Khayyati, S.* and **Tan, B.**, "Khayyati, S.* and **Tan, B.**, "A Machine Learning Approach for Implementing Data-driven Production Control Policies," *International Journal of Production Research*, Vol. 60, No. 10, 3107-3128, 2022.

https://doi.org/10.1080/00207543.2021.1910872

A Machine Learning Approach for Implementing Data-Driven Production Control Policies

Siamak Khayyati
Department of Industrial Engineering, Koç University, Rumeli Feneri Yolu, Istanbul, Turkey,34450,
Email: skhayyati13@ku.edu.tr

Barış Tan
College of Administrative Sciences and Economics, Koç University, Rumeli Feneri Yolu, Istanbul, Turkey, 34450,
Email: btan@ku.edu.tr

Given the extensive data being collected in manufacturing systems, there is a need for developing a systematic method to implement data-driven production control policies. For an effective implementation, first, the relevant information sources must be selected. Then, a control policy that uses the real-time signals collected from these sources must be implemented. We analyze the production control policy implementation problem in three levels: choosing the information sources, forming clusters of information signals to be used by the policy, and determining the optimal policy parameters. Due to the search-space size, a machine-learning-based framework is proposed. Using machine learning speeds up optimization and allows utilizing the collected data with simulation. Through two experiments, we show the effectiveness of this approach. In the first experiment, the problem of selecting the right machines and buffers for controlling the release of materials in a production/inventory system is considered. In the second experiment, the best dispatching policy based on the selected information sources is identified. We show that selecting the right information sources and controlling a production system based on the real-time signals from the selected sources with the right policy improve the system performance significantly. Furthermore, the proposed machine learning framework facilitates this task effectively.

Key words: production control, stochastic models, machine learning, real-time control, discrete event simulation

1. Introduction

Advances in the information collection technologies in manufacturing settings has increased the amount of information collected significantly (Tao et al. 2018). Despite the increase in the data collected, implementing data-driven production control policies still faces challenges in practice (Moeuf et al. 2018). One of these challenges is the mismatch between the amount of information available on the shop-floor and the amount of information needed for an effective production control policy for the system (Kusiak 2017).

On one end of the spectrum, no information or no relevant information is available about the

system, and the relevant information has to be inferred based on the data collected from the system. On the other end of the spectrum, the large quantity of the available data, makes discerning the relevant information a complicated task and renders most available modeling and optimization tools computationally prohibitive.

Although modeling tools and control policies for manufacturing systems have been studied extensively, the state of the art optimization tools are not often capable of modeling and optimizing complex systems effectively. Even in smaller models, incorporating partial information into the control policy can complicate the modeling and optimization of the system. In such cases, methods that utilize machine learning allow implementing effective production control policies (Cadavid et al. 2020).

Following the advances in real-time data collection in manufacturing systems, there is a need for developing a systematic method to implement data-driven production control policies effectively. In order to implement a data-driven control policy effectively, first, the most relevant information sources to control production flows in a given system must be identified. Then information clusters that will be used by the control policy must be formed based on the real-time data about the status of the selected information sources. These information clusters are referred as the *markings* in this study. Finally, the optimal parameters of the marking-dependent production control policy that authorizes production or the release of material into the system based on the real-time markings collected from the shop floor must be determined.

As a summary, we consider the overall problem of controlling a system with limited data and many sources of information. This problem is then broken down to three levels:

1. choosing the information sources (P_1),
2. forming the markings by clustering the signals from the selected information sources (P_2) and
3. determining the parameters of the optimal marking-dependent policy based on the selected markings (P_3).

As the number of information sources and the number of markings used to control production increase, implementing a complicated production control policy in a large-scale system leads to difficulties in implementation. For example, even for the small production system with 3 stations and 3 buffers we analyze in Section 9.1 as the first experimental setup, Problem P_2 has 2×10^{50} feasible solutions. In addition, the computational burden to determine the optimal control parameters increases exponentially. In this setting, using machine learning algorithms, more specifically, clustering algorithms for P_2 and supervised learning algorithms for P_1, P_2, and P_3 facilitates finding the solutions of these problems efficiently. In this study, we formulate P_1 and P_2 as regression problems and use supervised learning algorithms to determine the solutions.

The extensive use of sensors and digital machines has contributed to the prevalence of *big data problem* in manufacturing settings (Zhong et al. 2016). A reduction in the amount of information that is required for the control policy and the parameters that need to be optimized can be helpful in overcoming the challenges related to this problem. Furthermore, it has been shown that increasing the number of control parameters has diminishing returns for a marking-dependent control policy (Khayyati and Tan 2020). In a typical manufacturing system, the number of system states to consider is large and simulations that are used to generate the data points require significant time and computational resources to run. Consequently, optimizing policies that take into account the state of the whole system is specifically prone to yield the *big data problem* in manufacturing settings. This results in a difficulty in identifying the most important relations between system states, control parameters and system costs.

The main contribution of this work is proposing a machine learning framework for implementing a data-driven production control policy in partially observable production systems. This method includes selection of the information sources and clustering the signals from these sources to control production. Using clusters of information signals rather than all the information signals significantly increases the efficiency of optimizing the marking-dependent policy. As a result, the proposed method can be used to implement a real-time production control policy effectively.

The remainder of this work is organized as follows. Section 2 gives a review of the pertinent literature. The marking-dependent control policy is introduced in Section 3. Section 4 describes the partially observable production system we consider. The problem considered in this study is stated formally in Section 5. Section 6 gives the Discrete Event Simulation algorithm used to evaluate the performance of the system. The solution method we propose is discussed in Section 7. Section 8 explains how the data for the regression problem is generated to optimize the system. Section 9 gives the numerical experiments we have performed. Finally, Section 10 concludes the paper.

2. Literature Review

In the following, we give an overview of the literature on using machine learning methods for performance evaluation and control of production systems.

2.1. Machine Learning for Control of Production Systems

Machine learning has been used for controlling operations and processes in manufacturing (Monostori et al. 1996, Irani et al. 1993, Charest et al. 2018, Cadavid et al. 2020, Hwang and Jang 2020). It has also been used for controlling the manufacturing systems at the planning level (Priore et al. 2001). Can and Heavey (2012) investigate building meta models of Discrete Event Simulation (DES) models using DNN and genetic programming. A meta model of a DES refers to a function that maps the space of decision variables to the space of a performance measure, e.g., mapping the

base-stock level to the average cost or in a buffer allocation problem, different buffer allocations are considered the data points (the number of space allocated to each buffer is the feature) and the throughput is the response. The development of a meta model can both help optimize the DES by optimizing the meta model and it can be used as an easy to evaluate replacement for the DES in optimization problems with larger scope. Furthermore, uniform sampling of the solution space is used for generating the data. We aim to show the improvement resulting from machine learning in the process of optimization, referred to here as learning while optimizing, as well.

Li and Olafsson (2005) use machine learning more explicitly in the optimization procedure as opposed to using machine learning for estimating a performance measure. Li and Olafsson (2005) consider the dispatching problem as a classification problem, and uses machine learning to discriminate between two jobs, given the attributes of the jobs, and decide which one to dispatch earlier? For using machine learning in more complicated settings, agent based models have been suggested (Aissani et al. 2008, Monostori et al. 2006). Reinforcement learning methods have been applied to control systems with degrading machines and managing the delivery of material in manufacturing settings (Paraschos et al. 2020, Hwang and Jang 2020). Reinforcement learning methods have the capacity to be applied to a large variety of problems. However, there are drawbacks associated with applying them in practice. Difficulties in working effectively with partial information, handling systems with a large state-action space and generating explainable models have been identified as the main drawbacks of using reinforcement learning in practise. (Dulac-Arnold et al. 2019). By selecting the most relevant sources of information and forming a manageable numbers of groups of partial information signals, we decrease the effect of the large size of the state space and generate a relatively more interpretable policy. Zhang et al. (2020) gives a detailed review of the various applications of machine learning in manufacturing.

2.2. Machine Learning for Performance Evaluation

There are many studies in the literature that aim at predicting different performance measures related to a production system. Lingitz et al. (2018) use different machine learning methods for lead time prediction for a semiconductor producer. Alenezi et al. (2008) present a method to predict the time it takes for a new order that has arrived to be completed for the purpose of providing a due date with a given reliability based on snapshots of the production system (number of items in the buffers and machine availability) using support vector regression. Karaoglan and Karademir (2017) use neural networks for predicting flow times. Backus et al. (2006) aim at predicting the cycle time for a lot that is being produced based on historical data and features such as WIP in different parts of the system and the cycle time for similar lots that have gone through the same machines. They conclude that a regression tree is optimal for this task. Machine learning can also be used for detecting disturbances in the performance of a production system (Farooqui et al. 2020).

This work differs with the other works in the literature in three main ways. First, we consider the problem of selecting the sources of information alongside the problems related to finding the best policy. This approach addresses both the problem of determining the location of sensors in a production system and also the issues that might arise in implementation of real-time production control policies due to a lack of efficient solution methods for complex representations of a production system. To the best of our knowledge, this problem has not been addressed in this setting before. Second, the use of marking-dependent policy allows for determining an easy to implement policy that can utilize the information gathered from various locations in the production system. Third, our use of machine learning in determining the control policy allows for integration of data coming from various sources including historical data from different times despite potential changes in the system and simulated data.

3. Marking-Dependent Production Control Policy

In this work, we focus on implementing a specific data-driven control policy that is designed to utilize the partial information signals, clusters of which are referred as *markings*, emitted from the system in controlling the release of material into different parts of the system. This policy is an extension of the *marking-dependent threshold policy* introduced for production/inventory systems (Khayyati and Tan 2020). The proposed *marking-dependent production control policy* decides on the authorization of the release of material into the different parts of a production system for each of the identified markings and enforces the release authorization based on the last observed marking.

Figure 1 depicts a production system where the release of material into the system is controlled by using a given marking-dependent policy that utilizes information from two source of information.

The marking-dependent production policy differs from a state-dependent policy in that it does not assign a different set of actions to each possible state of the system. Instead, the observable information about the state of the system is summarized in the markings and the actions of the controller depend on the markings. This allows for a significant reduction of the number of control parameters for partially observable production systems. This reduction allows determining the optimal control parameters efficiently. As a result, the marking-dependent production control policy can be implemented as a real-time production control policy effectively.

4. Model

In the following, we give the formal description of the system under study, the marking-dependent policy and the optimization problem for determining the best parameters for the marking-dependent policy. Table 1 gives the description of the notation used throughout the paper.

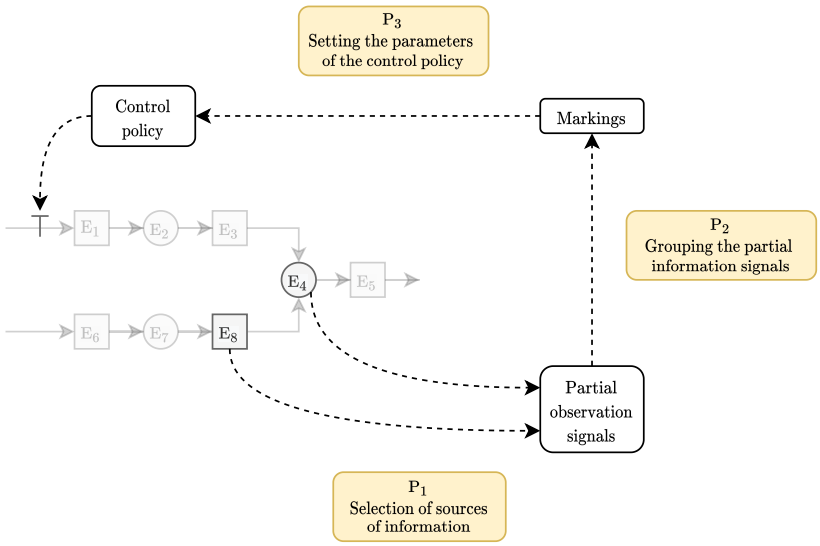

Figure 1 **Implementing a marking-dependent release control of a production system with information source selection (P_1), partial information signal grouping (P_2), and optimal control policy parameter setting (P_3) subproblems**

4.1. Production Plant

A production plant consists of n elements. Each element $E_i, i \in \{1, \ldots, n\}$ is a workstation or a buffer. The connections between these elements are referred to as the plant structure, represented by a matrix $\mathbf{C}$ whose (i, j)th entry indicates if element i feeds into element j. We consider a discrete state-space continuous time process describing the dynamics of a system consisting of stations and finite buffers. This representation is quite general and allows considering all production systems that can be modelled as open queueing networks with blocking.

Station i has m_i working states. Let $\eta = (\eta_1, \ldots, \eta_n)$ denote the state of the system, where $\eta_i \in A_i$ and A_i is the finite set of possible states for element E_i. The possible states for station i include $A_i = \{W_1, \ldots, W_{m_i}, B, S\}$, where W_j refers to the jth working state of station i including the down state, B indicates that station i is blocked, and S refers to starvation of station i. The state of buffer i is the number of items in the buffer $A_i = \{0, 1, \ldots, \omega_i\}$ where ω_i is the size of buffer i. The set of η tuples is denoted with $\{\eta\}$.

The parameters of all the elements, e.g., the statistical characteristics of the time spent in each state and the transition processes among the states for the machines and the buffer sizes are given in set $\mathbf{\Lambda}$. For example, for a production line with two reliable stations separated with a finite buffer with a capacity of ω_2 where the processing times are exponential random variables with rates μ_1 and μ_3, $\mathbf{\Lambda}$ includes the processing rates and the buffer size as the parameters of the system, i.e., $\mathbf{\Lambda} = \{\mu_1, \omega_2, \mu_3\}$. In case some or all of the parameters of the system are not all known a priori, the estimates of the parameters of the system based on the available traces and available

Table 1 The notation used throughout the paper (excluding the notation locally used in the discrete event simulation algorithm and the numerical experiments)

	Description
P_1	The problem of selecting the sources of information
P_2	The problem of forming the markings
P_3	The problem of setting the control parameters for a marking-dependent policy
$\mathbf{\Lambda}$	The parameters of a production system
$\bar{\mathbf{\Lambda}}$	The estimate values of the parameters of the production system
$\mathbf{I}$	The vector determining the sources of information in the production system
$\mathbf{Y}$	The matrix that defines the relation between the partial information observations and the markings
$\mathbf{W}$	A unique vector representation for $\mathbf{Y}$
$\mathbf{U}$	The parameters of the marking-dependent policy that govern the release of material into the elements of the production system based on each marking
Π	The cost of the production system
N	The number of collections of traces available
D^r	The rth collection of traces
$\bar{\Pi}\left(\mathbf{I},\mathbf{Y},\mathbf{U},\bar{\mathbf{\Lambda}},D^r\right)$	The estimated cost of the system based on D^r
E_i	The ith element in the production system (buffer or machine)
$\mathbf{C}$	The matrix that shows the connection of elements in a production system
n	Number of elements in the production system
η	State of the system
η_i	The state of element E_i
A_i	Set of possible states for element E_i
m_i	Number of working states of element i including the down state
ω_i	Buffer size of element i
I_i	The decision to use E_i as a source of information
$C\left(\mathbf{I}\right)$	The total cost of collecting information from sources specified by $\mathbf{I}$
$\hat{\eta}$	The state of the system based on the partial observations of system state
K	Number of markings
χ_i	Indicator that shows if the release into E_i can be controlled or not
$\bar{\Pi}^3\left(\mathbf{I},\mathbf{Y},\bar{\mathbf{\Lambda}}\right)$	Best cost for given markings and selected information sources
$\bar{\Pi}^2\left(\mathbf{I},\bar{\mathbf{\Lambda}}\right)$	Best cost for given selected information sources
$\bar{\Pi}^1\left(\bar{\mathbf{\Lambda}}\right)$	Best cost for the system
$\mathcal{I}_C$, $\mathcal{W}_C$	The set of candidate $\mathbf{I}$ vectors and the set of candidate $\mathbf{W}$ vectors
X, Y	The data matrix and the output vector for the regression problems
$f_{\mathbf{I}}$, $f_{\mathbf{W}}$	The regression models built for the selection of information sources and formation of markings respectively
$\mathcal{D}^{r,l}$	The rth data set available related to the lth set of system parameters
$\bar{\mathbf{\Lambda}}^l$	The estimates of the parameters of the system for instance l
$\mathbb{I}^l$, $\mathbb{W}^l$	The set of $\mathbf{I}$ and $\mathbf{W}$ tuples respectively evaluated based on the data available for the lth instance

information are included in $\bar{\mathbf{\Lambda}}$. Following the previous example of the two-station production line, if the parameters μ_1 and μ_3 are not known a priori, the available traces can be used to determine the estimates for μ_1 and μ_3 denoted with $\bar{\mu}_1$ and $\bar{\mu}_3$ respectively. In this case, $\bar{\mathbf{\Lambda}} = \{\bar{\mu}_1, \omega_2, \bar{\mu}_2\}$.

4.2. Information sources

Each element can be used as a source of information about the production plant. Let $I_i \in \{0,1\}$ denote the decision to use E_i as a source of information and let $C\left(\mathbf{I}\right)$ denote the total cost of collecting information from these sources, where $\mathbf{I} = (I_1, \ldots, I_n)$.

In the two-station production line example, if the interstation buffer and the second station are used as the information sources for the production control policy to decide on the release of the parts into the first station, $I_1 = 0$, $I_2 = 1$, $I_3 = 1$, and $\mathbf{I} = (0, 1, 1)$.

Then let $\hat{\eta} = (\hat{\eta}_1, \ldots, \hat{\eta}_n)$ where $\hat{\eta}_i = \begin{cases} \eta_i, \text{ if } I_i = 1 \\ -, \text{ if } I_i = 0 \end{cases}$ denotes the state of the system based on the partial observations of system state. In other words, if element E_i is not selected to be used as an information source or if the state of that element is not observable, the state of the system with partial information, $\hat{\eta}$ does not include the state of E_i. The set of $\hat{\eta}$ tuples is denoted with $\{\hat{\eta}\}$.

In the two-station production line example, when the buffer capacity is 2, the state space of the system has 5 states:

$$\{\eta\} = \{(\mathrm{W}_1, 0, \mathrm{S}), (\mathrm{W}_1, 0, \mathrm{W}_1), (\mathrm{W}_1, 1, \mathrm{W}_1), (\mathrm{W}_1, 2, \mathrm{W}_1), (\mathrm{B}, 2, \mathrm{W}_1)\}.$$

However, since the information on the first station is not collected, the state space of the system based on the partial observation of the system state has 4 states:

$$\{\hat{\eta}\} = \{(-, 0, \mathrm{S}), (-, 0, \mathrm{W}_1), (-, 1, \mathrm{W}_1), (-, 2, \mathrm{W}_1)\}.$$

4.3. Markings

Markings are the clusters of partial information signals used to decide on authorizing the release of material into a given element. Markings are obtained from the observable system states. Namely, the set of possible partial observations of the state of the system, $\hat{\eta}$, is mapped into a finite set of markings $\{1, \ldots, K\}$, where K is the total number of markings used by the control policy. We assume that the number of markings K is given. That is, we focus on controlling the release of parts into a production system with a given number of markings. Once the problem of determining the best K markings is solved, K can also be optimized by following a similar approach presented in this study.

Since the set of markings is much smaller than the set of $\hat{\eta}$ tuples, the assignment of $\hat{\eta}$ tuples to different markings can be viewed as putting $\hat{\eta}$ tuples in K clusters. Let $y_{i,k} \in \{0, 1\} : i \in \{1, \ldots, |\{\hat{\eta}\}|\}, k \in \{1, \ldots, K\}$ denote the decision to place partial observation tuple i in the cluster corresponding to marking k, $y_{i,k} = 1$, or $y_{i,k} = 0$ otherwise. Let $\mathbf{Y} = \{y_{i,k}\}$ denote the matrix that specifies a given formation of markings.

In the two-station production line example where the interstation buffer and the second machine are used as the information sources for the production control policy, let us assume that the buffer capacity is 2 and the release is allowed when there are less than 2 parts in the buffer. In this case, only two markings that track the number of parts in the buffer with respect to the set threshold

are used in the control policy, i.e., $K = 2$. One marking is used for the signals that show that there are less than 2 parts in the buffer and the other marking is used for the signal that the buffer has 2 parts and the second station is working. Accordingly, all the states that have less than 2 part in the buffer, $\{(-, 0, \mathrm{S}), (-, 0, \mathrm{W}_1), (-, 1, \mathrm{W}_1)\}$ are mapped into the first marking and the state that has 2 parts in the buffer, $\{(-, 2, \mathrm{W}_1)\}$ into the second marking. Accordingly,

$$\mathbf{Y} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

4.4. Marking-dependent control policy

The release of material into a given number of elements in the plant is controlled based on the last observed marking from the system. Let $\chi_i \in \{0, 1\}$ denote an indicator that shows if the release into E_i can be controlled (1) or not (0). We consider χ_i to be a predetermined characteristic of the system. If χ_i is set to 0 for element i, the release of material into element i will be always allowed.

Let $\mathbf{u}_j = (u_{1,j}, \ldots, u_{n,j})$ denote the control tuple in the system where $u_{i,j} \in \{0, 1\}$ denotes the decision to release material into E_i when the last observed marking is j. If the release is authorized, $u_{i,j} = 1$, and if it is not authorized $u_{i,j} = 0$.

When the release of material into E_i is not allowed, if E_i is a workstation, it will be starved upon completing the processing of a part. If E_i is a buffer, the workstation that feeds to it will be blocked upon completing the processing of a part.

In the two-station production line example where the release of material into the first station is allowed when there are less than 2 parts in the buffer, assuming only the release of parts into the first station can be controlled, i.e., $\chi_1 = 1, \chi_2 = 0, \chi_3 = 0$,

$$\mathbf{u}_1 = (1, 0, 0), \mathbf{u}_2 = (0, 0, 0).$$

4.5. Policy parameters

The parameters of the marking-dependent policy with K markings are K tuples $\mathbf{u}_1, \ldots, \mathbf{u}_K$ indicating the decision for all the elements for each marking. Let

$$\mathbf{U} = \begin{bmatrix} \mathbf{u}_1 \\ \vdots \\ \mathbf{u}_K \end{bmatrix} \tag{1}$$

denote the matrix specifying all the parameters of the marking-dependent policy.

In the two-station production line example when the release of material into the first station is allowed when there are less than 2 parts in the buffer,

$$\mathbf{U} = \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \end{bmatrix} = \begin{bmatrix} 1,0,0 \\ 0,0,0 \end{bmatrix}.$$

A marking-dependent threshold policy can be a specific case of the marking-dependent policy if the formation of the markings allows implementing a threshold policy. In Section 9.1, we demonstrate how different threshold policies can be implemented as marking-dependent policies in the production/inventory system analyzed.

5. Problem Description

Let $\Pi(\mathbf{I}, \mathbf{Y}, \mathbf{U}, \mathbf{\Lambda})$ denote the long-run total cost of the system with parameters $\mathbf{\Lambda}$ when it is controlled with the information sources $\mathbf{I}$, markings specified by $\mathbf{Y}$ and marking-dependent policy $\mathbf{U}$. The relevant operation, inventory carrying, and service-level related costs are included in the total cost function together with the cost $C(\mathbf{I})$ related to controlling the system with the selected sources given in $\mathbf{I}$.

In the numerical experiments, we focus on the problem of determining the best locations of a given number of information sources. A given limit L on the number of sources of information can be imposed by setting the total cost of using selected information sources, $C(\mathbf{I}) = \begin{cases} 0, & \text{if } \sum_i I_i \leq L \\ \infty, & \text{if } \sum_i I_i > L \end{cases}$. Since the cost function does not vary in the examples we considered, we use $\Pi(\mathbf{I}, \mathbf{Y}, \mathbf{U}, \mathbf{\Lambda})$ instead of $\Pi(\mathbf{I}, \mathbf{Y}, \mathbf{U}, \mathbf{\Lambda}, C)$, for notational convenience.

The main problem considered in this study is minimizing the total long-run cost of the system:

$$\min_{(\mathbf{I},\mathbf{Y},\mathbf{U})} \Pi(\mathbf{I}, \mathbf{Y}, \mathbf{U}, \mathbf{\Lambda}).$$

Since $\Pi(\mathbf{I}, \mathbf{Y}, \mathbf{U}, \mathbf{\Lambda})$ is not available, it is estimated using N different datasets that include collections of traces available about different stochastic processes in the production system that are necessary for simulating the system. These traces can be of different lengths due to the frequency of the events related to them. For example, the available traces for slow machines are shorter than available traces for faster machines. Let D^r denote the rth data set available, $r = 1, \ldots, N$. Then, the dataset is represented as $\{D^r : 1 \leq r \leq N\}$. Each data set contains the interevent times and machine state information for different elements in the system. Using this data, a discrete-event simulation algorithm can simulate the system and calculate the performance measures of interest. Section 6 gives a more detailed description of these datasets and the Discrete Event Simulation algorithm.

The aim is to find the best information sources, the best markings based on the states of the information sources, and the parameters of the marking-dependent policy based on the selected

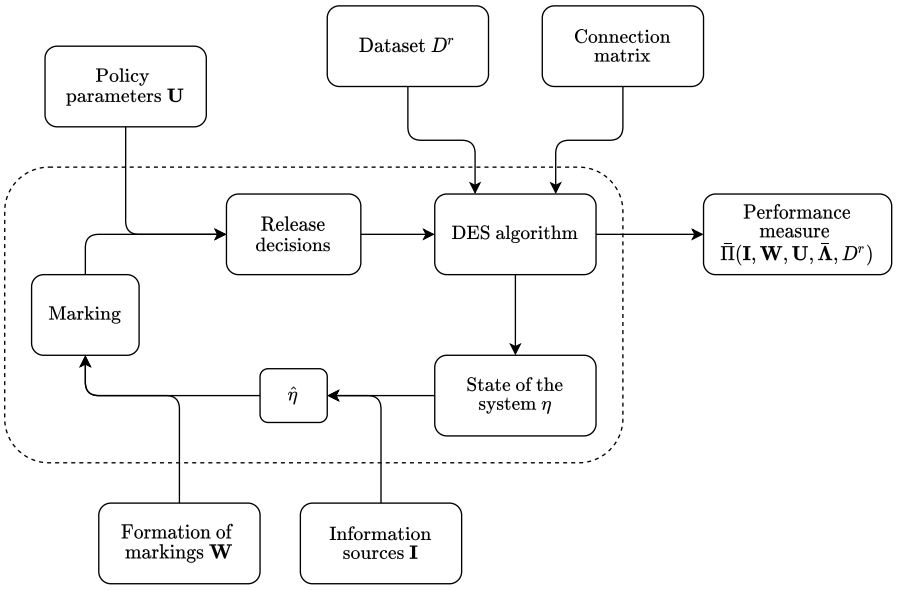

Figure 2 Evaluation of the production system

markings for the production system using the available data sets $\{D^r : 1 \leq r \leq N\}$. The estimated cost is denoted with $\bar{\Pi}\left(\mathbf{I}, \mathbf{Y}, \mathbf{U}, \bar{\mathbf{\Lambda}}, D^r\right)$ where $\bar{\mathbf{\Lambda}}$ denotes the exact values or approximations of the system parameters either known a priori or approximated from the available traces.

Given these definitions, the problem can be stated as

$$\min_{(\mathbf{I},\mathbf{Y},\mathbf{U})} \frac{\sum_{r=1}^{N} \bar{\Pi}\left(\mathbf{I}, \mathbf{Y}, \mathbf{U}, \bar{\mathbf{\Lambda}}, D^r\right)}{N}. \tag{2}$$

In order to explain the setting, consider the example given in Figure 1. In this production system with 5 machines and 3 buffers, the solution of this problem may yield selection of the buffer E_4 and the machine E_8 as the information sources. Once the information sources are selected, different combinations of state of E_8, whether it is blocked or not, and whether the buffer E_4 is full, empty, or partially full can be grouped together to form two markings. Then the decisions to release the material into the machine E_1 will depend on the observation of these signals. For example, observing that the machine E_8 is starved while the buffer E_4 is empty at a given time may allow release of material while observing E_8 is working and E_4 is full may not allow production on the machine E_1. The objective is making these selections optimally for a given system.

6. The Discrete Event Simulation Algorithm

The long-run cost of the system $\bar{\Pi}\left(\mathbf{I}, \mathbf{Y}, \mathbf{U}, \bar{\mathbf{\Lambda}}, D^r\right)$ given in Equation (2) is estimated by using simulation. Figure 2 depicts the evaluation of the production system to estimate the long-run cost of the system.

In the following, we give the Discrete Event Simulation (DES) algorithm for simulating a production system with the marking-dependent policy and given a dataset.

6.1. Dataset

In this section, the DES algorithm is given for one replication of the data without loss of generality. Hence, the superscript r that shows the index of the replication for the dataset D^r has been omitted for notational convenience.

The dataset D includes the interevent times for each element in isolation, the working states of the elements following the completion of an event, and the indicators that show whether the events are related to a service completion or a change in the working state without the completion of a service, e.g., due to breakdown and repair events. Hence, $D = \{\Delta_{i,l,c_i} : i \in \{1,\dots,n\}, l \in \{1,2,3\}, c_i \in \{1,\dots,|c_i|\}\}$ where $|c_i|$ is the length of the trace available for element i, $\Delta_{i,1,c_i}$ is the c_ith interevent time, $\Delta_{i,2,c_i}$ is the state following the c_ith event, and $\Delta_{i,3,c_i}$ is an indicator variable that shows whether the c_ith event is a process completion for element i. These parameters of the dataset D can be obtained from a Manufacturing Execution System database or generated by simulating the elements in isolation with the available data. All the variables used in the DES algorithm are described in Table 2.

For the example system with two stations with exponential processing times, the dataset includes the interevent times that are the processing times for station 1, $\Delta_{1,1,c_1}$, $c_1 \in \{1,\dots,|c_1|\}$ and the processing times for station 2, $\Delta_{3,1,c_3}$, $c_3 \in \{1,\dots,|c_3|\}$. These interevent times can be generated as exponential random variates with rates μ_1 and μ_2. Furthermore, since the stations are reliable, the state of the stations will be W_1 following each process completion. That is $\Delta_{1,2,c_1} = \mathrm{W}_1$, $c_1 \in \{1,\dots,|c_1|\}$ and $\Delta_{3,2,c_3} = \mathrm{W}_1$, $c_3 \in \{1,\dots,|c_3|\}$. Finally, since all the events are process completions in this example, the indicator variables will be the same, i.e., $\Delta_{1,3,c_i} = \Delta_{3,3,c_i} = 1$, $c_i \in \{1,\dots,|c_i|\}$.

This definition of traces is based on the assumption that the changes in the working states of a machine are process dependent. The failures, repair times or different down or up states are modeled as different working states. The blocking and starvation times are generated using the DES algorithm depending on the markings and the policy parameters.

6.2. The DES Algorithm

The DES algorithm presented here can model production/inventory systems, serial lines with heterogeneous parallel stations, assembly lines, and a combination of these systems. The methodology presented in this study to implement the marking-dependent production control policy can be used for other production systems with their associated DES algorithms. For modeling production/inventory systems, synchronization stations are modeled as stations with multiple incoming

Table 2 The description of the variables used in the DES algorithm

Variable	Description
NIS_i	Number of input streams to element i
$\mathcal{AB}$	Set of available buffers (downstream or upstream)
$\mathcal{DB}$	Set of downstream buffers
$\mathcal{UB}$	Set of upstream buffers
$\mathcal{CB}$	Set of upstream buffers chosen for elements with more than one input stream
$\Delta_{i,1,c_i}$	The c_ith inter-event time for element i
$\Delta_{i,2,c_i}$	The state that the c_ith event for element i takes it to
$\Delta_{i,3,c_i}$	The indicator that shows if the c_ith event causes the completion of a service
c_i	Inter-event time counter for element i
z	Indicator for flagging if resolution of blocking and starvation might be needed
FEL	Future event list

streams with zero processing time. In this model, the machines can be in various working states in addition to the blocked and the starved states. The various working states can be used for modeling downtimes as well. Table 2 gives the description of variables used in the DES algorithm.

Let FEL denote the future event list. The event list includes the information, $\mathbb{E} \in FEL$, regarding the changes in the system at specific times. Namely, the information for each change is recorded in $\mathbb{E} = (\mathbb{e}_1, \mathbb{e}_2, \mathbb{e}_3, \mathbb{e}_4)$ where $\mathbb{e}_1$ denotes the event time, $\mathbb{e}_2$ denotes the element that is affected directly by event $\mathbb{E}$, $\mathbb{e}_3$ denotes the state of element $\mathbb{e}_2$ after the event takes place, and $\mathbb{e}_4$ is an indicator that shows if the event is a process completion or a change in the state of element $\mathbb{e}_2$ without the completion of a process. If the controller cannot modify the release decision for element i ($\chi_i = 0$), we assume that the release of material into element i is always allowed, i.e., we set $u_{i,k} = 1$, $\forall i, k$.

Algorithm 1 gives the DES pseudocode for simulating the production system and Algorithm 2 gives the submodule for resolving blocking and starvation. In summary, this DES algorithm modifies the state of the system (η) locally based on the earliest future event and the element it affects, i.e., $\eta_{\mathbb{e}_2}$ where $\mathbb{E} = (\mathbb{e}_1, \mathbb{e}_2, \mathbb{e}_3, \mathbb{e}_4)$ is the earliest event in FEL. Then the algorithm checks if the state variable for other elements ($\eta_j, j \neq \mathbb{e}_2$) needs to change as well due to blocking or starvation being resolved. If the state of the system η is changed, then the check is repeated, otherwise the next earliest event is identified, and the algorithm continues. The algorithm stops when the trace provided for one of the elements is fully processed.

7. Solution Methodology

Once the dataset that includes the collected data as well as the synthetic data obtained by simulation, $\{D^r,\ r = 1, \ldots, N\}$ is obtained, the Discrete Event Simulation algorithm is used with the dataset to estimate the long-run costs for different selections of information sources, markings, and production control policy parameters. Then, the three subproblems of the problem given in Equation (2) for a production system with the estimated parameters $\bar{\mathbf{\Lambda}}$, the information sources $\mathbf{I}$, markings $\mathbf{Y}$, and the marking-dependent control policy $\mathbf{U}$ are formally stated as

Algorithm 1 DES pseudo code

while $|FEL|>0$ **do**
  $\mathbb{E}\leftarrow\mathbb{E}^{'}:\mathbb{e}_1^{'}\leq\mathbb{e}_1^{''}\forall\mathbb{E}^{''}\in FEL$ ▷ Identifying the earliest event in FEL
  $T\leftarrow\mathbb{e}_1$; $i\leftarrow\mathbb{e}_2$
  if $\mathbb{e}_4=1$ **then** ▷ If the current event is a *completion of a process*
    $\mathcal{DB}\leftarrow\{j:\mathbf{C}_{i,j}=1\}$ ▷ Down stream elements of element i
    if $|\mathcal{DB}|=0$ **then** $\eta_i\leftarrow\mathrm{S}$ ▷ The part goes to the sink
    else
      $\mathcal{AB}\leftarrow\mathcal{DB}\cap\{k:\eta_k<B_k\}\cap\{k:u_{k,\mathrm{m}}=1\}$ ▷ Identifying the available buffers
      if $|\mathcal{AB}|>0$ **then**
        $\eta_j\leftarrow\eta_j+1,\eta_i\leftarrow\mathrm{S}:j\in\mathcal{AB}$ ▷ The part goes to one of the available downstream buffers
      else
        $\eta_i\leftarrow\mathrm{B}$ ▷ Blocking
      end if
    end if
  else
    $FEL\leftarrow FEL\cup\{(T+\Delta_{i,1,c_i},i,\Delta_{i,2,c_i+1},\Delta_{i,3,c_i})\}$; $c_i\leftarrow c_i+1$
  end if
  $z\leftarrow1$
  while $z=1$ **do**
    $z\leftarrow0$
    call Algorithm 2 ▷ Blocking and starvation resolution
  end while
end while

Algorithm 2 Blocking and starvation resolution sub-module

$z\leftarrow1$ ▷ Blocking and starvation resolution
while $z=1$ **do**
  $z\leftarrow0$
  for $i\in\{1,\ldots,n\}$ **do**
    if i is a machine **then**
      if $\eta_i=\mathrm{S}\vee\eta_i=\mathrm{B}$ **then**
        Identify the marking m based on $\mathbf{I}$, $\mathbf{Y}$
        if $\eta_i=\mathrm{S}\wedge u_{i,\mathrm{marking}}=1$ **then**
          if $\exists j:\mathbf{C}_{j,i}=1$ **then** ▷ The machine feeds from the source
            $FEL\leftarrow FEL\cup\{(T+\Delta_{i,1,c_i},i,\Delta_{i,2,c_i+1},\Delta_{i,3,c_i})\}$; $c_i\leftarrow c_i+1$; $z\leftarrow1$
          else
            $\mathcal{UB}\leftarrow\{j:\mathbf{C}_{j,i}=1\}$; $\mathcal{AB}\leftarrow\{j:\eta_j\geq0,j\in\mathcal{UB}\}$
            if $|\mathcal{AB}|\geq NIS_i$ **then**
              $\eta_j\leftarrow\eta_j-1:j\in\mathcal{CB}\subset\mathcal{AB},|\mathcal{CB}|=NIS_i$; $\eta_i\leftarrow D_{i,2,c_i}$
              $FEL\leftarrow FEL\cup\{(T+\Delta_{i,1,c_i},i,\Delta_{i,2,c_i+1},\Delta_{i,3,c_i})\}$; $c_i\leftarrow c_i+1$; $z\leftarrow1$
            end if
          end if
        end if
      end if
      if $\eta_i=\mathrm{B}$ **then**
        if $\exists j:\mathbf{C}_{i,j}=1$ **then**
          $\eta_i\leftarrow\mathrm{S}$; $z\leftarrow1$ ▷ The part goes to the sink
        else
          $\mathcal{DB}\leftarrow\{j:\mathbf{C}_{i,j}=1\}$
          Identify the marking m based on $\mathbf{I}$, $\mathbf{Y}$
          $\mathcal{AB}\leftarrow\mathcal{DB}\cap\{k:\eta_k<B_k\}\cap\{k:u_{k,\mathrm{m}}=1\}$ ▷ Identifying the available buffers
          if $|\mathcal{AB}|\geq1$ **then** $\eta_j=\eta_j+1:j\in\mathcal{CB}\subset\mathcal{AB},|\mathcal{CB}|=1$; $\eta_i\leftarrow\mathrm{S}$; $z\leftarrow1$
          else $\eta_i\leftarrow\mathrm{B}$
          end if
        end if
      end if
    end if
  end for
end while

- P_3: Finding the best control parameters for given markings that are defined based on the selected information sources:

$$\bar{\Pi}^3\left(\mathbf{I},\mathbf{Y},\bar{\mathbf{\Lambda}}\right)=\min_{\mathbf{U}}\frac{\sum_{r=1}^{N}\bar{\Pi}\left(\mathbf{I},\mathbf{Y},\mathbf{U},\bar{\mathbf{\Lambda}},D^r\right)}{N}. \tag{3}$$

- P_2: Forming the best markings by clustering the partial information signals for the selected information sources:

$$\bar{\Pi}^2\left(\mathbf{I},\bar{\mathbf{\Lambda}}\right)=\min_{\mathbf{Y}}\bar{\Pi}^3\left(\mathbf{I},\mathbf{Y},\bar{\mathbf{\Lambda}}\right). \tag{4}$$

- P_1: Selecting the best information sources:

$$\bar{\Pi}^1\left(\bar{\mathbf{\Lambda}}\right)=\min_{\mathbf{I}}\bar{\Pi}^2\left(\mathbf{I},\bar{\mathbf{\Lambda}}\right). \tag{5}$$

The number of feasible solutions for these problems can be very different and the computational burden of solving the original problem can be unevenly distributed among them. The problem of selecting the best sources of information, P_1, can have approximately 2^n feasible solutions. The problem of finding the best formation for the markings by putting the partial information tuples into K clusters, P_2, has approximately $\left\{ {|\{\hat{\eta}\}| \atop K} \right\}$ feasible solutions, where $|\{\hat{\eta}\}| \cong \prod_{i=1}^{n}\left((1-I_i)+|A_i|\,I_i\right)$. Finally, the problem of finding the best policy parameters $\mathbf{u}_1,\ldots,\mathbf{u}_K$ for the marking-dependent policy, P_3, has approximately $\left(2^{\sum\chi_i}\right)^K$ feasible solutions.

For the first experimental set up given in Section 9.1, P_2 has 2×10^{50} feasible solutions. For the second experimental setup given in Section 9.2, P_1 has 286, P_2 has 5×10^{29}, and P_3 has $(2^3)^3=512$ solutions respectively.

8. Solution of the Optimization Problems with the Regression Models

In the following part, we discuss how the two higher levels of the problem of finding the best marking-dependent policy for a given number of information sources and markings (P_1, P_2) can be solved by using the regression methods. The lower-level problem, P_3 can be treated similarly. However, it is not discussed in this paper.

8.1. Information Source Selection Problem: $\mathbf{P}_1$

For a given selection of the information sources $\mathbf{I}$, evaluating $\bar{\Pi}^2\left(\mathbf{I},\bar{\Lambda}\right)$ is computationally expensive for all the problems irrespective of the solution method. A regression model allows determining the order of $\mathbf{I}$ tuples to be evaluated in a way that improves the objective function value that can be achieved in a given time period.

Due to the large number of possible $\mathbf{I}$ solutions for a given system that is 2^n for a system with n elements, it may not be possible to enumerate all these solutions or evaluate them using a fast learned model or a fast inaccurate simulation. For these reasons, we consider searching for the best solutions in a subset of the possible solutions. We refer the subset of solutions that can be

evaluated in a specified time period as the candidate solutions. The specified time period can be viewed as a computational budget.

Let $\mathcal{I}_C = \{\mathbf{I}^1, \mathbf{I}^2, \ldots, \mathbf{I}^{z_\mathbf{I}}\}$ denote the set of candidate $\mathbf{I}$ vectors. These candidate solutions can be obtained by using ordinal optimization (Ho et al. 1992), incorporating expert opinion, or alternatively generating $z_\mathbf{I}$ random tuples.

Due to computational constraints, the number of performance measure evaluations that are used to select the best information source tuple will be less than or equal to $z_\mathbf{I}$. Let us assume that at most σ performance measure evaluations can be completed with the specified computational budget using the DES algorithm given in Section 6.

Let $\mathbf{I}^t$ denote the tth information source selection that has been evaluated where $t < z$. The input to the regression problem is expressed as

$$X = \begin{bmatrix} \mathbf{I}^1 \\ \vdots \\ \mathbf{I}^t \end{bmatrix}, Y = \begin{bmatrix} \bar{\Pi}^2\left(\mathbf{I}^1, \bar{\mathbf{\Lambda}}\right) \\ \vdots \\ \bar{\Pi}^2\left(\mathbf{I}^t, \bar{\mathbf{\Lambda}}\right) \end{bmatrix}. \tag{6}$$

For the two-station production line example, this regression problem is given as

$$X = \begin{bmatrix} 0\ 1\ 1 \\ \vdots \\ 1\ 0\ 1 \end{bmatrix}, Y = \begin{bmatrix} \bar{\Pi}^2\left(\left[0\ 1\ 1\right], \{\bar{\mu}_1, \omega_2, \bar{\mu}_2\}\right) \\ \vdots \\ \bar{\Pi}^2\left(\left[1\ 0\ 1\right], \{\bar{\mu}_1, \omega_2, \bar{\mu}_2\}\right) \end{bmatrix} \tag{7}$$

where the estimated long-run costs $\bar{\Pi}^2\left(\left[0\ 1\ 1\right], \bar{\mu}_1, \omega_2, \bar{\mu}_2\}\right), \ldots, \bar{\Pi}^2\left(\left[1\ 0\ 1\right], \bar{\mu}_1, \omega_2, \bar{\mu}_2\}\right)$ are obtained by the DES algorithm for different selection of information sources.

Let $f_\mathbf{I}(\mathbf{I})$ denote the regression model built based on X and Y. Then, at a given iteration, the next information sources to be evaluated is selected based on the regression model that has been built by using all the evaluated information sources up to that iteration, i.e., $f_\mathbf{I}(\mathbf{I}^{t+j}) \leq f_\mathbf{I}(\mathbf{I}^{t+j+1}), \sigma - t - 1 \geq j \geq 1$. The process of learning $f_\mathbf{I}$ and selecting $\mathbf{I}^{t+j}$ using $f_\mathbf{I}$ can be repeated as more $\mathbf{I}$ tuples are evaluated and $f_\mathbf{I}$ becomes more accurate as a result. This process is referred as *learning while optimizing* and depicted in Figure 3.

This procedure orders the elements in $\mathcal{I}_C$ such that $f_\mathbf{I}(\mathbf{I}^{(1)}) < f_\mathbf{I}(\mathbf{I}^{(2)}) < \cdots < f_\mathbf{I}(\mathbf{I}^{(\sigma)})$ where $\mathbf{I}^{(k)}$ is the information source selection tuple that yields the kth lowest predicted average cost $f_\mathbf{I}(\mathbf{I}^{(k)})$.

Once the candidate $\mathbf{I}$ tuples are ordered according to the average cost prediction obtained by machine learning methods, the best information source that yields the minimum average cost can be obtained by evaluating the σ average costs by using the DES algorithm given in Section 6. As a result, the solution of the selection of information sources problem with σ objective function evaluations using the DES algorithm is given as

$$\mathbf{I}^*(\sigma) = \arg\min_{k \leq \sigma} \bar{\Pi}^2\left(\mathbf{I}^{(k)}, \bar{\mathbf{\Lambda}}\right), \tag{8}$$

$$\bar{\Pi}^{2*}(\sigma) = \bar{\Pi}^{2}\left(\mathbf{I}^{*}(\sigma), \bar{\mathbf{\Lambda}}\right). \tag{9}$$

We denote the best average cost that can be obtained among all the candidate solutions as $\bar{\Pi}^{2*} = \bar{\Pi}^{2*}(z_{\mathbf{I}})$.

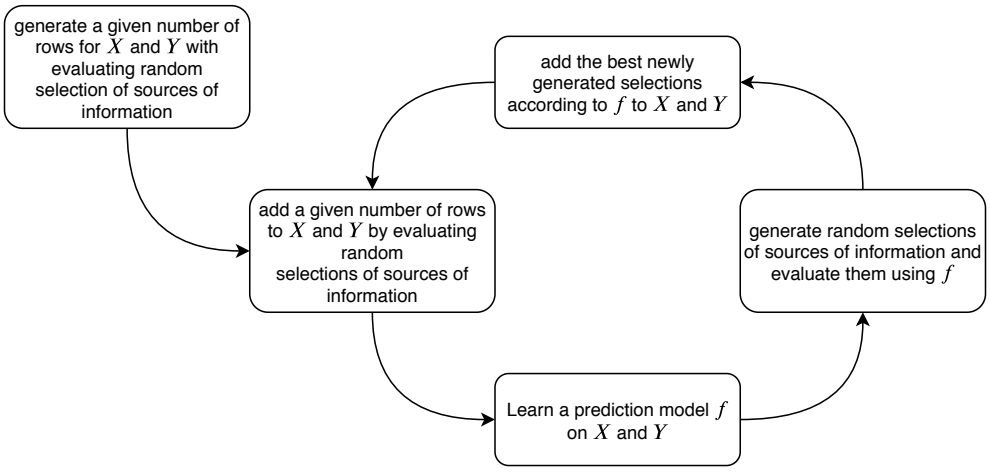

Figure 3 Flowchart for learning while optimizing

8.2. Marking Formation Problem: $\mathbf{P}_2$

The same procedure discussed for using regression for selection of sources of information can be applied to the formation of markings. The elements of $\mathbf{Y}$ that indicate a specific formation of markings are not good candidates for being used as features in a machine learning setting. This is due to the fact that for a given formation of markings, the number of representations increases exponentially with the number of markings. That is, there are $K!$ different $\mathbf{Y}$ representations for K markings. It is possible to add all the equivalent representations to the training data. However, this increases the number of data points by a factor of $K!$. In order to represent different marking clusters in the learning algorithm, we use the variable $w_{i,j} = \sum_{k=1}^{K} y_{i,k} y_{j,k}$ that is equal to one if marking i and j are in the same cluster and is equal to zero otherwise. This representation avoids symmetric formations, i.e., identical marking formations with different $\mathbf{Y}$ values. Let

$$\mathbf{W} = \left[\left[w_{1,2}, \ldots, w_{1,|\hat{\eta}|}\right], \ldots, \left[w_{2,3}, \ldots, w_{2,|\hat{\eta}|}\right], \ldots, \left[w_{i,i+1}, \ldots, w_{i,|\hat{\eta}|}\right], \ldots, \left[w_{|\hat{\eta}|-1,|\hat{\eta}|}\right]\right] \tag{10}$$

denote the tuple of $w_{i,j}$ values. $\mathbf{W}$ gives a unique representation of each formation of markings.

For the two-station production line example, the markings were formed with clustering all the states that have less than 2 part in the buffer, $\{(-, 0, \mathrm{S}), (-, 0, \mathrm{W}_1), (-, 1, \mathrm{W}_1)\}$ into the first marking and the state that has 2 parts in the buffer, $\{(-, 2, \mathrm{W}_1)\}$ into the second marking as described in Subsection 4.3. For these markings, their unique representation $\mathbf{W}$ is given as

$$\mathbf{W} = \left[\left[1,\ 1,\ 0\right], \left[1,\ 0\right], \left[0\right]\right].$$

Figure 4 depicts the relation between $\mathbf{Y}$ that represents the assignment of different states into the marking and its unique representation $\mathbf{W}$ for this example.

Since the numbering of markings $1,\ldots,K$ is arbitrary, for each $\mathbf{W}$ representation, there are $K!$ equivalent representations with identical grouping of partial information signals but different numbering for the markings. In other words, using the unique representation $\mathbf{W}$ decreases the size of the training set by a factor of $K!$.

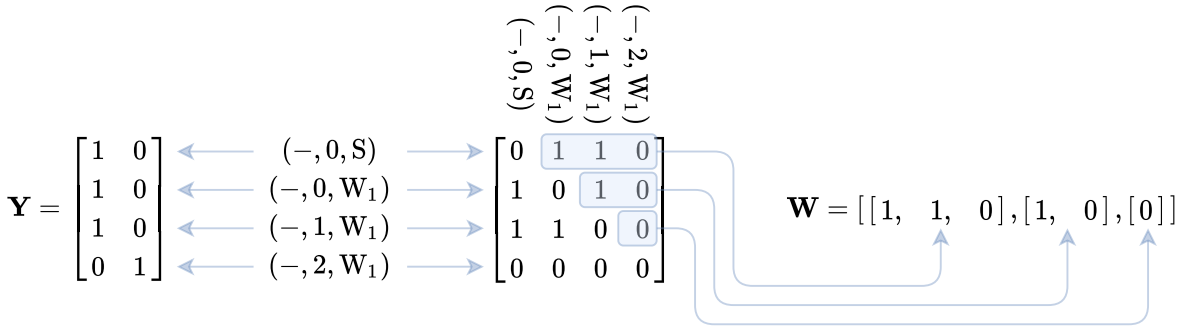

Figure 4 The relation between Y and W for the two-station system example

Although we do not consider choosing K here, this representation is not dependent on K and can allow for learning from formations based on different K values. Hence, we use $\bar{\Pi}^1(\mathbf{I},\mathbf{W})$ and $\bar{\Pi}^1(\mathbf{I},\mathbf{Y})$ interchangeably.

Let σ be the maximum number of performance measure evaluations that can be completed in a given computational time limit. Let $\mathcal{W}_C=\{\mathbf{W}^1,\mathbf{W}^2,\ldots,\mathbf{W}^{z\mathbf{W}}\}$ denote the set of candidate $\mathbf{W}$ tuples. The candidate solutions can be obtained by using the approaches discussed for the information source selection problem.

For a given selection of the sources of information $\mathbf{I}$, the problem of finding the next formation of markings to evaluate is expressed using the data specified as

$$X=\begin{bmatrix}\mathbf{W}^1\\ \vdots\\ \mathbf{W}^t\end{bmatrix},Y=\begin{bmatrix}\bar{\Pi}^1\left(\mathbf{I},\mathbf{W}^1,\bar{\mathbf{\Lambda}}\right)\\ \vdots\\ \bar{\Pi}^1\left(\mathbf{I},\mathbf{W}^t,\bar{\mathbf{\Lambda}}\right)\end{bmatrix} \tag{11}$$

where $\mathbf{W}^t$ denotes the tth formation of markings that has been evaluated.

Let $f_{\mathbf{W}}(\mathbf{W})$ denote the regression model built based on X and Y. Then, at a given iteration, the next marking formation to be evaluated is selected based on the regression model that has been built by using all the evaluated information sources up to that iteration, i.e., $f_{\mathbf{W}}(\mathbf{W}^{t+j})\leq f_{\mathbf{W}}(\mathbf{W}^{t+j+1}),\sigma-t-1\geq j\geq 1$. The process of learning $f_{\mathbf{W}}$ and selecting $\mathbf{W}^{t+j}$ using $f_{\mathbf{W}}$ can be repeated as more $\mathbf{W}$ tuples are evaluated and $f_{\mathbf{W}}$ becomes more accurate as a result. This procedure

orders the elements in $\mathcal{W}_C$ such that $f_{\mathbf{W}}(\mathbf{W}^{(1)}) < f_{\mathbf{W}}(\mathbf{W}^{(2)}) < \cdots < f_{\mathbf{W}}(\mathbf{W}^{(\sigma)})$ where $\mathbf{W}^{(k)}$ is the marking formation that yields the kth lowest predicted average cost.

The best marking formation that yields the minimum average cost among the candidates is obtained by comparing σ average costs by using the DES algorithm given in Section 6. As a result, the solution of the marking formation problem is given as

$$\mathbf{W}^*(\sigma) = \arg\min_{k \leq \sigma} \bar{\Pi}^3 \left(\mathbf{I}, \mathbf{W}^{(k)}, \bar{\mathbf{\Lambda}}\right), \tag{12}$$

$$\bar{\Pi}^{3*}(\sigma) = \bar{\Pi}^3 \left(\mathbf{I}, \mathbf{W}^*(\sigma), \bar{\mathbf{\Lambda}}\right). \tag{13}$$

The best average cost that can be obtained among all the candidate solutions is referred as $\bar{\Pi}^{3*} = \bar{\Pi}^{3*}(z_{\mathbf{W}})$.

8.3. Incorporating the Data Available about Different Parameter Sets

If the data about production systems that are similar to the production system of interest is available, the existing data can be used to improve the computational efficiency. In many cases, it is possible to obtain data about similar systems. First, it is common for the parameters of a production system to change over time due to replacement of machines due to maintenance or a change in the technology. Second, before enough data about the system can be gathered, the estimates about the parameters in the system change by new observations. The simulations performed during this period belong to systems with parameters close to the actual system. Therefore, the collected data can be informative about the new system. Third, a shared optimization resource can be used by different facilities with similar structure. It would be beneficial for the optimization resource to be able to pool the effort dedicated to similar facilities. In the numerical results, we consider a subset of these scenarios that is related to identical production system structures with similar parameters.

For incorporating the data and the simulation evaluations available from similar systems, we use the following notation. Let $\mathcal{D}^{r,l}$ denote the rth data set available related to the lth set of system parameters, referred to as the lth *instance* of the system. The aim is to find the best marking-dependent policy for a target instance of the system specified by l^* using the available datasets $\{\mathcal{D}^{r,l} : 1 \leq r \leq N^l, 1 \leq l \leq L\}$. Additionally, let $\bar{\mathbf{\Lambda}}^l$ denote the estimates of the parameters of the system for instance l. Let $\mathbb{I}^l$ and $\mathbb{W}^l$ denote the set of $\mathbf{I}$ and $\mathbf{W}$ tuples respectively that have been evaluated based on $\{\mathcal{D}^{r,l} : 1 \leq r \leq N^l\}$.

Then the regression problem for selection of sources of information can be expressed as

$$X = \begin{bmatrix} X^1 \\ \vdots \\ X^L \end{bmatrix}, Y = \begin{bmatrix} Y^1 \\ \vdots \\ Y^L \end{bmatrix}, \tag{14}$$

where

$$X^l = \begin{bmatrix} \bar{\mathbf{\Lambda}}^l & \mathbf{I}^1 \\ \vdots & \vdots \\ \bar{\mathbf{\Lambda}}^l & \mathbf{I}^t \end{bmatrix}, Y^l = \begin{bmatrix} \bar{\Pi}^2\left(\mathbf{I}^1, \bar{\mathbf{\Lambda}}^l\right) \\ \vdots \\ \bar{\Pi}^2\left(\mathbf{I}^t, \bar{\mathbf{\Lambda}}^l\right) \end{bmatrix}, \mathbf{I}^t \in \mathbb{I}^l. \tag{15}$$

Similarly, the regression problem for formation of markings (for given sources of information specified by I) can be expressed using Equation (14) where

$$X^l = \begin{bmatrix} \bar{\mathbf{\Lambda}}^l & \mathbf{W}^1 \\ \vdots & \vdots \\ \bar{\mathbf{\Lambda}}^l & \mathbf{W}^t \end{bmatrix}, Y^l = \begin{bmatrix} \bar{\Pi}^1\left(\mathbf{I}, \mathbf{W}^1, \bar{\mathbf{\Lambda}}^l\right) \\ \vdots \\ \bar{\Pi}^1\left(\mathbf{I}, \mathbf{W}^t, \bar{\mathbf{\Lambda}}^l\right) \end{bmatrix}, \mathbf{W}^t \in \mathbb{W}^l. \tag{16}$$

The regression models learned on this data can be used for evaluating candidate **I** and **W** tuples for every $l \in \{1, \ldots, L\}$. The candidates ordered by using the regression models can them be used for optimizing the information source selection and marking formation problems following the procedure described in Sections 8.1 and 8.2.

8.4. Solution of the Regression Problem by Using Machine Learning Approaches

Once the data for regression is formed, different machine learning algorithms can be used to solve the regression problem. In the numerical experiments, we compared neural networks, random forest, linear regression, Gaussian Process Regression, and genetic programming methods.

Initially inspired by the nervous system of animals, artificial **neural networks** (ANN) are one of the most studied and commonly used machine learning tools (Gurney 2014). They generate prediction functions that are composed of nested highly non-linear functions. Neural Networks are capable in estimating highly non-linear functions.

Random forest (RF) is an ensemble learning method (Breiman 2001). Random forest uses a number of regression trees where each tree is trained with a subset of the features. Random forests generate prediction functions that are composed of nested if statements. Random forest algorithm has been successful in many applications; however, they have a high memory requirement.

Linear regression (LR) fits a linear model to the data by minimizing the least square error over the training set. Linear regression can fit nonlinear functions through using nonlinear transformations of the features. However, given that there are many different ways for such transformations, the transformation has to be done on the basis of the specific application.

Gaussian Process Regression (GPR) is a non-parametric kernel-based probabilistic method that works well with noisy training data (Rasmussen and Williams 2006). GPR assumes the data to belong to a Gaussian process. Any finite set of points selected from a Gaussian process follows a Gaussian distribution. The functional form of the prediction models generated by GPR is hard to interpret. Training a GPR model involves inversion of matrices whose size is dependent on the number of data points. Hence, they are computationally costly for large data-sets.

Genetic programming is a method that generates prediction models that are a nested combination of multiplication and summation functions. Genetic programming uses methods very similar to genetic algorithm for minimizing the error on the training dataset (Koza and Koza 1992). Genetic programming uses tree structures for storing the prediction functions. The computational burden for genetic programming increases significantly as the number of features increases. This is particularly relevant to the problem of forming the markings, since the number of features has a quadratic relation with the number of possible partial information observations.

9. Numerical Experiments

In this section, we give two sets of numerical experiments to demonstrate how the best marking-dependent control policy can be implemented by selecting the information sources, forming markings, and determining the marking-dependent production policy by using the machine learning approach described in Section 7. We also compare different machine learning methods in performing this task.

The first set of numerical experiments focus on the problem of forming the markings and the second set on choosing the sources of information. In the first set of numerical experiments, we consider a given selection of sources of information for the production/inventory system. In this specific case, the problem of setting the policy parameters is easy to solve. Hence, we focus on the problem of forming the markings that has approximately 2×10^{50} solutions.

In the second set of numerical experiments, we aim at testing the performance of machine learning methods in learning from only one instance of the system where evaluation of the system is computationally expensive. Accordingly, in this experiment, we consider the problem of selecting 3 information sources from 13 elements in the system that has 286 solutions. Evaluating each possible solution for this system takes 3.6 hours on a personal computer.

In the numerical experiments, we compare the performance of the learning methods by using the properties of their convergence graphs (Beiranvand et al. 2017). A convergence graph shows the evolution of the best solution reached by an algorithm against the number of objective function evaluations used for reaching this solution. Using the convergence graph, another performance measure for each method is defined as the number of function evaluations needed to reach a solution close to the best known solution.

9.1. Production/Inventory System

9.1.1. Experimental Setting Description In the first set of numerical experiments, we consider a production/inventory system. By using this system, we show how the marking-dependent policy works and how the markings that are used to control release of material can be formed by using machine learning.

Table 3 Parameters of the production/inventory system

	Type	Information			
		Size	Processing time	Breakdown Probability	Repair time
E_1	Machine	-	$U[0, \mu_1]$	p_1	$U[0, r_1]$
E_2	Buffer	5	-	-	-
E_3	Machine	-	$U[0, \mu_3]$	-	-
E_4	Buffer	∞	-	-	-
E_5	Machine	-	$U[0, \mu_5]$	-	-
E_6	Buffer	Determined based on control parameters	-	-	-

The system we consider is a three-station production/inventory system as depicted in Figure 5. In this system, the release of material into machine E_5 is controlled to balance the accumulation of material in buffers E_4 and E_6. We first show how this system can be controlled by using the marking-dependent policy and then describe how a marking-dependent threshold policy can be expressed as a marking-dependent policy.

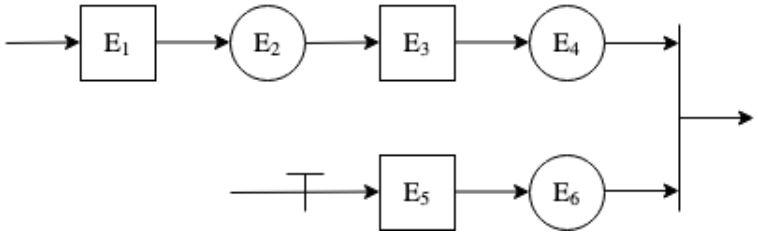

Figure 5 Production/inventory system

Table 3 gives the details about the system parameters. For different instances of the system, the processing time parameters, μ_1, r_1, μ_3, μ_5 have been drawn randomly from $[0.55, 0.85]$ and the breakdown probability p_1 is drawn randomly from $[0.055, 0.085]$. In this set of experiments, the information source is predetermined as the unreliable upstream machine (E_1) and its buffer (E_2) and the machine that is controlled to match the flows (E_5), i.e., $\mathbf{I} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$. Therefore, the problem of selecting the sources of information (P_1) is not addressed. The problem of forming the markings (P_2) is solved by using machine learning. Since the problem of setting the policy parameters has only two possible solutions, the release of material into machine E_5 is allowed or not, the problem of setting the policy parameters (P_3) has been solved by exhaustive search.

9.1.2. Marking-Dependent Policy For this system, the state of the system is expressed as

$$\eta = (\eta_1, \eta_2, \eta_3, \eta_4, \eta_5, \eta_6),$$

where, $\eta_1 \in \{W_1, W_2, B, S\}$, W_1 denotes the working state, W_2 denotes the down state of the station, $\eta_2 \in \{0, \ldots, \omega_2\}$, $\eta_3 \in \{W_1, B, S\}$, $\eta_4 \in \{0, \ldots, \omega_4\}$ where $\omega_4 \to \infty$, $\eta_5 \in \{W_1, B, S\}$, and $\eta_6 \in \{0, \ldots, \tau\}$, τ is the maximum inventory level allowed by the controller.

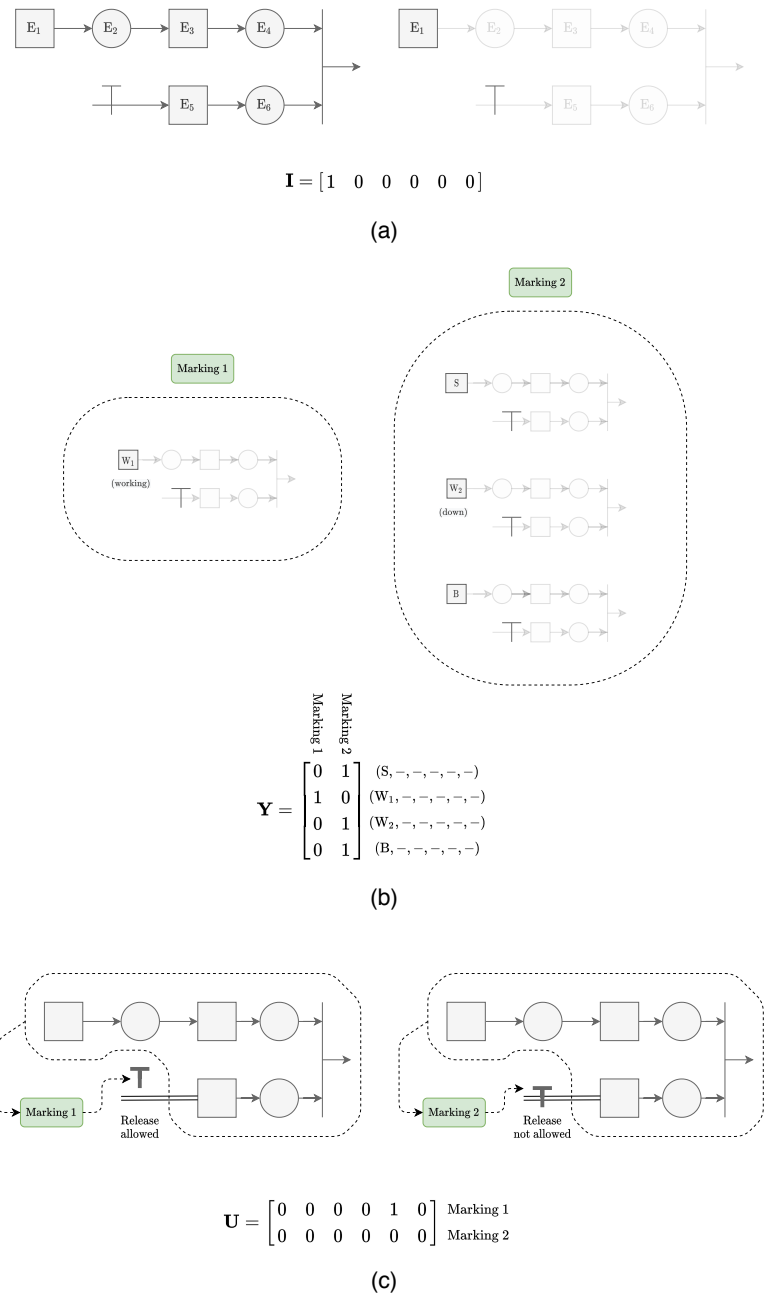

Figure 6 **The three levels of the problem: (a) selection of sources of information (b) forming the marking (c) setting the parameters of the marking-dependent policy.**

Based on this definition, we will now discuss how different production control policies can be implemented with this unified representation:

- In this system, if a threshold policy is to be implemented, the control decision can be made based on only the state of buffer E_6 that is controlled according to the threshold. Accordingly, the information selection matrix $\mathbf{I}$ is set as $\mathbf{I} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$.
- If a multi-threshold policy that takes the status of the unreliable machine into account is to

be implemented, then $\mathbf{I} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$.

• If a policy that also takes into account the status of the intermediate buffer is to be implemented, then $\mathbf{I} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$ and $\hat{\eta} = (\eta_1, \eta_2, -, -, -, \eta_6)$.

• A base-stock policy with base-stock level τ would require two markings ($K = 2$) with the formation of markings specified as

$$y_{i,j} = \begin{cases} 1 & \text{if the } i\text{th } \hat{\eta} \text{ tuple satisfies} \\ & \left(-, -, -, -, -, (-1)^{j+1}\eta_6 \leq (-1)^{j+1}\tau + \left(\frac{j-1}{2}\right)\right). \\ 0 & \text{o.w.} \end{cases} \tag{17}$$

• A multi-threshold policy that uses the threshold τ_{Down} when the machine is down and the threshold τ_{Up} when the machine is up can be implemented by using two markings ($K = 2$). The markings are formed according to

$$y_{i,j} = \begin{cases} & \text{if the } i\text{th } \hat{\eta} \text{ tuple satisfies} \\ 1, & \left(\eta_1 = \text{W}_2, -, -, -, -, (-1)^{j+1}\eta_6 \leq (-1)^{j+1}\tau_{\text{Down}} + \left(\frac{j-1}{2}\right)\right). \\ & \vee \left(\eta_1 = \text{W}_2, -, -, -, -, (-1)^{j+1}\eta_6 \leq (-1)^{j+1}\tau_{\text{Up}} + \left(\frac{j-1}{2}\right)\right) \\ 0, & \text{otherwise.} \end{cases} \tag{18}$$

The policies discussed here can be implemented by using the control parameters set as

$$\mathbf{U} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{19}$$

9.1.3. Machine Learning Setup In this experimental setup, we compared linear regression (LR), artificial neural networks (ANN), random forest (RF), Gaussian Process Regression (GPR) and genetic programming (GP) approaches to select the best markings.

For this problem, the response variable for all the machine learning methods is the average cost of the system. For each set of system parameters, we exclude the formations of markings with a cost five times larger than optimal as easy to discard solutions.

We use 5 features that are related to the system parameters. These system parameters include three machine rates, one breakdown probability and one repair time. The rest of the features describe the relation of the 198 $\hat{\eta}$ tuples with each other. Equation (10) describes how the formation of markings can be expressed as the relation between various $\hat{\eta}$ pairs. This results in using $198 \times (198-1)/2 = 19503$ additional features for the relation among the observed system states. As a result, 19508 features are used for the problem of forming the best markings.

From each of the system instances used for training, we pick 200 data points at random resulting in $200 \times 19 = 3800$ data points. The proportion of features to data points, that is 19503 vs 3800, makes the problem of forming the markings a *big data problem* (Zhong et al. 2016).

Neural Network Setup. We use a neural network with 3 hidden nodes for ANN. The number of hidden nodes is determined using inner cross-validation. Given the large number of features, we

choose the 5 system parameter features along with 195 randomly chosen among 19503 features as the inputs for the ANN. The response variable is the average cost of the system.

Random Forest Setup. For the random forest algorithm (RF), We use 3 trees to be trained based on all the available features. The number of tress is based on inner cross-validation. Random forest is internally able to search for the more relevant features among the large number of available features. As a result, all of the 19508 features are used as the inputs and the average cost is used as the output.

Linear Regression Setup. In linear regression, we do not use any transformations of the original features. Hence, the method produces a linear model based on all of the 19508 available features and the average cost is used as the output. Linear regression does not include any hyper parameters that need to be set using cross validation.

Gaussian Process Regression Setup. For GPR, we allow the choice of the basis functions and the kernel functions to be chosen from among a set of basis functions and kernel functions. GPR uses the kernel functions for forming the predictions. The kernel functions are used for building kernel matrices and the size of these matrices are determined by the number of data points and are independent of the number of features. This allows for a reduction in the computational effort caused by the large number of features for this problem.

Genetic Algorithm Setup. For genetic programming, we allow at most 40 nodes. The number of nodes is determined based on inner cross-validation. Similar to random forest, genetic programming internally searches for the most relevant features. Accordingly, all of the 19508 features are used as the inputs and the average cost is used as the output.

9.1.4. Results In these experiments, we generated 20 instances of the production/inventory system with random parameter sets for formation of markings. From these instances, each time, 19 were used for learning and 1 was used for testing. We focus on the average deviation from the best known cost as the main performance measure to compare the performance of different algorithms. Due to the size of the problem with approximately 2×10^{50} solutions, the optimal cost cannot be determined. As an alternative, the best known cost, denoted by $\bar{\Pi}^{2*}$ is calculated by searching a thousand randomly generated $\mathbf{Y}$ matrices.

Table 4 gives the accuracy of these methods in terms of predicting the value of the objective function. The Mean Absolute Percentage Error (MAPE) for one instance is defined as

$$\text{MAPE} = \frac{1}{1000} \sum_{i=1}^{1000} 100 \frac{|\bar{\Pi}^3(\mathbf{I}, \mathbf{W}^i, \bar{\mathbf{\Lambda}}) - \bar{\Pi}^{3*}|}{\bar{\Pi}^{3*}}$$

and the Root Mean Square Error (RMSE) for one instance is defined as

$$\text{RMSE} = \sqrt{\frac{1}{1000} \sum_{i=1}^{1000} (\bar{\Pi}^3(\mathbf{I}, \mathbf{W}^i, \bar{\mathbf{\Lambda}}) - \bar{\Pi}^{3*})^2}.$$

In Table 4, we report the average MAPE and average RMSE values that are the average MAPE and RMSE values over the 20 instances respectively. These results show than GPR yields more accurate predictions for the average cost.

In order to compare the performance of the different methods in terms of solving the marking formation problem, we focus on their ability to suggest the evaluation of better solutions earlier. We start with a randomly generated set of solutions $\mathcal{W}_C = \{\mathbf{W}^1, \mathbf{W}^2, \ldots, \mathbf{W}^{1000}\}$. For each method, and each instance, we define the convergence graph of each algorithm after σ performance evaluations using DES (Beiranvand et al. 2017) as

$$\epsilon_\sigma = \frac{100 \times \left(\min\left\{\bar{\Pi}^3\left(\mathbf{I}, \mathbf{W}^{(j_1)}, \bar{\mathbf{\Lambda}}\right), \bar{\Pi}^3\left(\mathbf{I}, \mathbf{W}^{(j_2)}, \bar{\mathbf{\Lambda}}\right), \ldots, \bar{\Pi}^3\left(\mathbf{I}, \mathbf{W}^{(j_\sigma)}, \bar{\mathbf{\Lambda}}\right)\right\} - \bar{\Pi}^{3*}\right)}{\bar{\Pi}^{3*}},$$

where $f_{\mathbf{W}}(\mathbf{W}^{(j_1)}) < f_{\mathbf{W}}(\mathbf{W}^{(j_2)}) < \cdots < f_{\mathbf{W}}(\mathbf{W}^{(j_\sigma)})$ and $\bar{\Pi}^{3*} = \min_{\mathbf{W} \in \mathcal{W}_C} \bar{\Pi}^3\left(\mathbf{I}, \mathbf{W}, \bar{\mathbf{\Lambda}}\right)$. We use $\bar{\epsilon}_\sigma$ as the average of the convergence graphs over the 20 instances. We take the area under the average of convergence graphs as the main performance measure for the methods.

Table 5 and Figure 7 summarize the results of these experiments. Figure 7 depicts the comparison between evaluating formations of makings randomly and evaluating them in an order suggested by different learning methods that have been trained on other instances of the system with different parameters. These results were obtained after eliminating formations that have a cost two times as much as the minimum cost in order to expedite the solution process.

The area under each curve in Figure 7 is the main performance measure reported in Table 5. The results indicate that using all the methods can be beneficial for this problem. Specifically, Gaussian regression process outperforms other methods. GPR is the fastest to reach a mean percentage error (MPE) of 1% from $\bar{\Pi}^{3*}$.

In these experiments, Genetic Programming performs worse in comparison to other methods. The main reason for the poor performance of GP is the very large number of features for this problem. Since GP's computational requirement is very sensitive to the tree sizes and the optimal tree sizes in turn relate to the number of relevant features, GP fails to find good models in a reasonable time for the problem of forming the best markings.

These results indicate that, in the cases where the evaluation of the system is costly, machine learning can reduce the cost of the system by decreasing the computational burden of the optimization.

9.2. Dispatching Problem

9.2.1. Experimental Setting Description A dispatching problem can be expressed as deciding on releasing material from a common source buffer into different routes. This can be done based on different characteristics of each route, e.g., the traffic present in each route. Such characteristics can be expressed in the markings.

Table 4 **The accuracy of the methods in predicting the average cost for the marking formation problem**

	Average MAPE	Average RMSE
Linear regression	74.99	3.74
Genetic programming	59.02	3.43
Artificial neural network	22.85	1.13
Random forest	16.71	1.63
Gaussian process regression	13.10	0.50

Table 5 **Performance on the test data related to marking formation problem for the production/inventory system (average performance over 20 instances of the system)**

	Area under the average of convergence graphs	Number of function evaluations to reach 5% MPE for $\bar{\Pi}^{3*}$	Number of function evaluations to reach 1% MPE for $\bar{\Pi}^{3*}$
Random formation of markings	289.28	7	76
Genetic programming	237.13	6	50
Artificial neural network	154.03	5	48
Linear regression	98.89	2	11
Random forest	86.80	4	23
Gaussian process regression	39.55	3	5

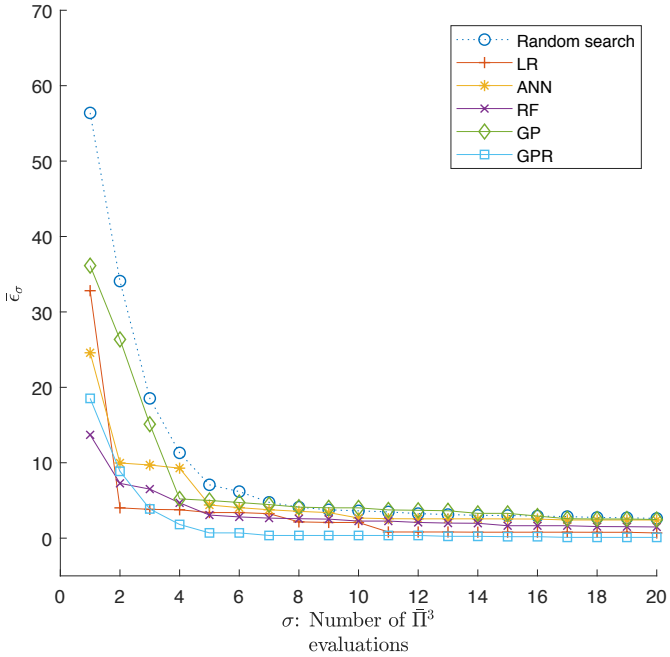

Figure 7 **Search for the best formation of markings for the production/inventory system based on learned models versus a random search (averaged over 20 different instances of the system)**

Figure 8 depicts the structure of the system we use. For this problem, we investigate finding the best selection of sources of information. The formation of markings has been performed through a random search, and we use a predefined solution for the policy parameters. For this system, the buffer sizes are set to 2 for all the buffers. E_1 is a reliable machine with processing rate 1

and all the other machines are unreliable machines with processing rate 1, breakdown probability 0.1 and repair rate 0.1. The processing times and the repair times are exponentially distributed random variables. We consider the problem of minimizing the cycle time in the system subject to a minimum throughput level set as 0.6 for this case.

The marking formation problem for this setup can have up to $\left\{ {4^3 \atop 3} \right\} \sim 5 \times 10^{29}$ feasible solutions. Since the marking formation problem is discussed in the previous experimental setup, in this setup, we focus on the information source selection problem and form the markings by using a random search.

In this system, although choosing sources of information is not trivial, the whole set of possible choices can be evaluated for assessing the performance of learned models. Namely, the problem of selecting 3 sources of information from the 13 elements present has $\frac{13!}{10!3!} = 286$ solutions. Evaluating these solutions on a personal computer takes 44 days. A computer cluster that runs up to 40 jobs in parallel was used to evaluate all of these 286 solutions in close to 24 hours. Since forming the best marking is time consuming for each choice of the sources of information, ordering the candidate solutions by using machine learning yields good results earlier.

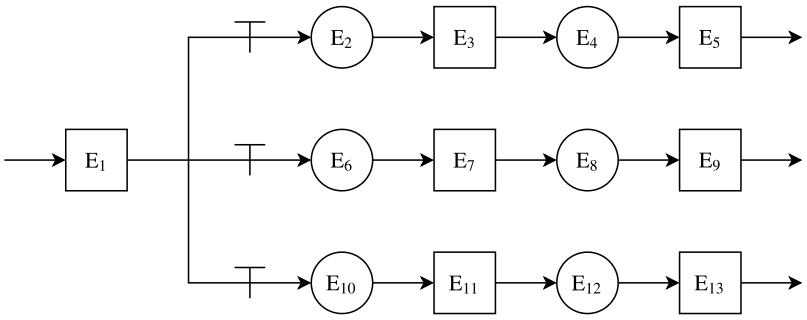

Figure 8 **The system considered for the information source selection problem**

9.2.2. Marking-Dependent Policy For this system, the state of the system is expressed as

$$\eta = (\eta_1, \eta_2, \eta_3, \eta_4, \eta_5, \eta_6, \eta_7, \eta_8, \eta_9, \eta_{10}, \eta_{11}, \eta_{12}, \eta_{13}),$$

where $\eta_1 \in \{\mathrm{W}_1, \mathrm{B}, \mathrm{S}\}$, $\eta_2, \eta_4, \eta_6, \eta_8, \eta_{10}, \eta_{12} \in \{0, 1, 2\}$ and $\eta_3, \eta_5, \eta_7, \eta_9, \eta_{11}, \eta_{13} \in \{\mathrm{W}_1, \mathrm{W}_2, \mathrm{B}, \mathrm{S}\}$.

In order to demonstrate the implementation of a marking-dependent control policy, consider a simple dispatching policy that dispatches material to the route where its processing is expected to start earlier. This policy needs to check all the buffers at the starting point of each route; hence the information selection tuple is set as $\mathbf{I} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$ and $\hat{\eta} = (-, \eta_2, -, -, -, \eta_6, -, -, -, \eta_{10}, -, -, -)$.

This policy would require all the $\hat{\eta}$ tuples where buffer j has the least number of items in it to be assigned to the same marking. That is, $K = 3$. This can be achieved by using the following marking formation:

$$y_{i,j} = \begin{cases} 1 & \text{if the } i\text{th } \hat{\eta} \text{ tuple satisfies} \\ & \eta_{2+4(j-1)} \leq \eta_k : k \in \{2, 6, 10\} \setminus \{2 + 4\,(j-1)\} \\ 0 & \text{otherwise.} \end{cases} . \tag{20}$$

Then, for each marking, the route to the buffer with the lowest inventory will be open and the rest of the routes will be barred as given by the following control parameters:

$$\mathbf{U} = \begin{bmatrix} 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0 \\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0 \\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0 \end{bmatrix} . \tag{21}$$

9.2.3. Machine Learning setup For the dispatching problem, we consider one instance of the system. In order to assess the improvements made possible by using machine learning in a more accurate way, every possible solution for the problem of selection of information sources has been evaluated. Since there is one instance of the system, the parameters of the system do not appear as features for machine learning. The features relate to a specific source being selected or not. Hence, there are only 13 features corresponding to 13 elements in the system for each data point.

The specifics of the machine learning algorithms used for this set of experiments is identical to that of Section 9.1.3. The only exception is using all of the 13 features in all the methods including in ANN and GP.

9.2.4. Results In the second set of numerical experiments, we have considered the system depicted in Figure 8. We have evaluated this system with one set of parameters. This system has been evaluated with different choices of sources of information.

For a given selection of sources of information, the formation of markings has been done using a random search and for a given formation of markings, the policy parameters have been set as given in Equation (21) due to the symmetry in of the system.

For the purpose of reaching accurate estimations, traces of length 10000 have been used for the evaluations of the system. Hence, for each selection of sources of information, on average 3.6 hours have been spent and 286 selections of sources of information have been evaluated in a high performance cluster. Then, for assessing the performance of the machine learning methods, 20 out of 286 solutions have been selected randomly for training and the remaining 266 solutions have been used for testing the methods. This experiment has been repeated for 100 times. Table 7 reports the accuracy of the predictions obtained by using different Machine Learning algorithms. The results show that all the algorithms yield accurate average cost predictions except Genetic Programming. Furthermore, GPR and Random Forest give more accurate results compared to the other methods. In the following, we show how *learning while optimizing* is effective for this case.

Table 6 Optimal assignment of $\hat{\eta}$ tuples to different markings

Marking		
1	2	3
$(-,-,-,-,-,-,-,-,S,-,B,2,-,)$	$(-,-,-,-,-,-,-,-,S,-,S,1,-,)$	$(-,-,-,-,-,-,-,-,S,-,S,0,-,)$
$(-,-,-,-,-,-,-,-,S,-,\mathrm{W}_1,1,-,)$	$(-,-,-,-,-,-,-,-,S,-,S,2,-,)$	$(-,-,-,-,-,-,-,-,S,-,\mathrm{W}_1,2,-,)$
$(-,-,-,-,-,-,-,-,S,-,\mathrm{W}_2,2,-,)$	$(-,-,-,-,-,-,-,-,S,-,\mathrm{W}_1,0,-,)$	$(-,-,-,-,-,-,-,-,\mathrm{W}_1,-,\mathrm{W}_1,0,-,)$
$(-,-,-,-,-,-,-,-,\mathrm{W}_1,-,S,0,-,)$	$(-,-,-,-,-,-,-,-,S,-,\mathrm{W}_2,0,-,)$	$(-,-,-,-,-,-,-,-,\mathrm{W}_2,-,S,0,-,)$
$(-,-,-,-,-,-,-,-,\mathrm{W}_1,-,S,1,-,)$	$(-,-,-,-,-,-,-,-,S,-,\mathrm{W}_2,1,-,)$	$(-,-,-,-,-,-,-,-,\mathrm{W}_2,-,\mathrm{W}_2,0,-,)$
$(-,-,-,-,-,-,-,-,\mathrm{W}_1,-,S,2,-,)$	$(-,-,-,-,-,-,-,-,\mathrm{W}_1,-,\mathrm{W}_2,1,-,)$	
$(-,-,-,-,-,-,-,-,\mathrm{W}_1,-,B,2,-,)$	$(-,-,-,-,-,-,-,-,\mathrm{W}_2,-,\mathrm{W}_2,2,-,)$	
$(-,-,-,-,-,-,-,-,\mathrm{W}_1,-,\mathrm{W}_1,1,-,)$		
$(-,-,-,-,-,-,-,-,\mathrm{W}_1,-,\mathrm{W}_1,2,-,)$		
$(-,-,-,-,-,-,-,-,\mathrm{W}_1,-,\mathrm{W}_2,0,-,)$		
$(-,-,-,-,-,-,-,-,\mathrm{W}_1,-,\mathrm{W}_2,2,-,)$		
$(-,-,-,-,-,-,-,-,\mathrm{W}_2,-,S,1,-,)$		
$(-,-,-,-,-,-,-,-,\mathrm{W}_2,-,S,2,-,)$		
$(-,-,-,-,-,-,-,-,\mathrm{W}_2,-,B,2,-,)$		
$(-,-,-,-,-,-,-,-,\mathrm{W}_2,-,\mathrm{W}_1,0,-,)$		
$(-,-,-,-,-,-,-,-,\mathrm{W}_2,-,\mathrm{W}_1,1,-,)$		
$(-,-,-,-,-,-,-,-,\mathrm{W}_2,-,\mathrm{W}_1,2,-,)$		
$(-,-,-,-,-,-,-,-,\mathrm{W}_2,-,\mathrm{W}_2,1,-,)$		

For this instance of the system, the optimal selection of information sources is selecting the first machine and the second buffer in the third route and selecting the second buffer in the second route, i.e., $\begin{bmatrix}0&0&0&0&0&0&0&0&1&0&1&1&0\end{bmatrix}$. The optimal formation of markings is given in Table 6, where element state 1 denotes a working machine and element state 2 denotes a machine under repair. Sources of information about two of the routes in the system have been chosen. Although calculating the average observable WIP level requires the steady state probability of observing each partial information signal, just averaging the WIP levels in observable parts of each route for each marking can give some information about the solution. However, deciphering the formation of markings for this case is not straightforward. This might be due to the fact that the random search for the best formation of markings might not have reached the optimal solution and contains some noise that makes interpreting the solution difficult.

Additionally, learning while optimizing has been compared to a blind search of the solution space with employing GPR, Artificial neural network, Linear regression and random forest after every 10 evaluations of the solutions.

Figure 9 and Table 8 show how these methods perform in terms of optimizing the objective function as the number of evaluations increase at each iteration. For this problem, 8 random function evaluations is enough to reach a 5% MPE for $\bar{\Pi}^{2*}$. Since the learning is started after 10 function evaluations, we do not report the number of iterations needed to reach 5% MPE for $\bar{\Pi}^{2*}$ in Table 8. The starting portion for the different approaches are similar because the machine learning methods are not used in choosing the solutions until enough data is available. However, with relatively limited data, GPR is able to suggest evaluating considerably better solutions.

Table 7 Performance on the test data related to selecting sources of information for the dispatching problem.

	Average MAPE	Average RMSE
Genetic programming	90.60	9.46
Linear regression	3.38	0.46
Artificial neural network	3.29	0.42
Random forest	2.73	0.35
GPR	2.17	0.27

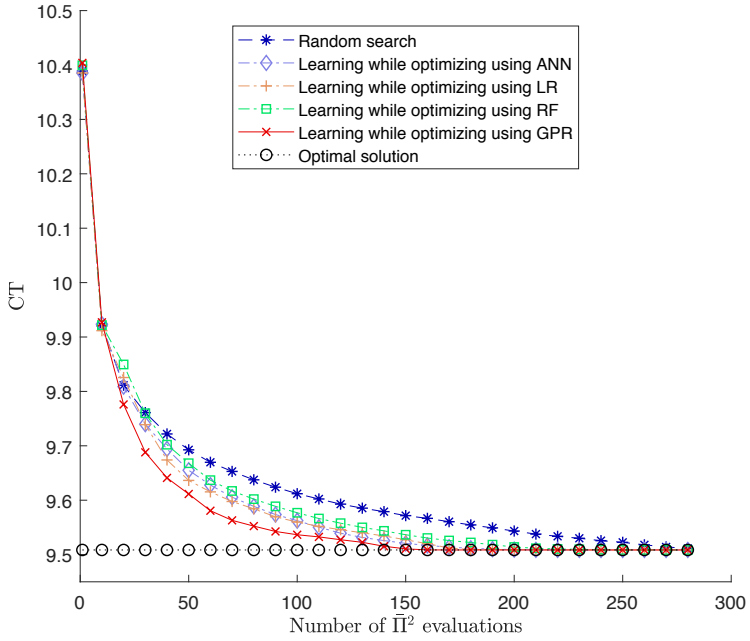

Figure 9 Comparison of Random Search and Different Machine Learning Algorithms that Use Learning While Optimizing for Selecting the Information Sources for the Dispatching Problem

Table 8 Performance of the learning while optimizing approach for the dispatching problem.

	Area under the average of convergence graphs	Number of function evaluations to reach 1% MPE for $\bar{\Pi}^{2*}$
Random formation of markings	324.22	109
Random forest	259.01	79
Artificial neural network	225.02	72
Linear regression	221.28	66
Gaussian process regression	177.78	52

10. Conclusions

In this paper, we consider the problem of implementing effective data-driven production control policies. Using extensive information sources and signals complicates the implementation of data-driven production control policies with limited marginal benefit.

By focusing on a specific data-driven production control policy, referred as the marking-

dependent production control policy, the problem is broken down to three levels, namely, choosing the right information sources, forming the right clusters of partial information signals, referred to as the markings, and determining the parameters of the optimal marking-dependent production control policy. We show how the first two problems, i.e., selecting the information sources and forming the markings to implement the marking-dependent production policy, can be solved using regression models generated by using different machine learning algorithms.

Using two experimental setups, we show how machine learning can facilitate implementation of production control in manufacturing systems. The first setup is a release control problem in a production/inventory system. The second setup is a dispatching problem. We compare the performance of Artificial Neural Networks, Genetic Programming, Linear Regression, Random Forest, and Gaussian Process Regression methods on the data-sets related to forming markings for a marking-dependent policy and the data set for selection of sources of information. Our results show that all the methods prove to be beneficial for both problems. Additionally, our results show the comparative advantage of random forest and Gaussian process regression algorithms for these problems.

We show that even if there are no evaluations of the system with parameters other than the latest available estimates of the system parameters, machine learning methods can be beneficial with very few data points through the procedure of learning while optimizing presented in this study.

This work can be extended in different directions. First, although the marking formation problem can be solved as it is by different machine learning tools, it can benefit from using different distance measures that can be defined between partial observation tuples of the state of the system. This approach would provide more information about the structure of the system in training. For example, for a subset of the information sources that are connected, a distance measure can be defined as the absolute difference in work-in-process inventory in those elements. Then, the clustering of the partial information tuples can be done based on a distance function that is a mixture of these distance functions. In training, the coefficients used for building this mixture can be optimized such that the resulting formations of markings on the training data have minimal cost. Hence, a machine learning tool can be developed that is not blind to the specific structure of the regression problem that is a result of markings being generated from partial observations of the system.

Next, the synthetic data generated by different parameter sets for the system can be viewed as multiple machine learning tasks. This allows incorporating multi-task learning methods into the optimization. Multi-task learning has been developed to improve the prediction models for tasks with few data points by receiving assistance from tasks with many data points. Additionally, multi-task learning can be used to transfer the domain knowledge between production systems

that have similar but not identical structures. Hence, both problems discussed here can be viewed as multi-task machine learning problems.

There are aspects of production systems that can be incorporated to the methods given in this work to analyze a wider range of production systems. These include modeling production systems with multiple products and batching processes. Multi-product systems naturally generate more data as the status of each buffer in these systems is a multi-dimensional vector. Hence, these systems can benefit greatly from methods that can identify suitable reductions of the information signals.

Finally, obtaining the training data can be greatly improved by combining simulation with the already-available data about the system. The available traces can be expanded by using methods such as bootstrap sampling. However, preserving the autocorrelation structure of the datasets requires some modifications to this method. Developing methods to combine the available data, simulation runs, and other approximations effectively allows increasing the training data sets that can be obtained in a given time period. These are left for future research.

As a summary, this work shows that an efficient and effective data-driven production control policy can be implemented in a complex production system by selecting the right information sources, the right markings obtained from these sources, and using the right production policy that utilizes these signals effectively. The computational burden of this large-scale optimization problem can be addressed by using the machine learning framework presented in this study and appropriate algorithms.

Acknowledgements

Research leading to these results has received funding from the EU ECSEL Joint Undertaking under grant agreement no. 737459 (project Productive4.0) and from TUBITAK (217M145).

References

Aissani, N., Beldjilali, B., and Trentesaux, D. (2008). Use of machine learning for continuous improvement of the real time heterarchical manufacturing control system performances. *International Journal of Industrial and Systems Engineering*, 3(4):474–497.

Alenezi, A., Moses, S. A., and Trafalis, T. B. (2008). Real-time prediction of order flowtimes using support vector regression. *Computers & Operations Research*, 35(11):3489–3503.

Backus, P., Janakiram, M., Mowzoon, S., Runger, C., and Bhargava, A. (2006). Factory cycle-time prediction with a data-mining approach. *IEEE Transactions on Semiconductor Manufacturing*, 19(2):252–258.

Beiranvand, V., Hare, W., and Lucet, Y. (2017). Best practices for comparing optimization algorithms. *Optimization and Engineering*, 18(4):815–848.

Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

Cadavid, J. P. U., Lamouri, S., Grabot, B., Pellerin, R., and Fortin, A. (2020). Machine learning applied in production planning and control: A state-of-the-art in the era of Industry 4.0. *Journal of Intelligent Manufacturing*, pages 1–28.

Can, B. and Heavey, C. (2012). A comparison of genetic programming and artificial neural networks in metamodeling of discrete-event simulation models. *Computers & Operations Research*, 39(2):424–436.

Charest, M., Finn, R., and Dubay, R. (2018). Integration of artificial intelligence in an injection molding process for on-line process parameter adjustment. In *2018 Annual IEEE International Systems Conference (SysCon)*, pages 1–6. IEEE.

Dulac-Arnold, G., Mankowitz, D., and Hester, T. (2019). Challenges of real-world reinforcement learning. *arXiv preprint arXiv:1904.12901*.

Farooqui, A., Bengtsson, K., Falkman, P., and Fabian, M. (2020). Towards data-driven approaches in manufacturing: an architecture to collect sequences of operations. *International Journal of Production Research*, pages 1–17.

Gurney, K. (2014). *An introduction to neural networks*. CRC press.

Ho, Y.-C., Sreenivas, R., and Vakili, P. (1992). Ordinal optimization of DEDS. *Discrete Event Dynamic Systems*, 2(1):61–88.

Hwang, I. and Jang, Y. J. (2020). Q (λ) learning-based dynamic route guidance algorithm for overhead hoist transport systems in semiconductor fabs. *International Journal of Production Research*, 58(4):1199–1221.

Irani, K. B., Cheng, J., Fayyad, U. M., and Qian, Z. (1993). Applying machine learning to semiconductor manufacturing. *IEEE Expert*, 8(1):41–47.

Karaoglan, A. D. and Karademir, O. (2017). Flow time and product cost estimation by using an artificial neural network (ANN): A case study for transformer orders. *The Engineering Economist*, 62(3):272–292.

Khayyati, S. and Tan, B. (2020). Data-driven control of a production system by using marking-dependent threshold policy. *International Journal of Production Economics*, 226:107607.

Koza, J. R. and Koza, J. R. (1992). *Genetic programming: on the programming of computers by means of natural selection*, volume 1. MIT press.

Kusiak, A. (2017). Smart manufacturing must embrace Big Data. *Nature*, 544(7648):23–25.

Li, X. and Olafsson, S. (2005). Discovering dispatching rules using data mining. *Journal of Scheduling*, 8(6):515–527.

Lingitz, L., Gallina, V., Ansari, F., Gyulai, D., Pfeiffer, A., and Monostori, L. (2018). Lead time prediction using machine learning algorithms: A case study by a semiconductor manufacturer. *PROCEDIA CIRP*, 72:1051–1056.

Moeuf, A., Pellerin, R., Lamouri, S., Tamayo-Giraldo, S., and Barbaray, R. (2018). The industrial management of SMEs in the era of industry 4.0. *International Journal of Production Research*, 56(3):1118–1136.

Monostori, L., Márkus, A., Van Brussel, H., and Westkämpfer, E. (1996). Machine learning approaches to manufacturing. *CIRP Annals*, 45(2):675–712.

Monostori, L., Váncza, J., and Kumara, S. R. (2006). Agent-based systems for manufacturing. *CIRP Annals*, 55(2):697–720.

Paraschos, P. D., Koulinas, G. K., and Koulouriotis, D. E. (2020). Reinforcement learning for combined production-maintenance and quality control of a manufacturing system with deterioration failures. *Journal of Manufacturing Systems*, 56:470–483.

Priore, P., De La Fuente, D., Gomez, A., and Puente, J. (2001). A review of machine learning in dynamic scheduling of flexible manufacturing systems. *Ai Edam*, 15(3):251–263.

Rasmussen, C. E. and Williams, C. K. (2006). *Gaussian Processes for Machine Learning*. the MIT Press, Cambridge, MA.

Tao, F., Qi, Q., Liu, A., and Kusiak, A. (2018). Data-driven smart manufacturing. *Journal of Manufacturing Systems*, 48:157–169.

Zhang, J., Arinez, J. F., Chang, Q., Gao, R. X., and Xu, C. (2020). Artificial intelligence in advanced manufacturing. *Journal of Manufacturing Science and Engineering*, pages 1–53.

Zhong, R. Y., Newman, S. T., Huang, G. Q., and Lan, S. (2016). Big data for supply chain management in the service and manufacturing sectors: Challenges, opportunities, and future perspectives. *Computers & Industrial Engineering*, 101:572–591.