Data-Enabled Modeling and Control for Smart Manufacturing Systems with Knowledge-Guided Machine Learning

---

A

Dissertation

Presented to

the faculty of the School of Engineering and Applied Science

University of Virginia

---

in partial fulfillment

of the requirements for the degree

Doctor of Philosophy

by

Jing Huang

May 2022

APPROVAL SHEET

This

Dissertation

is submitted in partial fulfillment of the requirements
for the degree of

Doctor of Philosophy

Author: Jing Huang

This Dissertation has been read and approved by the examing committee:

Advisor: Qing Chang

Advisor:

Committee Member: Haibo Dong

Committee Member: Tomonari Furukawa

Committee Member: Zongli Lin

Committee Member: Michael Brundage

Committee Member:

Committee Member:

Accepted for the School of Engineering and Applied Science:

Jennifer L. West, School of Engineering and Applied Science

May 2022

Acknowledgement

First and foremost, I would like to express my heartfelt gratitude to my advisor, Dr. Qing Chang, for her invaluable guidance and careful supervision over years. Without her long-standing support, it would not be possible for me to complete this journey. My appreciation also goes out to my collaborators Dr. Jianyu Su and Dr. Jianjing Zhang, and my Ph.D. advisory committee members Dr. Haibo Dong, Dr. Tomonari Furukawa, Dr. Zongli Lin, Dr. Peter Beling and Dr. Michael Brundage.

I also want to thank Dr. Jing Zou, Dr. Xinyan Ou, Mr. Tian Yu, Mr. Cheng Zhu, Ms. Chen Li, Mr. Kshitij Bhatta and all other current and previous members of Intelligent System Lab at Stony Brook University and University of Virginia for their assistance and friendship throughout my Ph.D. study. In addition, I would like to offer my special thanks to those who provided great assistance for my studies since my childhood, including Ms. Xueyu Lin and Ms. Ronglong Sun among many others.

And my biggest thanks to my family, especially my parents, for their tremendous support, love, and sacrifice. My lovely daughter Natalie was born one year before my graduation. She brings me so much joy and makes me a better parent and a better person every single day. I am forever grateful to my wife Yun, who stood by me all the time to share my achievements, happiness, and frustrations.

In loving memory of my grandfather Hengyun Huang

To my wife Yun Chen, my daughter Natalie Muge Huang, my parents, Shenyu Yu and Weifu Huang

Abstract

Manufacturing system is more interconnected and transparent with the deployment of distributed sensors and automatic machinery, as well as data storage and processing capabilities due to increasing availability of computing resources. Machine learning techniques are very promising in gaining useful insights from the huge volume of real-time data to facilitate system performance analysis and control decision makings. Despite of exciting advances in machine learning research and application in the past decade, it remains a challenging task to apply those techniques in the context of manufacturing industry. The expected improvements in productivity, quality and efficiency are still hampered by the salient gaps in real-time system modeling, system performance evaluation and prediction, and theory and algorithms for integrated decision making and optimization in the manufacturing domain.

In particular, reinforcement learning (RL) and multi-agent reinforcement learning (MARL), which aim at understanding the dynamics of the process/system and finding the optimization strategy through interactions with the environment, have opened up a new research avenue of the intended system performance enhancement without a rigid rulebook. However, manufacturing system is a complex engineering system with very high stochasticity and nonlinearity as well as great varieties in processes/products and scales. The system dynamics is deeply coupled with individual machines and processes, and constantly evolving due to not only internal factors, e.g., machine and process constraints, but also external circumstances like customer demands. This dissertation demonstrates a systematic way to use domain knowledge and systematic understanding of the manufacturing system to formulate typical control problems in RL/MARL framework in the manufacturing domain. In this dissertation, we start from analytical system modeling based on basic physics and then derive system properties, which are further used to guide

the problem formulation and algorithm implementation in a variety of significant prediction and control problems. The dissertation contributes to the body of research in manufacturing systems regarding the following aspects:

(1) A data-enabled system model for multi-product manufacturing system is established based on basic physic law, i.e., the conservation of the flow. The product-dependent cycle time and tool setup time are considered in the model. It closely connects the data collected from distributed sensors to the system states. The model shed lights on knowledge-guided machine learning problems formulation and solution.

(2) A hybrid framework combining deep learning and system modeling is developed to predict product completion time, which is critical to downstream tasks including production scheduling. Guided by system properties, a recurrent sequence in the prediction problem is discovered, and hence Long Short-Term Memory, a variant of Recurrent Neural Network, is applied.

(3) The preventive maintenance control problem is tackled using deep RL techniques. It demonstrates the formulation of manufacturing system control problems in the RL/MARL frameworks. By implementing both deep RL and deep MARL algorithms, it covers the preventive maintenance decision making in all spectra of manufacturing systems regarding the sizes and maintenance options.

(4) An innovative multi-agent control framework that integrates multiple levels of a manufacturing industry, including system level, process level and machine level, are proposed with the aim to optimize system performance considering both productivity and quality. The graph model and graph neural networks are applied to encode and integrate information across multiple levels and machines. Recursive Bayesian Estimation is applied to graph node feature engineering.

Contents

Chapter 1. Introduction

NIST (National Institute of Standards and Technology) defines smart manufacturing as “fully-integrated and collaborative manufacturing systems that respond in real time to meet the changing demands and conditions in the factory, supply network, and customer needs” [1]. It marks a historical stage of the development of modern manufacturing industry characterized by substantial innovations and changes driven by digitization, increased integration of sensors into production equipment, increasingly available data, and advances in robotics and automaton. The combination of these advances provides unprecedented opportunities to develop new and better ways of doing manufacturing [2]–[4]. However, the expected improvements in productivity, quality and efficiency are still hampered by the salient gaps in real-time system modeling, system performance evaluation and prediction, health management for manufacturing equipment and systems, and theory and algorithms for integrated decision making and optimization in the manufacturing domain.

On the one hand, there has been a notable lack of modeling method that reflects the real-time dynamics of manufacturing systems. A manufacturing system is a combination of humans, machinery, and equipment that are bound by a common material and information flow [5]. Such systems are inherently stochastic, complex, and dynamic, as different components closely interact with each other in a highly stochastic environment. Traditional system modeling methods, e.g., those based on Markov chain [6]–[8], are mostly based on simplified system structure and assumptions. These modeling methods are aimed to evaluate the long-run system performance of a manufacturing system given system parameters, e.g., machine cycle times and buffer capacities. They are usually applied to the system planning

and design stage. A manufacturing system suffers from random disruptions, such as a random machine downtime and material shortage, or control inputs, such as preventive maintenance actions and production scheduling. The wide deployment of the smart manufacturing technologies largely increased the system transparency by providing detailed data regarding these random disruptions and system component status in a real-time fashion [9]–[11]. A novel system modeling method must be developed to describe and track the real-time system dynamics, and form the basis for performance analysis, prediction, and effective control in today's smart manufacturing systems, despite the challenges posed by the increased complexity of the system structure and more diversified product types.

On the other hand, innovative and rigorous system modeling paves the way for acquiring meaningful systematic understandings and knowledge, which are fundamental to implementing knowledge-guided machine learning and reinforcement learning. In recent years, the availability of large datasets combined with the improvement in algorithms and the exponential growth in computing power led to an unparalleled surge of interest in the topic of machine learning and reinforcement learning [12]. Machine learning has also seen increasing utilization across all levels of the manufacturing system hierarchy [13]–[16]. However, compared with the successes of machine learning in specific applications of process monitoring, optimization, utilization is limited at the system level of analysis and decision-making. This is primarily attributable to the stochastic and non-linear dynamical nature of manufacturing systems and the complex multi-stage processes and dependencies among vast amounts of heterogeneous data generated therein. An in-depth understanding of a problem, its causes, consequences, and desired solution state must be known or well investigated to improve the likelihood of effective machine learning tool selection and subsequent model building, data analysis, and interpretation. These matters all deal with the need to have adequate domain expertise during the problem definition phase which is vital to ensure that all aspects of the problem are well understood, and no key data or assumption is overlooked [17]. Therefore, an innovative system modeling method will be established utilizing the sensor data. Based on the modeling and domain knowledge,

important topics including system performance evaluation/prediction and real-time control problems will be discussed in this dissertation. The remainder of this dissertation is organized as follows:

Chapter 2: An analytical modeling method for multi-product system based on conservation of the flow is established.

Chapter 3: A hybrid framework combining deep learning and system modeling is built to predict the product completion time in multi-product system.

Chapter 4: Rolling horizon method for corrective maintenance control.

Chapter 5: Preventive maintenance policies based on reinforcement learning and multi-agent reinforcement learning are proposed.

Chapter 6: An integrated manufacturing process-system control framework is proposed based on graph neural network and multi-agent reinforcement learning.

Chapter 7: Summary of scientific contributions, remarks on knowledge-guided machine learning and future work directions.

Chapter 2. Multi-Product System Modeling

2.1.Background

To meet the diverse customer demands, the manufacturing companies strive to expand their product lines. Rather than building separate and specific production lines for new product types, it is preferable to incorporate similar products into one common production line. On the one hand, the capital investment on equipment could be reduced tremendously by sharing common machines and tools in a multi-product line. On the other hand, multi-product line is more flexible in product throughputs, and thus can better cope with the fluctuations in market demands. Therefore, the multi-product system has been the dominant architecture of manufacturing system in many industries, including the automotive industry, the semi-conductor industry, etc.

A high-fidelity modeling for manufacturing is the foundation for systematic analysis, performance prediction, and effective control. In the past, there have been a lot of research efforts devoted to modeling the multi-product systems [18]–[20]. For example, Dasci & Karakul [18] adopt the decomposition approach for modeling a single-stage two-product system. But this modeling method is limited to single-stage systems, and cannot be generalized to multi-machine systems, in which machines have complicated interactions among each other, e.g., blockages and starvations. Sagawa et al. [19] modeled the multi-product system with apt analogy between production system and electronic system, based on which the state space equation for multi-product system is established. However, the modeling method has an underlying assumption that the buffer capacities are always infinite. It excludes a major portion of the production systems in the real industry, where the buffer capacity is often limited by spaces and costs. Another important realistic

consideration in the system modeling is the tool setup time. The multi-product system usually saves a lot of capital investment on machine and equipment since they can be shared among different product types [21]. It requires process engineers take considerable efforts to incorporate the process needs of all product types into shared equipment or tools. Despite this, sometimes it is unavoidable that some dedicated tools have to be designed for specific product types. In this case, the machine has to set up dedicated tools between two consecutive products if they are of different types, and the extra time used to change the tools are referred to as tool setup time. The setup time further adds to the complications in modeling the multi-product system, and therefore a lot of modeling methods [18], [19] just assume non-delay changeover between product types. Therefore, it is crucial to consider the complex interactions among machines due to various factors, including random machine failures and finite buffer capacities etc., as well as the tool setup time, only based on which can we further explore and understand the system properties.

2.2.System modeling

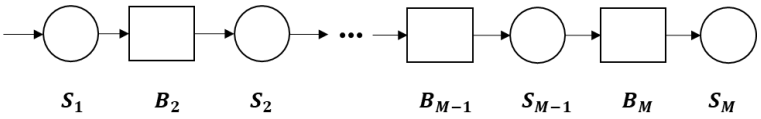


Figure 2.1. General structure of a serial production system

A manufacturing system as shown in Figure 2.1 is a stochastic dynamic system, which can be modeled by the state space equation:

$$\dot{\boldsymbol{b}}(t) = \boldsymbol{F}\big(\boldsymbol{b}(t), \boldsymbol{U}(t), \boldsymbol{W}(t)\big) \tag{2.1}$$

$$\boldsymbol{Y}(t) = \boldsymbol{H}\big(\boldsymbol{b}(t)\big) \tag{2.2}$$

The physical meanings of the variables and functions are as the following:

- $\boldsymbol{b}(t) = [b_2(t), b_3(t), \dots, b_M(t)]^T$ are the buffer levels at time t. The buffer level is treated as the system state in this model;

- $\boldsymbol{U}(t) = [u_1(t), u_2(t), \dots, u_M(t)]^T$ are the control inputs for machines at time t. In this research, $\boldsymbol{U}(t)$ are the control decisions of production scheduling for each machine;
- $\boldsymbol{W}(t) = [W_1(t), W_2(t), \dots, W_M(t)]^T$ are the random disturbances to the system. In this research, it refers to the machine random failures.

$$W_i(t) = \begin{cases} 1, \text{if machine } S_i \text{ is down at time } t \\ 0, \text{otherwise} \end{cases} \tag{2.3}$$

- $\boldsymbol{F}(*) = [F_2(*), F_2(*), \dots, F_M(*)]^T$ is the dynamic function for the system state;
- $\boldsymbol{Y}(t) = [Y_1(t), Y_2(t), \dots, Y_M(t)]^T$ is the system output at time t, where $Y_i(t)$ denotes the accumulated production count of machine S_i up to time t;
- $\boldsymbol{H}(*) = [H_1(*), H_2(*), \dots, H_M(*)]^T$ is the observation function, which relates the system state to the system output;

For multi-product lines, the system dynamics is heavily coupled with the production sequence. In this research, one product is represented with a $K \times 1$ vector $\boldsymbol{q}_n$, where $n \in \mathbb{Z}^+$ is the index of the product. The k^{th} entry is one if product $\boldsymbol{q}_n$ is of type k, and other entries are all zeros, i.e.

$$\boldsymbol{q}_n(k) = \begin{cases} 1, & \text{if the type of } n^{th} \text{ product is } k \\ 0, & \text{otherwise} \end{cases} \tag{2.4}$$

A product sequence is a matrix that assembles all the product in sequence. Let $\boldsymbol{Q}(t)$ denote the product sequence at time t, where

$$\boldsymbol{Q}(t) = [\boldsymbol{q}_1, \boldsymbol{q}_2, \dots, \boldsymbol{q}_n, \dots] \tag{2.5}$$

For example, the product sequence for a production line that processes three product types may have the following form:

$$\boldsymbol{Q} = \begin{bmatrix} 0 & 1 & 1 & 0 & \dots \\ 1 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 1 & \dots \end{bmatrix}$$

Based on this product sequence, the first product $\boldsymbol{q}_1 = [0, 1, 0]^T$ is of type two, the second and third products, denoted as $\boldsymbol{q}_2 = [1, 0, 0]^T$ and $\boldsymbol{q}_3 = [1, 0, 0]^T$

respectively, are both of type one, and the fourth product $\boldsymbol{q}_4 = [0, 0, 1]^T$ is of type three;

For every machine, it has a fixed cycle time when processing a particular product type. Therefore, the cycle time of a machine can be different when it is processing different types of products. The cycle time for machine S_i is represented as

$$\boldsymbol{T}_i = [T_{i1}, T_{i2}, \dots, T_{iK}]^T \tag{2.6}$$

where T_{ik}, $k = 1, 2, \dots, K$, is the cycle time of machine S_i when process a product of type k.

When one machine processes two consecutive products of different types, it might require additional time to set up dedicated tools for the incoming product. The additional time needed when switching product type is referred to as tool setup time. The setup time for machine S_i is denoted with a $K \times K$ matrix

$$\boldsymbol{T_i^{st}} = \left[T_{i,kl}^{st}\right]_{K\times K} \tag{2.7}$$

where $T_{i,kl}^{st} \geq 0$ is the tool setup time when machine S_i switches from product type k to type l. To be concrete, the setup time matrix $\boldsymbol{T}_i^{st}$ has the following form.

$$\boldsymbol{T}_i^{st} = \begin{bmatrix} 0 & T_{i,12}^{st} & \cdots & T_{i,1K}^{st} \\ T_{i,21}^{st} & 0 & \cdots & T_{i,2K}^{st} \\ \vdots & \vdots & \ddots & \vdots \\ T_{i,K1}^{st} & T_{i,K2}^{st} & \cdots & 0 \end{bmatrix} \tag{2.8}$$

The tool setup time representation proposed in this paper implies that the setup time can be ***sequence-dependent***, as $T_{i,kl}^{st}$ and $T_{i,lk}^{st}$ are not necessarily equal, i.e., tool setup time for machine S_i switching from product type k to type l can be different from switching product types of the other way.

The cycle time of machine S_i can be calculated as $\boldsymbol{q}_n^T \boldsymbol{T}_i$, when machine S_i is processing product $\boldsymbol{q}_n$. In addition, there could be a setup time if product $\boldsymbol{q}_n$ and $\boldsymbol{q}_{n-1}$ are of different types. Leveraging the setup time representation, the setup time can be conveniently calculated as $\boldsymbol{q}_{n-1}^T \boldsymbol{T}_i^{st} \boldsymbol{q}_n^T$. The processing time needed for

product $\boldsymbol{q}_n$, i.e., the actual cycle time, is the summation of the original cycle time $\boldsymbol{q}_n^T \boldsymbol{T}_i$ and the setup time $\boldsymbol{q}_{n-1}^T \boldsymbol{T}_i^{st} \boldsymbol{q}_n$.

2.3.Model derivation based on conservation of flow

The dynamic function is derived based on the conservation of the flow. At any time t, the production counts of any two machines S_i and S_j, $\forall i \neq j$, satisfy:

$$Y_i(t) - Y_j(t) = \begin{cases} \sum_{k=i+1}^{j} b_k(t) - \sum_{k=i+1}^{j} b_k(0), & i < j \\ \sum_{k=j+1}^{i} b_k(0) - \sum_{k=j+1}^{i} b_k(t), & i > j \end{cases} \tag{2.9}$$

The production difference cannot exceed a certain boundary β_{ij}, where β_{ij} is the condition that the buffer levels between two machines are full ($i < j$) or empty ($i > j$).

$$\beta_{ij} = \begin{cases} \sum_{k=i+1}^{j} B_k - \sum_{k=i+1}^{j} b_k(0), & i < j \\ \sum_{k=j+1}^{i} b_k(0), & i > j \end{cases} \tag{2.10}$$

When $Y_i(t) - Y_j(t) < \beta_{ij}$, machine S_j does not constrain machine S_i; when $Y_i(t) - Y_j(t) = \beta_{ij}$, machine S_i is referred to be starved (if $i > j$) or blocked (if $i < j$) by machine S_j. Let $\theta_i(t), i = 1,2,\dots,M$, denotes the operational status of machine S_i at time t. Since we only consider machine random failures in this research, we have

$$\theta_i(t) = 1 - W_i(t) \tag{2.11}$$

Let $\boldsymbol{\zeta}(t) = \left[\zeta_{ij}(t)\right]_{M \times M}$ be a matrix used to indicate the interactions among machines:

$$\zeta_{ij}(t) = \begin{cases} 1, & if\ Y_i(t) - Y_j(t) = \beta_{ij}, i \neq j \\ \infty, & otherwise \end{cases} \tag{2.12}$$

It is noted that machine S_i has to operate at the processing speed of machine S_j if S_i is constrained by S_i.

$$v_i(t) = \min\left\{\frac{\zeta_{ij}(t)\theta_j(t)}{T_j(t)}, \frac{\theta_i(t)}{T_i(t)}\right\} \tag{2.13}$$

where $T_i(t)$ is the *actual cycle time* of machine S_i at time t. In a single-product system, the cycle time is a fixed value for each machine, however, in a multi-product system the cycle time of a machine is dependent on the product type it is processing.

To determine the actual cycle time of one machine at a particular time t, one just needs to identify the product index n that the machine is processing at that time point. Let $n_i(t)$ be the index of the product that machine S_i is processing at time t, then we have

$$n_i(t) = \left\lfloor Y_i(t) + \sum_{j=i+1}^{M} b_j(0) \right\rfloor + 1 \tag{2.14}$$

where $\lfloor * \rfloor$ is the floor operator, e.g., $\lfloor 3.14 \rfloor = 3$, and $b_j(0)$ is the initial buffer level of buffer B_j. Therefore, the cycle time of machine S_i at time t is

$$T_i(t) = \boldsymbol{q}_{n_i(t)}^T \boldsymbol{T}_i + \boldsymbol{q}_{n_i(t)-1}^T \boldsymbol{T}_i^{st} \boldsymbol{q}_{n_i(t)} \tag{2.15}$$

We can extend Eq. (2.15) by considering the interactions between machine S_i and all the other machines in the line.

$$v_i(t) = \min\left\{\frac{\zeta_{i1}(t)\theta_1(t)}{T_1(t)}, \frac{\zeta_{i2}(t)\theta_2(t)}{T_2(t)}, \dots, \frac{\theta_i(t)}{T_i(t)}, \dots, \frac{\zeta_{iM}(t)\theta_M(t)}{T_M(t)}\right\} \tag{2.16}$$

The change rate of $b_i(t)$ is the speed difference between its downstream machine S_i and upstream machine S_{i-1}.

$$\dot{b_i}(t) = v_i(t) - v_{i-1}(t) = F_i\big(\boldsymbol{b}(t)\big) \tag{2.17}$$

The accumulated production count of machine S_i is

$$Y_i(t) = \int_0^t v_i(\tau) d\tau = H_i\big(\boldsymbol{b}(t)\big) \tag{2.18}$$

Thus, the dynamic function $\boldsymbol{F}(*)$ and the measurement function $\boldsymbol{H}(*)$ are derived for the multi-product serial production line. With this model, the system states and other important variables at any time point can be obtained as long as the production sequence $\boldsymbol{Q}(t)$ and machine random failures $\boldsymbol{W}(t)$ up to that time point is known.

2.4.System identification use case

There are a lot of potential applications of the derived mathematical model for multi-product system, including product completion time prediction in Chapter 3. In this chapter, a simple example on system identification will be demonstrated. Based on the modeling method proposed in the preceding section, we will conduct the analysis on the performance of a two-machine-one-buffer system with two product types. The tool setup time is not considered in this demonstrated case. The system architecture is as shown in Figure 2.2.

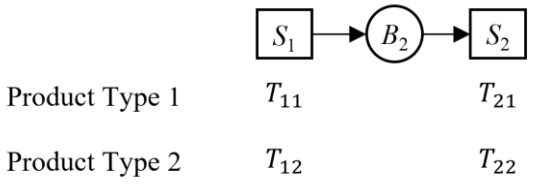


Figure 2.2. A simple two-machine-two-product system

For single-product systems, people usually take the production counts within a given period of time, e.g., one shift, one week, etc., as the performance measure. However, for multi-product systems, the production counts cannot serve as a justified performance measure, since during a given time period, the production count varies with product types.

To find a proper measure for multi-product system, we should take a look into the daily operation of a manufacturing system. The total number of a particular product type to be produced is not arbitrary. The production manager would receive a specific production order from the production planning department. The production manager aims to arrange the production sequence such that the order

could be processed efficiently. Therefore, the total time it takes to finish all the orders could be a direct measure of the efficiency.

Suppose the production order is $\{N_1, N_2\}$, which means N_1 products of type 1 and N_2 products of type 2 are demanded, then the total number of the products to be produced is

$$N = N_1 + N_2 \tag{2.19}$$

The performance measure for the two-machine-one-buffer system is the order finish time, denotes as T.

$$T = \inf\{t | Y_2(t) = N\} \tag{2.20}$$

The order finish time T highly depends on the production sequence $\boldsymbol{Q}$ and the system architecture. The exact relationship among them has yet to be investigated. In this section, we will give an upper bound and a lower bound for the order finish time.

$$T_L \leq T \leq T_U \tag{2.21}$$

where

$$T_L = \max_{i=1,2}\{N_1 T_{i1} + N_2 T_{i2}\} \tag{2.22}$$

and

$$T_U = N_1 \max_{i=1,2}\{T_{i1}\} + N_2 \max_{i=1,2}\{T_{i2}\} \tag{2.23}$$

The upper bound is derived by assuming that the buffer capacity is zero. Then the two machines are aggregated to one virtual machine with cycle times $\max_{i=1,2}\{T_{i1}\}$ for product type 1 and $\max_{i=1,2}\{T_{i2}\}$ for product type 2. Then the total time it takes to finish N_1 product type 1 and N_2 product type 2 is $N_1 \max_{i=1,2}\{T_{i1}\} + N_2 \max_{i=1,2}\{T_{i2}\}$.

From the perspective of manufacturing process, the cycle time is the least time for one machine to finish all the process on a single product. The essential time for machine S_i to finish all the processes on N_1 product 1 and N_2 product 2 is $N_1 T_{i1} +$

N_2T_{i2}. Regardless the system architecture and production sequence, the order finish time can never be less than $\max_{i=1,2}\{N_1T_{i1} + N_2T_{i2}\}$, which yields the lower bound.

These bounds would provide the production manager with a quick reference for the order finish time. More importantly, the lower bound T_L renders a clear target of order finish time for the production manager, who endeavors to arrange the production sequence $\boldsymbol{Q}$ to achieve a better performance.

Definition 2.1. *For a two-machine-one-buffer production line with intermediate buffer capacity B_2, given the production order $\{N_1, N_2\}$ and the cycle times $\boldsymbol{T}_1$ and $\boldsymbol{T}_2$, if there exists a production sequence $\boldsymbol{Q}$, such that the order finish time T achieves the lower bound T_L, i.e. $T = T_L = \max_{i=1,2}\{N_1T_{i1} + N_2T_{i2}\}$, then the production sequence $\boldsymbol{Q}$ belongs to the best sequences.*

Based on the definition, the best sequence might not be unique. The problem of interest is that given the system architecture and the production order, whether or not the best production sequences exist.

Let η_1 and η_2 be the ratio of product 1 and 2 in the production order, i.e., $\eta_1 = N_1/N$ and $\eta_2 = N_2/N$. Then T_L can be rewritten as:

$$T_L = N \cdot \max_{i=1,2}\{\eta_1T_{i1} + \eta_2T_{i2}\} \tag{2.24}$$

The machine corresponds to T_L is said to be the slowest machine for the production order $\{N_1, N_2\}$, denoted as S_{M^*}, where

$$M^* = \arg\max_{i=1,2}\{\eta_1T_{i1} + \eta_2T_{i2}\} \tag{2.25}$$

This is closely analogous to the slowest machine in the single-product system. In a single-product system, the system capacity is directly determined by the slowest machine and any stoppage of the slowest machine would lead to permanent production loss [22], [23]. Similarly, in multi-product scenario, the slowest machine S_{M^*} should also always process the products at its rated speeds, otherwise, the best system performance, i.e., lower bound of order finish time T_L, can never be achieved.

Therefore, a production sequence belongs to the best sequences only when the slowest machine S_{M^*} is not constrained, i.e., blocked or starved, during the processing of the whole production order.

Note that the slowest machine S_{M^*} in multi-product systems is not necessarily the slowest machine for a particular product type. Machine $S_{M_1^*}$ and $S_{M_2^*}$ are said to be the slowest machine for product type 1 and type 2 respectively, where $M_1^* = \arg\max_{i=1,2}\{T_{i1}\}$ and $M_2^* = \arg\max_{i=1,2}\{T_{i2}\}$.

There are three cases regarding the relationship among the slowest machines for product types and that for the production order, see Table. 1. Note that the situation where $M_1^* = M_2^* \neq M^*$ (the last two columns) does not exist.

Table 2.1. Different scenarios for slowest machine locations

Slowest Machines	M^*	M_1^*	M_2^*
Case 1	1	1	1
	2	2	2
Case 2	2	1	2
	2	2	1
Case 3	1	2	1
	1	1	2
Not Exist	2	1	1
	1	2	2

In the following, we will conduct case-by-case discussions on the relationship between the existence of the best sequences and the system architecture.

Case 1: $M^* = M_1^* = M_2^*$

In this case, the slowest machine for the production order is the same to the slowest machine for each product type. The lower bound T_L can be rewritten as $T_L = \max_{i=1,2}\{N_1 T_{i1} + N_2 T_{i2}\} = N_1 T_{M^*1} + N_2 T_{M^*2}$. The upper bound T_U can be

rewritten as $T_U = N_1 \max_{i=1,2}\{T_{i1}\} + N_2 \max_{i=1,2}\{T_{i2}\} = N_1 T_{M_1^*1} + N_2 T_{M_2^*2} = N_1 T_{M^*1} + N_2 T_{M^*2}$.Compare upper and lower bounds, we have $T = T_L = T_U$. It is noted that the above derivation is independent of the production sequence $\boldsymbol{Q}$. Therefore, in Case 1, any production sequence is the best sequence. In clean case scenario, the system performance is exactly the same regardless of the production sequence and buffer capacity.

Case 2: $\boldsymbol{M_1^* \neq M_2^*}$ and $\boldsymbol{M^* = 2}$

Without loss of generality, we conduct the analysis based on $M^* = M_2^* = 2$ and $M_1^* = 1$. In this case, the slowest machine, i.e., machine S_2, should never be starved in order to achieve a best sequence.

The simple fact is that feeding product 2 to the system would help accumulate the buffer level, which is preferable, and feeding product 1 would drain the buffer, which leads to starvation when the buffer is empty.

To avoid such starvation, at time $t = 0$, the first product fed to the system should be product 2, which helps accumulate the buffer level. After the buffer reaches a desirable level, product type 1 can be fed to the system such that the starvation would not arise.

Let γ be the critical number of product 2 fed to the system, such that at least one product of type 1 can be processed without causing starvation. Together they form a small product bundle (Figure 2.3). The buffer level goes back to zero exactly when the bundle is finished. And then another bundle can be fed to the system.

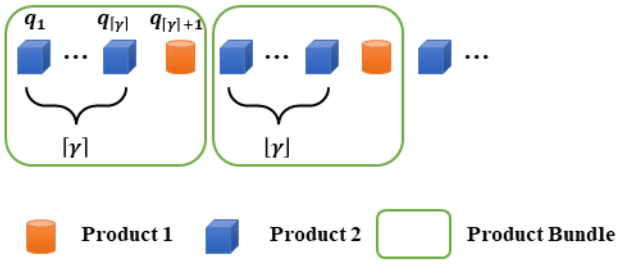


Figure 2.3. Bundled sequence for case 2

Now we try to find the critical value γ. The buffer level after feeding γ products 2 is $\gamma T_{12}(1/T_{12} - 1/T_{22})$. The critical condition is that the time for machine S_1 to process one product of type 1 should be equal to that for machine S_2 to process all the remaining product 2 in the buffer and one product of type 1, i.e.,

$$T_{11} = \gamma T_{12}\left(\frac{1}{T_{12}} - \frac{1}{T_{22}}\right) \cdot T_{22} + T_{21} \tag{2.26}$$

Solve the equation for the value of γ.

$$\gamma = \frac{T_{11} - T_{21}}{T_{22} - T_{12}} \tag{2.27}$$

There are two things to note. Firstly, there is a minimum requirement for the ratio of product 2 in the order to ensure that we can pair every one product 1 with at least γ product 2, i.e.,

$$\eta_2 : \eta_1 \geq \gamma : 1 \tag{2.28}$$

Insert Eq. (2.27) into Eq. (2.28), yields

$$\eta_1 T_{21} + \eta_2 T_{22} \geq \eta_1 T_{11} + \eta_2 T_{12} \tag{2.29}$$

Actually, this inequality always holds, since machine S_2 is the slowest machine. Therefore, the requirement for the product ratio is always fulfilled.

Additionally, during the processing of the bundle, there is a maximum value of the buffer level, which can be interpreted as the requirement for the buffer capacity. In other words, if the buffer capacity is lower than the maximum buffer level during the processing of the bundle, then product 1 would always cause starvation no matter how many product 2 has been fed to the system before it.

If $T_{11} \geq T_{22}$, the buffer level starts to decline once product 1 has been fed to the system. The peak level is the buffer level after γ product 2 have been fed to the system. Since γ might not always be an integer, we need to round it to its ceil.

$$B_2 \geq \left\lceil \frac{T_{11} - T_{21}}{T_{22} - T_{12}} \right\rceil \frac{T_{22} - T_{12}}{T_{22}} \tag{2.30}$$

If $T_{11} < T_{22}$, the buffer level continues to grow when product 1 is fed to the system and starts to decline only when all the remaining product 2 has left the buffer.

$$B_2 \geq \lceil \gamma \rceil \frac{T_{22} - T_{12}}{T_{22}} + \left(\frac{1}{T_{11}} - \frac{1}{T_{22}} \right) \lceil \gamma \rceil \frac{T_{22} - T_{12}}{T_{22}} \tag{2.31}$$

Simplify the equation, we have

$$B_2 \geq \left\lceil \frac{T_{11} - T_{21}}{T_{22} - T_{12}} \right\rceil \frac{T_{22} - T_{12}}{T_{11}} \tag{2.32}$$

Combine Eq. (2.31) and Eq. (2.32), the best sequence exists only if the buffer capacity meets the following requirement

$$B_2 \geq \left\lceil \frac{T_{11} - T_{21}}{T_{22} - T_{12}} \right\rceil \frac{T_{22} - T_{12}}{\min\{T_{11}, T_{22}\}} \tag{2.33}$$

In conclusion, for Case 2, the best production sequence exists if the buffer capacity satisfies the above condition. If the condition is satisfied, then one of the best production sequences is as shown in Figure 2.3. We call the production sequence as smallest bundle sequence, since you cannot further reduce the number of each product types in the bundle. The smallest bundle sequence is the one that requires the least buffer capacity to be best sequence.

Case 3: $M_1^* \neq M_2^*$ and $M^* = 1$

Similarly, we only conduct the analysis based on $M^* = M_2^* = 1$ and $M_1^* = 2$. In order to achieve the best sequence, the slowest machine, i.e., machine S_1, should never be blocked.

If the buffer capacity is infinite, then machine S_1 would never be blocked, and any production sequence is the best sequence. In contrast, if the buffer capacity is zero, then S_1 is blocked whenever product 1 is fed to the system and the best sequence does not exist. Therefore, there must also exist a critical buffer capacity in this case, which directly determines the existence of the best production sequences.

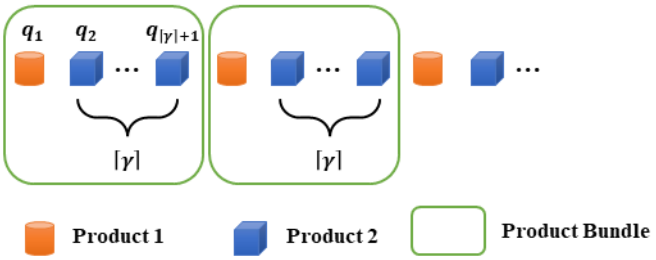


Figure 2.4. Bundled sequence for case 3

Again, we try to process the products with the smallest bundle, since it requires the least buffer capacity. The bundle contains only one product 1 and γ product 2 (Figure 2.4). The buffer level after feeding product 1 is $T_{11}(1/T_{11} - 1/T_{21})$. And γ is the critical number of product 2 fed to the system after product 1 such that the buffer level resumes empty. Following the same procedure, we will have

$$\eta_1 T_{11} + \eta_2 T_{12} \geq \eta_1 T_{21} + \eta_2 T_{22} \tag{2.34}$$

This inequality always holds since machine S_1 is the slowest machine. Therefore, the existence of the best production sequence is always independent of the specific production order.

Similar to the previous case, the buffer capacity should be larger than the maximum buffer level during the processing of the smallest bundle. If $T_{12} \geq T_{21}$, the buffer starts to decline once product 2 is fed to the system.

$$B_2 \geq \frac{T_{21} - T_{11}}{T_{21}} \tag{2.35}$$

If $T_{12} < T_{21}$, the buffer level continues to grow when product 2 is fed to the system and starts to decline only when all the remaining product 1 has left the buffer.

$$B_2 \geq \frac{T_{21} - T_{11}}{T_{21}} + \left(\frac{1}{T_{12}} - \frac{1}{T_{21}}\right) \cdot \left(\frac{T_{21} - T_{11}}{T_{21}}\right) \cdot T_{21} \tag{2.36}$$

Simplify the above equation, yields

$$B_2 \geq \frac{T_{21} - T_{11}}{T_{12}} \tag{2.37}$$

We combine Equation (40) and (42), then the requirement for the existence of the best production sequence is:

$$B_2 \geq \frac{T_{21} - T_{11}}{\min\{T_{12}, T_{21}\}} \tag{2.38}$$

Therefore, for Case 3, the only condition for the existence of best production sequences is the buffer capacity requirement in Eq. (2.38).

2.5. Summary

In this chapter, an analytical system model for multi-product system is proposed. The product-dependent cycle time and tool set-up time are considered in the model. The model is analytically derived based on the conservation of the flow. It provides a solid foundation for system performance prediction and control. A use case for analyzing the best sequence in two-machine-two-product system is also demonstrated in this chapter.

2.6. Related work

Part of the results presented in this chapter have been published in [24]–[29].

Chapter 3. Product Completion Time Prediction

3.1.Background

Product completion time (PCT), sometimes also referred to as makespan, remaining time, lead time or flow time, is the time needed to complete the required manufacturing processes on the product. In manufacturing systems, the accurate prediction of PCT is fundamental to production scheduling [30]–[32]. The latest evolvements of manufacturing systems have posed new challenges in PCT prediction. On the one hand, driven by intense market competitions, notable increases in product diversity and product system complexity have made it more difficult to predict PCT. On the other hand, with the rise of Manufacturing as a Service (MaaS) [33], [34], PCT is no longer just an internal reference for production manager to schedule productions, but also has become a major commitment to customers. Therefore, effective PCT prediction is much needed in the context of today's smart and customer-oriented manufacturing.

The existing methods for PCT prediction could be categorized into three categories, namely, analytical, simulation and data-driven methods. The analytical methods are mostly built on stochastic process models, e.g., queueing models and Markov Chain etc. For example, Altendorfer & Jodlbaure [35] derive the expected PCT in M/M/1 production systems using queueing theory, where M/M/1 denotes a single-machine production system whose processing time and time between order arrivals both follow exponential distribution. Savasaneril et al. [36] discusses the lead time quotation problem in a similar production system but with the consideration of inventory level. The applications of analytical methods are mostly

limited to simple systems with strict assumptions. The production system investigated in this paper is the multi-product serial production line, which is known to have extremely complex, non-linear and stochastic dynamics in nature. Rajagopalan & Karimi [37] manages to analytically derive the PCT in multi-product serial production line considering transfer time and setup time, but only by assuming there is no stochastic factors.

Unlike analytical methods, simulation methods [38]–[41] should work for most types of production systems as long as a reliable simulator can be established. Hubl et al. [38] introduces a flexible discrete event simulation model for PCT prediction considering stochastic factors in processing times, tool setup times and purchasing lead times. In [40], a method based on Petri Net is proposed to predict PCT in flexible manufacturing systems. In essence, the simulation based PCT prediction relies on large numbers of repetitive simulations to obtain sufficient amounts of random samples, so as to compute the expected PCT. Consequently, the nuisance of simulation methods is that it requires long computing time and enormous computing resources, which prohibits the simulation methods from being used in real-time production scheduling scenarios.

The data-driven methods [42]–[45] take advantages of the historical data to discover the hidden patterns mapping from input observations to PCT with statistical learning techniques. For example, Lingitz et al. [44] conducts a case study on PCT prediction using real data from a semiconductor production system. The raw data are lumped into several different regression models, including linear model, Random Forest, Support Vector Machine and K-Nearest Neighbors etc., in order to find the optimal model with the highest prediction accuracy. Recently, researchers have also attempted to apply deep learning techniques to PCT prediction. In [45], a real-time PCT prediction method is proposed based on Deep Belief Network (DBN) [46], [47]. The types and waiting list of all work-in-process (WIP) products are captured using Radio-Frequency Identification (RFID), and then the data is fed to the DBN to predict the PCT in a real-time fashion. However, the referred research [45] does not consider the frequent random disturbances in the

production system. In [42], besides the RFID data that reflect the WIP product status, Fang et al. [42] further includes real-time data regarding machine and tool conditions in the proposed PCT method using Deep Stacked Sparse Autoencoder (S-SAE) model [48].

Thanks to the easy access to computing resources and availability of abundant real-time data on today's plant floor, data-driven methods, especially those based on deep learning, have taken the PCT prediction to the next level. However, the deep learning-based method also has its disadvantages. In [42], [45], the deep learning model, either DBN or S-SAE, developed for completion time prediction almost works as a 'black box' – all kinds of data collected from plant floor are indiscriminately lumped into deep learning algorithms. In these studies, PCT prediction is approached as purely machine learning problems. Few studies in this area are based themselves on a clear system model and not much domain knowledge is incorporated into the machine learning model construction, which turns out to be a great challenge facing the deep learning-based methods for PCT prediction.

In this context, we propose to a hybrid approach for product completion time prediction by integrating a novel system model and deep learning technique. The system model and knowledge are not only used to guide the deep learning model development. More importantly, the system model is directly used in the process of predicting PCT. The advantages of the proposed hybrid approach are two-fold. Firstly, the training efficiency and prediction accuracy of the deep learning model could be greatly improved with the guidance of domain knowledge. Secondly, compared to pure deep learning approach, the production managers in real industry would be more likely to embrace the proposed hybrid approach as it incorporates the rigorously derived system understanding and knowledge acquired from plant floor.

3.2.Problem descriptions

In this chapter, PCT of product $\boldsymbol{q}_n$ refers to the time duration it takes to complete all the remaining processes for the product. In literatures, PCT is also referred to as makespan, remaining time, lead time or flow time, although sometimes they might have slightly different meanings. We will consistently use the term PCT in this paper. The formal definition of PCT is given in the following.

Definition 3.1. *Given a product sequence $\boldsymbol{Q}$, at time t, the product completion time (PCT) for the n^{th} product $\boldsymbol{q}_n$ in product sequence $\boldsymbol{Q}$, denoted as $y_n(t)$, is defined as*

$$y_n(t) = max\{0, inf\{t' \in \mathbb{R}^+ | Y_M(t') \geq n\} - t\} \tag{3.1}$$

where $Y_M(t')$ is the accumulative production count of the end-of-line machine S_M at time t'.

In the definition, $Y_M(t') \geq n$ means that the accumulative production count $Y_M(t')$ of the end-of-line machine S_M has surpassed the product index n at time t', and $\inf\{t' \in \mathbb{R}^+ | Y_M(t') \geq n\}$ is the exact time point when product $\boldsymbol{q}_n$ leaves the system and is deemed as a completed product. Therefore, $\inf\{t' \in \mathbb{R}^+ | Y_M(t') \geq n\} - t$ is the time needed to complete product $\boldsymbol{q}_n$ from current time t, which is defined as PCT.

Compared to traditional PCT definitions, the proposed definition extends the concept of PCT to product level and real-time scenario, which would meet the practical needs in modern manufacturing systems. On the one hand, the proposed definition extends the concept of product completion time to every single product in the sequence, including not only incoming product, but also WIP products that are already in the line, as traditional PCT predictions mostly concern only the former case. Clear information on when a customized product could be completed, either it is yet to be processed or it is being processed at the moment, would add massive value to the business. On the other hand, the PCT defined in this paper is evaluated in real time. It answers the question that how much time needed to

complete the product starting from current time t given real-time system status. Therefore, we can leverage abundant data collected from smart manufacturing system regarding the fast-changing system status to make more precise predictions on PCT.

However, the PCT prediction is not a trivial task. PCT of a product can be decomposed into two parts, i.e., the time duration the product is being processed by machines, and the time duration it waits in the queue to be processed. Therefore, PCT is partially affected by machine random failures. For single-machine system or closely connected production system without intermediate buffers, any random failures on machine(s) before a product is completed can be directly added to PCT. The PCT prediction on these systems is equivalent to predicting total durations of machine random failures in a given time period. However, PCT prediction in multi-product system described in this paper is far more complicated than just predicting machine random failures. It is hard to determine the real impacts of machine random failures on PCT in multi-product serial lines with considering intermediate buffers, since it is coupled with real-time system status such as buffer contents and preceding products. There is not even a closed-form representation for PCT if all the future random failures are known. Therefore, there is an enormous state space in PCT prediction problem that is intractable if only with analytical model.

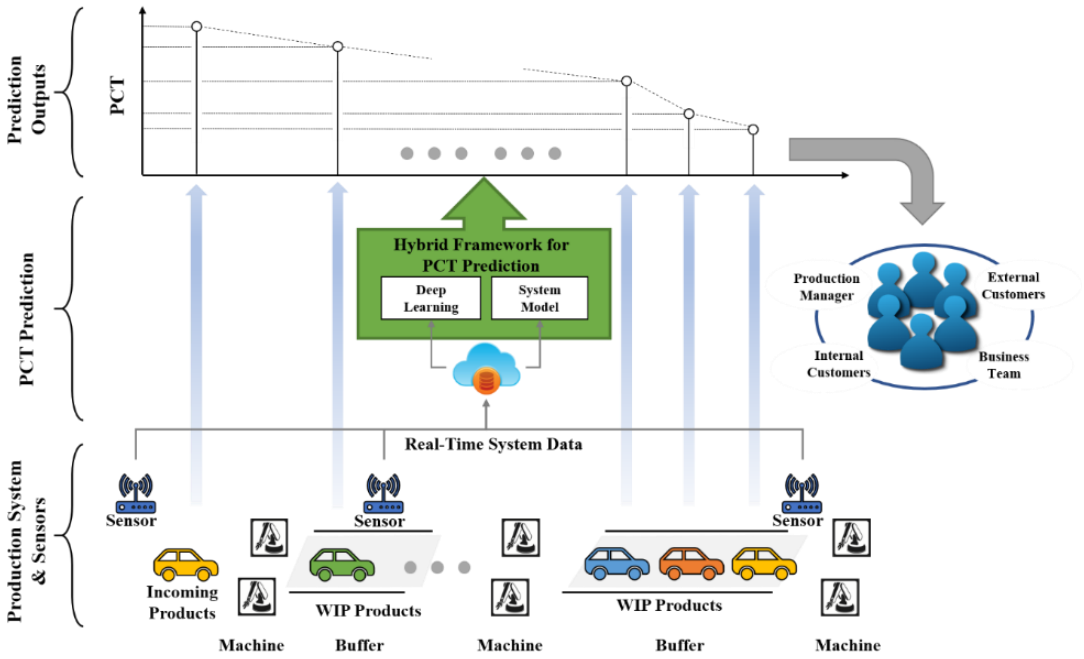


Figure 3.1. Framework for real-time PCT prediction in production system

The general approach of the proposed PCT prediction can be described as below. As shown in Figure 3.1, the sensors keep monitoring the real-time system status and transmitting the data to factory cloud. These data might include: (1) machine operation status; (2) WIP product types and locations in the system; (3) products incoming or the production manager intends to add to the incoming queue. A PCT prediction system will process these data in a real-time fashion to predict the PCT for every product, either WIP or incoming. The predicted PCT may serve multiple purposes, for example, production manager can schedule productions by trying different incoming production sequence to compare the PCTs, or customers that already placed their orders can track the PCT in real time.

3.3.System properties

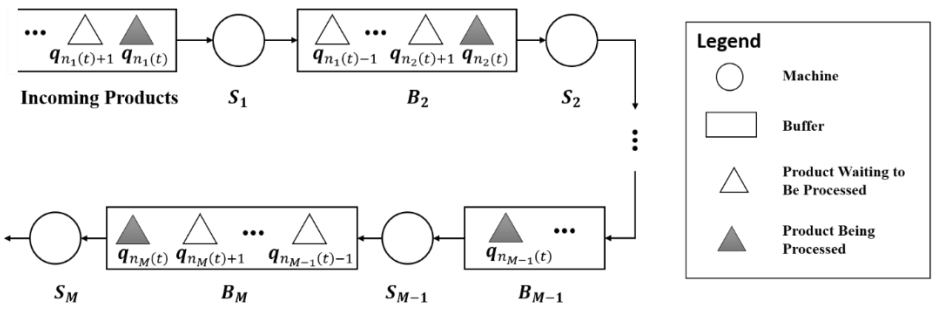


Figure 3.2. Real-time status of a multi-product system

Given a multi-product system as shown in Figure 3.2, the formal definition of the product completion time is given in the following.

Definition 3.2. *Given a product sequence* $\boldsymbol{Q}$*, at time* t*, the product completion time (PCT) for the* n^{th} *product* $\boldsymbol{q}_n$ *in product sequence* $\boldsymbol{Q}$*, denoted as* $y_n(t)$*, is defined as*

$$y_n(t) = max\{0, inf\{t' \in \mathbb{R}^+ | Y_M(t') \geq n\} - t\} \tag{3.2}$$

where $Y_M(t')$ *is the accumulative production count of the end-of-line machine* S_M *at time* t'.

In addition, three important properties are derived based on the proposed model and systematic understandings.

Property 3.1. There exists a strict lower bound $\hat{y}_n(t)$ for the PCT of product $\boldsymbol{q}_n$, denoted as $y_n(t)$, at time t, and it can be evaluated as

$$\tilde{y}_n(t) = max\{0, inf\{t' \in \mathbb{R}^+ | Y_M(t'; \boldsymbol{w}(t'') = \boldsymbol{0}, \forall t'' > t) \geq n\} - t\} \tag{3.3}$$

where $\tilde{y}_n(t)$ denotes the strict lower bound for $y_n(t)$ and $Y_M(t'; \boldsymbol{w}(t'') = \boldsymbol{0}, \forall t'' > t)$ is the production count of the end-of-line machine S_M at time t' assuming there are not any random downtime events in the system starting from time t, i.e. $\boldsymbol{w}(t'') = \boldsymbol{0}, \forall t'' > t$.

Remark 1: This property arises from the observation that machine downtime events, if impacting PCT, would only lead to delay of production and elongation of the PCT. Therefore, PCT lower bound is derived by assuming clean case beginning from the time when we evaluate PCT. Note that the lower bound $\tilde{y}_n(t)$ is a deterministic number given the system status at time t. In general cases, it does not have a closed-form representation due to complex interactions among machines. However, we can utilize the proposed system model to recursively calculate $\hat{y}_n(t)$ by assigning $\boldsymbol{w}(t'') = \mathbf{0}, \forall t'' > t$. This property lays the foundation for a hybrid approach to the PCT prediction.

Property 3.2. The PCT of product $\boldsymbol{q}_n$ is independent of that of any products coming after $\boldsymbol{q}_n$ in the product sequence $\boldsymbol{Q}$, i.e.

$$p\big(y_n(t)|y_{n'}(t)\big) = p\big(y_n(t)\big), \forall n' > n, \forall t \geq 0 \tag{3.4}$$

Remark 2: Property 3.2 states the correlations of PCT among products. As products are processed in sequence, a product could be processed by a machine only when all the products waiting before it has been processed. Therefore, the PCT of $\boldsymbol{q}_n$ could only be affected by its preceding products. In other words, any products in behind would not affect the product $\boldsymbol{q}_n$ regarding PCT at all.

Property 3.3. At time t, the PCT of product $\boldsymbol{q}_n$ is independent of the status of machines if $\boldsymbol{q}_n$ has already completed the process on those machines, i.e.

$$p\big(y_n(t)|w_i(t)\big) = p\big(y_n(t)\big), \forall t \geq 0, if\ n > n_i(t) \tag{3.5}$$

Remark 3: Similar to Property 3.2, this property states that the completion of product $\boldsymbol{q}_n$ is independent of machines in behind. These two properties inspire the construction of a recurrent structure in the PCT prediction problem, which leads to the use of a deep learning model specialized in sequence input.

3.4.Hybrid deep learning framework

3.4.1. Long Short-Term Memory

Neural networks are set of algorithms which closely resemble the human brain and are designed to learn the relationship between inputs and outputs through data. As neural network is applied to many different problems, the basic feedforward structure no longer meets some particular needs, and therefore there have been some special neural networks, among which is the recurrent neural network (RNN) [49]. RNN is a family of neural network that is specialized for processing sequential inputs.

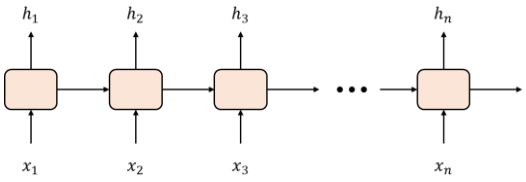


Figure 3.3. Recurrent Neural Network structure [22]

Let $\boldsymbol{x} = [x_1, x_2, \dots, x_n, \dots]$ denotes the sequential inputs. RNN assumes that the output from current step is depending on previous steps in the sequence. As shown in Figure 3.3, RNN has the form of a chain of repeating modules of neural network. It has an output h_n for each step and takes output from previous step and current input to generate output for current step, i.e.,

$$h_n = f(h_{n-1}, x_n; \theta) \tag{3.6}$$

where h_{n-1} is the state of previous step, x_n is the input of the current step, $f(*; \theta)$ is the recurrent function defined by parameters θ. As sequential dependencies are very common in many practical problems, there have been a number of successful applications of RNN or its variants, e.g., machine translation, prediction problems and video tagging etc. One of the most prominent variants of RNN is the Long Short-Term Memory (LSTM) [50]. The LSTM shares the same chain-like structure in Figure 3.3, but there is an internal hidden state called as cell state c_n inside LSTM cells (Figure 3.4). The LSTM has the ability to remove or add information to the cell state, carefully regulated by structures called gates.

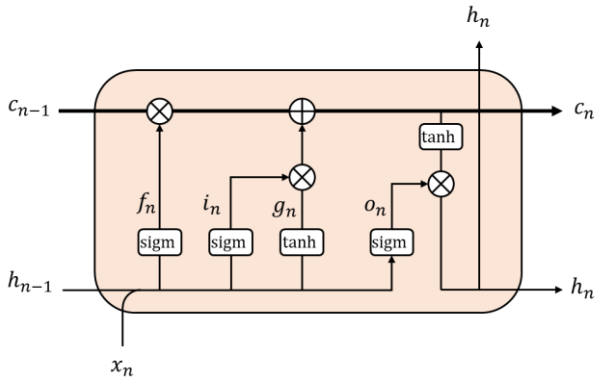


Figure 3.4. Internal structure of recurrent unit in LSTM [23]

In brief, there are an 'input gate' that ensures the cell state C_n only absorb useful information from the new input x_n, a 'forget gate' that discards obsolete information in cell state C_n, and an 'output gate' that generates output h_n. One can refer to [50] for detailed updating rule in LSTM cell given input x_n and output h_{n-1} from previous step. With cell state and special gates setting, LSTM is able to keep useful information from many steps ago, and therefore it is especially capable of capturing long-term dependencies. In this paper, we will use LSTM and the system model proposed in Section 3 to establish a hybrid approach to PCT prediction.

3.4.2. Recurrent sequence discovery in PCT prediction

Based on Properties 3.2 and 3.3, the products and machines in the system can be modeled as recurrent units. For example, the completion time of product $\boldsymbol{q}_{n_M(t)}$ only depends on its own features, e.g., product type and cycle time etc., and the status of machine S_M. Therefore, a recurrent sequence is constructed beginning from the end-of-line machine S_M and propagates backwards along the line. Figure 3.5 also shows how the production line in real time is mapped to a recurrent sequence.

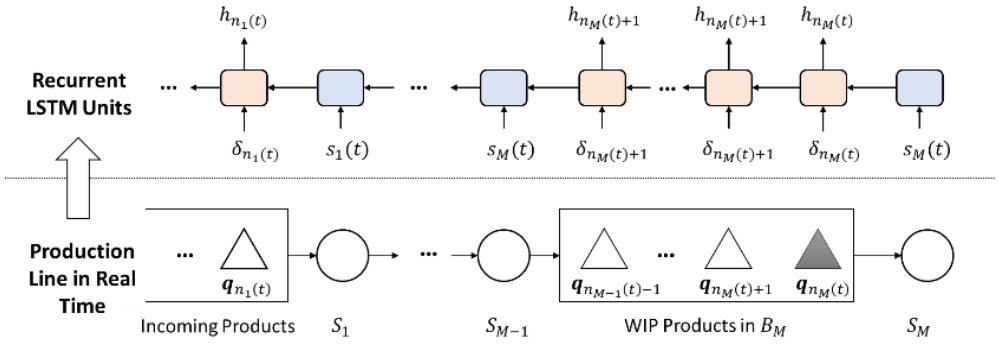


Figure 3.5. The multi-product line in real time corresponds to a recurrent sequence

After establishment of the recurrent sequence, one important task is to identify the inputs to the recurrent units in LSTM. Since our goal is to predict PCT, any factors that affect PCT should be included in the inputs. For machine S_i, these real-time factors can be represented as $s_i(t)$.

$$s_i(t) = \begin{bmatrix} w_i(t) \\ r_i(t) \end{bmatrix} \tag{3.7}$$

where $w_i(t)$ indicates if machine is up or down at time t, and $r_i(t) \in (0,1)$ denotes the ratio of process machine S_i has completed on its current product $\boldsymbol{q}_{n_i(t)}$.

For product $\boldsymbol{q}_n$, either WIP or incoming, it is the actual cycle time on each machine that could affect the final PCT. Let δ_n denotes the processing time, including cycle time and tool setup time, of product $\boldsymbol{q}_n$ on each machine, we have

$$\delta_n = \begin{bmatrix} \boldsymbol{q}_n^T \boldsymbol{T}_1 + \boldsymbol{q}_{n-1}^T \boldsymbol{T}_1^{st} \boldsymbol{q}_n \\ \boldsymbol{q}_n^T \boldsymbol{T}_2 + \boldsymbol{q}_{n-1}^T \boldsymbol{T}_2^{st} \boldsymbol{q}_n \\ \vdots \\ \boldsymbol{q}_n^T \boldsymbol{T}_M + \boldsymbol{q}_{n-1}^T \boldsymbol{T}_M^{st} \boldsymbol{q}_n \end{bmatrix} \tag{3.8}$$

Therefore, the sequential input $\boldsymbol{x}$ to the LSTM is obtained by arranging machines and products in the correct sequence.

$$\boldsymbol{x} = \left[s_M(t), \delta_{n_M(t)}, \delta_{n_M(t)+1}, \dots, \delta_{n_1(t)-1}, s_1(t), \delta_{n_1(t)}, \delta_{n_1(t)+1}, \dots, \delta_{n_1(t)+N} \right] \tag{3.9}$$

where N is the number of incoming products that one wants to include for PCT prediction in a given sequence. Consequently, a recurrent sequence has been established for PCT prediction.

3.4.3. Hybrid framework combining LSTM and system model

In this research, PCT is predicted with a hybrid approach by combining the strengths of both deep learning and system model. The overall architecture of the proposed hybrid approach is shown in Figure 3.6.

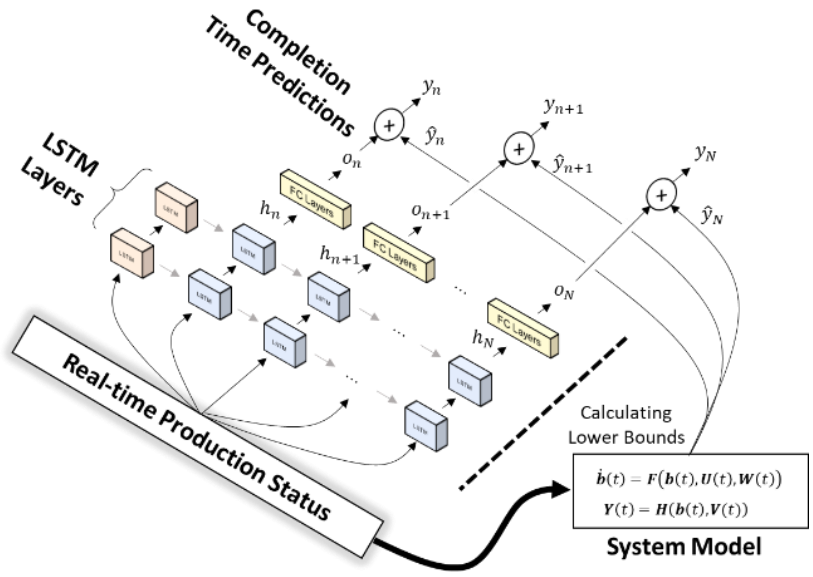


Figure 3.6. A hybrid PCT prediction model combining LSTM and system model

Given the recurrent sequence $\boldsymbol{x}$, LSTM will have output for each recurrent unit. Following the updating rules in the LSTM, we can calculate the outputs from recurrent units in sequence. Note that there might be multiple LSTM layers, since it is not uncommon for people to stack two or more LSTM layers to refine the outputs in complicated problems. In PCT prediction problem, we are not interested in the outputs from units representing machines, since there are no predictions associated with them. The machine units are only used to provide LSTM with necessary information regarding their operation status that could affect PCT. Let h_n represents the output from recurrent units representing product $\boldsymbol{q}_n$ with input δ_n, then LSTM outputs $\boldsymbol{h}$ corresponding to inputs $\boldsymbol{x}$ are

$$\boldsymbol{h} = \left[h_{n_M(t)}, h_{n_M(t)+1}, \dots, h_{n_1(t)-1}, h_{n_1(t)}, h_{n_1(t)+1}, \dots, h_{n_1(t)+N}\right] \qquad (3.10)$$

The output h_n is further processed with fully connected (FC) layers to reduce the dimension to one, since PCT is a single scalar for each product. In our problem,

the final output could have been a real number that represents the PCT. However, according to Property 1, for any product $\boldsymbol{q}_n$, there exists a strict lower bound $\tilde{y}_n$ for its actual PCT y_n, where $\tilde{y}_n$ is a deterministic number given the system status. Instead of directly predicting the PCT from the deep learning model, we will train the model to only predict the distance from the lower bound to its target PCT. Let o_n be the non-negative output of FC layers connected to h_n, then

$$o_n = g(h_n; \theta_2) \tag{3.11}$$

where $g(*; \theta_2)$ denotes the FC layers with parameters θ_2. As mentioned in previous section, lower bound $\tilde{y}_n$ can be directly computed using the system model. Therefore, the final PCT prediction of product $\boldsymbol{q}_n$ is given by combining the results from the deep learning model and analytical system model, i.e.

$$\hat{y}_n = \tilde{y}_n + o_n \tag{3.12}$$

where $\hat{y}_n$ denotes the predicted value of the PCT y_n. In training time of the LSTM, the loss function is the mean square error between target PCT y_n given by training data and predicted PCT $\hat{y}_n$ given by the hybrid approach, i.e.

$$\min_{\theta} \left\{ \frac{1}{\left(n_1(t) + N - n_M(t)\right)} \sum_{n=n_M(t)}^{n_1(t)+N} \left(y_n - (\tilde{y}_n + o_n)\right)^2 \right\} \tag{3.13}$$

where $\theta = [\theta_1, \theta_2]$ is the parameters for the deep learning model, including parameters θ_1 for the LSTM layers and parameters θ_2 for the FC layers. In this way, a hybrid framework for PCT prediction has been established, since system model computes $\tilde{y}_n$, while deep learning model predicts o_n.

The proposed hybrid approach has the following advantages:

(1) The proposed framework would increase the training efficiency and prediction accuracy compared to the pure deep learning approach, as we have incorporated precise estimation on part of the final result using the analytical system model.

(2) The proposed framework would effectively avoid unrealistic predictions that underestimate PCT. In Eq. (28), o_n given by LSTM is a non-negative number. Hence, the proposed framework would never give a prediction that is lower than $\tilde{y}_n$. However, a pure deep learning approach does not carry the same guarantee.

(3) The proposed framework has a good compatibility with the ever-changing system status by using LSTM. Regular neural networks mostly require a fixed-sized input, but the number of WIP products, as well as the incoming products, in the production system is always changing over time. In contrast to regular neural networks, LSTM is designed to process variable sequenced inputs, and there is no limitation on the length of the sequence it can process.

(4) The proposed framework largely simplifies the input data structure for the deep learning model. In LSTM, a lot of information regarding system status are implicitly embedded in the recurrent sequence. By arranging machine status $s_i(t)$ or product feature δ_n in a recurrent sequence, we are actually providing the deep learning model with abundant information regarding current system-level status, e.g., buffer levels, blockages, and preceding products etc.

3.5.Experiments and validation

3.5.1. Numerical experiment setup

A numerical experiment is conducted in a six-machine-eight-product production system. The production system in this case study has six machines and five buffers, i.e. $M = 8$. Each buffer has a capacity of five, i.e., $B_i = 5, \forall i = 2,3 \dots ,8$. The system is capable of processing ten different product types, i.e., $K = 10$. The cycle times fall in a range of $[1,4]$, and the setup times have a range of $[0,1]$.

The training data has two possible sources. The first possible source is the real historical data collected from plant floor, while the second is the simulation, as it is

convenient to construct a reliable simulator based on the system model proposed in this paper. In practice, the choice of the training data sources should be discussed case by case. Sometimes, one can even take a mixed approach by combining data from the two sources. In whichever case, it is important to figure out if the machine status is Markovian or non-Markovian. For non-Markovian machines, the machine status in the future depends on not only current status but also the status history before reaching current status. In this scenario, we will have to either stack machine status w_i over a period of time or use another RNN to extract the machine hidden state. In this paper, however, for demonstration purpose we will use dataset generated from simulation based on Markovian machines. The machine reliability parameters as shown in Table 3.1. $MTBF$ denotes the mean time between failures, and $MTTR$ is the mean time to repair.

Table 3.1. Machine reliability parameters in this case study

Machine	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8
MTBF	150	152	102	132	111	135	126	155
MTTR	6	5	3	5	8	4	7	8

Using the simulation based on our model, we have generated around 10,000 data points in total. Each data point consists of input $\boldsymbol{x}$, which includes real-time machine status and product features, and target makespan $\boldsymbol{y}$. The dataset is split by a ratio of 70:30 to construct a training set and a testing set respectively.

3.5.2. Training process and results

We have conducted 10-fold validation to find the optimal neural network hyper-parameters for our model. In the final model, there are two LSTM layers, and each layer has 128 hidden units. The LSTM layers are followed by two FC hidden layers with 128 hidden units each. The outputs of both LSTM layers are normalized before any further processing. We concatenate the outputs from both LSTM layers to be the input to FC layers. The activation function for the fully connected layer is 'ReLU'. LSTM often experiences the problem of gradient exploding [51]. Therefore, the gradient clipping technique is adopted to clip the gradient value by

range $[-1,1]$ in order to avoid gradient exploding. In order to overcome overfitting, a dropout rate of 0.2 is used in the two LSTM layers [52]. As for FC layers, we use both L1 and L2 regularization technique, and the penalty coefficients are 0.08 and 0.1 respectively.

The training process of the hybrid model is as shown in Figure 3.7Figure 3.7. Training progress for the hybrid deep learning model. The optimizer used in this case study is RMSProp with learning rate $\alpha = 0.0001$ and other parameters set as defaulted. The training batch size is 32. The deep neural network model is deployed on TensorFlow platform. Both training loss and testing loss decrease substantially with the training epochs.

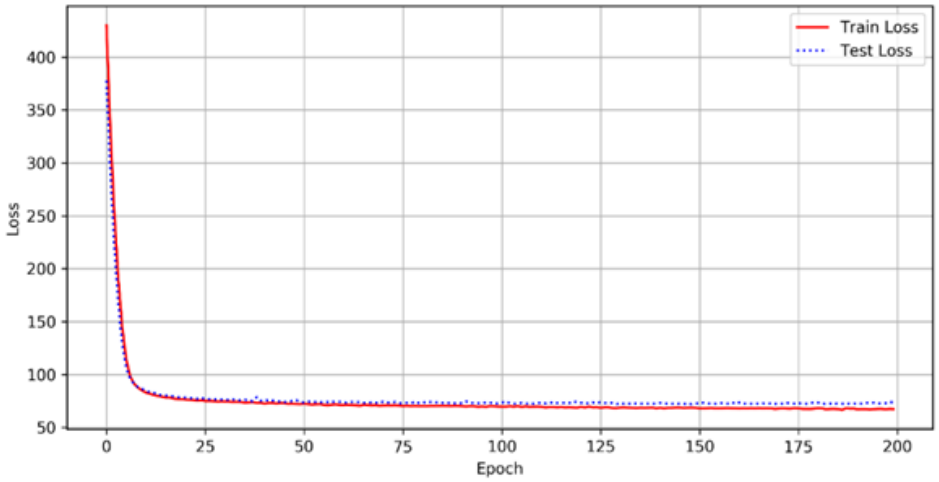


Figure 3.7. Training progress for the hybrid deep learning model

3.5.3. Model performance comparison and interpretation

The proposed hybrid PCT prediction is compared with other machine learning models [53], including linear regression (LR), multi-layer perceptron (MLP), and random forest (RF), and k-nearest neighbors (KNN). In order to validate if the use of model in the hybrid approach improves the performance, we also train a pure LSTM model for PCT prediction. . Since these models have a different input format from that for LSTM, we construct the training dataset following the format given in [42], in which each data point represents a single product. Note that the newly constructed training dataset represents the same system status in the original dataset.

The only difference is the way that data is formatted for the needs of these comparison machine learning models. Each data point contains the following categories:

- Real-time production task: task composition and total WIP level.
- Real-time production status: WIP waiting in buffer, WIP in processing, planned processing time.
- Real-time machine condition: real-time machine status.
- Product index.

The output is the PCT y_i of the product corresponding to the product index given in the input. Following the dataset structure, the input has a dimension of 80. The newly constructed dataset has a total number of 240,080 data points. Given the dataset, the details of the comparison machine learning models are as follows.

(1) **LR**: The LR model predicts the PCT by a linear combination of given input variables.

(2) **MLP**: Through cross-validation, the MLP model is determined to have three hidden layers with 128 units in each layer. The activation function is 'Relu', while the L2 regularization is set to be 0.3 to avoid overfitting.

(3) **RF**: This is an ensemble learning method that constructs a multitude of decision trees and outputs the prediction averaged across all individual trees. The maximum depth in this model is eight. We use the RF regressor in the scikit learn package in Python to construct the RF model.

(4) **KNN**: The only hyper-parameter in KNN is k, which is the number of nearest neighbors to take into consideration when predicting the output of a new datapoint. In this demonstrated case, we find $k = 3$ to be the optimal choice.

(5) **Pure LSTM**: In this model, we still use the original dataset with recurrent format. The only difference from the proposed hybrid model is that we do not use the analytical model to compute PCT bound. Instead, we rely on the

LSTM to predict the entire PCT value. We use the same LSTM architecture in the proposed hybrid model and increase the training epochs to 300 from 200.

These models are compared regarding five performance metrics commonly used in regression tasks, including mean absolute error (MAE), mean absolute percentage error ($MAPE$), root mean squared error ($RMSE$), R-square (R^2), and realistic prediction ratio (RPR).

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{3.14}$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \times 100\% \tag{3.15}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{3.16}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)}{\sum_{i=1}^{n}(y_i - \bar{y})} \tag{3.17}$$

$$RPR = \frac{\sum_{i=1}^{n}[\hat{y}_i \geq \tilde{y}_i]}{n} \times 100\% \tag{3.18}$$

where y_i is target PCT, $\hat{y}_i$ is predicted PCT, $\bar{y}$ is the mean of target PCT, and n is the total number of products that are considered in the test dataset. Note that *RPR* is a performance metric specially designed for PCT prediction problem. If a prediction model gives PCT prediction $\hat{y}_i$ lower than its bound $\tilde{y}_i$, according to PCT Property 1, the prediction is unrealistic in the real world. These misleading predictions unrealistically underestimate PCT and might present production managers with problems in production scheduling and customer satisfaction etc. The metric RPR is designed to evaluate the percentage of predictions that are realistic, i.e., $\hat{y}_i \geq \tilde{y}_i$, among all products considered in the test dataset. Apparently,

one should pursue higher RPR when designing PCT prediction models. The performance comparison is presented in Table 3.2 and Figure 3.8.

Table 3.2. Performance comparison among different machine learning models

Model	LR	MLP	RF	KNN	Pure LSTM	Hybrid Model
MAE	5.77	7.21	5.62	7.94	5.42	5.02
MAPE	35.10%	32.86%	23.08%	63.47%	34.04%	10.81%
RMSE	8.742	8.60	8.81	12.39	8.61	8.32
R^2	0.905	0.902	0.904	0.804	0.908	0.914
RPR	89.91%	99.09%	91.29%	70.61%	91.09%	100%

For the comparison models, the inputs contain all the available data in the system, some of which, according to our analysis, are not relevant in the prediction target. For example, the number of incoming products would never affect the PCT of any WIP products. By using LSTM, we avoid including these irrelevant data in the prediction. However, through the comparison between Pure LSTM and hybrid model, it is noted that LSTM alone, although has slightly better performance than other comparison models, is still not enough to deliver satisfactory prediction performance. More importantly, it does not convey the guarantee that all predictions are realistic. Therefore, the use of analytical model in the hybrid approach is indispensable. Furthermore, the fact that Pure LSTM requires more training epochs, i.e., 300 epochs compared to 200 for hybrid model, indicates that the hybrid model reduces the computing time and resource needed for training the deep learning model in PCT prediction.

In Figure 3.8, we plot all the datapoints in the test dataset with real PCT as the horizontal axis and predicted PCT as the vertical axis. We use blue dots to represent realistic predictions, i.e., $\hat{y}_i \geq \tilde{y}_i$, and magenta dots to represent unrealistic predictions, i.e., $\hat{y}_i < \tilde{y}_i$. The $RPRs$ for LR, RF and Pure LSTM are around 90%. In other words, these models provide production managers with around 10% of the total predictions that can never be realized. MLP has a RPR as high as 99.09%, however, its overall prediction accuracy is not satisfactory. The 100% RPR from the proposed hybrid model is theoretically guaranteed when we construct the prediction model.

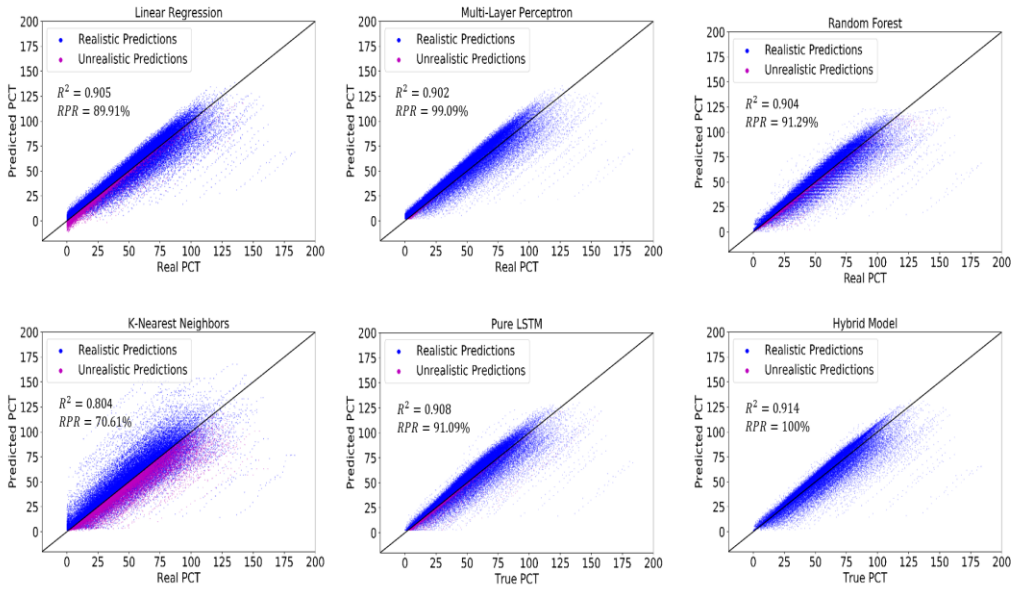


Figure 3.8. The plots of true PCT vs. predicted PCT for all datapoints in test dataset

3.5.4. PCT prediction output demonstration

Figure 3.9 present a typical output given a system status input from the test dataset. The output has the same recurrent sequence as the input. Since recurrent unit representing a machine has not output, we place a tick on the horizontal axis. The prediction output can be conveniently interpreted. As shown in Figure 3.9, based on the ticks, we can tell that there are three WIP products in buffer B_8 and the hybrid model gives PCT predictions that are very close to real PCT, while there is no product in buffer B_7 that locates between machine S_6 and S_7. Some annotations along with the PCT bounds are added in the graph for better illustrating the prediction output.

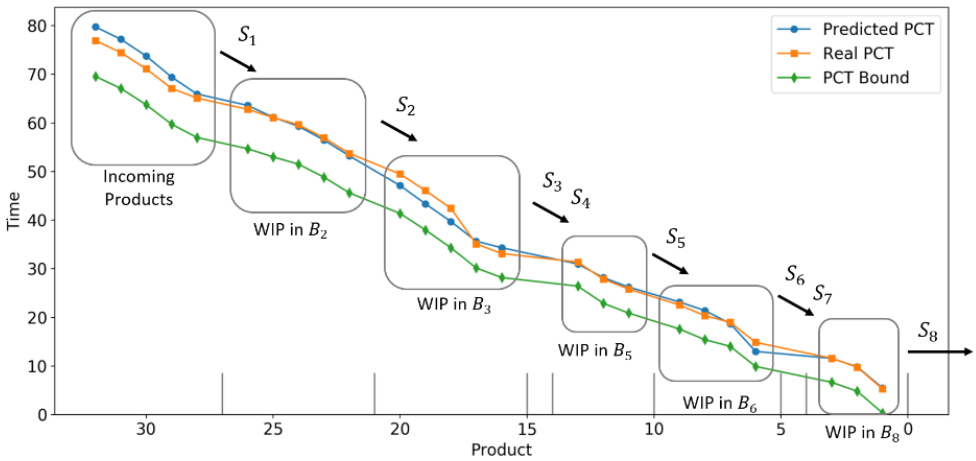


Figure 3.9. PCT prediction output interpretation

Further, we select another 12 datapoints from the test dataset and show their outputs in Figure 3.10. Since the production system is dynamic, we can find that the numbers of products in the system vary among different system status. Products closer to the right in the graph are closer to being completed, and hence have smaller PCT values. In most cases, the prediction errors tend to be smaller for products with smaller PCT. PCT can also be interpreted as the time a product has to stay within the system before it is completed. When the staying time is longer, there could be more random failures and therefore the PCT is more likely to be affected. Therefore, the PCT of a product with larger product index is not only larger than that of a product with smaller product index but also more uncertain. However, since we predict PCT in a real-time fashion, in practice we would continue to observe system status and output predictions. Consequently, given one specific product we will have more confident predictions as time passes.

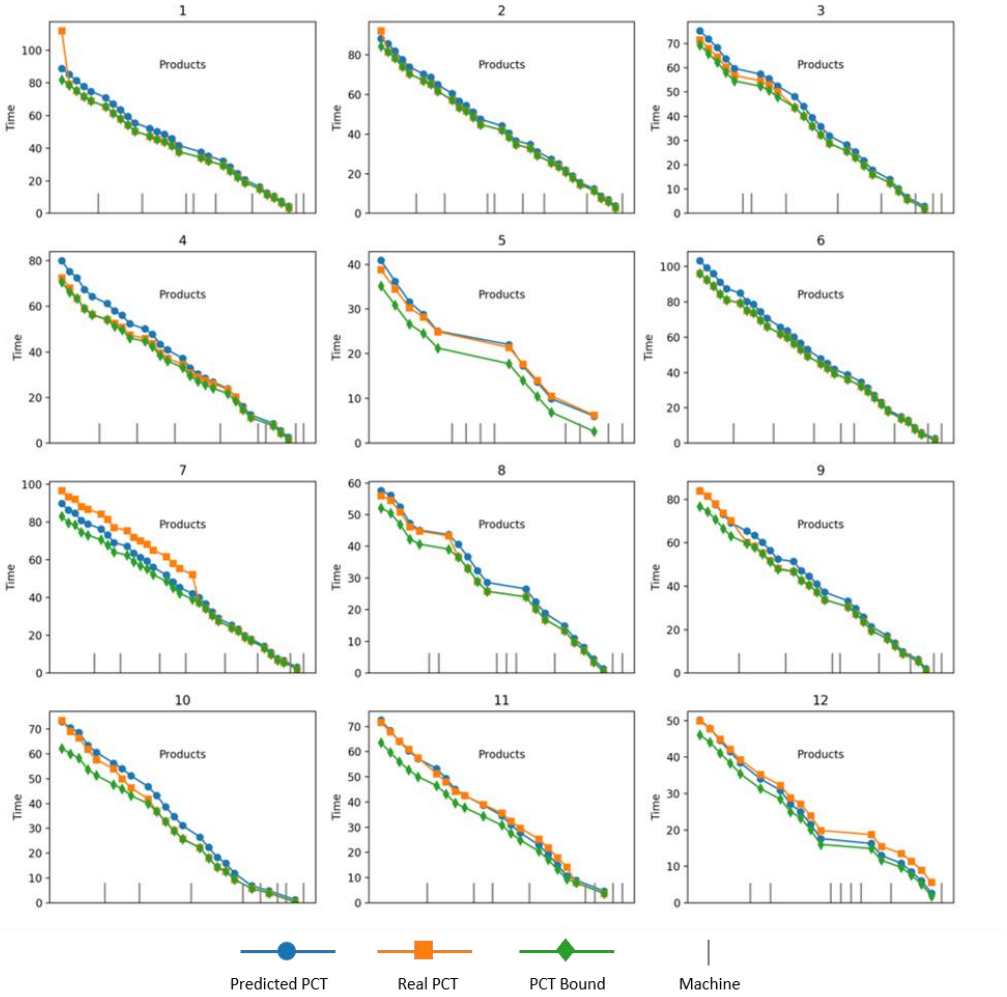


Figure 3.10. Predictions from hybrid model given different system status as inputs

To sum up the case demonstrated in this section, we successfully implement the hybrid PCT prediction model in a dataset generated from simulation. The comparisons with other models indicate that the hybrid model is consistently better than other models regarding all performance metrics. The hybrid model as it is derived and constructed guarantee 100% RPR, which means it would never give predictions that are unrealistic unlike other comparison models. Therefore, the case study presented in this section proves that the proposed hybrid model for PCT prediction in multi-product system is effective.

3.6. Summary

This chapter proposes a hybrid approach to PCT prediction, which combines the strengths of both analytical system model and deep learning technique. To achieve effective PCT prediction, the mathematical model for multi-product serial production line in Chapter 2 is utilized. Based on the characteristics of production system, a recurrent sequence is discovered in PCT prediction problem, in which machines and products are modeled as recurrent units in LSTM, a prominent variant of RNN that is specialized in sequential prediction problem. Instead of adopting a pure deep learning approach, we combine system model and LSTM to establish a hybrid model for PCT prediction, which is proved to be effective in a demonstration case. In this research, the system model and domain knowledge are not only used to guide the construction of deep learning model, but also, more importantly, the system model is directly integrated with the deep learning model in prediction task.

3.7. Related work

Part of the results presented in this chapter have been published in [26], [54].

Chapter 4. Rolling Horizon Method for Corrective Maintenance Control

4.1.Background

Manufacturers are seeking to operate the production in a more efficient and cost-effective way when facing the growing competition and globalization. Since capital expenditure incurred by maintenance accounts for a large portion of the overall cost in production activities, optimal maintenance decision making is one of the fundamental aspects in achieving this end.

Random failure occurs when a machine deteriorates to a certain level due to usage and aging. Upon the random failure, a maintenance action has to be carried out in order to restore the machine to an acceptable operation condition. In industrial practice, maintenance is not necessarily to completely replace the failed machine with a new one. Based on the structure of the machine, multi-level maintenance options could be available. A perfect maintenance, or replacement, is recovering the machine 'as good as new', while a minimal maintenance is recovering the machine 'as bad as old', which only resumes its operation without changing deterioration status. Imperfect maintenances are recovering the machine to somewhere between old and new.

The options of different maintenance levels largely expand the research on maintenance. Based on the levels of maintenance effects, a lot of maintenance policies have been proposed in the past decades [55]–[57], most of which were based on single-unit system. Decision making on maintenance is essentially a trade-off between production performance and machine reliability. For single-unit systems, the unavailability of the system during maintenance can be directly

counted towards the production loss. Therefore, the maintenances scheduling, which is modeled as a stochastic process, can be conveniently optimized such that some long-term criterion, such as maintenance cost rate or system unavailability, is minimized.

In contrast with the single-unit system, Wang [58] concludes that the traditional maintenance policies for multi-unit system fall into two categories, i.e. group maintenance policies [59], [60] and opportunistic policies[61], [62]. These policies generally originate from the observation that in a multi-unit system the failure of any one component will immediately halt the whole system, during when other components can be maintained simultaneously without incurring extra production losses. However, it is not necessarily the case in a general multi-stage manufacturing system, where machines work asynchronously, and intermediate buffers reduce the spread of stoppages. It is noted that some studies [63], [64] on maintenance in multi-stage manufacturing systems are also based on the assumption that there are no buffers between machines. These studies, along with the policies for single-unit and multi-unit systems, might not work well for a general multi-stage manufacturing system.

Multi-stage manufacturing systems are characterized by their complex structures of strongly interconnected machines and stochastic dynamics. [65] It is difficult to obtain the optimal maintenance policy in a general multi-stage manufacturing system. First, the relationship between machine stoppage due to maintenance and system production loss is unclear. Our previous studies [66] reveal that the system production loss of machine stoppage heavily depends on the system states. For example, given the same maintenance duration on the same machine at different time, the production loss could be very different since the system states are dynamic. But the system state at any time is very difficult to evaluate with transition probability, because it is well known that the close-form solution of system states only exists for two-machine-one-buffer system and systems with infinite buffers or no buffers [67]. As a result, it is nearly impossible to obtain a

globally optimal maintenance policy in the long run for a multi-stage manufacturing system.

To address the complexity, some simulation-based studies have been conducted to find the optimal maintenance schedule [68]. For instance, Arab et al. [69] incorporated remaining reliability of machines and work-in-process inventories into the simulation model to search for the optimal maintenance schedule. However, changes on the process and equipment, which are norm in today's manufacturing industry, will lead to corresponding changes and reconstructions in the simulation models, and subsequently a higher resource consumption. Nahas et al. [70] attempted to allocate the buffers subject to a limited total buffer space aiming at minimizing the maintenance costs. This is applicable in system design stage rather than the maintenance control on an existing system. References [71], [72] both study the integration of maintenance control and quality control in multi-stage manufacturing systems, but they focus on the relationship between machine status and product quality. Some works [66], [73], [74] try to find the opportunities for maintenance in multi-stage manufacturing system, but the machine reliability and aging is ignored. Therefore, a systematic method is desired to manage maintenance through both careful analysis of the machine reliability and production dynamics [75].

While information about production systems has become increasingly transparent, detailed, and real-time, some researchers attempted to obtain optimal control strategy on the system by utilizing the real-time data collected by distributed sensors. Zou et al.[76] was able to establish a data-driven model for production system analysis, in which the diagnosis and prognosis of production losses were established. The model has been applied to energy control [77], gantry assignment [78] and etc. in multi-stage manufacturing systems. Consequently, it is feasible to utilize a maintenance policy based on real-time system information and implemented in real-time manner instead of a globally optimal maintenance policy. In this paper, the production losses of maintenance actions are properly estimated based on the real-time system information. The real-time maintenance cost rate is

established. A real-time maintenance policy of selecting the proper level of maintenance upon machine failure is proposed in this chapter.

4.2.Sequential decision-making formulation and solutions

Most of the control problems in manufacturing systems can be mathematically represented as an objective function and relevant constraints. Due to the dynamic nature of manufacturing systems, the sequential decision-making is an appropriate framework for most control problems.

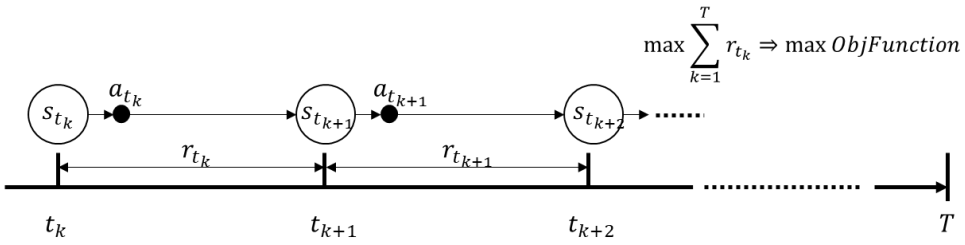


Figure 4.1. Sequential decision-making framework illustration

As shown in Figure 4.1, the continuous time within the planning horizon, e.g., a shift or a day etc., is discretized into steps. The discretion is based on specific problems but generally falls into two possible categories. One is the event-triggered mechanism, which means the time step is the time when a decision or an action is needed to respond to an event. For example, in this chapter, the CM decision is triggered by a random failure. The other is discretion over equal time intervals. For example, in PC decision making presented in later chapter, an equal time interval is chosen to consider PM decisions.

At each step t_k, a state s_{t_k} representing the real-time status of the system can be observed. Based on the state, an action a_{t_k} is chosen and then implemented in the system. Upon next time step t_{k+1}, a reward is given by the system to reflect the goodness of the chosen action. If one can guarantee that the maximization of accumulated rewards is equivalent to maximization/minimization of the objective function, then the control problem can be solved by solving this sequential decision-making problem.

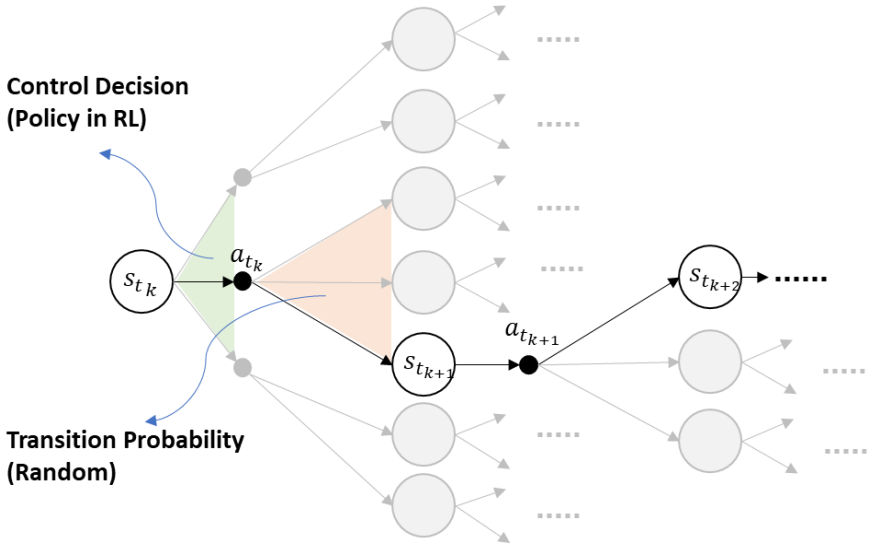


Figure 4.2. Sequential decision making as a search tree

The sequential decision making is challenging because of its enormous search space due to large state space. In Figure 4.2, the sequential decision-making process is expanded to reflect all the possible transitions. The goal is to find a policy or rationale to choose actions given a state. There are two methods to solve such a sequential decision-making problem, namely, rolling horizon method and reinforcement learning method.

Rolling horizon method only consider a look-ahead time window (e.g., until next failure). By limiting the consideration within a certain time window, we effectively trim the depth of the search tree. Subsequently, we find approximation of future state transitions and rewards within the time window, which effectively trim the width of the search tree. By these two steps of approximation, suboptimal solutions can be found to optimize system performance, but only within a certain time window. Usually, rolling horizon method is fast during online execution and requires no training process. However, a good approximation is not always available to a given control problem. Further, due to the approximation, if available, there is no guarantee on solution optimality.

Reinforcement learning (RL) uses state values to represent the expected accumulated reward starting from one state. The state values are obtained through

iterative training conducted offline. The policy is executed by choosing action towards the subsequent states that have the highest state value. RL is also fast during online execution but needs large amounts of computing time and resource for training. It has some guarantee on policy optimality. It fits some control problems in manufacturing systems well since a lot of historical data are available and can be leveraged to construct simulation environment. In this dissertation, both methods will be demonstrated for different problems. In this chapter, the CM problem is solved by a rolling horizon methods.

4.3. Corrective maintenance modeling and cost analysis

Given the failure-time distribution $p_i(t^*)$ of machine S_i, the failure rate is

$$\lambda_i(t^*) = \frac{p_i(t^*)}{1 - \int_0^{t^*} p_i(\tau)d\tau} \tag{4.1}$$

It can be shown that the failure rate $\lambda_i(t^*)$ is increasing when the failure-time distribution of the machine has the positive aging property. In this scenario, the imperfect maintenance or perfect maintenance options are preferred. Otherwise, if the failure rate $\lambda_i(t^*)$ of the machine is decreasing with respect to its age, a minimal maintenance is always preferred. The imperfect maintenance has been modeled through several approaches, including failure-rate reduction [79] and virtual-age reduction [80]. For the ease of derivation, we adopt the virtual-age approach to model the maintenance effects.

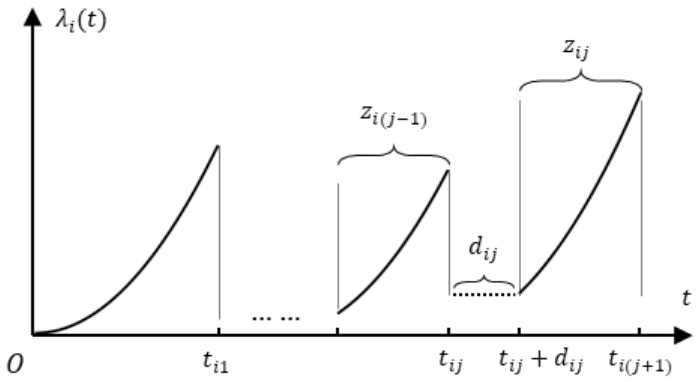


Figure 4.3. Maintenance actions taken at a machine

The age of machine S_i is continuously increasing with time until it breaks down and a maintenance has to be imposed. If the maintenance effect is not perfect, the machine after maintenance will behave as if it already has an initial age, which is referred to as virtual age. The virtual age of machine S_i right after its j^{th} maintenance action is denoted as v_{ij}^m. The superscript m is used to denote a value derived by assuming that the maintenance level is m. When the maintenance level has been determined, the superscript would be forsaken hereafter. The virtual age v_{ij}^m is

$$v_{ij}^m = A_i^m\left(v_{i(j-1)} + z_{i(j-1)}\right) \tag{4.2}$$

where $v_{i(j-1)}$ is the virtual age of machine S_i right after its $(j-1)^{th}$ maintenance action, $z_{i(j-1)}$ is its survival time after $(j-1)^{th}$ maintenance action, and A_i^m is the age reduction factor of maintenance level m. Clearly, $A_i^m = 0$ corresponds to a replacement since the virtual age of machine is reduced to zero. $A_i^m = 0$ corresponds to a minimal maintenance while $0 < A_i^m < 1$ relates to imperfect maintenances.

The time to failure t^* follows the distribution conditioning on the machine's virtual age v_{ij}^m, i.e.,

$$p_i\left(t^*\middle|v_{ij}^m\right) = \frac{p_i\left(t^* + v_{ij}^m\right)}{\int_{v_{ij}^m}^{\infty} p_i(\tau)d\tau}, t^* > 0 \tag{4.3}$$

The maintenance cost consists of resource cost and production loss due to machine stoppage. The resource cost includes part replacement and other consumable expenses, which varies with maintenance level. Given a maintenance action $\vec{e}_{ij} = (i, m, t_{ij}, d_{ij}^m)$, depending on the maintenance level m, the cost of j^{th} maintenance action on machine S_i is evaluated as

$$C_{ij}^m = c_i^m + c_p \left(PL_{\vec{e}_{ij}}^m + PLR_{\vec{e}_{ij}}^m \right) \tag{4.4}$$

where c_i^m is the resource cost. c_p is the profit per part. $PL_{\vec{e}_{ij}}^m$ and $PLR_{\vec{e}_{ij}}^m$ are the permanent production loss and production loss risk caused by $\vec{e}_{ij}$ respectively. The latter two terms heavily rely on system states, and they will be derived in following sections through careful analysis on the production line dynamics

An optimal maintenance decision cannot be made directly based on the cost, since a replacement probably costs much higher than a minimal one, but it ensures the machine operate for a longer period of time. Therefore, the real-time maintenance cost rate R_{ij}^m is introduced as maintenance cost per unit time before next failure arrives.

$$R_{ij}^m = \frac{C_{ij}^m}{z_{ij}^m + d_{ij}^m} \tag{4.5}$$

where d_{ij}^m is the duration of maintenance, which is a deterministic value related to the maintenance type m, and z_{ij}^m is the lifetime of machine S_i after the maintenance, which is an unknown random variable following distribution $p_i\left(t^* \middle| v_{ij}^m\right)$ in Equation (3). The expected cost rate after its j^{th} maintenance is

$$E\left[R_{ij}^m\right] = \int_0^\infty \frac{C_{ij}^m}{t^* + d_{ij}^m} p_i(t^* | v_{ij}^m) dt^* \tag{4.6}$$

$E[R_{ij}^m]$ is the expected cost per unit time before the next failure. When a random failure occurs at a machine, a maintenance of a proper level should be chosen to minimize the expected cost rate. This cost rate formulation will be used to guide optimal decision making on maintenance levels.

4.4.Product loss and production loss risk evaluation

The maintenance action causes machine stoppage and leads to different system states. The impacts of a maintenance action $\vec{e}_{ij}$ on the system are twofold, i.e., permanent production loss $PL_{\vec{e}_{ij}}$ and production loss risk $PLR_{\vec{e}_{ij}}$. The permanent production loss is the direct outcome of the machine stoppage, and the production loss risk results from the disturbance to overall system states caused by $\vec{e}_{ij}$.

4.4.1. Permanent production loss evaluation

Opportunity window is the largest possible stoppage duration of a machine before such stoppage induces permanent production loss [22], [81], [82]. As discussed in previous section, the opportunity window depends not only the current buffer status but also the downtime events $\boldsymbol{E}$. In stochastic scenario, the full downtime event list $\boldsymbol{E}$ is unknown since there will be unexpected random failures in the future. However, in the context of maintenance, a subset $\widehat{\boldsymbol{E}}$ of downtime events $\boldsymbol{E}$ is known, namely those maintenance actions already initiated before or right at current time t and lasting beyond time t.

Since $\widehat{\boldsymbol{E}} \subset \boldsymbol{E}$, it can be concluded that $PL_{\widehat{\boldsymbol{E}}} \leq PL_{\boldsymbol{E}}$. Then the opportunity window estimated with $\widehat{\boldsymbol{E}}$ is the lower bound of that computed with $\boldsymbol{E}$. Therefore, the subset $\widehat{\boldsymbol{E}}$ can be used instead of $\boldsymbol{E}$ to safely estimate $OW_i(t)$.

$$OW_i(t) = \overline{OW}_i(t) + PL_{\widehat{\boldsymbol{E}}}[t, t + OW_i(t)] \cdot T_{M^*} \tag{4.7}$$

Consider a maintenance action $\vec{e}_{ij} = (i, m_{ij}, t_{ij}, d_{ij})$, if $d_{ij} \leq OW_i(t)$, the maintenance won't incur permanent production loss; however, if $d_{ij} > OW_i(t)$, then the slowest machine will be stopped and the production time loss is the excessive time that $\vec{e}_{ij}$ lasts beyond $OW_i(t)$. To conclude, the permanent production loss of a maintenance action $\vec{e}_{ij}$, denoted as $PL_{\vec{e}_{ij}}$, is

$$PL_{\vec{e}_{ij}} = \max\left\{\frac{d_{ij} - OW_i(t_{ij})}{T_{M^*}}, 0\right\} \tag{4.8}$$

4.4.2. Production loss risk evaluation

Given a maintenance $\vec{e}_{ij}$, it causes permanent production loss $PL_{\vec{e}_{ij}}$ when it lasts longer than its opportunity window and the loss last until $\vec{e}_{ij}$ is completed. Hence $PL_{\vec{e}_{ij}}$ only measures the impact of $\vec{e}_{ij}$ on the system within its presence. But the downtime event $\vec{e}_{ij}$ will impact the system in a more profound manner since it leads to totally different system states. The impact of a maintenance action beyond the "current" permanent production loss will be evaluated in this section.

The stoppage at machine S_i due to maintenance action $\vec{e}_{ij}$ spread to nearby machines sequentially when it causes blockage or starvation. The opportunity windows of nearby machines gradually alter. Considering a subsequent downtime event, the permanent production loss incurred by $\vec{e}_{k*}$ may also be changed due to the altered opportunity windows. In principle, the total loss cannot be directly attributed to the initial action $\vec{e}_{ij}$, but $\vec{e}_{ij}$ does indirectly impact the loss of the subsequent downtime event.

To ensure the resilience of the system to future random failures, we take the difference of production losses of the very first subsequent downtime event with and without $\vec{e}_{ij}$ as production loss risk, denoted as $PLR_{\vec{e}_{ij}}$.

Suppose that after time t_{ij}, the very first random failure occurs on machine S_k $(k = 1, 2, \dots, M)$ at time t^* and a replacement is taken. Since $\vec{e}_{k*}$ is the first random failure after time t_{ij}, there is no other downtime event between t_{ij} and $t_{ij} + t^*$. Only downtime events $\vec{e}_{ij}$ and $\vec{e}_{k*}$ are appended to the known downtime list $\widehat{\boldsymbol{E}}$. The full downtime list $\boldsymbol{E}$ is

$$\boldsymbol{E} = \left[\widehat{\boldsymbol{E}}, \vec{e}_{ij}, \vec{e}_{k*}\right] \tag{4.9}$$

With the current system state and fully observed downtimes $\boldsymbol{E}$ in the future, the buffer levels $\boldsymbol{b}(t_{ij} + t^*)$ at time $t_{ij} + t^*$ can be derived, which further can be used to compute opportunity window $OW_k(t_{ij} + t^*)$. Then the permanent production loss $PL_{\vec{e}_{k*}}(t^*)$ caused by $\vec{e}_{k*}$ can be evaluated as

$$PL_{\vec{e}_{k*}}(t^*) = \max\left\{\frac{d_k^{1c} - OW_k(t_{ij} + t^*)}{T_{M^*}}, 0\right\} \tag{4.10}$$

The probability associated with $PL_{\vec{e}_{k*}}(t^*)$ is $p(k, t_{ij}, t^*)$, which is the probability that the first random failure arrives at machine S_k at time $t_{ij} + t^*$. The machines are independent with each other regarding reliability. Therefore $p(k, t_{ij}, t^*)$ is the joint probability of M machines, i.e.

$$p(k, t_{ij}, t^*) = p_k(t_{ij}, t^*) \prod_{l=1, l\neq k}^{M} \left[1 - \int_0^{t^*} p_l(t_{ij}, \tau) d\tau\right] \tag{4.11}$$

The expected value of production loss of first downtime event with $\vec{e}_{ij}$ can be evaluated as

$$E[PL_{\vec{e}_{k*}}] = \sum_{k=1}^{M} \int_0^{\infty} p(k, t_{ij}, t^*) PL_{\vec{e}_{k*}}(t^*) dt^* \tag{4.12}$$

Note that $PL_{\vec{e}_{k*}}(t^*)$ is directly caused by the potential random failure $\vec{e}_{k*}$, not the initial downtime event $\vec{e}_{ij}$. The impact of $\vec{e}_{ij}$ lies in that it might alter the value of $PL_{\vec{e}_{k*}}(t^*)$. The production loss of $\vec{e}_{k*}$ without $\vec{e}_{ij}$ should also be evaluated in order to identify the real impact of $\vec{e}_{ij}$. Considering a scenario without the downtime event $\vec{e}_{ij}$, the full downtime list $\widetilde{\boldsymbol{E}}$ is

$$\widetilde{\boldsymbol{E}} = [\widehat{\boldsymbol{E}}, \vec{e}_{k*}] \tag{4.13}$$

Following the similar aforementioned procedure , at time $t_{ij} + t^*$, the buffer levels $\widetilde{\boldsymbol{b}}(t_{ij} + t^*)$, opportunity window $\widetilde{OW}_k(t_{ij} + t^*)$, and production loss $\widetilde{PL}_{\vec{e}_{k*}}(t^*)$ can be computed in sequence. Therefore, the expected production loss of first downtime event $\vec{e}_{k*}$ without $\vec{e}_{ij}$ is

$$E[\widetilde{PL}_{\vec{e}_{k*}}] = \sum_{k=1}^{M} \int_0^{\infty} \tilde{p}(k, t_{ij}, t^*) \widetilde{PL}_{\vec{e}_{k*}}(t^*) dt^* \tag{4.14}$$

where $\tilde{p}(k, t_{ij}, t^*)$ is the probability that machine S_k fails at time $t_{ij} + t^*$. Note that $\tilde{p}(k, t_{ij}, t^*) \neq p(k, t_{ij}, t^*)$, since $\tilde{p}(k, t_{ij}, t^*)$ is derived by assuming that machine S_i doesn't receive $\vec{e}_{ij}$. Finally, $PLR_{\vec{e}_{ij}}$ can be estimated by the difference of expected production losses with and without $\vec{e}_{ij}$.

$$PLR_{\vec{e}_{ij}} = E\left[PL_{\vec{e}_{k*}}\right] - E\left[\widetilde{PL}_{\vec{e}_{k*}}\right] \tag{4.15}$$

The term $PLR_{\vec{e}_{ij}}$ is the impact of $\vec{e}_{ij}$ on the production loss of next random failure. By incorporating it into the maintenance cost rate function, a maintenance decision at current time would always consider its impact on the whole system in future time.

4.5. CM control based on rolling horizon method

From the evaluation of production losses, we can conclude that the maintenance cost for machines in a manufacturing system heavily depends on the real-time system state. As discussed in Section I, it is extremely difficult to find an optimal maintenance policy for a global time horizon. Therefore, a feasible approach is to develop a control policy implemented on a real-time basis to obtain the near-optimal maintenance decisions.

At any time t, distributed sensors monitor machine operation status in the system. If random failure at machine S_i is detected, i.e., $W_i(t) = 1$, then a maintenance action should be taken on machine S_i. Depending on the structure of the machine, maintenance of multiple levels $m_i = 1c, 2c, \dots, N_i c$ might be available for the failed machine. For each eligible maintenance level m_i, the duration of maintenance d_m and age reduction factor A_i^m are known. By assuming a maintenance action $\vec{e}_{i*} = (i, m, t, d_{i*}^m)$ at machine S_i, the expected maintenance cost rate $E[R_{i*}^m]$ can be derived according to Equation (6). Then the optimal maintenance level m_{i*} should minimize the expected cost rate, i.e.

$$m_{i*} = \arg\min_m \{E[R_{i*}^m], m = ,1c, 2c, \dots, N_i c\} \tag{4.16}$$

4.6.Experiments and validation

To demonstrate the effectiveness of the proposed real-time maintenance policy, extensive numerical studies are performed. In baseline policies, upon random failures, the maintenance level is static. The other policy is the real-time maintenance policy proposed in this paper. The overall profit of the system, denoted as $\mathcal{P}(T)$, is taken as the main performance measure.

$$\mathcal{P}(T) = Total\ Revenue - Total\ Cost = c_p \cdot X_M(T) - \mathcal{M}(T) \tag{4.17}$$

where $c_p \cdot X_M(T)$ is the production revenue and $\mathcal{M}(T)$ is the total maintenance resource cost during time span $[0, T]$. Let $K_i^m(T), m = 1c, 2c, \ldots$ denotes the total number of level-m maintenance received by the machine S_i up to time T. The total maintenance resource cost is

$$\mathcal{M}(T) = \sum_{i=1}^{M} \sum_{\substack{all\ m \\ m=1c,\ldots,N_i c}} K_i^m(T) c_i^m \tag{4.18}$$

We construct 50 different serial production lines by randomly selecting machine and buffer parameters from the following sets:

$$M \in \{3, 20\}$$

$$T_i \in [1, 5]\ min, i = 1, 2, \ldots, M$$

$$B_i \in [2, 40], i = 2, 3, \ldots, M$$

$$b_i(0) \in [0, B_i], i = 2, 3, \ldots, M$$

In this case study, failure-time of the machines are assumed to follow Weibull distribution, which is a typical increasing-failure-rate distribution widely used in machine reliability analysis. The probability density function of failure-time of machine S_i is given as

$$p_i(t^*) = \exp\left(-\left(\frac{t^*}{\alpha_i}\right)^{\beta_i}\right) \tag{4.19}$$

where α_i is the scale parameter, and β_i is the shape parameter. The conditional probability of fail-time given its virtual age v_{ij}^m is

$$p_i\left(t^*\middle|v_{ij}^m\right) = \frac{\beta_i}{\alpha_i}\left(\frac{t^* + v_{ij}^m}{\alpha_i}\right)^{\beta_i} \exp\left(\left(\frac{v_{ij}^m}{\alpha_i}\right)^{\beta_i} - \left(\frac{t^* + v_{ij}^m}{\alpha_i}\right)^{\beta_i}\right) \qquad (4.20)$$

For every machine S_i in randomly generated production lines, the shape parameter is $\beta_i = 2$, and the scale parameter α_i and initial virtual age v_{i0} are randomly generated according to following set:

$$\alpha_i \in [500, 2000]\ min$$

$$v_{i0} \in [0{,}1000]\ min$$

For the ease of implementation and further analysis, the maintenance parameters of each machine are set to be identical. Four maintenance levels for each machine are given, namely replacement ($1c$), minimal maintenance ($4c$), and two levels of imperfect maintenance ($2c, 3c$). The parameters regarding each maintenance level is as shown in Table 4.1.

Table 4.1. Maintenance parameters for CM control case study

Maintenance Level m	Age Reduction Factor A_i^m	CM Duration d_i^m	Resource Cost c_i^m
$1c$	0	30	2000
$2c$	0.3	20	1200
$3c$	0.6	15	900
$4c$	1.0	5	300

Three static maintenance policies, i.e., policy $1c$, policy $2c$ and policy $3c$, are adopted for comparison. In these policies, upon random failures the machine will always receive a maintenance of level $1c, 2c$ and $3c$ respectively.

The profit per part is $c_p = 200\,\$$. Each production line is simulated to continuously run for 4 weeks, i.e., the simulation time horizon is $T = 40320\ min$.

For each randomly generated production line and each policy, the simulation is repeated for 10 times to compute the average overall profit.

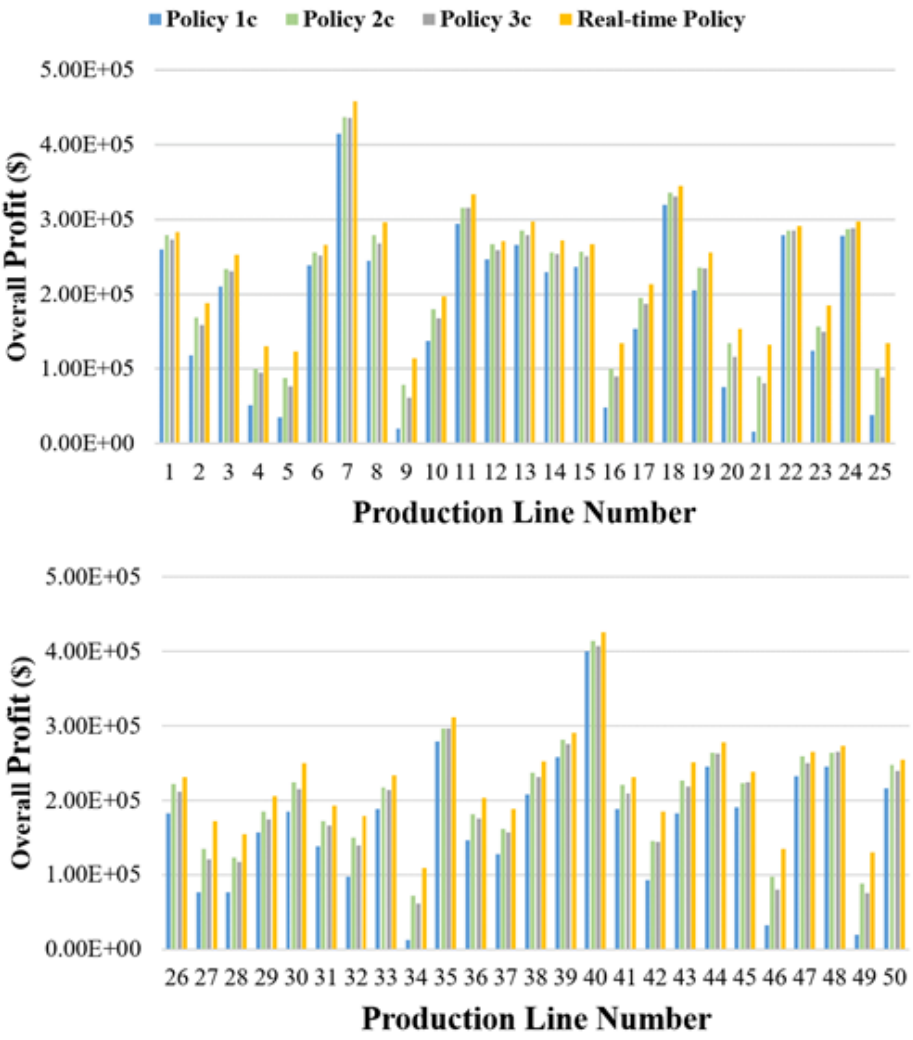


Figure 4.4. The comparison of overall profits among policies

Figure 4.4 illustrates the simulation result comparison. In all 50 production lines, the overall profits using real-time policy are greater than those using static maintenance policies. On average, the overall profit using real-time policy is 31.36%, 11.48% and 15.37% greater than those using policy $1c$, policy $2c$, and policy $3c$ respectively.

To further analyze our policy, a special production line is constructed. All six machines in the line have identical reliability parameters, but different cycle time

due to process planning. The slowest machine in the line is S_4. We calculate the total numbers of replacement and imperfect maintenance (including minimal maintenance) under the real-time maintenance policy during one-time simulation.

Table 4.2. Maintenance records for the CM case study

Machine	Total replacement count K_i^{1c}	Total imperfect maintenance count $\sum_{m=2c,3c,4c} K_i^m$	Total maintenance count $\sum K_i^m$
S_1	27	66	93
S_2	27	66	93
S_3	28	63	91
S_4 (slowest)	32	45	77
S_5	29	54	83
S_6	27	79	106

As shown in Table 4.2, although all the machines are identical regarding reliability parameters, it is noted that the slowest machine S_4 received more replacements and less imperfect maintenances than any other machines. The replacement number decreases, and imperfect maintenance number increases for the machine further away from the slowest machine. The machine far away from the slowest machine is more likely to take imperfect maintenance than replacement. Besides the stochastic factors, the reason for this phenomenon is mainly that the opportunity window tends to be larger for the machines far away from the slowest machine. Imperfect maintenances, which typically last shorter than a replacement, is less likely to induce permanent production loss on these machines and therefore is preferable to replacement.

One may also note that the total maintenance number are increasing with the distance from the slowest machine. The decrease in replacement times would inevitably impair the reliability of the machine, and thus more random failures can be expected. However, since these machines have larger opportunity windows, they are more resilient to random failures. It is reasonable for these machines to take

more imperfect maintenances in order to reduce resource costs. This case illustrates that the maintenance decisions could be very different for identical machines when they are placed at different locations in a multi-stage manufacturing system.

To conclude, the real-time maintenance policy proposed in this paper is effective to choose proper maintenance levels in accordance with the production system dynamics.

4.7. Summary

In this chapter, a rolling horizon method is used to solve the corrective maintenance decision making problem in manufacturing systems. The real-time maintenance cost rate is established to facilitate the real-time decision making on multi-level maintenance in a multi-stage manufacturing system. Based on the data-driven mathematical model of the manufacturing system, the permanent production loss and production loss risk incurred by the maintenance action are derived. The proposed control framework is able to make cost-effective maintenance decisions considering multiple maintenance levels.

4.8. Related work

Part of the results presented in this chapter have been published in [83]–[85].

Chapter 5. Reinforcement Learning for Preventive Maintenance Control

5.1.Background

Preventive maintenance (PM) is an intricate matter as it relates to many other aspects of modern industrial practices [86]. The PM policy aims in improving system reliability, preventing the occurrence of system failures, and reducing maintenance costs [56]. It is shaped by the specific application scenario and the characteristics of the target system. Regarding the structure of the system of interest, the maintenance policies can be categorized into single-unit policies and multi-unit policies. The single-unit policies are designated for those standalone systems and they have been extensively investigated by Wang [56]. Some examples are age-dependent policies [87] and periodic PM policies etc. Since the single-unit system operates independently, the relationship, either deterministic [87] or stochastic [88], between the maintenance decision and the overall maintenance cost is usually known. Therefore, the single-unit maintenance could often be modeled as a stochastic process, in which optimal PM decision variables can be obtained by minimizing the maintenance cost rate, or, equivalently, maximizing the machine availability in some circumstances. For the serial production line, the maximum machine availability does not guarantee an optimal maintenance cost. The system production loss caused by a maintenance action is conditioning on the buffer states [89], [90], for which in general we cannot derive the probability distribution [67].

Furthermore. it is noteworthy that real manufacturing systems vary with different available maintenance actions, and machine maintenance requirements, and system size. Random failure occurs when a machine deteriorates to a certain

level due to usage and aging. Upon the random failure, a maintenance action has to be carried out in order to restore the machine to an acceptable operating condition. Such a maintenance option is referred to as corrective maintenance (CM). The consequences of machine random failures can be unpredictable, and even catastrophic in some situations [56]. To reduce such random failures, the preventive maintenance (PM) is much needed, which proactively maintains a machine even it is not failed and keeps machines in a desired reliability level. Besides, in industrial practice, maintenance is not necessarily to replace the machine with a new one. Based on the structure of the system, multi-level maintenance options could be available. A perfect maintenance, or replacement, is recovering the machine 'as good as new', while a minimal maintenance is recovering the machine 'as bad as old', which only resumes its operation without changing the deterioration status. Imperfect maintenances are recovering the machine to somewhere between old and new.

Unlike in the single-unit system, the components within the multi-unit system have structural or operational dependencies on each other. Maintenance policies have been developed based on the specific structures of the systems, including serial systems [91], the parallel systems [92] and *k*-out-of-*n* systems [93] etc. The "group maintenance" and "opportunistic maintenance" are the building blocks for most of the existing maintenance policies for the close-interconnected serial systems. The group maintenance policy [59], [94] conducts multiple maintenance actions simultaneously to merge and reduce the production losses, while the opportunistic maintenance policy [95]–[97] identifies the time window, in which the inserted PM will not incur extra production losses. They are inspired by the observation that when one machine is under maintenance, the others can receive maintenance at the same time without incurring extra production loss. However, it does not hold in a general serial production line because the buffers among machines could delay the propagation of the machine stoppage from the maintained machine to its adjacent machines [89], [98].

It turns out that neither traditional single-unit policies nor multi-unit policies could be directly applicable to the serial production lines. Therefore, considerable research efforts have been devoted to deriving feasible maintenance policies for the serial production lines. There are several works [99]–[101] that are aimed to derive maintenance policy for two-machine-one-buffer serial production lines. Fitouhi et al. [99] presented a Markov Chain based method to evaluate the system performance under different PM policies, however, the policy considered in this work failed to incorporate the system dynamics since PM actions were determined based on only two variables, i.e. degradation states of the two machines. In the contrast, Karamatsoukis et al. [100] included the buffer levels in the state definition when they tried to obtain PM policies using Markov Decision Process (MDP). Wang et al. [101] derived the PM policy based on semi-MDP for a two-machine-one-buffer production line considering quality inspections. Machine degradation states are assumed to be directly related to product quality performance and non-conforming parts would be scrapped immediately. Although these works found feasible PM policies under different assumptions, the approaches proposed in [100], [101] lack scalability and cannot be extended to the more general cases with more machines and buffers.

For longer serial production lines, Arab et al. [69] searched for the optimal maintenance schedule using genetic algorithm in order to maximize the throughput. Ramirez-Hernandez et al. [102] used approximate dynamic programming (ADP) to optimize the maintenance schedule in a five-machine production line. However, both works simplifies the maintenance problem as inserting known maintenance tasks, in the form of downtime events, into the production shifts, and assumed that the maintenance schedules would not impact the machine reliability status at all. Kang et al. [103] proposed an aggregation-based approximation method for obtaining maintenance policies for the synchronous production line, i.e. the cycle time for each machine is identical. But real production lines are usually not perfectly balanced, so that it is important to consider the different machine processing speeds when optimizing PM policies [99]. In [104], a CM policy considering imperfect maintenance effects was proposed for the serial production

line, but the PM was not included in the work. Therefore, a systematic approach to deriving PM policies for general serial production lines must be developed to address the above challenges.

Regarding the mathematical techniques used in obtaining maintenance policies, quite a few methods have been applied in literatures, including renewal process [87], Markov Chain [99], heuristic methods [69], and MDP [99]–[103] etc. It is noted that MDP is a particularly common modeling method for maintenance problem in complex systems including serial production lines, since maintenance is often a sequential decision-making problem with multi-dimensional states and actions. However, it is important to realize that the performance of MDP-based maintenance policies can vary tremendously depending on the problem formulation and solving techniques.

On the one hand, the problem formulation refers to properly defining the three components, namely state, action, and reward, according to the problem characteristics and objective. It requires thorough understanding of system dynamics in serial production lines. For example, if some key variables are not included in the state definition, the PM decisions would fail to reflect the real system dynamics. In [99], the buffer level is not considered when making PM decisions, hence the PM decision might be the identical no matter the buffer is full or empty. But one should also strive not to include redundant variables, especially in today's manufacturing systems, which usually have huge amounts of data from various sensors. In this paper, the PM problem is also formulated as an MDP but with the guidance of our previously derived systematic knowledge of the serial production lines.

For years, scheduled and other "preventive" maintenance strategies have been the norm – achieving maintenance objectives through regular equipment inspections and scheduled maintenance at pre-determined intervals based on operational time, cycles, units, etc. However, this "fixed" PM policy may ignore a machine's real degradation and its impact on system level throughput loss. The complexity of a production system leads to an extremely large state space of the

maintenance problem. In the industrial practice, PM schedules are usually planned solely based on individual machine's ages recommended by machine vendors, which tend to be conservative and largely ignore the intricate interactions among machines in the serial production line. As the information of the production systems have been increasingly transparent and detailed, a PM policy would be preferable if all necessary machine-level and system-level information are fully incorporated into the policy. However, it inevitably leads to a problem with an enormous state space, which is intractable with traditional model-based planning methods. It is promising to embrace new tools and methodologies emerging in artificial intelligence and machine learning areas to develop intelligent decision-making support systems for production and maintenance management.

On the other hand, the solution to an MDP is an optimal policy that gives the best action for each state, such that the expected accumulated reward is maximized. There have been a lot of techniques that can effectively solve the MDP, and some of them have been applied to maintenance problems. Dynamic Programming (DP) is an exact and model-based approach to solving MDP, where model refers to the complete transition probabilities among states. Therefore, in the context of PM problem in serial production lines, DP is only applicable to two-machine-one-buffer line [100], or needs cruel approximations when applied to longer lines [103]. In the contrast, Reinforcement Learning (RL) is a category of techniques obtaining the optimal policy for MDP through the interactions between agents and the uncertain environment [105]. Most of the RL algorithms is model-free, i.e., the state transition probabilities are not required. Therefore, the model-free RL algorithms well suit the PM problem in general production lines, for which the system state space explodes exponentially with increased machine numbers. Instead of the transition probabilities, a reliable simulator, or experiment if feasible, that faithfully reflects the uncertain environment needs to be set up for the implementation of RL algorithms. Regarding the serial production line, the general-purpose commercial software, e.g., Simul8 and Simulink etc., have long been used for its simulation. However, the simulation setup is often arduous, and the efficiency and accuracy are not guaranteed. In [98], a data-driven model for production lines is established

based on dynamic system and conservation of the flow. The model is derived analytically, and therefore it is not only accurate but also has high computation efficiency compared to general-purpose simulation software. In this paper, we will leverage it to simulate the production system dynamics under PM for training process of the RL agent.

The selection of the RL algorithms is also a crucial question. According to how the policy is represented, RL algorithms can be categorized into policy-based methods, value-based methods, and actor-critic [105]. In RL, policy is a function mapping from state to action. The policy-based methods seek to directly parameterize the policy. Simple parameterizations could be, for example, linear combination of polynomial features or basis functions. The performance of policy-based methods heavily depends on how the features or basis functions are constructed. For problems with high dimension and inherent complexity, simple parameterizations are not sufficient due to their limitations on representation power. In contrast to policy-based methods, value-based methods represent the policy implicitly with state-values or state-action-values, where 'value' is the expected accumulated reward starting from a given state or taking a given action. The naïve Q-learning is one of the most widely used value-based methods in researches on PM problems because of its simplicity and robustness. Wang et al. [101] presents the application of naïve Q-learning to PM problem in two-machine-one-buffer line. However, to some extent, naïve Q-learning also suffers from the "curse of dimensionality" mainly because the naïve Q-learning uses a table to record the Q-value for all state-action pairs. The problem with large state spaces is not just the memory needed for large tables, but the time and data needed to fill them accurately [105]. The actor-critic method adopts both policy parameterization and value function in its algorithm, and therefore suffers from the drawbacks of both. In conclusion, despite the fact that these primitive RL algorithms are robust and accessible, the lack of scalability is preventing them from being applied to solve a range of real-world problems with large state space like the PM problem discussed in this paper. To this end, in recent years, the emergence of deep learning allows the RL to go 'deep' as well and results in a series of DRL algorithms. The DRL

scales RL to interesting decision-making problems in practice that were previously intractable [106]. Some of the DRL applications in recent years include, for example, Atari video games [107], [108] and the game of Go [109] etc. Thanks to its good scalability and efficiency, the DRL shows great potential to solve complex problems in the manufacturing industry that are unsolvable with conventional techniques.

5.2.PM problem description

We consider a serial production line that consists of M machines and $M-1$ buffers with limited capabilities as shown in Figure 5.1. The arrows depict the direction of material flow in the system. The material is referred to as a final product once it has been processed by all machines sequentially. Otherwise, it is said to be an intermediate part.

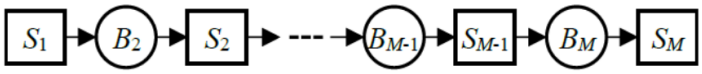


Figure 5.1. The structure of a serial production line

The serial production line is described as follows:

- Each buffer B_i has a finite capacity. With the abuse of notation, the maximum capacity of buffer B_i is also denoted as B_i.
- Each machine S_i has a rated cycle time T_i.
- The lifetime of machine S_i follows a known distribution $p_i(x)$, which can usually be obtained by experiments or from vendors.
- The maintenance durations d_i^{PM} and d_i^{CM} include not only the time performing maintenance but also the response time and preparation time needed before the maintenance starts.
- Both CM and PM would incur some fixed resource costs, including costs of new parts and all other consumable expenses. The resource costs of a CM and a PM on machine S_i are c_i^{CM} and c_i^{PM} respectively.

An intermediate part finished by machine S_i flows to its downstream buffer B_{i+1} if buffer B_{i+1} is not full, and otherwise machine S_i is said to be blocked. Machine S_i starts a new cycle by receiving one part from its upstream buffer B_i if buffer B_i is not empty, and otherwise machine S_i is said to be starved. The blockage or starvation makes an operational machine to stand idle. If one machine is undergoing maintenance, its downstream buffers gradually drain, and upstream buffers fill up due to the machine stoppage and thus causing blockage or starvation in its adjacent operational machines. The stoppage and idleness of these machines might finally lead to the system-level production loss. The production loss due to the maintenance activities accounts for a significant portion of the overall maintenance related costs. We denote the system production loss caused by the maintenance activities as PL.

The extent to which the maintenance action can restore a machine's health state is referred to as maintenance effect. In this work, we use Kijima Model II [110] to model the effects of different levels of maintenance actions. Let g_i denote the machine's age prior to maintenance, then the machine's age immediately after the maintenance, denoted by g_i', is given by

$$g_i' = g_i \times r_i \tag{4.1}$$

where $0 \leq r_i < 1$ is the recovery factor of the maintenance on machine S_i. $r_i = 0$ indicates a perfect maintenance since the machine age g_i' is set to be zero after maintenance. By contrast, $0 < r_i < 1$ relates to an imperfect maintenance, as the post-maintenance age g_i' , also referred to as 'virtual age' in Kijima (1989), starts somewhere between zero and the original age g_i. In other words, imperfect maintenance does not fully restore machine's health condition and hence earlier.

Let π denotes the PM policy for a serial production line. The PM policy π instructs when and which machine should be turned off and receive a PM. Let $C(t;\pi)$ denotes all the costs caused by the maintenance activities up to time t under the PM policy π, then

$$C(t;\pi) = c_p \cdot \int_0^t PL(t)dt + \sum_{i=1}^{M} c_i^{PM} N_i^{PM}(t) + \sum_{i=1}^{M} c_i^{CM} N_i^{CM}(t) \quad (4.2)$$

where c_p is the profit per part and $\int_0^t PL(t)dt$ is the accumulative system production loss up to time t. $N_i^{PM}(t)$ and $N_i^{CM}(t)$ are the total PM and CM times conducted on machine S_i up to time t respectively.

$$N_i^{PM}(t) = \int_0^t a_i(\tau)d\tau \quad (4.3)$$

$$N_i^{CM}(t) = \int_0^t w_i(\tau)d\tau \quad (4.4)$$

An optimal PM policy π^* should minimize the long-run maintenance cost rate, which is maintenance cost per unit time. Therefore, the objective of the maintenance problem in this paper can be represented as:

$$\pi^* = \arg\min_{\pi} \left\{ \lim_{t\to\infty} \frac{C(t;\pi)}{t} \right\} \quad (4.5)$$

With this objective function, the problem to be studied in this paper is to develop methods to find the optimal PM policy π^* for the serial production line, such that the long-run maintenance cost rate is minimized. Since the production systems vary dramatically in their scales, ranging from two-machine line to systems consisting of dozens of machines, it is important to ensure that the developed methods should cover all those different scales.

5.3.Deep RL for PM decision making

The PM decision making problem in manufacturing system is formulated as a reinforcement learning problem. In the context of PM problems, the stepwise decision is to determine whether or not we conduct PM actions on machines. The rule governing the action selection is referred to as a policy, denoted as $\pi(a|s)$.

$$\pi(a|s) = \Pr(a_t = a|s_t = s) \quad (4.6)$$

Subsequently, a scalar reward r_t will be observed, which reflects the goodness of the action a_t in state s_t. The accumulated reward is referred to as return, denoted as G_t.

$$G_t = r_t + \sum_{k=1}^{\infty} \gamma^k r_{t+k} \tag{4.7}$$

where γ is a discount factor, which is used to make a trade-off between immediate reward and future rewards. If the immediate reward is preferred, a small γ will be used and vice versa. An optimal policy π^* is supposed to maximize the expected return, i.e.

$$\pi^* = \arg\max_{\pi} \{\mathbb{E}_\pi[G_t] | s = s_t\} \tag{4.8}$$

Before we can apply RL algorithms to obtaining the ultimate PM policy π^*, we need to first define the three key components properly, i.e., state s_t, action a_t, and reward function r_t, in the MDP that models the PM problem in serial production lines.

Given the state s_t, one should be able to fully comprehend the production system status such that an action that suits the status can be determined. Three factors are essential for the PM decision making in the serial production line, namely

- The machine ages $g_i(t), i = 1,2, \dots, M$, specify the probability of random failures on each machine;
- The buffer levels $b_i(t), i = 2,3, \dots, M$, denote the status of the production line, which directly relate to the system production losses caused by PM actions and random failures;
- The remaining maintenance duration $d_i^r(t), i = 1,2, \dots, M$, indicate all the ongoing maintenance activities on each machine.

Consequently, the state s_t is defined as:

$$s_t = [g_1(t), \dots, g_M(t), b_2(t), \dots, b_M(t), d_1^r(t), \dots, d_M^r(t)] \tag{4.9}$$

In contrast with the PM, the CM is passively triggered by a random failure, which is beyond the control of the agent. Therefore, the action a_t is limited to the PM decisions. The action a_t is a vector consisting of M binary variables indicating whether we turn off machines for PM or not at time t.

$$a_t = [a_1(t), a_2(t), \dots, a_M(t)] \tag{4.10}$$

where,

$$a_i(t) = \begin{cases} 0, & \text{leave machine } S_i \text{ as it is} \\ 1, & \text{turn off machine } S_i \text{ for PM} \end{cases} \tag{4.11}$$

Note that $d_i^r(t) > 0$ implies that there is an ongoing maintenance on machine S_i, and thus machine S_i cannot receive a PM under such circumstance. The action a_t is said to be illegal if it intends to assign a PM on machine S_i given $d_i^r(t) > 0$. At each time step, it is only allowed to select an action from the legal action sets $A(s_t)$.

$$A(s_t) = \{a_t | \forall i, a_i(t) = 0, \text{if } d_i^r(t) > 0\} \tag{4.12}$$

For simple serial systems without intermediate buffers, any machine stoppage would immediately count towards the system production loss. However, with the existence of intermediate buffers, the relationship between system production loss and machine stoppage duration is not trivial. It has been proved in [89] that *in a serial production line a maintenance action causes system production loss if and only if the slowest machine is impeded, i.e. blocked or starved, by the stoppage*. Let S_{M^*} denotes the slowest machine in a serial production line, where $M^* = \arg\max_i \{T_i, i = 1,2, \dots, M\}$. In the stepwise simulation scenario, the system production loss $PL(t)$ can also be conveniently calculated as

$$PL(t) = \frac{1}{T_{M^*}} - \left(Y_{M^*}(t) - Y_{M^*}(t-1)\right) \tag{4.13}$$

where $1/T_{M^*}$ is the ideal production increment of the slowest machine S_{M^*} without any disturbance, and the second term $Y_{M^*}(t) - Y_{M^*}(t-1)$ is the actual incremental production counts of the slowest machine S_{M^*}. The difference between them is the system production loss during the time step. The overall maintenance cost includes

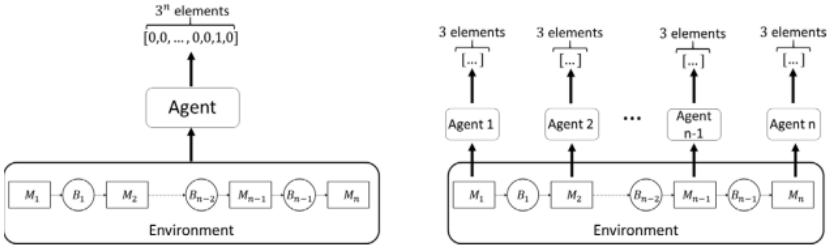
the resource cost of all PMs and CMs, and the system production loss caused by those maintenances. Therefore, the reward function r_t is defined as:

$$r_t = -c_p \cdot PL(t) - \sum_{i=1}^{M} w_i(t) c_i^{CM} - \sum_{i=1}^{M} a_i(t) c_i^{PM} \tag{4.14}$$

where $\sum_{i=1}^{M} w_i(t) c_i^{CM}$ and $\sum_{i=1}^{M} a_i(t) c_i^{PM}$ are the resource costs at time step t incurred by CMs and PMs respectively, and $c_p \cdot PL(t)$ is the profit loss caused by the system production loss $PL(t)$ during the time step. The reward is negative because we seek to maximize the accumulated reward, and equivalently the overall maintenance cost can be minimized.

5.4.Deep MARL for scaled up problem

The RL-based framework has issues being generalized to larger scaled applications due to action space explosion. In comparison, MARL is less prone to action space explosion as its action space is independent of number of agents in the system [101], [111], [112]. This is because, under the MARL's decentralized setting, the agents make decisions independently.



(a) Serial production line RL formulation (b) Serial production line MARL formulation

Figure 5.2. Comparison of action space size between RL and MARL formulation

Figure 5.2 depicts the action spaces of a n-machine production line under RL and MARL respectively. For MARL formulation that employs the CTDE framework, agents condition their actions on the local partial observations and are optimized by the gradients that are dependent on global states. Therefore, we need to design local observations o that provide the basis for agents' actions and global

states s that encapsulate information that is useful for training. Given a machine S_i three factors that are essential to its PM decision making are:

- The age of machine S_i, g_i.
- The upstream and downstream buffer status related to b_{i-1} and b_i.
- Machine's remaining maintenance duration, d_i^r, if applicable. In addition, the status of S_i's immediate adjacent machines is also useful because it might relate to the blockage or starvation of machine S_i.

Consequently, the local observation for machine Ma is defined as:

$$o^n = [g_i, g_{i-1}, g_{i+1}, b_{i-1}, B_i - b_i, d_i^r, d_{i-1}^r, d_{i+1}^r] \quad (4.15)$$

Note that we use buffer vacancy $B_i - b_i$ instead of buffer level b_i to represent the real-time status of the downstream buffer. This is because the buffer vacancy is a more sufficient criteria than buffer level in determining if the machine is blocked or not. Furthermore, the s that represents the global state of the production line is obtained by concatenating local observations from all machines and it is written as:

$$s = [o^1, o^2, \dots, o^M] \quad (4.16)$$

Action definition and reward definitions are similar to those in RL formulation. The minor difference is that the action is tweaked to include more levels of imperfect maintenance actions.

5.5.Experiments and validation

5.5.1. Double Deep Q-Network for PM control

Traditional tabular Q-learning uses a table to keep records of all the state-action values. However, when the state space of the problem is larger, tabular Q-learning can be inefficient and even infeasible. Given the definition of the state s_t in previous section, the machine ages $g_i(t) \in [0, \infty)$, the buffer levels $b_i(t) \in \{0,1, \dots, B_i\}$, and remaining maintenance time $d_i^r(t) \in [0, d_i^{CM}]$. Theoretically, the state space of the problem discussed in this paper is infinite. To deal with such

problems with extremely large state space, researchers strived to approximate the Q-values $Q(s,a)$ with some functions $\hat{Q}(s,a;\boldsymbol{\theta})$.

$$\hat{Q}(s,a;\boldsymbol{\theta}) \approx Q(s,a) \tag{4.17}$$

where $\boldsymbol{\theta}$ is the parameters of the approximation function. Instead of learning the exact Q-values for all state-action pairs, the agent learns the parameters $\boldsymbol{\theta}$ that can generalize a function well to approximate all the Q-values. Among the existing Q-learning algorithms with function approximations, the Double Deep Q-Network (DDQN) [108] is the state-of-the-art algorithm. In the DDQN, the 'network' refers to the neural network [113] that is used to approximate the state-action values. The typical architecture of the neural network for the state-action value approximation is as shown in Figure 5.3. The inputs are the state s, and the outputs are state-action values $\hat{Q}(s,a^n;\boldsymbol{\theta}), n = 1,2,\dots,N$, where N is the total number of possible actions. In a serial production line with M machines, the number of possible actions is $N = 2^M$.

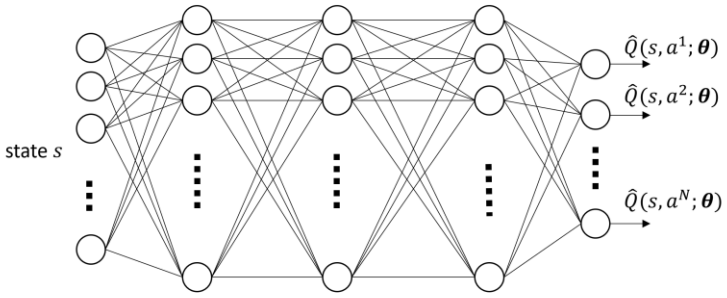


Figure 5.3. A typical architecture of the Q-network

The reinforcement learning problem formulated for PM decision making is trained through DDQN, shown in Figure 5.4. The neural network has two fully connected hidden layers, and each layer has 64 hidden units. Since the machine number is 4, the size of the input layer is 11 and that of output layer is 16. The size of the experience memory is N_{mem}= 500,000. The minibatch size is b =32. The step intervals for the state randomization and neural network duplication are $C_1 = 1{,}000$ and $C_2 = 10{,}000$. The decision time interval during training is chosen as k = 30 mins, i.e., every 30 minutes the agent needs to determine a PM action and input

it to the production system. The future reward discount factor is set to be γ = 0.95. The parameter for ϵ-greedy is initially set to be 0.8, linearly reduced to 0.1 when the iteration reaches 300,000 steps and fixed to 0.1 afterwards. Regarding the gradient descent, the optimizer used in this case is *RMSprop* [114], in which three parameters are η=0.00025, $\epsilon' = 0.01$, and ρ=0.95. We implement the proposed algorithm using TensorFlow with 4 GPUs and 4 CPUs. The total training steps are 2,000,000.

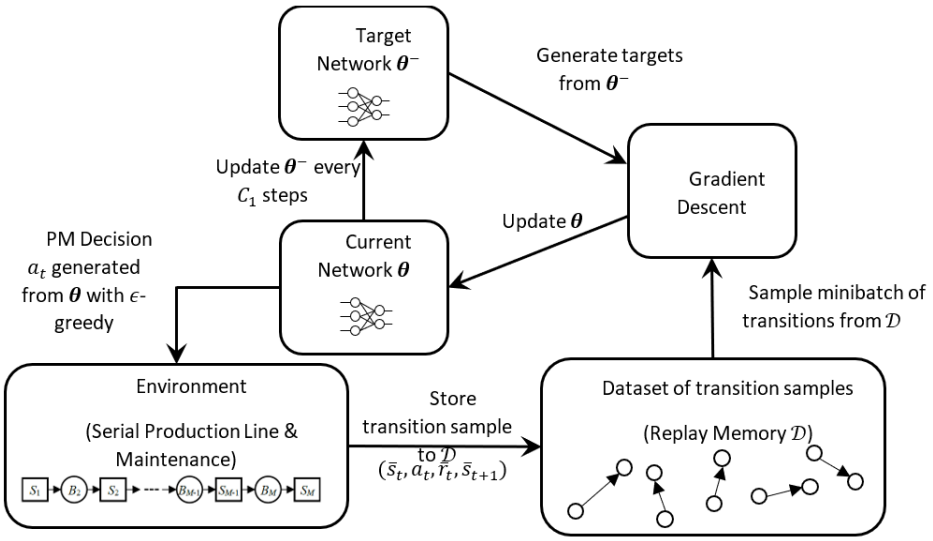


Figure 5.4. The flowchart of the DQN training process for PM decision making

To monitor the training progress, every 10,000 steps we run the production system with the PM policy defined by the latest neural network parameters $\boldsymbol{\theta}$ for 100,000 minutes and observe the average rewards per minute, which is as shown in Figure 5.5. It can be observed that the agent is making steady progress throughout the training iterations despite some noises.

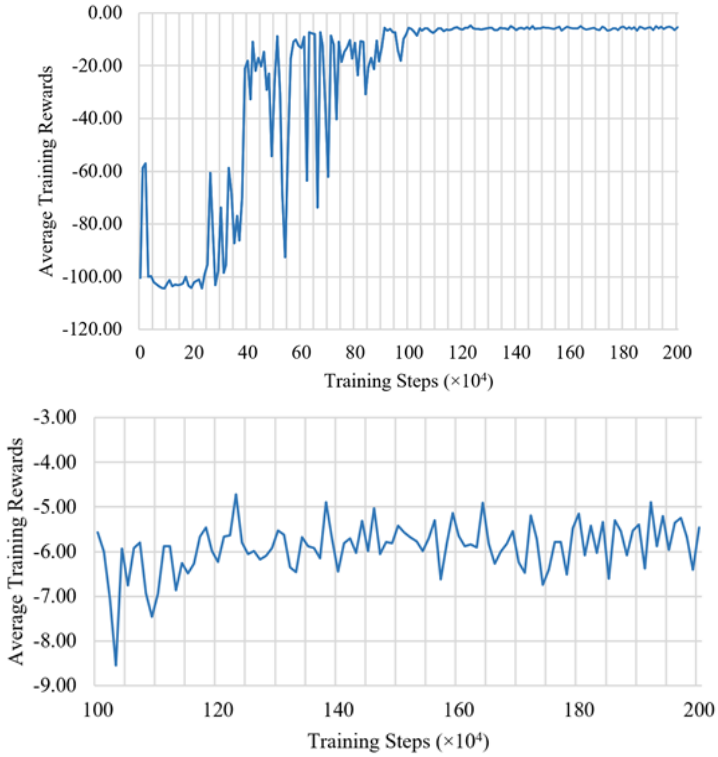


Figure 5.5. DQN training progress shows convergence

To evaluate the performance of the learned policy, three other scenarios are considered, including the run-to-failure scenario, the age-dependent PM policy, and opportunistic PM policy.

Run-to-Failure (R2F): The Run-to-Failure scenario is used to evaluate the system performance if no PM is conducted throughout the time horizon. Each machine in the production line keeps running until it encounters a random failure. Hence, CM is the only type of maintenance that is conducted in this scenario. Any other PM policy is deemed as effective only when it improves system performance from the Run-to-Failure scenario.

Age-Dependent Policy: In this policy, we follow the traditional norm - one machine will receive a PM whenever its age exceeds a predetermined age threshold. The age-dependent policy is the most widely used PM policy in the real industry. The optimal age threshold is derived to minimize the cost rate of each individual machine according to Ross [87].

Opportunistic Maintenance (OM): OM policy is inspired from the observation that once there is a machine undergoing CM, other machines can receive PM without incurring extra system production losses. Under OM policy, whenever one or multiple machines fails in the serial production line, we will conduct CM on those failed machines. In the meantime, we will turn off all other operational machines for PM.

According to the experiment results, all the three policies are effective PM policies, because they all improve the system performance from the run-to-failure scenario. However, the learned policy outperforms both the age-dependent policy and opportunistic policy in all the ten initial states sets. On average, the learned policy reduces the overall maintenance cost rate by 8.77% and 6.25% comparing to the age-dependent policy and opportunistic policy, respectively. Table 5.1 also lists the 95% confidence interval for the average maintenance cost rates under difference PM policies. We can observe that the 95% confidence intervals of the average cost rates for the learned policy are much lower and have no overlap with that for other policies, which indicates strong statistical significance.

Table 5.1. Average maintenance cost rates and 95% confidence intervals

Initial State	Run-to-failure	Age-dependent Policy	Opportunistic Policy	Learned Policy
1	7.92 [7.81, 8.02]	6.68 [6.53, 6.83]	6.56 [6.41, 6.7]	6.18 [6.06, 6.31]
2	7.98 [7.9, 8.07]	6.84 [6.72, 6.95]	6.57 [6.44, 6.69]	6.15 [6.03, 6.28]
3	7.89 [7.77, 8.01]	6.86 [6.75, 6.96]	6.54 [6.41, 6.68]	6.12 [6.0, 6.24]
4	7.88 [7.76, 7.99]	6.71 [6.59, 6.82]	6.6 [6.49, 6.71]	6.1 [5.98, 6.22]
5	7.92 [7.82, 8.02]	6.72 [6.59, 6.86]	6.47 [6.36, 6.58]	6.16 [6.03, 6.28]
6	7.8 [7.69, 7.92]	6.61 [6.48, 6.74]	6.53 [6.41, 6.66]	6.18 [6.04, 6.33]
7	7.93 [7.81, 8.04]	6.78 [6.65, 6.9]	6.49 [6.38, 6.6]	6.09 [5.97, 6.21]
8	7.78 [7.66, 7.9]	6.7 [6.55, 6.84]	6.51 [6.36, 6.65]	6.03 [5.92, 6.14]
9	7.9	6.64	6.59	6.09

	[7.77, 8.03]	[6.51, 6.77]	[6.47, 6.72]	[5.99, 6.2]
10	7.8 [7.69, 7.92]	6.65 [6.53, 6.76]	6.53 [6.39, 6.67]	6.2 [6.09, 6.31]

To closely examine the learned policy, we take a random initial state and run the production line with the learned policy. 20 consecutive maintenance records along with the states when each maintenance was conducted are drawn from the maintenance history. The records are listed in Table 5.2.

Table 5.2. Maintenance record and GM/OM analysis

Time $\boldsymbol{t}$ (min)	PM/CM	Failed machines	Machine ages $\boldsymbol{g_i(t)}$	PM Decisions $\boldsymbol{a_t}$	GM	OM
5781	CM	S_6	[326, 17, 104, 101, 100, 152]	-		
5782	PM	-	[327, 18, 105, 102, 101, 0]	[1, 0, 0, 0, 0, 0]		Y
6018	CM	S_6	[227, 254, 341, 338, 337, 212]	-		
6019	PM	-	[228, 255, 342, 339, 338, 0]	[1, 0, 0, 1, 1, 0]	Y	Y
6020	PM	-	[0, 256, 343, 0, 0, 0]	[0, 0, 1, 0, 0, 0]		Y
6044	CM	S_2	[16, 280, 17, 15, 14, 1]	-		
6217	CM	S_2	[189, 142, 190, 188, 187, 174]	-		
6218	PM	-	[190, 0, 191, 189, 188, 175]	[0, 0, 1, 1, 1, 1]	Y	Y
6219	PM	-	[191, 0, 0, 0, 0, 0]	[1, 0, 0, 0, 0, 0]		Y
6588	PM	-	[360, 340, 361, 358, 357, 360]	[0, 0, 1, 1, 1, 1]	Y	
6589	PM	-	[361, 341, 0, 0, 0, 0]	[1, 0, 0, 0, 0, 0]		
6680	CM	S_1	[81, 432, 85, 82, 81, 84]	-		
6916	CM	S_5	[208, 668, 321, 318, 317, 320]	-		
6917	PM	-	[209, 669, 322, 319, 0, 321]	[0, 0, 1, 1, 0, 0]	Y	Y
6918	PM	-	[210, 670, 0, 0, 0, 322]	[1, 1, 0, 0, 0, 1]	Y	Y

The 15 consecutive maintenances include 9 PMs and 6 CMs. Interestingly, the "opportunistic maintenance" (OM) and the "group maintenance" (GM) can be observed in the records. To be specific, OM is to conduct PMs on certain machines when there is an unscheduled failure or repair "opportunity" on other machines, and GM is to conduct PMs on multiple machines simultaneously to reduce the total maintenance related cost. In many existing studies on the maintenance problem in multi-unit systems, the GM and OM are the starting points for deriving maintenance policies [59], [96]. In other words, those studies first restrict the maintenance actions to GM or OM, and further optimize the decision variables for GM and OM to obtain the final policies. For example, the opportunistic PM policy that was used

for comparison purpose in this research has one significant decision variable τ, which is the time interval to conduct PM on all machines. However, in this research we do not take this heuristic approach. Rather, we formulate the policy as a learning problem based on system property (i.e., permanent production loss) and indeed the learned policy is able to make GM and OM decisions when appropriate. From the construction of the reward function, the agent is never explicitly rewarded if it conducts the GM or OM. The decisions of OM and GM are something that the agent learned itself throughout the training process.

When looking into the learning process, it is not difficult to explain the phenomenon. For the GM, the action space includes all the possible combinations of the PM actions on each machine, which means that the GMs are always available for the agent to select. If the GM action in some state yielded better accumulated reward, then the algorithm would increase state-action value to favor the GM in the particular state in the future. Similarly, during the training process the agent might also conduct OM when there are random failures on other machines, which would also change the state-action value to encourage or discourage the OM in the future.

Therefore, the GM and OM observed in the learned policy is a logical outcome as long as the problem formulation is rational, and the solution technique is effective. This interesting finding further validates the deep reinforcement learning based approach proposed in this dissertation.

5.5.2. MARL for scaled up PM control problems

In this experiment, the Value Decomposition Actor Critic (VDAC) [115] is applied to the MARL policy training. VDAC consists of distributed actors that make decisions for designated machines and a central critic that estimate the global state-value. The architecture of VDAC is as shown in Figure 5.6

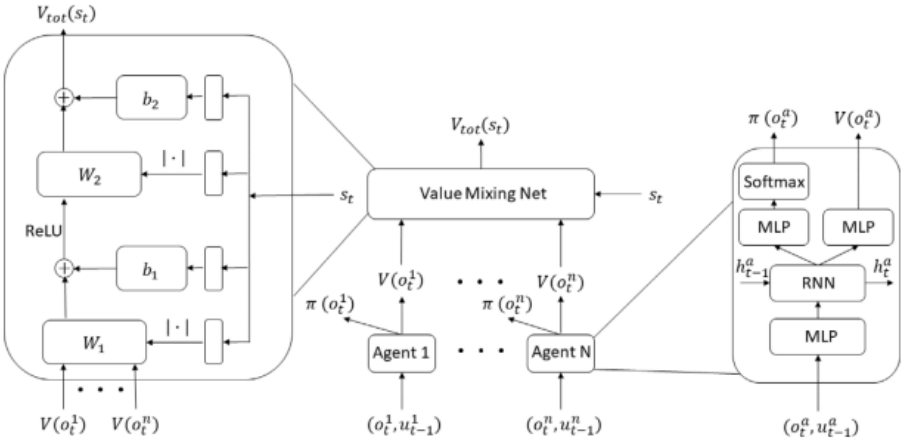


Figure 5.6. VADC architecture

VDAC consists of distributed actors that make decisions for designated machines and a central critic that estimate the global state-value V_{tot}. As shown Figure 5.6, the actor network takes inputs as observations o_t^a and actions u_{t-1}^a of the previous step $t-1$, outputs a multinomial policy $\pi(o_t^a)$ for timestep t. It also estimates the state-value of the local observation $V(o_t^a)$. To capture the temporal dependencies within agents' trajectories, Gated Recurrent Unit (GRU) is Incorporated in actor networks. Note that to speed up training as well as save memory, actor networks share the same weights. The recent advance of MARL literature [111], [112], [116], [117] is coupled by this parameter sharing technique. The value mixing network, which takes input as local state-values $V(o_t^a)$ and outputs the global state-value $V_{tot}(s)$, serves as a central critic. To incorporate the global state information that is unavailable to actors, the parameters of the value mixing network is generated from a hypernetwork which takes input as the global state s_t.

Actors are optimized by following gradients that depend on the central critic. Let θ_π to denote actor network parameters and θ_V to denote hypernet parameters for generality. The actor network is optimized by following the policy gradient given by:

$$\nabla_{\theta_\pi} J = \mathbb{E}_\pi \left[\sum_n \nabla_{\theta_\pi} \log \pi(u^n|\tau^n) \left(Q(s, \boldsymbol{u}) - V(s) \right) \right] \tag{4.18}$$

where $V(s)$ is estimated by a central critic and $Q(s, \boldsymbol{u}) = r + \gamma V(s')$. The critic is optimized by minibatch gradient descent to minimize the following loss:

$$L_t(\theta_V) = \left(y_t - V_{tot}(s_t; \theta_V)\right)^2 \tag{4.19}$$

where $y_t = \sum_{i=0}^{k-1} \gamma^i r_t + \gamma^k V(s_{t+k})$ is the target value.

The training process of VDAC is conducted using distributed A2C framework, where multiple simulation episodes are rolled out parallelly to increase the computing efficiency. The training procedure for the PM control problem is as shown in Figure 5.7. The training procedure can be generally divided into two parts: (1) Obtaining experience, and (2) optimizing parameters. In the phase of obtaining experience, multiple episodes are rolled out independently to increase experience sampling efficiency. Note that central critic is absent during this phase. During the parameter optimization phase, the data acquired in the sampling phase will be discarded once it is used to optimize network parameters. Therefore, no experience replay buffer is needed.

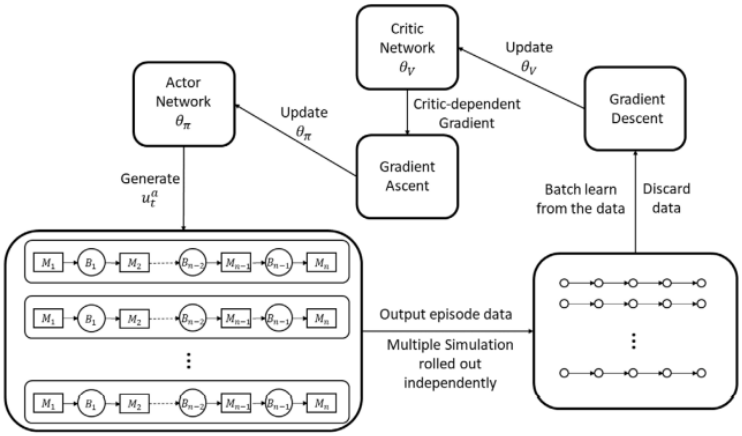


Figure 5.7. Training process of VDAC for PM problem

The training monitoring as shown in Figure 5.8 indicates steady improvement and finally convergence of the training process.

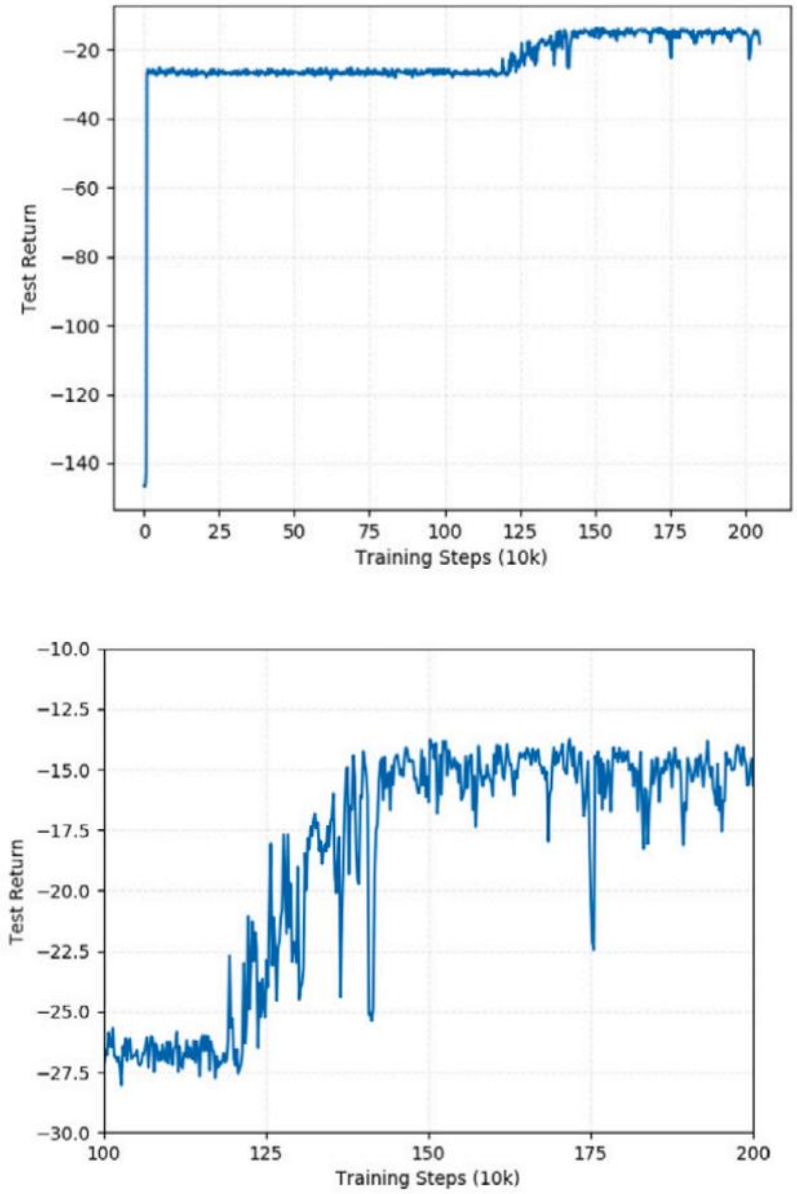


Figure 5.8. Training progress shows convergence

The learned distributed policies are tested against several benchmarks, including DDQN. In this experiment, run-to-failure and opportunistic maintenance (OM) policy are considered along with other three policies that work for situations where imperfect maintenance effect is considered:

Deep Q-learning (DQN): The single-agent RL approach also presented in this chapter. DQN algorithm works well for a six-machine-five-buffer system when

only considering perfect maintenance effect. In this case study, both the number of machines and number of PM options are increased, which lead to a much larger action space.

Group Maintenance (GM): Given GM policy, all the machines would receive PM simultaneously. GM policy tends to reduce the impacts of PM on the whole system by enforcing concurrent machine stoppages. There is one pivotal decision variable in GM policy, which is the time interval τ for carrying out group PM. Since there is no analytical method to derive τ due to the ultra-complexity of the production system, the optimal τ can be found through Monte Carlo simulation.

Opportunistic Group Maintenance (OGM): OGM policy is a combination of OM policy and GM policy. Similar to GM, there is also a predefined time interval τ. If there is a random failure occurs before the time interval τ arrives, OM policy would be triggered, i.e., all other operational machines are turned off for PM. If the time interval τ is reached without machine random failures, GM policy would be carried out so that all the machines receive PM simultaneously. We also leverage Monte Carlo simulation to derive the optimal τ.

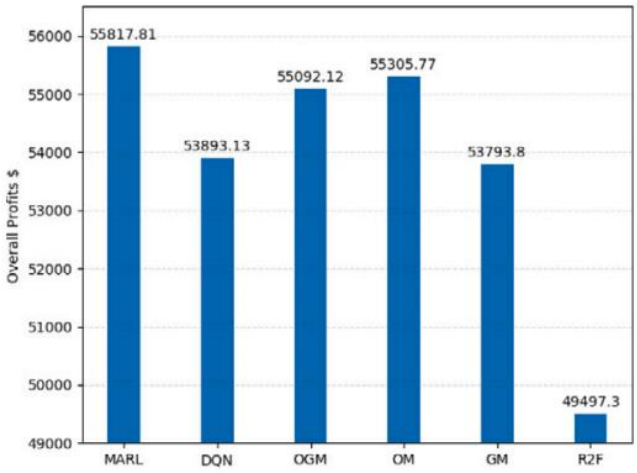


Figure 5.9. Performance comparison between different policies

In general, all methods other than R2F are effective policies. This is because they achieved higher average profit over R2F, as shown in Figure 5.9. Among the effective policies, MARL policy reports the best average profit and throughput per

episode. OM policy performs slightly better than OGM in profit although the OM policy reports less throughput than OGM. And GM has the least profit compared with other 3 effective policies. For baseline policies such as OGM, OM, and GM, they are set to either always perform level 1 PM or always perform level 2 PM. Since level 1 PM required less resource and time, the best performance is achieved by choosing level 1 PM. OGM and GM have a decision variable τ which is searched through a pyramid number of simulation and therefore possibly captures the dynamics in the production line.

Table 5.3. A portion of maintenance records by MARL agents for GM examination

Time	Machine age	Action	GM
649	[89, 57, 267, 341, 197, 497]	[0, 0, 0, 0, 0, 2]	
759	[199, 167, 377, 451, 307, 101]	[0, 0, 0, 2, 0, 0]	
814	[254, 222, 432, 44, 362, 156]	[0, 0, 2, 0, 0, 0]	
847	[287, 255, 25, 77, 395, 189]	[2, 0, 0, 0, 0, 0]	
585	[1, 266, 36, 88, 406, 200]	[0, 0, 0, 0, 2, 0]	
891	[34, 299, 69, 121, 21, 233]	[0, 2, 0, 0, 0, 0]	
1144	[287, 244, 322, 374, 274, 486]	[2, 0, 0, 0, 0, 0]	
1155	[1, 255, 333, 385, 285, 497]	[0, 0, 0, 0, 0, 2]	
1188	[34, 288, 366, 418, 318, 24]	[0, 2, 0, 0, 0, 0]	
1221	[67, 24, 399, 451, 351, 57]	[0, 0, 0, 2, 2, 0]	Y

To examine the MARL policies patterns, a portion of maintenance records is also pulled from the simulation history as shown in Table 5.3. At $t = 1221$, machines S_4 and S_5 conduct PM simultaneously. Note that machine S_5 conducts PM at an age of 406 at $t = 858$, whereas it conducts PM at an age of 351 when $t = 1221$. The PM at $t = 1221$ can be seen as an GM as machine S_5 accommodates its maintenance schedule to that of machine S_4. Therefore, by extending from single-agent RL to MARL, the GM pattern is preserved, which implies that the PM problem formulation in MARL is also effective. In conclusion, the proposed MARL framework is validated by the numerical experiments. The learned policy outperforms all other benchmarks.

5.6. Summary

The PM decision making in a serial production line is a complex problem due to its exploding state space and complicated interactions among machines. The problem is proposed to be solved using a deep reinforcement learning approach in this paper. One of the important prerequisites to successfully solving the problem is that a modeling method for the serial production line is adopted, such that we can correctly capture the dynamics of the system and ensure the good computation efficiency during learning process.

The numerical experiment proves that a good maintenance policy for the serial production line can be obtained by using the proposed deep reinforcement learning approach. In addition, we observe group maintenance and opportunistic maintenance in the learned policy. As two of the most important building blocks for the maintenance policies in the multi-unit systems, the group maintenance and opportunistic maintenance were originated from human reasoning. In this research, they are obtained by reinforcement learning without giving the agent any prior concepts and rules. Therefore, if the problems are properly formulated based on thorough understanding of the system properties, we can further exploit the great potentials of the artificial intelligence (AI) and machine learning techniques to facilitate complicated decision-makings in the manufacturing industry.

5.7. Related work

Part of the results presented in this chapter have been published in [83], [84], [118], [119].

Chapter 6. Manufacturing Process-System Integrated Control

6.1.Background

As manufacturing systems become increasingly interconnected to meet the accelerating demand of productivity, efficiency, and flexibility [1], challenges arise constantly for finding new methods supporting efficient operation of smart manufacturing system. There have been a lot of research efforts dedicated to improving each aspect of manufacturing systems, including throughput improvement [7], [8], [120], quality assurance [71], [72], [121]–[123], tool state monitoring [124]–[130] etc. However, these related research works are mostly separate. An integrated approach to combining these aspects for global operation optimization. which could have undoubtedly benefitted from advances in these subareas, is missing in both literature and real-world applications.

At the system level, traditional system-level modeling and analysis methods mostly only take aggregate parameters from process or machine as inputs, e.g., average cycle times and machine reliability distributions etc., and therefore tend to ignore the delicate dynamics at the machine and process level. For example, one of the prominent research problems in this area is the long-term steady-state system performance evaluation for production systems [131]. Related research assumes known and fixed cycle times and buffer capacities, as well as machine reliability distributions, which typically could be Bernoulli, Geometric, or Exponential distributions. Based on these aggregate parameters, the goal is often to calculate performance metrics in steady state, such as throughput and work-in-process, through approximation methods like aggregation and decomposition. However,

variations at process level and machine level, which could easily lead to changes in cycle time and machine status, are not considered in these works. For instance, in machining processes, sometimes it is not uncommon to adjust process control parameters such as spindle speed when appropriate. These adjustments could result in frequent shifts from average cycle time and violate the major assumption in existing methods. Another category of system-level analysis methods pursues production loss diagnosis in real time through sensor data [29]. This line of research is more aligned with the control purpose as of this paper. However, process-level quality performance is not entertained in these methods either, as both throughput and production loss are metrics in overall production quantities without distinguishing compliant products from defective ones. Therefore, a more comprehensive modeling method that could seamlessly integrate various aspects, including system-, process-, and machine-level dynamics, are much needed to facilitate integrated control of the manufacturing systems. In this paper, we will model a manufacturing system with graph model by treating each machine as node and material flow as links. With the graph model, we can incorporate all relevant information from system/process/machine as well as the interactions among machines in the dynamic node feature.

At the machine level, tool state is one of the most important as well as most dynamic facets of machine conditions. For example, in machining systems, machines utilize cutting or grinding tools to remove excessive materials in order to complete the desired process. However, machine tool state is subject to deterioration due to continuous usage, which could affect both process-level and system-level dynamics. On the one hand, process-level quality performance is heavily dependent on machine tool state, e.g., a worn-out cutting tool tends to be extremely inefficient in removing materials from the part. The real-time tool state is a requisite in order to determine the process control parameters that best fit the tool state and guarantee desired process quality performance. For example, one could strategically increase the cutting time to compensate for the tool inefficiency if tool wears out moderately. On the other hand, treatment or replace operations that aim to restore the tool condition are warranted if the tool state deteriorates to a

certain level. The extra time needed for tool treatment (e.g., dressing in grinding process) or replacement effectively introduces downtime events to the system, which is one of the dominant factors influencing the system-level dynamics [132].

The major challenge in machine tool state monitoring/identification lies in that tool state is not directly observable due to physical constraints. Usually, sensor technologies and machine learning models are adopted to monitor real-time tool state in an indirect manner. For grinding machines, researchers use acoustic emission sensors to collect acoustic data from exterior of grinding machine, and then apply different machine learning models, e.g., clustering, neural networks, and support vector machine etc., to classify the underlying status of grinding wheel [124], [133]. Lenz et al. [127] propose a holistic data analytics framework to collect and process data to gain insights into various aspects of machine conditions, including tool wear status, machine energy consumption, and process quality etc. Recently, it is reported that state-of-the-art deep learning architectures, including convolutional neural networks and recurrent neural networks, have been intensively used to work on raw sensor data to identify underlying machine health conditions [134].

Despite these exciting advances in sensor and machine learning based tool state inference, there are two obstacles that we have to overcome to take full advantage of these advances to facilitate the integrated control. First, the tool state inference results are not directly actionable at system level. Machines have complicated interactions among each other in a manufacturing system. Unfavorable tool state in one machine does not always warrant an immediate stoppage and replacement action, because its stoppage could potentially propagate to its downstream or upstream machines due to starvation and blockage. Decision making at system level requires careful coordination among machines. A similar problem is the maintenance decision making in serial production lines. Decision frameworks based on (multi-agent) reinforcement learning have been proved to be effective to tackle ultra-complexity in such problems [118], [119]. Second, the tool state inference results are inherently unreliable due to uncertainties in sensor readings

and/or irreducible errors in machine learning models. Therefore, it is unadvisable to directly use the tool state inference results in downstream control framework. Therefore, in this paper we will use Recursive Bayesian Estimation to preprocess the tool state inference result to reduce its uncertainty level, in order to facilitate a more informed decision making.

In summary, processes, machines, and system are entangled with each other in a manufacturing system to jointly impact the overall system performance. As research progressing for each individual level, opportunity arises for a systematic approach to integrating these different levels and achieving optimal system performance regarding not only production quantity but also product quality. Undoubtedly, it requires adoption of an array of innovative methods and algorithms to cope with the intricate nature of the problem. With the increasing data availability and computing resources, data-driven machine learning methods are particularly promising in solving such problem.

6.2.Problem description

In this paper, we consider a typical multi-stage manufacturing system consisting of multiple machines and buffers. A part is completed after being processed sequentially through all the stages. Figure 6.1 shows a simplified camshaft production line with four stages.

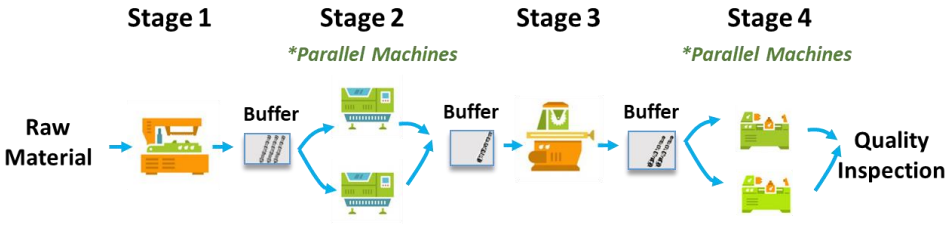


Figure 6.1. A portion of the grinding line for camshaft manufacturing system

Corresponding to the production line, each of the stage or machine has a underlying process model as shown in Figure 6.2. A product can be described by several key features, which would evolve through the processing at each stage. The

product feature at stage j, $j \in [1, \dots, m]$, is denoted as q_j. In essence, the process model describes the relationship between two consecutive stages and its dependency on process control parameters and tool state [135], [136]. In order to take other random factors into consideration, we may represent the *process model* for machine i at stage j in a probabilistic form:

$$p(q_j|q_{j-1}, u_i, x_{i,k}^t) \tag{6.1}$$

where $u_{i,j}$ the process control input for machine i at stage j, and $x_{i,k}^t$ is the tool state for machine i at k^{th} pass. The feature of the final product is denoted as q_M, where M is the numbering of final stage. In order to determine the quality of the product, the final product feature will be compared against a quality standard Q^*, i.e., product is compliant if $q_M \in Q^*$, and defective otherwise.

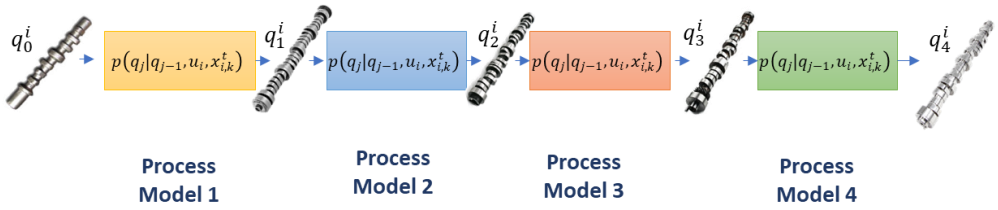


Figure 6.2. The underlying process models for the production line

In current industrial practice, the parameter control u_i is often set to be fixed or determined only to optimize local process performance. In this paper, however, we aim at adaptively adjusting the process control parameter u_i considering not only local conditions but also the status at system level, as well as real-time tool state.

6.3.Problem formulation

To formulate the research problem, two quantities that relate to the performance of both processes and system are denoted in this work:

- **Yield** y: number of qualified products among total output.
- **Defect** d: number of defective products among total output.

Accordingly, the problem studied in this paper is then presented as follows:

Given the manufacturing system as described, establish an integrated process-system modelling approach, and build an automated control scheme to find optimal adaptive policies for each machine to adjust process parameters for each product with the aim of maximizing the system yield, i.e.,

$$u_{i,j}(t) = \arg\max_{u_{i,j}(t)} \{y(T)\}, \forall t \in [0, T] \tag{6.2}$$

where $y(T) = f\left(\left\{q_m, u_{i,j}, x_{i,j}^t \middle| m \in \boldsymbol{M}, n \in \boldsymbol{N}\right\}\right)$ is the total system yield during a given time horizon T. All the constraints at different levels are denoted by $\boldsymbol{C}$.

With the manufacturing system modelled by a graph model, the node feature should embed all relevant real-time information across all levels. Particularly, in order to address the uncertainties in tool state inference, we will not directly plug tool state reference results, e.g., sensor reading or machine learning output, into the node feature. Instead, Recursive Bayesian Estimation (RBE) will be applied to construct a tool state belief based on domain knowledge on tool deterioration and sensor/model uncertainty. Graph Neural Network (GNN), specifically Graph Attention Network (GAT), will be applied to process the node features to learn meaningful node embedding that incorporates both local and global information through GNN operations. For the integrated control purpose, each node will then be treated as an agent in the Multi-Agent Reinforcement Learning (MARL) framework. State-of-the-art GNN and MARL algorithms will be implemented to train learnable parameters in GNN-MARL networks to learn the optimal multi-agent policy for adaptively adjusting process parameters to optimize overall performance.

6.4.Manufacturing system graph and GNN

In this subsection, we will model the manufacturing system with graph model. Firstly, general manufacturing systems often have non-Euclidian structures, for example, due to flexible routings with parallel machines. Graph model has ultra-high flexibility since dependencies among any two machines can be represented as links. Secondly, graph model allows us to incorporate heterogenous information

from multiple levels by simply putting these information in the node feature. Last but not the least, GNN as a special type of neural network directly operating on a graph provide an efficient way to further processing these node features to comprehensively integrate information across machines for downstream control task.

6.4.1. Graph model and node feature definition

The manufacturing system graph is defined by treating each machine as node and material flows between machines as links. In the integrated control problem, the node feature is defined as:

$$x_n = [bl_n, bv_n, x_k^t, w_n, \alpha_n, r_n, u_n, n, m] \tag{6.3}$$

The node feature contains information across different levels of the manufacturing system:

- bl_n and bv_n denote the machine's immediate upstream buffer level and downstream buffer vacancy respectively.
- x_k^t is the tool state.
- w_n and α_n describe the machine operating status, where w_n is a binary variable indicating whether machine is stopped by random interruptions or tool changes, and α_n denotes the completion ratio of current product.
- r_n is the feature of the product currently being processed by the machine.
- u_n is the process control parameters previously applied on current machine. This past action is included to form an observation-action trajectory to help local agent reason the true system state.
- n and m are machine number and stage number respectively. Since the proposed control framework is based on deep MARL, n and m are included to uniquely identify the machine to facilitate the parameter sharing in neural networks.

The node feature only represents the local status of the machine. In Section 4.2, these node features will be further processed through GNN to generate node embeddings that incorporate both local and global status.

6.4.2. Recursive Bayesian Estimation for tool state belief

Apparently, it is not advisable to directly use realized observation $\tilde{z}_k$ for control or decision making, especially when the uncertainty in observation is ineligible. Recursive Bayesian Estimation (RBE) is an effective method to reduce the observation uncertainty using the domain knowledge represented by the transition model and observation model [137]–[139]. Give the tool state *belief* at $(k-1)^{th}$ pass, one step of *prediction* is conducted to update the belief using transition model:

$$p(x_k^t|\tilde{z}_{1:k-1}) = \sum_{x_{k-1}=1}^{\mathcal{T}} p(x_k^t|x_{k-1})p(x_{k-1}^t|\tilde{z}_{1:k-1}) \tag{6.4}$$

Note that $p(x_k|\tilde{z}_{1:k-1})$ is updated purely based on the transition model. We need to correct this tool state belief using what we observed as tool status at k^{th} pass, denoted as $\tilde{z}_k$, and our knowledge on observation uncertainty. The one-step *correction* is hence conducted to obtain the tool state belief at k^{th} pass:

$$p(x_k^t|\tilde{z}_{1:k}) = \frac{l(x_k^t|\tilde{z}_k)p(x_k^t|\tilde{z}_{1:k-1})}{\sum_{x_k=1}^{\mathcal{T}} l(x_k^t|\tilde{z}_k)p(x_k^t|\tilde{z}_{1:k-1})} \tag{6.5}$$

where $l(x_k^t|\tilde{z}_k)$ is the likelihood function corresponding to the observation model $p(z_k|x_k^t)$. In the integrated control proposed in this paper, we will use tool belief $p(x_k|\tilde{z}_{1:k})$ instead of the observation $\tilde{z}_k$ to represent the tool status. Note that transition model and observation model do not need to be perfect in order to carry out RBE. Rather, RBE would fully exploit domain knowledge to infer underlying tool state from imperfect models and observations.

6.4.3. Graph neural network for node embedding

In a multi-agent control framework, it is important to have a communication mechanism among machines, so that each machine is aware of not only its own status but also others' conditions in the neighborhood. In this way, machines are

more likely to make coordinated and informed decisions, which is critical to achieving the common goal, i.e., maximization of system yield.

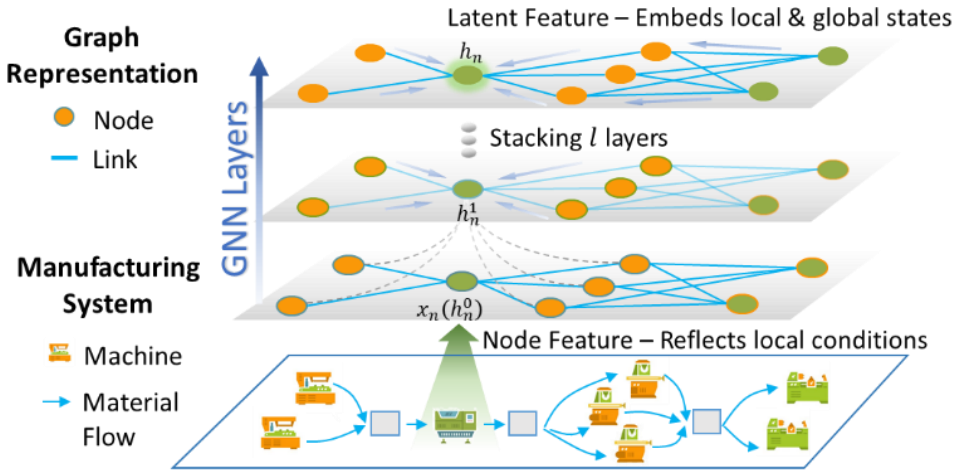


Figure 6.3. Graph neural network on manufacturing system graph

In this paper, we will use graph attention network (GAT) [140], [141] to process the node features x_n to generate the node embeddings h_n^L. By setting $h_n^0 = x_n$, the embedding of node n can be obtained by the following Aggregate & Combine rule:

$$h_n^{l+1} = \sigma\left(\sum\nolimits_{j\in(m\cup n)} \alpha_{nj}^l h_j^l W_V^l\right) \tag{6.6}$$

where $\sigma(*)$ is an activation function, m denotes all immediate neighboring nodes to node n, W_V^l is learnable weight matrix applied to embedding h_j^l on previous layer, and α_{nj}^l is the attention weight assigned to node j. The attention weight is a dynamic coefficient that determines the amount of information pulled from node j. It is calculated using a multi-head dot-product attention (MHDPA) with learnable query matrix W_Q^l and key matrix W_K^l [142], [143]. The attention weight α_{nm}^l can be written as:

$$\alpha_{nm}^l = \frac{\exp\left(\frac{h_n^l W_Q^l \left(h_m^l W_K^l\right)^T}{\sqrt{d_k}}\right)}{\sum_{k\in(m\cup n)} \exp\left(\frac{h_n^l W_Q^l \left(h_k^l W_K^l\right)^T}{\sqrt{d_k}}\right)} \tag{6.7}$$

where d_k is the dimension of the latent feature at layer l. By stacking L GAT layers, one can obtain the node embedding of the node. Essentially, it is a process of aggregating and refining information from neighbors further and further away with increased layers. Compared to the initial node feature, the final node embedding combines both local and global information. Learnable parameters W_V^l, W_Q^l and W_K^l are initialized to random numbers and trained with downstream tasks.

6.5.MARL control problem formulation

The reasons for adopting a multi-agent scheme in the integrated control are two-fold. Firstly, individual process control often has a moderate or even large action space. For example, if one machining process has two discretized control parameters with 10 options each, then the action space for individual machine sums up to 100. If this problem is formulated in single-agent RL framework, the total action space for the system increases exponentially with number of machines in the system, which could be computationally intractable and lead to divergence of the RL training process. Second, the manufacturing system is a typical multi-component system and a global objective, i.e., system yield, is available. It is natural to define individual agent and a common play game, which could fit into the MARL framework well. In this work, we will formulate the MARL-based integrated control problem using the Dec-POMDP framework. In this subsection, we will define the definitions for observation o^n, action u^n, and reward function r_t in the Dec-POMDP.

- **Observation definition**

Since node embedding incorporates both machine's local operating status and global status at the system level, it is a far better choice for the agent observation than the local node feature. Therefore, the observation o^n for agent n is defined to be

$$o^n = h_n^L \tag{6.8}$$

where L is the number of layers in GNN, and h_n^L is the final node embedding output from the L-layer GNN. For training purpose, the global state should also be defined. In MARL, the concatenation of agent observations can serve as the global state. Since our observation definition is different from traditional MARL problem, it is sufficient to concatenate the node features rather than the node embeddings. The global state can be represented as:

$$s = [x_1, \dots, x_N] \tag{6.9}$$

- **Action definition**

In the integrated control problem, for each machine represented by agent, the control action is to set appropriate process control parameters. The controllable parameters depend on specific manufacturing process and even the machine type/series. To make the discussion more concrete, the grinding process will be used as an example. In grinding process, the controllable parameters often include the depth of cut c and grinding speed v. Therefore, the action space for a grinding machine is

$$U^n = \{[c, v]\} \tag{6.10}$$

- **Decision time**

The decision time means the time point when the observations need to be collected and a control decision needs to be made. The continuous time is discretized by those decision times. In the integrated control problem, the decision time is when any of the machines loads a part from upstream buffer and prepares to process it. At this time point, the machine is awaiting a control decision, i.e., as set of process control parameters, from the control framework. Due to the heterogeneity in machine operations, not all of the machines are up for a control decision at one decision time point. Therefore, we add a dummy action for those machines that do not require control actions at that moment, during which other actions will be masked out.

- **Reward definition**

The reward setting is nontrivial in RL/MARL applications in manufacturing systems [117], [144]. In MARL, reward function is used to evaluate the goodness of the joint action given the system state. The reward drives the agents to continuously improve their policies accordingly to pursue better system performance. In this paper, we define the reward function as:

$$r_t = y_t - d_t \tag{6.11}$$

where y_t is the stepwise system yield, and d_t is the stepwise defect. The reward function is aligned with the problem objective, i.e., maximization of system yield, with slight difference by adding negative rewards for defective products. Therefore, it would not only reward the agents when compliant products are produced, but also penalize them when defective ones emerge. We will demonstrate that faster convergence is guaranteed by adding the defect penalty in the experiments.

6.6.Experiments and validation

The manufacturing system considered in this case study has four grinding stages. Three intermediate buffers are placed between stages and each of them has a capacity of ten. There are six grinding machines in total. Both Stage 2 and Stage 4 have two parallel machines. Therefore, the system forms a graph consisting of six nodes, and machine nodes in consecutive stages are connected through links. At the machine level, the grinding wheel has four states, including sharp, intermediate, dull, and worn. The tool state transition model $p(x_k^t|x_{k-1}^t)$ and observation model $p(z_k^t|x_k^t)$ for the grinding wheel state are:

$$p(x_k^t|x_{k-1}^t) = \begin{bmatrix} 0.60 & 0.27 & 0.08 & 0.05 \\ 0 & 0.65 & 0.26 & 0.09 \\ 0 & 0 & 0.70 & 0.30 \\ 0 & 0 & 0 & 1.00 \end{bmatrix}$$

$$p(z_k^t|x_k^t) = \begin{bmatrix} 0.70 & 0.17 & 0.12 & 0.01 \\ 0.10 & 0.70 & 0.16 & 0.04 \\ 0.25 & 0.10 & 0.60 & 0.05 \\ 0.22 & 0.18 & 0.05 & 0.55 \end{bmatrix}$$

The tool state transition model strongly suggests that the tool state, if not staying the same, would only deteriorate towards worse. The observation model reveals that the observation is not perfect. The grinding wheel restores its condition as sharp when a dressing or replacement operation is conducted every 10 passes. We assume that such an operation takes 3 units of time, during which the machine is unavailable. Hence, the dressing operation effectively introduces downtimes to the system. The system is assumed to run on a ten-hour shift, i.e., the simulation episode time horizon is set to be 600 minutes.

At the process level, we adopted a simplified grinding process model [145] for demonstration purpose. The key feature of the product is characterized by the surface roughness of the four sequential grinding processes, i.e., $q = [q_1, q_2, q_3, q_4]$. The final product is deemed qualified if $\sum_{m=1}^{4} q_m \leq 5$ μm, or defective otherwise. We adopt a simplified grinding process model [13] to relate the surface roughness and cycle time to process parameters, machining speed v_n and depth of cut a_n. Therefore, the MARL agent's action space is $U^n = \{[c_n, v_n]\}$, where $c_n \in$ {12.0, 13.5, 15.0, 16.5, 18.0} μm and $v_n \in$ {0.30, 0.35, 0.40, 0.45, 0.50} m/s. Table 6.1 shows the process model and cycle time model for each stage given the tool state is sharp.

Table 6.1. Process models for the grinding system

Stage	Process Model	Cycle time
1	$q_1 \sim N\left(\left(\frac{v_1 c_1}{30}\right)^{0.9}, 5.48 v_1^2 c_1^{1.4}\right)$	$T_1 = \frac{0.005}{v_1 c_1}$
2	$q_2 \sim N\left(\left(\frac{v_2 c_2}{30}\right)^{0.85}, 5.48 v_2^2 c_2^{1.35}\right)$	$T_2 = \frac{0.010}{v_2 c_2}$
3	$q_3 \sim N\left(\left(\frac{v_3 c_3}{30}\right)^{0.9}, 5.48 v_3^2 c_3^{1.5}\right)$	$T_3 = \frac{0.007}{v_3 c_3}$
4	$q_4 \sim N\left(\left(\frac{v_4 c_4}{30}\right)^{0.85}, 5.48 v_4^2 c_4^{1.3}\right)$	$T_4 = \frac{0.010}{v_4 c_4}$

In general, as grinding wheel wears, the surface of the grinding wheel becomes finer, which could lead to lower surface roughness on product as well. However,

its cutting efficiency drops significantly, and therefore more time is required to remove the desired amount of material from the part to fulfil dimension standards. Using the process models in sharp tool state as baselines, the process models under other tool states are modelled with two series of coefficients. The coefficients for surface roughness models and cycle times are [1.0, 0.99, 0.98, 0.96] and [1.0, 1.2, 1.4, 2.1] respectively. For example, given dull tool state and same process control parameters, the surface roughness is 0.98 of that under sharp tool state on average, but it needs 40% more cycle time to finish the process at that stage. Therefore, rational process control decisions require best knowledge of the current tool state, given which the trade-off between surface roughness and production efficiency needs to be well balanced to achieve high system yield. Since process modeling is not the focus of this paper, we adopt a simplified process model in this case study. Due to the model-free nature of the MARL algorithm and hence the proposed integrated control framework, one can fit in more accurate and specific process models as needed.

6.6.1. GNN-MARL architecture and training process

The GNN and actor network architectures are as shown in Figure 6.4. A skip connection is used to connect node feature to node embedding in order to preserve more information from the local node. The critic network architecture is as shown in Figure 6.5. The algorithm alternates between sampling experience from parallel simulation episodes and optimizing network parameters. The training is conducted with a step limit of 10 million.

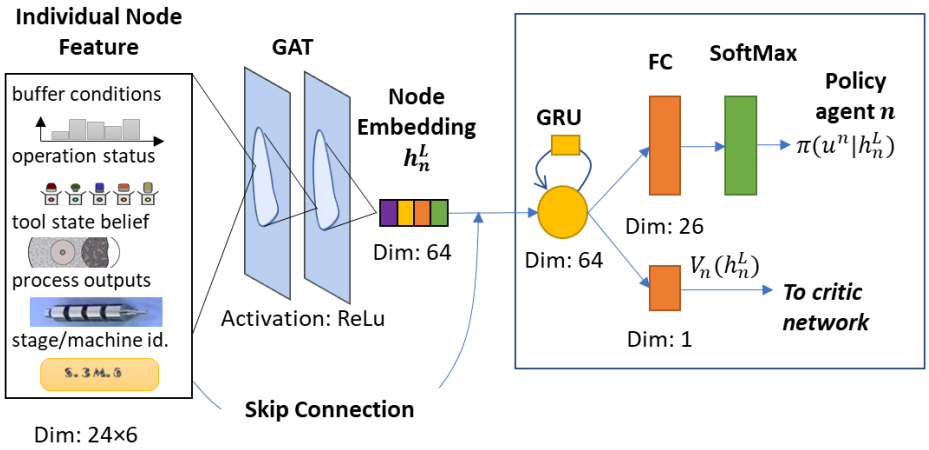


Figure 6.4. GNN and actor network architecture

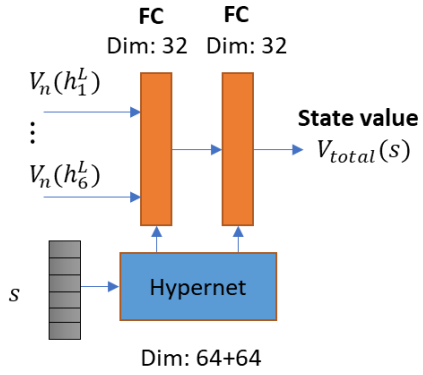


Figure 6.5. Critic network architecture

The training process is halted every 10,000 steps and test episodes are rolled out based on the policies at that moment to observe training progress. From Figure 6.6 we can see that the test return, i.e., accumulated reward, improves significantly since the training begins. After around three to four million steps, the test return stays stable, which strongly indicates convergence.

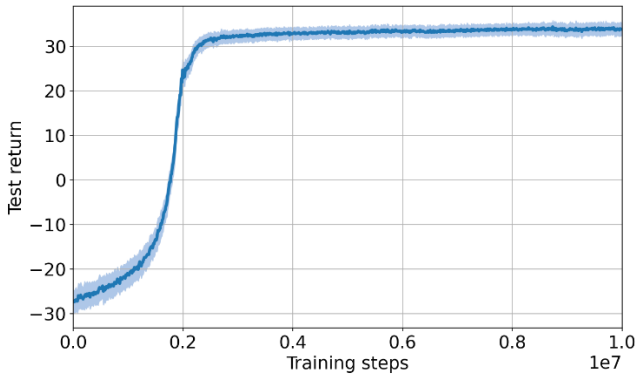


Figure 6.6. Training progress shows convergence

Since return is more of an algorithm-wise term, we also monitor the training progress in the control problem context. The outputs, yields, and defects in those test episodes are also recorded as shown in Figure 6.7. In the beginning, the output is high but most of them are defective parts. Over time, the overall production speed is slowed, and system yield is improved, which means agents learn to coordinate with each other to improve the ratio of compliant products. When the training converges, a high system yield is achieved, while the defect is kept at a very low level. Therefore, the training process of GNN-MARL successfully converges, and produces desired distributed control policies for machines.

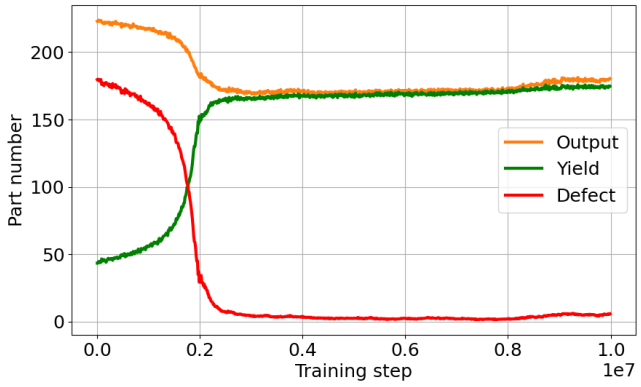
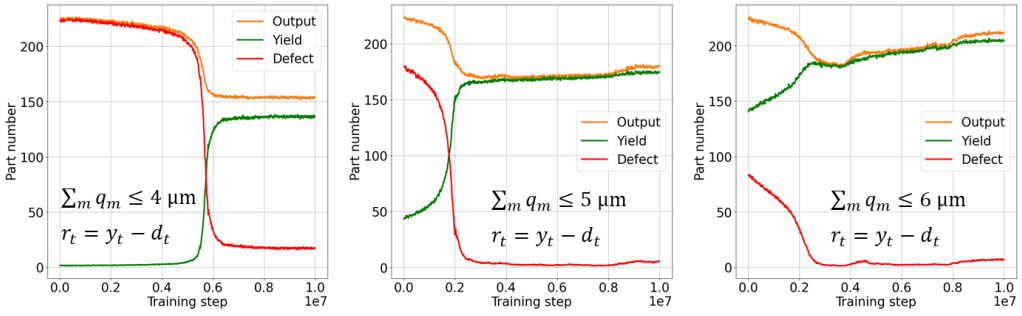


Figure 6.7. System performance improvement throughout training process

The above shows the training process given the quality standard $\sum_m q_m \leq 5$ µm. In order to further validate the effectiveness of the proposed framework, additional experiments are conducted with different quality standards. First, we tighten the quality standard to 4 µm and rerun the training with same

hyperparameters. As shown in Figure 6.8(a), the convergence is reached although taking more steps, which is reasonable since the task is more difficult than the original one as shown in Figure 6.8(b). Similarly, we loosen the quality standard to 6 μm and also reach convergence as shown in Figure 6.8(c). These additional experiments further demonstrate that the convergence and performance of the proposed framework.



(a) (b) (c)

Figure 6.8. Convergences under different quality standards

6.6.2. Numerical analysis on reward function setting

In this paper, we propose the reward function as $r_t = y_t - d_t$, which includes a defect penalty. There could be a more intuitive alternative: $r_t = y_t$, which is more aligned with the problem objective of maximizing yield. In this subsection, we conducted extra experiments with the alternative reward function, and compare its performance with our proposal. Using the alternative reward function, we rerun the training process for three different quality standards. The training process for 4 μm fail to converge as shown in Figure 6.9(a). The training process for 5 μm and 6 μm do converge as shown in Figure 6.9(b) and Figure 6.9(c) respectively, but the performance is worse than the original reward setting as shown in Figure 6.8(b) and Figure 6.8(c).

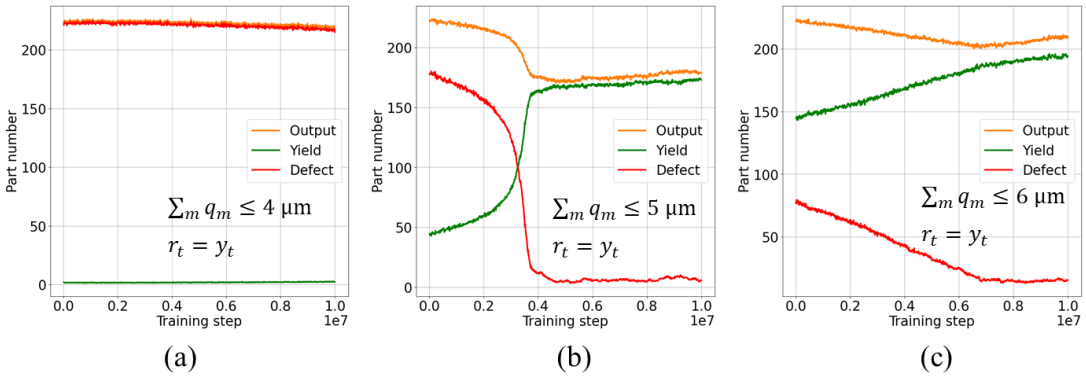


Figure 6.9. Training results without defect penalty in reward function

The results prove that inclusion of defect penalty is necessary. We can take a closer look into the return improvement as shown in Figure 6.10 for 4 µm quality standard. There is barely any improvement throughout the ten million steps.

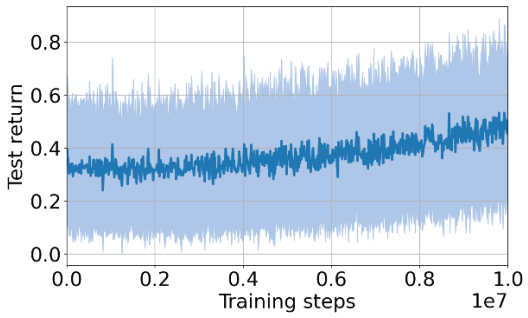
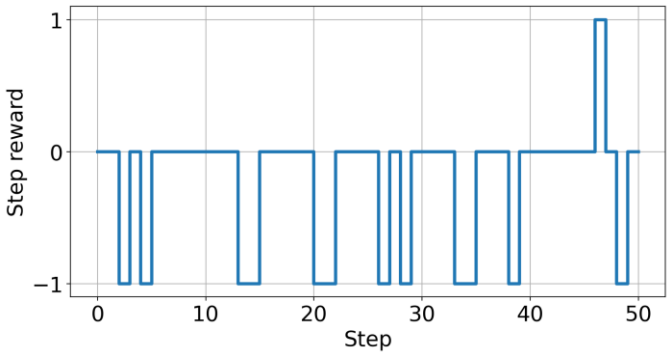
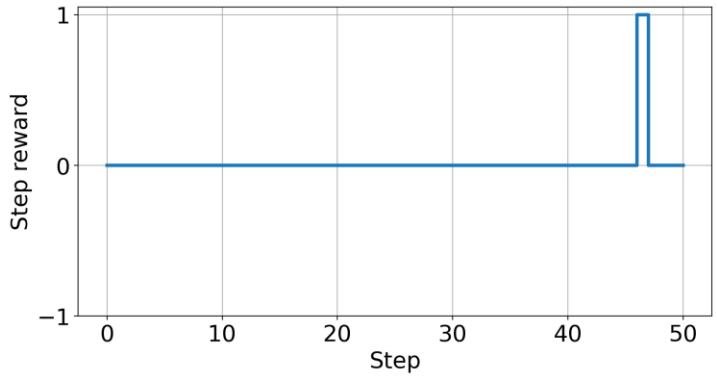


Figure 6.10. Stalled training progress without defect penalty

In the same case, we further analyze the reward signals received by the agents given different reward functions.



(a) Step rewards during training with defect penalty in reward function



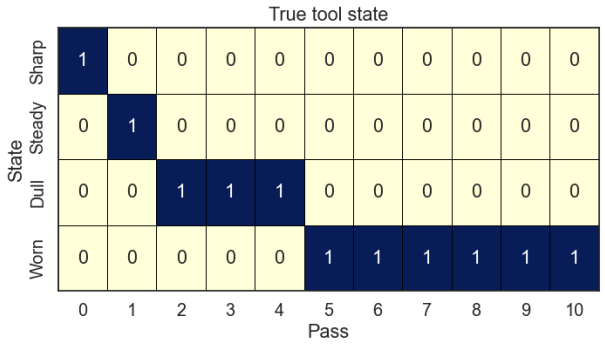
(b) Step rewards during training without defect penalty in reward function

Figure 6.11. Step rewards during training given different reward settings

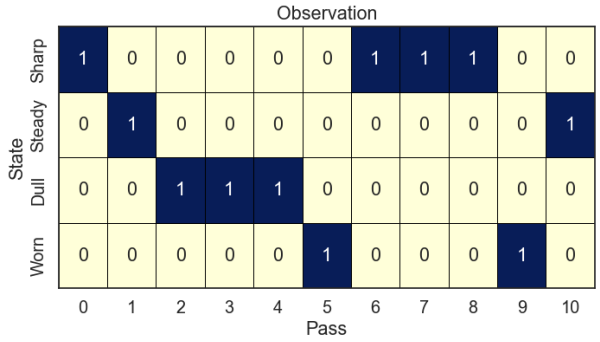
As shown in Figure 6.11, agents receive very sparse reward in the reward setting without defect penalty. In that case, agents have little feedback on how good/bad their actions are. It impedes their improvements and leaves the agents exploring blindly in random directions. Therefore, the learning process is too slow to achieve convergence. That explains the learning curve in Figure 6.10, which does not show divergence but very slow improvement. Therefore, the analysis on reward setting stresses the importance of proper reward setting in RL applications within the manufacturing industry.

6.6.3. Demonstration of RBE for tool state estimation

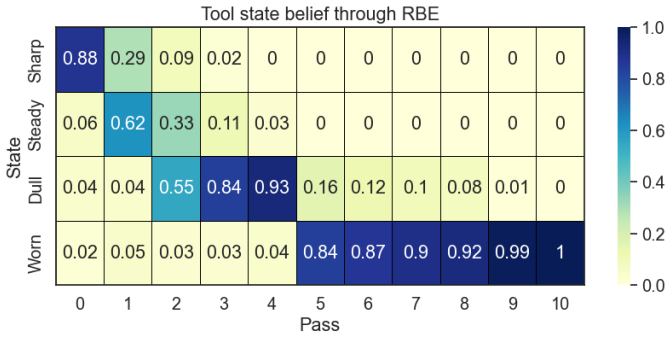
In this subsection, the results from tool state belief are presented. As shown in Figure 6.12, in ten passes, the belief updated by RBE closely follows the true tool state, which is hidden. Although the observations make errors in some of the steps, the beliefs are able to correct that error by sticking to the true state with high confidence. For example, in step 10 the belief rejects the wrong observation. Ther reason is that the domain knowledge in state transition suggests that the tool is only getting worse over time. It is not likely that the tool has a steady state after so several observations suggesting severe deterioration. Therefore, the RBE for node feature engineering is an effective method to link tool state inference to its practical applications in control problems.



(a) True tool state evolves in 10 grinding passes



(b) Tool state as indirectly observed with uncertainties



(c) Tool state beliefs obtained by RBE

Figure 6.12. Illustration of RBE applied to tool state inference

Different from the observation, the belief represents the tool state in a probabilistic form by assigning a probability to each of the possible tool states. The belief follows the true tool state very well. RBE is able to correct the wrong observations. It even totally rejects the wrong observation at pass 10. This could be

because the domain knowledge on tool state transition strongly suggests that tool state would not become better unless a dressing/replacement takes place. The belief over the first nine passes already indicates a worst tool state. Therefore, at pass 10 RBE directly rejects the wrong observation of a better tool state. In conclusion, the proposed tool state estimation based on RBE in this paper is effective.

Comparing the framework with and without RBE, it is found that RBE does improve the system performance significantly as shown in Figure 6.13. The method with RBE has higher a system yield.

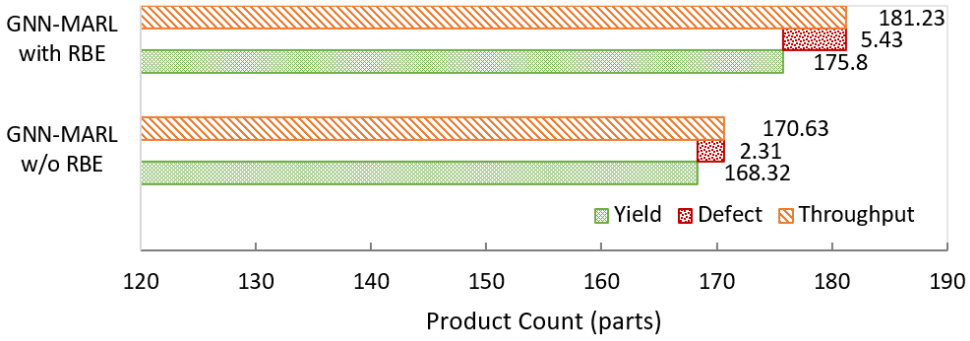


Figure 6.13. Comparison of system performance with and without RBE

6.7.Summary

An integrated method for manufacturing processes and system modelling and distributed adaptive control are developed for multi-stage manufacturing system, based on GNN and MARL. GNN encodes complex system dynamics in machine nodes by aggregating real-time information from the neighboring machines. MARL models each machine as individual agent and learns adaptive process parameter control policy to cooperatively achieve the goal of global maximization of system yield. It is numerically proved that the use of Recursive Bayesian Network improves the system performance significantly as it reduces the error in tool state observations. In addition, it also demonstrates that the reward function setting is a critical issue in control problems in the manufacturing systems.

6.8.Related work

Part of the results shown in this chapter have been in [146].

Chapter 7. Concluding Remarks

7.1.Summary of scientific contributions

Entering the stage of smart manufacturing, data regarding the real-time system, process, and machine operation status are increasingly available. However, it is a serious research problem to find ways to better use those data. Machine learning and reinforcement learning have demonstrated success in a lot of areas outside of manufacturing industry. However, it requires thorough understanding and domain knowledge of the manufacturing to formulate problems and identify meaningful solution approaches. In this dissertation, system modeling and system properties discussion have laid the foundation for further applications of machine learning and reinforcement learning methods to important performance prediction and control problems. The contributions of this dissertation are listed as follows:

(1) Establish a system modeling for multi-product system considering product-dependent cycle time and tool setup time. The model is analytically derived from basic physics, i.e., conservation of the flow. Using the past sensor data, one can efficiently evaluate the system states in a recursive manner based on the state-space formulation of the system model.

(2) Derive useful properties from the system modeling, guided by which a recurrent prediction problem is formulated for the product completion time prediction. We refrain from taking a pure data-driven approach to the problem. Instead, the deep learning architecture (LSTM) is combined with the system model to deliver more accurate prediction result, which is critical to downstream tasks, including production scheduling, and customer satisfaction.

(3) Formulate the preventive maintenance control as a sequential decision-making problem and pave the way of the application of deep reinforcement learning technique. It demonstrates a systematic way to formulate control problems in the manufacturing to fit in the deep reinforcement learning domain. The reward setting is guided by system understanding, namely, the evaluation of permanent production loss.

(4) Demonstrate the derivation process to choose appropriate reinforcement learning algorithms for different application scenarios: single-agent DDQN algorithm for moderate problem scale, and multi-agent VDAC algorithm for large-scale systems considering imperfect maintenance effects.

(5) Prove the effectiveness of the RL-based preventive maintenance policies by observing agent's behavior. We discover that RL agents occasionally conduct patterns that are originally derived from human reasoning. It reveals that state-of-the-art machine learning algorithms are particularly promising to solve complex problems in manufacturing industries, given correct problem formulation and appropriate method selection.

(6) Establish the process-system multi-level control framework. The system is modeled using a graph by treating each machine as node and material flow as link. Graph modeling along with powerful GNN allows integration of real-time information from multiple levels. It bridges the gap between process, machine, and system analysis, which are typically separate in existing research. It paves the way for smarter manufacturing and could significantly improve the system performance.

7.2.Remarks on knowledge-guided machine learning

In recent years, machine learning, including deep learning and reinforcement learning, have demonstrated great successes in several well-defined problems, including image classification, natural language processing, and recommendation systems etc. However, there are many more areas that could benefit from machine

learning, including manufacturing systems discussed in this dissertation. Apart from the technical details presented throughout this dissertation, there are some further remarks on knowledge-guided machine learning:

(1) It is important to understand the source of randomness in a problem. For example, for manufacturing systems, the major source of randomness is the machine downtimes, which are dependent on machine reliability, while the randomness in tool state originates from the material-wise wear. However, the system states are evolving on a real-time and stochastic basis due to a combination of both deterministic mechanisms and random factors. In manufacturing systems, the changes in buffer states are deterministic if those random factors are realized. It should be clear throughout the problem formulation process.

(2) Problem formulation is more important than specific algorithms. There are new machine learning algorithms emerging on a regular basis [147], [148]. It is unwise to tie the problem formulation to any specific algorithms, since one algorithm can be outperformed by another in years if not months. Therefore, the general framework should be employed. For example, MDP and Dec-POMDP are two frameworks for formulating RL and MARL respectively. Once the control problems are formulated in these frameworks, with preset interface, the problem can be solved by any state-of-the-art RL/MARL algorithms.

(3) Exploration is an important technique. In RL, exploration is the key to improve RL policy. It is also true for knowledge-guided machine learning. Since there is a lack of rigorous derivations of optimal hyperparameters in machine learning, one should explore more possible architectures and settings in machine learning to improve the system performance. It is also important to test the proposed solution against more diverse system scales and characteristics. For example, in this dissertation the MARL based PM policy is proposed since it was found that RL agent diverges when machine number increases. The reward setting in process-system integrated control is also validated by changing the quality standards.

7.3.Future work

As manufacturing systems varies from one to another, there are still a lot of open questions regarding the applications of machine learning and reinforcement learning, which could be addressed in the future. This dissertation will name a few potential working directions:

(1) Human-AI collaboration [149]–[153]. AI and human have each other's complementary strengths. AI has the speed, scalability, and accuracy in dealing with huge amount of data, while it lacks creativity and adaptiveness facing situations beyond quantitative aspects. Human is not known for its quantitative capability, but has the deductive reasoning, specialist expertise, and social skills that are not achievable by current AI systems. Therefore, it is promising to integrate human into the AI system for better system performance.

(2) Continuous learning for constantly changing system dynamics and environment. RL relies on accurate simulation environments to learn its policies. However, the simulated environment might well deviate from the real world due to, for example, inference error, hidden states etc. Another factor is that manufacturing system states transition model could change over time. Therefore, the continuous learning related techniques [154], [155] could be a potential tool to address these challenges.

(3) Multi-functional RL agent. Control problems in manufacturing systems are often dealt with separately. Even though the RL could be applied to different problems, it is common to formulate separate RL problems. However, these decisions from different RL agents could couple with each other, which could impair the performance regarding one or all of the control problem objectives. It could be beneficial, if possible, to train one multi-functional RL agent that are capable of making decisions for multiple control tasks.

List of Figures

List of Tables

Bibliography

[1] kristy.thompson@nist.gov, "Smart Manufacturing Operations Planning and Control Program," *NIST*, Apr. 25, 2014. https://www.nist.gov/programs-projects/smart-manufacturing-operations-planning-and-control-program (accessed Mar. 14, 2021).

[2] D.-H. Kim., "Smart Machining Process Using Machine Learning: A Review and Perspective on Machining Industry," *Int. J. Precis. Eng. Manuf.-Green Technol.*, vol. 5, no. 4, pp. 555–568, Aug. 2018, doi: 10.1007/s40684-018-0057-y.

[3] J. Wang, Y. Ma, L. Zhang, R. X. Gao, and D. Wu, "Deep learning for smart manufacturing: Methods and applications," *J. Manuf. Syst.*, vol. 48, pp. 144–156, 2018.

[4] D. Wu, C. Jennings, J. Terpenny, R. X. Gao, and S. Kumara, "A comparative study on machine learning algorithms for smart manufacturing: tool wear prediction using random forests," *J. Manuf. Sci. Eng.*, vol. 139, no. 7, 2017.

[5] G. Chryssolouris, *Manufacturing Systems: Theory and Practice*. Springer Science & Business Media, 2013.

[6] J. Li and S. M. Meerkov, *Production Systems Engineering*. Springer Science & Business Media, 2008. doi: 10.1007/978-0-387-75579-3.

[7] Y. Bai, J. Tu, M. Yang, L. Zhang, and P. Denno, "A new aggregation algorithm for performance metric calculation in serial production lines with exponential machines: design, accuracy and robustness," *Int. J. Prod. Res.*, vol. 0, no. 0, pp. 1–18, May 2020, doi: 10.1080/00207543.2020.1757777.

[8] J. Tu, Y. Bai, M. Yang, L. Zhang, and P. Denno, "Real-Time Bottleneck in Serial Production Lines With Bernoulli Machines: Theory and Case Study,"

IEEE Trans. Autom. Sci. Eng., vol. 18, no. 4, pp. 1822–1834, Oct. 2021, doi: 10.1109/TASE.2020.3021346.

[9] K. Kammerer, B. Hoppenstedt, R. Pryss, S. Stökler, J. Allgaier, and M. Reichert, "Anomaly Detections for Manufacturing Systems Based on Sensor Data—Insights into Two Challenging Real-World Production Settings," *Sensors*, vol. 19, no. 24, Art. no. 24, Jan. 2019, doi: 10.3390/s19245370.

[10] J. Yuqin, W. Peixia, and L. Yue, "Study of manufacturing system based on neural network multi-sensor data fusion and its application," in *IEEE International Conference on Robotics, Intelligent Systems and Signal Processing, 2003. Proceedings. 2003*, Oct. 2003, vol. 2, pp. 1022–1026 vol.2. doi: 10.1109/RISSP.2003.1285729.

[11] Y. Bao, L. Ren, L. Zhang, X. Zhang, and Y. Luo, "Massive sensor data management framework in Cloud manufacturing based on Hadoop," in *IEEE 10th International Conference on Industrial Informatics*, Jul. 2012, pp. 397–401. doi: 10.1109/INDIN.2012.6301192.

[12] J. Schmidt, M. R. G. Marques, S. Botti, and M. A. L. Marques, "Recent advances and applications of machine learning in solid-state materials science," *Npj Comput. Mater.*, vol. 5, no. 1, p. 83, Dec. 2019, doi: 10.1038/s41524-019-0221-0.

[13] J. Zhang, P. Wang, R. Yan, and R. X. Gao, "Long short-term memory for machine remaining life prediction," *J. Manuf. Syst.*, vol. 48, pp. 78–86, Jul. 2018, doi: 10.1016/j.jmsy.2018.05.011.

[14] J. Grezmak, J. Zhang, P. Wang, and R. X. Gao, "Multi-stream convolutional neural network-based fault diagnosis for variable frequency drives in sustainable manufacturing systems," *Procedia Manuf.*, vol. 43, pp. 511–518, 2020, doi: 10.1016/j.promfg.2020.02.181.

[15] J. Zhang, H. Liu, Q. Chang, L. Wang, and R. X. Gao, "Recurrent neural network for motion trajectory prediction in human-robot collaborative assembly," *CIRP Ann.*, vol. 69, no. 1, pp. 9–12, 2020, doi: 10.1016/j.cirp.2020.04.077.

[16] T. Yu, J. Huang, and Q. Chang, “Optimizing task scheduling in human-robot collaboration with deep multi-agent reinforcement learning,” *J. Manuf. Syst.*, vol. 60, pp. 487–499, 2021.

[17] J. F. Arinez, Q. Chang, R. X. Gao, C. Xu, and J. Zhang, “Artificial Intelligence in Advanced Manufacturing: Current Status and Future Outlook,” *J. Manuf. Sci. Eng.*, vol. 142, no. 11, p. 110804, Nov. 2020, doi: 10.1115/1.4047855.

[18] A. Dasci and M. Karakul, “Performance evaluation of a single-stage two-product manufacturing system operating under pull-type control,” *Comput. Oper. Res.*, vol. 35, no. 9, pp. 2861–2876, 2008, doi: 10.1016/j.cor.2006.12.024.

[19] J. K. Sagawa and M. S. Nagano, “Modeling the dynamics of a multi-product manufacturing system: A real case application,” *Eur. J. Oper. Res.*, vol. 244, no. 2, pp. 624–636, 2015, doi: 10.1016/j.ejor.2015.01.017.

[20] W. Feng, L. Zheng, and J. Li, “Scheduling policies in multi-product manufacturing systems with sequence-dependent setup times and finite buffers,” *Int. J. Prod. Res.*, vol. 50, no. 24, pp. 7479–7492, 2012, doi: 10.1080/00207543.2011.653455.

[21] S. J. Hu *et al.*, “Assembly system design and operations for product variety,” *CIRP Ann. - Manuf. Technol.*, vol. 60, no. 2, pp. 715–733, 2011, doi: 10.1016/j.cirp.2011.05.004.

[22] J. Zou, J. Arinez, Q. Chang, and Y. Lei, “Opportunity Window for Energy Saving and Maintenance in Stochastic Production Systems,” *J. Manuf. Sci. Eng.*, vol. 138, no. 12, p. 121009, Dec. 2016, doi: 10.1115/1.4033757.

[23] X. Ou, Q. Chang, J. Arinez, and J. Zou, “Gantry Work Cell Scheduling through Reinforcement Learning with Knowledge-guided Reward Setting,” *IEEE Access*, vol. 6, pp. 14699–14709, 2018, doi: 10.1109/ACCESS.2018.2800641.

[24] J. Huang, Q. Chang, and J. Arinez, “Modeling and Analysis of Multi-Product Manufacturing Systems With Two Machines and One Buffer,” presented at the ASME 2019 14th International Manufacturing Science and Engineering Conference, Nov. 2019. doi: 10.1115/MSEC2019-2956.

[25] J. Huang, Q. Chang, and J. Arinez, “Modeling and Dynamic Assignment of the Adaptive Buffer Spaces in Serial Production Lines,” *J. Manuf. Sci. Eng.*, vol. 143, no. 3, p. 031005, Mar. 2021, doi: 10.1115/1.4048377.

[26] J. Huang, Q. Chang, and J. Arinez, “Product Completion Time Prediction Using A Hybrid Approach Combining Deep Learning and System Model,” *J. Manuf. Syst.*, vol. 57, pp. 311–322, Oct. 2020, doi: 10.1016/j.jmsy.2020.10.006.

[27] C. Li, J. Huang, and Q. Chang, “Data-Enabled Permanent Production Loss Analysis for Serial Production Systems with Variable Cycle Time Machines,” *IEEE Robot. Autom. Lett.*, vol. 6, no. 4, pp. 6418–6425, 2021.

[28] C. Li, J. Huang, and Q. Chang, “Data-Enabled Real-Time Modeling for Production Systems with Variable Cycle Time,” *Procedia Manuf.*, vol. 53, pp. 561–567, 2021.

[29] J. Zou, Q. Chang, J. Huang, J. Arinez, and G. Xiao, “Analysis of end-of-state impact on manufacturing system production performance,” in *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*, 2018, pp. 823–828.

[30] S. R. Hejazi * and S. Saghafian, “Flowshop-scheduling problems with makespan criterion: a review,” *Int. J. Prod. Res.*, vol. 43, no. 14, pp. 2895–2929, Jul. 2005, doi: 10.1080/0020754050056417.

[31] M. E. Aydin and E. Öztemel, “Dynamic job-shop scheduling using reinforcement learning agents,” *Robot. Auton. Syst.*, vol. 33, no. 2–3, pp. 169–178, Nov. 2000, doi: 10.1016/S0921-8890(00)00087-7.

[32] F. Winter, E. Demirovic, N. Musliu, and C. Mrkvicka, “Solution Approaches for an Automotive Paint Shop Scheduling Problem,” in *Proceedings of the Twenty-Ninth International Conference on Automated Planning and Scheduling*, 2021, vol. 29(1), pp. 573–581.

[33] D. Wu, M. J. Greer, D. W. Rosen, and D. Schaefer, “Cloud manufacturing: Strategic vision and state-of-the-art,” *J. Manuf. Syst.*, vol. 32, no. 4, pp. 564–579, Oct. 2013, doi: 10.1016/j.jmsy.2013.04.008.

[34] M. Hasan and B. Starly, “Decentralized cloud manufacturing-as-a-service (CMaaS) platform architecture with configurable digital assets,” *J. Manuf. Syst.*, vol. 56, pp. 157–174, Jul. 2020, doi: 10.1016/j.jmsy.2020.05.017.

[35] K. Altendorfer and H. Jodlbauer, “An analytical model for service level and tardiness in a single machine MTO production system,” *Int. J. Prod. Res.*, vol. 49, no. 7, pp. 1827–1850, Apr. 2011, doi: 10.1080/00207541003660176.

[36] S. Savaşaneril, P. M. Griffin, and P. Keskinocak, “Dynamic Lead-Time Quotation for an M/M/1 Base-Stock Inventory Queue,” *Oper. Res.*, vol. 58, no. 2, pp. 383–395, Apr. 2010, doi: 10.1287/opre.1090.0717.

[37] D. Rajagopalan and I. A. Karimi, “Completion times in serial mixed-storage multiproduct processes with transfer and set-up times,” *Comput. Chem. Eng.*, vol. 13, no. 1–2, pp. 175–186, Jan. 1989, doi: 10.1016/0098-1354(89)89016-7.

[38] A. Hubl, K. Altendorfer, H. Jodlbauer, M. Gansterer, and R. F. Hartl, “Flexible model for analyzing production systems with discrete event simulation,” in *Proceedings of the 2011 Winter Simulation Conference (WSC)*, Phoenix, AZ, USA, Dec. 2011, pp. 1554–1565. doi: 10.1109/WSC.2011.6147873.

[39] M. Li, F. Yang, H. Wan, and J. W. Fowler, “Simulation-based experimental design and statistical modeling for lead time quotation,” *J. Manuf. Syst.*, vol. 37, pp. 362–374, Oct. 2015, doi: 10.1016/j.jmsy.2014.07.012.

[40] J. Medina-Marin, D. Gradi, and F. Nuñez-Piña, “A Petri Net Model to obtain the Makespan in the Flow Shop Scheduling Problem,” *Proc. World Congr. Eng. Comput. Sci.*, vol. Vol. 2, p. 5, 2016.

[41] M. Manzini and M. Urgo, “Makespan estimation of a production process affected by uncertainty: Application on MTO production of NC machine tools,” *J. Manuf. Syst.*, vol. 37, pp. 1–16, Oct. 2015, doi: 10.1016/j.jmsy.2015.10.001.

[42] W. Fang, Y. Guo, W. Liao, K. Ramani, and S. Huang, “Big data driven jobs remaining time prediction in discrete manufacturing system: a deep learning-

based approach," *Int. J. Prod. Res.*, vol. 58, no. 9, pp. 2751–2766, May 2020, doi: 10.1080/00207543.2019.1602744.

[43] S. Huang, Y. Guo, D. Liu, S. Zha, and W. Fang, "A Two-Stage Transfer Learning-Based Deep Learning Approach for Production Progress Prediction in IoT-Enabled Manufacturing," *IEEE Internet Things J.*, vol. 6, no. 6, pp. 10627–10638, Dec. 2019, doi: 10.1109/JIOT.2019.2940131.

[44] L. Lingitz, "Lead time prediction using machine learning algorithms: A case study by a semiconductor manufacturer," *Procedia CIRP*, vol. 72, pp. 1051–1056, 2018, doi: 10.1016/j.procir.2018.03.148.

[45] C. Wang and P. Jiang, "Deep neural networks based order completion time prediction by using real-time job shop RFID data," *J. Intell. Manuf.*, vol. 30, no. 3, pp. 1303–1318, Mar. 2019, doi: 10.1007/s10845-017-1325-3.

[46] G. E. Hinton, "Deep belief networks," *Scholarpedia*, vol. 4, no. 5, p. 5947, 2009.

[47] R. Salakhutdinov and I. Murray, "On the quantitative analysis of deep belief networks," in *Proceedings of the 25th international conference on Machine learning*, 2008, pp. 872–879.

[48] M. G. Coutinho, M. F. Torquato, and M. A. Fernandes, "Deep neural network hardware implementation based on stacked sparse autoencoder," *IEEE Access*, vol. 7, pp. 40674–40694, 2019.

[49] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016.

[50] K. Greff, R. K. Srivastava, J. Koutnik, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A Search Space Odyssey," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 10, pp. 2222–2232, Oct. 2017, doi: 10.1109/TNNLS.2016.2582924.

[51] S. Kanai, Y. Fujiwara, and S. Iwamura, "Preventing Gradient Explosions in Gated Recurrent Units," *Adv. Neural Inf. Process. Syst.*, pp. 435–444, 2017.

[52] W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent Neural Network Regularization," *ArXiv14092329 Cs*, Feb. 2015, Accessed: Aug. 14, 2020. [Online]. Available: http://arxiv.org/abs/1409.2329

[53] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*, vol. 1. Springer series in statistics New York, 2001.

[54] J. Huang, Q. Chang, and J. Arinez, "Predicting the Distribution of Product Completion Time in Multi-Product Manufacturing Systems," in *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*, Aug. 2021, pp. 316–323. doi: 10.1109/CASE49439.2021.9551468.

[55] H. Pham and H. Wang, "Imperfect maintenance," *Eur. J. Oper. Res.*, vol. 94, no. 3, pp. 425–438, 1996, doi: 10.1016/S0377-2217(96)00099-9.

[56] H. Wang, "A survey of maintenance policies of deteriorating systems," *Eur. J. Oper. Res.*, vol. 139, no. 3, pp. 469–489, Jun. 2002, doi: 10.1016/S0377-2217(01)00197-7.

[57] E. Ruschel, E. A. P. Santos, and E. de F. R. Loures, "Industrial maintenance decision-making: A systematic literature review," *J. Manuf. Syst.*, vol. 45, pp. 180–194, 2017, doi: 10.1016/j.jmsy.2017.09.003.

[58] H. Wang, "A survey of maintenance policies of deteriorating systems," *Eur. J. Oper. Res.*, vol. 139, no. 3, pp. 469–489, 2002, doi: 10.1016/S0377-2217(01)00197-7.

[59] M. Shafiee and M. Finkelstein, "An optimal age-based group maintenance policy for multi-unit degrading systems," *Reliab. Eng. Syst. Saf.*, vol. 134, pp. 230–238, 2015, doi: 10.1016/j.ress.2014.09.016.

[60] N. Chalabi, M. Dahane, B. Beldjilali, and A. Neki, "Optimisation of preventive maintenance grouping strategy for multi-component series systems: Particle swarm based approach," *Comput. Ind. Eng.*, vol. 102, pp. 440–451, 2016, doi: 10.1016/j.cie.2016.04.018.

[61] C. Zhang, W. Gao, S. Guo, Y. Li, and T. Yang, "Opportunistic maintenance for wind turbines considering imperfect, reliability-based maintenance," *Renew. Energy*, vol. 103, pp. 606–612, 2017, doi: 10.1016/j.renene.2016.10.072.

[62] X. Zhou, L. Xi, and J. Lee, "Opportunistic preventive maintenance scheduling for a multi-unit series system based on dynamic programming," *Int. J. Prod. Econ.*, vol. 118, no. 2, pp. 361–366, 2009, doi: 10.1016/j.ijpe.2008.09.012.

[63] T. Xia, L. Xi, E. Pan, and J. Ni, “Reconfiguration-oriented opportunistic maintenance policy for reconfigurable manufacturing systems,” *Reliab. Eng. Syst. Saf.*, vol. 166, no. November 2016, pp. 87–98, 2017, doi: 10.1016/j.ress.2016.09.001.

[64] T. Xia, L. Xi, X. Zhou, and J. Lee, “Dynamic maintenance decision-making for series-parallel manufacturing system based on MAM-MTW methodology,” *Eur. J. Oper. Res.*, vol. 221, no. 1, pp. 231–240, 2012, doi: 10.1016/j.ejor.2012.03.027.

[65] J. Li and S. Meerkov, *Production systems engineering*. 2008.

[66] Q. Chang, J. Ni, P. Bandyopadhyay, S. Biller, and G. Xiao, “Maintenance Opportunity Planning System,” *J. Manuf. Sci. Eng.*, vol. 129, no. 3, p. 661, 2007, doi: 10.1115/1.2716713.

[67] J. Li, D. E. Blumenfeld, N. Huang, and J. M. Alden, “Throughput analysis of production systems: recent advances and future topics,” *Int. J. Prod. Res.*, vol. 47, no. 14, pp. 3823–3851, Jul. 2009, doi: 10.1080/00207540701829752.

[68] A. Alrabghi and A. Tiwari, “State of the art in simulation-based optimisation for maintenance systems,” *Comput. Ind. Eng.*, vol. 82, pp. 167–182, Apr. 2015, doi: 10.1016/J.CIE.2014.12.022.

[69] A. Arab, N. Ismail, and L. S. Lee, “Maintenance scheduling incorporating dynamics of production system and real-time information from workstations,” *J. Intell. Manuf.*, vol. 24, no. 4, pp. 695–705, 2013, doi: 10.1007/s10845-011-0616-3.

[70] N. Nahas, “Buffer allocation and preventive maintenance optimization in unreliable production lines,” *J. Intell. Manuf.*, 2017.

[71] M. Colledani and T. Tolio, “Integrated quality, production logistics and maintenance analysis of multi-stage asynchronous manufacturing systems with degrading machines,” *CIRP Ann. - Manuf. Technol.*, vol. 61, no. 1, pp. 455–458, 2012, doi: 10.1016/j.cirp.2012.03.072.

[72] B. Bouslah, A. Gharbi, and R. Pellerin, “Integrated production, sampling quality control and maintenance of deteriorating production systems with

AOQL constraint," *Omega U. K.*, vol. 61, pp. 110–126, 2016, doi: 10.1016/j.omega.2015.07.012.

[73] J. Ni, X. Gu, and X. Jin, "Preventive maintenance opportunities for large production systems," *CIRP Ann. - Manuf. Technol.*, vol. 64, no. 1, pp. 447–450, 2015, doi: 10.1016/j.cirp.2015.04.127.

[74] X. Gu, X. Jin, W. Guo, and J. Ni, "Estimation of active maintenance opportunity windows in Bernoulli production lines," *J. Manuf. Syst.*, vol. 45, pp. 109–120, 2017, doi: 10.1016/j.jmsy.2017.08.005.

[75] P. Muchiri, L. Pintelon, L. Gelders, and H. Martin, "Development of maintenance function performance measurement framework and indicators," *Int. J. Prod. Econ.*, vol. 131, no. 1, pp. 295–302, May 2011, doi: 10.1016/J.IJPE.2010.04.039.

[76] J. Zou, Q. Chang, and Y. Lei, "Production Loss Diagnosis and Prognosis Using Model-based Data-driven Method," *IFAC-Pap.*, vol. 49, no. 12, pp. 1585–1590, 2016, doi: 10.1016/j.ifacol.2016.07.806.

[77] J. Zou, Q. Chang, J. Arinez, and G. Xiao, "Data-driven modeling and real-time distributed control for energy efficient manufacturing systems," *Energy*, vol. 127, pp. 247–257, 2017, doi: 10.1016/j.energy.2017.03.123.

[78] X. Ou, Q. Chang, J. Arinez, and J. Zou, "Knowledge-guided Reinforcement Learning for Gantry Work Cell Scheduling," *IEEE Access*, pp. 1–1, 2018, doi: 10.1109/ACCESS.2018.2800641.

[79] Y. H. Lie, Chang Hoon; Chun, "An Algorithm for Preventive Maintenance Policy," *IEEE Trans. Reliab.*, vol. 35, no. 1, pp. 71–75, 1986, doi: 10.1109/TR.1986.4335352.

[80] M. Kijima, "Some results for repairable systems with general repair," vol. 26, no. 1, pp. 89–102, 1989.

[81] J. Liu, Q. Chang, G. Xiao, and S. Biller, "The costs of downtime incidents in serial multistage manufacturing systems," *J. Manuf. Sci. Eng.*, vol. 134, no. 2, 2012.

[82] L. Li, Q. Chang, and J. Ni, "Data driven bottleneck detection of manufacturing systems," *Int. J. Prod. Res.*, vol. 47, no. 18, pp. 5019–5036, 2009.

[83] J. Huang, Q. Chang, J. Arinez, and G. Xiao, “A Maintenance and Energy Saving Joint Control Scheme for Sustainable Manufacturing Systems,” *Procedia CIRP*, vol. 80, pp. 263–268, 2019.

[84] J. Huang, Q. Chang, J. Zou, and J. Arinez, “A Real-Time Maintenance Policy for Multi-Stage Manufacturing Systems Considering Imperfect Maintenance Effects,” *IEEE Access*, vol. 6, pp. 62174–62183, 2018, doi: 10.1109/ACCESS.2018.2876024.

[85] J. Huang, Q. Chang, J. Zou, J. Arinez, and G. Xiao, “Real-time Control of Maintenance on Deteriorating Manufacturing System,” in *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*, 2018, pp. 211–216.

[86] J. Huang, Q. Chang, J. Arinez, and G. Xiao, “A Maintenance and Energy Saving Joint Control Scheme for Sustainable Manufacturing Systems,” *Procedia CIRP*, vol. 80, pp. 263–268, 2019, doi: 10.1016/j.procir.2019.01.073.

[87] S. M. Ross, *Introduction to Probability Models*. 2014. doi: 10.1080/00401706.1998.10485493.

[88] M.-H. Ye, “Optimal replacement policy with stochastic maintenance and operation costs,” *Eur. J. Oper. Res.*, vol. 44, no. 1, pp. 84–94, Jan. 1990, doi: 10.1016/0377-2217(90)90317-5.

[89] J. Liu, Q. Chang, G. Xiao, and S. Biller, “The Costs of Downtime Incidents in Serial Multistage Manufacturing Systems,” *J. Manuf. Sci. Eng.*, vol. 134, no. 2, p. 021016, 2012, doi: 10.1115/1.4005789.

[90] J. Zou, Q. Chang, J. Arinez, G. Xiao, and Y. Lei, “Dynamic production system diagnosis and prognosis using model-based data-driven method,” *Expert Syst. Appl.*, vol. 80, pp. 200–209, Sep. 2017, doi: 10.1016/j.eswa.2017.03.025.

[91] V. Ebrahimipour, A. Najjarbashi, and M. Sheikhalishahi, “Multi-objective modeling for preventive maintenance scheduling in a multiple production line,” *J. Intell. Manuf.*, vol. 26, no. 1, pp. 111–122, 2013, doi: 10.1007/s10845-013-0766-6.

[92] A. Barros, A. Grall, and C. Berenguer, “Joint modelling and optimization of monitoring and maintenance performance for a two-unit parallel system,” *Proc. Inst. Mech. Eng. Part O J. Risk Reliab.*, vol. 221, no. 1, pp. 1–11, 2007, doi: 10.1243/1748006XJRR31.

[93] K. S. de Smidt-Destombes, M. C. van der Heijden, and A. van Harten, “Joint optimisation of spare part inventory, maintenance frequency and repair capacity for k-out-of-N systems,” *Int. J. Prod. Econ.*, vol. 118, no. 1, pp. 260–268, 2009, doi: 10.1016/j.ijpe.2008.08.058.

[94] R. P. Nicolai and R. Dekker, “Optimal Maintenance of Multi-component Systems: A Review,” *Springer Ser. Reliab. Eng.*, vol. 8, pp. 263–286, 2008, doi: 10.1007/978-1-84800-011-7_11.

[95] H. Ab-Samat and S. Kamaruddin, “Opportunistic maintenance (OM) as a new advancement in maintenance approaches,” *J. Qual. Maint. Eng.*, vol. 20, no. 2, 2014, doi: 10.1108/JQME-04-2013-0018.

[96] T. Xia, X. Jin, L. Xi, and J. Ni, “Production-driven opportunistic maintenance for batch production based on MAM-APB scheduling,” *Eur. J. Oper. Res.*, vol. 240, no. 3, pp. 781–790, 2015, doi: 10.1016/j.ejor.2014.08.004.

[97] R. Laggoune, A. Chateauneuf, and D. Aissani, “Opportunistic policy for optimal preventive maintenance of a multi-component system in continuous operating units,” *Comput. Chem. Eng.*, vol. 33, no. 9, pp. 1499–1510, 2009, doi: 10.1016/j.compchemeng.2009.03.003.

[98] J. Zou, Q. Chang, Y. Lei, and J. Arinez, “Production System Performance Identification Using Sensor Data,” *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 48, no. 2, pp. 255–264, Feb. 2018, doi: 10.1109/TSMC.2016.2597062.

[99] M. C. Fitouhi, M. Nourelfath, and S. B. Gershwin, “Performance evaluation of a two-machine line with a finite buffer and condition-based maintenance,” *Reliab. Eng. Syst. Saf.*, vol. 166, no. March, pp. 61–72, 2017, doi: 10.1016/j.ress.2017.03.034.

[100] C. C. Karamatsoukis and E. G. Kyriakidis, “Optimal maintenance of two stochastically deteriorating machines with an intermediate buffer,” *Eur. J.*

Oper. Res., vol. 207, no. 1, pp. 297–308, 2010, doi: 10.1016/j.ejor.2010.04.022.

[101] X. Wang, H. Wang, and C. Qi, "Multi-agent reinforcement learning based maintenance policy for a resource constrained flow line system," *J. Intell. Manuf.*, vol. 27, no. 2, pp. 325–333, 2016, doi: 10.1007/s10845-013-0864-5.

[102] J. A. Ramírez-Hernández and E. Fernandez, "Optimization of preventive maintenance scheduling in semiconductor manufacturing models using a simulation-based approximate dynamic programming approach," *Proc. IEEE Conf. Decis. Control*, pp. 3944–3949, 2010, doi: 10.1109/CDC.2010.5717523.

[103] Y. Kang and F. Ju, "Flexible Preventative Maintenance for Serial Production Lines with Multi-stage Degrading Machines and Finite Buffers," *IISE Trans.*, vol. 5854, pp. 1–28, 2019, doi: 10.1080/24725854.2018.1562283.

[104] J. Huang, Q. Chang, J. Zou, and J. Arinez, "A Real-time Maintenance Policy for Multi-stage Manufacturing Systems Considering Imperfect Maintenance Effects," *IEEE Access*, pp. 1–1, 2018, doi: 10.1109/ACCESS.2018.2876024.

[105] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 2018.

[106] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Process. Mag.*, vol. 34, no. 6, pp. 26–38, 2017, doi: 10.1109/MSP.2017.2743240.

[107] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 2016-Janua, no. 7540, pp. 2315–2321, 2016, doi: 10.1038/nature14236.

[108] H. van Hasselt, A. Guez, and D. Silver, "Deep Reinforcement Learning with Double Q-Learning," *Thirtieth AAAI Conf. Artif. Intell.*, Mar. 2016.

[109] D. Silver *et al.*, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7585, pp. 484–489, 2016, doi: 10.1038/nature16961.

[110] M. Kijima, "Some Results for Repairable Systems with General Repair," *J. Appl. Probab.*, pp. 89–102, 1989.

[111] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual Multi-Agent Policy Gradients," *ArXiv170508926 Cs*, Dec. 2017, Accessed: Oct. 14, 2020. [Online]. Available: http://arxiv.org/abs/1705.08926

[112] J. Su, S. Adams, and P. A. Beling, "Counterfactual Multi-Agent Reinforcement Learning with Graph Convolution Communication," 2020, [Online]. Available: https://arxiv.org/abs/2004.00470

[113] T. Hastie, R. Tibshirani, and J. Friedman, "The Elements of Statistical Learning," *Math. Intell.*, vol. 27, no. 2, pp. 83–85, 2001, doi: 10.1198/jasa.2004.s339.

[114] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA Neural Netw. Mach. Learn.*, vol. 4, no. 2, pp. 26–31, 2012.

[115] J. Su, S. Adams, and P. A. Beling, "Value-Decomposition Multi-Agent Actor-Critics," *ArXiv200712306 Cs*, Aug. 2020, Accessed: Sep. 01, 2020. [Online]. Available: http://arxiv.org/abs/2007.12306

[116] K. Son, D. Kim, W. J. Kang, D. E. Hostallero, and Y. Yi, "Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning," in *International Conference on Machine Learning*, 2019, pp. 5887–5896.

[117] W. Böhmer, T. Rashid, and S. Whiteson, "Exploration with Unreliable Intrinsic Reward in Multi-Agent Reinforcement Learning," *ArXiv190602138 Cs*, Jun. 2019, Accessed: Mar. 26, 2022. [Online]. Available: http://arxiv.org/abs/1906.02138

[118] J. Huang, Q. Chang, and J. Arinez, "Deep reinforcement learning based preventive maintenance policy for serial production lines," *Expert Syst. Appl.*, vol. 160, p. 113701, Dec. 2020, doi: 10.1016/j.eswa.2020.113701.

[119] J. Su, J. Huang, S. Adams, Q. Chang, and P. A. Beling, "Deep multi-agent reinforcement learning for multi-level preventive maintenance in manufacturing systems," *Expert Syst. Appl.*, vol. 192, p. 116323, 2022.

[120] M. Wang, H. Huang, and J. Li, “Transients in flexible manufacturing systems with setups and batch operations: Modeling, analysis, and design,” *IISE Trans.*, vol. 0, no. 0, pp. 1–18, May 2020, doi: 10.1080/24725854.2020.1766728.

[121] M. Colledani., “Design and management of manufacturing systems for production quality,” *CIRP Ann.*, vol. 63, no. 2, pp. 773–796, 2014, doi: 10.1016/j.cirp.2014.05.002.

[122] Y. Wei, D. Wu, and J. Terpenny, “Decision-Level Data Fusion in Quality Control and Predictive Maintenance,” *IEEE Trans. Autom. Sci. Eng.*, p. 11.

[123] U. S. B. Rakiman and A. T. Bon, “Production Line: Effect of Different Inspection Station Allocation,” *Procedia Eng.*, vol. 53, pp. 509–515, 2013, doi: 10.1016/j.proeng.2013.02.066.

[124] A. Arun, K. Rameshkumar, D. Unnikrishnan, and A. Sumesh, “Tool Condition Monitoring Of Cylindrical Grinding Process Using Acoustic Emission Sensor,” *Mater. Today Proc.*, vol. 5, no. 5, pp. 11888–11899, 2018, doi: 10.1016/j.matpr.2018.02.162.

[125] T. Benkedjouh, K. Medjaher, N. Zerhouni, and S. Rechak, “Health assessment and life prediction of cutting tools based on support vector regression,” *J. Intell. Manuf.*, vol. 26, no. 2, pp. 213–223, Apr. 2015, doi: 10.1007/s10845-013-0774-6.

[126] C. Cooper *et al.*, “Convolutional neural network-based tool condition monitoring in vertical milling operations using acoustic signals,” *Procedia Manuf.*, vol. 49, pp. 105–111, 2020, doi: 10.1016/j.promfg.2020.07.004.

[127] J. Lenz, T. Wuest, and E. Westkämper, “Holistic approach to machine tool data analytics,” *J. Manuf. Syst.*, vol. 48, pp. 180–191, Jul. 2018, doi: 10.1016/j.jmsy.2018.03.003.

[128] D. A. Tobon-Mejia, K. Medjaher, and N. Zerhouni, “CNC machine tool’s wear diagnostic and prognostic by using dynamic Bayesian networks,” *Mech. Syst. Signal Process.*, vol. 28, pp. 167–182, Apr. 2012, doi: 10.1016/j.ymssp.2011.10.018.

[129] P. Wang, Z. Liu, R. X. Gao, and Y. Guo, “Heterogeneous data-driven hybrid machine learning for tool condition prognosis,” *CIRP Ann.*, vol. 68, no. 1, pp. 455–458, 2019, doi: 10.1016/j.cirp.2019.03.007.

[130] Y. Zhou and W. Sun, “Tool Wear Condition Monitoring in Milling Process Based on Current Sensors,” *IEEE Access*, vol. 8, pp. 95491–95502, 2020, doi: 10.1109/ACCESS.2020.2995586.

[131] J. Li, D. E. Blumenfeld, N. Huang, and J. M. Alden, “Throughput analysis of production systems: recent advances and future topics,” *Int. J. Prod. Res.*, vol. 47, no. 14, pp. 3823–3851, 2009.

[132] J. Zou, Q. Chang, J. Arinez, G. Xiao, and Y. Lei, “Dynamic production system diagnosis and prognosis using model-based data-driven method,” *Expert Syst. Appl.*, vol. 80, pp. 200–209, Sep. 2017, doi: 10.1016/j.eswa.2017.03.025.

[133] P. Sachin Krishnan and K. Rameshkumar, “Grinding wheel condition prediction with discrete hidden Markov model using acoustic emission signature,” *Mater. Today Proc.*, vol. 46, pp. 9168–9175, 2021, doi: 10.1016/j.matpr.2019.12.428.

[134] R. Zhao, R. Yan, Z. Chen, K. Mao, P. Wang, and R. X. Gao, “Deep Learning and Its Applications to Machine Health Monitoring: A Survey,” *ArXiv161207640 Cs Stat*, Dec. 2016, Accessed: Feb. 21, 2022. [Online]. Available: http://arxiv.org/abs/1612.07640

[135] M. Tomov, M. Kuzinovski, and P. Cichosz, “Development of mathematical models for surface roughness parameter prediction in turning depending on the process condition,” *Int. J. Mech. Sci.*, vol. 113, pp. 120–132, Jul. 2016, doi: 10.1016/j.ijmecsci.2016.04.015.

[136] X. Wang, S. Takaki, and J. Yamagishi, “An autoregressive recurrent mixture density network for parametric speech synthesis,” in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, New Orleans, LA, Mar. 2017, pp. 4895–4899. doi: 10.1109/ICASSP.2017.7953087.

[137] T. Furukawa, F. Bourgault, B. Lavis, and H. F. Durrant-Whyte, “Recursive Bayesian search-and-tracking using coordinated UAVs for lost targets,” in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, 2006, pp. 2521–2526.

[138] N. Bergman, “Recursive Bayesian estimation: Navigation and tracking applications,” PhD Thesis, Linköping University, 1999.

[139] R. Karlsson and F. Gustafsson, “Recursive Bayesian estimation: bearings-only applications,” *IEE Proc.-Radar Sonar Navig.*, vol. 152, no. 5, pp. 305–313, 2005.

[140] F. Scarselli, M. Gori, Ah Chung Tsoi, M. Hagenbuchner, and G. Monfardini, “The Graph Neural Network Model,” *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 61–80, Jan. 2009, doi: 10.1109/TNN.2008.2005605.

[141] J. Zhou *et al.*, “Graph Neural Networks: A Review of Methods and Applications,” *ArXiv181208434 Cs Stat*, Jul. 2019, Accessed: Apr. 29, 2020. [Online]. Available: http://arxiv.org/abs/1812.08434

[142] A. Galassi, M. Lippi, and P. Torroni, “Attention in Natural Language Processing,” *IEEE Trans. Neural Netw. Learn. Syst.*, pp. 1–18, 2020, doi: 10.1109/TNNLS.2020.3019893.

[143] A. P. Parikh, O. Täckström, D. Das, and J. Uszkoreit, “A Decomposable Attention Model for Natural Language Inference,” *ArXiv160601933 Cs*, Sep. 2016, Accessed: Jan. 06, 2021. [Online]. Available: http://arxiv.org/abs/1606.01933

[144] X. Ou, Q. Chang, and N. Chakraborty, “Simulation study on reward function of reinforcement learning in gantry work cell scheduling,” *J. Manuf. Syst.*, vol. 50, pp. 1–8, Jan. 2019, doi: 10.1016/j.jmsy.2018.11.005.

[145] O. Horiuchi and T. Shibata, “Computer Simulations of Cylindrical Plunge Grinding - Influence of Work Stiffness on Grinding Accuracy -,” *Key Eng. Mater.*, vol. 329, pp. 51–56, Jan. 2007, doi: 10.4028/www.scientific.net/KEM.329.51.

[146] J. Huang, J. Zhang, Q. Chang, and R. X. Gao, “Integrated process-system modelling and control through graph neural network and reinforcement learning,” *CIRP Ann.*, vol. 70, no. 1, pp. 377–380, 2021.

[147] M. I. Jordan and T. M. Mitchell, “Machine learning: Trends, perspectives, and prospects,” *Science*, vol. 349, no. 6245, pp. 255–260, 2015.

[148] Z. C. Lipton and J. Steinhardt, “Troubling trends in machine learning scholarship,” *ArXiv Prepr. ArXiv180703341*, 2018.

[149] S. Amershi *et al.*, “Guidelines for human-AI interaction,” in *Proceedings of the 2019 chi conference on human factors in computing systems*, 2019, pp. 1–13.

[150] M. H. Jarrahi, “Artificial intelligence and the future of work: Human-AI symbiosis in organizational decision making,” *Bus. Horiz.*, vol. 61, no. 4, pp. 577–586, 2018.

[151] Y. Mou and K. Xu, “The media inequality: Comparing the initial human-human and human-AI social interactions,” *Comput. Hum. Behav.*, vol. 72, pp. 432–440, 2017.

[152] A. Rai, P. Constantinides, and S. Sarker, “Next Generation Digital Platforms:: Toward Human-AI Hybrids,” *Mis Q.*, vol. 43, no. 1, pp. iii–ix, 2019.

[153] Q. Yang, A. Steinfeld, C. Rosé, and J. Zimmerman, “Re-examining whether, why, and how human-AI interaction is uniquely difficult to design,” in *Proceedings of the 2020 chi conference on human factors in computing systems*, 2020, pp. 1–13.

[154] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, “Continual lifelong learning with neural networks: A review,” *Neural Netw.*, vol. 113, pp. 54–71, May 2019, doi: 10.1016/j.neunet.2019.01.012.

[155] S. Narvekar, “Curriculum Learning for Reinforcement Learning Domains: A Framework and Survey,” *ArXiv Prepr. ArXiv200304960*, 2020.