Data-driven control of a production system by using marking-dependent threshold policy

by

Siamak Khayyati

A Dissertation Submitted to the
Graduate School of Sciences and Engineering
in Partial Fulfillment of the Requirements for
the Degree of

Doctor of Philosophy

in

Industrial Engineering and Operations Management

January, 2020

Data-driven control of a production system by using marking-dependent threshold policy

Koç University

Graduate School of Sciences and Engineering

This is to certify that I have examined this copy of a doctoral dissertation by

Siamak Khayyati

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Committee Members:

_____

Prof. Dr. Barış Tan, (Advisor)

_____

Prof. Dr. Fikri Karaesmen

_____

Prof. Dr. Mine Çağlar

_____

Prof. Dr. Ahmet Barış Balcıoğlu

_____

Assoc. Prof. Dr. Aybek Korugan

Date: _____

To my dear parents and my brother Arash

ABSTRACT

With shop-floor data becoming more available, production systems can be controlled more effectively by using data-driven control methods. This thesis focuses on the following research question: how can the decision to produce or not to produce at any time be given depending on the real-time information about a production system?; how can the collected data be used directly in optimizing the policy parameters?; what is the effect of using different information sources on the performance of the system? and how can this choice be made? In order to answer these questions, a production/inventory system that consists of a production stage that produces to stock to meet random demand is considered. The system is not fully observable but partial production and demand information, referred to as markings is available. We propose using the marking-dependent threshold policy to decide whether to produce or not based on the observed markings in addition to the inventory and production status at any given time. An analytical method that uses a matrix geometric approach is developed to analyze a production system controlled with the marking-dependent threshold policy when the production, demand, and information arrivals are modeled as Marked Markovian Arrival Processes. A mixed integer programming formulation is presented to determine the optimal thresholds. Then a mathematical programming formulation that uses the real-time shop-floor data for joint simulation and optimization (JSO) of the system is presented. Using numerical experiments, we compare the performance of the JSO approach to the analytical solutions. The results indicate that the marking-dependent policy introduced here is able to make use of the available

partial information effectively and the data-driven joint simulation and optimization is an efficient way for setting the parameters of the policy. Next, we investigate how machine learning methods can be employed to facilitate the optimization of systems controlled with the marking-dependent policies and compare the performance of a number of well-known machine learning methods on this task. Through numerical experiments we show that using machine learning and active learning can improve the performance of complex production systems by speeding up time-consuming optimization problems. Finally, we implement the concepts of this thesis using a Lego production system and discuss the usage of Lego production systems in educational and research related to data-driven control.

ÖZETÇE

Üretim alanı verileri daha erişilebilir olduğunda, veriye dayalı kontrol yöntemleri kullanılarak üretim sistemleri daha etkin bir şekilde kontrol edilebilir. Bu tezde şu sorulara odaklanılmıştır: bir üretim sisteminde gerçek zamanlı bilgilere istinaden, üretme veya üretmeme kararı nasıl verilebilir? Elimizeki veriler doğrudan politika parametrelerini optimize etmekte nasıl kullanılabilir? Farklı bilgi kaynakları kullanmanın sistem performansı üzerindeki etkisi nedir? Bu seçim nasıl yapılır? Bu soruları cevaplamak için, rasgele talebi karşılamak üzere stoğa üretım yapan bir üretim / envanter sistemi ele alınmıştır. Bu sistem tam olarak gözlemlenmemekte ancak işaret (marking) olarak adlandırılan kısmi üretim ve talep bilgileri kullanılabilmektedir. Herhangi bir zamanda envanter ve üretim durumuna ek olarak, gözlemlenen işaretlere dayanarak üretime karar vermek için işarete bağlı eşik politikasını kullanmayı öneriyoruz. Üretim, talep ve bilgi varışları işaretli Markov varış süreci olarak modellendiğinde, işarete bağımlı eşik politikası ile kontrol edilen bir üretim sistemini analiz etmek için matris geometrik bir yöntem kullanan analitik bir yöntem geliştirilmiştir. Optimal eşikleri belirlemek için karışık tamsayılı programlama formülasyonu sunulmaktadır. Sonrasında bütünleşik benzetim ve eniyileme (BBE) için gerçek zamanlı alan verilerini kullanan bir matematiksel programlama formülasyonu sunulmuştur. Sayısal deneyler kullanarak BBE yönteminin performansı analitik çözümlerle karşılaştırılmıştır. Sonuçlar burada sunulan işarete bağlı politikanın mevcut kısmi bilgileri etkin bir şekilde kullanabildiğini ve veriye dayalı bütünleşik benzetim ve eniyilemenin, kontrol parametrelerini ayarlamak için etkili bir yöntem olduğunu göstermektedir. Daha sonra

işarete bağlı politikalarla kontrol edilen sistemlerin optimizasyonunu kolaylaştırmak ve bir dizi yapay öğrenme yönteminin performansını karşılaştırmak için yapay öğrenme yöntemlerinin nasıl kullanılabileceği araştırılmıştır. Sayısal deneyler sayesinde yapay öğrenmenin ve aktif öğrenmenin kullanılmasının, zaman alıcı optimizasyon problemlerini hızlandırarak karmaşık üretim sistemlerinin performansını artırabileceği gösterilmiştir. Son olarak, bu tezde sunulan yaklaşımlar bir Lego üretim sistemine uygulanmış ve Lego üretim sistemlerinin veri odaklı kontrol ile ilgili eğitim ve araştırmalarda nasıl kullanılabileceği tartışılmıştır.

ACKNOWLEDGMENTS

Firstly, I would like to express my deep gratitude to my advisor Prof. Dr. Barış Tan for all his help and support during my Ph.D study. He has guided me through many obstacles in my research and helped me improve in many aspects of scientific research. I'm grateful for all the things that I have learned from him during my study.

I would like to specially thank Prof. Dr. Fikri Karaesmen, for his valuable feedbacks and comments during the various progress report sessions of my thesis. I also would like to thank the rest of my thesis committee: Prof. Dr. Mine Çağlar, Prof. Dr. Ahmet Barış Balcıoğlu and Assoc. Prof. Dr. Aybek Korugan, for their insightful comments and encouragement and their constructive suggestions.

I express my deepest gratitude to my friend and colleague Arezou Rahimi. She has always supported me during my time in Koç University. I also would like to thank my colleagues, Masoud Shahmanzari, Niloofar Mehrnia and Davood Pirayesh, for their friendship and precious supports. I thank Adar Bayan and Özgün Ozan Nacitarhan for their efforts in completing the set-up for the Lego lab.

Research leading to the results in this thesis has received funding from the EU-ECSEL Joint Undertaking under grant agreement no. 737459 (projectProductive4.0) and from TUBITAK (217M145) and I have recieved financial support during my study from Koç University.

Last but not least, I would like to thank my family: my parents and my brother Arash, for supporting me spiritually throughout writing this thesis and my life in general.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

MAP	Markov Arrival Process
MMAP	Marked Markov Arrival Process
JSO	Joint Simulation and Optimization
CTMC	Continous Time Markov Chain
MILP	Mixed Integer Linear Programming
QBD	Quasi Birth and Death process

Chapter 1

INTRODUCTION

Increasing usage of industrial data collection hardware and software has allowed manufacturers to collect extensive shop-floor data. Developing data-driven production control methods that utilize this abundant collected data in an effective way can bring significant benefits. This thesis is motivated by the need of developing implementable data-driven control policies that use the selected information to match supply and demand in an effective way. We aim at answering the following research questions: *how can the decision to produce or not to produce at any time be given depending on the real-time information about a production/inventory system?; how can the collected data be used directly in optimizing the policy parameters?; what is the effect of using different information sources on the performance of the system?; and how can the best information sources be selected among the alternatives?*

In order to investigate these research questions, we first consider a production/inventory system that consists of a production stage that produces to stock to meet random demand. The system is not fully observable but partial production and demand information, referred to as the *markings* arrive together with demand or production, or separately. The production is controlled with the marking-dependent threshold policy where the decision to produce or not to produce is given depending on the observed markings in addition to the inventory and production status at any given time. The marking-dependent threshold policy is an easily implementable policy: if the production stage is idle and the inventory position is less than the threshold determined

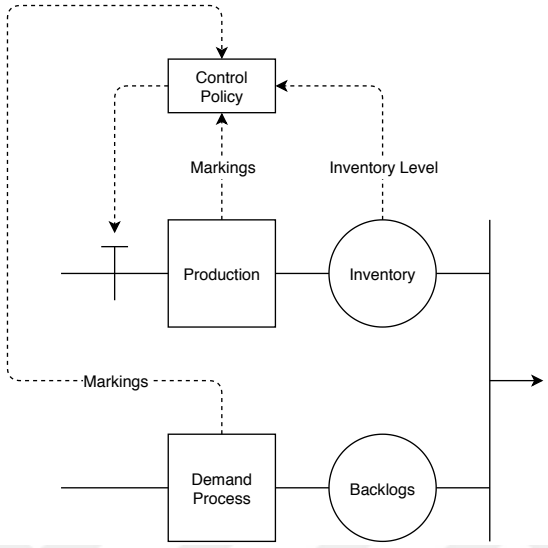

Figure 1.1: A production/inventory system controlled with the marking-dependent threshold policy

for the last observed values of the markings, production of a new item is triggered with the release of the material into the production stage. Figure 1.1 illustrates this production/inventory system.

We first present an analytical method that uses a matrix analytical approach to analyze a production/inventory system controlled with the marking-dependent threshold policy where the demand and information arrivals and the production times, are modeled as Marked Markovian Arrival Processes (MMAP). The MMAP framework allows us to model correlated arrivals and general inter-arrival time distributions (Qi-Ming and Neuts, 1998). A mixed integer programming (MIP) formulation is developed to determine the optimal thresholds for the marking-dependent threshold policy depending on the markings selected to control the system.

We then propose using the data-driven joint simulation and optimization (JSO) approach as an online control method for manufacturing systems. Joint simulation and optimization refers to mathematical programming problems that are able to gen-

erate a simulation run of a given system (Schruben, 2000). We develop the JSO approach by formulating a mixed integer program to determine the parameters of the threshold policy by using the shop-floor data together with the existing statistical information about the processes. The data-driven JSO uses the available data directly for optimization and does not impose any distributional assumptions for the processes and does not require independence of the inter-arrival times.

The analytical method presented here allows us to evaluate the performance of the data-driven approach under different information scenarios by comparing the results obtained based on different-length traces with the analytical results. An experimental setup for the release control of a feeder line where the demand for the feeder is generated by an unreliable production line with two stations separated by a finite buffer is used. The release into the feeder work station is controlled to minimize the expected holding and backlog costs. The effects of using different markings related to the availability of information on the machine status and the buffer status are investigated. This system is analyzed by using the joint simulation and optimization approach with different markings and with the different-length inter-departure and processing time data collected from the shop-floor and the available statistical information. The results are compared with the analytical results.

Next, we examine how the best information sources to control a production system can be selected. We give the machine learning problems arising from this approach for a more general setting. When the structure of a system is more complex compared to the available analytical models that usually focus on one aspect of a system, few optimization methods can be used for improving the system. These methods are for the most part variations and extensions of simulation optimization. These methods are time-costly, and when the control policy uses the information gathered from the whole system, the dependency between the different parts of the system increases, decreasing the accuracy of decomposition methods in performance evaluation and

optimization.

If the computational capacity were not limited, from a pure optimization point of view, the best way to deal with these systems would be considering each possible observation of the system as a marking and using exhaustive search for finding the best parameter set for this large set of markings. Following this approach with limited computational capacity will be far from optimal since the set of possible observations of a real system is often extremely large. To overcome these difficulties, machine learning tools can be employed. Machine learning has been used for control of specific processes in production systems, performance evaluation of the production systems and solving specific shop-floor control problems (Li and Olafsson, 2005; Karaoglan and Karademir, 2017; Can and Heavey, 2012). However, to our knowledge, it has not been used for the process of selection of information sources and summarizing information signals emitted from partially observable production systems.

We define the problem of optimizing a manufacturing system controlled by the marking-dependent policy in three levels, each of which can benefit from using machine learning. The optimization problem can be stated as choosing which elements of the system to observe, how to form the markings and what to do when each marking is received to minimize the cost of the system. This problem can be broken down to the following three steps, choosing the sources of information to monitor, assigning the possible observations of the system to different markings based on this choice, and finding the best policy parameters for each marking. Hence, the problem is transformed into a multi-level problem where the two higher-level problems can be solved faster by using machine learning. More specifically we consider the usage of regression problems for this purpose. This is done by making the machine learning tools guess the objective function of each level based on the points that have been already evaluated in addition to evaluations of the system with parameters close to that of the actual system if such evaluations are available.

To show the effectiveness of this approach, two experimental settings are used. For assessing the performance of the machine learning methods for the problem of formation of markings, the production/inventory system given in Section 2.2 is used and for selection of information sources an experimental set-up is used where the main question is dispatching the products to three similar routes. These experiments indicate that machine learning can be beneficial in identifying a small number of markings that are sufficient for effective control of the system.

Lastly, we implement the methods developed in this thesis in a physical production system model built by using Lego. In addition to educational purposes, Lego production systems have been used for studies that investigate methods to control production systems based on direct use of the shop-floor data (Lugaresi et al., 2019c; Travaglini, 2018). A physical system allows for collection of data that includes the details and dependencies in the production systems that arise from the specifics of a real system and are often simplified in the process of modeling and optimization. These systems also allow for feedback loops between the actual system and the digital representation of it, including the simulation modules and the optimization models based on simulation.

We use the physical replication of the experimental setup described in Section 2.2 as the test-bed for the methods presented here. In this Lego production system, the processing times are randomly generated but the data collected from the setting includes the time for the additional operations that are necessary for the set-up to work e.g. the time spent by the parts on the conveyors, the time the system spends on communication with the computer that controls the set-up and the time for the gates to function. For this reason, the true distribution for the inter-event times in the system is unknown and the true cost of the policy parameters suggested by different methods can only be assessed by employing those policy parameters in the system. In summary, the Lego production system has been used as a proof of concept for

data-driven control methods discussed here.

There are three main contributions of this thesis. The first one is modeling and analysis of a production/inventory system where the production, demand, and information processes are modeled as Marked Markovian Arrival Processes and controlled with a multiple-threshold policy. With the multiple-threshold policy, production is allowed at a given time when the current inventory position is below the target threshold level corresponding to the latest observed markings. The optimal thresholds for this system are determined by using a mathematical programming formulation that uses the embedded matrix geometric solution of the quasi birth and death process. To our knowledge there is no work in the literature to determine the optimal thresholds in a stochastic model of a controlled queue by using a mathematical programming formulation. The second one is developing a joint simulation and optimization method to determine the control parameters based on the shop-floor data collected from the system. Although there are studies in the literature that use JSO to design production systems, this study proposes using JSO as an online control method for the first time. The JSO literature does not include formulations for a system controlled with a policy that uses different thresholds at different times. The mixed integer program presented in this study extends the range of discrete event models that can be simulated and optimized by using the JSO approach. Our analytical and numerical results show that the marking-dependent control policy together with a JSO approach that determines the policy parameters works effectively as a data-driven control method for manufacturing. Next, it defines the problems of selecting the information sources and forming the markings as sub-levels of the problem of finding the best marking-dependent policy, and shows how regression methods can solve these problems. Lastly, it discusses the implementation of these methods using a Lego production system.

The remainder of this thesis is organized as follows. The production/inventory system and an example system are given in Chapter 2. Chapter 3 gives the analysis

Table 1.1: A summary of the structure of the thesis

	Model	Main problems	Main methods
Chapter 2	Partially observable production/inventory system with correlated data	Finding optimal thresholds for a marking-dependent threshold policy	MMAP modeling and matrix analytic algorithm
Chapter 3	Partially observable production/inventory system with correlated data	Finding optimal thresholds for a marking-dependent threshold policy	Data-driven joint simulation and optimization
Chapter 4	General partially observable production system	Selecting information sources and forming the markings	Supervised machine learning
Chapter 5	Partially observable production/inventory system with correlated data	Implementing the marking-dependent threshold policy	Implementation of physical system using Lego

and the solution method for controlling a partially observable production/inventory system with correlated data. Chapter 4 discusses the data-driven control method for this system. Chapter 5 identifies two machine learning problems that can increase the efficiency of optimization tools used for optimizing production systems controlled with a marking-dependent policy. Chapter 6 describes the implementation of the aforementioned methods in a Lego production system. The literature review related to each subject is given in its corresponding chapter. Finally Chapter 7 concludes the thesis. Table 1.1 gives an overview of this thesis.

Chapter 2

PRODUCTION/INVENTORY SYSTEM MODEL DESCRIPTION

Here, we introduce the model that is analyzed as is in Chapters 3-4 and analyzed after some generalization in Chapter 5.

2.1 Problem Description

We first consider a production/inventory system that consists of a production stage that produces to stock to meet random demand as given in Figure 1.1. Each demand arrival is demanding one item. If there are available items in the inventory, the demand is satisfied and leaves the system. Otherwise, the demand is backlogged until an item is available. The release of the material into the production process is controlled by a policy that determines whether or not to produce depending on the information available at a given time. Figure 1.1 depicts the system.

2.1.1 Model

Production

The production time process can be in one of w different states. At an arbitrary time t, The state of the production time process is denoted by $\eta_W(t) \in \{1, \ldots, w\}$. When the production starts, it cannot be preempted. $M(t) \in \{0, 1\}$ indicates if the production stage is working (1) or idle (0). The production times can be correlated. This general representation of the production process allows analyzing various settings

where the release of materials into the production stage is controlled to match the output of the production stage with the demand.

Demand

The demand arrival process can be in one of d different states. The state of the demand process is denoted by $\eta_D(t) \in \{1, \ldots, d\}$. The demand inter-arrival times can be correlated. The demand for the production stage can be generated by another stage in production. For example, in the example described in Section 2.2, the output from an unreliable production line generates the demand for the production/inventory system.

Inventory Position

The difference between the cumulative production and cumulative demand at time t is referred to as the inventory position and denoted by $X(t)$. The inventory level and the backlog level are denoted by $X^+(t) = \max\{X(t), 0\}$ and $X^-(t) = \max\{-X(t), 0\}$ respectively. Every item in the inventory generates a cost of c^+ per unit time, and every demand waiting in the line generates a backorder cost of c^- per unit time.

Markings

The information signals referred to as the *markings* arrive as attached to the demand arrivals or production completions, or arrive separately. There are C_D markings for the demand process and C_W markings for the production process. The index of the last observed marking from the information and demand process at time t is denoted by $c_D(t) \in \{1, \ldots, C_D\}$. The index of the last observed marking from the production time process is denoted by $c_W(t) \in \{1, \ldots, C_W\}$. The markings are uniquely determined by the demand and production states. The arrival of information

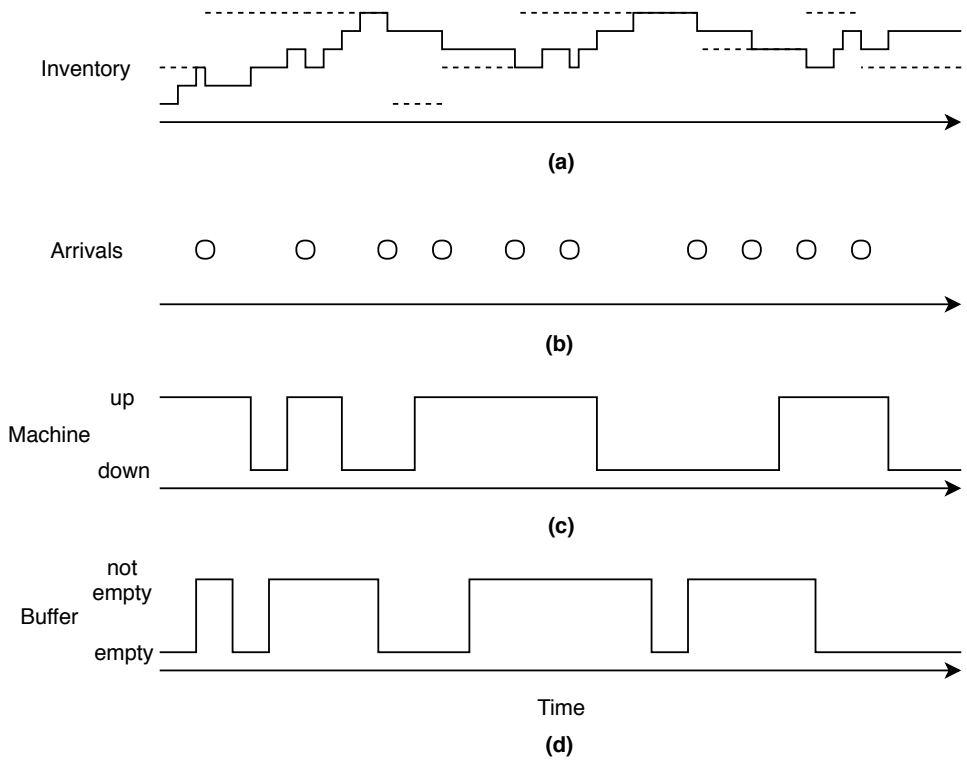

Figure 2.1: A sample path of the marked trace for the system described in Section 2.2

signals can be correlated. The number of markings is much smaller than the number of states, i.e., $C_D \ll d$ and $C_W \ll w$.

Figure 2.1 depicts the sample path of the marked trace for the system described in Section 2.2. In this example, the output from an unreliable production line generates the demand for the production/inventory system. The controller can use up to 4 markings for the demand process: 2 markings for the unreliable station (up or down) or 2 markings for the buffer level (empty or not empty) or 4 markings for both (down and empty, up and empty, down and not empty, and up and not empty).

State-Space Model

The state of the system at time t is $(X(t), \eta_D(t), \eta_W(t))$. We consider the case where the demand and production states $\eta_D(t)$ and $\eta_W(t)$ are not fully observable. Partial information about the system is available through the inventory status, the production status, and the marking processes, $(X(t), M(t), c_D(t), c_W(t))$. We model the system as a continuous-time discrete state-space process $\{(X(t), M(t), c_D(t), c_W(t)), t \geq 0\}$.

2.1.2 Production Control Problem

The optimization problem we consider is deciding on whether to produce or not depending on the inventory level, the production status, the marking of the last demand and information arrival, and the marking of the last production time in order to minimize the average inventory holding and backlog cost of the system in the long run.

Decision

Let $u^l(X(t), M(t), c_D(t), c_W(t))$ denote the decision to produce ($u^l = 1$) or not to produce ($u^l = 0$) depending on the inventory position, the machine status, and the last observed markings for demand and production at time t under policy l.

Objective

The problem is finding the optimal policy that minimizes the steady-state average cost π in the long run:

$$\pi^* = \min_l J^l = \mathbb{E}\left[\frac{1}{T}\lim_{T\to\infty}\int_{t=0}^{T}\left(c^+X^+(t) + c^-X^-(t)\right)dt \middle| X(0), \eta_D(0), \eta_W(0)\right]. \quad (2.1)$$

Optimal Control Policy

If the production times are i.i.d. exponential random variables and the inter-arrival times are i.i.d exponential random variables, then a base-stock policy that allows production when the inventory position is less than a threshold would be the optimal policy when the production and demand states are fully observable (Gavish and Graves, 1980; Sobel, 1982). Extensions of the basic base-stock model show that a state-dependent threshold policy is optimal for various systems under different assumptions (Song and Zipkin, 1993; Treharne and Sox, 2002; Arifoğlu and Özekici, 2011). For a system with correlated production and demand processes modeled as MAPs, Dizbin and Tan (2018) present the state-dependent threshold policy as the optimal policy.

The model presented in this study can be considered as a modified version of this problem where the controller cannot take action at certain states that are not observable, and the state-space is extended to include additional states corresponding to the markings. In this modified system, the state-dependent threshold policy would be the optimal policy. Since the state space includes the states corresponding to the markings, the marking-dependent threshold policy is a candidate for the optimal policy. However, a formal proof of the optimality of the marking-dependent control policy is not given in this thesis.

Marking-Dependent Threshold Policy

We refer to a control policy where the decision to produce or not depends on the inventory position and a threshold that depends on the last observed pair of markings as the *marking-dependent* threshold policy. Using the marked processes, the marking of each arrival and the marking of each production time bring some information about the state of the system. The control policy assigns a threshold to each pair of markings as opposed to assigning a threshold to the states of the processes, and production will

continue until this threshold is reached.

The marking-dependent threshold policy compares the inventory position at time t with the threshold level set for the last observed markings of the demand and production processes to trigger production. Namely, let S_{c_D,c_W} be the threshold level for the marking pair (c_D, c_W), $c_D \in \{1, 2, ..., C_D\}$, $c_W \in \{1, 2, ..., C_W\}$. The arrival of a demand while the last observed marking pair is (c_D, c_W) triggers production if there is no item being produced at that moment and the inventory level is less than or equal to S_{c_D,c_W}. Upon completion of a part, production is allowed to continue if the inventory level is less than S_{c_D,c_W}. If the inventory level reaches S_{c_D,c_W}, the production stops until a new demand or information signal arrives. The production will not start if the threshold that corresponds to the newly observed marking pair is lower than the inventory level. The control policy can be expressed as

$$u(X(t), M(t), c_D(t), c_W(t)) = \begin{cases} 1 & \text{if } X(t) < S_{c_D,c_W} \text{ and } M(t) = 0 \\ 0 & \text{otherwise} \end{cases}. \quad (2.2)$$

Parameters of the Marking-Dependent Threshold Policy

The parameters of the marking-dependent threshold policy are the thresholds corresponding to different markings. The threshold set $S = \{S_{c_D,c_W}\}$, $c_D \in \{1, 2, ..., C_D\}$, $c_W \in \{1, 2, ..., C_W\}$ includes $C_D \times C_W$ thresholds of the control policy. We define the lowest and the highest threshold levels as $\underline{S} = \min_{c_D,c_W} S_{c_D,c_W}$ and $\overline{S} = \max_{c_D,c_W} S_{c_D,c_W}$ respectively.

With much fewer parameters compared to a state-dependent threshold policy and as a policy that does not require full observability of the system, the marking-dependent threshold policy can easily be implemented in a production system. The implementation requires determining the markings that will be collected from the shop floor and setting the threshold for each marking. Once the thresholds are de-

termined, the real-time production decision can be given by comparing the inventory level with the thresholds corresponding to the last observed markings.

2.2 Example

Before describing the general model and the solution methodologies in detail, we first introduce a specific system to show the operation of the marking-dependent threshold policy in a production system. Note that the analytical method presented in Chapter 3 and the data-driven method described in Chapter 4 can be applied to a wider range of models and are not restricted to the assumptions of the specific system presented in this section.

We consider a setting where the downstream of a production system is fed by two upstream flows and the release into one flow is controlled to minimize the holding and backlog to synchronize the flows. This is similar to a setting where the release of the material into a feeder line is synchronized with the flow of the main assembly line. Figure 2.2 illustrates this production line together with the workstation that forms the production/inventory system.

The output of the upstream flow of the main assembly line is synchronized with the parts supplied by the feeder station denoted by WS_1. WS_1 has exponential processing time with rate μ_1. The release of material to WS_1 is controlled with a marking-dependent threshold policy.

The demand stream to the work station WS_1 is generated by a two-machine line with one unreliable machine WS_2, and one reliable machine WS_3 and a finite inter-station buffer of size q. The processing times of WS_2 and WS_3 are exponentially distributed random variables with rates μ_2 and μ_3. The throughput of the line is denoted with TP.

The unreliable machine WS_2 has operation-dependent failures and the failure time is exponentially distributed with rate $\gamma\mu_2$. If a breakdown occurs, the repair process

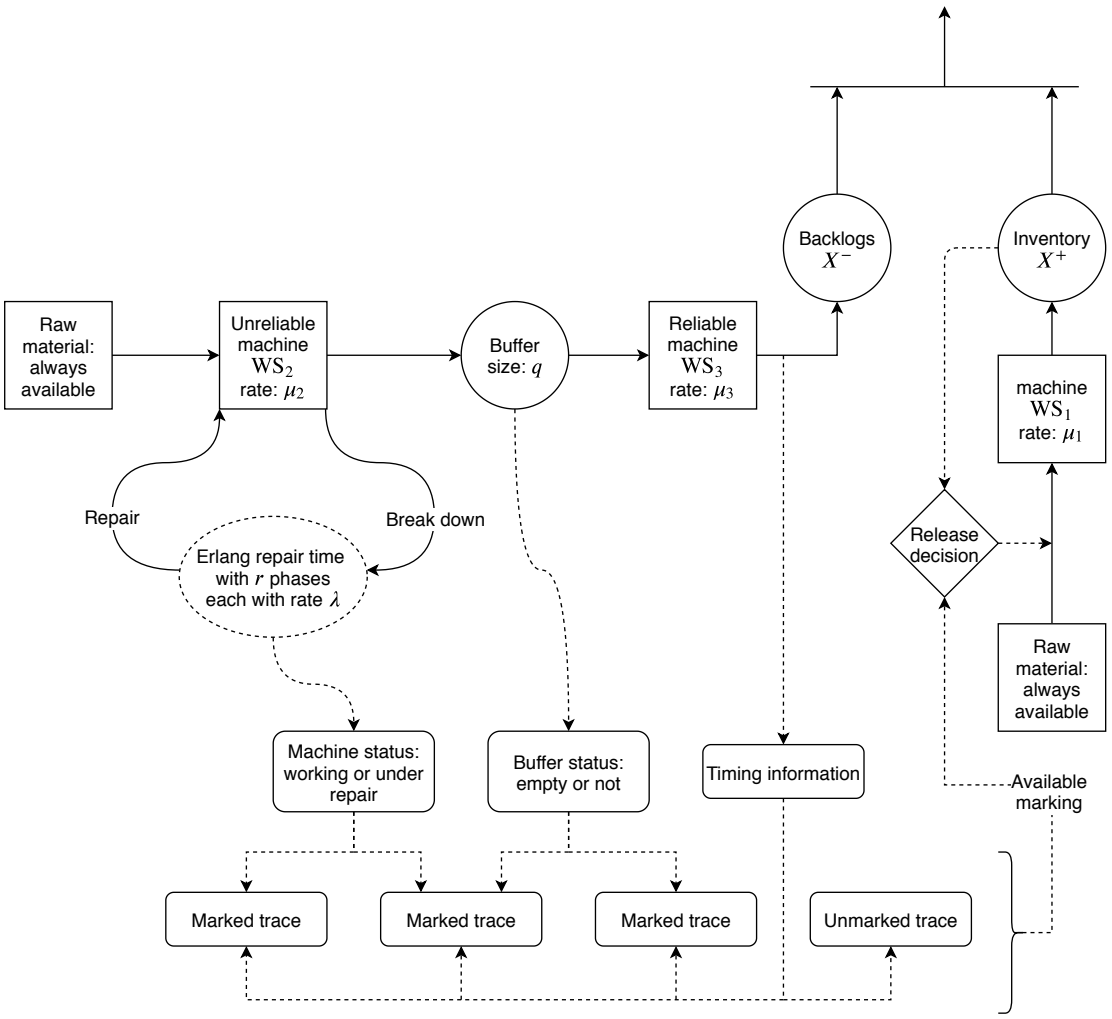

Figure 2.2: A production line with an unreliable machine. The output of this line is used as the information and demand process in the numerical experiments.

starts. The repair time follows an Erlang distribution with r phases each with rate λ. The phase of the repair process is not observable.

The markings collected from the shop-floor are related to the repair status of WS_2, whether the repair process is in progress or not, and the buffer status, whether the buffer is empty or not at the time of a demand arrival.

The inventory status of the buffer is recorded as *empty* or *not empty* as a mark-

ing. Figure 2.1 depicts a sample realization for this case. With each arrival (Figure 2.1b), the controller receives a marking that can include the corresponding machine availability status (Figure 2.1c) and/or the buffer status (Figure 2.1d). According to the production completion times and demand arrivals, the corresponding inventory sample path is shown in Figure 2.1a, where the dashed lines represent the threshold levels and the solid line represents the inventory level. In Figure 2.1, if both the machine availability and the buffer status is included in the marking, then there are $C_D = 4$ markings. If only the machine availability or the buffer status is used, then $C_D = 2$. If only the inter-arrival time is used by the controller and no additional information is used, then $C_D = 1$.

2.2.1 Plan of the Analysis

For given marked traces from a system, there are two alternatives for determining the best threshold levels. The first alternative is fitting stochastic processes to the traces and then using the fitted processes in the analytical method presented in Chapter 3. The second alternative is using the data-driven JSO that directly uses the data in optimization presented in Chapter 4. While we make the assumption that the process $\{(X(t), M(t), c_D(t), c_W(t)), t \geq 0\}$ can be modeled as a Marked Markovian Arrival Process in Chapter 3, we do not impose any assumptions for this process for the data-driven JSO method presented in Section 4.1. The analytical model presented in Chapter 3 also allows us to evaluate the performance of the data-driven method presented in Chapter 4. We use the specific system given in Section 2.2 to illustrate the application of these two different approaches.

In Section 4.2, we use this experimental setup to evaluate the data-driven JSO and compare its performance with the analytical solution. We will also discuss the effect of using different markings for control.

Chapter 3

ANALYSIS AND OPTIMIZATION OF A PRODUCTION/INVENTORY SYSTEM BY USING THE MARKING-DEPENDENT THRESHOLD POLICY AND MARKED MARKOV ARRIVAL PROCESSES

In order to analyze the production/inventory system, we first develop the quasi birth and death process representation of the system with MMAP production times and MMAP demand arrivals for the given values of the threshold for each marking pair. We then use the matrix geometric method to determine the steady-state probabilities. The steady-state probabilities give us the expected cost of the system. Finally, by using a mathematical programming formulation, we determine the set of the threshold levels for each marking pair that minimizes the expected cost.

In the following we give a review of the literature related to stochastic modelling and control of production/inventory systems.

Controlling production to match random supply with random demand has been the subject of numerous studies in the last 40 years. For a production/inventory system with i.i.d. exponential inter-arrival and service times, it has been shown that a base-stock policy is optimal e.g., (Gavish and Graves, 1980; Sobel, 1982). That is, production continues until the inventory level reaches a threshold. In this study,

---

The main portions of this work have been included in a manuscript that has been accepted for publication in the *International Journal of Production Economics* and is available from URL:https://doi.org/10.1016/j.ijpe.2019.107607

we consider demand and service processes that can have correlated inter-event times. Empirical studies and analytical models of production systems have shown that the autocorrelation of the inter-event times in these systems is not negligible (Wein, 1988; Schömig and Mittler, 1995; Tan and Lagershausen, 2017; Wein, 1988; Dizbin and Tan, 2018). Dizbin and Tan (2018) investigate how correlated inter-event times affect the steady-state behavior of a production/inventory system and show, via numerical experiments, how ignoring autocorrelation can lead to considerable losses.

In order to capture possible inter-dependency in the inter-event times, we use the framework of Markov arrival processes (MAP) introduced by Neuts (1979). Markov arrival processes are generalizations of phase-type distribution in the sense that the inter-event times of a Markov arrival process has phase-type distribution. However, unlike the phase-type distribution, after an arrival, the information about the phase of the system is not lost, hence the inter-arrival times might be dependent. In order to analyze MAPs, quasi birth and death processes (QBD) are commonly used in the literature. We mainly use the results of Ost (2013) and Horváth et al. (2010) for QBDs. In order to model a production system controlled with a marking-dependent threshold policy that uses partial information about the system status, we use the Marked MAPs (MMAP) (Qi-Ming and Neuts, 1998). MMAPs have been mostly used for modeling arrival of different classes of customers. In this work, we contribute to this literature by proposing using the MMAP framework to capture different sources of information about the state of the arrival process in different markings in a control setting.

The number of studies on the production control of systems with correlated demand and service processes is limited. These studies suggest that state-dependent base-stock policy is an effective control policy for these systems. Song and Zipkin (1993) prove the optimality of a state-dependent base-stock policy when production times are i.i.d. and the demand is a Markov modulated Poisson process. For systems

with correlated processes that can be modeled with MAPs, Dizbin and Tan (2018) evaluate the performance of the state-dependent base-stock policy and discuss the optimality of the state-dependent base-stock policy. Similarly, the studies on the production control for systems with partially observable processes also suggest that state-dependent base-stock policy can be used as a control mechanism (Bertsimas and Paschalidis, 2001; Treharne and Sox, 2002; Karaesmen et al., 2004; Arifoğlu and Özekici, 2011; Bayraktar and Ludkovski, 2010).

Our contribution to the literature on stochastic modelling and control of production/inventory systems is modelling and analysis of a partially observable production/inventory system controlled with a threshold policy that uses limited information about the system and proposing a solution method that is based on mathematical programming to determine the optimal thresholds.

3.0.1 Marked MAP Representation for Arrival Processes with Partial Information

In order to model an arrival process along with the associated partial information, we use the Marked MAP (MMAP) representation. A MMAP describes a MAP whose arrivals are marked.

Marked Markovian Arrival Processes

A MAP A is described by the matrices $(\mathbf{A0}, \mathbf{A1})$, where $\mathbf{A0}$ records the transitions that do not result in an arrival and $\mathbf{A1}$ records the transitions that result in an arrival. The infinitesimal generator matrix of the process A is $\mathbf{A0} + \mathbf{A1}$.

MMAPs are extensions of MAPs that allow for different types of arrivals (Qi-Ming and Neuts, 1998). A MMAP A is described by the matrices $(\mathbf{A0}, \mathbf{A1}_1, \ldots, \mathbf{A1}_C)$, where $\mathbf{A0}$ records the transitions that do not result in an arrival and $\mathbf{A1}_c$ records the transitions that result in an arrival marked c. The infinitesimal generator matrix of the process A is $\mathbf{A0} + \sum_{c=1}^{C} \mathbf{A1}_c$.

Model of Part Arrivals with Markings

We consider information arrivals at different times not necessarily coupled with part arrival times. Let $L = (\mathbf{L0}, \mathbf{L1}_1, \ldots, \mathbf{L1}_C, \mathbf{L2}_1, \ldots, \mathbf{L2}_C)$ denote the MMAP that captures the dynamics of the information and part arrivals. The transition rates corresponding to a part arrival coupled with the information marked i are captured in $\mathbf{L1}_i$. The transition rates corresponding to an arrival of a information signal marked i without a part arrival are captured in $\mathbf{L2}_i$.

Model of Production

Production happens one item at a time with a production time that evolves according to a MMAP. Once the production of an item starts, it cannot be interrupted until completion. We assume that the state of the production time process does not change during the idle periods.

The MMAP describing the production time process is denoted by

$$W = (\mathbf{W0}, \mathbf{W1}_1, ..., \mathbf{W1}_{C_W}) .$$

The number of states in W is w. Since a production cannot be interrupted, arrival of information about the production time process without completion of a production cannot be used for the control of the system. For this reason, entries of $\mathbf{W2}_1, ..., \mathbf{W2}_{C_W}$ are zero without loss of generality. For notational convenience, $\mathbf{W2}_1, ..., \mathbf{W2}_{C_W}$ have been dropped from the definition of W.

Model of Demand Arrivals with Markings

Similarly, we also assume that a unit of demand arrives according to a MMAP. Information about the demand process arrive with or without the arrival of a demand. Let $D = (\mathbf{D0}, \mathbf{D1}_1, \ldots, \mathbf{D1}_{C_D}, \mathbf{D2}_1, \ldots, \mathbf{D2}_{C_D})$ denote the MMAP of information and de-

Table 3.1: Description of the notation used in the matrix geometric analysis

Notation	Description
$\mathbf{I}_n$	identity matrix of size $n \times n$
$\mathbf{e}_{i,n}$	row vector of length n with 1 for ith entry and 0 elsewhere
$\mathbf{J}_{i,j,n}$	square matrix of size $n \times n$, with a single entry with value one in row i and column j
$\mathbf{1}_{n,m}$	$n \times m$ matrix of ones
$\mathbf{0}_{n,m}$	$n \times m$ matrix of zeros
$\delta_{\{x\}}$	binary indicator function specifying if statement x is correct or not
x^T	transpose of x
$\otimes$	Kronecker product

mand. The transition rates corresponding to a demand arrival marked i are captured in $\mathbf{D1}_i$. The transition rates corresponding to arrival of demand information marked i without a demand arrival are captured in $\mathbf{D2}_i$.

3.0.2 Evaluation of the Production/Inventory System for Given Thresholds

The process of shortfalls is not affected if all the thresholds are increased by a constant value. Therefore, in order to analyze the process $\{(X(t), M(t), c_D(t), c_W(t)), t \geq 0\}$ for given thresholds, we focus on the shortfall from the largest threshold $\overline{S}$. The shortfall level increases when a part is produced and decreases when a demand arrives and stays the same for all other transitions that do not change the inventory position. We define the number of shortfalls from the largest threshold value as $Y(t) = \overline{S} - X(t)$ and analyze the process $\{(Y(t), M(t), c_D(t), c_W(t)), t \geq 0\}$ as a QBD.

Model for the Shortfalls

The state space structure of the process $\{(Y(t), M(t), c_D(t), c_W(t)), t \geq 0\}$ is given in Figure 3.1. Let $\mathbf{Q}$ denote the CTMC generator matrix of the system, and in addition, let $\bar{\mathbf{Q}}_{Y,Y'}$ denote the sub-block of $\mathbf{Q}$ corresponding to transitions from shortfall level Y to shortfall level Y'.

With these submatrices, $\mathbf{Q}$ has the structure given in Equation (3.1). In this

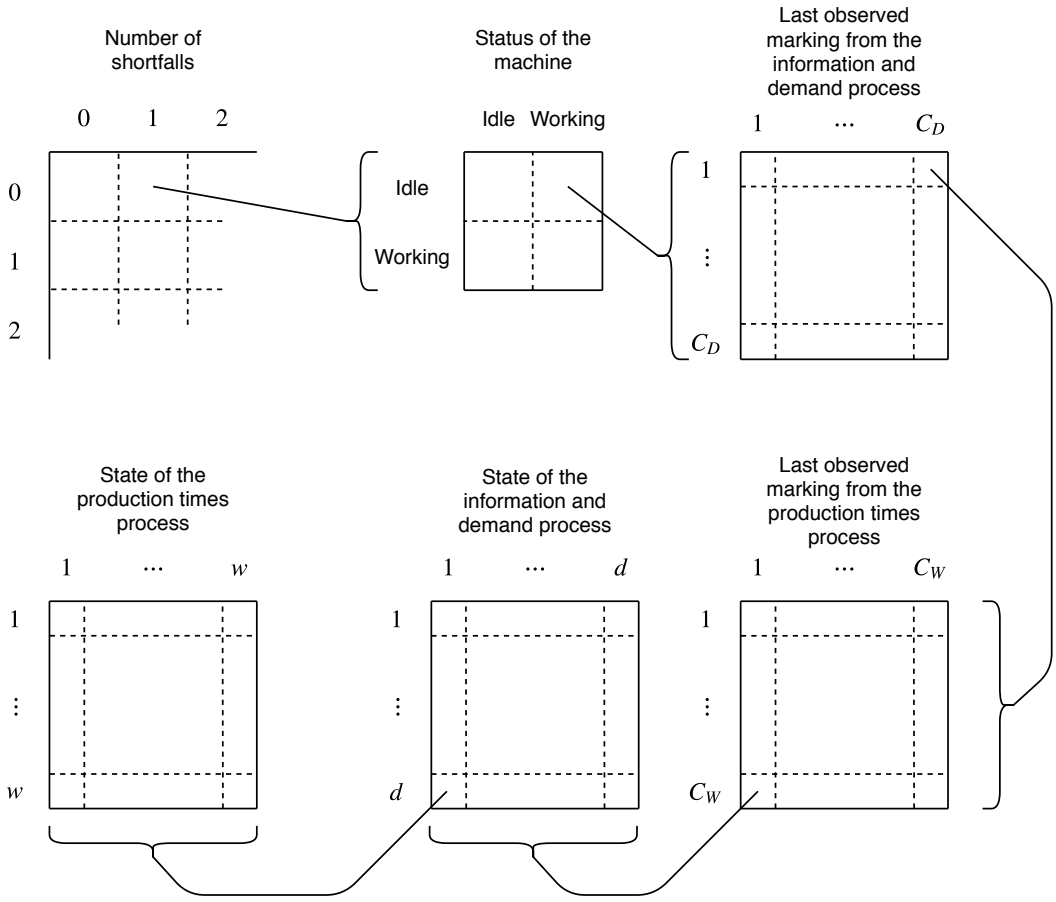

Figure 3.1: State-space structure of the process of shortfalls.

Quasi Birth and Death process, the size of each block of the corresponding CTMC for a given level of the shortfall will be $n \times n$ where $n = 2C_D C_W dw$.

$$\mathbf{Q} = \begin{bmatrix} \bar{\mathbf{Q}}_{0,0} & \bar{\mathbf{Q}}_{0,1} & & & \\ \bar{\mathbf{Q}}_{1,0} & \bar{\mathbf{Q}}_{1,1} & \bar{\mathbf{Q}}_{1,2} & & \\ & \bar{\mathbf{Q}}_{2,1} & \bar{\mathbf{Q}}_{2,2} & \bar{\mathbf{Q}}_{2,3} & \\ & & \ddots & \ddots & \ddots \\ & & & & \end{bmatrix}. \tag{3.1}$$

The sub-blocks of $\mathbf{Q}$ can be specified for the cases where the shortfall increases by one, stays the same, or decreases by one. The notation used in the determination of the sub-matrices for each case is given in Table 3.1.

The first sub-block $\bar{\mathbf{Q}}_{Y,Y+1}$ is related to the transitions that increase the level of shortfalls. Therefore, only demand arrivals appear in this sub-matrix:

$$\begin{aligned} \bar{\mathbf{Q}}_{Y,Y+1} &= \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \otimes \sum_{i_1=1}^{C_D} \sum_{j_1=1}^{C_W} \sum_{i_2=1}^{C_D} \sum_{j_2=1}^{C_W} \delta_{\left\{\overline{S}-(Y+1)\geq S_{i_2,j_2}\right\}} \mathbf{J}_{i_1 j_1, i_2 j_2, C_D C_W} \otimes \left(\delta_{\{j_1=j_2\}} \mathbf{D1}_{i_2} \otimes \mathbf{I}_w\right) \\ &+ \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \otimes \sum_{i_1=1}^{C_D} \sum_{j_1=1}^{C_W} \sum_{i_2=1}^{C_D} \sum_{j_2=1}^{C_W} \delta_{\left\{\overline{S}-(Y+1)< S_{i_2,j_2}\right\}} \mathbf{J}_{i_1 j_1, i_2 j_2, C_D C_W} \otimes \left(\delta_{\{j_1=j_2\}} \mathbf{D1}_{i_2} \otimes \mathbf{I}_w\right) \\ &+ \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} \otimes \mathbf{0}_{C_D C_W dw, C_D C_W dw} \\ &+ \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \otimes \left(\sum_{i=1}^{C_D} \left((\mathbf{e}_{i,C_D} \otimes \mathbf{1}_{C_D}) \otimes \mathbf{I}_{C_W} \otimes \mathbf{D1}_i \otimes \mathbf{I}_w \right) \right), \quad Y \geq 0. \end{aligned}$$

The first term in the above equation refers to transitions from states where the machine is *idle* to states where the machine is still *idle*. (i_1, j_1) denotes the marking pair before the arrival, and (i_2, j_2) refers to the marking pair after the arrival. Hence, making a transition to an *idle* state depends on the inventory level $\bar{S} - (Y+1)$ being greater than or equal to the threshold for the current marking pair S_{i_2,j_2}. The second term is the counterpart of the first term for the transition to a *working* state. The third term is zero because if the machine is *working*, an increase in the shortfall level cannot make it *idle*. The fourth term records the transitions that only affect the last observed marking pair. They change the last observed marking from the demand process.

The second sub-block $\bar{\mathbf{Q}}_{Y,Y}$ includes the transitions that do not affect the number of shortfalls:

$$
\begin{aligned}
\bar{\mathbf{Q}}_{Y,Y} &= \delta_{\{Y>0\}} \left(\begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \otimes \mathbf{I}_{C_D} \otimes \mathbf{I}_{C_W} \otimes \mathbf{I}_d \otimes \mathbf{W0} \right) + (\mathbf{I}_2 \otimes \mathbf{I}_{C_D} \otimes \mathbf{I}_{C_W} \otimes \mathbf{D0} \otimes \mathbf{I}_w) \\
&+ \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \otimes \sum_{i_1=1}^{C_D} \sum_{j_1=1}^{C_W} \sum_{i_2=1}^{C_D} \sum_{j_2=1}^{C_W} \delta_{\left\{\overline{S}-Y \geq S_{i_2,j_2}\right\}} \mathbf{J}_{i_1 j_1, i_2 j_2, C_D C_W} \otimes \left(\delta_{\{j_1=j_2\}} \mathbf{D2}_{i_2} \otimes \mathbf{I}_w \right) \\
&+ \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \otimes \sum_{i_1=1}^{C_D} \sum_{j_1=1}^{C_W} \sum_{i_2=1}^{C_D} \sum_{j_2=1}^{C_W} \delta_{\left\{\overline{S}-Y < S_{i_2,j_2}\right\}} \mathbf{J}_{i_1 j_1, i_2 j_2, C_D C_W} \otimes \left(\delta_{\{j_1=j_2\}} \mathbf{D2}_{i_2} \otimes \mathbf{I}_w \right) \\
&+ \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} \otimes \mathbf{0}_{C_D C_W dw, C_D C_W dw} \\
&+ \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \otimes \left(\sum_{i=1}^{C_D} \left((\mathbf{e}_{i,C_D} \otimes \mathbf{1}_{C_D}) \otimes \mathbf{I}_{C_W} \otimes \mathbf{D2}_i \otimes \mathbf{I}_w \right) \right), \quad Y \geq 0.
\end{aligned}
$$

These transitions include the phase changes in the processes that do not result in any arrivals and also the information arrivals received without a demand arrival. The

first set of transitions in the above equation are recorded in the first two terms. The other terms are similar to the previous equation, i.e. the arrival of an information signal can cause an *idle* machine to start working but cannot stop a *working* machine.

The last sub-block $\bar{\mathbf{Q}}_{Y,Y-1}$ includes the transitions that take place when a part is produced:

$$\begin{aligned}
\bar{\mathbf{Q}}_{Y,Y-1} &= \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \otimes \mathbf{0}_{C_D C_W dw, C_D C_W dw} \\
&+ \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \otimes \mathbf{0}_{C_D C_W dw, C_D C_W dw} \\
&+ \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} \otimes \sum_{i_1=1}^{C_D} \sum_{j_1=1}^{C_W} \sum_{i_2=1}^{C_D} \sum_{j_2=1}^{C_W} \delta_{\{\overline{S}-(Y-1) \geq S_{i_2,j_2}\}} \mathbf{J}_{i_1 j_1, i_2 j_2, C_D C_W} \otimes \left(\delta_{\{i_1=i_2\}} \mathbf{I}_d \otimes \mathbf{W1}_{j_2}\right) \\
&+ \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \otimes \sum_{i_1=1}^{C_D} \sum_{j_1=1}^{C_W} \sum_{i_2=1}^{C_D} \sum_{j_2=1}^{C_W} \delta_{\{\overline{S}-(Y-1) < S_{i_2,j_2}\}} \mathbf{J}_{i_1 j_1, i_2 j_2, C_D C_W} \otimes \left(\delta_{\{i_1=i_2\}} \mathbf{I}_d \otimes \mathbf{W1}_{j_2}\right), \qquad Y \geq 1.
\end{aligned}$$

In the above equation, the first two terms are zero because production cannot be completed if the machine is not *working*. The third term records the transitions to an *idle* state, depending on the threshold for the last observed marking pair S_{i_2,j_2}. The fourth term records transitions to a *working* state, from a *working* state, following the same pattern.

The Matrix Geometric Method

We determine the steady-state probabilities of the shortfall process by using the matrix geometric method (Ost, 2013). In order to use the matrix geometric method, the repeating levels and the boundary levels are specified, and $\mathbf{Q}$ is represented in the format given in Equation (3.2).

$$\mathbf{Q} = \begin{bmatrix} \mathbf{B}_{0,0} & \mathbf{B}_{0,1} & & & & \\ \mathbf{B}_{1,0} & \mathbf{B}_{1,1} & \mathbf{A}_0 & & & \\ & \mathbf{B}_{2,1} & \mathbf{A}_1 & \mathbf{A}_0 & & \\ & & \mathbf{A}_2 & \mathbf{A}_1 & \mathbf{A}_0 & \\ & & & \ddots & \ddots & \ddots \end{bmatrix} \tag{3.2}$$

When $Y > \overline{S} - \underline{S} + 1$, the sub-blocks are not dependent on Y. For these values of the shortfall, the boundary and repeating sub-matrices can be specified as given in Equations (3.3)-(3.8).

$$\mathbf{B}_{0,0} = \begin{bmatrix} \bar{\mathbf{Q}}_{0,0} & \bar{\mathbf{Q}}_{0,1} & & & & & \\ \bar{\mathbf{Q}}_{1,0} & \bar{\mathbf{Q}}_{1,1} & \bar{\mathbf{Q}}_{1,2} & & & & \\ & \bar{\mathbf{Q}}_{2,1} & \bar{\mathbf{Q}}_{2,2} & \bar{\mathbf{Q}}_{2,3} & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & \bar{\mathbf{Q}}_{\overline{S}-\underline{S}+1,\overline{S}-\underline{S}} & \bar{\mathbf{Q}}_{\overline{S}-\underline{S}+1,\overline{S}-\underline{S}+1} & \bar{\mathbf{Q}}_{\overline{S}-\underline{S}+1,\overline{S}-\underline{S}+2} \\ & & & & \bar{\mathbf{Q}}_{\overline{S}-\underline{S}+2,\overline{S}-\underline{S}+1} & \bar{\mathbf{Q}}_{\overline{S}-\underline{S}+2,\overline{S}-\underline{S}+2} \end{bmatrix} \tag{3.3}$$

$$\mathbf{B}_{0,1} = \begin{bmatrix} \mathbf{0}_n \\ \vdots \\ \mathbf{0}_n \\ \bar{\mathbf{Q}}_{\overline{S}-\underline{S}+2,\overline{S}-\underline{S}+3} \end{bmatrix} \tag{3.4}$$

$$\mathbf{B}_{1,0} = \begin{bmatrix} \mathbf{0}_n & \cdots & \mathbf{0}_n & \bar{\mathbf{Q}}_{\overline{S}-\underline{S}+3,\overline{S}-\underline{S}+2} \end{bmatrix} \tag{3.5}$$

$$\mathbf{B}_{1,1} = \bar{\mathbf{Q}}_{\overline{S}-\underline{S}+3,\overline{S}-\underline{S}+3} \tag{3.6}$$

$$\mathbf{B}_{2,1} = \bar{\mathbf{Q}}_{\overline{S}-\underline{S}+4,\overline{S}-\underline{S}+3} \tag{3.7}$$

$$\mathbf{A}_0 = \bar{\mathbf{Q}}_{\overline{S}-\underline{S}+5,\overline{S}-\underline{S}+6}, \mathbf{A}_1 = \bar{\mathbf{Q}}_{\overline{S}-\underline{S}+5,\overline{S}-\underline{S}+5}, \mathbf{A}_2 = \bar{\mathbf{Q}}_{\overline{S}-\underline{S}+5,\overline{S}-\underline{S}+4} \tag{3.8}$$

Since $\mathbf{B}_{0,0}$ depends on the threshold levels, the equations related to $\mathbf{B}_{0,0}$ must be solved for a given set of threshold levels.

Determining the Steady-State Probabilities

Given the aforementioned sub-matrices, the steady-state distribution can be calculated by using the successive substitution method. Let $\mathbf{p}$ denote the vector of steady-state probabilities. The elements of $\mathbf{p}$ are organized in sub-vectors according to the representation of $\mathbf{Q}$ in Equation (3.2): $\mathbf{p} = (\mathbf{b}, \mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2, \ldots)$.

The balance equations where $\mathbf{pQ} = 0$ and $\mathbf{p1} = 1$ can be written according to the partitioning of $\mathbf{Q}$ as

$$\mathbf{b}\mathbf{B}_{0,0} + \mathbf{v}_0\mathbf{B}_{1,0} = \mathbf{0}, \tag{3.9}$$

$$\mathbf{b}\mathbf{B}_{0,1} + \mathbf{v}_0\mathbf{B}_{1,1} + \mathbf{v}_1\mathbf{B}_{2,1} = \mathbf{0}, \tag{3.10}$$

$$\mathbf{v}_j\mathbf{A}_0 + \mathbf{v}_{j+1}\mathbf{A}_1 + \mathbf{v}_{j+2}\mathbf{A}_2 = \mathbf{0}, \quad j = 0, 1, 2, \ldots, \tag{3.11}$$

$$\mathbf{b}\mathbf{1} + \sum_{j=1}^{\infty} \mathbf{v}_j\mathbf{1} = 1. \tag{3.12}$$

The solution to the matrix difference equation given in Equation (3.11) is obtained by the matrix geometric structure as

$$\mathbf{v}_j = \mathbf{v}_0\mathbf{R}^j, \quad \mathbf{R} \in \mathbb{R}^{n\times n}. \tag{3.13}$$

where matrix $\mathbf{R}$ can be calculated independently of $\mathbf{v}_0$ by successive iterations with

$$\mathbf{R} = -\left(\mathbf{A}_0 + \mathbf{R}^2\mathbf{A}_2\right)\mathbf{A}_1^{-1}, \tag{3.14}$$

starting from $\mathbf{R} = \mathbf{0}_n$, until convergence (Ost, 2013). Given matrix $\mathbf{R}$, $\mathbf{p}$ is obtained by solving the linear equations (3.9)-(3.12). Finally, when the steady-state distribution $\mathbf{p}$ is obtained, the average cost of the system can be calculated as

$$\pi(S) = \sum_{x=0}^{\infty} \sum_{i=1+nx}^{n(x+1)} p_i \left(c^+\left[\overline{S} - x\right]^+ + c^-\left[x - \overline{S}\right]^+\right), \tag{3.15}$$

where p_i is the ith element of $\mathbf{p}$ and $\mathbf{prob}(X = x) = \sum_{i=1+nx}^{n(x+1)} p_i$.

3.0.3 Mathematical Programming Formulation to Determine the Optimal Thresholds

The method described in the preceding section yields the steady-state performance measures of the system for the *given* threshold values. In this section, we present a mathematical programming formulation to determine the *optimal* thresholds for the marking-dependent threshold policy. At every given inventory level and the last observed marking pair, if the machine is idle, there is a choice of starting production or doing nothing. Since the thresholds are non-negative and the transition rates between the repeating levels do not depend on the threshold levels, this problem can be restated as choosing among the possible options for the boundary state transitions of a QBD in order to minimize the steady-state cost of the system.

In other words, depending on the choice, some transitions will be added or eliminated from $\mathbf{B}_{0,0}$. Hence, the possible options for the boundary state transitions recorded in $\mathbf{B}_{0,0}$ are a function of a set of actions that the controller can take. In order to determine the thresholds in a computationally efficient way, we formulate the problem as a MIP problem.

We first introduce the MIP formulation for a more general problem where the objective is deciding on adding or eliminating transitions between the boundary states of a QBD in order to minimize the steady-state cost generated by the QBD.

Let $y_k : k \in \{1, .., K\}$ denote the binary actions that the controller can take. In the case of the marking-dependent threshold policy, y_k values correspond to the decision to continue production for different inventory levels and marking pairs. If $y_k = 1$, certain transitions are allowed and if $y_k = 0$, these transition are prohibited. Hence, $\mathbf{B}_{0,0} = \mathbf{B}^0_{0,0} + \sum_{k=1}^{K} y_k \mathbf{B}^k_{0,0}$, where $\mathbf{B}^0_{0,0}$ is the boundary transition matrix if $y_k = 0, \forall k$ and $\mathbf{B}^k_{0,0}$ denotes the transitions that depend on $y_k = 1$.

Any admissible action must correspond to a well-defined transition matrix. Note that $\mathbf{B}^k_{0,0}$ can have negative off-diagonal values. These negative values represent the transitions that are prohibited when $y_k = 1$. However, $\mathbf{B}_{0,0}$ cannot have off-diagonal

negative values at any feasible point. This must be reflected in the constraints on y_k values. These constraints along with other operational constraints on the binary variables can be represented as $\mathbf{Ey} \geq \mathbf{g}$, where $\mathbf{y}$ denotes the vector of binary variables, $\mathbf{E}$ is a matrix with K columns, and $\mathbf{g}$ is a column vector of appropriate size. Then the problem can be expressed as given in Equations (3.16)-(3.22) :

$$\min \mathbf{f}\mathbf{p}^T \tag{3.16}$$

$$\mathbf{p} = \begin{bmatrix} \mathbf{b} & \mathbf{v}_0 & \mathbf{v}_1 & \mathbf{v}_2 & \cdots \end{bmatrix} \tag{3.17}$$

$$\mathbf{f} = \begin{bmatrix} \mathbf{f_b} & \mathbf{f}_{\mathbf{v}_0} & \mathbf{f}_{\mathbf{v}_1} & \mathbf{f}_{\mathbf{v}_2} & \cdots \end{bmatrix} \tag{3.18}$$

$$\mathbf{b}\mathbf{1}_{m,1} + \mathbf{v}_0(\mathbf{I}_n - \mathbf{R})^{-1}\mathbf{1}_{n,1} = 1 \tag{3.19}$$

$$\begin{bmatrix} \mathbf{b} & \mathbf{v}_0 \end{bmatrix} \begin{bmatrix} \mathbf{B}^0_{0,0} + \sum_{k=1}^{K} y_k \mathbf{B}^k_{0,0} & \mathbf{B}_{0,1} \\ \mathbf{B}_{1,0} & \mathbf{B}_{1,1} + \mathbf{R}\mathbf{A}_2 \end{bmatrix} = \mathbf{0}_{1,m+n} \tag{3.20}$$

$$y_k \in \{0,1\}^K \tag{3.21}$$

$$\mathbf{Ey} \geq \mathbf{g}, \tag{3.22}$$

where the size of $\mathbf{B}^0_{0,0}$ is denoted by m and the size of $\mathbf{A}_0$ (referred to as the block size) is denoted by n. The vector of cost rates for each state of the CTMC is denoted by $\mathbf{f}$. The cost rates for the boundary states is denoted by $\mathbf{f_b}$, and $\mathbf{f}_{\mathbf{v}_0}, \mathbf{f}_{\mathbf{v}_1}, \ldots$ denote the cost rates for the repeating states. Since $\mathbf{R}$ does not depend on the actions of the controller, it is calculated prior to solving the problem and treated as a parameter in this formulation.

This formulation contains quadratic constraints in Equation (3.20) and infinite number of variables. Given that $\mathbf{R}$ is calculated before solving the problem, the infinite number of variables can be reduced to finite variables using the matrix geometric

relations. In addition, the quadratic constraints can be linearized using additional variables. Let $\left(\mathbf{u}_k^+ - \mathbf{u}_k^-\right)^T = y_k \left(\mathbf{B}_{0,0}^k\right)^T \mathbf{b}^T$ denote the contribution of y_k to the steady-state Equations (3.20). Hence, the problem can be reformulated as the MIP given in Equations (3.23)-(3.32).

$$\min \begin{bmatrix} \mathbf{f_b} & \left(\sum_{j=0}^{\infty} \mathbf{R}^j \left(\mathbf{f}_{\mathbf{v}_j}\right)^T \right)^T \end{bmatrix} \begin{bmatrix} \mathbf{b}^T \\ {\mathbf{v}_0}^T \end{bmatrix} \tag{3.23}$$

$$\begin{bmatrix} \mathbf{1}_{1,m} & \left((\mathbf{I}_n - \mathbf{R})^{-1} \mathbf{1}_{n,1} \right)^T \end{bmatrix} \begin{bmatrix} \mathbf{b}^T \\ {\mathbf{v}_0}^T \end{bmatrix} = 1 \tag{3.24}$$

$${\mathbf{B}_{0,0}^0}^T \mathbf{b}^T + \sum_{k=1}^{K} \left(\mathbf{u}_k^+ - \mathbf{u}_k^-\right)^T + {\mathbf{B}_{1,0}}^T {\mathbf{v}_0}^T = \mathbf{0}_{m,1} \tag{3.25}$$

$$\begin{bmatrix} {\mathbf{B}_{0,1}}^T & {\mathbf{B}_{1,1}}^T + \left(\mathbf{R}\mathbf{A}_2\right)^T \end{bmatrix} \begin{bmatrix} \mathbf{b}^T \\ {\mathbf{v}_0}^T \end{bmatrix} = \mathbf{0}_{n,1} \tag{3.26}$$

$$\left(\mathbf{B}_{0,0}^k\right)^T \mathbf{b}^T \leq \left(\mathbf{u}_k^+ - \mathbf{u}_k^-\right)^T + (1 - y_k)\,\mathbf{M} \quad 1 \leq k \leq K \tag{3.27}$$

$$\left(\mathbf{B}_{0,0}^k\right)^T \mathbf{b}^T \geq \left(\mathbf{u}_k^+ - \mathbf{u}_k^-\right)^T - (1 - y_k)\,\mathbf{M} \quad 1 \leq k \leq K \tag{3.28}$$

$$\mathbf{u}_k^{+T} \leq y_k \mathbf{M} \quad 1 \leq k \leq K \tag{3.29}$$

$$\mathbf{u}_k^{-T} \leq y_k \mathbf{M} \quad 1 \leq k \leq K \tag{3.30}$$

$$y_k \in \{0,1\}^K \tag{3.31}$$

$$\mathbf{E}\mathbf{y} \geq \mathbf{g} \tag{3.32}$$

The big-M values in the $\mathbf{M}$ vectors in Equation (3.27) and Equation (3.29) can be set to $\mathbf{M} = \sum_{i=1}^{m} {\mathbf{e}_{i,m}}^T \max\left\{\mathbf{B}_{0,0}^k {\mathbf{e}_{i,m}}^T\right\}$ and in Equation (3.28) and Equation (3.30)

Table 3.2: Number of variables and constraints of the MIP

Variables		Constraints
Continuous	Binary	
$(2K+1)m+n$	K	$(4K+1)m+n+1$ (in addition to $\mathbf{Ey} \geq \mathbf{g}$)

to $\mathbf{M} = -\sum_{i=1}^{m} \mathbf{e}_{i,m}{}^{T} \min\left\{\mathbf{B}_{0,0}^{k}\mathbf{e}_{i,m}{}^{T}\right\}$. When $\mathbf{B}_{0,0}^{k}$ matrices are sparse, this setting eliminates a considerable chunk of the variables prior to solving the model using their respective big-M values. In addition, after eliminating these variables, a large number of constraints will be eliminated as well because their relevance depended on the ability of the eliminated variables to take positive values. The number of variables and constraints in this formulation is given in Table 3.2.

Given the general formulation (3.23)-(3.32), in order to solve the problem of choosing the optimal thresholds for the marking-dependent threshold policy, the relevant binary variables have to be defined and $\mathbf{B}_{0,0}^{0}, \mathbf{B}_{0,0}^{k}$ must be specified accordingly. The steps for forming these matrices are given in Appendix 3.1.

In summary, given the MMAP representations of the demand and production D, W, the costs c^{-}, c^{+}, we first determine the matrix $\mathbf{R}$ using successive substitution using Equation (3.14). Then, we use an estimate for the upper bound for the thresholds, denoted by $\bar{X}$ to build $\mathbf{B}_{0,0}^{0}, \mathbf{B}_{0,0}^{k}, \mathbf{B}_{1,0}, \mathbf{B}_{0,1}, \mathbf{B}_{1,1}, \mathbf{A}_{2}$ with the method given here using Equations (3.4)-(3.8) and (3.39)-(3.40) in Section 3.1. Finally, all this information is given as parameters to the MIP formulation given in Equations (3.23)-(3.32). Then, the solution of the MIP problem is converted to the optimal threshold levels using Equation (3.37) in Section 3.1.

3.1 Specifying the Intermediate Matrices for MIP

Let $\bar{X}$ denote the maximum threshold level admissible. Let $\psi_{P,i,j} : 1 \leq i \leq C_D, 1 \leq j \leq C_W, 0 \leq X < \bar{X}$ denote the binary variable for the decision to continue production when the inventory position is X and marking pair (i, j) has been observed last. Hence, $\psi_{X,i,j} = \delta_{\{S_{i,j} > X\}}$ holds. Given this definition, for a policy to be a threshold policy, the following set of constraints must hold. Note that the constraints given by Equation (3.33) correspond to Equation (3.32) in the general formulation:

$$\psi_{P,i,j} \leq \psi_{P-1,i,j} : 1 \leq i \leq C_D, 1 \leq j \leq C_W, 0 < X < \bar{X}. \tag{3.33}$$

For determining $\mathbf{E}$ and $\mathbf{g}$, we first choose an order for matching $\psi_{X,i,j}$ values and the elements of the vector $\mathbf{y}$. This order is given in Equation (3.34), where $\ddot{\psi}_X$ and $\dot{\psi}_{X,i}$ are intermediary sub-vectors of $\mathbf{y}$. Let $f(X, i_1, i_2) = XC_DC_W + (i_1 - 1)C_W + i_2$ denote a function from $\{0, .., \bar{X} - 1\} \times \{1, .., C_D\} \times \{1, .., C_W\}$ to $\{1, .., \bar{X}C_DC_W\}$, used for pointing to ψ_{X,i_1,i_2} in $\mathbf{y}$. Using this order and the function f, the constraints given in Equation (3.33) can be specified by $\mathbf{E}$ and $\mathbf{g}$ given in Equations (3.35)-(3.36):

$$\mathbf{y} = \begin{bmatrix} \ddot{\psi}_0 & \cdots & \ddot{\psi}_{\bar{X}-1} \end{bmatrix}, \ddot{\psi}_X = \begin{bmatrix} \dot{\psi}_{X,1} & \cdots & \dot{\psi}_{X,C_D} \end{bmatrix}, \dot{\psi}_{X,i} = \begin{bmatrix} \psi_{X,i,1} & \cdots & \psi_{X,i,C_W} \end{bmatrix}, \tag{3.34}$$

$$\mathbf{E} = \sum_{X=1}^{\bar{X}-1} \sum_{i_1=1}^{C_D} \sum_{i_2=1}^{C_W} \left(\mathbf{e}_{(X-1)C_DC_W + (i_1-1)C_W + i_2, (\bar{X}-1)C_DC_W} \right)^T \otimes \left(\mathbf{e}_{f(X,i_1,i_2), \bar{X}C_DC_W} - \mathbf{e}_{f(X-1,i_1,i_2), \bar{X}C_DC_W} \right), \tag{3.35}$$

$$\mathbf{g} = \mathbf{0}_{(\bar{X}-1)C_DC_W,1}. \tag{3.36}$$

Moreover, given a vector $\mathbf{y}$, the threshold level for marking pair (i_1, i_2) can be calculated as

$$S_{i_1,i_2} = \sum_{X=0}^{\bar{X}-1} \mathbf{y}_{f(X,i_1,i_2)}. \tag{3.37}$$

In the following we describe determining the alternative forms of the boundary state transitions. Let $\tilde{\mathbf{Q}}_{X_1,x_2}(\Psi)$ denote the sub-block of $\mathbf{Q}$ corresponding to transitions from inventory position X_1 to inventory position X_2, as a function of the binary variables $\Psi \in \{0,1\}^{\bar{X}C_DC_W}$. Given this notation, the sub-blocks of $\mathbf{Q}$ as a function of Ψ can be specified by the following equations:

$$\begin{aligned}
\tilde{\mathbf{Q}}_{X,X-1}(\Psi) &= \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \otimes \sum_{i_1=1}^{C_D}\sum_{j_1=1}^{C_W}\sum_{i_2=1}^{C_D}\sum_{j_2=1}^{C_W} (1-\psi_{X-1,i_2,j_2})\mathbf{J}_{i_1j_1,i_2j_2,C_DC_W} \otimes \left(\delta_{\{j_1=j_2\}}\mathbf{D1}_{i_2} \otimes \mathbf{I}_w\right) \\
&+ \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \otimes \sum_{i_1=1}^{C_D}\sum_{j_1=1}^{C_W}\sum_{i_2=1}^{C_D}\sum_{j_2=1}^{C_W} \psi_{X-1,i_2,j_2}\mathbf{J}_{i_1j_1,i_2j_2,C_DC_W} \otimes \left(\delta_{\{j_1=j_2\}}\mathbf{D1}_{i_2} \otimes \mathbf{I}_w\right) \\
&+ \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} \otimes \mathbf{0}_{C_DC_Wdw,C_DC_Wdw} \\
&+ \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \otimes \left(\sum_{i=1}^{C_D} ((\mathbf{e}_{i,C_D} \otimes \mathbf{1}_{C_D}) \otimes \mathbf{I}_{C_W} \otimes \mathbf{D1}_i \otimes \mathbf{I}_w)\right), \quad X \leq \bar{X},
\end{aligned}$$

$$
\begin{aligned}
\tilde{\mathbf{Q}}_{X,X}(\Psi) &= \delta_{\{X<\bar{X}\}}\left(\begin{bmatrix}0&0\\0&1\end{bmatrix}\otimes\mathbf{I}_{C_D}\otimes\mathbf{I}_{C_W}\otimes\mathbf{I}_d\otimes\mathbf{W0}\right)+(\mathbf{I}_2\otimes\mathbf{I}_{C_D}\otimes\mathbf{I}_{C_W}\otimes\mathbf{D0}\otimes\mathbf{I}_w)\\
&+\begin{bmatrix}1&0\\0&0\end{bmatrix}\otimes\sum_{i_1=1}^{C_D}\sum_{j_1=1}^{C_W}\sum_{i_2=1}^{C_D}\sum_{j_2=1}^{C_W}(1-\psi_{X,i_2,j_2})\mathbf{J}_{i_1j_1,i_2j_2,C_DC_W}\otimes\left(\delta_{\{j_1=j_2\}}\mathbf{D2}_{i_2}\otimes\mathbf{I}_w\right)\\
&+\begin{bmatrix}0&1\\0&0\end{bmatrix}\otimes\sum_{i_1=1}^{C_D}\sum_{j_1=1}^{C_W}\sum_{i_2=1}^{C_D}\sum_{j_2=1}^{C_W}\psi_{X,i_2,j_2}\mathbf{J}_{i_1j_1,i_2j_2,C_DC_W}\otimes\left(\delta_{\{j_1=j_2\}}\mathbf{D2}_{i_2}\otimes\mathbf{I}_w\right)\\
&+\begin{bmatrix}0&0\\1&0\end{bmatrix}\otimes\mathbf{0}_{C_DC_WdW,C_DC_WdW}\\
&+\begin{bmatrix}0&0\\0&1\end{bmatrix}\otimes\left(\sum_{i=1}^{C_D}((\mathbf{e}_{i,C_D}\otimes\mathbf{1}_{C_D})\otimes\mathbf{I}_{C_W}\otimes\mathbf{D2}_i\otimes\mathbf{I}_w)\right),\quad X\le\bar{X},
\end{aligned}
$$

$$
\begin{aligned}
\tilde{\mathbf{Q}}_{X,X+1}(\Psi) &= \begin{bmatrix}1&0\\0&0\end{bmatrix}\otimes\mathbf{0}_{C_DC_WdW,C_DC_WdW}\\
&+\begin{bmatrix}0&1\\0&0\end{bmatrix}\otimes\mathbf{0}_{C_DC_WdW,C_DC_WdW}\\
&+\begin{bmatrix}0&0\\1&0\end{bmatrix}\otimes\sum_{i_1=1}^{C_D}\sum_{j_1=1}^{C_W}\sum_{i_2=1}^{C_D}\sum_{j_2=1}^{C_W}(1-\psi_{X+1,i_2,j_2})\mathbf{J}_{i_1j_1,i_2j_2,C_DC_W}\otimes\left(\delta_{\{i_1=i_2\}}\mathbf{I}_d\otimes\mathbf{W1}_{j_2}\right)\\
&+\begin{bmatrix}0&0\\0&1\end{bmatrix}\otimes\sum_{i_1=1}^{C_D}\sum_{j_1=1}^{C_W}\sum_{i_2=1}^{C_D}\sum_{j_2=1}^{C_W}\psi_{X+1,i_2,j_2}\mathbf{J}_{i_1j_1,i_2j_2,C_DC_W}\otimes\left(\delta_{\{i_1=i_2\}}\mathbf{I}_d\otimes\mathbf{W1}_{j_2}\right),\quad X<\bar{X}.
\end{aligned}
$$

Then, the boundary transitions denoted by $\tilde{\mathbf{B}}_{0,0}(\Psi)$ can be derived by Equation (3.38). In addition, $\mathbf{B}^0_{0,0}, \mathbf{B}^k_{0,0}$ can be calculated accordingly using Equations (3.39)-(3.40).

$$\tilde{\mathbf{B}}_{0,0}(\Psi) = \begin{bmatrix} \tilde{\mathbf{Q}}_{\bar{X},\bar{X}}(\Psi) & \tilde{\mathbf{Q}}_{\bar{X},\bar{X}-1}(\Psi) & & & & & \\ \tilde{\mathbf{Q}}_{\bar{X}-1,\bar{X}}(\Psi) & \tilde{\mathbf{Q}}_{\bar{X}-1,\bar{X}-1}(\Psi) & \tilde{\mathbf{Q}}_{\bar{X}-1,\bar{X}-2}(\Psi) & & & & \\ & \tilde{\mathbf{Q}}_{\bar{X}-2,\bar{X}-1}(\Psi) & \tilde{\mathbf{Q}}_{\bar{X}-2,\bar{X}-2}(\Psi) & \tilde{\mathbf{Q}}_{\bar{X}-2,\bar{X}-3}(\Psi) & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & \tilde{\mathbf{Q}}_{0,1}(\Psi) & \tilde{\mathbf{Q}}_{0,0}(\Psi) & \tilde{\mathbf{Q}}_{0,-1}(\Psi) & \\ & & & & \tilde{\mathbf{Q}}_{-1,0}(\Psi) & \tilde{\mathbf{Q}}_{-1,-1}(\Psi) & \end{bmatrix} \tag{3.38}$$

$$\mathbf{B}^0_{0,0} = \tilde{\mathbf{B}}_{0,0}(\Psi), \psi_{X,i,j} = 0 : 1 \le i \le C_D, 1 \le j \le C_W, 0 \le X < \bar{X} \tag{3.39}$$

$$\mathbf{B}^k_{0,0} = \tilde{\mathbf{B}}_{0,0}(\Psi) - \mathbf{B}^0_{0,0}, \psi_{X,i,j} = \delta_{\{f(X,i,j)=k\}} : 1 \le i \le C_D, 1 \le j \le C_W, 0 \le X < \bar{X} \tag{3.40}$$

Given this representation, in terms of the general formulation, $n = 2C_DC_Wdw$ and $m = n(\bar{X}+2)$ and $K = C_DC_W\bar{X}$ and $\mathbf{E}$ has $C_DC_W\bar{X}$ rows. Hence, the final formulation has $2C_DC_W(1+(2+\bar{X})(1+2C_DC_W\bar{X}))dw$ continuous and $C_DC_W\bar{X}$ binary variables and $1+C_DC_W(\bar{X}+2(3+\bar{X}(1+4C_DC_W(2+\bar{X})))dw)$ constraints. However, in practice, as $\bar{X}$ increases, $\mathbf{B}^k_{0,0}$ matrices become more sparse, and as a result, a considerable number of the continuous variables and constraints can be eliminated prior to solving the problem using the respective big-M values (The big-M values for the redundant variables will be zero).

Chapter 4

DATA-DRIVEN JOINT SIMULATION AND OPTIMIZATION

Implementing the analytical method presented in Chapter 3 as a real-time control policy requires estimating the MMAP representations for the production, demand, and information inter-event times by using the historical data.

Alternatively, the parameters for the marking-dependent threshold policy can be determined by simulating a system controlled with the marking-dependent threshold policy and then using a simulation optimization approach.

In this Chapter, we present a mathematical programming approach for joint simulation and optimization of a production system that is controlled with the marking-dependent threshold policy.

4.1 Data-Driven Control of the Production/Inventory System by Using Marking-Dependent Threshold Policy

We give the outline of the discrete event simulation algorithm for the production inventory system presented in this study in Section 4.1.5.

The main portions of this work have been included in a manuscript that has been accepted for publication in the *International Journal of Production Economics* and is available from URL:https://doi.org/10.1016/j.ijpe.2019.107607

4.1.1 Literature Review

In the following, we first review the literature related to usage of joint simulation and optimization as a data-driven control method

In this study, we propose using joint simulation and optimization as a data-driven control method. Data-driven control can be defined as a control scheme that does not make use of explicit information about the mathematical model of the input data streams (Akcay et al., 2011; Hou and Wang, 2013). Data-driven optimization methods have been developed in close relation to robust optimization (Bertsimas and Thiele, 2006; Rudin and Vahn, 2018; Gallego and Moon, 1993).

The joint simulation and optimization method we propose uses the empirical data together with the available statistical information. This allows using the JSO approach as a data-driven control method based on the real-time data collected from the shop-floor combined with the simulated data by using the bootstrapping approach. Schruben (2000) was the first to introduce a joint simulation and optimization formulation that was able to replicate a simulation run of a system.

Most of the joint simulation and optimization formulations make use of objective functions that ensure that events start and finish at the correct time instances. The literature that followed this method was for the most part concerned with scheduling (Helber et al., 2011; Alfieri and Matta, 2012b). This is because scheduling also aims at making sure that every event happens as soon as possible. Whereas minimizing the average cost in an inventory system does not necessarily schedule events to happen as soon as possible. Chan and Schruben (2004) argue that finding the exact event times without making use of the objective function is only possible through adding a large number of integer variables. Since the exact event times are needed to model the marking-dependent base-stock policy in a model with both backlog and inventory costs, our formulation is a mixed integer programming formulation.

The work of Schruben (2000) has been extended to design and control of pro-

duction systems. Alfieri and Matta (2012b) give the mathematical programming formulation for simulation of different types of production systems. Alfieri and Matta (2012b) and Pedrielli et al. (2015) give formulations for pull control policies however when the control parameters are variables in the problem, they do not consider the backlog costs. Alfieri and Matta (2012a) give approximate mathematical programming formulations for the aforementioned problems by using the concept of time buffers. Tan (2015) uses joint simulation and optimization approach to model and analyze flow rate control problems for production systems with continuous and discrete state space. Hosseini and Tan (2017) extend these results to model a two-stage continuous flow production system with a finite buffer.

We contribute to this stream of literature by proposing JSO as a real-time control method and providing a joint simulation and optimization formulation for controlling a system with multiple thresholds. This formulation can be used to determine the parameters of the control policy by using the collected data with the available statistical information about the processes.

4.1.2 Data

The data-driven JSO approach uses a trace of information and demand arrival times, their markings, the indicators for demand and information, a trace of the production times and their markings gathered from the facility together with the available statistical information about the random variables. The collected traces can also be used to generate additional replications by using the bootstrapping methods. In this study, the details of combining collected and generated traces with the simulated data are not given and we use a general definition of a trace that combines collected, generated, and simulated arrival times.

The information and demand inter-arrival times are denoted as $\mathcal{T} = \{\tau_1, \tau_2, ...\}$. The time of the ith arrival is denoted with $a_i = \sum_{k=1}^{i} \tau_i$ and the fist arrival happens

at time τ_1. The corresponding markings at the moment of these arrivals are denoted by $\alpha_i \in \{1, \ldots, C_D\}$. An indicator vector $\Xi = \{\xi_i\}$ is used to distinguish a demand arrival coupled with information ($\xi_i = 1$) and an information arrival without the demand ($\xi_i = 0$) for the ith arrival. The trace of the observed production times is denoted as $G = \{g_1, g_2, ...\}$ and their markings are denoted by $\beta_k \in \{1, \ldots, C_W\}$.

4.1.3 Joint Simulation and Optimization Representation

In this section, we give the joint simulation and optimization formulation for the problem of determining the optimal marking-dependent thresholds S_{c_D,c_W} for each marking pair that minimize the average total cost for given traces. Table 4.1 gives a short description of the variables and parameters of the problem.

The full mathematical programming formulation of the problem for one replication is given in Equations (4.1)-(4.32).

Objective Function

The objective is minimizing the total inventory and backlog costs for all the demand arrivals. Since there are no lost sales in this model, when the system starts with no items in the inventory, the item that takes $g_{\sum_{l\leq i}\xi_l}$ time units to produce will be given to the customer arriving at a_i if $\xi_i = 1$. If $\xi_i = 0$, then arrival i does not produce any inventory or backlog cost directly. Hence let R_k be the time that the kth production starts and F_k be the time that it ends, then the total cost produced by arrival i will be $\xi_i \left(c^- \left[F_k - a_i\right]^+ + c^+ \left[a_i - F_k\right]^+\right)$ where $k = \sum_{l\leq i} \xi_l$. As a result, the total cost of the system will be $\sum_i \xi_i \left(c^- \left[F_{\sum_{l\leq i}\xi_l} - a_i\right]^+ + c^+ \left[a_i - F_{\sum_{l\leq i}\xi_l}\right]^+\right)$. Since this objective function is not linear, we linearize it by introducing variables $\Delta_i^+ = \left[a_i - F_{\sum_{l\leq i}\xi_l}\right]^+$ and $\Delta_i^- = \left[F_{\sum_{l\leq i}\xi_l} - a_i\right]^+$ that represent the earliness and lateness for each item. Then, the objective is minimizing the sum of the total earliness and lateness with

Table 4.1: Description of parameters and variables

		domain	description
parameter	c^+	$\mathbb{R}_+$	inventory cost for one item in inventory in unit time
	c^-	$\mathbb{R}_+$	backlog cost for one customer waiting for a unit time
	a_i	$\mathbb{R}_+$	the time of the i'th arrival of information
	ξ_i	$\{0,1\}$	indicator for arrivals of demand alongside information
	g_k	$\mathbb{R}_+$	the k'th process time
	α_i	$\{1,..,C_D\}$	the marking of the information and demand process observed at a_i
	β_k	$\{1,..,C_W\}$	the marking of the production time process observed at F_k
	M	-	a big enough value
variable	Δ_i^+	$\mathbb{R}_+$	the amount of time that product i is produced earlier than customer i arriving
	Δ_i^-	$\mathbb{R}_+$	the amount of time that product i is produced later than customer i arriving
	X_i^1	$\mathbb{Z}$	inventory position exactly after a_i
	X_i^2	$\mathbb{Z}$	inventory position exactly after F_k
	R_k	$\mathbb{R}_+$	beginning of k'th production
	F_k	$\mathbb{R}_+$	completion of k'th production
	U_i^a	$\mathbb{B}$	indicates if production starts at a_i
	U_k^f	$\mathbb{B}$	indicates if production resumes at F_k
	M_i	$\mathbb{B}$	indicates if the machine is working exactly before a_i
	S_{c_D,c_W}	$\mathbb{N}$	threshold level for the marking pair (c_D, c_W)
	$O_{i,k}^1$	$\mathbb{B}$	$a_i > F_k \rightarrow O_{i,k}^1 = 1$
	$O_{i,k}^2$	$\mathbb{B}$	$a_i < F_k \rightarrow O_{i,k}^2 = 1$
	$O_{i,k}^3$	$\mathbb{B}$	$O_{i,k}^3 = 1 \rightarrow a_i = F_k$
	$O_{i,k}^4$	$\mathbb{B}$	$a_i > R_k \rightarrow O_{i,k}^4 = 1$
	$O_{i,k}^5$	$\mathbb{B}$	$a_i < R_k \rightarrow O_{i,k}^5 = 1$
	$O_{i,k}^6$	$\mathbb{B}$	$O_{i,k}^6 = 1 \rightarrow a_i = R_k$
	O_k^7	$\mathbb{B}$	$O_k^7 = 1 \rightarrow R_k = F_{k-1}$
	O_k^8	$\mathbb{B}$	$R_k > f_{k-1} \rightarrow, O_k^8 = 1$

respect to the cost rates c^- and c^+ as given in Equation (4.1). The equations used to linearize the objective function are expressed in Equations (4.2)-(4.4).

$$\min z = \sum_i \xi_i \left(c^+ \Delta_i^+ + c^- \Delta_i^- \right) \tag{4.1}$$

subject to

$$\Delta_i^+ - \Delta_i^- = a_i - F_k \qquad \forall i,k : \sum_{l \le i} \xi_l = k \tag{4.2}$$

$$\Delta_i^+ \le O_{i,k}^1 \mathrm{M} \qquad \forall i,k : \sum_{l \le i} \xi_l = k \tag{4.3}$$

$$\Delta_i^- \le \left(1 - O_{i,k}^1\right) \mathrm{M} \qquad \forall i,k : \sum_{l \le i} \xi_l = k \tag{4.4}$$

Marking-Dependent Threshold Policy

The marking-dependent threshold policy can be enforced by using two rules. These rules make use of the inventory level and the state of the machine at each decision epoch. The first rule is that if the machine is not working at the moment of an arrival and the inventory level is less than the threshold level, production must start. Let M_i denote the status of the machine at time a_i, where $M_i = 1$ indicates that the machine has been working when ith arrival happens. Let X_i^1 denote the inventory position exactly after a_i. Let U_i^a denote the decision made at a_i, where $U_i^a = 1$ if production starts at a_i and $U_i^a = 0$ otherwise. If the inventory position exactly after a_i, is less than the threshold $X_i^1 < S_{\alpha_i}$, either the machine must be already working meaning $M_i = 1$ or it must start working meaning $U_i^a = 1$. This relation is expressed in Equation (4.5). Figure 4.2(c) illustrates this relation. Otherwise, if the inventory position is more than or equal to the threshold level, regardless of whether the machine is idle or not, a new production cannot be initiated. Equation (4.6) expresses this relation as depicted in Figure 4.2(d).

$$U_i^a + M_i \ge O_{i,k+1}^2 + O_{i,k}^1 - 2 + \frac{S_{\alpha_i,\beta_k} - X_i^1}{\mathrm{M}} \qquad \forall i,k \tag{4.5}$$

$$U_i^a \leq 2 - (O_{i,k+1}^2 + O_{i,k}^1) + 1 - \frac{X_i^1 - S_{\alpha_i,\beta_k} + 1}{\mathrm{M}} \qquad \forall i,k \tag{4.6}$$

The second rule is that after the completion of a production, if the inventory level is less than the threshold level set by the last arrival, then the production must be resumed. Let X_k^2 denote the inventory position exactly after F_k. Let U_k^f denote the decision made at F_k, where $U_k^f = 1$ means production will be resumed immediately. To enforce the second rule, the last arrival before F_k must be identified. For a pair i and k, a_i is the last arrival before F_k if $a_i < F_k < a_{i+1}$. Using the dummy variables given in Table 4.1, this can be expressed as $O_{i,k}^2 + O_{i+1,k}^1 = 2$. Hence production must continue if $X_k^2 < S_{\alpha_i,\beta_k}$ and $O_{i,k}^2 + O_{i+1,k}^1 = 2$. This relation is expressed in Equation (4.7). Figure 4.2(e) illustrates this relation. Given $O_{i,k}^2 + O_{i+1,k}^1 = 2$, if $X_k^2 \geq S_{\alpha_i,\beta_k}$, production cannot be resumed. This relation is expressed in Equation (4.8) and in Figure 4.2(f).

$$U_k^f \geq O_{i,k}^2 + O_{i+1,k}^1 - 2 + \frac{S_{\alpha_i,\beta_k} - X_k^2}{\mathrm{M}} \qquad \forall i,k \tag{4.7}$$

$$U_k^f \leq 2 - (O_{i,k}^2 + O_{i+1,k}^1) + 1 - \frac{X_k^2 - S_{\alpha_i,\beta_k} + 1}{\mathrm{M}} \qquad \forall i,k \tag{4.8}$$

The Inventory and the Machine Status

Since the control policy uses the inventory position as an input to operate, we express the inventory position as a function of the arrival and production processes. Since the inventory position at each arrival together with the production completion capture the inventory process, we define the variables X_i^1 and X_k^2 as the inventory position exactly after a_i and F_k. Hence X_i^1 can be calculated as the number of items produced before the ith arrival subtracted by $\sum_{l \leq i} \xi_i$. Let $O_{i,k}^1$ be a binary dummy variable that is 1 if $F_k \leq a_i$. Then the relation in Equation (4.9) adjusts the inventory position.

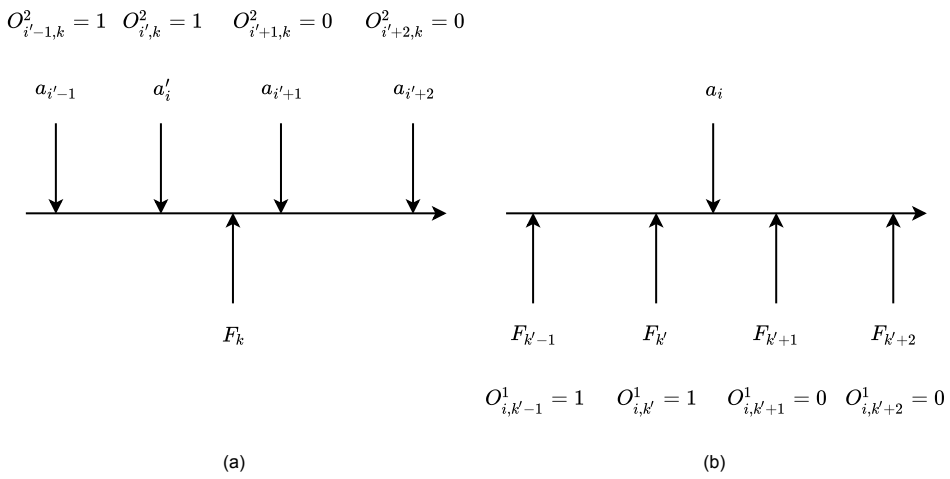

Figure 4.1: (a) Inferring the inventory position exactly after completion of a production using dummy variables $O^2_{i,k}$. (b) Inferring the inventory level exactly after an arrival using dummy variables $O^1_{i,k}$.

Similarly, with the appropriate dummy variable, X^2_k is adjusted in Equation (4.10). Figure 4.1 illustrates these equations.

$$X^1_i = \sum_k O^1_{i,k} - \sum_{l \le i} \xi_l \qquad \forall i \tag{4.9}$$

$$X^2_k = k - \sum_i \xi_i O^2_{i,k} \qquad \forall k \tag{4.10}$$

The decision to start production also depends on whether the machine is working or not. Let M_i denote the state of the machine exactly before a_i, where $M_i = 1$ indicates that the machine is already working when ith demand arrives. For the machine to be working at a_i, there must exist a k such that $R_k < a_i < F_k$ holds. Equations (4.11)-(4.12) enforce this relation. Figure 4.2(a)-(b) illustrates these relations.

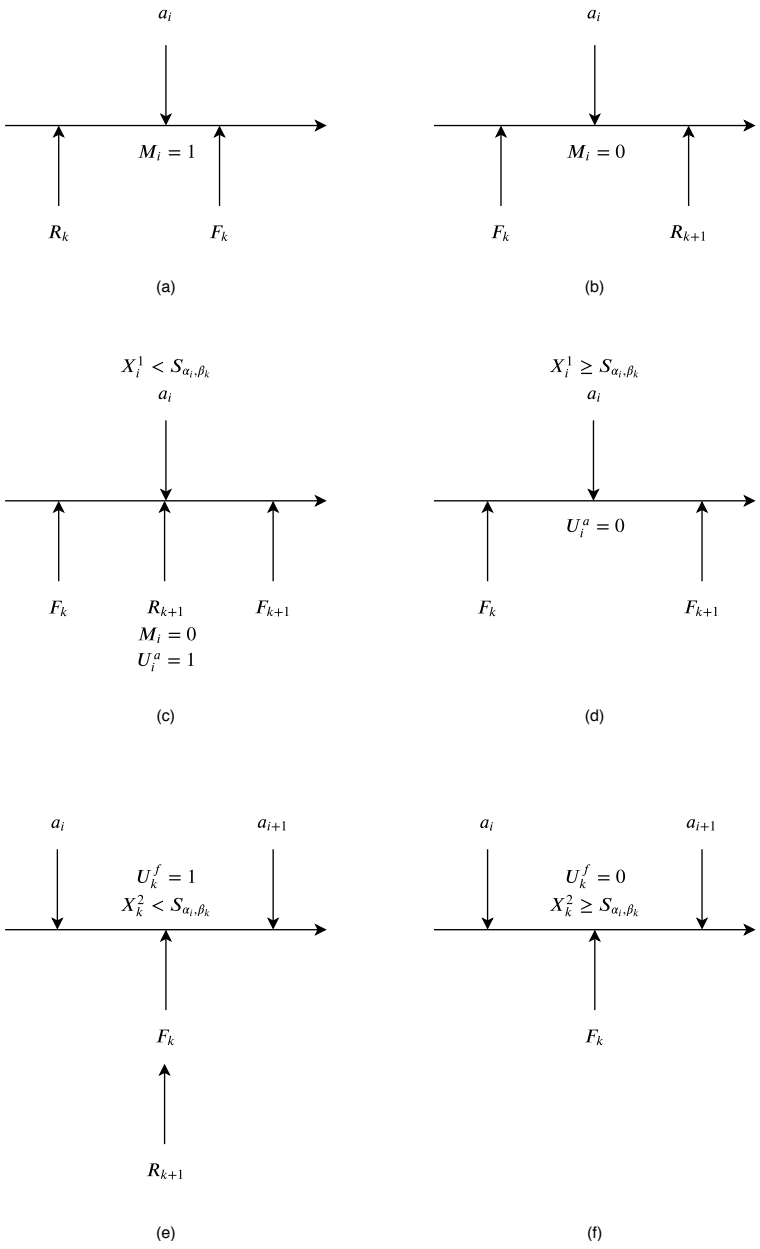

Figure 4.2: Configurations of event times and their corresponding variables and decisions.

$$M_i \geq O^2_{i,k} + O^4_{i,k} - 1 \qquad \forall i, k \tag{4.11}$$

$$M_i \leq 2 - O^1_{i,k} - O^5_{i,k+1} \qquad \forall i, k < |\mathcal{T}| \tag{4.12}$$

Other Constraints

Equation (4.13) prevents the starting of a production from appearing as if the machine has been already working. Equations (4.14) and (4.15) express that only one item can be produced at a time and item i takes g_i time units to produce. Equations (4.16)-(4.17) give the relation between the production decisions and the dummy variables. Inequalities (4.18)-(4.30) express the relation between the dummy variables and the arrival and production times a_i, R_k and F_k. Equation (4.31) ensures that every production is either triggered by the completion of the previous production or the arrival of a demand. Equation (4.32) ensures that no arrival triggers a production out of order.

$$M_i + \sum_k O^6_{i,k} \leq 1 \qquad \forall i \tag{4.13}$$

$$R_i \geq F_{i-1} \qquad \forall i > 1 \tag{4.14}$$

$$F_i = R_i + g_i \qquad \forall i \tag{4.15}$$

$$U^a_i = \sum_k O^6_{i,k} \qquad \forall i \tag{4.16}$$

$$U^f_k = O^7_{k+1} \qquad \forall k < |\mathcal{T}| \tag{4.17}$$

$$O^1_{i,k} \leq 1 + \frac{(a_i - F_k)}{\mathrm{M}} \qquad \forall i, k \tag{4.18}$$

$$O^2_{i,k} \leq 1 - \frac{(a_i - F_k)}{\mathrm{M}} \qquad \forall i,k \tag{4.19}$$

$$O^1_{i,k} \geq \frac{(a_i - F_k)}{\mathrm{M}} \qquad \forall i,k \tag{4.20}$$

$$O^2_{i,k} \geq -\frac{(a_i - F_k)}{\mathrm{M}} \qquad \forall i,k \tag{4.21}$$

$$O^1_{i,k} + O^2_{i,k} + O^3_{i,k} = 1 \qquad \forall i,k \tag{4.22}$$

$$O^4_{i,k} \leq 1 + \frac{(a_i - R_k)}{\mathrm{M}} \qquad \forall i,k \tag{4.23}$$

$$O^5_{i,k} \leq 1 - \frac{(a_i - R_k)}{\mathrm{M}} \qquad \forall i,k \tag{4.24}$$

$$O^4_{i,k} \geq \frac{(a_i - R_k)}{\mathrm{M}} \qquad \forall i,k \tag{4.25}$$

$$O^4_{i,k} \geq -\frac{(a_i - R_k)}{\mathrm{M}} \qquad \forall i,k \tag{4.26}$$

$$O^4_{i,k} + O^5_{i,k} + O^6_{i,k} = 1 \qquad \forall i,k \tag{4.27}$$

$$O^7_k \leq 1 - \frac{(R_k - f_{k-1})}{\mathrm{M}} \qquad \forall i, k > 1 \tag{4.28}$$

$$O^7_k \leq 1 + \frac{(R_k - f_{k-1})}{\mathrm{M}} \qquad \forall i, k > 1 \tag{4.29}$$

$$O^7_1 = 0 \tag{4.30}$$

$$O^7_k + \sum_i O^6_{i,k} \geq 1 \qquad \forall i \tag{4.31}$$

$$O^6_{i,k} \leq 1 - \frac{\sum\limits_{j<i,l>k} O^6_{j,l}}{\mathrm{M}} \qquad \forall i,k \tag{4.32}$$

Table 4.2 gives the number of variables and constraints in this formulation. The number of the constraints and binary variables is not affected by C_D and C_W and the main contributor to the complexity of the problem is the trace length. Since the number of variables increases with the square of the trace length, using the mathemat-

Table 4.2: Number of variables and constraints of the MIP representation of data-driven JSO.

Variables			Constraints
Continuous	Binary	Integer	
$4\|\mathcal{T}\|$	$6\|\mathcal{T}\|^2 + 5\|\mathcal{T}\|$	$2\|\mathcal{T}\| + C$	$17\|\mathcal{T}\|^2 + 6\|\mathcal{T}\| + 3\sum_i \xi_i$

ical programming formulation to determine the optimal thresholds requires significant computing power and memory. For the cases where the computing power is not sufficient to implement JSO, the discrete event simulation formulation given in Section 4.1.5 with a search algorithm can be used.

4.1.4 Cutting Plane Formulation

In the following we give an alternative formulation for the problem of determining the optimal threshold levels for the marking-dependent threshold policy through the data-driven JSO. This formulation eliminated some variables by introducing new constraints, the majority of which will be non-binding in the optimal solution. This formulation is given as a starting point for devising a cutting plane algorithm and assumes that only the demand process generates markings. The extension to the model with both the demand and production processes generating markings and the implementation of a the cutting plane algorithm based on this formulation are left for future research.

Given that in this problem enforcing that all the decisions for a given marking are identical guarantees that the inventory process follows the rules of the marking-dependent threshold policy, there is a way for enforcing these rules whiteout involving the inventory level explicitly. This approach for formulating the problem generates a large number of constraints, on the other hand avoids using big-M values in some of the constraints.

Namely this approach for formulating the problem expresses the rules of the

marking-dependent threshold policy in the following way: if two arrivals belong to the same marking of the demand process, then it is not feasible for production to start in one, if in the other one inventory level is lower and the machine is idle.

Equations (4.33)-(4.35) along with figure 4.3 explain how these constraints enforce the marking-dependent threshold policy. Hence we can omit I_i^1 and I_k^2 and constraints (4.5)-(4.10) from the formulation and add the constraints given in Equations (4.33)-(4.35). This omits the instances where the big-M value is related to the maximum difference of the threshold level and the inventory level.

The configuration depicted in Figure 4.3(a) is infeasible if $\alpha_i = \alpha_q$ and $k-i \leq r-q$, the fact that production has not resumed after F_k means that $S_{\alpha_i}\} \leq k-i$, while production starting between a_q and a_{q+1} implies that $S_{\alpha_q} > r-q$, which in turn implies $r-q < S_q = S_{\alpha_i} \leq k-i$. This constraint can be expressed as

$$O^2_{i,k}+O^1_{i+1,k}+O^8_{k+1}+O^5_{q,r+1}+O^4_{q+1,r+1} \leq 4 \qquad \forall i,k,q,r : \alpha_i = \alpha_q, i \neq q, k-i \leq r-q. \tag{4.33}$$

The configuration depicted in Figure 4.3(b) is infeasible if $\alpha_i = \alpha_q$ and $k-i \geq r-q$, that is because a_i triggering production at R_k implies that $S_{\alpha_i} \geq k-i$ while r'th production not starting at a_q given the machine is idle implies that $S_{\alpha_q} \leq r-1-q < r-q$, hence $k-i \leq S_{\alpha_i} = S_{\alpha_q} < r-q$. This constraint can be expressed as

$$O^6_{i,k} + O^1_{q,r-1} + O^5_{q,r} \leq 2 \qquad \forall i,k,q,r : \alpha_i = \alpha_q, i \neq q, k-i \geq r-q. \tag{4.34}$$

The configuration depicted in Figure 4.3(c) is infeasible if $\alpha_{i-1} = \alpha_i = \alpha_q$ and $k-i \leq r-q$, that is because a_i triggering production at R_k and $\alpha_{i-1} = \alpha_i$ together imply that $S_{\alpha_i} = k-i$ while $r+1$'th production starting at F_r implies that $S_{\alpha q} > r-q$,

Table 4.3: Number of variables and parameters of the mathematical programming formulations

	Number of variables			Number of constraints
	Binary	Integer	Continuous	
Single threshold problem	$6\|\mathcal{T}\|^2 + 2\ \|\mathcal{T}\|$	$2\|\mathcal{T}\| + 1$	$4\|\mathcal{T}\|$	$14\|\mathcal{T}\|^2 + 8\ \|\mathcal{T}\| - 1$
Multiple threshold problem	$6\|\mathcal{T}\|^2 + 2\|\mathcal{T}\|$	$3\|\mathcal{T}\|$	$4\|\mathcal{T}\| + C$	$17\|\mathcal{T}\|^2 + 10\|\mathcal{T}\| + 1$
Cutting plane formulation	$6\|\mathcal{T}\|^2 + 2\|\mathcal{T}\|$		$4\|\mathcal{T}\|$	$2\|\mathcal{T}\|^4 - 2\|\mathcal{T}\|^3 + 12\|\mathcal{T}\|^2 + 6\|\mathcal{T}\| + 1$

hence $k - i = S_{\alpha i} = S_{\alpha q} > r - q$. This constraints can be expressed as

$$O^6_{i,k} + O^2_{q,r} + O^1_{q+1,r} + O^7_{r+1} \leq 3 \qquad \forall i, k, q, r : \alpha_{i-1} = \alpha_i = \alpha_q, i \neq q, k - i \leq r - q. \tag{4.35}$$

4.1.5 Discrete Event Simulation Algorithm for Evaluating the System Performance for Given Traces

Algorithm 1 generates the inventory process of a system controlled with marking-dependent threshold policy. In this pseudo-code, $\widehat{x}$ represents variable x's value at the current time $\widehat{T}$, and x_e represents the value of x immediately after event e. Once the inventory process for a threshold matrix S has been calculated, it can be used for evaluating $S + x\mathbf{1}_{C_D,C_W}$ by adding x to P_e.

The algorithm starts with initializing the variables in line 1. Then one arrival event is placed in the future event list (FEL) in line 2. Then, while FEL is not empty, the earliest event, denoted by E, is identified in line 5. Then based on the type of this event (d for arrivals and s for productions), the last observed marking ($\widehat{\alpha}$ or $\widehat{\beta}$) is updated in lines 7-8. Then, the current threshold level $\widehat{S}$ is set based on the last observed markings $\widehat{\alpha}$ and $\widehat{\beta}$ in line 9. After the current threshold level is set, the effect

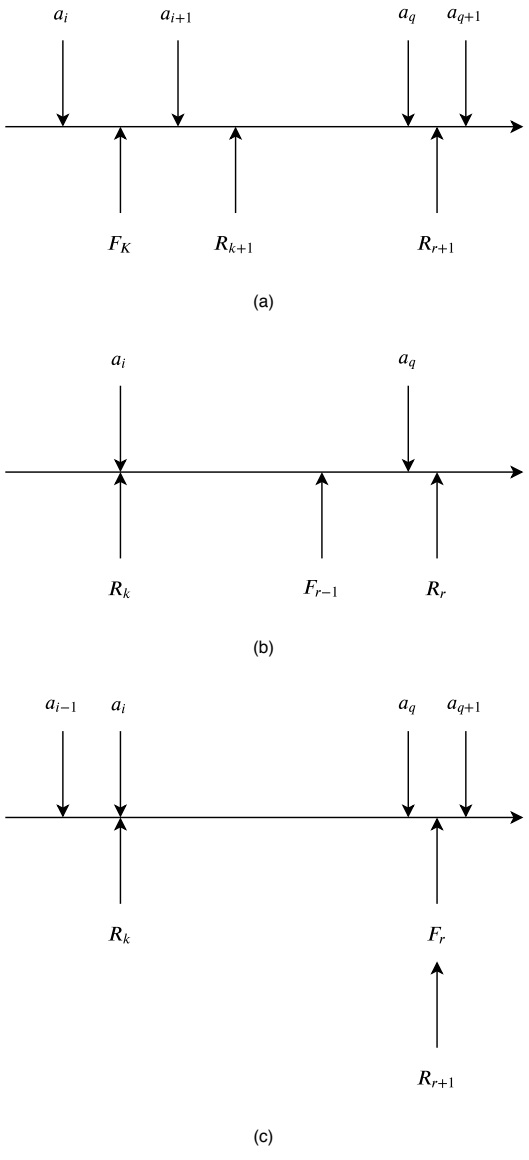

Figure 4.3: Configurations that can be infeasible under a marking-dependent threshold policy

of event E on the other system variables must be assessed.

If event E is an arrival, the inventory level will decrease if the arrival is an arrival of information and demand. This is expressed in line 11. After each arrival, the next arrival is scheduled and added to FEL. This is expressed in line 12. If the inventory level is less than the current threshold level and the machine is idle, production must start, meaning a production completion event should be added to FEL and the machine's current status WS must become *working*. This is expressed in lines 14-15.

Alternatively, if the earliest event E is a production completion event, the inventory level should increase by one, as expressed in line 19. Then, if the inventory level is less than the current threshold level, the current machine status WS should remain *working* and a new production completion must be scheduled accordingly. Otherwise, the machine must become *idle*. Lines 21-24 express these tasks. Finally, the event that has took place must be erased from FEL. This is expressed in line 27.

4.2 Analysis of the Example

In this section, we analyze the specific system introduced in Section 2.2 by using the methods presented in Chapter 3 and Section 4.1. We evaluate the system for a range of parameter values under different information scenarios: the buffer size B, the processing rate of the unreliable machine WS_2, the processing rate of the reliable machine WS_3, the number of the repair phases r, the rate of the repair phases λ, the failure probability γ, the ratio of the production rate to the arrival rate μ_1/TP, and the ratio of the backlog cost to inventory cost c^-/c^+ are varied as shown in Table 4.4. 5576 different cases are used to analyze the system in four different information availability scenarios. In this experimental setup, the information about the demand is only recorded at the arrival instances.

In this system, $qr + 2r + q + 3$ states are required to model the information and demand processes. Therefore, the optimal state-dependent threshold policy that uses

Algorithm 1 DES algorithm

$i \leftarrow 1$, $k \leftarrow 1$, $e \leftarrow 1$, $T_e \leftarrow 0$, $X_e \leftarrow 0$, $e \leftarrow e+1$, WS $\leftarrow$ idle, $\widehat{T} \leftarrow 0$
$FEL \leftarrow \left\{\left(\mathrm{d}, \widehat{T} + \tau_i, \alpha_i, \xi_i, \phi\right)\right\}$, $i \leftarrow i+1$
$\widehat{\beta} \leftarrow 1$
while $|FEL| > 0$ **do**
  $\widehat{E} \leftarrow E : E_2 < E_2' \forall E, E' \in FEL$
  $\widehat{T} \leftarrow \widehat{E}_2$, $T_e \leftarrow \widehat{T}$
  if $E_1 = \mathrm{d}$ **then** $\widehat{\alpha} \leftarrow E_3$ **end if**
  if $E_1 = \mathrm{s}$ **then** $\widehat{\beta} \leftarrow E_5$ **end if**
  $\widehat{S} \leftarrow S_{\widehat{\alpha},\widehat{\beta}}$
  if $E_1 = \mathrm{d}$ **then**
    $X_e \leftarrow X_{e-1} - \widehat{E}_4$
    if $i \leq |\mathcal{T}|$ **then** $FEL \leftarrow FEL \cup \left\{\left(\mathrm{d}, \widehat{T} + \tau_i, \alpha_i, \xi_i, \phi\right)\right\}$, $i \leftarrow i+1$ **end if**
    if WS $=$ idle, $X_e < \widehat{S}$ **then**
      WS $\leftarrow$ working
      if $k \leq |\mathcal{T}|$ **then** $FEL \leftarrow FEL \cup \left\{\left(\mathrm{s}, \widehat{T} + g_k, \phi, \phi, \beta_k\right)\right\}$, $k \leftarrow k+1$
 end if
    end if
  end if
  if $E_1 = \mathrm{s}$ **then**
    $X_e \leftarrow X_{e-1} + 1$
    if $X_e < \widehat{S}$ **then**
      WS $\leftarrow$ working
      if $k \leq |\mathcal{T}|$ **then** $FEL \leftarrow FEL \cup \left\{\left(\mathrm{s}, \widehat{T} + g_k, \phi, \phi, \beta_k\right)\right\}$, $k \leftarrow k+1$
 end if
    else
      WS $\leftarrow$ idle
    end if
  end if
  $FEL \leftarrow FEL \backslash \widehat{E}$
  $e \leftarrow e+1$
end while

all the information will have up to 36 thresholds for 36 states. Even for the case where all the states are fully observable, determining all the optimal thresholds by using the information traces collected in real time is not computationally tractable.

Table 4.4: The range of parameters used in the numerical experiments

Buffer size q	$\{1, 2, 3\}$
The processing rate of the unreliable machine μ_2	$\{1, 2, 3\}$
The processing rate of the reliable machine μ_3	$\{1, 2, 3\}$
The number of the repair phases r	$\{2, 4, 6\}$
The rate of each repair phase λ	$\{0.5, 0.7, 0.9\}$
Breakdown probability γ	$\{0.01, 0.05, 0.09\}$
The production rate to throughput ratio μ_1/TP	$\{1.5, 2, 2.5\}$
The backlog cost rate to inventory cost rate ratio c^-/c^+	$\{2, 3, 4, 5\}$

4.2.1 Evaluation Methodology

In our evaluation methodology, we first use the analytical method presented in Chapter 3 to analyze the production/inventory system with the given production and demand processes and with the given markings selected for control. We determine the optimal marking-dependent thresholds and the optimal cost by using the MIP formulation given in Section 3.0.3.

In order to compare the performance of the data-driven approach with the exact evaluation, we generate different-length traces for the given system parameters and the selected markings. We then use the JSO approach described in Section 4.1.3 for short traces and the DES approach given in Section 4.1.5 with a search algorithm for long traces. We denote the set of the thresholds determined by the JSO formulation given the information scenario specified by a and b and trace length $|\mathcal{T}|$ as $S^{\mathrm{JSO(a,b,|\mathcal{T}|)}}$.

As another benchmark, we use the same demand and information arrival trace to fit an exponential or a phase type distribution that matches the first three moments. We set a threshold based on these fitted distribution using the analytical methods. We denote the thresholds determined after fitting an exponential distribution and a phase-type distribution to a trace of length $|\mathcal{T}|$ as $S^{\mathrm{EXP}(|\mathcal{T}|)}$ and $S^{\mathrm{PH}(|\mathcal{T}|)}$ respectively. Figure 4.4 illustrates the evaluation process.

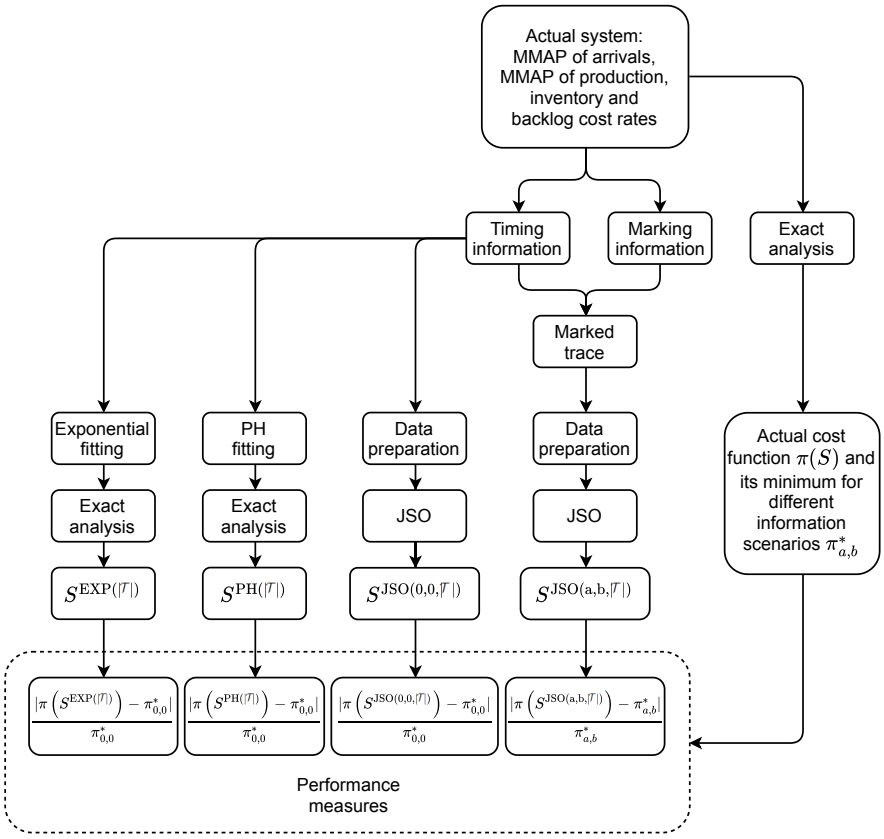

Figure 4.4: Scheme of the performance evaluation for the control methods.

4.2.2 Effect of Marking Selection on the Performance of the Production/Inventory System

Deciding on the markings to be used in the marking-dependent threshold policy in a right way improves the performance of the control policy. In this section, we investigate the following questions: *what will be the performance if only one marking is to be used; should we use the buffer status or the production status as the selected marking?; and what is the additional benefit of using both of the markings?*

In order to investigate these questions, each case is analyzed by using the exact method with the corresponding MMAP representations of the processes depending on the number of markings for the given parameters. When both the buffer status and the machine status are used, the marking-dependent threshold policy uses four thresholds to control the system. If only the buffer status or the machine status is used as a marking, the control policy uses two thresholds. If these markings are not used and the controller only uses the demand inter-arrival times, then only one threshold is used. The average cost of the system under an information scenario with the optimal thresholds is denoted by $\pi^*_{a,b}$ where $a \in \{0,1\}$ indicates whether the production marking is used ($a = 1$) or not ($a = 0$), and $b \in \{0,1\}$ indicates whether the buffer status marking is used ($b = 1$) or not ($b = 0$). For each case, after the optimal thresholds are determined, the minimum cost $\pi^*_{a,b}$ has been compared to the cost obtained when both of the information sources are used by the controller $\pi^*_{1,1}$.

Table 4.5 shows the results of these experiments. The results indicate that the marginal contribution of using both markings is lower than the contribution of using only the production status marking on average. As the expected repair time increases, the number of repair phases increases, and the breakdowns become more frequent, the contribution of using the production marking is higher than the contribution of using the buffer marking. In these cases, the effect of the breakdowns become more pronounced for the system and the disparity between the demand and production rate increases the value of information. If the production rate is much higher than the demand rate, it will be able to react to trigger or halt production depending on the markings in a faster way. Large backlog costs also decrease the value of information, as it becomes less attractive to carry backlogs based on the estimation that the demand will temporarily be low.

Table 4.6 shows the effect of the parameters on the optimal threshold level for each marking when both the buffer status and the production status are used as markings.

In this case, the control policy uses 4 thresholds corresponding to 4 different marking pairs. On average, the threshold for the marking *machine under repair and buffer empty* is the lowest and the threshold for the marking *machine in working condition and buffer not empty* is the largest. As the number of repair phases increases, the coefficient of variation of the repair time decreases. As a result, all thresholds decrease. Similarly, as the breakdowns become more frequent, all the threshold levels increase to protect the system from the downtime. As the backlog cost increases compared to the inventory cost and as the production becomes slower compared to the demand, all thresholds increase. Furthermore, the threshold for the marking *machine in working condition and buffer not empty* is more sensitive to these parameters as opposed to the parameters related solely to the demand process. Overall, the threshold for the marking *machine under repair and buffer empty* is the most sensitive threshold to the parameters of the demand process.

4.2.3 Performance of the Data-Driven Method

In order to capture the effect of the trace length, we conduct the experiments for traces that have 100, 1000, 10000 observed event arrivals. We use the JSO to analyze the system for a total of 66912 cases depending on the range of parameters, informational settings, and trace length. For each case, 5 traces were generated and analyzed by the JSO approach.

Table 4.7 reports the performance of the JSO approach with the given trace lengths compared to the optimal analytical solution for the case where both sources of information are used to control the system. The percentage deviation of the costs obtained by using the JSO compared to the analytical solution, $\frac{\left|\pi(S^{\mathrm{JSO(a,b,|\mathcal{T}|)}})-\pi^*_{1,1}\right|}{\pi^*_{1,1}}$ is reported for each case.

Figure 4.5 depicts the average costs obtained with the JSO approach with different levels of information and different trace lengths $\pi(S^{\mathrm{JSO(a,b,|\mathcal{T}|)}})$. The results show that

Table 4.5: Percentage difference of average cost using optimal analytical solution with one or both of the sources of information with respect to using the analytical solution with none of the sources of information: $\frac{\left|\pi^*_{a,b}-\pi^*_{0,0}\right|}{\pi^*_{0,0}}$

		Available Information		
	Parameter Level	Only buffer status (two thresholds) $\frac{\lvert\pi^*_{0,1}-\pi^*_{0,0}\rvert}{\pi^*_{0,0}}$	Only machine status (two thresholds) $\frac{\lvert\pi^*_{1,0}-\pi^*_{0,0}\rvert}{\pi^*_{0,0}}$	Both the machine and buffer statuses (four thresholds) $\frac{\lvert\pi^*_{1,1}-\pi^*_{0,0}\rvert}{\pi^*_{0,0}}$
Buffer size q	1	2.14	4.57	5.46
	2	2.81	4.24	5.68
	3	3.10	3.89	5.72
Rate of unreliable machine μ_2	1	2.39	3.23	5.10
	2	2.78	4.61	6.00
	3	2.81	4.98	5.78
Rate of reliable machine μ_3	1	2.15	3.07	3.89
	2	3.20	5.13	6.84
	3	2.94	5.41	7.38
Number of repair phases r	2	2.04	2.43	3.80
	4	2.85	4.98	6.35
	6	3.67	7.05	8.39
Rate of repair phases λ	0.5	3.18	5.65	7.01
	0.7	2.61	4.17	5.53
	0.9	2.33	3.37	4.73
Breakdown probability γ	0.01	2.05	2.56	4.01
	0.05	3.07	5.29	6.68
	0.09	3.05	5.47	6.67
Service rate to throughput rate ratio $\frac{\mu_1}{TP}$	1.5	0.94	1.31	1.73
	2	2.40	3.85	5.03
	2.5	3.68	5.97	7.93
Backlog cost rate to inventory cost rate ratio $\frac{c^-}{c^+}$	2	2.53	5.33	6.56
	3	2.51	4.04	5.54
	4	2.57	3.69	5.03
	5	3.07	3.73	5.13
Average		2.66	4.25	5.61

as the trace length increases, the cost obtained with the JSO approach gets closer to the optimal cost obtained with the analytical approach under each information scenario. That is, the JSO approach makes use of the additional information to reduce cost. However, the selection of the markings to be used in the marking-dependent threshold policy has a significant effect on the performance. When no additional markings are used and the control only uses the unmarked trace that includes the demand inter-arrival times, the performance of the JSO with the longest trace is worse than the analytical solution for the case both the production and inventory status are

Table 4.6: Average threshold levels for the markings

		Marking			
	Parameter Level	machine under repair and buffer empty	machine in working condition and buffer empty	machine under repair and buffer not empty	machine in working condition and buffer not empty
q	1	0.75	1.81	1.18	2.29
	2	0.75	1.73	1.28	2.32
	3	0.73	1.68	1.37	2.33
μ_2	1	0.81	1.67	1.33	2.51
	2	0.70	1.74	1.24	2.24
	3	0.72	1.82	1.25	2.17
μ_3	1	0.90	1.83	1.33	2.20
	2	0.62	1.70	1.24	2.35
	3	0.60	1.63	1.21	2.47
r	2	0.97	1.83	1.47	2.43
	4	0.63	1.71	1.14	2.25
	6	0.42	1.61	1.04	2.13
λ	0.5	0.57	1.71	1.19	2.25
	0.7	0.76	1.76	1.28	2.34
	0.9	0.84	1.75	1.33	2.33
γ	0.01	0.49	1.50	1.15	2.00
	0.05	0.77	1.79	1.29	2.38
	0.09	1.06	2.03	1.42	2.67
$\frac{\mu_1}{TP}$	1.5	2.09	3.08	2.14	3.62
	2	0.60	1.66	1.10	2.23
	2.5	0.22	1.18	1.00	1.75
$\frac{c^-}{c^+}$	2	0.31	1.32	1.08	1.81
	3	0.64	1.67	1.20	2.25
	4	0.94	1.94	1.37	2.52
	5	1.20	2.16	1.50	2.79
Average		0.74	1.74	1.27	2.31

used as markings in addition to the inter-arrival times.

As the demand arrival pattern is interrupted by breakdowns due to longer down times, the information about the machine is expected to become more valuable. The results confirm that decreasing the repair rate has an adverse effect on the performance of the JSO when the production status information is not used in the markings.

Table 4.7: Percentage difference of the average cost using JSO with different information levels with respect to the optimal analytical solution using both sources of information given different trace lengths

		Available Information											
		Unmarked trace $\frac{\lvert\pi(S^{\text{JSO}(0,0,\lvert\mathcal{T}\rvert)})-\pi^*_{1,1}\rvert}{\pi^*_{1,1}}$ (one threshold)			Trace with buffer status only $\frac{\lvert\pi(S^{\text{JSO}(0,1,\lvert\mathcal{T}\rvert)})-\pi^*_{1,1}\rvert}{\pi^*_{1,1}}$ (two thresholds)			Trace with machine status only $\frac{\lvert\pi(S^{\text{JSO}(1,0,\lvert\mathcal{T}\rvert)})-\pi^*_{1,1}\rvert}{\pi^*_{1,1}}$ (two thresholds)			Trace with buffer and machine status $\frac{\lvert\pi(S^{\text{JSO}(1,1,\lvert\mathcal{T}\rvert)})-\pi^*_{1,1}\rvert}{\pi^*_{1,1}}$ (four thresholds)		
		Trace Length			Trace Length			Trace Length			Trace Length		
		100	1000	10000	100	1000	10000	100	1000	10000	100	1000	10000
q	1	14.67	6.44	5.58	14.38	5.47	3.56	11.16	2.74	0.99	11.25	2.31	0.32
	2	13.18	6.96	6.08	12.23	5.08	3.40	10.48	3.24	1.53	10.18	2.43	0.37
	3	11.81	6.99	6.19	12.32	4.93	3.20	11.20	3.59	1.88	10.40	2.37	0.38
μ_2	1	12.27	6.40	5.33	11.70	4.87	3.01	10.52	3.71	1.88	9.67	2.43	0.34
	2	13.25	7.23	6.48	12.79	5.47	3.72	10.33	2.91	1.29	10.14	2.31	0.38
	3	15.01	7.10	6.29	15.30	5.32	3.43	12.48	2.64	0.89	12.65	2.42	0.40
μ_3	1	9.94	4.86	4.19	11.11	3.83	2.06	9.75	2.67	1.04	9.75	2.31	0.37
	2	19.64	7.82	6.81	14.57	5.70	3.92	12.48	3.23	1.62	11.99	2.25	0.37
	3	15.47	7.85	6.76	14.22	6.15	4.45	11.27	3.32	1.60	10.56	2.34	0.33
r	2	13.30	4.86	4.09	10.91	3.62	2.11	10.36	3.11	1.61	9.89	2.12	0.36
	4	11.98	7.40	6.34	14.51	5.91	3.76	11.79	3.48	1.31	11.65	2.71	0.36
	6	13.03	8.78	7.86	14.17	6.92	4.74	10.91	2.93	1.13	10.39	2.50	0.37
λ	0.5	15.50	8.17	7.14	14.47	6.14	4.27	11.56	3.11	1.21	11.37	2.52	0.35
	0.7	13.00	6.80	5.93	13.61	5.30	3.45	11.41	3.09	1.38	11.06	2.36	0.37
	0.9	11.34	5.96	5.10	11.83	4.74	2.85	10.27	3.22	1.47	9.82	2.32	0.36
γ	0.01	10.46	4.42	4.04	11.12	3.56	2.27	9.49	2.65	1.59	9.09	1.65	0.29
	0.05	16.12	8.34	7.32	14.10	6.20	4.23	11.78	3.17	1.36	11.28	2.53	0.40
	0.09	13.25	8.17	6.82	14.70	6.42	4.07	11.97	3.60	1.10	11.88	3.03	0.39
$\frac{\mu_1}{TP}$	1.5	16.54	3.95	2.23	15.78	4.61	1.44	14.78	3.75	0.83	14.97	3.55	0.61
	2	11.05	6.76	6.13	12.11	5.04	3.53	9.97	2.71	1.21	9.46	2.18	0.30
	2.5	12.24	10.22	9.82	12.03	6.53	5.60	8.49	2.95	2.02	7.82	1.48	0.18
$\frac{c^-}{c^+}$	2	12.92	7.88	7.19	12.77	6.12	4.60	9.43	2.67	1.27	9.37	1.95	0.24
	3	13.38	7.06	6.21	13.09	5.39	3.73	10.94	3.12	1.42	10.37	2.33	0.34
	4	13.27	6.48	5.45	13.41	4.82	3.06	11.43	3.18	1.35	11.06	2.43	0.41
	5	13.55	6.48	5.39	13.96	5.24	2.70	12.52	3.57	1.37	12.21	2.90	0.47
Average		13.28	6.98	6.06	13.31	5.39	3.52	11.08	3.14	1.35	10.75	2.40	0.36

Figure 4.6 compares the performance of JSO to the methods where the analytical method is used with the parameters estimated from the unmarked traces. The percentage loss compared to the solution with the analytical method that uses the

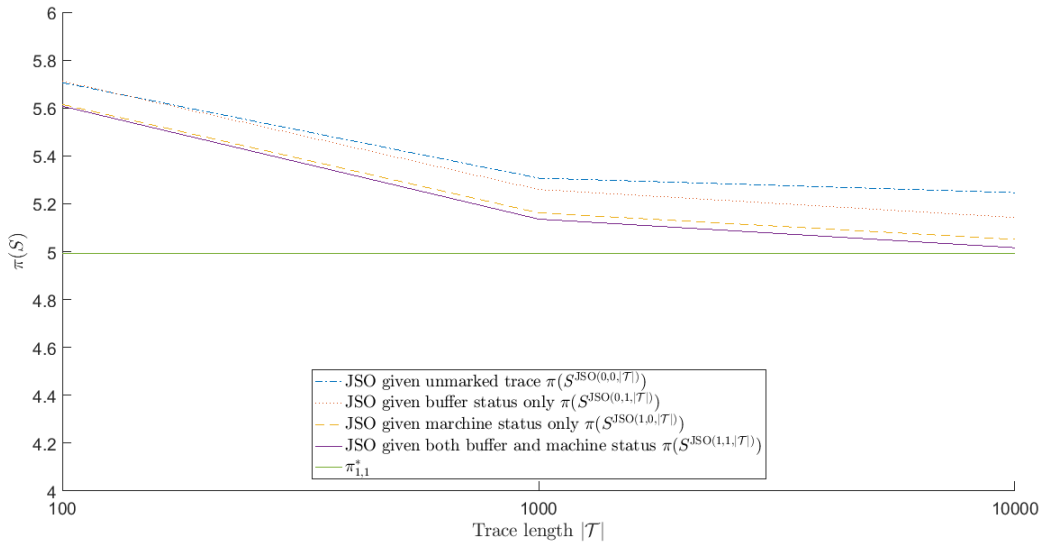

Figure 4.5: The average cost of using JSO with different trace-lengths and information levels compared to the average optimal analytical cost with both sources of information.

unmarked traces $\frac{|\pi(S)-\pi^*_{0,0}|}{\pi^*_{0,0}}$ is used for this comparison. With the assumption that the process times are not correlated, twenty bootstrap samples of the process times have been used in data preparation for the JSO. It is observed that using the analytical method with the fitted phase-type distribution yields results close to the JSO. When the trace length is sufficiently long, both methods yield the same results. However, when the trace length is short, parameter fitting methods cannot capture the distributions and therefore the performance is worse than the performance of the JSO approach. The average deviation with respect to the optimal analytical solution for the methods were 2.66% for JSO, 3.16% for exponential fitting and 4.15% for phase-type fitting. Fitting an exponential distribution that captures only the first moment outperforms the results obtained for the case where a phase-type distribution that matches the first three moments when the trace length is short.

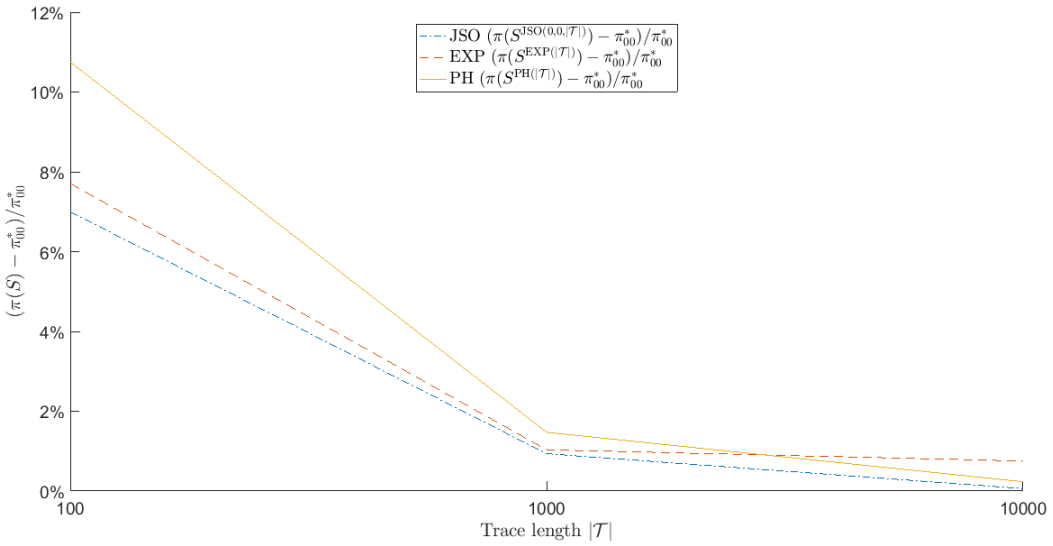

Figure 4.6: Percentage difference of average cost using exponential and phase-type fitting methods and JSO with respect to the optimal analytical solution given unmarked traces.

Table 4.7 shows the effect of the parameters and information scenarios on the performance of data-driven JSO. It is worth noting that many of the patterns observed in Table 4.5 are present for the data-driven JSO with traces of 1000 and 10000 length. However, for short traces these patterns are not present. This indicates that with very short traces, having more information to use and having more thresholds to set might not translate to a better control of the system due to the difficulty of estimating the control policy parameters.

The results from Figures 4.5, 4.6 and Table 4.7 show that the marking-dependent threshold policy implemented with the JSO approach as a real-time control policy is an effective way of matching supply and demand.

4.3 Input data preparation

Depending on the situation, we may have different levels of information about the system at hand, for each source of uncertainty distributional information may be available or only a marked or unmarked trace may be at hand, furthermore the source might have i.i.d. or correlated inter-event times. We look at the situation where for the arrivals and the processing times either a stochastic process (MAP or MMAP) is available or only a trace. In case of unmarked traces there is prior knowledge if each trace has i.i.d. or correlated inter-event times.

In each scenario, we need replications of the traces to use in joint simulation and optimization. This is straight forward in the case that we have the MAP or MMAP representation for a process, however if we have a trace we use bootstrap samples of that trace only if it is not correlated. The bootstrap method was introduced in Efron (1992) for reducing the variance of an estimator in machine-learning. This method is based on the idea that given independence between the available instances of the data, a random selection of these data points can act as a data set from the same source.

If for both the arrival times and the process times a correlated trace is available, we cannot make more replications of each trace individually, but given that the two processes are independent we can shift them along each other and build a number of shorter replications. Figure 4.7 depicts this. In the case where a marked trace is at hand, we can not exchange two inter-arrival times if they are located between two different markings, in addition we cannot let a bootstrap sample change the number or order of marking(Because this imposes additional assumptions about the stochastic process generating the marked trace). In addition, inside a segment of the trace that the marking is not changing (we refer to this simply as a "segment" for simplicity), autocorrelation might be present or not. Hence, first we break down the trace into segments, then if the autocorrelation in the segments of a marking are below a certain

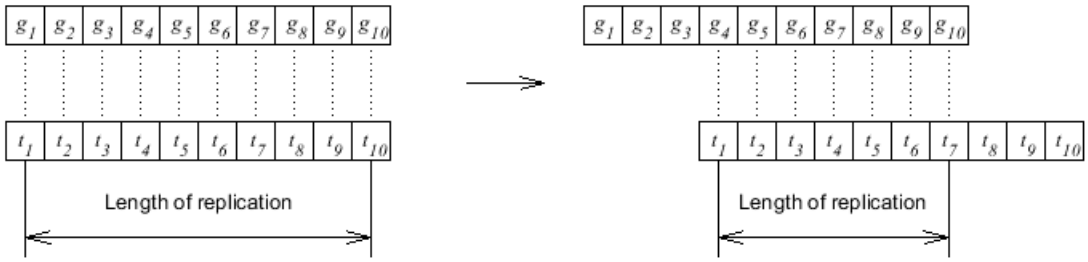

Figure 4.7: Shifting two correlated traces alongside each other to build new replication.

threshold we change these segments with bootstrap samples from the same marking, leaving the order and the number of the markings intact. We refer to a trace generated in such a manner as a Marked bootstrap sample. Table 4.8 summarizes this way of building enough replications for joint simulation and optimization.

Table 4.8: Making more replications given different levels of information

Arrivals				Process Times			
				i.i.d		correlated	
				data	stochastic process	data	stochastic process
	Single marking	i.i.d	data	use bootstrap samples for t and g	use bootstrap samples for t and generate g	permute t	use bootstrap samples for t and generate g
			stochastic process	generate t and use bootstrap samples for g	-	generate t	-
		correlated	data	permute g	generate g	shift t and g	generate g
			stochastic process	generate t and use bootstrap samples for g	-	generate t	-
	Multiple marking		data	use marked bootstrap samples for t and bootstrap samples	use marked bootstrap samples for t and generate g	use marked bootstrap samples for t, if not possible, shift t and g	use marked bootstrap samples for t and generate g
			stochastic process	generate t and use bootstrap samples for g	-	generate t	-

Chapter 5

SUPERVISED LEARNING FOR SELECTION OF INFORMATION SOURCES AND FORMING THE MARKINGS FOR A MARKING-DEPENDENT POLICY

Given the extensive data available for control of manufacturing systems, there is a need for developing methods that are capable of identifying the most relevant information to control a given system. The choice of sources of information to be used influences the performance of the system. Given the sources of information, the parameters of the control policy also need to be optimized. In Chapters 3-4 we considered the analytical analysis and data-driven control for a simple model for given information sources. In this Chapter, we generalize the approach to selection of information sources and forming the markings. 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 sources of information,

(2) forming the markings and

(3) determining the optimal marking-dependent policy.

We pose the first two problems as regression problems and use machine learning approaches. In the absence of sufficient data collected from a production system machine learning approaches allow using evaluations of the system with synthetic data obtained from simulation together with the limited available data. In addition, given the large computational effort required for optimizing complicated systems, machine learning can speed up optimization procedures.

We first formulate these problems as regression problems and then, we show how machine learning can facilitate the optimization of manufacturing systems, using numerical experiments.

In our experimental setting, the first system we analyze is a production/inventory system where summarizing the partial information signals emitted from the demand process is non-trivial. The second system we analyze is a production system where the goal is finding the best dispatching policy using few sources of information.

Despite the advances in the information collection technology in manufacturing settings and the extensive literature on modeling and control of manufacturing systems, data-driven control of production systems still faces challenges in practice. Among these challenges is the mismatch between the amount of information available on the shop-floor and the amount of information needed for an effective control policy for the system. On the 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 inter-event times in 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. In practice, relatively few actual systems will fall purely on the extremes of this spectrum. However, solutions for these cases can be used for building solutions for the mixed cases.

In this work, we develop solutions for the latter case, using machine learning tools. Although modeling tools and control policies for manufacturing systems have been studied extensively, for very complex manufacturing systems, often the state of the art optimization tools are not capable of modeling and optimizing the systems in practice. Even in smaller models, incorporating partial information into the control policy can complicate the modeling and optimization of the system. In Chapter 3, we have proposed using the marking-dependent threshold policy for a production

Table 5.1: Parameters for the system that analyzed in 5.2. In this system, The processing times and repair times are exponential random variables.

	Type	Information			
		Size	Processing rate	Breakdown Probability	Repair rate
E_1	Machine		1.7713	0.12	0.1396
E_2	Buffer	1	-	-	-
E_3	Machine		1.0208	0.19	0.2521
E_4	Buffer	4	-	-	-
E_5	Machine	-	1.6336	0.07	0.1338
E_6	Buffer	∞	-	-	-
E_7	Machine	-	1.0304	-	-
E_8	Buffer	Determined based on control parameters	-	-	-

system with correlated partially observable processes. One of the main contributors to the complexity and computational burden of using the marking-dependent threshold policy is the number of markings and consequently the number of thresholds. In addition, our numerical experiments indicate that increasing the number of thresholds has diminishing returns.

Figure 5.1 depicts an example of such diminishing returns. Figure 5.2 depicts this system and Table 5.1 gives its parameters that have been chosen randomly. Different instances of this system with random parameters were generated for numerical experiments, but only the instance with the parameters given in Table 5.1 has been used for the results given here . For this model, at most 11 markings could have been used but due to computational limitations the figure depicts up to 4 thresholds. Here for each given number of thresholds, 100 random formations of markings have been evaluated and the minimum has been chosen, except for the case with one threshold, where only one formation of markings is possible. For this reason, the formation of markings has to be done with a given number of desired thresholds in mind.

In addition to the computational burden and the diminishing returns, suggesting

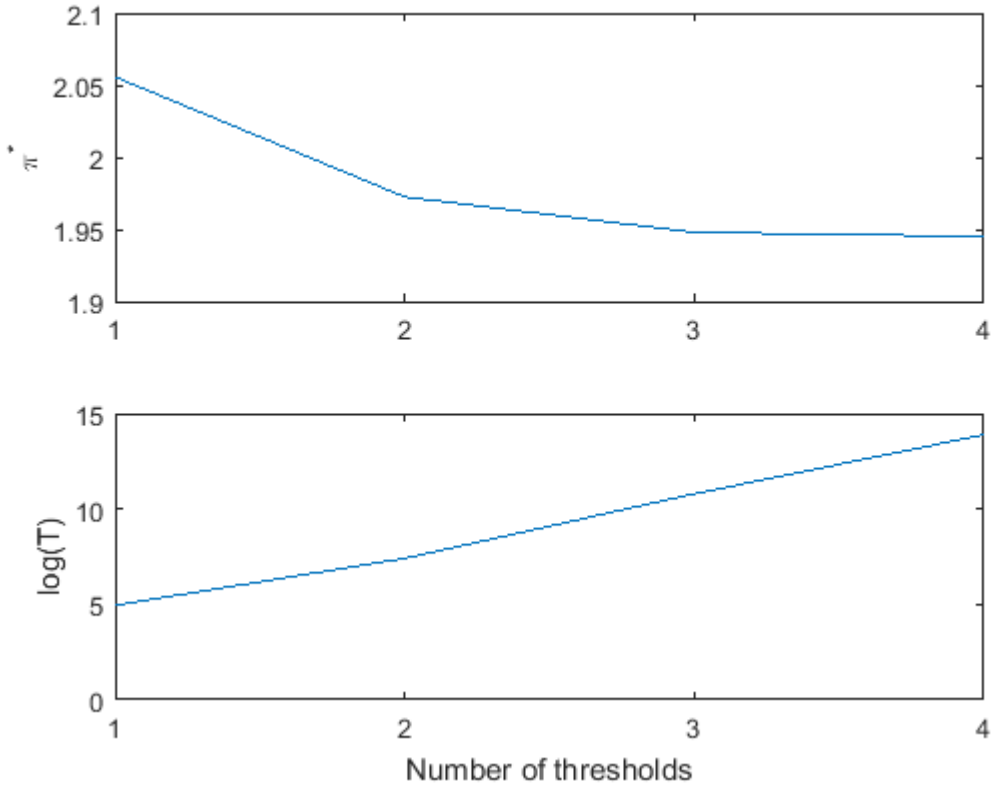

Figure 5.1: Using more thresholds has diminishing returns while the computational burden of solving the problem increases rapidly (T denotes the total computation time in seconds).

a very complicated policy for control of a complicated system might lead to difficulties in implementation. To remedy these issues, it is possible to approximate the behavior of the system with a simpler model and/or by using a simpler control policy. In both cases, the simplification comes at a cost. Although the losses might not be quantifiable, it is desirable to perform the simplification in a way that minimizes the losses due to modelling errors and using simpler policies. We use machine learning as a tool for achieving this goal.

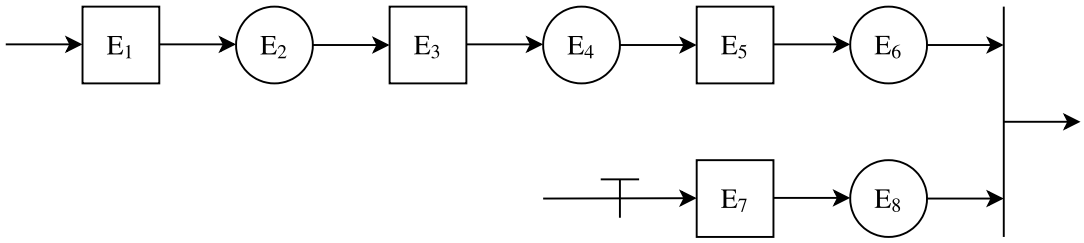

Figure 5.2: System set-up for the experiments related to diminishing returns of using more thresholds

5.1 Literature Review

5.1.1 Machine Learning for Control of Operations and Processes

Machine learning has been used for controlling operations and processes in manufacturing (Monostori et al., 1996; Irani et al., 1993; Charest et al., 2018). It has also been used for controlling the manufacturing systems on the planning level (Priore et al., 2001). Can and Heavey (2012) investigate building meta models of 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.

5.1.2 Machine Learning for Performance Evaluation

Many works in the literature aim at predicting a performance measure for each item that is being produced. Lingitz et al. (2018) use different machine learning methods for lead time prediction for a semiconductor producer. Alenezi et al. (2008) aim at predicting 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 availabilities) using support vector regression. Karaoglan and Karademir (2017) use ANN 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.

5.1.3 Machine Learning for Control of Production Systems

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 problem of dispatching 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).

5.1.4 Machine Learning Methods

In the following we briefly review the machine learning methods that we have used in the numerical experiments.

Initially inspired by the nervous system of animals, artificial neural networks 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. They have proved exceptionally powerful for some machine learning applications such as image recognition and are very capable in estimating highly non-linear functions.

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

Linear regression is a widely used regression model that fits a linear model to the data by minimizing the least square error over the training set. Given that linear regression, on its own, is not capable of dealing with non-linear relations, in-order for it to fit such functions, nonlinear transformations of the features have to be fed to it. However, given that there are many different ways for such transformations, this has to be done on the basis of the specific application.

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.

This study contributes to the literature in the following ways. The study gives the formulation for the problem of controlling a production system by using the marking-dependent policy as two supervised learning problems. The first problem is selection of sources of information and the second problem is formation of markings. Next,

through numerical experiments we show the advantages of using machine learning in the process of optimization referred to as learning while optimizing. Also, we give a comparison of different machine learning methods and their application to the two levels of problems for finding the best marking-dependent policy.

5.2 Problem Description

In the following, we give the description of the system under study, the marking-dependent policy and the optimization problem for determining the best parameters for the marking-dependent policy.

5.2.1 Description of a Production Plant

A production plant consists of n elements. Each element $\mathrm{E}_i, i \in \{1, \ldots, n\}$ is a work station 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. Let $\eta = (\alpha_1, \ldots, \alpha_n)$ denote the state of the system, where $\alpha_i \in A_i$, A_i being the finite set of possible states for element E_i. The possible states for a buffer are the number of items in the buffer $A_i = \{0, 1, \ldots\}$. For a reliable machine with one working state the possible states include $A_i = \{\text{working}, \text{blocked}, \text{starved}\}$ and for a machine with m working states the possible states include

$$A_i = \{\mathrm{WS}_1, \ldots, \mathrm{WS}_m, \text{blocked}, \text{starved}\},$$

where WS_i refers to the i'th working state. Similar to models where breakdowns are represented by long processing times, an unreliable machine can be modeled by a long working state that corresponds to the repair process. This state is only visited upon a breakdown.

5.2.2 Description of 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 $C(\mathbf{I})$ denote the total cost of collecting information from these sources, where $\mathbf{I} = (I_1, \ldots, I_n)$. Then let $\hat{\eta} = (\hat{\eta}_1, \ldots, \hat{\eta}_n), \hat{\eta}_i = \begin{cases} \eta_i & \text{if} \quad \mathbf{I}_i = 1 \\ - & \text{if} \quad \mathbf{I}_i = 0 \end{cases}$ denote the state of the system based on the partial observations of system state.

5.2.3 Description of markings

The set of possible partial observations of the state of the system ($\hat{\eta}$) can be mapped to a finite set of labels $\{1, \ldots, K\}$ referred to as markings. 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 and let $\mathbf{Y} = \{y_{i,k}\}$ denote the matrix that specifies a given formation of markings.

5.2.4 Description of the 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 χ_i denote an indicator that shows if the release into E_i can be controlled. And let $\mathbf{u} = (u_1, \ldots, u_n)$ denote the control vector in the system where $u_i \in \{0,1\}$ denotes the decision to release material into E_i or not.

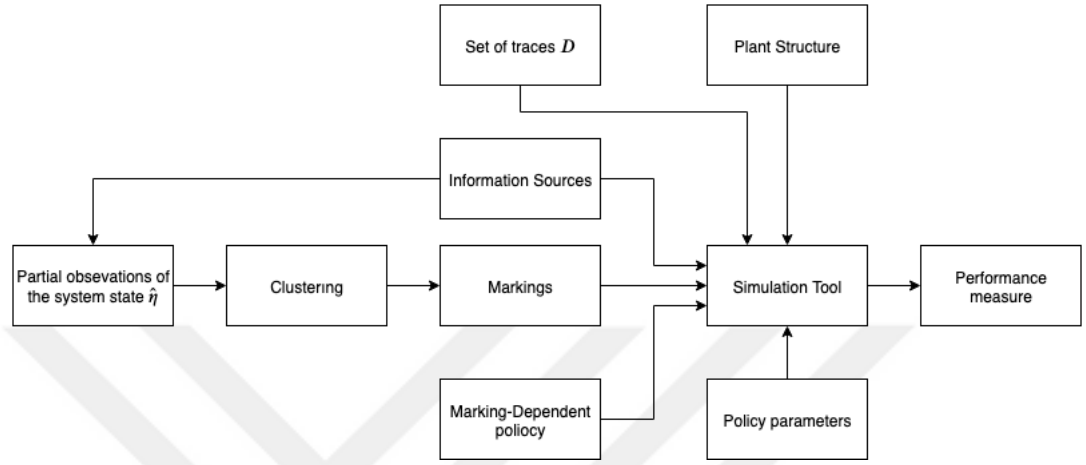

Figure 5.3: Evaluation of the production system

5.2.5 Description of policy parameters

The parameters of the marking-dependent policy are K vectors $\mathbf{u}_1, \ldots, \mathbf{u}_K$ indicating the decision for each element and marking. Let

$$\mathbf{U} = \begin{bmatrix} \mathbf{u}_1 \\ \vdots \\ \mathbf{u}_K \end{bmatrix} \tag{5.1}$$

denote the matrix specifying all the parameters of the marking-dependent policy. Marking dependent threshold policy can be a specific case of the marking-dependent policy if the formation of the markings allows for a threshold policy.

5.3 Control Problem

Let $\Pi(\mathbf{I}, \mathbf{Y}, \mathbf{U})$ denote the cost of the system for information sources specified by $\mathbf{I}$, markings specified by $\mathbf{Y}$ and marking-dependent policy parameters specified by $\mathbf{U}$. Then the problem of interest can be stated as $\min_{(\mathbf{I},\mathbf{Y},\mathbf{U})} \Pi(\mathbf{I}, \mathbf{Y}, \mathbf{U})$. However, since

$\Pi(\mathbf{I}, \mathbf{Y}, \mathbf{U})$ is not available, it is approximated using $\bar{\Pi}(\mathbf{I}, \mathbf{Y}, \mathbf{U}, D)$, where D denotes the set of traces available about different stochastic processes in the production system that are necessary for simulating the system.

The traces in D can be of different lengths due to the frequency of the events related to them, i.e. in a realistic setting the available traces for slow machines would be shorter than available traces for faster machines. Often, by using data gathered at different times or other means such as using simulated data or bootstrapping methods discussed in Section 4.3 different replications of the data can be available. In addition, it might be the case that before enough data about the system can be gathered, data sets related to systems with parameters close to the available estimates of the system parameters can be generated.

Let $D^{r,l}$ denote the rth data set available related to the lth set of system parameters (referred to as the l'th *instance* of the system). The aim is to find the best marking-dependent policy for a target instance of the system using the available datasets $\{D^{r,l} : 1 \leq r \leq R, 1 \leq l \leq L\}$.

Let ϵ denote the index for the target instance. The data related to ϵ is assumed to correspond to the real parameters of the system, either gathered directly from the system or generated using methods that keep the distributions intact e.g. bootstrap sampling of independent traces. It might be the case that the optimization starts with no evaluation of the system with datasets related to ϵ, this is a subset of the general case discussed here and is referred to as *learning while optimizing* and is addressed in Section 5.5.1. Given these definitions, the problem can be stated as

$$\min_{(\mathbf{I},\mathbf{Y},\mathbf{U})} \frac{\sum_{r=1}^{N} \bar{\Pi}(\mathbf{I}, \mathbf{Y}, \mathbf{U}, D^{r,\epsilon})}{N}. \tag{5.2}$$

5.4 Solution Methodology

Given the complexity of this problem, for solving the problem, we break the problem into three levels as given in the following.

1. P_1: finding the best control parameters for given markings

$$\bar{\Pi}^{1,l}(\mathbf{I},\mathbf{Y}) = \min_{\mathbf{U}} \frac{\sum_{r=1}^{N} \bar{\Pi}\left(\mathbf{I},\mathbf{Y},\mathbf{U},D^{r,l}\right)}{N} \tag{5.3}$$

2. P_2: forming the best markings by clustering the partial information signals for given information sources

$$\bar{\Pi}^{2,l}(\mathbf{I}) = \min_{\mathbf{Y}} \bar{\Pi}^{1,l}(\mathbf{I},\mathbf{Y}) \tag{5.4}$$

3. P_3: finding the best information sources to base the formation of markings on

$$\bar{\Pi}^{3,l} = \min_{\mathbf{I}} \bar{\Pi}^{2,l}(\mathbf{I}) \tag{5.5}$$

Since the goal is optimizing the marking-dependent policy for the target system, rather than the other instances, we are interested in the solution for the problem

expressed in Equation (5.5) for ϵ which would be equivalent to solving the control problem stated in Equation (5.2).

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

5.5 Specifying the Data for the Regression Problems for the Optimization of the System

In the following, we discuss how the two higher levels of the problem of finding the best marking dependent policy (P_2,P_3) can benefit from using regression methods. The lower-level problem can be treated similarly, however, it is not discussed here as the lower level problem for the numerical experiments given here is easily solved by exhaustive search.

5.5.1 Regression Problem for Selection of Information Sources

Given that for a given selection of the information sources $\mathbf{I}$, irrespective of the solution method for (P_1) and (P_2), evaluating $\bar{\Pi}^{2,\epsilon}(\mathbf{I})$ is computationally expensive, the order of $\mathbf{I}$ vectors that are being evaluated can be improved using a regression model. Let $\mathbf{I}^t$ denote the tth information source selection that has been evaluated. The input to

the regression problem can be expressed as

$$X = \begin{bmatrix} \mathbf{I}^1 \\ \vdots \\ \mathbf{I}^t \end{bmatrix}, Y = \begin{bmatrix} \bar{\Pi}^2 \left(\mathbf{I}^1\right) \\ \vdots \\ \bar{\Pi}^2 \left(\mathbf{I}^t\right) \end{bmatrix}. \tag{5.6}$$

Let $f(\mathbf{I})$ denote the regression model built based on X, Y, then $\mathbf{I}^{t+1} = \arg\min f(\mathbf{I})$ can be evaluated next as an educated guess. If there is not enough data available about the plant such that evaluating $\bar{\Pi}(\mathbf{I}, \mathbf{Y}, \mathbf{U}, D^{r,\epsilon})$ through simulation will be erroneous, for avoiding non-efficient control of the production system while more data is gathered, the system can be evaluated using a set of arbitrary parameters beforehand and once estimates of the parameters in the production system are available, a machine learning method can use the results from the arbitrary parameter sets for predicting the best selection of markings given the estimated parameters.

Figure 5.4 depicts the sequence of events in this procedure. Let $\mathbf{\Lambda}^l$ denote the set of parameters of the l'th instance of the production system. Then, the machine learning problem can be expressed as a regression problem with the matrices given in Equation 5.7, where each block of rows are related to a specific set of system parameters and the last block is related to the available evaluations of system performed based on observed system parameters. We refer to this problem as *Selection of Information Sources Using Machine Learning*.

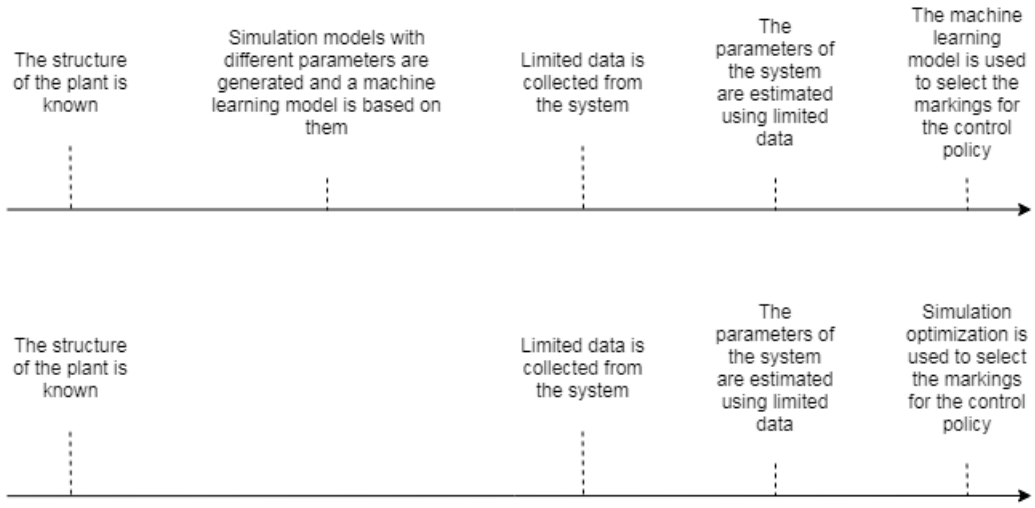

Figure 5.4: time line of the procedure for gathering the data and optimizing the poduction system

$$X = \begin{bmatrix} \mathbf{\Lambda}^1 & \mathbf{I}^1 \\ \mathbf{\Lambda}^1 & \vdots \\ \mathbf{\Lambda}^1 & \mathbf{I}^T \\ \vdots & \vdots \\ \mathbf{\Lambda}^\epsilon & \mathbf{I}^1 \\ \mathbf{\Lambda}^\epsilon & \vdots \\ \mathbf{\Lambda}^\epsilon & \mathbf{I}^{\bar{t}} \end{bmatrix}, Y = \begin{bmatrix} \bar{\Pi}^{2,1}\left(\mathbf{I}^1\right) \\ \vdots \\ \bar{\Pi}^{2,1}\left(\mathbf{I}^T\right) \\ \vdots \\ \bar{\Pi}^{2,\epsilon}\left(\mathbf{I}^1\right) \\ \vdots \\ \bar{\Pi}^{2,\epsilon}\left(\mathbf{I}^{\bar{t}}\right) \end{bmatrix} \tag{5.7}$$

Learning While Optimizing

The regression problem stated earlier using Equation 5.6 is a special case of the *Selection of Information Sources Using Machine Learning* problem. This special case can be beneficial when evaluations about other instances of the system are not available, or there is enough data available about the system that the estimates of the parameters of the system are not expected to change much by new observations.

Hence, evaluating the target instance ϵ is prioritized. In such a setting, if the performance evaluations are much more time consuming compared to applying the machine learning tools, learning can guide the order of the solutions that are being evaluated, helping find good solutions earlier. Furthermore, when few evaluations of the system are available, the training can be repeated when more evaluations become available. Namely, after new solutions have been evaluated, they can be added to the training set and the learner can be retrained, increasing the quality of its guesses about the cost and subsequently the best solution. We refer to this procedure as *learning while optimizing*. This is the utilization of the concept of active learning in the optimization procedure (Settles, 2009). However, in the application of controlling manufacturing systems, the procedure is heavily slanted towards exploitation as opposed to exploration since understanding the shape of the performance measure is secondary compared to optimizing the system. Figure 5.5 depicts a flowchart of this procedure for selection of sources of information.

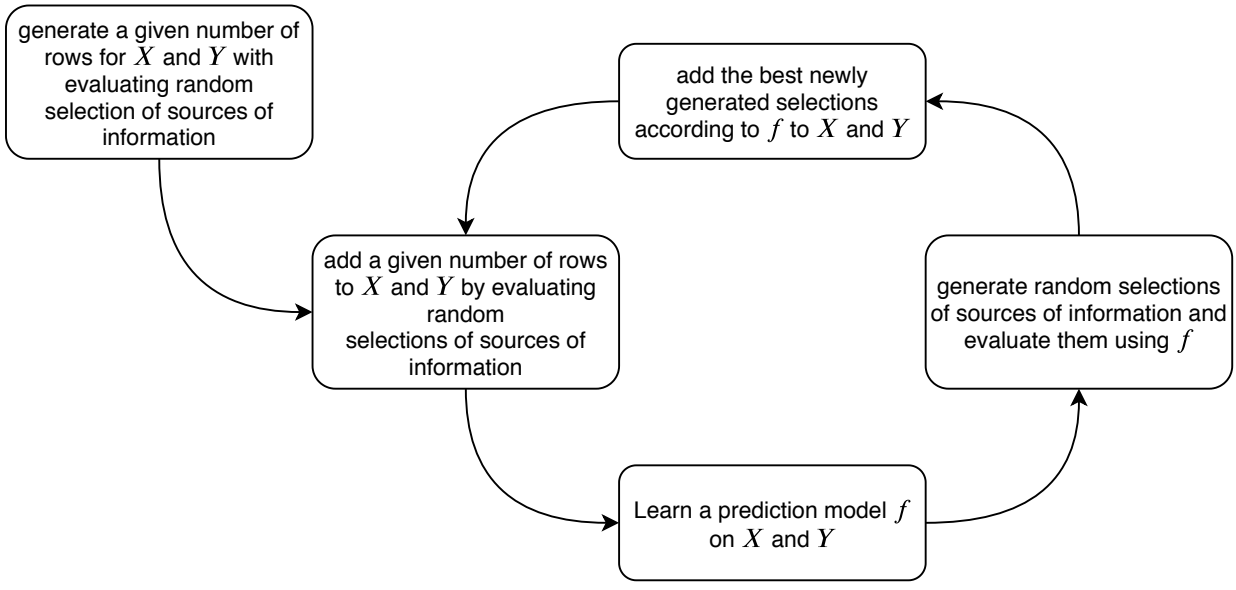

Figure 5.5: Flowchart for learning while optimizing

5.5.2 Regression Problem for Formation of Markings

The same procedure discussed for using regression for selection of sources of information can be applied to the formation of markings. For avoiding symmetric formations (identical marking formations with different $\mathbf{Y}$ values) 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. Furthermore, let

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

denote the vector of $w_{i,j}$ values. Hence, we use $\bar{\Pi}^{1,l}(\mathbf{I}, \mathbf{W})$ and $\bar{\Pi}^{1,l}(\mathbf{I}, \mathbf{Y})$ interchangeably. Then, for a given selection of the sources of information $\mathbf{I}$, the problem of finding the next formation of markings to evaluate can be expressed using the data specified in Equation 5.8. We refer to this problem as *Formation of Markings Using Machine Learning*.

$$X = \begin{bmatrix} \mathbf{\Lambda}^1 & \mathbf{W}^1 \\ \mathbf{\Lambda}^1 & \vdots \\ \mathbf{\Lambda}^1 & \mathbf{W}^T \\ \vdots & \vdots \\ \mathbf{\Lambda}^\epsilon & \mathbf{W}^1 \\ \mathbf{\Lambda}^\epsilon & \vdots \\ \mathbf{\Lambda}^\epsilon & \mathbf{W}^{\bar{t}} \end{bmatrix}, Y = \begin{bmatrix} \bar{\Pi}^{1,l}(\mathbf{I}, \mathbf{W}^1) \\ \vdots \\ \bar{\Pi}^{1,l}(\mathbf{I}, \mathbf{W}^T) \\ \vdots \\ \bar{\Pi}^{1,\epsilon}(\mathbf{I}, \mathbf{W}^1) \\ \vdots \\ \bar{\Pi}^{1,\epsilon}(\mathbf{I}, \mathbf{W}^{\bar{t}}) \end{bmatrix} \tag{5.8}$$

5.6 Numerical Experiments

In this section, we give two sets of numerical experiments that we have used for demonstrating how machine learning can benefit the process of optimization for con-

trolling a manufacturing system and comparing 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.

5.6.1 Production/inventory system

Experimental Setting Description

In the first set of numerical experiments we consider a production/inventory system to show how the marking-dependent policy works and what types of systems can benefit from the usage of machine learning in the process of forming the markings for them.

The system we consider is a three station production/inventory system as depicted in Figure 5.6. In this system, the release of material into machine E_5 can be controlled to balance the accumulation of material in buffers E_4 and E_6. In the following, we show how this system can be controlled using the marking-dependent policy. In addition, using this system we describe how a marking-dependent threshold policy can be expressed as a marking-dependent policy.

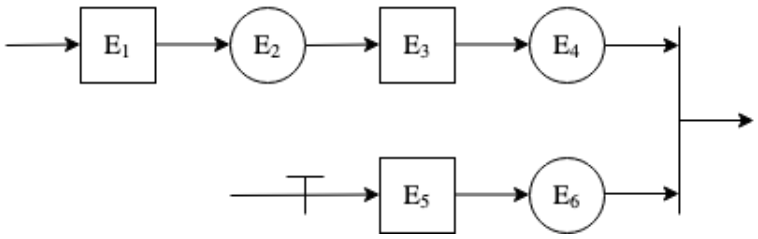

Figure 5.6: Production/inventory system

Table 5.2 gives the details about the different properties of this system. For different instances of the system, the parameters μ_1, r_1, μ_3, μ_5 have been drawn randomly from $[0.55, 0.85]$ and p_1 is drawn randomly from $[0.055, 0.085]$. Figure 5.7 depicts how

Table 5.2: 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	-	-	-

each step of the problem is solved for this set of experiments. For this set of experiments the problem of setting the policy parameters has only two possible solutions and has been solved by exhaustive search.

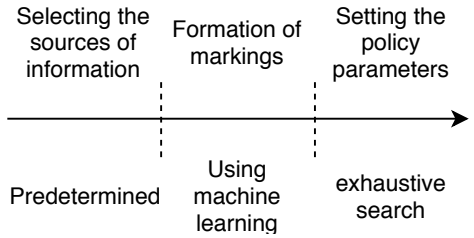

Figure 5.7: Different steps of solving the control problem for the production/inventory system

Marking-Dependent Policy

For this system, the state of the system can be expressed as

$$\eta = (\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6),$$

where, $\alpha_1 \in \{\text{working}, \text{down}, \text{blocked}, \text{starved}\}$, $\alpha_2 \in \{0, \ldots, q\}$, q being the capacity of buffer E_2,

$$\alpha_3, \alpha_5 \in \{\text{working}, \text{blocked}, \text{starved}\},$$

$\alpha_4 \in \{0, \ldots, \infty\}$ and $\alpha_6 \in \{0, \ldots, \bar{S}\}$, $\bar{S}$ being the maximum inventory level allowed by the controller. In this system, if a threshold policy is considered, $\mathbf{I} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$, implying that only the buffer we aim to match with the threshold needs to be observed. If a multi threshold policy is considered that takes the status of the unreliable machine into account, then $\mathbf{I} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$, similarly if the policy also takes into account the status of the intermediate buffer, then $\mathbf{I} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$ and $\hat{\eta} = (\alpha_1, \alpha_2, -, -, -, \alpha_6)$. A base-stock policy with base-stock level S would require two markings ($K = 2$) with the formation of markings specified by Equation 5.9.

$$y_{i,j} = \begin{cases} 1 & \text{if the } i'\text{th } \hat{\eta} \text{ vector satisfies} \\ & \left(-, -, -, -, -, (-1)^{j+1}\alpha_6 \leq (-1)^{j+1}S + \left(\frac{j-1}{2}\right)\right) \\ 0 & \text{o.w.} \end{cases} \tag{5.9}$$

Moreover, a multi threshold policy that uses S_{Down} for when the machine is down and S_{Up} otherwise can be implemented by using two markings ($K = 2$) and the formation of markings given in Equation 5.10.

$$y_{i,j} = \begin{cases} & \text{if the } i'\text{th } \hat{\eta} \text{ vector satisfies} \\ 1 & \left(\alpha_1 = \text{Down}, -, -, -, -, (-1)^{j+1}\alpha_6 \leq (-1)^{j+1}S_{\text{Down}} + \left(\frac{j-1}{2}\right)\right) \\ & \quad \vee \left(\alpha_1 = \text{Up}, -, -, -, -, (-1)^{j+1}\alpha_6 \leq (-1)^{j+1}S_{\text{Up}} + \left(\frac{j-1}{2}\right)\right) \\ 0 & \text{o.w.} \end{cases} \tag{5.10}$$

Both the policies discussed here can be implemented by using the control parameters given in Equation 5.11.

$$\mathbf{U} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{5.11}$$

Machine Learning Set-up

For the experiments here, for linear regression, ANN and random forest we have used the Matlab implementations and for genetic programming, our own implementation. Given the large number of features for the problem of forming the best markings (19508 features), for ANN we choose 200 features randomly and use a neural net with 10 hidden nodes. Figure 5.8 depicts this neural net. For random forest we use 10 trees to be trained based on all the available features and for genetic programming we allow at most 10 nodes.

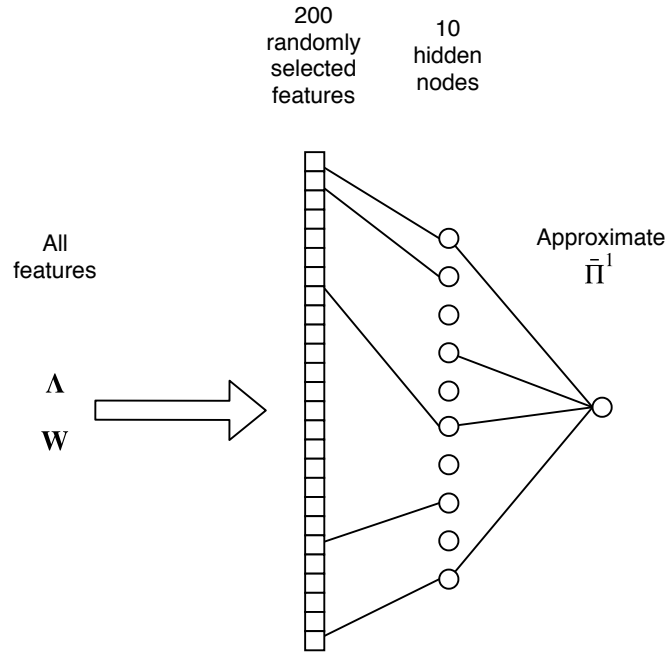

Figure 5.8: Search for the best formation of markings for the production/inventory system based on a learned neural net versus a random search

Results

In this section we give the results of the numerical experiments we have performed for showing how machine learning can improve the process of determining the best formation of markings and comparing the performance of different machine learning methods for this task.

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. Table 5.3 summarizes the average of normalized cost of each method and the percentage of times each method was beneficial on the test instances.

These results indicate using all the methods except for GP is beneficial for this problem. Specifically RF outperforms other method.

The main reason for the poor performance of the GP is the very large number of features for this problem. Since GP's computational requirements 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 reasonable time for the problem of forming the best markings.

Next, in order to show how the good performance of the machine learning methods can translate to savings, we have conducted an experiment to compare searching for the best marking without knowledge about another instance of the system with searching for the best marking using an instance of the system other than the target instance.

Figure 5.9 depicts the comparison between evaluating formations of makings randomly and evaluating them in an order suggested by a learning method that has been trained on one instance of the system with different parameters. These results were obtained after eliminating formations that are worse than the median random formation. This is done because the eliminated formations are so costly that they can be

Table 5.3: Performance on the test data related to formation of markings for the production/inventory system.

	Average of normalized costs	Percentage of times each method was preferable to using a random formation
Random formation of markings	0.4166	-
Artificial neural network	0.1810	80%
Linear regression	0.1949	80%
Random forest	0.0486	100%
Genetic programming	0.4460	45%
Minimum cost formation	0.0477	-

eliminated early on without much computational effort.

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.

5.6.2 Dispatching Problem

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.

Figures 5.10-5.11 depicts the structure and parameters of the system we use and how each step of the problem is solved. For this system, the buffer sizes are set to 5 for all the buffers, E_1 is a reliable machine with processing rate 3 and all the other machines are unreliable machines with processing rate 1, breakdown probability 0.1 and repair rate 0.2. Although formation of markings is not trivial for this problem and can have up to $\left\{ {6^3 \atop 3} \right\} \gg 10^{40}$ feasible solutions, for the purpose of this work we

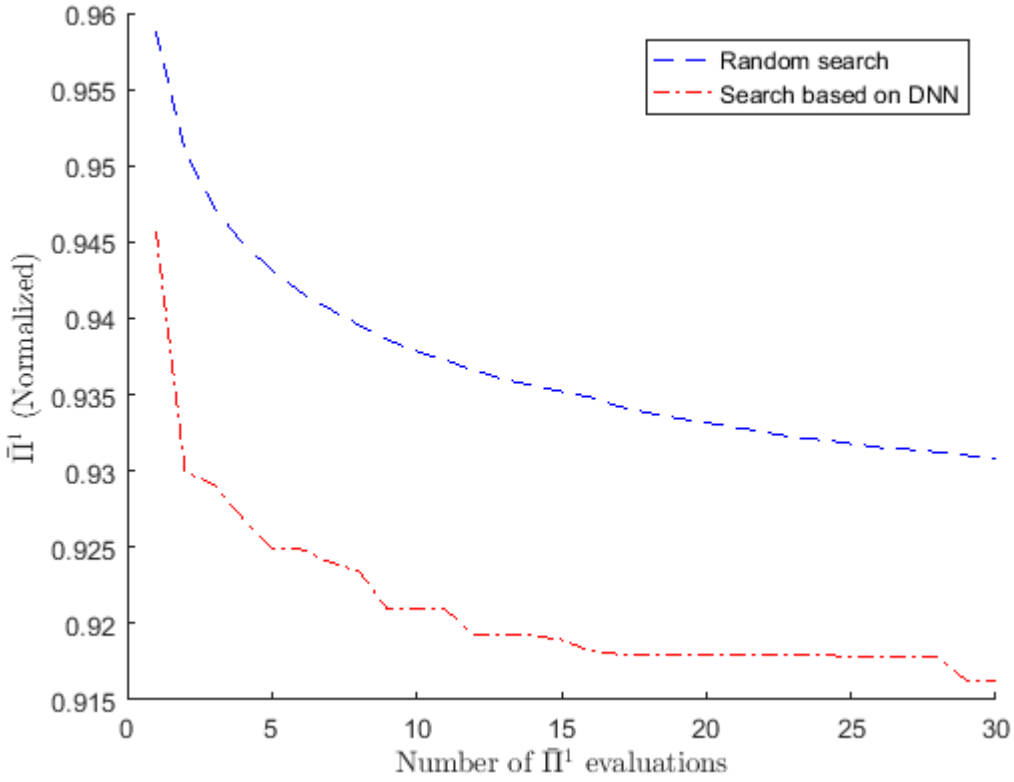

Figure 5.9: Search for the best formation of markings for the production/inventory system based on a learned neural net versus a random search

focus on improving the selection of sources of information for this setting and form the markings by a random search.

This system has been chosen since choosing sources of information for it is not trivial, while the whole set of possible choices can be evaluated for assessing the performance of learned models. Namely, three sources of information were allowed to be chosen from the 13 elements present in the system. This problem has $\frac{13!}{10!3!} = 286$ solutions. Evaluation of all these 286 solutions was done using a cluster that runs up to 40 jobs in parallel, reducing 205 days of computational time to less than a week due to parallelization. Hence, since for each choice of the sources of information, forming

the best marking is time consuming, it can be beneficial to order the solutions to be evaluated in a way that will yield good results earlier.

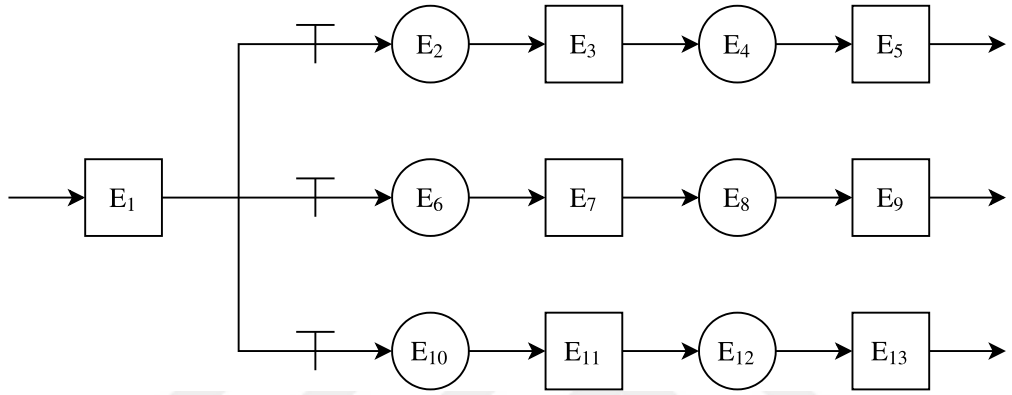

Figure 5.10: The system considered for selection of sources of information

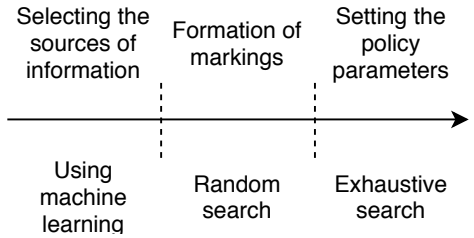

Figure 5.11: Different steps of solving the dispatching problem

Marking-Dependent Policy

For this system, the state of the system can be expressed as

$$\eta = (\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6, \alpha_7, \alpha_8, \alpha_9, \alpha_{10}, \alpha_{11}, \alpha_{12}, \alpha_{13}),$$

where, $\alpha_1 \in \{\text{working}, \text{blocked}, \text{starved}\}$, $\alpha_2, \alpha_4, \alpha_6, \alpha_8, \alpha_{10}, \alpha_{12} \in \{0, 1, \ldots, 5\}$ and

$$\alpha_3, \alpha_5, \alpha_7, \alpha_9, \alpha_{11}, \alpha_{13} \in \{\text{working}, \text{down}, \text{blocked}, \text{starved}\}.$$

Consider a simple 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 $\mathbf{I} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$ and $\hat{\eta} = (-, \alpha_2, -, -, -, \alpha_6, -, -, -, \alpha_{10}, -, -, -)$. This policy would require all the $\hat{\eta}$ vectors where buffer j has the least number of items in it to be assigned to the same marking implying $K = 3$. This can be achieved by using the formation of markings specified by Equation 5.12.

$$y_{i,j} = \begin{cases} 1 & \text{if the } i'\text{th } \hat{\eta} \text{ vector satisfies} \\ & \alpha_{2+4(j-1)} \leq \alpha_k : k \in \{2, 6, 10\} \setminus \{2 + 4(j-1)\} \\ 0 & \text{o.w.} \end{cases} \tag{5.12}$$

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 control parameters given in Equation 5.13.

$$\mathbf{U} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{5.13}$$

Machine Learning Set-up

For the dispatching problem, we only consider one instance of the system. This is because for better assessing the improvements made possible by using machine learning, we have evaluated every possible solution for the problem of selection of

information sources. And this task required more than 205 days of computation that were performed in less than a week on a cluster. For this reason, the parameters of the system do not appear as features for machine learning. In addition, since we consider selecting the sources of information, the features relate to a specific source being selected or not. Hence, there are only 13 features for each data point. For this reason, there is no need for selecting a subset of features for the methods that are computationally more sensitive to the number of features i.e. ANN and GP. Apart from this, the specifics of the machine learning tools used for this set of experiments is identical to that of Section 5.6.1.

Results

In the second set of numerical experiments, we have considered the system depicted in Figure 5.10. 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 chosen by exhaustive search.

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 17.2 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, 10 out of 286 solutions have been selected randomly for training and the remaining 276 solutions have been used for testing the methods. This experiment has been repeated for 100 times and the results are given in Table 5.5. In this case, the advantage of using machine learning methods is limited because there are no other instances of the system available. However, in the following we show how *learning while optimizing* is effective for this case.

Table 5.4: Optimal assignment of $\hat{\eta}$ vectors to different markings

Marking		
1	2	3
$(1,-,2,-,-,-,S,-,-,-,-,-,-)$	$(1,-,1,-,-,-,S,-,-,-,-,-,-)$	$(1,-,1,-,-,-,1,-,-,-,-,-,-)$
$(1,-,B,-,-,-,1,-,-,-,-,-,-)$	$(1,-,2,-,-,-,B,-,-,-,-,-,-)$	$(1,-,1,-,-,-,2,-,-,-,-,-,-)$
$(1,-,B,-,-,-,2,-,-,-,-,-,-)$	$(1,-,S,-,-,-,2,-,-,-,-,-,-)$	$(1,-,1,-,-,-,B,-,-,-,-,-,-)$
$(1,-,S,-,-,-,1,-,-,-,-,-,-)$	$(1,-,S,-,-,-,S,-,-,-,-,-,-)$	$(1,-,2,-,-,-,1,-,-,-,-,-,-)$
$(1,-,S,-,-,-,B,-,-,-,-,-,-)$	$(2,-,1,-,-,-,B,-,-,-,-,-,-)$	$(1,-,2,-,-,-,2,-,-,-,-,-,-)$
$(2,-,1,-,-,-,1,-,-,-,-,-,-)$	$(2,-,2,-,-,-,S,-,-,-,-,-,-)$	$(1,-,B,-,-,-,B,-,-,-,-,-,-)$
$(2,-,2,-,-,-,B,-,-,-,-,-,-)$	$(2,-,B,-,-,-,1,-,-,-,-,-,-)$	$(1,-,B,-,-,-,S,-,-,-,-,-,-)$
$(2,-,B,-,-,-,B,-,-,-,-,-,-)$	$(2,-,B,-,-,-,S,-,-,-,-,-,-)$	$(2,-,1,-,-,-,2,-,-,-,-,-,-)$
$(2,-,S,-,-,-,1,-,-,-,-,-,-)$	$(2,-,S,-,-,-,2,-,-,-,-,-,-)$	$(2,-,1,-,-,-,S,-,-,-,-,-,-)$
$(2,-,S,-,-,-,B,-,-,-,-,-,-)$	$(B,-,1,-,-,-,1,-,-,-,-,-,-)$	$(2,-,2,-,-,-,1,-,-,-,-,-,-)$
$(2,-,S,-,-,-,S,-,-,-,-,-,-)$	$(B,-,2,-,-,-,S,-,-,-,-,-,-)$	$(2,-,2,-,-,-,2,-,-,-,-,-,-)$
$(B,-,1,-,-,-,2,-,-,-,-,-,-)$	$(B,-,B,-,-,-,B,-,-,-,-,-,-)$	$(2,-,B,-,-,-,2,-,-,-,-,-,-)$
$(B,-,1,-,-,-,S,-,-,-,-,-,-)$	$(B,-,B,-,-,-,S,-,-,-,-,-,-)$	$(B,-,1,-,-,-,B,-,-,-,-,-,-)$
$(B,-,2,-,-,-,2,-,-,-,-,-,-)$	$(B,-,S,-,-,-,S,-,-,-,-,-,-)$	$(B,-,2,-,-,-,1,-,-,-,-,-,-)$
$(B,-,2,-,-,-,B,-,-,-,-,-,-)$	$(S,-,1,-,-,-,S,-,-,-,-,-,-)$	$(B,-,B,-,-,-,1,-,-,-,-,-,-)$
$(B,-,S,-,-,-,2,-,-,-,-,-,-)$	$(S,-,2,-,-,-,2,-,-,-,-,-,-)$	$(B,-,B,-,-,-,2,-,-,-,-,-,-)$
$(B,-,S,-,-,-,B,-,-,-,-,-,-)$	$(S,-,B,-,-,-,B,-,-,-,-,-,-)$	$(B,-,S,-,-,-,1,-,-,-,-,-,-)$
$(S,-,2,-,-,-,1,-,-,-,-,-,-)$	$(S,-,B,-,-,-,S,-,-,-,-,-,-)$	$(S,-,1,-,-,-,1,-,-,-,-,-,-)$
$(S,-,S,-,-,-,2,-,-,-,-,-,-)$	$(S,-,S,-,-,-,B,-,-,-,-,-,-)$	$(S,-,1,-,-,-,2,-,-,-,-,-,-)$
		$(S,-,1,-,-,-,B,-,-,-,-,-,-)$
		$(S,-,2,-,-,-,B,-,-,-,-,-,-)$
		$(S,-,2,-,-,-,S,-,-,-,-,-,-)$
		$(S,-,B,-,-,-,1,-,-,-,-,-,-)$
		$(S,-,B,-,-,-,2,-,-,-,-,-,-)$
		$(S,-,S,-,-,-,1,-,-,-,-,-,-)$
		$(S,-,S,-,-,-,S,-,-,-,-,-,-)$

For this instance of the system, the optimal selection of information sources is

$$\begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

and the optimal formation of markings is given in Table 5.4. No buffer has been chosen as an information source. This can be attributed to the relatively large size of the buffers for this system that makes them less likely to block the upstream machine and generate waiting time, however the machine failures can directly generate waiting time for all the items waiting in the upstream buffer. 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 performing random forest after every 5 evaluations of the solutions. Equation (5.14) gives an example of the initial training data with five randomly chosen solutions (each row corresponds to a **I** vector) and their corresponding throughput. Since we consider one instance of this system, the system parameters can be dropped from the features as they are identical for all the rows.

These experiments have been repeated for 1000 replications and the results are depicted in Figure 5.12. The starting portion for the two approaches are similar because random forest is not used in choosing the solutions until enough data is available. However, with relatively limited data, random forest is able to suggest evaluating considerably better solutions.

$$X = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}, Y = \begin{bmatrix} 1.5461 \\ 1.6041 \\ 1.5510 \\ 1.5760 \\ 1.5393 \end{bmatrix} \tag{5.14}$$

5.7 Conclusions

In this work we define two machine learning problems that can assist in finding the best marking-dependent policy for production systems in practical time. We compare the performance of different machine learning algorithms 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 except for genetic programming for forming the markings. This can be attributed to the fact that the problem of forming the markings has a structure

Table 5.5: Performance on the test data related to selecting sources of information for the dispatching problem. Percentage of times the method was preferable to using a random selection of sources of information using 10 selections for learning. The 10 learning data points were selected randomly from the 286 solutions for one instance of the system and the experiment was repeated 100 times.

	Average percentage deviation from the optimal selection	Percentage of times each method was preferable to using a random selection
Random selection of sources	3.7584	-
Artificial neural network	3.6231	52%
Linear regression	3.2847	58%
Random forest	2.8419	84%
Genetic programming	3.4324	61%

that results in data with a very large number of features and this is computationally harmful to genetic programming. Additionally, our results show the comparative advantage of random forests for both of the 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 very beneficial with very few data points through the procedure of learning while optimizing.

This work can be extended in different directions. First, although the problem of formation of the markings 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 vectors of the state of the system. 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, allowing for incorporating multi-task learning methods into the optimization. Multi-task learning has been developed to

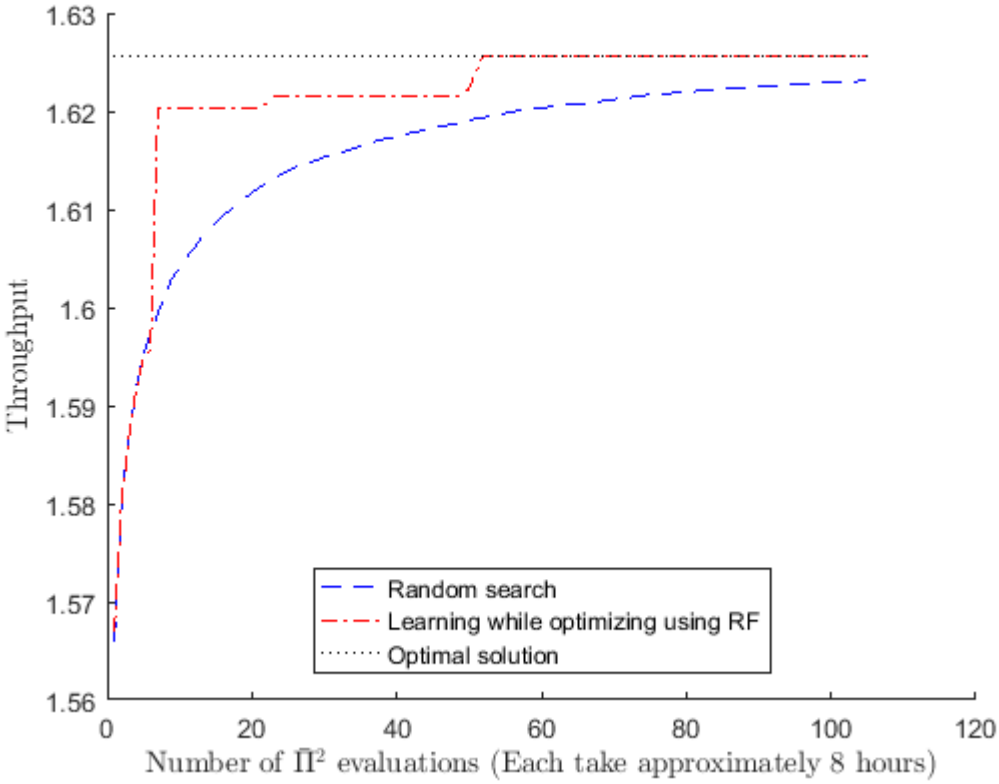

Figure 5.12: Comparison of random search and Learning While Optimizing for the dispatching problem.

improve the prediction models for tasks with few data points by receiving assistance from tasks with many data points. Hence, both problems discussed here can be viewed as multi-task machine learning problems.

Chapter 6

IMPLEMENTATION OF THE DATA-DRIVEN JOINT SIMULATION AND OPTIMIZATION METHOD USING A LEGO PRODUCTION SYSTEM

In this chapter, we discuss the implementation of the example setting from Section 2.2 by using a Lego production system. The Lego production system consists of conveyors, workstations, gates, sensors and EV3 bricks that enable the connection of the physical system with the computer that controls the set-up. In the following we first introduce different components of the set-up and give the algorithm for control of the set-up with the making-dependent threshold policy. Then, we conduct experiments where for given parameters for the set-up, the actual inter-event times generated by the set-up are collected and compared to that of the analytical models. Finally, we apply the data-driven JSO method, exponential fitting and phase-type fitting to the traces gathered from the physical system.

6.1 Literature Review

To counter the lack of hands-on experience with production systems that many engineering students have Lego production systems have been used with educational purposes. Lego production systems have been used for teaching simulation at Politechnico di Milano, Korea Advanced Institute of Science and Technology and other universities (Jang and Yosephine, 2016; Syberfeldt, 2010; Lugaresi et al., 2019a; Sanchez and

Bucio, 2011). The systems used in these studies include reliable and unreliable, closed loop and open loop lines.

As tools for facilitating research, Lego production systems have been used as proof of concept for the digital twin and real-time simulation (Lugaresi et al., 2019b; Travaglini, 2018). The digital twin concept in manufacturing has been defined as a digital representation of a system that can accommodate synchronization between the real system and its simulations. Real-time simulation refers to initializing simulation optimization with the current state of the system and employing the identified solutions in the real system as soon as the solution is ready.

Lego production systems have also been used for automatic generation of simulation models based on data collected from production systems without additional knowledge about the structure of the system (Lugaresi et al., 2019c). Lego production systems can provide a test best for studies that consider the processes of data gathering, data processing and parameters fitting along with optimization as opposed to assuming all these steps have been performed prior to optimization.

6.2 Construction of System Elements

Here, we describe the construction of the elements of the Lego production line separately. These elements can be connected together to form different production systems.

6.2.1 Transportation

Transportation of material between the work stations and the synchronization station is carried out by using conveyors as depicted in Figures 6.1-6.2. The conveyors also act as the buffers for the system.

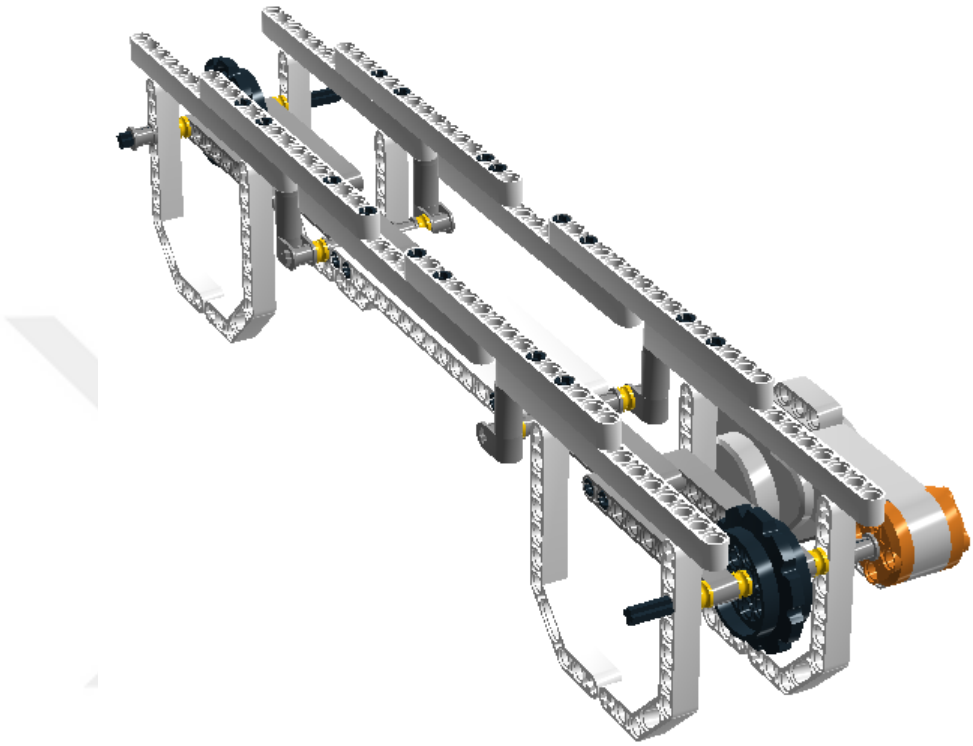

Figure 6.1: Model of a conveyor without pallets

6.2.2 Work Stations and Gates

Work stations are built by using belts that hold the parts while the work station is working and move the part downstream when working on the part is finished. The parts are fed to the workstations using gates. Gates fulfill two purposes. First, they separate the parts from the workstation and allow control of the release of parts into the workstation; second, they provide a force orthogonal to the path of a part allowing for bending the conveyor path. A work and a gate are depicted in Figures 6.3-6.4 and a gate connected to a workstation is depicted in Figure 6.5

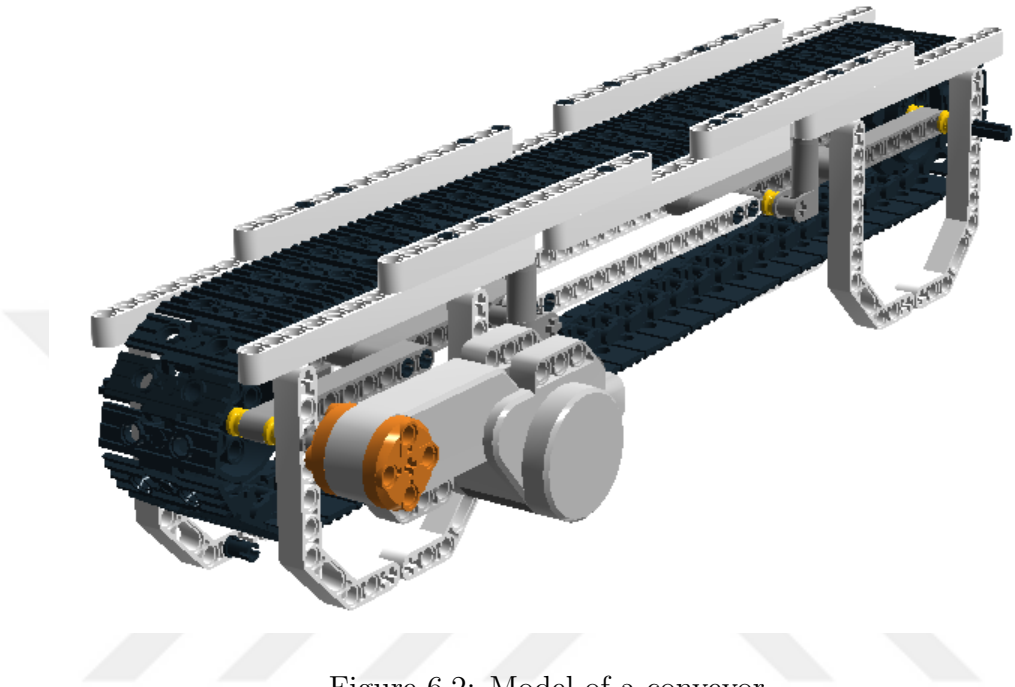

Figure 6.2: Model of a conveyor

6.2.3 Sensors

Sensors provide the data that will be used by the algorithm that controls the system. We use color sensors that report the color of the object under them. Figure 6.7 depicts a sensor with its stand that can be installed in different locations in the system for different purposes. However, most of the sensors are used for the control of a workstation subject to blocking and starvation. A workstation subject to blocking and starvation requires three sensors to operate, the *gate sensor* that signals starvation, the *machine sensor* that indicates if a part is in the workstation or not and a *blocking sensor* that signals blocking if the downstream buffer is full. Hence, the placement of the *blocking sensor* for a workstation determines the size of the buffer between that workstation and the downstream work station. Figure 6.8 depicts the placement of sensors for a workstation subject to blocking and starvation.

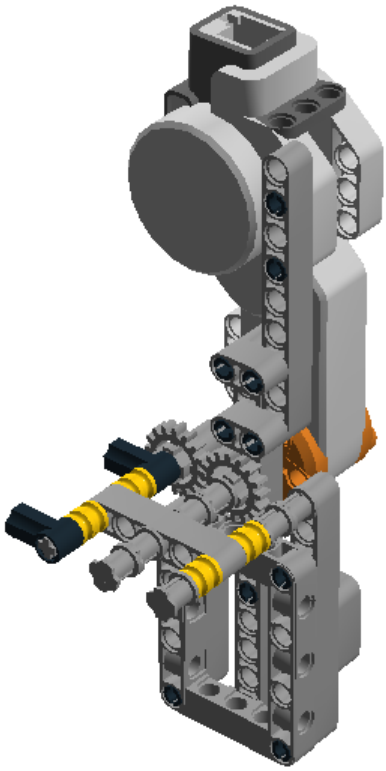

Figure 6.3: A work station

6.2.4 Synchronization Station

A synchronization station is a station that allows the material to continue their path only if all the buffers feeding into the synchronization station have at-least one item in them. The synchronization station is built using two gate mechanisms and two sensors as depicted in Figure 6.9. The arrows in Figure 6.9(a) show the direction of the flow of the material and the arrow in 6.9(b) shows the movement of the gate mechanism for taking the parts in and ejecting them to the downstream buffer.

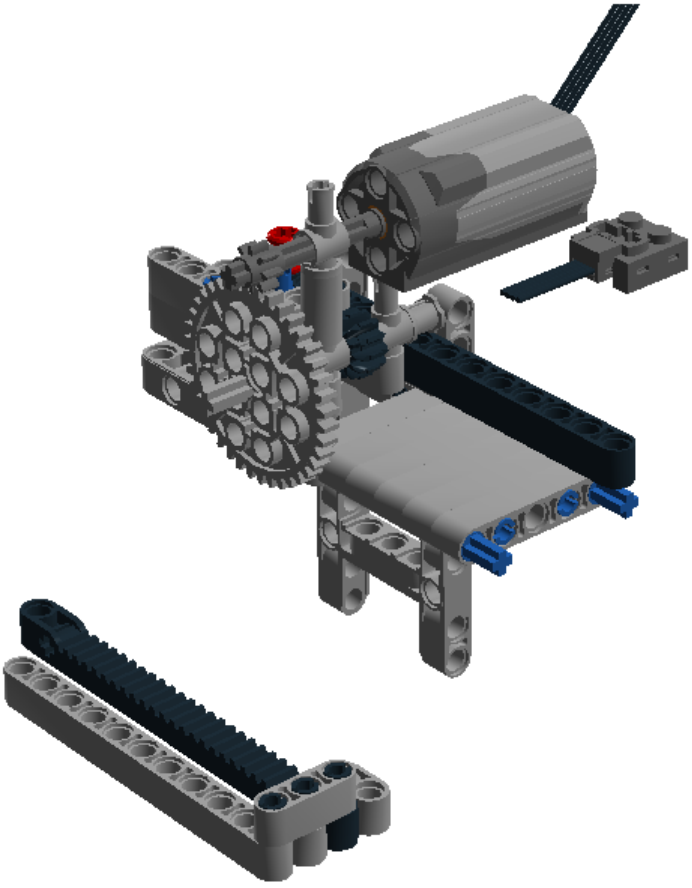

Figure 6.4: A gate mechanism

6.3 Model

We have constructed the Lego production/inventory system as a physical model of the system introduced in Section 2.2. Figures 6.10-6.11 depict the system and its schematic representation. In Figure 6.10 the arrows show the direction of the flow of the material in the system. In this system, the release of parts to WS_1 is controlled to balance the inventory accumulated in the downstream buffers of WS_1 and WS_3. The parts waiting in these two buffers generate costs with different cost rates. Also, WS_2 is unreliable and its status (*in working condition* or *under repair*) forms the markings

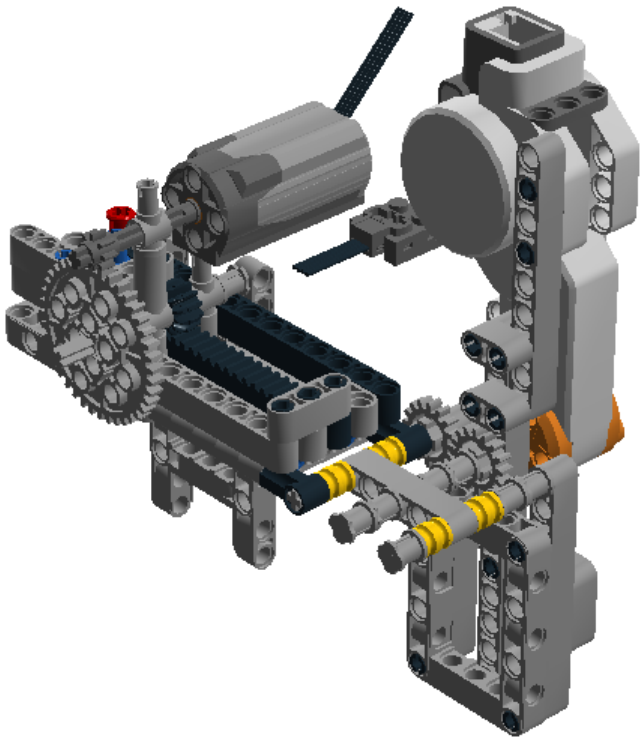

Figure 6.5: A gate attached to a work station

used for making the decision to release parts into WS_1. Here we do not consider marking generated based on the buffer between WS_2 and WS_3. For the experiments we have used the parameters given in Table 6.1 for the Lego production/inventory system.

6.3.1 Centralized Control for the Lego Production/Inventory System

For controlling the system with the marking dependent policy, when deciding on the release of material to WS_1, the status of WS_2 should be considered. However as depicted in Figure 6.3 the sensors related to these workstations are connected to different EV3 bricks. For this reason, the control of the system cannot be totally

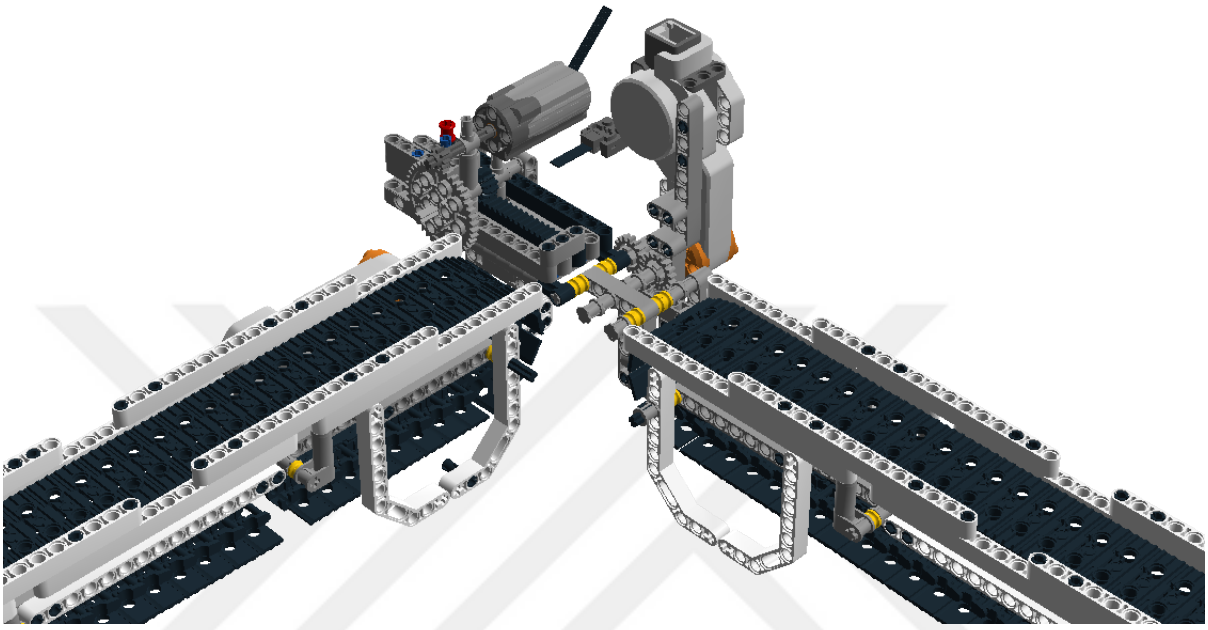

Figure 6.6: A workstation and its upstream and downstream buffers

decentralized. We use a centralized design for the control of the system, where a single algorithm controls all the activities in the system. The system components are connected to EV3 bricks, and the EV3 bricks are connected to the computer that controls the system via bluetooth. A Matlab script on this computer controls the system. Travaglini (2018) uses an ethernet modem for connecting the EV3 bricks to the computer and uses a different session for controlling the components connected to each EV3 brick. Algorithm 2 along with its submodules Algorithms 3-5, give the pseudocode for controlling the Lego production/inventory system with the marking-dependent threshold policy. Table 6.2 gives the description of the notation used for the pseudocode.

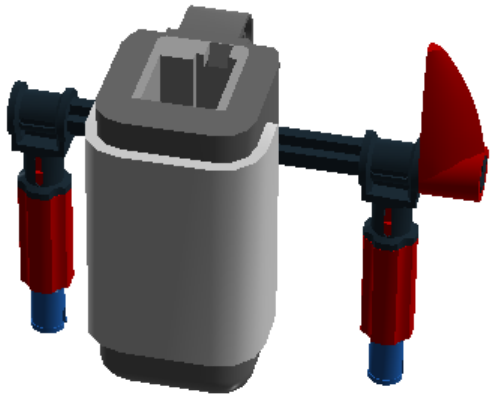

Figure 6.7: A gate attached to a work station

6.4 Numerical Results

6.4.1 Parameter Setting for the Experiments

With the system parameters given in Table 6.1, in-order to run the system we need to specify a parameter that relates the mean of the exponential distributions with unit time. Here, the time scale parameter determines the relation between the system parameters and the physical time unit i.e. if the time scale parameter is equal to 1, the average stand-alone processing time of WS_2 will be 1 second and if the time scale parameter is equal to 20, the average stand-alone processing time of WS_2 will be 20 seconds. We use four values for the time scale parameter in our experiments. For each value of the time scale parameters, we run the system with different pairs of policy parameters $(S_{\text{Down}}, S_{\text{Up}})$. For each run of the system, we collect the traces of the demand and production times and the sample path of the inventory position to be able to calculate the average cost for a given set of policy parameters and time scale parameter. Figures 6.12-6.14 depict the properties of these traces and Figure 6.15 depicts one of the sample paths related to each time scale parameter.

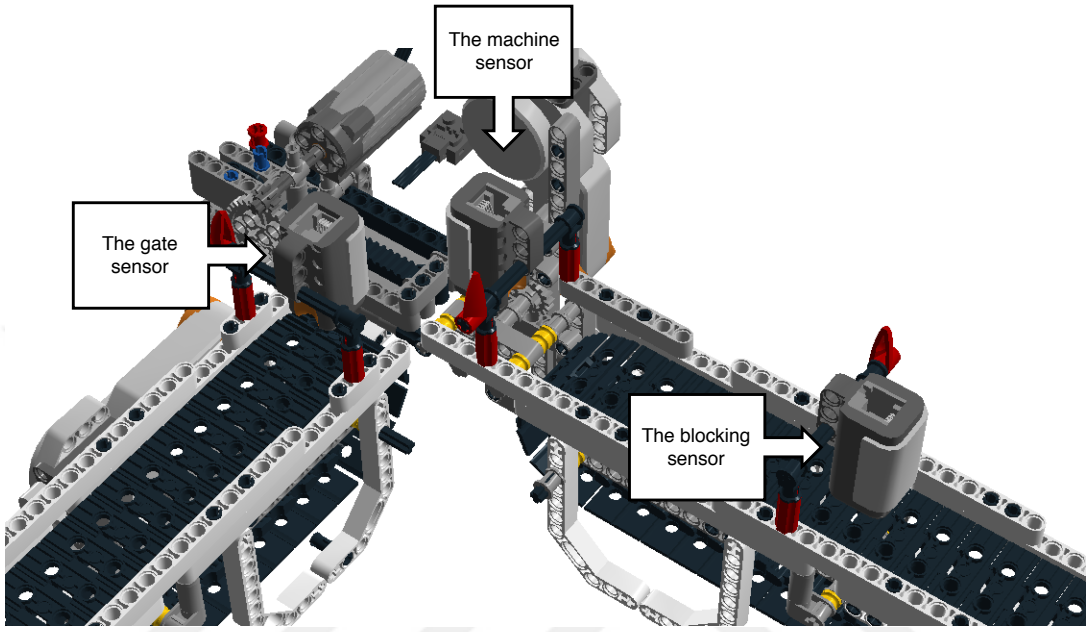

Figure 6.8: The placement of the sensors on for a workstation

6.4.2 Properties of The Analytical Model of the System

Using the system parameters given in Table 6.1, the schematic representation of the system given in Figure 6.6 and the analytical methods given in Chapter 3, the system can be evaluated for different values of S_{Down} and S_{Up}. Figure 6.16 depicts the analytical cost function $\pi_{1,0}(S_{\text{Down}}, S_{\text{Up}})$ for this set of system parameters and Figure 6.17 depicts the cost function $\pi_{0,0}(S)$ for a single threshold policy. The cost function is more sensitive to the threshold level when the machine is in working condition. This is because the machine is more-often in working condition rather than being repaired. In addition, Figures 6.18-6.19 give the properties of the analytical models of the demand process.

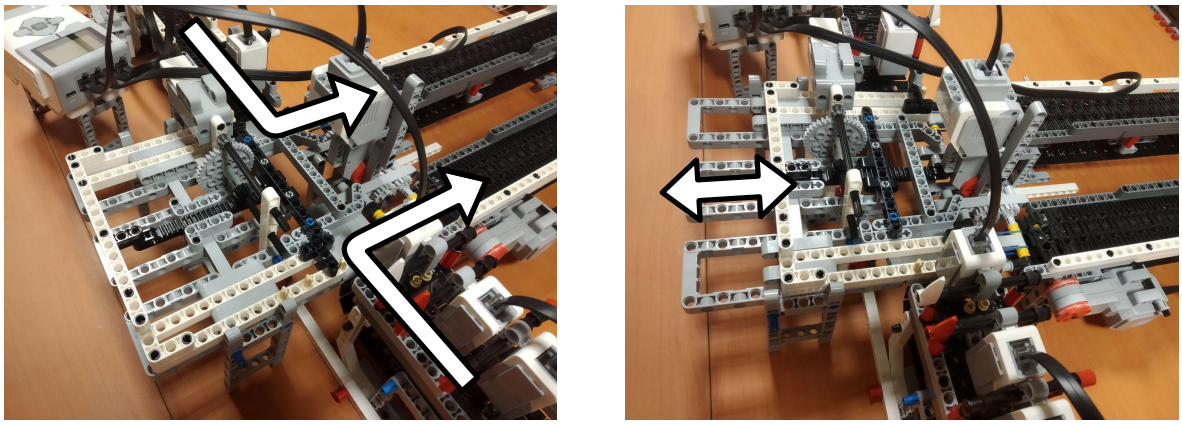

Figure 6.9: The synchronization station

6.4.3 The Disparity Between the Properties of the Analytical Models of the Processes and the Observed Traces and its Main Contributors

Although the time scale parameter does not affect the analytical models, it influences the properties of the data gathered from the physical system because of delays related to the physical system that are not modeled in the analytical models. As the time scale parameter decreases, the effect of these delays become more important. For very small values of the time scale parameter, the delays mostly determine the inter-event time distributions as opposed to the stand alone processing times. This is depicted in the samples given in Figure 6.15.

The two main reasons that cause a disparity between the analytical models and the empirical data are ignoring the small delays in the system and correlation between the different inter-event times in the system due to the specifics of the system. In this case the delays are caused by the conveyors and gates and the dependency is caused by centralized control of the system. More specifically, for this system, the following factors contribute most to the inaccuracy of the analytical models:

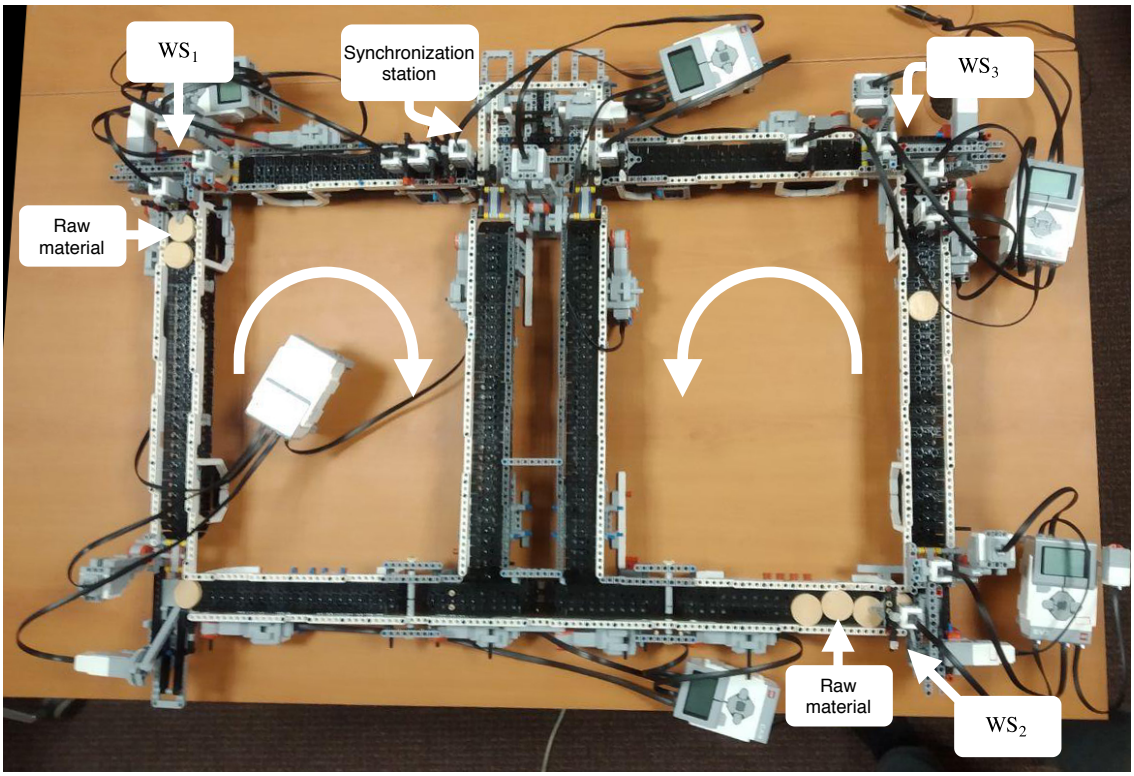

Figure 6.10: The Lego production/inventory system

- The travel time of the parts on the conveyors
- The time spent for opening and closing the gates
- The time spent for communications between the physical system and the computer that controls the system
- Time spent for centralized control of the system

The time spent on transportation can be modeled by pseudo-stations with infinite buffer and infinite servers as depicted in Figure 6.20. This approach is employed for modeling transportation systems in the literature related to mobility on demand

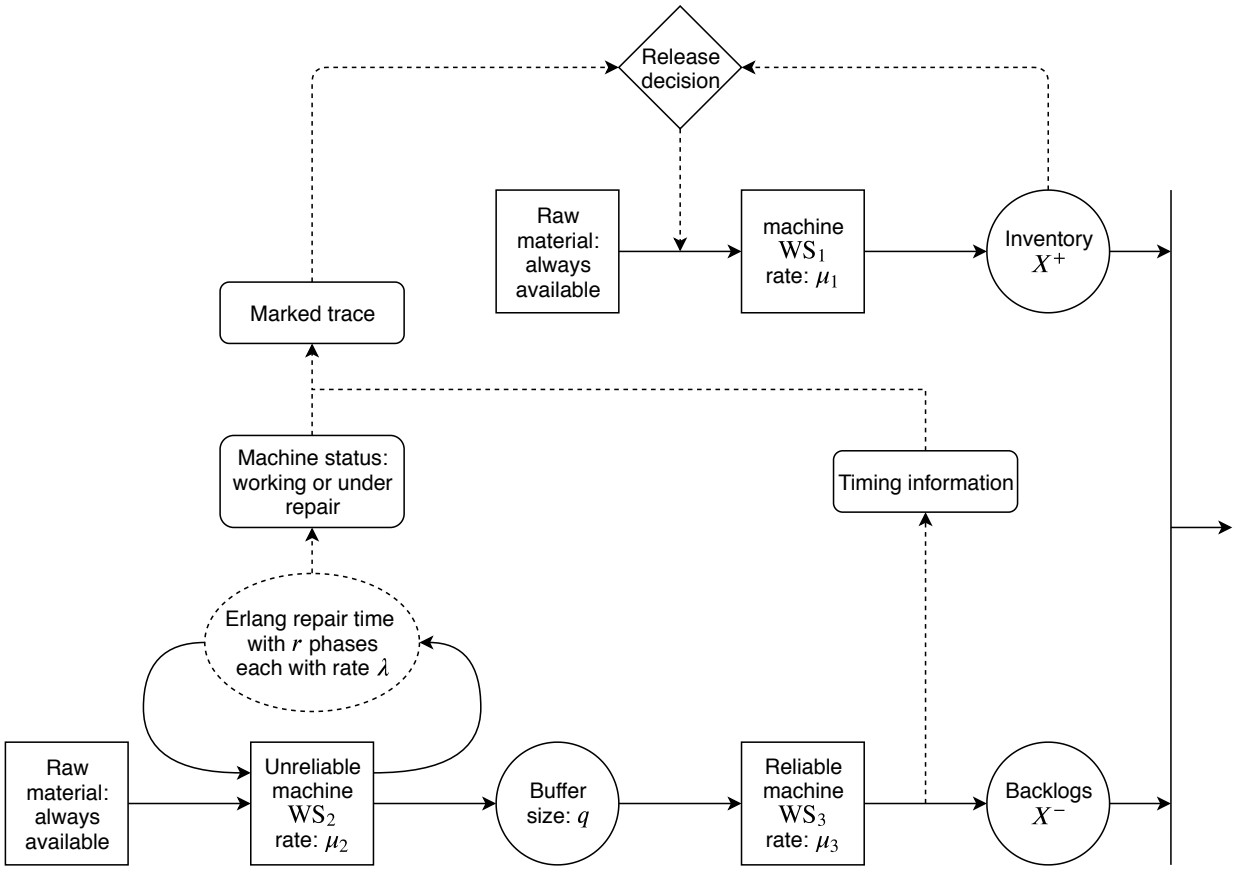

Figure 6.11: The schematic representation of the Lego production/inventory system

(Zhang and Pavone, 2016). The time spent on centralized control of the system can be modeled by a pseudo station with one machine that has to produce an item for every other workstation in-order for them to be able to function. Figure 6.21 depicts such a pseudo machine.

Table 6.1: The values of the system parameters

System parameter	Description	Value
q	The capacity of the buffer between WS_2 and WS_3	1
μ_2	Processing rate of WS_2 (processing times are exponentially distributed)	1
μ_3	Processing rate of WS_3 (processing times are exponentially distributed)	1
r	Number of the phases of each repair time	4
λ	The rate of each repair phase (duration of each repair phase is exponentially distributed)	0.5
Repair time mean		8
Repair time CV		0.5
γ	Breakdown probability	0.09
μ_1	Processing rate of WS_1 (processing times are exponentially distributed)	1.0054

6.5 Performance of the Data-driven JSO on the Lego Production/inventory set-up

In this section, we apply the control methods based on parameter fitting and the data-driven JSO as described in Chapter 4 to the traces gathered from the physical system. Here, we consider setting a single threshold for the system. For the phase-type fitting method, we allow using matrices of up to size 10×10. For all the methods we set the maximum admissible threshold level to 100. Table 6.3 gives the thresholds each methods suggests. For small time scale parameters, exponential and phase-type fitting do not generate satisfactory answers. This is because the inter-event time distributions for both traces are affected significantly by the delays in the system and do not resemble the exponential distribution when the time scale parameter is relatively small. In other words, there is less stochasticity in the system, but exponential fitting and phase-type fitting fail to consider this in setting the threshold.

Table 6.2: Description of the variables used in the algorithm for control of the Lego system.

Variable	Description
T	The value of the clock
α	The status of WS_2
SV	The value for the switch that turns the system on and off
SF_i	Scheduled finishing time for WS_i
$\dot{b}$	A trace of binary indicators that shows if a breakdown will occur or not
$\dot{g}_i$	The trace of processing times for WS_i
$\dot{r}_i$	The trace of repair times for WS_2
$i_{g_1}, i_{g_2}, i_{g_3}, i_b, i_r$	Counters for the traces

This effect is significantly less harmful to JSO. For the largest time scale parameter phase-type fitting suggests using a threshold closer to the optimal analytical threshold. However, it should be noted that given the disparities between the analytical solutions and the behavior of the system assessing the true optimal solution with the accuracy sufficient to compare the methods needs much longer runs of the physical system with these thresholds.

Table 6.3: The threshold level suggested by each method

		Method		
		JSO	EXP	PH
Time scale parameter	1	3	100	64
	4	2	48	6
	10	2	25	4
	20	1	4	2

6.6 Future Directions

This work can be extended in different directions. The Lego production system can be used for devising and testing data-driven methods that consider the current state

Algorithm 2 Algorithm for control of the Lego production/inventory system

Initialize EV3s, motors and sensors
Start the clock
$X \leftarrow 0$
$i_{g_1}, i_{g_2}, i_{g_3}, i_r, i_b \leftarrow 1$
$SV \leftarrow$ off
$\alpha \leftarrow$ in working condition
while $T <$ time limit **do**
    if The touch sensor is pushed **then**
        wait for 0.4 seconds
        if The touch sensor is pushed **then**
            if $SV =$ on **then**
                Turn the conveyors off
                $SV \leftarrow$ off
            else
                Turn the conveyors on
                $SV \leftarrow$ on
            end if
        end if
    end if
    if $SV =$ on **then**
                ▷ WS_2
        Call the submodule for WS_2
                ▷ WS_3
        Call the submodule for WS_3
                ▷ Synchronization Station
        if both of the Synchronization station's sensors detect a part **then**
            Take a part in from each of the upstream buffers
            Release the parts to the buffer that closes the system's loop
        end if
                ▷ WS_1
        Call the submodule for WS_1
    end if
end while
Save the generated trajectory of X

of a system and take into account that a good solution must be reached before the system changes drastically. The advantage of using a physical system for evaluating

Algorithm 3 Submodule for WS_1

if WS_1's gate sensor sees a part **then**
  if WS_1's machine sensor does not see a part **then**
    $o_1 \leftarrow \alpha =$ "Under repair" $\wedge$ S_{Down}'s sensor does not see a part
    $o_2 \leftarrow \alpha =$ "In working condition" $\wedge$ S_{Up}'s sensor does not see a part
    if $o_1 \vee o_2$ **then**
      WS_1 takes a part in from the upstream conveyor
      $SF_1 \leftarrow T + \dot{g}_1(i_{g_1})$, $i_{g_1} \leftarrow i_{g_1} + 1$
    end if
  end if
end if
if The SW_1's machine sensor sees a part **then**
  if $SF_1 < T$ **then**
    WS_1 releases a part to the downstream conveyor
    $X \leftarrow X + 1$
  end if
end if

Algorithm 4 Submodule for WS_2

if WS_2's gate sensor sees a part **then**
  if WS_2's machine sensor does not see a part **then**
    WS_2 takes a part in from the upstream conveyor
    if $\dot{b}(i_b)$ **then**
      $SF_2 \leftarrow T + \dot{g}_2(i_{g_2}) + \dot{r}(i_r)$, $i_{g_2} \leftarrow i_{g_2} + 1$, $i_r \leftarrow i_r + 1$
      $\alpha \leftarrow$ Under repair
    else
      $SF_2 \leftarrow T + \dot{g}_2(i_{g_2})$, $i_{g_2} \leftarrow i_{g_2} + 1$
    end if
    $i_b \leftarrow i_b + 1$
  end if
end if
if The SW_2's machine sensor sees a part **then**
  if $SF_2 < T \wedge SW_2$'s blocking sensor does not see a part **then**
    WS_2 releases a part to the downstream conveyor
    $\alpha \leftarrow$ In working condition
  end if
end if

Algorithm 5 Submodule for WS_3

if WS_3's gate sensor sees a part **then**
  if WS_3's machine sensor does not see a part **then**
    WS_3 takes a part in from the upstream conveyor
    $SF_3 \leftarrow T + \dot{g}_3(i_{g_3})$, $i_{g_3} \leftarrow i_{g_3} + 1$
  end if
end if
if The SW_3's machine sensor sees a part **then**
  if $SF_3 < T \wedge SW_3$'s blocking sensor does not see a part **then**
    WS_3 releases a part to the downstream conveyor
    $X \leftarrow X - 1$
  end if
end if

different methods is that the ability of the methods to react to the properties of the system that escape modeling can be assessed. As shown here, even relaxing the common assumptions about independent exponential inter-event times can fall short of matching the analytical models and the behavior of a physical system. To address this, Q-learning can be combined with the data-driven methods to fine-tune the policies on the fly. In addition, data-driven JSO can be used for generating a basis for ordinary behavior of a system for training outliers detection learning methods.

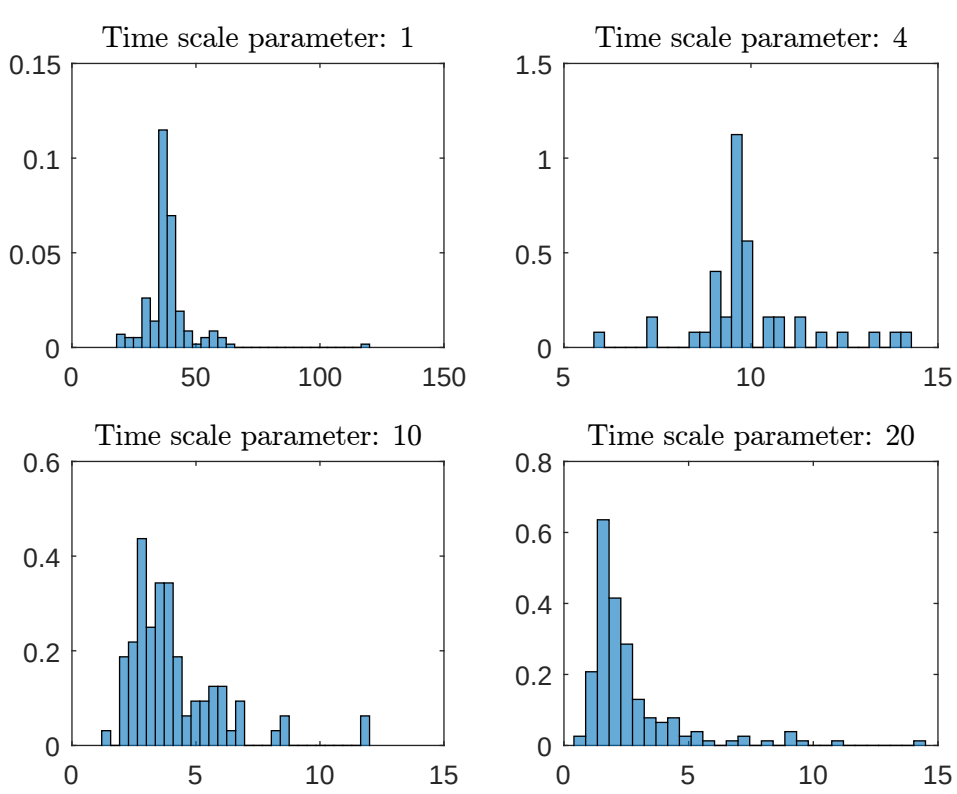

Figure 6.12: The distribution of the demand inter-arrival times gathered from the physical system

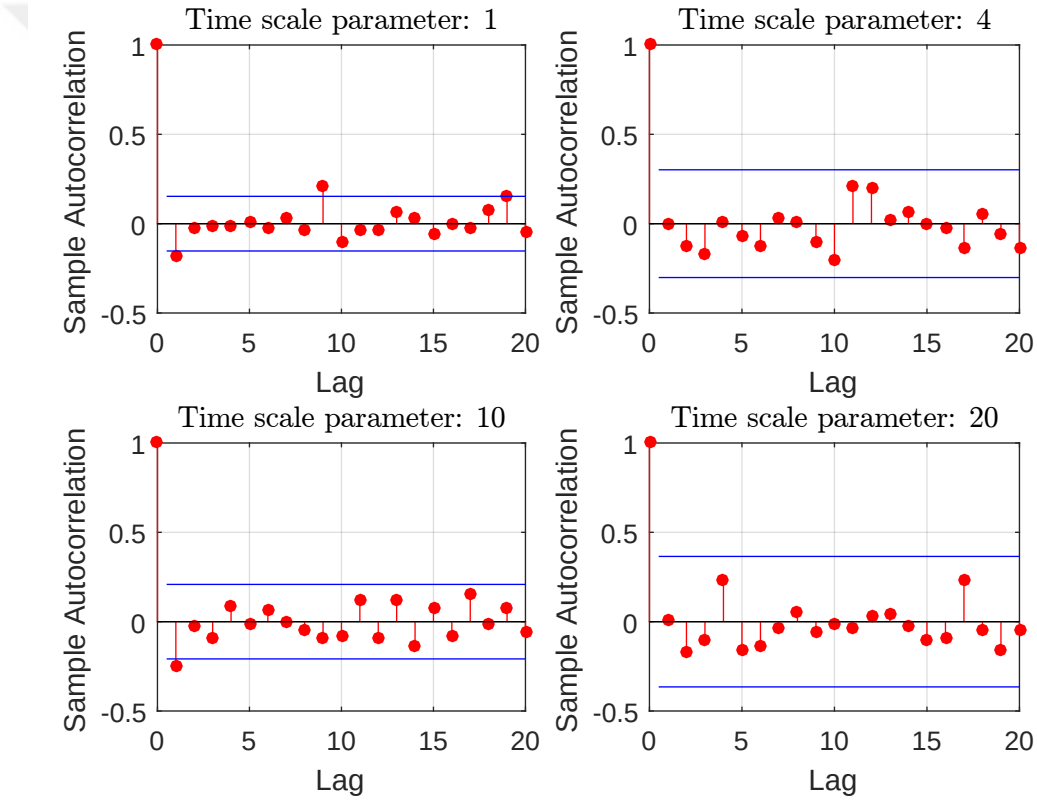

Figure 6.13: The autocorrelation function of the demand inter-arrival times gathered from the physical system

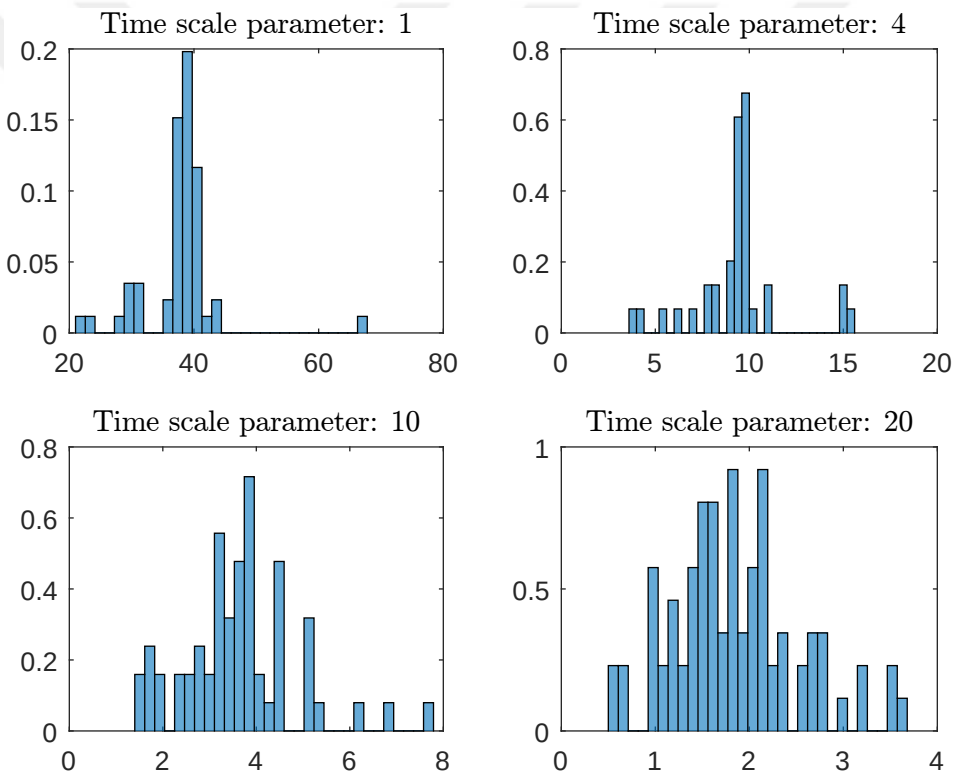

Figure 6.14: The distribution of the production times gathered from the physical system

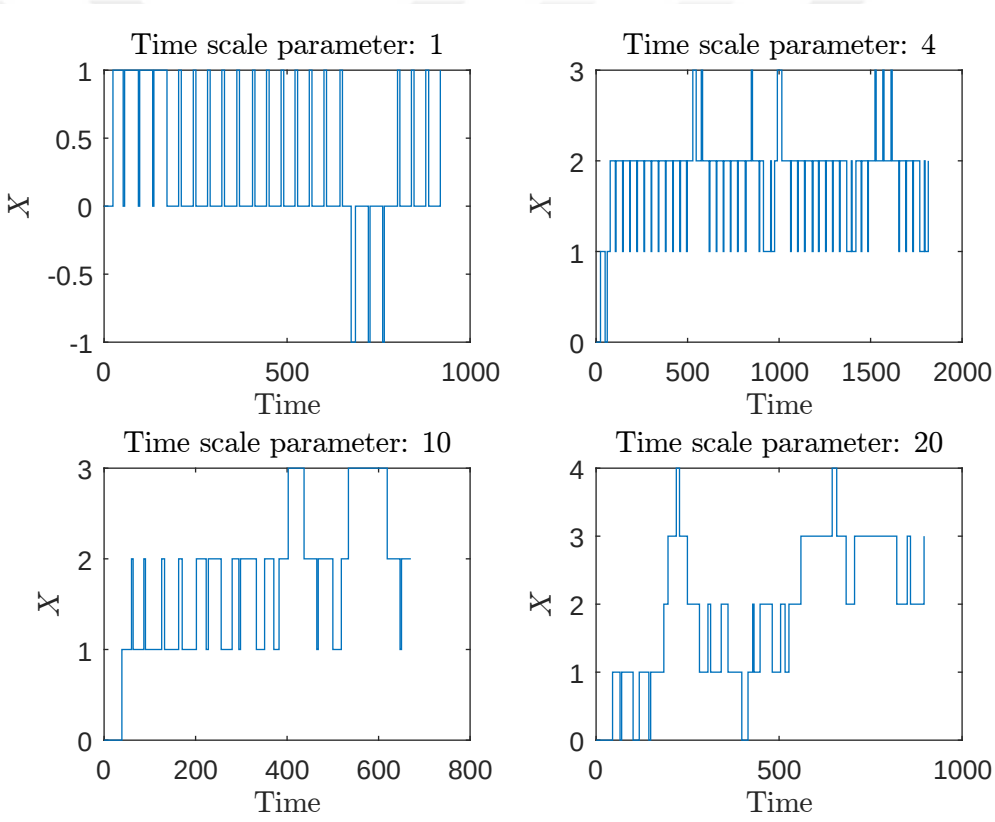

Figure 6.15: Sample paths of the inventory position gathered from the physical system

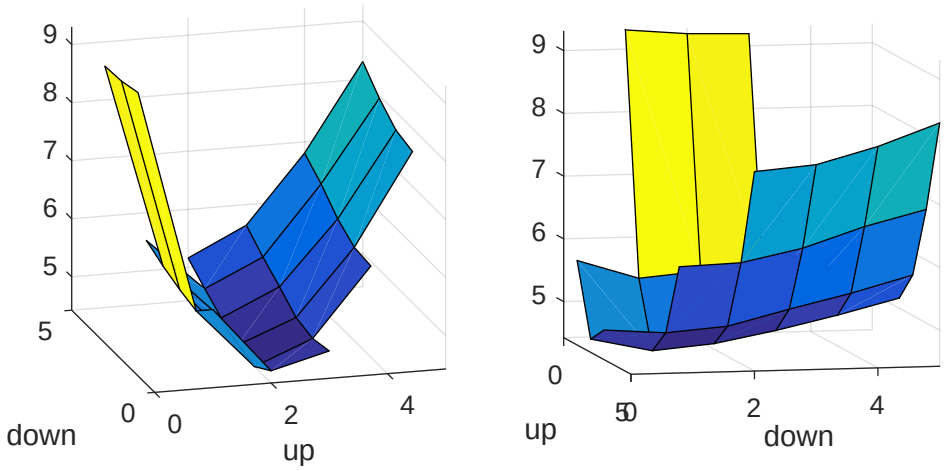

Figure 6.16: Analytical cost function $\pi_{1,0}(S_{\text{Down}}, S_{\text{Up}})$

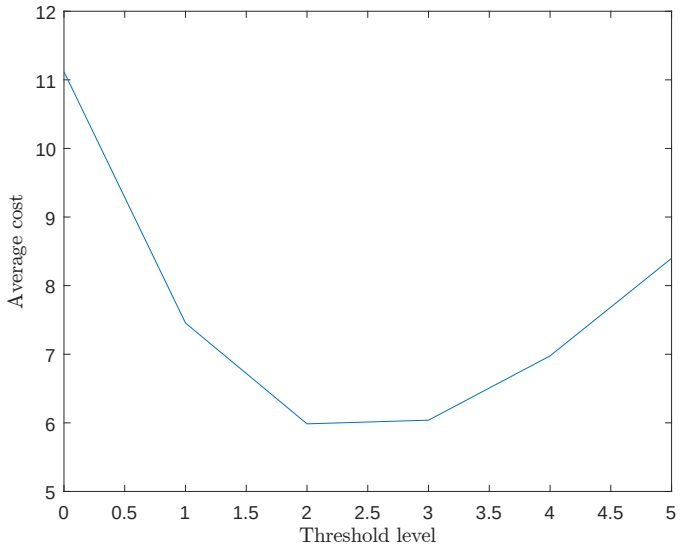

Figure 6.17: Analytical cost function $\pi_{0,0}(S)$

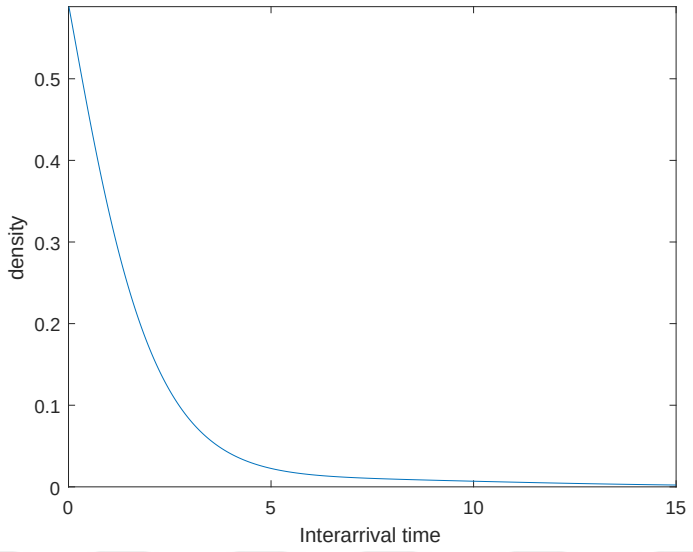

Figure 6.18: The analytical distribution of the demand inter-arrival times

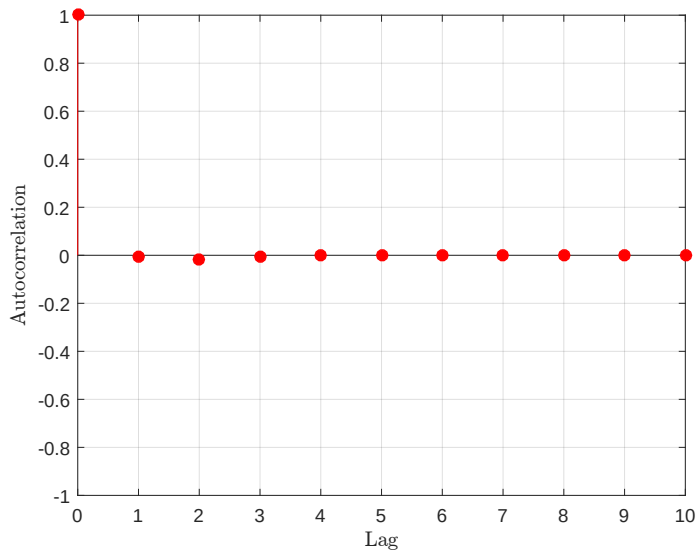

Figure 6.19: The analytical autocorrelation function of the demand inter-arrival times

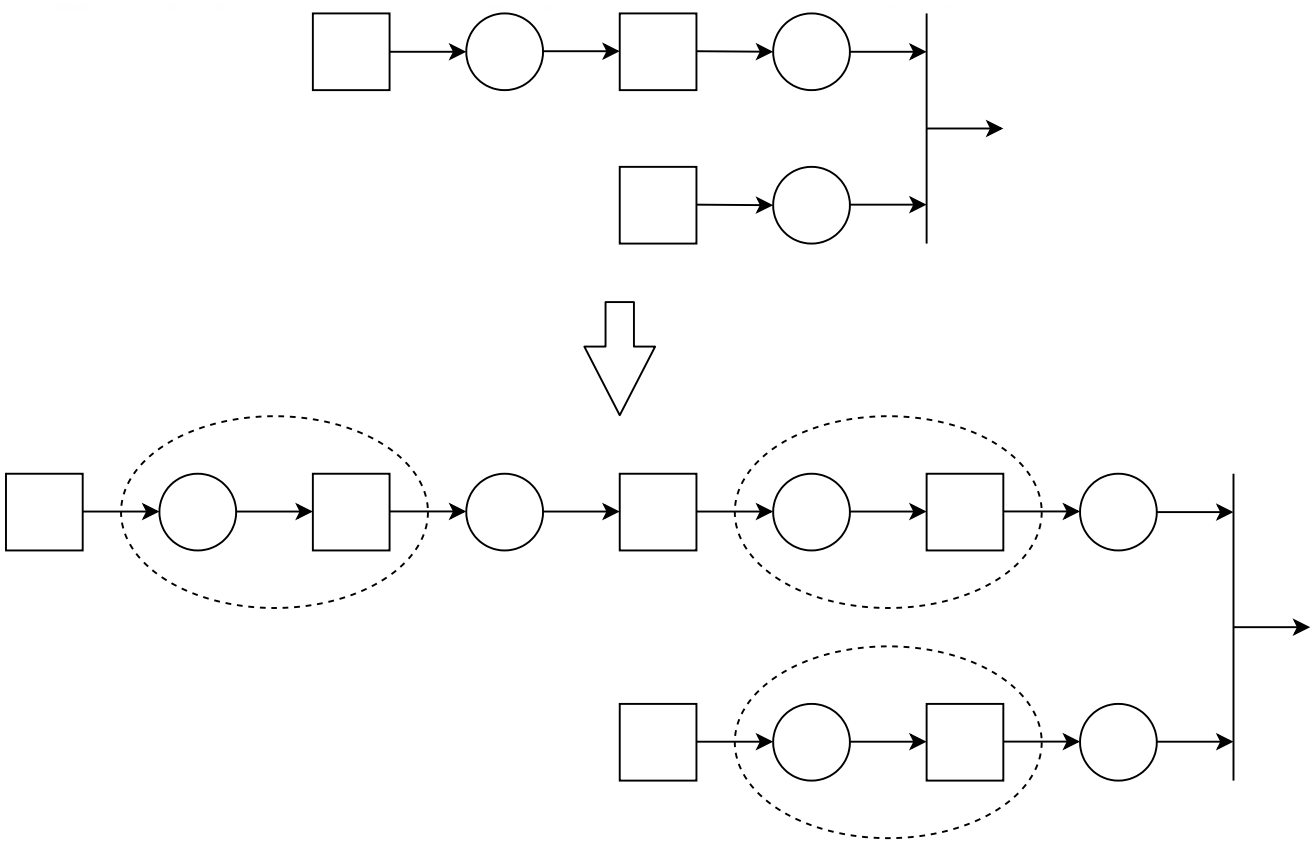

Figure 6.20: Modeling the time spent on transportation of material

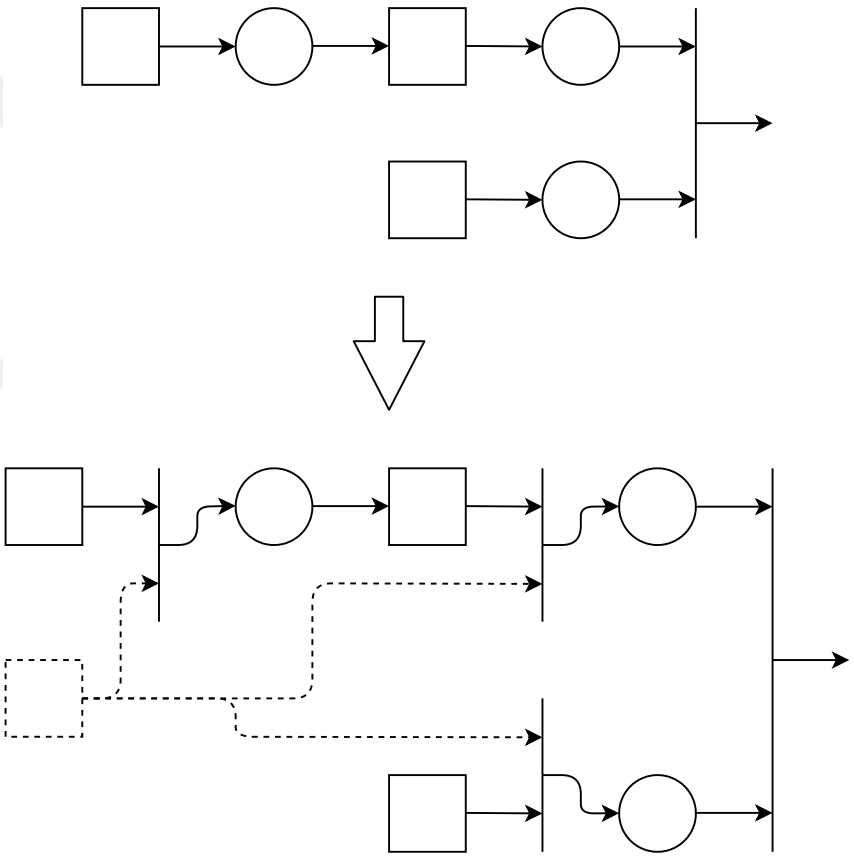

Figure 6.21: Modeling the time spent on centralized control

Chapter 7

CONCLUSION

This thesis investigates the data-driven control of partially observable manufacturing systems. The features of data-driven control include reducing mathematical assumptions on the system under study and the direct use of data in the control scheme.We propose using the marking-dependent threshold policy as an effective data-driven method to control production. We show that the parameters of the marking-dependent control policy can be determined with the joint simulation and optimization approach that combines the data collected from the shop floor with the available statistical information about the demand, production, and information arrival processes.

The marking-dependent threshold policy is easy to implement: a different threshold level is determined for each marking. When a marking is received, production of a new item is triggered if there is no item being produced at that moment and the inventory level is less than or equal to the threshold associated with that marking. The completion of a production triggers a production if the inventory level is less than the threshold associated with the last observed marking. If the inventory level reaches the threshold, the production stops until a new demand arrives, in which case it will stay idle if the newly observed marking corresponds to a threshold level lower than the inventory level.

We propose an analytical method based on the matrix geometric method for evaluating the performance of a production/inventory system controlled with the marking-dependent threshold policy where the demand, production, and information arrival

processes are modelled as Marked Markovian Arrival Processes. We introduce a MIP formulation for determining the optimal thresholds based on this analytical method.

For the data-driven JSO method, we give the mathematical programming representation of the problem as a mixed integer program. Given the considerable increase in the power of solvers in solving mathematical programming problems, the MIP formulation is given as the first step in devising efficient solution methods for the problem with shorter trace lengths. In addition, we give the discrete event simulation procedure for the system if longer traces are to be used to determine the parameters of the marking-dependent control policy.

We use an experimental setup to evaluate the performance of the marking-dependent threshold policy that uses different markings to control the release of material. We use this setup to determine the effects of using different markings and also to evaluate the JSO approach with simulated shop-floor data with different trace lengths.

Our numerical results show that the JSO-based approach is an effective method for controlling the system. With sufficiently long traces, the proposed method is able to use the information effectively and yields results that are close to the performance obtained with the analytical solution for the same information level. The value of information about the status of a machine increases as the variability of the repair times decreases and the repair time increases. The information about the machine status is more valuable than the information about its downstream buffer. The JSO-based approach is an efficient way for determining the optimal threshold levels given limited data. Furthermore, this method outperforms parametric methods that first fit distributions ignoring the autocorrelation of the demand process and then determine the control parameters based on the analytical model with the fitted parameters.

Next, we consider the optimization procedure for controlling more complicated production systems controlled based on the marking information. For the more gen-

eral setting, we consider marking-dependent policies which are generalizations of the marking-dependent threshold policy. We breakdown the problem of finding the optimal making-dependent policy to three levels: selecting the information sources, forming the markings and setting the policy parameters.

We formulate the problems of finding the optimal information sources and finding the optimal formation of markings for the marking-dependent policy as machine learning tasks, and through numerical experiments, we show how using machine learning can make these tasks easier and result in better policies. Given the specific structure of these problems, machine learning methods tailored to each specific problem can be developed based on clustering algorithms and multi-task learning methods. We leave the development of these methods for future research.

Finally, we implement the methods introduced in this thesis by using a Lego introduction system. This implementation shows how the specifics of the physical system can cause considerable disparities between the real system and the mathematical models of it and how data-driven control can mitigate the adverse effect of these disparities.

This thesis can be extended in different directions. The mathematical programming problems for finding the optimal marking-dependent thresholds can become more efficient by using the optimization methods such as benders decomposition and the cutting plane algorithm. It can be the case that there is knowledge that the demand process changes its behavior over time, but the markings are not available. Hence, in addition to finding out the best threshold levels, there is a need to label the arrivals with proper markings. Assigning markings to arrivals based on a series of recently observed inter-event times by using supervised learning can be a direction for extending this work. Here, we have used well-known supervised learning tools for selection of information sources and forming the markings, however, these problems have specific structures that can be leveraged in tailoring these machine learning to

these specific problems. Furthermore, multi-task learning can be employed for a more proper use of the evaluations of system instances with different system parameters. Another important question regarding learning from systems with different parameters is: *How can we learn from systems with different parameters and structures and apply the resulting knowledge to the target system?* In other words, the machine learning method should both predict the performance of specific selections of information sources and be able to determine how similar two production plants are despite having different number of elements and a different plant structure.

BIBLIOGRAPHY

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.

Akcay, A., Biller, B., and Tayur, S. (2011). Improved inventory targets in the presence of limited historical demand data. *Manufacturing & Service Operations Management*, 13(3):297–309.

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.

Alfieri, A. and Matta, A. (2012a). Mathematical programming formulations for approximate simulation of multistage production systems. *European Journal of Operational Research*, 219(3):773–783.

Alfieri, A. and Matta, A. (2012b). Mathematical programming representation of pull controlled single-product serial manufacturing systems. *Journal of Intelligent Manufacturing*, 23(1):23–35.

Arifoğlu, K. and Özekici, S. (2011). Inventory management with random supply and imperfect information: A hidden markov model. *International Journal of Production Economics*, 134(1):123–137.

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.

Bayraktar, E. and Ludkovski, M. (2010). Inventory management with partially observed nonstationary demand. *Annals of Operations Research*, 176(1):7–39.

Bertsimas, D. and Paschalidis, I. C. (2001). Probabilistic service level guarantees in make-to-stock manufacturing systems. *Operations Research*, 49(1):119–133.

Bertsimas, D. and Thiele, A. (2006). Robust and data-driven optimization: Modern decision-making under uncertainty. *INFORMS tutorials in operations research: models, methods, and applications for innovative decision making*, 3.

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.

Chan, W. K. and Schruben, L. W. (2004). Generating scheduling constraints for discrete event dynamic systems. In *Proceedings of the 2004 Winter Simulation Conference*, pages 568–576.

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.

Dizbin, N. and Tan, B. (2018). Modelling and analysis of the impact of correlated inter-event data on production control using markovian arrival processes.

Efron, B. (1992). Bootstrap methods: another look at the jackknife. In Kotz, Samuel, J. N. L., editor, *Breakthroughs in statistics*, pages 569–593. Springer.

Gallego, G. and Moon, I. (1993). The distribution free newsboy problem: review and extensions. *Journal of the Operational Research Society*, 44(8):825–834.

Gavish, B. and Graves, S. C. (1980). Technical note—a one-product production/inventory problem under continuous review policy. *Operations Research*, 28(5):1228–1236.

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

Helber, S., Schimmelpfeng, K., Stolletz, R., and Lagershausen, S. (2011). Using linear programming to analyze and optimize stochastic flow lines. *Annals of Operations Research*, 182(1):193–211.

Horváth, A., Horváth, G., and Telek, M. (2010). A joint moments based analysis of networks of map/map/1 queues. *Performance Evaluation*, 67(9):759–778.

Hosseini, B. and Tan, B. (2017). Simulation and optimization of continuous-flow production systems with a finite buffer by using mathematical programming. *IISE Transactions*, 49(3):255–267.

Hou, Z.-S. and Wang, Z. (2013). From model-based control to data-driven control: Survey, classification and perspective. *Information Sciences*, 235:3–35.

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

Jang, Y. J. and Yosephine, V. S. (2016). Lego robotics based project for industrial engineering education. *The International journal of engineering education*, 32(3):1268–1278.

Karaesmen, F., Liberopoulos, G., and Dallery, Y. (2004). The value of advance demand information in production/inventory systems. *Annals of Operations Research*, 126(1-4):135–157.

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.

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

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.

Lugaresi, G., Lin, Z., Frigerio, N., Zhang, M., and Matta, A. (2019a). Active learning experience in simulation class using a lego based manufacturing system. In *2019 Winter Simulation Conference*. 2019 Winter Simulation Conference.

Lugaresi, G., Travaglini, D., and Matta, A. (2019b). A lego manufacturing system as demonstrator for a real-time simulation proof of concept. In *2019 Winter Simulation Conference*. 2019 Winter Simulation Conference.

Lugaresi, G., Zanotti, M., Tarasconi, D., and Matta, A. (2019c). Manufacturing systems mining: Generation of real-time discrete event simulation models. In *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, pages 415–420. IEEE.

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.

Neuts, M. F. (1979). A versatile markovian point process. *Journal of Applied Probability*, pages 764–779.

Ost, A. (2013). *Performance of communication systems: A model-based approach with matrix-geometric methods*. Springer Science & Business Media.

Pedrielli, G., Alfieri, A., and Matta, A. (2015). Integrated simulation–optimisation of pull control systems. *International Journal of Production Research*, 53(14):4317–4336.

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.

Qi-Ming, H. and Neuts, M. F. (1998). Markov chains with marked transitions. *Stochastic processes and their applications*, 74(1):37–52.

Rudin, C. and Vahn, G.-Y. (2018). The big data newsvendor: Practical insights from machine learning.

Sanchez, A. and Bucio, J. (2011). Improving the teaching of discrete-event control systems using a lego manufacturing prototype. *IEEE Transactions on Education*, 55(3):326–331.

Schömig, A. K. and Mittler, M. (1995). Autocorrelation of cycle times in semiconductor manufacturing systems. In *Proceedings of the 27th conference on Winter simulation*, pages 865–872. IEEE Computer Society.

Schruben, L. W. (2000). Mathematical programming models of discrete event system dynamics. In *Proceedings of the 32nd conference on Winter simulation*, pages 381–385. Society for Computer Simulation International.

Settles, B. (2009). Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences.

Sobel, M. J. (1982). The optimality of full service policies. *Operations Research*, 30(4):636–649.

Song, J.-S. and Zipkin, P. (1993). Inventory control in a fluctuating demand environment. *Operations Research*, 41(2):351–370.

Syberfeldt, A. (2010). A lego factory for teaching simulation-based production optimization. In *Industrial Simulation Conference, ISC'2010, June 7-9, 2010, Ramada Plaza Hotel, Budapest, Hungary*, pages 89–94. EUROSIS-ETI.

Tan, B. (2015). Mathematical programming representations of the dynamics of continuous-flow production systems. *IIE Transactions*, 47(2):173–189.

Tan, B. and Lagershausen, S. (2017). On the output dynamics of production systems subject to blocking. *IISE Transactions*, 49(3):268–284.

Travaglini, D. (2018). Manufacturing system based on lego-robotics: Development of physical and digital models. Master's thesis, Politecnico Milano, Milan, Italy.

Treharne, J. T. and Sox, C. R. (2002). Adaptive inventory control for nonstationary demand and partial information. *Management Science*, 48(5):607–624.

Wein, L. M. (1988). Scheduling semiconductor wafer fabrication. *IEEE Transactions on semiconductor manufacturing*, 1(3):115–130.

Zhang, R. and Pavone, M. (2016). Control of robotic mobility-on-demand systems: a queueing-theoretical perspective. *The International Journal of Robotics Research*, 35(1-3):186–203.

A.1 PSMMAPGen: Automatic MMAP Generation Application for Production Systems

Given that using the analytical methods presented in this thesis requires the Marked MAP representation of production systems and there is no application that already generates these MMAPs, we have developed an easy to use application for automatic generation of MMAPs for user-specified production systems and information sources. Figure 1 depicts the interface of the application. In the following we give a description of the userinterface of the application and the different elements that can be used in the model of a production system using `PSMMAPGen`.

Using `PSMMAPGen` requires `Matlab Runtime 2016b` to be installed on the device. Using `Matlab Runtime 2016b` does not require a full installation of the `Matlab` software. Different versions of the `Matlab Runtime` program cannot be substitutes for each other. However, multiple versions can be installed on the same device. Two example files named `example_system_1.mat` and `example_system_2.mat`are included in the application for illustration purposes. Figure 2-3 depict these systems and their output autocorrelation function.

Figure 4 depicts the toolbar of the application. The first tool from the left is the `new file` tool that resets the window. The second and the third tools are the `open` and `save` tools that open and save using the file name and address specified in the `File name` field. The next three tools are the standard `pan`, `zoom in` and `zoom out` tools. The next tool is the `evaluate` tool. By clicking this tool the MMAP of the production system will be generated and the autocorrelation function of the MMAP will be plotted in the autocorrelation plot box. The next tool is the `stop` tool. This tool stops the evaluation of the system. The next tool is the `select` tool. Using `select` the properties of the elements of the production system can be modified. The next tool is the `erase` tool that removes an element and all its connections from the

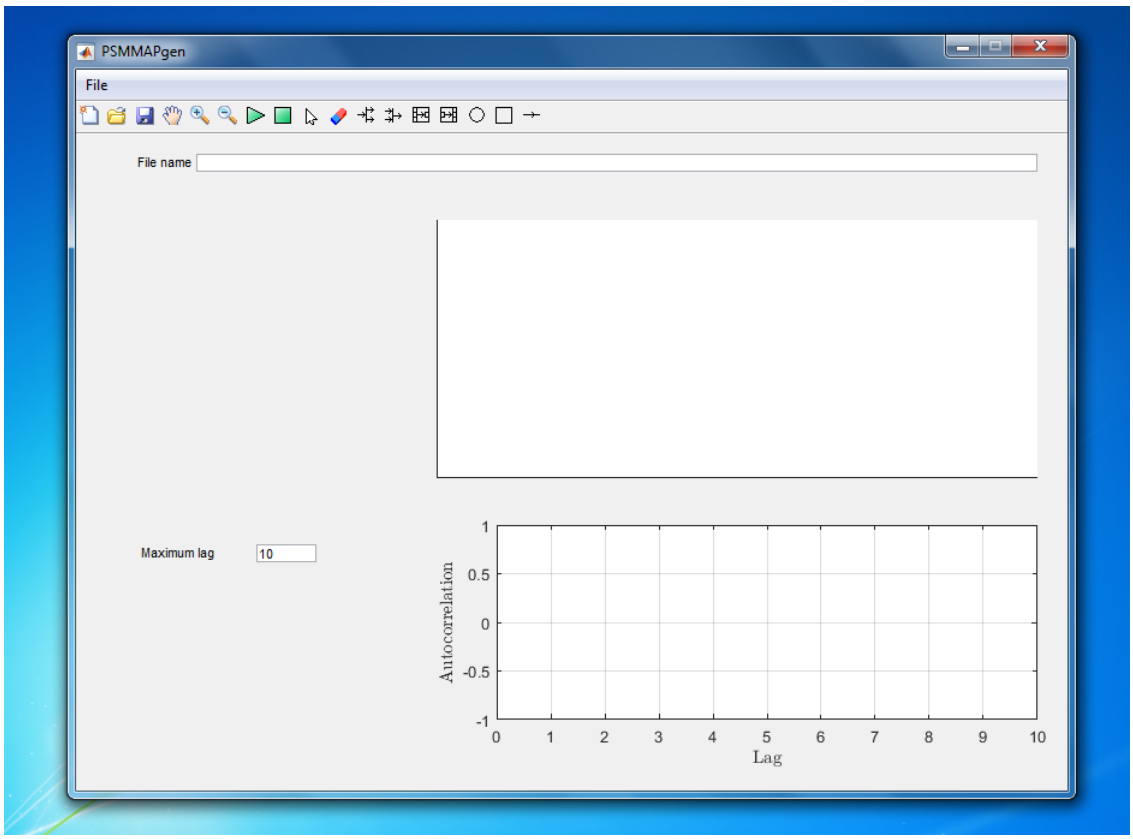

Figure 1: Screen-shot of the application

system. The next six tools are for adding new elements to the system. These tools are discussed in detail in the following. The last tools is the `connection` tool that adds connections between the elements of the system. The connections are directed and the direction reflects the flow of material in the system.

- The Tools for Adding New Elements
  - The `Machine` tool: the second tool on the toolbar from the right, the `Machine` tool adds machines to the system. Each machine has exponential processing times, and can be reliable or unreliable. A machine has the following editable properties. Figure A.1 depicts the fields related to the

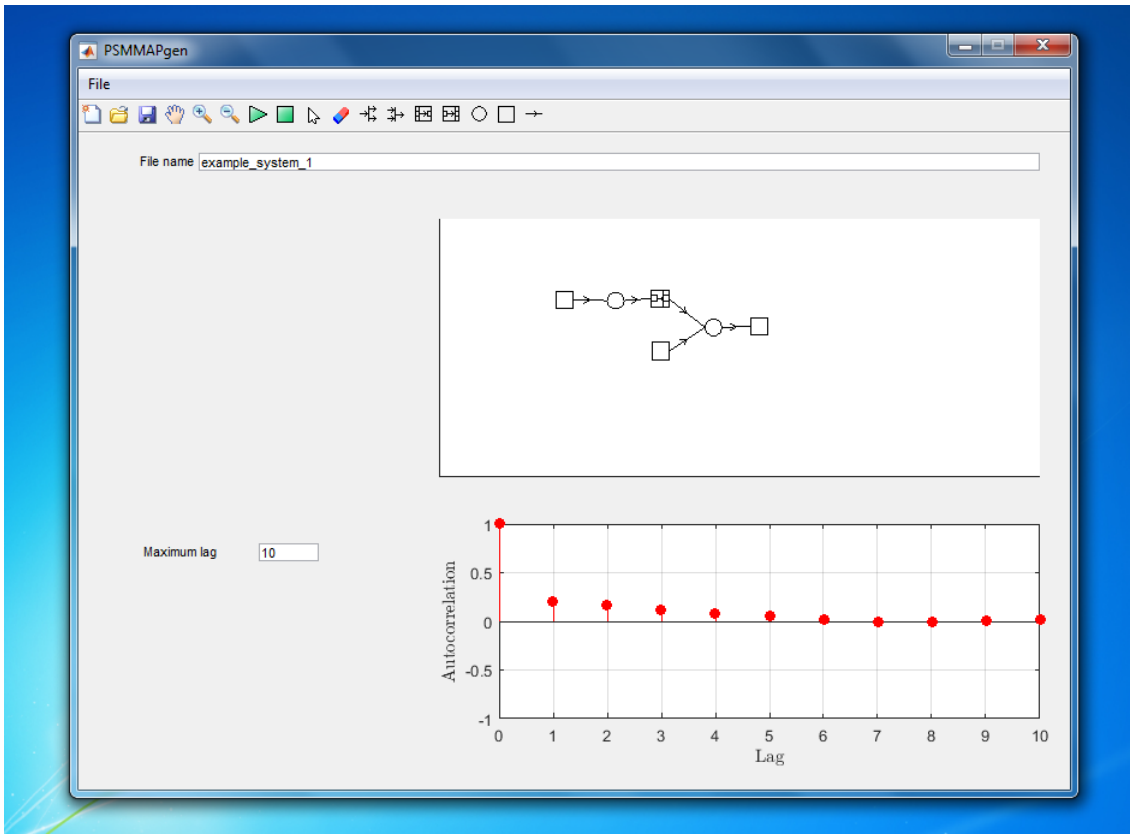

Figure 2: Screen-shot of the application's result for the example system from the file `example_system_1.mat`

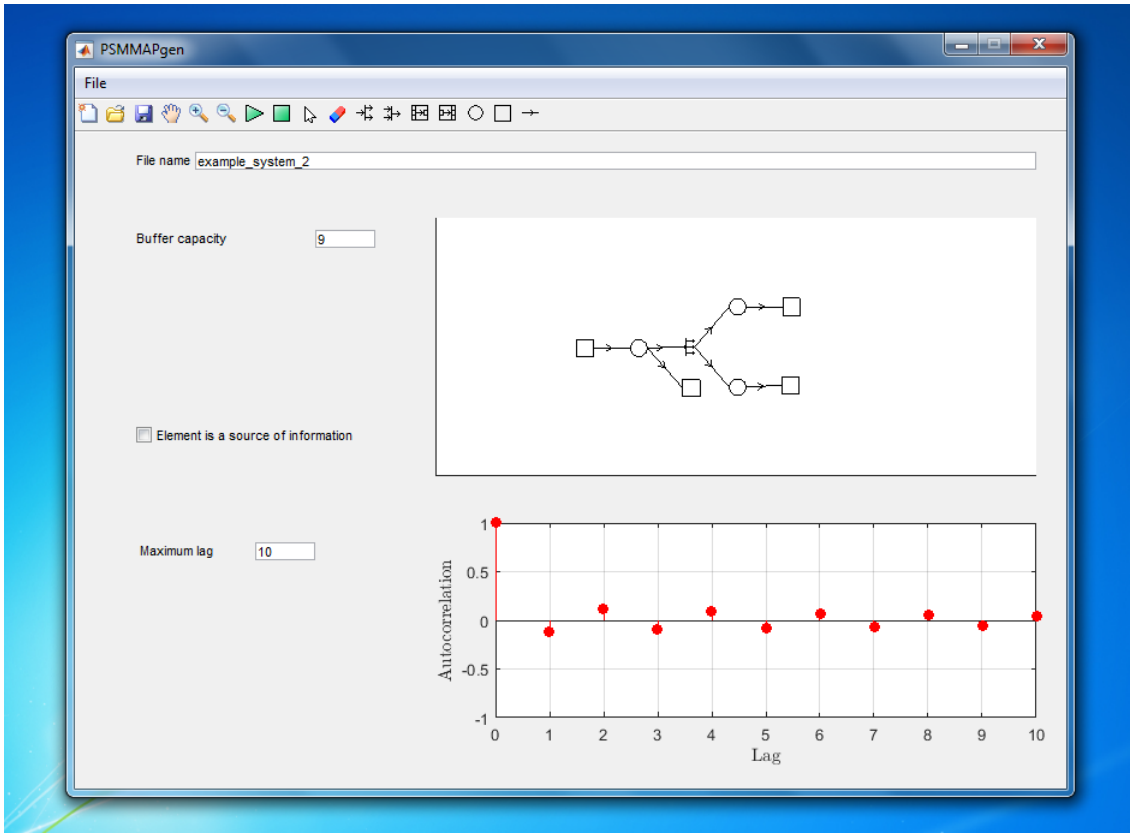

Figure 3: Screen-shot of the application's result for the example system from the file `example_system_2.mat`

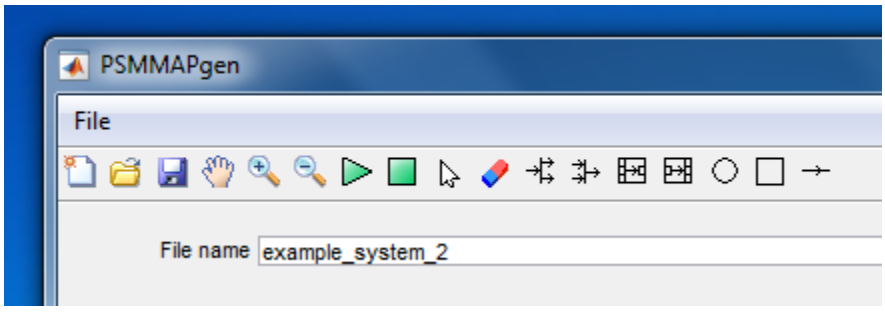

Figure 4: The toolbar of the application

properties of a machine element.

  * `Machine is unreliable`: This property can be "on" or "off"
  * `Processing rate`: This property should be a positive number
  * `Breakdown rate`: This property should be a positive number
  * `Repair rate`: This property should be a positive number. The repair times are exponentially distributed.
  * `Element is a source of information`: This property can be "on" or "off". If "on", all the possible states of the machine will be observable including the blocked and starved states.

– The `Buffer` tool: the third tool on the toolbar from the right, the `Buffer` tool adds buffers to the system. A buffer has the following editable properties.

  * `Buffer capacity`: This property should be a non-zero integer.
  * `Element is a source of information`: This property can be "on" or "off". If "on", all the possible states of the buffer will be observable.

– The `Unbatching station` tool: the fourth tool on the toolbar from the right, the `Unbatching station` tool adds unbatching stations to the system. An unbatching station takes a part in and instantaneously delivers multiple parts. An unbatching station has one editable property and that is the `Output batch size` property.

– The `Batching station` tool: the fifth tool on the toolbar from the right, the `Batching station` tool adds batching stations to the system. A batching station takes multiple parts in and instantaneously delivers multiple one part. A batching station has one editable property and that is the `Input batch size` property.

  - The `Synchronization station` tool: the sixth tool on the toolbar from the right, the `Synchronization station` tool adds synchronization/assembly stations to the system. A synchronization station can feed from multiple upstream buffers, once there is material in all the upstream buffers connected to it, the synchronization station takes a part in from each of the upstream buffers and instantaneously delivers one part. A synchronization station has no editable properties.
  - The `Disassembly station` tool: the sixth tool on the toolbar from the right, the `Disassembly station` tool adds disassembly stations to the system. A disassembly station can feed to multiple downstream buffers, once there is available space in all the downstream buffers connected to it, the disassembly station takes a part in and instantaneously delivers one part to each of its downstream buffers. A disassembly station has no editable properties.

- Features for Future Implementations:
  - Release control knobs
  - Changing exponential transition times to MAP transition times

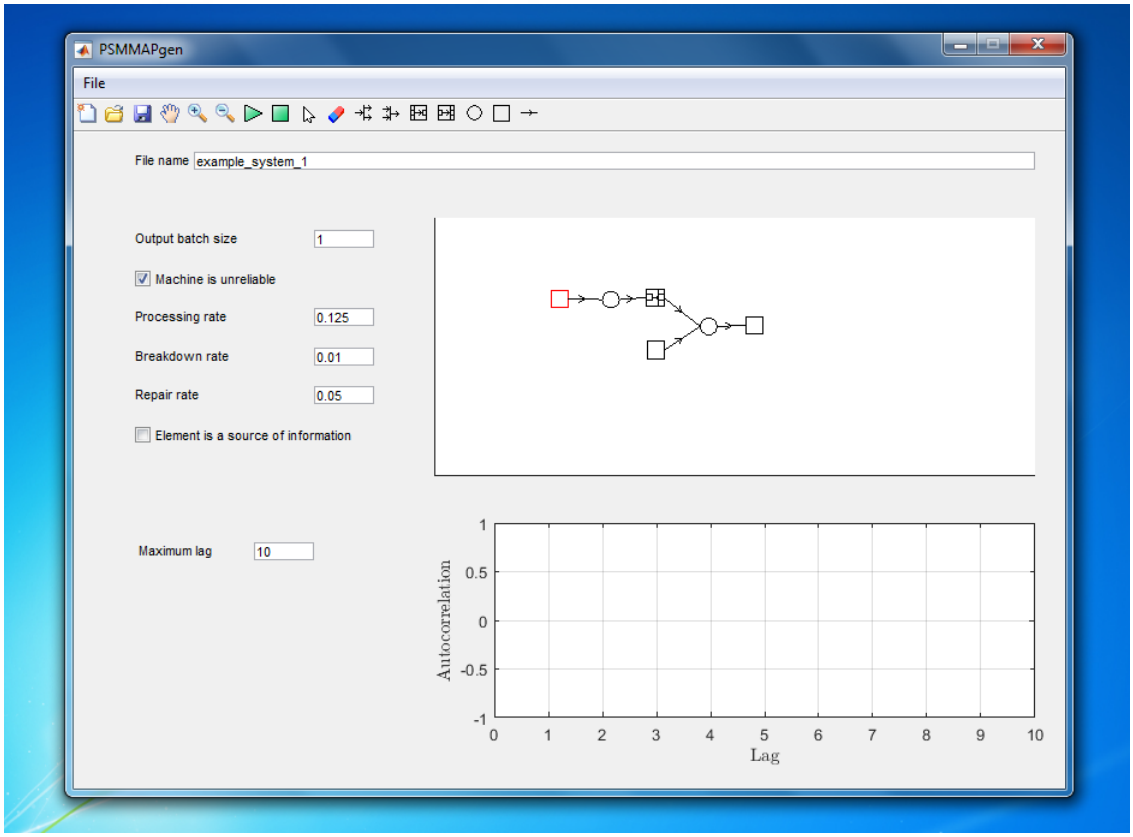

Figure 5: The properties of a machine element