the later epochs, the model does not overfit the training data. Further training of the model might reduce both losses, but the last epochs suggest that the improvement is only marginal while the computation costs are high. Additionally, a validation loss of $4.20 \cdot 10^{-6}$ suggests good performance on new data points.

To analyze the performance, we apply the trained RNN to new data points that are not included in the training data set and compare the prediction with the simulated throughput. In Fig. 6, we apply the trained RNN to four and six-machine flow lines with processing rate $\mu_i = 1\,\mathrm{TU}^{-1}$, buffer size $C_i = 20$ and length of the replenishment interval $r = 60\,\mathrm{TU}$ and compare it with the simulation. Fig. 6a shows the variation in order up-to level S_i for four-machine lines, while Fig. 6b illustrates the same analysis for six-machine lines. In both cases there is no observable difference between the curves of the throughput predicted with the RNN and the simulated throughput. The RNN can predict the throughput of new four and six-machine lines with parameter configurations that lie within the training data set very accurately. Hence, the RNN can interpolate, and the prediction accuracy is high.

The advantage of using RNNs is their ability to accept inputs of different lengths. Thus, we are not limited to flow lines with four and six machines. We also apply the trained network to longer and shorter flow lines to gain insights into the extrapolation capabilities of the RNN. Table 3 shows the mean time in seconds for the prediction of one out of 1000 flow lines randomly generated flow lines with varying lengths and parameters within the ranges of Table 2, as well as the achieved MSE and Mean Absolute Percentage Error (MAPE) compared to the simulation of these lines. For the unbalanced lines, all parameters are drawn independently and randomly. In the case of balanced lines, we draw only one value for all processing rates, all buffer sizes and all order-up-to levels, respectively. For all different lengths of flow lines, an MSE below 0.09 can be achieved. For unbalanced flow lines with 4 to 6 machines, the MSE is comparable with the final validation loss of the training process after 50 epochs, and the MAPE is around 1 %. (Note, that the training objective was to minimize the MSE: the MAPE was not considered during the training process.) For longer lines

[2] The following results correspond to the numerical study in Südbeck (2023, Chapt. 4.3, pp. 74).

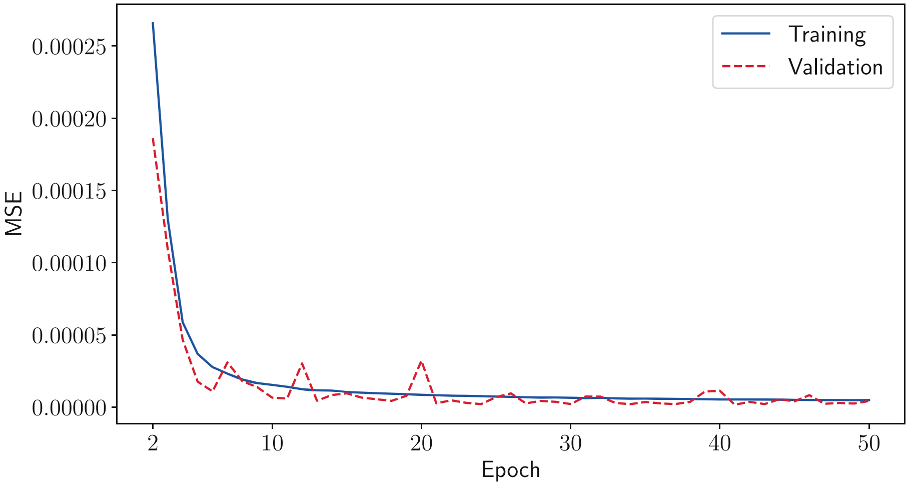


Fig. 5 Training and validation loss

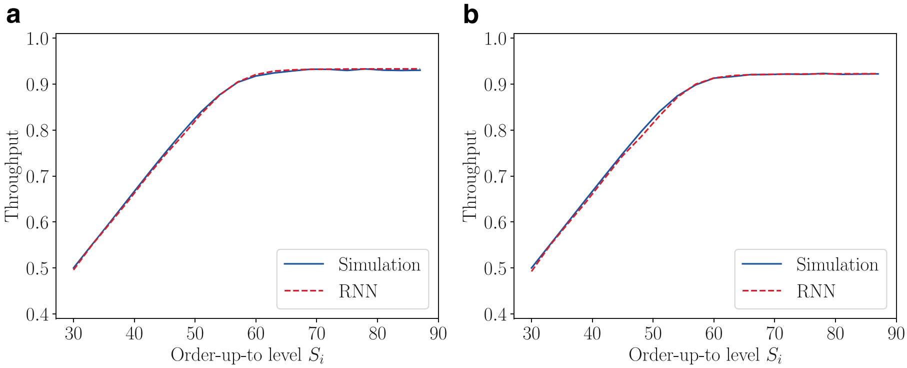


Fig. 6 Throughput of 4- and 6-machine lines with processing rate $\mu_i = 1\,\mathrm{TU}^{-1}$, buffer size $C_i = 20$ and length of the replenishment interval $r = 60\,\mathrm{TU}$ with variation in order up-to level S_i predicted via RNN and simulation. **a** 4-machine line, **b** 6-machine line

both error measures increase slightly. The main advantage of using neural networks for the prediction of the throughput is the reduction in computation time compared to the simulation of such systems. In all cases, the throughput prediction of one flow line takes less than 0.002 seconds and does not grow substantially as the length of the flow lines increases, while the simulation time of longer flow lines increases substantially. The MSE and MAPE values in Table 3 lead to the conclusion that the prediction accuracy can be very high, even for flow lines with more than 4 or 6 machines for which the RNN was initially trained. The prediction accuracy decreases for short lines with only two or three machines (cases that had not been included in the training data). For balanced lines, we observe smaller error measures for lines with 2 to 10 machines but higher errors for longer lines.

Table 3 Prediction time and MSE and MAPE of the RNN for randomly generated flow lines with different lengths (1000 instances each)

	Simulation	RNN	Unbalanced flow lines		Balanced flow lines	
$\lvert\mathcal{I}\rvert$	Time [s]	Time [s]	MSE	MAPE	MSE	MAPE
2	0.3012	0.0001	0.0110	10.3850%	0.0024	4.8126%
3	0.4232	0.0001	0.0011	2.8260%	0.0003	0.9985%
4	0.6347	0.0002	1.61×10^{-4}	1.0105%	6.02×10^{-5}	0.5943%
5	0.7714	0.0002	1.25×10^{-4}	0.9207%	1.07×10^{-4}	0.7793%
6	0.8829	0.0003	1.34×10^{-4}	0.9251%	9.61×10^{-5}	0.7569%
8	1.1158	0.0004	0.0002	1.0736%	0.0001	0.8562%
10	1.3756	0.0004	0.0002	1.2150%	0.0002	1.1388%
15	2.1097	0.0007	0.0002	1.3247%	0.0003	1.3947%
20	2.7902	0.0009	0.0003	1.5153%	0.0004	1.7215%
25	3.3674	0.0012	0.0002	1.5896%	0.0005	1.9007%
30	4.8685	0.0015	0.0002	1.6955%	0.0006	2.1079%

If we apply the network to concrete examples of flow lines of different lengths, we obtain the results of Fig. 7. The default configuration of these lines is $r = 60$, $\mu_i = 1$ and $C_i = 20$ and for all machines i. As the number of machines within the flow line increases, the prediction accuracy decreases slightly. The evaluation of flow lines with 10 machines in Fig. 7a is very accurate except for large order up-to levels. With increasing flow line length the accuracy decreases for small order up-to levels. The RNN systematically overestimates the throughput for high order up-to levels and underestimated the throughput for small order up-to levels. For moderate order up-to levels, we obtain an accurate approximation of the simulated throughput for all considered flow line lengths. In all four figures, the two curves for the simulation and the RNN converge at a throughput of approximately 0.70 products per time unit, which is near the mean throughput in the training data set. This observation supports the assumption that prediction accuracy is high in ranges with many training data points. In these ranges, extrapolation to longer flow lines is possible. Outside of these ranges, the ability to extrapolate decreases as the number of machines increases. Additionally, the RNN is trained with unbalanced flow lines, where the throughput is determined mainly by the bottleneck of the flow line. In contrast, the flow line in Fig. 7d is balanced. For balanced systems, the prediction of the throughput is more complex because all machines and buffers have the same influence, while in unbalanced systems the throughput is determined mainly by one parameter of the bottleneck machine.

Fig. 8a shows the percentage error of the prediction with the RNN of 1000 randomly generated, unbalanced flow lines with 4 machines as a function of the simulated throughput of the line. The MAPE for these lines is around 1%. We observe higher deviations from the simulated throughput for flow lines with a comparably small or large throughput. The RNN provides more accurate results for flow lines with a throughput between 0.5 and 0.8. In this range, we have most of the training data. Hence, we could decrease the errors with including more training data. Fig. 8b shows the same analysis for 1000 randomly generated, unbalanced flow lines with

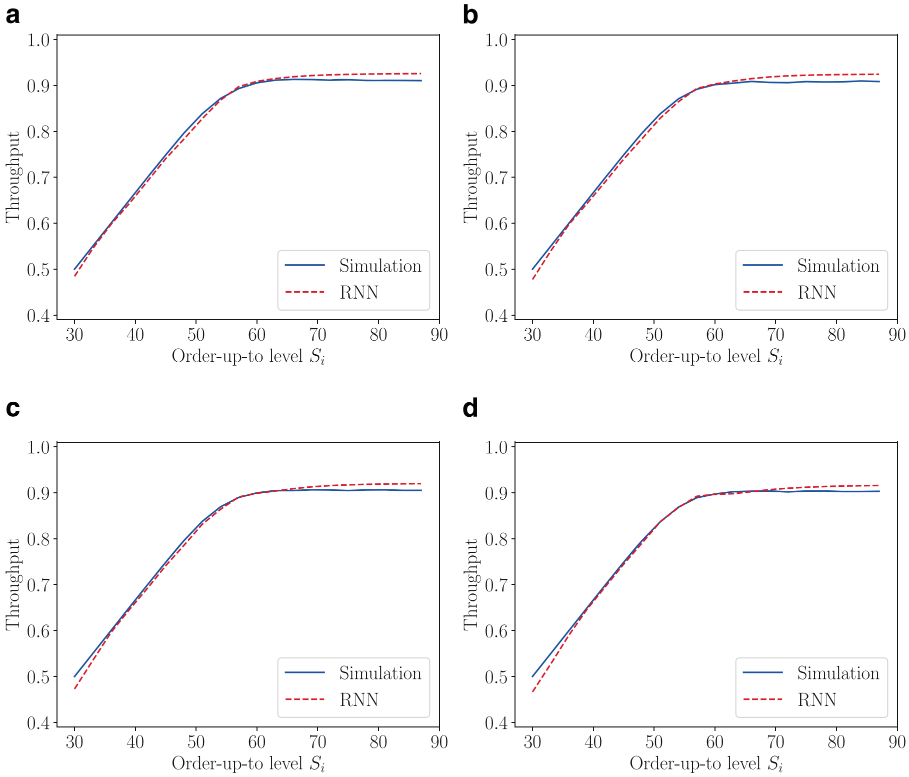


Fig. 7 Throughput of flow lines with different lengths with variation in all order-up-to levels predicted via RNN and simulation. **a** 10-machine line, **b** 15-machine line, **c** 20-machine line, **d** 30-machine line

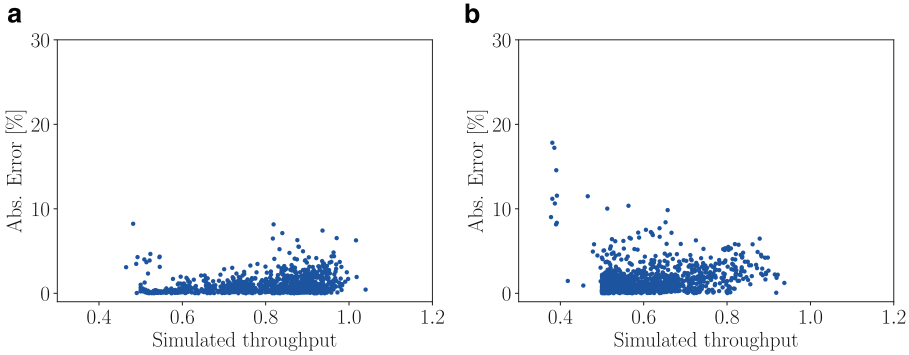


Fig. 8 Percentage error in the throughput prediction of 1000 randomly generated, unbalanced flow lines with 4 and 30 machines. **a** 4-machine lines, **b** 30 machine lines

30 machines. We observe a significant increase in the percentage error and in the scattering compared to those for 4 machine lines. The observed effect for flow lines with low throughput increases with increasing line length.

5 Simultaneous and Successive Optimization of Buffer Allocation and Material Supply

5.1 Optimization Approaches

To solve the optimization problem as described in Sect. 2.3, we propose two different approaches that both utilize the RNN to evaluate a given flow line configuration:[3]

- Gradient Search (GS)
- Simulated Annealing (SA)

Both methods are frequently and efficiently used for solving the buffer allocation problem (see e.g. Helber (2000, 2001) and Spinellis et al. (2000)). It is possible to use a GS approach because the RNN is an approximation of a high-dimensional continuous function, and we can hence easily determine numerical approximations of the gradients of the throughput function and use this information for the optimization process. Unlike a discrete-event simulation, the RNN can evaluate even non-integer values for the buffer sizes and order up-to levels, which facilitates the numerical approximation of those gradients. Additionally, we use both approaches for a simultaneous as well as a successive optimization. In the successive optimization, we firstly optimize the buffer allocation and secondly the material supply.

In the next part, we describe the procedures for the simultaneous optimization approaches. Both approaches can be easily adapted to the successive approach with fixing the material supply in the first iteration and the buffer sizes in a second iteration of the optimization procedure.

The idea of the Gradient Search algorithm (see Figure 9) is to start with a solution which is feasible in terms of the required throughput as it has large buffer sizes $\overline{C}$, high order-up-to levels $\overline{S}$ and a high milkrun frequency $\frac{1}{r}$. We then iterate between two phases of gradient-based moves in which we simultaneously modify buffer sizes, order-up-to levels and milkrun frequencies.

In the Phase-I move, we have a current solution with a throughput estimate (determined via the RNN) which exceeds the required throughput $TH^{\min}$. During the Phase I, we use as the move direction the gradient

$$\nabla f\left(\overline{C}, \overline{S}, \frac{1}{r}\right) = \left(k_1^B, k_2^B, ..., k_{I-1}^B, k_1^M, k_2^M, ..., k_I^M, k^T \cdot T\right) \tag{11}$$

of the objective function (5) multiplied by (-1). Moving against the direction of this gradient $\nabla f(\overline{C}, \overline{S}, \frac{1}{r})$, we reduce the investment sum until we just meet the desired minimum throughput $TH^{\min}$ of the system. This can easily be organized as a bisection search. After the Phase-I move, we have found a solution which is less costly than our initial solution and just meets the throughput requirement.

In the subsequent Phase-2 moves, we try to re-allocate the investment sum for the Phase-I solution in such a way that we get a system configuration which requires the same investment, but yields a higher throughput. To this end, we aim at redis-

[3] Further explanations of the optimization approaches can be found in Südbeck (2023, Chapt. 6, pp. 143).

tributing the investment in buffers, order-up-to levels and the delivery frequency. In other words, we want to make sure that the current investment budget is held constant while the throughput of the system should increase again. To achieve this, we first determine a numerical approximation of the gradient $\nabla TH(\overline{C}, \overline{S}, \frac{1}{r})$ of the throughput function. However, if we now took this gradient as a move direction, we would not only increase the throughput, but also the required investment. In order to make sure that the net effect of the change of the system configuration on the required budget is zero, we have to project this gradient $\nabla TH(\overline{C}, \overline{S}, \frac{1}{r})$ of the throughput function on the constraint that the net change of the objective function (5) is zero. This can be done using Rosen's gradient projection method (Rosen (1960); Rosen (1961)). To obtain the projected gradient which locally increases the throughput while keeping the required investment constant, we apply formula (12), where $N = (k_1^B, ..., k_{I-1}^B, k_1^M, ..., k_I^M, k^R \cdot T)^T$ is a vector of the cost coefficients and $\nabla TH(\overline{C}, \overline{S}, \frac{1}{r})$ the numerical approximation of the gradient of the throughput function stemming from the RNN.

$$s = -\left[I - N\left(N^T N\right)^{-1}\right] \cdot \nabla TH\left(\overline{C}, \overline{S}, \frac{1}{r}\right) \tag{12}$$

With the information of the projected gradients s and starting solution $y = (\overline{C}, \overline{S}, \frac{1}{r})$, i.e, a system configuration, we can determine solution y_2. All solutions between solution y and y_2 in the direction of s have the same cost but differ with respect to the throughput. We can apply the golden section search (Kiefer 1953) to try to increase the throughput while spending the same budget.

Within GS, we therefore perform an iterative combination of a bisection search (to reduce the investment budget while retaining feasibility) and a golden section search (to increase the throughput while retaining the investment budget). In all steps, we simultaneously update all variables C_i, S_i and $\frac{1}{r}$. The procedure is summarized in Algorithm 9. We know that the throughput is a concave function in buffer sizes $\overline{C}$, order-up-to levels $\overline{S}$ and the (milkrun) delivery frequency $\frac{1}{r}$. If buffer sizes and order-up-to levels could in reality be real-valued and if the RNN represented a perfectly accurate representation of the throughput function, this procedure would always find the globally optimal solution to any required degree of accuracy. However, our RNN is only giving us an approximation $\widehat{TH}$ of the throughput. This approximation is not entirely accurate and in particular, not perfectly concave. We stop the Golden Section Search in the Phase-II moves if we cannot find an improving solution. The resulting solution usually contains non-integer buffer sizes and order-up-to levels. By rounding up all those fractional values, we can achieve feasibility. For instances in which we want to have rather small buffer sizes, this can be a quite crude approach, but for those with larger buffer sizes, it should work well. This will be confirmed in our numerical results.

As a second approach, we use SA to simultaneously optimize the buffer allocation and material supply. We apply the following simple neighborhood operators to generate a new solution nearby a given solution:

```
Choose initial feasible starting solution $(\overline{C},\overline{S},\frac{1}{r})$;
BetterSolutionFound = True;
while BetterSolutionFound do
    BetterSolutionFound = False;
    Perform Bisection Search to reduce the required investment;
    if Bisection Search found less costly feasible solution then
        BetterSolutionFound = True;
    end
    Perform Golden Section Search to increase the throughput for the current
      investment;
end
Round up buffer sizes and order-up-to levels $(\overline{C},\overline{S})$;
return $(\overline{C},\overline{S},\frac{1}{r})$ and the required investment;
```

Fig. 9 Gradient Search

1. Remove/add one buffer at a random machine.
2. Remove/add one material unit at a random machine.
3. Remove/add one material unit at a random machine and add/remove one buffer space at the buffer in front of this (another random) machine.
4. Remove a random amount of buffer units from a random buffer.
5. Remove/add a random amount of buffer units from a random buffer and add/remove the amount to/from another randomly selected buffer.
6. Add/subtract a random number to/from the milkrun cycle length and add/subtract the same number multiplied with the corresponding processing rate μ_i (rounded) to/from the material levels of all machines i.
7. Subtract a random number from the milkrun cycle length.

In each iteration, we select with equal probabilities one operator randomly and evaluate the new solution with the RNN. Within SA, we accept worse solutions with a given probability and depending on the solution quality to escape local optima. Analogous to the cooling process of metal, this probability reduces in each iteration. We use a logarithmic cooling schedule and the Metropolis acceptance criterion (Metropolis et al. 1953). We terminate the SA when a given time limit is reached.

5.2 Effect of parameter variation on optimization outcomes

We now use the two optimization algorithms in the simultaneous and successive approach to optimize the configuration of a balanced flow line with $I = 6$ machines and 5 buffers.[4] The machines $i \in \{1, ..., I\}$ share a common processing rate of $\mu_i = 1$ and order-up-to level cost coefficient of $k_i^M = 100$ monetary units (MU) and buffer cost coefficient of $k_i^B = 500$ MU. We set the cost per delivery $k^R = 10.98$ MU and the system lifetime $T = 100{,}000$. We compute all optimization results on an Intel Cascade Lake Xeon Gold 6230N CPU with 20 cores, a 2.3 GHz processor and 30 MB

[4] The following results correspond to the numerical study in Südbeck (2023, Chapt. 7.1, pp 177).

Table 4 Costs of best solution found for initial configuration

Time Limit	$\mathrm{GS}^{\mathrm{sim}}$[MU]	$\mathrm{SA}^{\mathrm{sim}}$[MU]	$\mathrm{GS}^{\mathrm{succ}}$[MU]	$\mathrm{SA}^{\mathrm{succ}}$[MU]
1 s	**65,721.15**	68,507.42	64,363.23	64,129.15
10 s	65,721.15	68,507.42	64,363.23	64,129.15
100 s	65,721.15	68,353.42	64,363.23	**64,127.56**

of cache of the cluster system of the Leibniz University Hannover, Germany. The SA algorithm was implemented in Python, compatible with the RNN implementation in Keras, the Python deep learning API. While the GS approach was implemented in C++. In the starting solution, we set all buffers to 40, all order up-to levels as well as the milkrun cycle length to 90. For the initial solution, the RNN approximates a throughput of 0.9406 and the simulation obtains a throughput of 0.9420 with costs of 16222.22 MU. The minimum throughput required is 80% of the slowest machine. In this case $TH^{\min} = 0.8$. Hence, the starting solution of the optimization approach is feasible. We execute the SA with three different time limits of 1, 10, and 100 seconds. Table 4 shows the total costs of the best-found configurations. Please note, that we cannot set a time limit within GS since the search procedure ends when the algorithm converges. For this configuration, the algorithm took about 1.5032 seconds in the simultaneous approach and 0.1396 seconds in the successive approach to find the best solution it can find.

Table 4 shows that all procedures can find very good solutions within a short time. $\mathrm{GS}^{\mathrm{sim}}$ finds the best solution with the simultaneous optimization approach, and $\mathrm{SA}^{\mathrm{succ}}$ finds the overall best solution. This configuration includes for the five buffers a buffer allocation pattern of $(6, 7, 8, 5, 6)$ and order-up-to levels of 40–42 units at all machines with a milkrun cycle length of 46.95 TU. The corresponding throughput predicted with the RNN is 0.8000. To check the feasibility of this solution, we simulate the configuration and get a throughput of 0.8046. Thus, the solution is very close to the boundary throughput but is still feasible. We can observe an inverted bowl shape in the buffer sizes with one outlier behind machine 4, while the material is supplied to all machines uniformly. Since the material ratio $\frac{S_i}{r}$ differs between

Table 5 Comparison of performance for both approaches with increasing required throughput $TH^{\min}$ for a time limit of 10 seconds

	$\mathrm{GS}^{\mathrm{sim}}$	$\mathrm{SA}^{\mathrm{sim}}$	$\mathrm{GS}^{\mathrm{sukz}}$	$\mathrm{SA}^{\mathrm{sukz}}$
$TH^{\min}$ $[\mathrm{TU}^{-1}]$	$\frac{Z^{\mathrm{GS}}-Z^{\mathrm{best}}}{Z^{\mathrm{best}}}$ [%]	$\frac{Z^{SA}-Z^{\mathrm{best}}}{Z^{\mathrm{best}}}$ [%]	$\frac{Z^{\mathrm{GS}}-Z^{\mathrm{best}}}{Z^{\mathrm{best}}}$ [%]	$\frac{Z^{SA}-Z^{\mathrm{best}}}{Z^{\mathrm{best}}}$ [%]
0.50	52.8739	1.6347⋆	5.1560	0⋆
0.55	45.2761	0⋆	5.1336	0.2780⋆
0.60	35.2469⋆	0	3.7886	0.0616⋆
0.65	34.1639	1.9918	4.0287	0
0.70	29.9677	3.2056	2.3337	0
0.75	4.6734	3.2546	4.9547	0
0.80	2.4825	6.8273	0.3650	0
0.85	1.2532	5.5380	0.6390	0
0.90	0	4.5954	5.8035	4.2585⋆

Table 6 Analysis for increasing order up-to level cost k_i^M for a time limit of 10 seconds

	GS$^{\text{sim}}$	SA$^{\text{sim}}$	GS$^{\text{sukz}}$	SA$^{\text{sukz}}$	Summary of best solution		
k_i^M [GE]	$\frac{Z^{\text{GS}}-Z^{\text{best}}}{Z^{\text{best}}}$ [%]	$\frac{Z^{SA}-Z^{\text{best}}}{Z^{\text{best}}}$ [%]	$\frac{Z^{\text{GS}}-Z^{\text{best}}}{Z^{\text{best}}}$ [%]	$\frac{Z^{SA}-Z^{\text{best}}}{Z^{\text{best}}}$ [%]	r	$\frac{\sum_{i=1}^{I} S_i}{I}$	$\frac{\sum_{i=1}^{I-1} C_i}{I-1}$
10	4.7769	3.8080	1.3678	0	137.79	119.17	6.4
100	2.4825	6.8273	0.3650	0	47.77	41.83	6.4
200	0.7790	5.3041	1.3719	0	32.03	28.67	6.4
300	3.7658	11.1363	1.6297	0	27.13	24.67	6.4
400	1.1558	5.7747	2.3543	0	24.08	21.83	6.4
500	4.7385	23.6798	1.1653	0	20.85	19.33	6.4
600	3.6688	7.6545	0.7083	0	19.15	18	6.4
700	4.9081	21.3800	1.3046	0	18.28	17.33	6.4
800	4.4501	8.2942	0.0675	0	17.90	17.33	6.4
900	2.1475	4.3479	0	0.2329	16.26	15.83	6.6
1000	1.0220	1.7667	0.9676	0	16.44	16.17	6.4

0.8519 and 0,8945 for all machines i, there are no material shortages. Based on this solution, we now analyze the system behavior when varying individual parameters.

First, we analyze the effect of the required minimum throughput TH^{min} on the performance of the approaches. Table 5 shows the relative deviations from the best solution found for all four approaches for increasing minimum throughput TH^{min}. The star (*) indicates that the 95% confidence interval of the simulation lies below the minimum throughput TH^{min}. In theses cases, the inaccuracy of the RNN leads to the fact that the minimum throughput is just no longer fulfilled. However, the deviations are very small, and therefore do not have to be significant, and may also be attributed to the simulation. This arises for very low and very high required minimum throughput TH^{min}. For moderate cases, the simulated solutions meet the minimum throughput TH^{min}. We observe a superiority of the successive solutions over the simultaneous ones, especially when using the SA approach. For low minimum throughput TH^{min}, the throughput as a function of the small buffer sizes is somewhat angular, which has to pose problems for a gradient search. The performance of GS$^{\text{sim}}$ increases with increasing TH^{min}. The higher minimum throughput results in higher required buffer spaces within the flow line. For these configurations, rounding all values up to the next integer leads to a small increase relative to the total costs. Additionally, the underlying function approximated by the RNN is more smooth for large buffers (see Fig. 2a), so the gradients provide more reliable information at these points. Thus, the GS can perform more accurately if a relatively high throughput is desired.

Second, we analyze the influence of the material cost parameter. Table 6 shows the performance analysis for increasing order up-to level cost coefficient k_i^M on the left side and the changes in the best solution found on the right side. The cost parameters for buffer sizes and delivery remain constant. The required minimum throughput is 0.8. Therefore, as order up-to level cost k_i^M increases, both the relationship to buffer costs k_i^B and to delivery cost k_i^R changes. Again, the successive optimization with SA find the best results in most cases. The simulated throughput meets the minimum

Table 7 Deviation of the best found solution and the simulated throughput with increasing flow line length for a time limit of 10 seconds

	GS^{sim}	SA^{sim}	GS^{sukz}	SA^{sukz}
$\lvert\mathcal{I}\rvert$	$\frac{Z^{\text{GS}}-Z^{\text{best}}}{Z^{\text{best}}}$ [%]	$\frac{Z^{SA}-Z^{\text{best}}}{Z^{\text{best}}}$ [%]	$\frac{Z^{\text{GS}}-Z^{\text{best}}}{Z^{\text{best}}}$ [%]	$\frac{Z^{SA}-Z^{\text{best}}}{Z^{\text{best}}}$ [%]
2	28.4663*	0*	1.5423*	0.1283*
4	1.0841	4.8353	0.2456	0
6	2.4825	6.8273	0.3650	0
10	1.5215	14.3628	0	25.1841
20	19.0693	20.5696*	2.2169	0
30	19.0599	26.3569*	2.4979	0*

throughput in all cases. The right side of Table 6 shows aggregated information on the best solution found. At low order up-to level cost k_i^M, we observe high order up-to levels and a long milkrun cycle length r. When k_i^M rises, this leads to a decrease of order up-to levels with more frequent deliveries. Hence, the material ratio $\frac{S_i}{r}$ remains constant. Within SA, our operator # 6 (see Sect. 5.1) tends to maintain approximately the relationship between cycle length and order up-to levels. Thus, the results of SA are superior to those of GS. Furthermore, the results in the last column of Table 6 show that high order up-to levels can not substitute expensive buffers. However, each of these solutions was found by a successive optimization procedure in which order up-to level costs are not taken into account in the first step. In the simultaneous optimization, slightly higher buffers can be observed.

Table 7 shows the performance of the four approaches for different flow line lengths and for a time limit of 10 seconds. Again, successive approaches lead to better solutions than simultaneous ones. For two machine lines none of the found solutions are feasible concerning the minimum throughput (as stated by the star *). This results from the higher prediction error of the RNN (see Table 3). A similar problem arises for long flow lines with 30 machines as the accuracy of the RNN decreases with increasing flow line length. To counteract this, the minimum throughput could be minimally increased within the optimization process so that the solutions remain feasible even with minor deviations from the simulation which are, as stated above, extremely accurate.

Both successive approaches SA and GS lead to promising results. The problem knowledge included in the SA approach can be helpful especially for frequent delivery and expensive line-side material. The GS worked well for instance where we can assume a smooth curve of the throughput. In steeper and more angular areas of small buffer sizes, it suffers from the regression accuracy of the RNN because the gradients are less meaningful there. For further numerical analyses on this system refer to Südbeck (2023, Chapt. 7.1, pp 177).

6 Conclusion and Further Research

In this article, we proposed an RNN to evaluate the throughput of a stochastic milkrun-supplied flow line. We showed how to configure the RNN in an efficient manner and how the selection of a flexible RNN as a special ANN allows us to analyze flow lines of variable length with a high accuracy and a speed that outperforms any discrete-event simulation. Furthermore, we showed how to systematically create the required training data for the neural network using orthogonal latin hypercube sampling. A large number of simulated flow lines can serve as a sufficient database to train the neural network. The most accurate results for extrapolating to longer flow lines have been achieved for parameter ranges with many training data points.

In addition, we showed how to use the RNN within different optimization approaches for milkrun-supplied flow lines. We developed a method that uses the approximation of a curve by the RNN to obtain better solutions using the gradients. Additionally, we implemented an SA approach with customized neighborhood operators. We tested these approaches in successive and simultaneous optimization procedures. These approaches turned out to be feasible methods to achieve an optimized flow line design in a short time using a well-trained RNN. The resulting systems show a structure that agrees well with established theoretical knowledge about flow line behavior, e.g., the bowl-shaped allocation of buffers for balanced lines. Further, optimal material supply design implies frequent replenishment cycles and lower material levels in case of expensive material storage costs.

Due to its speed, the proposed approaches can be applied for a variety of practical decision problems in production. They are especially valuable in dynamic time-dependent settings as e.g. in ramp-up production. Future research should address further applications of ANNs to stochastic production systems with different configurations and requirements. Specifically, the combination of exact or analytical methods with ANNs should be investigated.

Funding Open Access funding enabled and organized by Schmalenbach-Gesellschaft and German Academic Association for Business Research.

Data Availability The training data that support the findings of this study are openly available via https://doi.org/10.25835/8mfbqd5n.

Declarations

Conflict of interest The authors declare that they have no competing interests.

Ethical standards The authors declare to comply with the journal's ethical standards. The reported studies did not involve human participants or their data, nor did they involve plants or animals.

References

Abadi, M., A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G.S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, R. Jozefowicz, Y. Jia, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. 2015. *Tensorflow: large-scale machine learning on heterogeneous systems*

Altiok, T. 1997. *Performance analysis of manufacturing systems*. New York, NY: Springer.

Altiparmak, F., B. Dengiz, and A.A. Bulgak. 2002. *Optimization of buffer sizes in assembly systems using intelligent techniques*. Proceedings of the winter simulation conference., 1157–1162. IEEE.

Arinez, J.F., Q. Chang, R.X. Gao, C. Xu, and J. Zhang. 2020. Artificial intelligence in advanced manufacturing: current status and future outlook. *Journal of Manufacturing Science and Engineering* https://doi.org/10.1115/1.4047855.

Buzacott, J.A., and J.G. Shantikumar. 1993. *Stochastic models of manufacturing systems*. Englewood Cliffs, NJ: Prentice Hall.

Chang, Q., C. Pan, G. Xiao, and S. Biller. 2013. Integrated modeling of automotive assembly line with material handling. *Journal of Manufacturing Science and Engineering* https://doi.org/10.1115/1.4023365.

Cho, K., B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. 2014. *Learning phrase representations using rnn encoder-decoder for statistical machine translation*. Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)., 1724–1734.

Chollet, F., et al. (2015). Keras. https://github.com/fchollet/keras

Ciemnoczolowski, D.D., and Y.A. Bozer. 2013. Performance evaluation of small-batch container delivery systems used in lean manufacturing – part 2: number of kanban and workstation starvation. *International Journal of Production Research* 51(2):568–581. https://doi.org/10.1080/00207543.2012.656331.

Dallery, Y., and S.B. Gershwin. 1992. Manufacturing flow line systems: a review of models and analytical results. *Queueing Systems* 12(1):3–94. https://doi.org/10.1007/BF01158636.

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

Demir, L., S. Tunali, and D.T. Eliiyi. 2014. The state of the art on buffer allocation problem: a comprehensive survey. *Journal of Intelligent Manufacturing* 25(3):371–392. https://doi.org/10.1007/s10845-012-0687-9.

Géron, A. 2019. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: concepts, tools, and techniques to build intelligent systems*, 2nd edn. O'Reilly.

Gershwin, S.B. 1987. An efficient decomposition method for the approximate evaluation of tandem queues with finite storage space and blocking. *Operations Research* 35(2):291–305. https://doi.org/10.1287/opre.35.2.291.

Gershwin, S.B. 1994. *Manufacturing systems engineering*. Englewood Cliffs, N.J: PTR Prentice Hall.

Gershwin, S.B., and J.E. Schor. 2000. Efficient algorithms for buffer space allocation. *Annals of Operations Research* 93(1):117–144. https://doi.org/10.1023/A:1018988226612.

Helber, S. 2000. Kapitalwertorientierte Pufferallokation in stochastischen Fließproduktionssystemen. *Schmalenbachs Zeitschrift für betriebswirtschaftliche Forschung* 52(5):211–233. https://doi.org/10.1007/BF03372616.

Helber, S. 2001. Cash-flow-oriented buffer allocation in stochastic flow lines. *International Journal of Production Research* 39(14):3061–3083. https://doi.org/10.1080/00207540110056144.

Jang, K.Y., K. Yang, and C. Kan. 2003. Application of artificial neural network to identify non-random variation patterns on the run chart in automotive assembly process. *International Journal of Production Research* https://doi.org/10.1080/0020754021000042409.

Kiefer, J. 1953. Sequential minimax search for a maximum. *Proceedings of the American Mathematical Society* 4(3):502. https://doi.org/10.1090/S0002-9939-1953-0055639-3.

Kingma, D.P., and J.L. Ba. 2015. *Adam: a method for stochastic optimization: presented as a conference paper at the 3rd international conference for learning representations*

Li, J., and S.M. Meerkov. 2009. *Production systems engineering*. Boston, MA: Springer.

Li, C., H. Wang, and B. Li. 2016. Performance prediction of a production line with variability based on grey model artificial neural network. In *Proceedings of the 35th Chinese control conference*, ed. J. Chen, Q. Zhao, 9582–9587. Piscataway, NJ: IEEE.

Li, J., D.E. Blumenfeld, and J.M. Alden. 2006. Comparisons of two-machine line models in throughput analysis. *International Journal of Production Research* 44(7):1375–1398.

Li, J., D.E. Blumenfeld, N. Huang, and J.M. Alden. 2009. Throughput analysis of production systems: recent advances and future topics. *International Journal of Production Research* 47(14):3823–3851. https://doi.org/10.1080/00207540701829752.

Metropolis, N., A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller. 1953. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics* 21(6):1087–1092. https://doi.org/10.1063/1.1699114.

Mindlina, J., and H. Tempelmeier. 2021. Performance analysis and optimisation of stochastic flow lines with limited material supply. *International Journal of Production Research* https://doi.org/10.1080/00207543.2021.1954712.

Papadopoulos, H.T., and C. Heavey. 1996. Queueing theory in manufacturing systems analysis and design: a classification of models for production and transfer lines. *European Journal of Operational Research* 92(1):1–27. https://doi.org/10.1016/0377-2217(95)00378-9.

Papadopoulos, C.T., M.E.J. O'Kelly, M.J. Vidalis, and D. Spinellis. 2009. *Analysis and design of discrete part production lines*. New York, NY: Springer.

Papadopoulos, C.T., J. Li, and M.E.J. O'Kelly. 2019. A classification and review of timed markov models of manufacturing systems. *Computers & Industrial Engineering* 128:219–244. https://doi.org/10.1016/j.cie.2018.12.019.

Rosen, J.B. 1960. The gradient projection method for nonlinear programming. Part I. Linear constraints. *Journal of the Society for Industrial and Applied Mathematics* 8(1):181–217.

Rosen, J.B. 1961. The gradient projection method for nonlinear programming: Part II. Nonlinear constraints. *Journal of the Society for Industrial and Applied Mathematics* 9(4):514–532.

Rumelhart, D.E., G.E. Hinton, and R.J. Williams. 1986. Learning representations by back-propagating errors. *Nature* 323:533–536.

Shi, C. 2012. *Efficient buffer design algorithms for production line profit maximization*. Ph. D. thesis, Massachusetts Institute of Technology.

Spinellis, D.D., C. Papadopoulos, and J. MacGregor Smith. 2000. Large production line optimization using simulated annealing. *International Journal of Production Research* 38(3):509–541. https://doi.org/10.1080/002075400189284.

Südbeck, I. 2023. *Analyse und Optimierung stochastischer Fließproduktionssysteme mit begrenzter Materialverfügbarkeit unter Verwendung von Methoden des maschinellen Lernens*, 1st edn., Norderstedt: Books on Demand.

Südbeck, I., J. Mindlina, A. Schnabel, and S. Helber. 2022. November. Using recurrent neural networks for the performance analysis and optimization of stochastic milkrun-supplied flow lines. *Technical Report* https://doi.org/10.1007/s00291-023-00737-9.

Tan, B., and S. Khayyati. 2021. Supervised learning-based approximation method for single-server open queueing networks with correlated interarrival and service times. *International Journal of Production Research* https://doi.org/10.1080/00207543.2021.1887536.

Tsadiras, A.K., C.T. Papadopoulos, and M. O'Kelly. 2013. An artificial neural network based decision support system for solving the buffer allocation problem in reliable production lines. *Computers & Industrial Engineering* 66(4):1150–1162. https://doi.org/10.1016/j.cie.2013.07.024.

Wang, L., F. Sun, D. Lin, and M.Q. Liu. 2018. Construction of orthogonal symmetric latin hypercube designs. *Statistica Sinica* https://doi.org/10.5705/ss.202017.0075.

Weiss, S., and R. Stolletz. 2015. Buffer allocation in stochastic flow lines via sample-based optimization with initial bounds. *OR Spectrum* 37(4):869–902. https://doi.org/10.1007/s00291-015-0393-z.

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

Weiss, S., J.A. Schwarz, and R. Stolletz. 2019. The buffer allocation problem in production lines: formulations, solution methods, and instances. *IISE Transactions* 51(5):456–485. https://doi.org/10.1080/24725854.2018.1442031.

Yan, C.B., Q. Zhao, N. Huang, G. Xiao, and J. Li. 2010. Formulation and a simulation-based algorithm for line-side buffer assignment problem in systems of general assembly line with material handling. *IEEE Transactions on Automation Science and Engineering* 7(4):902–920. https://doi.org/10.1109/TASE.2010.2046892.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.