Modeling and Control of Mobile Multi-Skilled Robot Operated Flexible Manufacturing Systems

---

A

Thesis

Presented to

the faculty of the School of Engineering and Applied Science

University of Virginia

---

in partial fulfillment

of the requirements for the degree

Master of Science

by

Kshitij Bhatta

May 2023

APPROVAL SHEET

This

Thesis

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

Master of Science

Author: Kshitij Bhatta

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

Advisor: Qing 'Cindy' Chang

Advisor:

Committee Member: Sarah Sun

Committee Member: Tariq Iqbal

Committee Member:

Committee Member:

Committee Member:

Committee Member:

Accepted for the School of Engineering and Applied Science:

Jennifer L. West, School of Engineering and Applied Science

May 2023

Acknowledgement

I would like to begin by expressing my deepest appreciation and gratitude to my advisor, Dr. Qing Chang, for her unwavering support and guidance throughout my time at UVA. She has been a remarkable mentor who has helped shape my academic and professional journey. I am also grateful to my undergraduate advisor, Dr. Francesco Sorrentino, for introducing me to the world of research and providing me with a strong foundation in the field.

I would like to extend my heartfelt thanks to my committee members, Dr. Sarah Sun and Dr. Tariq Iqbal, for their invaluable insights and expertise. Their courses have been among the most inspiring and thought-provoking that I have taken in my academic career.

I am also deeply grateful to my labmates, Dr. Jing Huang, Mr. Tian Yu, Ms. Chen Li, Mr. Muhammad Waseem, and Ms. Mehregan Gilani Nejad, with whom I have shared countless unforgettable moments. Their friendship, support, and encouragement have been instrumental in my academic success.

Lastly, I want to express my profound gratitude to my family and relatives for their unwavering love and support. I owe a special debt of gratitude to my parents for their unending support and sacrifices throughout the years. I am also grateful to my sister for her constructive criticism, which has helped shape me into the person I am today. Lastly, I would like to thank my soon-to-be wife, Bindica, for her unrelenting love and support through all the highs and lows of the last six years of our lives together.

Abstract

With the advent of Industry 4.0, there has been a tremendous leap in the field of smart manufacturing. Furthermore, recent advancements in machine learning and robotics are making human-robot collaboration a reality, ushering in yet another industrial revolution that many are calling Industry 5.0. In this context, the use of mobile collaborative robots (cobots) offers high flexibility for manufacturing systems, which calls for studies from the production system's perspective on how to effectively integrate robots into such systems. This thesis aims to provide a mathematical framework for a mobile multi-skilled robot-operated Flexible Manufacturing System (FMS).

The thesis focuses on the development of a comprehensive model for an FMS utilizing mobile multi-skilled robots and the establishment of an effective real-time control strategy using performance metrics derived from the model. The impact of incorporating robots into the manufacturing system is examined, and potential improvements in overall performance are explored.

Initially, a mathematical model for the FMS is established, and performance metrics are derived from the model that are used to solve a robot assignment control problem. Maintenance scheduling and tool changes are then incorporated into the model to create a more holistic representation of a manufacturing system. A comparison is made between models that only consider system-level information and those that only consider process-level information, with the integrated model significantly outperforming the others.

In summary, this thesis contributes to the development of appropriate frameworks for analyzing manufacturing systems that incorporate mobile multi-skilled robots and demonstrates the effectiveness of using performance metrics derived from mathematical models for control. The integration of maintenance scheduling and tool changes into the model provides a more comprehensive and realistic representation of the manufacturing system.

Contents

Chapter 1

Introduction

Manufacturing system modeling is the process of creating a mathematical representation of a manufacturing system in order to understand and optimize its operations. It involves analyzing the different components of the system, including machines, people, materials, and information, and how they interact with each other to produce goods or services. By creating a model of the system, manufacturers can simulate different scenarios, identify bottlenecks, and improve the efficiency of the system. This can lead to reduced costs, increased productivity, and improved quality. Furthermore, the model can be used to formulate control strategies which improve performance of the system in real-time. Thus, manufacturing system modeling is a valuable tool for manufacturers looking to stay competitive in an increasingly complex and global marketplace.

Modeling and control of manufacturing systems has its roots in the field of operations research, which emerged in the mid-20th century as a way to apply mathematical and analytical techniques to improve the efficiency and effectiveness of complex systems [1]. In the 1950s and 1960s, researchers began applying these techniques to manufacturing systems, recognizing that the principles of operations research could be used to optimize production processes, reduce costs, and increase productivity [2].

One of the earliest and most influential approaches to manufacturing system modeling was

developed by the mathematician and computer scientist George Dantzig, who is often credited with pioneering the field of linear programming. In the 1950s, Dantzig developed a mathematical model for optimizing production schedules that took into account the availability of resources, the sequence of operations, and other factors. This model, known as the "transportation problem," became a cornerstone of modern supply chain management [3].

In the decades that have followed, researchers have continued to develop new approaches to modeling and optimizing manufacturing systems, using techniques such as queuing theory, simulation, optimization algorithms and recently, machine learning. In 1984, Browne first introduced the term "Flexible Manufacturing" in his paper, defining it as an integrated system capable of processing medium-sized volumes of various part types with flexibility in machine, process, and operation [4]. Since then, numerous papers have explored the concept of flexibility in manufacturing, presenting different but valid ideas [5, 6, 7]. As such, flexibility in the context of FMS is not a clearly defined concept as there is no consensus on its meaning. There are various approaches to defining flexibility, such as the ability to produce different parts without extensive retooling, modify production schedules, handle multiple parts, or adjust production levels quickly to accommodate changes in demand or to shift capacity to different products or services. Despite the ambiguity of the term 'flexible', a key characteristic of FMS that has been widely accepted is its ability to modify its states and actions to meet various requirements without incurring significant costs in terms of time, effort, cost, or performance penalties [8].

Numerous innovative designs of FMS have been developed, with Toyota's flexible assembly lines being a prime example. These lines are capable of producing multiple vehicle models in any production sequence, enabling them to quickly respond to changes in market demand, shorten lead times, and improve production planning and ordering [9]. Multi-skilled robots are also being utilized in FMS. These robots can perform multiple tasks through multi-functional manipulators or quick-change end effectors and when mobile, can provide the flexibility to perform various tasks at different workstations scattered throughout the plant floor [10].

A similar approach has already been widely studied and implemented known as the "walking worker system" [11, 12]. This is where instead of having fixed workers in each workstation, the

workers are multi-skilled and walk from station to station. However, most studies that consider walking workers employ pre-defined heuristic polices and focus on steady state-analysis. Thus, despite significant research on the flexibility of manufacturing systems [4, 5, 6], there has been minimal performance evaluation of such systems. Traditional analytical and simulation methods typically focus on the steady-state dynamics of manufacturing systems using unrealistic models (e.g. Bernoulli process) [13, 14, 15], while the available literature on transient dynamics only considers fixed configuration production systems [16, 17, 18].

Thus, the overarching goal of this thesis is to create a comprehensive model for a flexible manufacturing system (FMS) that utilizes mobile multi-skilled robots, and to establish an effective real-time control strategy using performance metrics derived from the model. The study examines the impact of incorporating robots into the manufacturing system and explores how it can improve overall performance.

To achieve this goal, the rest of this thesis is organized as follows:

- Chapter 2 delves into the mathematical modeling of the FMS and the derivation of performance metrics that are then used to solve a robot assignment control problem.

- Chapter 3 builds on the previous chapter's model by incorporating maintenance scheduling and tool changes, along with robot assignment. This chapter stresses the importance of integrating these aspects into the manufacturing system control.

- Finally, chapter 4 presents prospective avenues for future research and provides a summary of the scientific contributions made by this thesis.

Chapter 2

Dynamic Robot Assignment

2.1 Background

In recent years, the manufacturing industry has increasingly shown interest in robotics technology as a means of improving efficiency, productivity, and quality while reducing costs and errors. Collaborative robots (cobots), which can work alongside human workers in the same workspace, are one of the most promising types of robots for manufacturing applications. Studies have shown that cobots can lead to improvements in productivity, quality, and safety [19]. Mobile robots are another type of robot that can be used for various tasks such as material handling, assembly, inspection, and maintenance in manufacturing facilities. These robots can move autonomously throughout the factory floor, reducing the need for human intervention and increasing efficiency [20].

Apart from cobots and mobile robots, recent research has focused on the development of social robots for manufacturing applications. Social robots are designed to interact with humans in a natural way and can perform various tasks such as product assembly, quality control, and customer service. For example, Yu et al. [21] proposed a framework for task allocation and scheduling in a human-robot collaboration environment, where a social robot is used to assist human workers in an assembly line. Similarly, Luo et al. [22] proposed a multi-robot system for quality inspection

in manufacturing, where social robots are used to assist human inspectors in identifying defects in products.

Advancements in machine learning and artificial intelligence have also enabled the development of more sophisticated robotic systems for manufacturing. Chen et al. [23] proposed a deep learning-based approach for robotic manipulation in manufacturing, where a robot is trained to grasp and manipulate objects in a dynamic environment. Li et al. [24] proposed an intelligent robot system for welding, where a robot is equipped with sensors and machine learning algorithms to improve the accuracy and quality of the welding process.

Despite the considerable work in the field of robotics for manufacturing, most studies have focused on how a robot can be designed to perform a particular activity, rather than investigating the effect of using robots in a general manufacturing system. In this chapter, we explore how a mobile multi-robot operated manufacturing system can be mathematically modeled, the type of real-time performance metrics that can be derived from the model, and how this domain knowledge can be leveraged to formulate a control strategy that effectively controls the assignment of robots.

2.2 System Description and Notations

Consider a mobile multi-skilled robot operated FMS which consists of w workstations and r mobile multi-skilled robots. There are $w-1$ intermediate buffers positioned between each workstation. The i^{th} robot is represented as $R_i(i = 1,\ldots,r)$, while the workstations and buffers are represented as $W_i(i = 1,\ldots,w)$ and $B_i(i = 2,\ldots,w)$, respectively. Each workstation can be operated by all robots which that can move freely within the plant floor. An illustration is shown in Fig. 2.1. The chapter adopts the following notation:

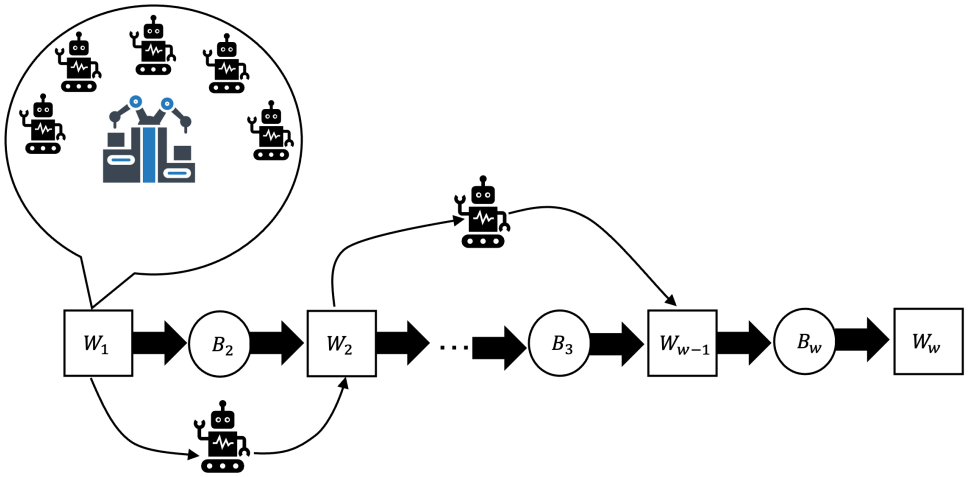

Figure 2.1: Illustration of mobile multi-skilled robot operated flexible serial production line

1. The set of all workstations in the production line is denoted by $\mathscr{W} = W_1, \ldots, W_w$, where W_i represents the i^{th} workstation, for $i = 1, 2, ..., w$.

2. The set of all robots in the production line is denoted by $\mathscr{R} = R_1, \ldots, R_r$, where R_j represents the j^{th} robot, for $j = 1, 2, ..., r$.

3. A $r \times 1$ vector $\mathbf{u}(t)$ is used to denote the assignment of all robots at time t, as the robots are able to move from one workstation to another. The element $u_j(t) = i$ if robot R_j belongs to workstation W_i. Each robot is assigned to a workstation at any given time.

4. The number of robots assigned to workstation W_i at time t is denoted by $m_i(t)$, for $i = 1, 2, \ldots, w$.

5. The i^{th} disruption event is denoted by $\vec{e}_i = (j, t_i, d_i)$, where j is the number of the robot that is down at time t_i for a duration of d_i, and $i = 1, 2, ..., n$ is the total number of disruption events, while $j = 1, 2, ..., r$ is the number of robots.

6. A robot is considered *working* in workstation W_i if it is assigned to the workstation and does not suffer any disruption events. The total number of working robots in workstation W_i is denoted by $\hat{m}_i(t)$, for $i = 1, 2, \ldots, w$.

7. The vector of disruption status' of all robots is denoted by $\boldsymbol{\theta} = [\theta_1(t), \theta_2(t), \ldots, \theta_r(t)]^T$,

where each element, θ_j is 1 if R_j suffers from a disruption event at time t, and 0 otherwise.

8. The $w \times 1$ vector of number of robots assigned (m_i) and working ($\hat{m}_i$) in all workstations, denoted by $\mathbf{M}(t)$ and $\hat{\mathbf{M}}(t)$ respectively, are referred to as the assignment configuration and working configuration of the production line at time t. The configuration of the production line denotes how many robots are assigned/working in each workstation.

9. Each workstation has a specified base cycle time T_i, where $i = 1, 2, ..., w$. This is the time it takes one robot in workstation W_i to produce one part. However, since each workstation can have more than one robot working, the total cycle time for each workstation at any time t is $\frac{T_i}{\hat{m}_i(t)}$. This direct proportionality is a simplification for mathematical purposes, and a different relation does not change the mathematical model to be developed. Note that $\hat{m}_i(t) \leq m_i(t) \forall t$ because it is not necessarily true that all assigned robots will be working.

10. The production speed of workstation W_i at time t is represented by $\mathfrak{s}_i(t)$.

11. The buffer levels at time t are denoted by $\mathbf{b}(t) = [b_2(t), b_3(t), ..., b_w(t)]^T$, where B_i is the buffer capacity of the i^{th} buffer, and $i = 2, 3, \ldots, w$.

12. $MTBF_j$ denotes the mean time between failure of robot R_j, and $MTTR_j$ denotes the mean time for the repair of robot R_j.

We make the following assumptions:

1. The system maintains a constant number of robots, which is represented by the equation $\sum_{i=1}^{w} m_i(t) = r$ for all times t.

2. Each workstation W_i has an upper limit $\hat{m}_i{}^{[U]}$ on the number of robots working on it at any given time, such that $\hat{m}_i(t)$ is always less than or equal to $\hat{m}_i{}^{[U]}$ for all times t.

3. A workstation is considered "slowed down" if the number of robots working on it at a particular time t is less than the number of robots assigned to it, denoted by $\hat{mi}(t)$ and mi respectively. A workstation is considered "down" if there are no robots working on it, i.e., $\hat{m}_i(t) = 0$ for $i = 1, 2, ..., w$.

4. A workstation is said to be "blocked" (or partially blocked) if it is functioning normally, but its immediate downstream buffer is full and the subsequent downstream workstation is either "down" or "slowed down".

5. A workstation is said to be "starved" (or partially starved) if it is operational, but its immediate upstream buffer is empty and the subsequent upstream workstation is either "down" or "slowed down".

6. The first workstation W_1 will never experience "starvation", while the last workstation W_w will never be "blocked", as the aim is to study the performance of this isolated production line.

2.3 System Model

Creating a control mechanism for a manufacturing system requires the development of a mathematical model of the system. Manufacturing systems are complex and dynamic, as they operate under both controlled inputs and unpredictable disturbances. One commonly used method for modeling these systems is through a state space equation. In the case of the Mobile Multi-Skilled Robot Operated FMS, the control input is represented by an assignment vector denoted as $\mathbf{u}$, while the production count of the final product represents the system output. By considering these variables, the state space equation for the Mobile Multi-Skilled Robot Operated FMS can be established as:

$$\dot{\mathbf{X}}(t) = \mathbf{F}\big(\mathbf{X}(t), \mathbf{u}(t), \boldsymbol{\theta}(t)\big) \tag{2.1}$$

where each component of Eq. (2.1) is defined as:

- $\mathbf{X}(t) = [X_1(t), \ldots, X_w(t)]^T$ is the state of the system. Each element, $X_i(t)$ represents the accumulated production counts of the i^{th} workstation up to time t.

- $\mathbf{F}(*)$ is a function that defines production speed of all workstations at any time t.

- $\mathbf{u}(t)$ is the control input to the system at time t. Here, it is the assignment vector.

- $\boldsymbol{\theta}(t)$ is the vector of disruption status' of each robot.

The value of $\mathbf{X}(t)$ is influenced by the aggregate cycle time of each workstation at time t, which is in turn determined by the number of robots operating at each workstation. The quantity of robots allocated to workstation W_i is represented by m_i:

$$m_i(t) = \sum_{j=1}^{r} \mathbf{1}_{u_j(t)=i} \tag{2.2}$$

where $\mathbf{1}_{u_j(t)=i}$ is a binary variable indicating whether or not robot R_j is assigned to workstation W_i. Assuming all assigned robots have arrived, the number of robots working at a workstation W_i can be defined using m_i and $\boldsymbol{\theta}$ as,

$$\hat{m}_i(t) = m_i(t) - \sum_{j=1}^{r} \theta_j(t)\mathbf{1}_{u_j(t)=i} \tag{2.3}$$

This shows the dependence of $\mathbf{X}(t)$ on $\mathbf{u}$ and $\boldsymbol{\theta}$.

The difference in accumulated production between any two workstations W_i and W_j ($i \neq j$), denoted by $\tau_{ij}(t)$, within a time period $[0,t]$ is determined by the conservation of flow, and can be expressed as:

$$\tau_{ij}(t) = \begin{cases} \sum_{k=j+1}^{i} b_k(0) - \sum_{k=j+1}^{i} b_k(t) & i > j \\ \sum_{k=i+1}^{j} b_k(t) - \sum_{k=i+1}^{j} b_k(0) & i < j \end{cases} \tag{2.4}$$

The upper limit of $\tau_{ij}(t)$ is determined by the relative location of workstation W_i with respect to W_j. Depending on this location, W_i can either be blocked or starved when $\tau_{ij}(t)$ exceeds a certain boundary, denoted by β_{ij}. Specifically, if W_j is located downstream of W_i and all buffers between them are full, then W_i is considered blocked. On the other hand, if W_j is located upstream of W_i and all buffers between them are empty, then W_i is considered starved. If we call this boundary β_{ij},

$$\beta_{ij} = \begin{cases} \sum_{k=j+1}^{i} b_k(0) & i > j \\ \sum_{k=i+1}^{j} B_k - \sum_{k=i+1}^{j} b_k(0) & i < j \end{cases} \tag{2.5}$$

The aforementioned boundary condition indicates that the difference between $X_i(t)$ and $X_j(t)$

should not exceed $\beta_{i,j}$. Nevertheless, if $X_i(t) - X_j(t) = \beta_{i,j}$ and $\frac{\hat{m}_i}{T_i}(t) > \frac{\hat{m}_j}{T_j}(t)$, then the speed of workstation W_i is limited by workstation S_j. Hence, an operational workstation operates either at its own pace or at the pace of its limiting workstation. This can be expressed mathematically as follows:

$$\dot{X}_i(t) = min\Big\{\zeta\Big(\big(X_i(t) - X_j(t)\big) - \beta_{ij}, \frac{\hat{m}_j(t)}{T_j}\Big), \frac{\hat{m}_i(t)}{T_i}\Big\} \tag{2.6}$$

where,

$$\zeta(u,v) = \begin{cases} +\infty \text{ if } u < 0 \\ v \text{ if } u = 0 \end{cases}$$

Comparing to all workstations in the system,we have,

$$\dot{X}_i = min\begin{cases} \zeta\Big(\big(X_i(t) - X_1(t)\big) - \beta_{i1}, \frac{\hat{m}_1(t)}{T_1}\Big) \\ \zeta\Big(\big(X_i(t) - X_2(t)\big) - \beta_{i2}, \frac{\hat{m}_w(t)}{T_2}\Big), \\ \vdots \\ \frac{\hat{m}_i(t)}{T_i} \\ \vdots \\ \zeta\Big(\big(X_i(t) - X_w(t)\big) - \beta_{ij}, \frac{\hat{m}_w(t)}{T_w}\Big) \end{cases} \tag{2.7}$$

$$= f_i(X_i(t), \hat{m}_i(t))$$

Since, $\hat{m}_i(t)$ is directly dependent on $u_i(t)$ and $\theta_i(t)$,

$$f_i(X_i(t), \hat{m}_i(t)) = f_i(X_i(t), u_i(t), \theta_i(t)) \tag{2.8}$$

Extending the idea to all workstations, we have,

$$\dot{\mathbf{X}}(t) = \begin{cases} f_1(X_i(t), u_i(t), \theta_i(t)) \\ f_2(X_i(t), u_i(t), \theta_i(t)) \\ \vdots \\ f_w(X_i(t), u_i(t), \theta_i(t)) \end{cases} \tag{2.9}$$

$$= \mathbf{F}(\mathbf{X}(t), \mathbf{u}(t), \boldsymbol{\theta}(t))$$

The output of the line is the output of the end-of-the-line workstation,

$$\mathbf{Y}(t) = X_w(t) = [0, 0, \ldots, 1]\mathbf{X}(t) \tag{2.10}$$

2.4 Evaluating Performance

A manufacturing system is a stochastic system that frequently experiences disruptions, resulting in downtime of machines. Therefore, it is essential to mitigate production losses caused by these random downtimes. Various control schemes can be implemented depending on the system used, such as turning machines on and off or adjusting cycle times. However, real-time system performance needs to be measured to enable the implementation of these strategies. Studies [25, 17, 26] have shown that Opportunity Window (OW) and Permanent Production Loss (PPL) are a valuable metric for evaluating just this. Unlike steady state metrics like throughput, which are uniform for all workstations in a serial line, PPL provides a more dynamic view of the production system where individual robots/workstations are impacted by each disruption event. This allows for the classification of events into those that result in permanent production loss and those that do not, which simplifies and strengthens control design. Additionally, OW can be viewed as the critical time that a disruption event must last to cause permanent production loss. This parameter is significant because it measures the time available for the control mechanism to make adjustments to the system before the system loses throughput. Here, we will first briefly introduce OW and PPL,

and then modify the definitions to the fit the new system.

2.4.1 Opportunity Window and Permanent Production Loss for a Serial Manufacturing Line

In a serial manufacturing line that operates sequentially and has limited buffer capacities, the opportunity window of a particular workstation W_j is denoted as $OW_j(t)$ and represents the duration required for all buffers between W_j and the last slowest machine, W_{w^*}, to fill up.

Definition 1. *Opportunity window of a workstation W_j, denoted by $OW_j(T_d)$, is the longest possible downtime due to a disruption event $\vec{e} = (j, T_d, d)$, that would not result in permanent production loss at the end of the line machine, i.e,*

$$\begin{aligned} OW_j(T_d) = sup\{d \geq 0 : s.t. \exists T^*(d) \\ \int_0^T \mathfrak{s}_w(t)dt = \int_0^T \mathfrak{s}_w(t,\vec{e})dt, \forall T \geq T^*(d)\} \end{aligned} \tag{2.11}$$

where $\int_0^T \mathfrak{s}_w(t)dt$ and $\int_0^T \mathfrak{s}_w(t,\vec{e})dt$ represent the production volume of the last workstation W_w at time t in the absence and presence of disruption event $\vec{e}$ respectively.

If the duration of a disruption event $\vec{e} = (j,t,d)$ exceeds $OW_j(t)$, then for any workstation W_j in the production line, there is a $T^* \geq t + d$, which may vary based on the location of W_j in relation to the slowest workstation W_{w^*}, such that:

$$\int_0^T \mathfrak{s}_w(t')dt' - \int_0^T \mathfrak{s}_w(t',\vec{e})dt' = \frac{d - OW_j(t)}{T_{w^*}} \forall T > T^* \tag{2.12}$$

Equation (2.12) defines the Permanent Production Loss (PPL), which refers to the production that cannot be recovered by the system under any circumstances due to the disruption event $\vec{e}$. The equation shows that PPL begins to accumulate once the impact of the disruption event reaches the last slowest machine W_{w^*}.

As the cycle times in a mobile multi-skilled robot operated FMS changes dynamically due to the movement of robots between workstations, the original definitions of PPL and OW are no longer applicable, and an extension to these concepts is necessary.

2.4.2 Extension to Mobile Multi-Skilled Robot Operated FMS

In order to extend the conepts of OW and PPL to the Mobile Multi-Skilled Robot Operated FMS, we first define what is called the "Ideal Clean Case".

Definition 2. *The Ideal Clean Case is a virtual scenario where there are no disruption events and all machines have enough resources to operate.*

This scenario represents the best possible condition for the system to be in. For a mobile multi-skilled robot operated FMS, the ideal clean case is when there are no disruption events and each workstation has at least one robot working.

Definition 3. *Permanent production loss of a manufacturing system is defined as the difference between the ideal clean case output, $Y_c(T)$ and the real output, $Y(T)$, that cannot be recovered.*

$$PPL = Y_c(T) - Y(T) \tag{2.13}$$

In a fixed configuration manufacturing system, the ideal clean case is straightforward as it occurs when all machines are operational, and it represents the highest achievable throughput of the system. However, in a mobile multi-skilled robot operated serial FMS, each workstation can have a variable number of robots ranging from one to $\hat{m}^{[U]}$, and the number of robots in one workstation can affect the robots' availability in another workstation. Thus, the ideal clean case can be achieved by several configurations that can be permutations of each other. Therefore, we introduce the notion of the ideal clean configuration, which refers to the configuration that results in the maximum possible throughput of the system among all the configurations that satisfy the ideal clean case. The uniqueness of this configuration is proven mathematically in Theorem 1.

Ideal Clean Configuration

According to the conservation of flow principle, a workstation in a serial line can operate at either its own rated speed or at the rate of its slower neighbor when the buffer between them is full/empty [27]. Therefore, in the ideal clean case, the system will eventually reach a steady state where the production speed is largely determined by the speed of the slowest workstation. As a result, the

ideal configuration must be one that minimizes the total cycle time of the slowest workstation among all possible configurations. However, there may be multiple configurations that meet this criterion. To determine the best configuration among the possible clean configurations, the time it takes for the system to reach steady state must also be considered. Prior to reaching steady state, the end-of-the-line workstation operates at its maximum speed until it is slowed down by neighboring slower workstations and ultimately by the slowest workstation. Therefore, the longer it takes for the end-of-the-line workstation to slow down, the greater the throughput obtained. Using this thought process, the ideal clean configuration is defined as follows,

Definition 4. *An ideal clean configuration* $\mathbf{M_c}$*, is the one that satisfies the ideal clean case scenario and the following two conditions (1)The total cycle time of the slowest workstation* $\frac{T_{w^*}}{m_w^*}$ *is minimized from among all possible configurations, and (2)The time taken for the end-of-the-line workstation to be slowed down is maximized.*

Theorem 1. *The ideal clean configuration* $\mathbf{M_c}$ *of a serial FMS with r robots, w workstations and a base cycle time of* $\mathbf{T}$*, is* ***unique****.*

Proof. Let $\mathscr{M}$ be the set of all possible configurations of an arbitrary production line L, where L is defined by the parameters r, w, and $\mathbf{T}$. The set $\mathscr{M}$ can be expressed as $\{1,2,\ldots,m^{[U]}\}^w$ subject to the constraint $\sum_i^w m_i = r$, where $\{*\}^w$ represents the Cartesian product of the set w times. Suppose L has an ideal configuration, $\mathbf{M_c} = [m_{1,c},\ldots,m_{w,c}]$.

We can always find one or multiple configurations in the set $\mathscr{M}$ such that the total cycle time for their slowest workstation W_{w^*} is the minimum among all elements of $\mathscr{M}$. According to condition (1) of definition 4, this must also be true for the ideal clean configuration $\mathbf{M_c}$. Moreover, since this cycle time is the minimum possible value for the slowest workstation, it is evident that this is the lowest possible total cycle time among all workstations from all elements of $\mathscr{M}$, i.e,

$$\frac{T_{w^*,c}}{m_{w^*,c}} = S \leq \frac{T_i}{m_i} \quad \forall \mathbf{M} \in \{1,2,\ldots,m^{[U]}\}^w \text{ s.t. } \sum_i^w m_i = r \tag{2.14}$$

$\frac{T_{w^*,c}}{m_{w^*,c}}$ denotes the total cycle time of the slowest workstation of the ideal clean configuration and $\frac{T_i}{m_i}$ is the total cycle time of any workstation due to any configuration from $\mathscr{M}$. S is the

minimum possible total cycle time for all workstations from among all configurations, i.e, no configuration can have a workstation with the total cycle time lower than S. For condition (2) to be satisfied, $W_{w^*,c}$ needs to be the furthest away from the end-of-the-line workstation, from among the configurations that satisfy Eq. (2.14), i.e, the subscript index of $W_{w^*,c}$ satisfies,

$$(w^*, c) = \text{argmax}(|w - w_i^*|), i = 1, \ldots, k \tag{2.15}$$

where k is the total number of configurations that satisfy Eq. (2.14).

Let $\mathbf{M_c}$ satisfy both Eqs. (2.14) and (2.15). Now suppose there exists another configuration, $\tilde{\mathbf{M}}$, obtained from the set $\mathscr{M}$, such that both Eqs. (2.14) and (2.15) are satisfied. Since $\tilde{\mathbf{M}}$ satisfies both conditions, and the base cycle time ($\mathbf{T}$) is the same for both configurations, the number of robots in the slowest workstation for both configurations has to be the same, i.e, $m_{w^*,c} = \tilde{m}_{w^*}$. This means that if $\mathbf{M_c}$ and $\tilde{\mathbf{M}}$ are distinctive ideal clean configurations, then $m_i \neq \tilde{m}_i$ has to be satisfied for at least one workstation that is not the slowest workstation. This will lead to one of the two cases:

1. $$m_{i,c} > \tilde{m}_i \implies \frac{T_i}{m_{i,c}} < \frac{T_i}{\tilde{m}_i} \tag{2.16}$$

   It is true that,

   $$\frac{T_i}{m_{i,c}} > S \implies -\frac{T_i}{m_{i,c}} < -S \tag{2.17}$$

   Adding Eq. (2.16) and Eq. (2.17), we get,

   $$S < \frac{T_i}{\tilde{m}_i} \tag{2.18}$$

   which implies $\tilde{W}_i$ is the slowest workstation in $\tilde{\mathbf{M}}$ which violates Eq. (2.15).

2. $$m_{i,c} < \tilde{m}_i \implies \frac{T_i}{m_{i,c}} > \frac{T_i}{\tilde{m}_i} \tag{2.19}$$

It is true that,

$$\frac{T_i}{m_{i,c}} > S \tag{2.20}$$

Subtracting Eq. (2.20) from Eq. (2.19), we get,

$$S > \frac{T_i}{\tilde{m}_i} \tag{2.21}$$

which violates Eq. (2.14).

Therefore, $m_i = \tilde{m}_i \forall\ i = 1, 2, \ldots w$ which means $\tilde{\mathbf{M}}$ and $\mathbf{M_c}$ are the same. Thus, uniqueness is proved. □

To determine the unique ideal clean configuration for a given serial FMS system with a specified number of robots r, workstations w, and base cycle time $\mathbf{T}$, a double optimization problem can be formulated. The solution to this problem is the ideal clean configuration denoted by $\mathbf{M}_c$.

$$\max_{\tilde{\mathbf{M}}} |w^* - w| \text{ s.t. } \begin{cases} \tilde{\mathbf{M}} \in \tilde{\mathscr{M}} = \min_{\mathbf{M}} \frac{T_{w^*}}{m_{w^*}} \text{ s.t. } \begin{cases} \frac{T_{w^*}}{m_{w^*}} = max(\mathbf{T} \oslash \mathbf{M}) \\ \sum_{i=1}^{w} m_i = r \\ \mathbf{M} \in \{1, 2, \ldots, \hat{m}^{[U]}\}^w \end{cases} \end{cases} \tag{2.22}$$

where $\oslash$ represents element-wise division. The first optimization step involves identifying a set of solutions, $\tilde{\mathscr{M}}$, that minimizes the total cycle time of the slowest workstation. The second optimization step determines the configuration from $\tilde{\mathscr{M}}$ that maximizes the distance between the slowest workstation w^* and the end-of-the-line workstation w. The result of this double optimization problem is the ideal clean configuration, $\mathbf{M}_c$. Recursive methods can be used to solve this problem efficiently, providing a quick and easy way to determine the ideal clean configuration without the need for time-consuming simulations.

Redefining OW and PPL

Every multi-skilled robot operated serial FMS has a unique ideal clean configuration that represents the maximum possible throughput of the system. Any deviation from the ideal clean configuration can be considered a disruption event. To provide more clarity, we introduce a new term

called "effective disruption event ($\vec{\hat{e}}$)" which is defined as follows:

Definition 5. *An effective disruption event, denoted by $\vec{\hat{e}}(i, T_d, d)$ is defined as any event that causes workstation W_i to have $\hat{m}_i(t) < m_{i,c}$, starting at time T_d and lasting for a time period of d.*

Furthermore, assumption 3 is also modified as,

A workstation is "slowed down" if the number of robots working on it at time t is less than the number of robots assigned to it in the clean case scenario, i.e, $\hat{m}_i(t) < m_{i,c}$ and it is "down" if there are no robots working, i.e, $\hat{m}_i(t) = 0$ for $i = 1, 2, ..., w$.

By utilizing the concept of effective disruption event, it becomes feasible to describe any type of PPL in the system, whether it stems from actual disruption events on robots or from an ineffective control strategy. Consequently, the opportunity window for a mobile multi-skilled robot operated serial FMS can be redefined as follows:

Definition 6. *Opportunity window of a workstation W_j, denoted by $OW_j(T_d)$, is the longest possible downtime/slow-down due to an effective disruption event $\vec{\hat{e}} = (j, T_d, d)$, that would not result in permanent production loss at the end of the line machine, i.e,*

$$\begin{aligned} OW_j(T_d) = sup\{d \geq 0 : s.t. \exists T^*(d) \\ \int_0^T \mathfrak{s}_w(t)dt = \int_0^T \mathfrak{s}_w(t, \vec{\hat{e}})dt, \forall T \geq T^*(d)\} \end{aligned} \tag{2.23}$$

The Permanent Production Loss (PPL) can be defined similarly,

$$\int_0^T \mathfrak{s}_w(t')dt' - \int_0^T \mathfrak{s}_w(t', \vec{\hat{e}})dt' = \big(d - OW_j(t)\big)\left(\frac{\hat{m}_{w^*,c}}{T_{w^*}} - \frac{\hat{m}_{w^*}(t)}{T_{w^*}}\right), \forall T > T^* \tag{2.24}$$

The permanent production loss caused by a single effective disruption event is represented by PPL_i. If there are n such events within the time horizon $[0, T]$, the total permanent production loss would be the sum of PPL_i for all effective disruption events. Therefore, we can express the total permanent production loss as follows:

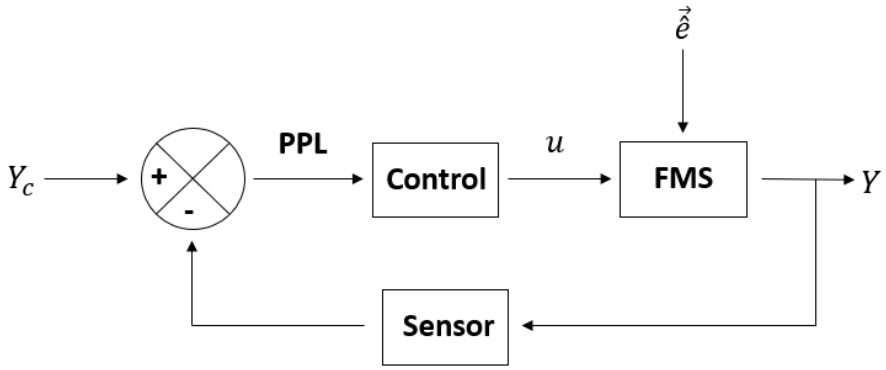

Figure 2.2: Control block diagram for mobile multi-skilled robot operated FMS

$$PPL(T) = \sum_{i=1}^{n} PPL_i. \tag{2.25}$$

2.5 Control Problem

To adapt to robot downtime, the control strategy for the mobile multi-skilled robot operated serial FMS needs to be dynamic, unlike the static ideal clean configuration in the absence of disruption events. Therefore, the assignment vector $\mathbf{u}(t)$ needs to be adjusted in real-time based on the system's current state. The effective downtime concept converts the control problem to one with a set point. The set point is the ideal clean case output Y_c, the control variable is the assignment vector $\mathbf{u}(t)$, and the input variable is PPL. The control framework is shown in Figure 2.2.

The control variable $\mathbf{u}(t)$ is coupled with the system state $\mathbf{X}(t)$ and $\boldsymbol{\theta}(t)$, meaning that inefficient control decisions lead to additional downtime. This makes control design a complex task. Moreover, there is no general closed-form representation for a multi-stage FMS, as described in Section II, which makes classic control methods all but impossible to apply.

To address these issues, a control strategy based on machine learning is used. Given that real-time robot assignment to workstations is a sequential decision-making problem, an MDP is considered the most appropriate approach to model it. Additionally, model-free reinforcement learning (RL) has demonstrated its effectiveness in solving such problems [28, 29].

2.5.1 MDP framework and Model-Free Reinforcement Learning

Markov Decision Process (MDP) is a mathematical framework to model sequential decision-making problems. It is defined as a tuple $\langle \mathscr{S}, \mathscr{A}, P, R, \gamma \rangle$, where:

- $\mathscr{S}$ is the state space, which represents the set of all possible states the system can be in.
- $\mathscr{A}$ is the action space, which represents the set of all possible actions that can be taken at each state.
- P is the state transition probability function, which gives the probability of transitioning from one state to another, given an action.
- R is the reward function, which gives a numerical reward for each transition from one state to another.
- γ is the discount factor, which determines the importance of future rewards.

The goal in an MDP is to find a policy, denoted as π, which is a mapping from states to actions [30]. The policy determines the action to take at each state to maximize the expected cumulative reward, denoted as $V^{\pi}(s)$, which is the expected sum of rewards starting from state s and following policy π. The optimal policy, denoted as π^*, is the policy that maximizes the expected cumulative reward, and the optimal value function, denoted as $V^*(s)$, is the expected cumulative reward when following the optimal policy.

Reinforcement learning (RL) is a subfield of machine learning that deals with decision-making problems in which an agent interacts with an environment to learn an optimal policy through trial and error [31]. In RL, the agent learns to maximize the cumulative reward by exploring the environment and updating its policy based on the observed rewards.

Model-free RL is a type of RL in which the agent learns the optimal policy without explicitly modeling the environment's dynamics or the transition probabilities [31]. Instead, the agent learns by interacting with the environment, observing the rewards obtained for each action, and updating its policy based on these rewards. Model-free RL algorithms can be categorized into two main types: value-based and policy-based.

In value-based RL, the agent learns the optimal value function, $V^*(s)$ or $Q^*(s,a)$, which represents the expected cumulative reward when following the optimal policy. The agent then derives the optimal policy from the optimal value function. Value-based RL algorithms include Q-learning [32], SARSA [33], and Deep Q-Networks (DQNs) [34].

In policy-based RL, the agent learns the optimal policy directly by parameterizing the policy and updating the policy parameters based on the observed rewards. Policy-based RL algorithms include REINFORCE [35], Actor-Critic [36], and Proximal Policy Optimization (PPO) [37].

2.5.2 Problem Formulation

To model the behavior of robots in an environment as a Markov Decision Process (MDP), it is necessary to define their movements in detail. This can be achieved by introducing intermediate states that the robots must traverse when moving between workstations. These intermediate locations provide a more accurate representation of robot movement as they account for the time delay incurred when robots are moving between workstations. The presence of these intermediate locations is illustrated in Fig. 2.3. The number of intermediate locations required is fixed and depends on the distance between the workstations. Additionally, there is no upper limit on the number of robots that can occupy each intermediate location at any given time. The set of all intermediate locations is denoted as $\mathscr{I}$.

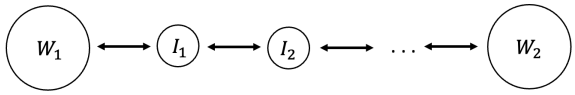

Figure 2.3: Illustration of intermediate locations within two workstations in the production line

2.5.3 States

The agent's state is composed of several factors including the assignment of each robot, the location of each robot, the disruption status of each robot, and the buffer status. Specifically, the state vector s is defined as,

$$s = [\mathbf{u}, \mathbf{l}, \boldsymbol{\theta}, \mathbf{b}] \tag{2.26}$$

where **u** denotes the target workstation for each robot and is determined by the selected action. The location state, **l**, indicates the current location of each robot in the production line, which can be either a workstation W_i or one of the intermediate locations I_i. This feature enables the environment to simulate real-world robot movement. The disruption status of each robot is represented by $\boldsymbol{\theta}$, and finally, **b** represents the buffer level at each buffer.

2.5.4 Actions

To maximize the cumulative reward in the FMS system, it is crucial to assign each robot to a specific workstation. Since there are w workstations available, each robot can be assigned to any of the w workstations, resulting in w choices for each robot. For all robots, the total number of possible choices will be w^r. In mathematical terms, we define the action set A as follows:

$$A = [W_{1,1}, W_{1,2}, \ldots, W_{1,r}, W_{2,1}, W_{2,2} \ldots, W_{w,r}] \tag{2.27}$$

Here, $W_{i,j}$ represents the assignment of robot R_j to workstation W_i.

2.5.5 State Transition Probability Matrix

In a manufacturing system, the probability of state transitions is influenced by several factors such as random disruption events, repair times, and system dynamics. Given the complexity of such systems, it is challenging to accurately calculate these probabilities without making strong assumptions that may not hold in the real world. To address this, we use a model-free algorithm that learns these probabilities from sampled experiences instead of explicitly calculating them.

2.5.6 Reward

To determine the desirability of an action based on the current state, a scalar value called the reward is used. The selection of an appropriate reward function is crucial for ensuring the convergence of the algorithm to the optimal policy. In this study, we adopt the negative step-wise permanent production loss as the reward function, as it has been shown to be effective in similar studies [28, 29]. The reward function is defined as the negative sum of the step-wise permanent production

	R_1	R_2	R_3	R_4
$MTBF_i$(min)	50	30	60	80
$MTTR_i$(min)	4	5	8	3

	W_1	W_2	W_3
T_i(min)	6	7	8

	B_2	B_3
B_i	6	8
(ϕ_{max}, ϕ_{min})	(1,5)	(1,7)

Table 2.1: Parameters for Mobile multi-skilled robot operated FMS

loss:

$$R(s,a,s') = -PPL(s,a,s'), \tag{2.28}$$

where *PPL* can be calculated using Eq. (2.25).

2.5.7 Double DQN Learning

Q-learning is a popular algorithm in reinforcement learning used for finding the optimal action-selection policy based on the maximum expected cumulative reward [32]. It learns a Q-value function that estimates the expected reward of taking a particular action in a given state. The Q-value function is updated using the Bellman equation, which expresses the expected value of the current state-action pair as the sum of the immediate reward and the expected value of the next state-action pair. Q-learning has been successfully applied in various domains, including games, robotics, and finance.

Deep Q-Networks (DQNs) are an extension of Q-learning that use deep neural networks to approximate the Q-value function. DQNs were introduced by Mnih et al. in 2015 [38] and have been shown to achieve state-of-the-art results in various game environments. DQNs use a neural network to approximate the Q-value function, taking the current state as input and producing a vector of Q-values for all possible actions. The Q-value for the selected action is then used to update the network using backpropagation.

Double Deep Q-Networks (DDQNs) were introduced as a modification to DQNs to address the overestimation of Q-values that can occur in the original algorithm [34]. In DDQNs, two

separate neural networks are used to approximate the Q-value function, and the target Q-values are calculated using the Q-network with the best action selection rather than the one being updated. This approach has been shown to improve stability and reduce overestimation in the Q-value estimates. Another modification of DDQN is the use of a technique called "experience replay." This technique involves storing transitions (i.e., state, action, reward, next state) in a replay buffer and sampling them randomly during the training process. This approach helps to remove the correlation between subsequent experiences and allows the agent to learn from past experiences more efficiently.

For this study, we opt to use the Double DQN algorithm since it has fared well in several applications with large state-spaces like the case with the robot assignment problem.

2.6 Case Study

To demonstrate the effectiveness of the proposed control strategy, a simulation experiment is conducted on an FMS with $w = 3$ workstations, $r = 4$ robots, and two buffers. The aim of the experiment is to train the agent to learn the optimal assignment policy using the proposed method and validate its effectiveness by comparing it with other policies. The experiment comprises two steps: (1) training the agent to obtain the optimal assignment policy and (2) validating the effectiveness of the learned policy.

Random disruption events are generated using the Mean Time Between Failure ($MTBF$) and Mean Time To Repair ($MTTR$) values of individual robots, assuming an exponential distribution. These values, along with the base cycle time of each workstation $T_i (i = 1, 2, \ldots w)$ and the capacity of each buffer $B_i (i = 2, 3, \ldots w)$, are listed in Table 2.1. The initial buffer level $b_i (i = 2, 3, \ldots w)$ is randomly selected between 0 and B_i to ensure the robustness of the system. The number of intermediate locations between each workstation $|\mathscr{I}|$ is set to 1, making the total number of locations 5.

In the experiment, the agent is trained using the double deep Q-learning algorithm to learn the optimal assignment policy. The learned policy is then compared with other policies to validate its effectiveness. The simulation results demonstrate the superiority of the proposed control strategy,

with the learned policy outperforming the other policies in terms of the production loss and buffer level. This shows the effectiveness of the proposed approach in improving the performance of FMSs.

2.6.1 Training

During the training phase of the simulation experiment, the proposed method is used to obtain the optimal robot assignment policy. The simulation is run for several episodes until convergence, where each episode represents a week's time, i.e, 3000 minutes assuming 10 hours a day on a 5 day workweek. The unit time for the simulation is set to 1 minute, and at the end of each episode, the state is reset to a random start to ensure robustness. A deep neural network with two dense hidden layers, each containing 32 nodes, is used for Q-value approximation. The learning rate α is set to 0.0001, and an ε-decay schedule is implemented, where the value of epsilon is initialized at $\varepsilon = 1$ and decays down to $\varepsilon = 0.1$ in 100,000 steps and stays constant afterwards. The reward used in the simulation is -PPL, and the discount factor γ is set to 0.95.

The convergence of the reward obtained during training is shown in Fig. 2.4. As seen in the figure, at around 200,000 steps, the reward stabilizes, and the policy obtained is considered good enough for robot assignment. The learned policy is then validated by comparing it with other policies in the next step of the experiment.

2.6.2 Performance Evaluation and Comparison

The performance of the system is assessed by comparing the production counts achieved using different policies in a one-week period. These policies include the learned policy based on the proposed method, a random assignment policy, a baseline static policy with no control, and two knowledge-guided heuristic policies. The no-control and random assignment are self explanatory whereas the knowledge guided heuristic control schemes are developed using the concepts of opportunity window, ideal clean case and permanent production loss, and can outperform a system without any dynamic assignment. These control methods are sensor enabled and are similar to what are employed in industry.

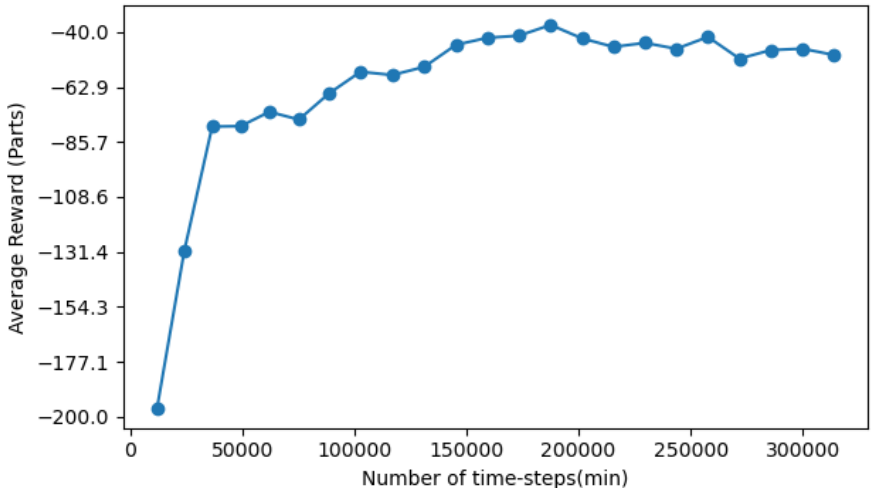

Figure 2.4: Training data showing, Reward v.s. Time-steps

Buffer-guided Robot Assignment(BgRA)

The Buffer-guided Robot Assignment (BgRA) is a control scheme for dynamic robot assignment which uses real time buffer levels upstream and downstream of the last slowest machine to reduce permanent production loss. The main idea behind this control is that increasing the opportunity window due to a disruption event will decrease the permanent production loss in the system (2.24). To realize this strategy, sensors are used to measure buffer levels at every time-step and a maximum and minimum buffer threshold (ϕ_{max}, ϕ_{min}) is defined beyond which the workstations upstream or downstream of the buffer are at risk of getting blocked(partially) or starved(partially). The value of this threshold, which is usually a percentage of the buffer capacity, is user dependent and can be tuned based on how long a robot takes to move from one workstation to another. Even though measurements are taken every time-step, control action is only taken when all robots have reached their assigned workstations. Consider the following notation:

1. $\mathscr{R}_i$ as the set of all robots working in workstation W_i, $i = 1,2,\ldots,w$, i.e, a robot R_j belongs to $\mathscr{R}_i$ if $u_j = i$. Therefore, $\hat{m}_i = |\mathscr{R}_i|$.

2. $x_{|\alpha|} \in_R \mathscr{R}_i$ represents randomly choosing a set x which contains α elements from $\mathscr{R}_i$.

3. $u(x_{|\alpha|} \in_R \mathscr{R}_i) = i+1$ represents updating the randomly chosen robots' assignment to to $i+1$.

This strategy only allows robots to move between neighbors where a neighbor of workstation W_i is defined as the immediate workstation on both sides of it, i.e, W_{i+1} and W_{i-1}. Every control action in BgRA has 3 steps and the assignment vector $\mathbf{u}$ is updated every step. It is important to note that a change in $\mathbf{u}$ corresponds to a change in $\mathscr{R}_i$ and $\hat{m}_i$, so all of them are updated simultaneously.

1. Initially, some robots are assigned between W_{w^*}, W_{w^*+1} and W_{w^*-1} depending on whether there is a disruption event at W_{w^*}.

   - If $\hat{m}_{w^*,c}$- $\hat{m}_{w^*} = \beta > 0$, consider $\alpha = min(\beta, \hat{m}_{w^*-1})$ and implement:

   $$\begin{aligned} u(x_{|\alpha|} \in_R \mathscr{R}_{w^*-1}) &= w^* \text{ if } B_{w^*} - b_{w^*} > b_{w^*+1} \\ u(x_{|\alpha|} \in_R \mathscr{R}_{w^*+1}) &= w^* \text{ if } B_{w^*} - b_{w^*} < b_{w^*+1} \end{aligned} \tag{2.29}$$

   - If $\hat{m}_{w^*,c}(t) - \hat{m}_{w^*} = \beta = 0$, no new assignment.
   - If $\hat{m}_{w^*,c}(t) - \hat{m}_{w^*} = \beta < 0$, consider $\alpha = |\beta|$ and implement:

   $$\begin{aligned} u(x_{|\alpha|} \in_R \mathscr{R}_{w^*}) &= w^* + 1 \text{ if } B_{w^*} - b_{w^*} > b_{w^*+1} \\ u(x_{|\alpha|} \in_R \mathscr{R}_{w^*}) &= w^* - 1 \text{ if } B_{w^*} - b_{w^*} < b_{w^*+1} \end{aligned} \tag{2.30}$$

   If $\mathscr{R}_{w^*-1}$ is empty meaning there are no robots working in W_{w^*-1}, the random choice results in an empty set, therefore no new assignment is made. The same idea applies to steps 2 and 3 below.

2. Using the updated $\mathscr{R}_i$s and m_is from step 1, the following steps are repeated for all workstations upstream of W_{w^*} except the first workstation.

   - If $b_i \geq \phi_{max}$ and $\frac{\hat{m}_i}{T_i} > \frac{\hat{m}_{i+1}}{T_{i+1}}$

   $$u(x_{|1|} \in_R \mathscr{R}_i) = i - 1 \tag{2.31}$$

   - If $b_i \leq \phi_{min}$ and $\frac{\hat{m}_i}{T_i} < \frac{\hat{m}_{i+1}}{T_{i+1}}$

   $$u(x_{|1|} \in_R \mathscr{R}_{i-1}) = i \tag{2.32}$$

   - No change otherwise.

3. Meanwhile, also using the updated $\mathscr{R}_i$s and m_is from step 1, the following steps are repeated for all workstations downstream of W_{w^*} except the last workstation. This can be done in parallel with step 2. Each workstation upstream of W_{w^*} is also denoted by W_i:

   - If $b_{i-1} \geq \phi_{max}$ and $\frac{\hat{m}_{i-1}}{T_{i-1}} > \frac{\hat{m}_i}{T_i}$

   $$u(x_{|1|} \in_R \mathscr{R}_{i+1}) = i \tag{2.33}$$

   - If $b_{i-1} \leq \phi_{min}$ and $\frac{m_{i-1}}{T_{i-1}} < \frac{m_i}{T_i}$

   $$u(x_{|1|} \in_R \mathscr{R}_i) = i+1 \tag{2.34}$$

   - No change otherwise.

Since the first and last workstations are farthest from the slowest one, filling or emptying of their adjacent buffers (b_2 and b_w) cannot be controlled using this strategy. In fact, the goal is to move robots such that any effective disruption event upstream(downstream) of the slowest workstation gets "propagated" to the first(last) workstation by replacing the non-operational robots with operational ones upstream(downstream) of the affected workstation.

Ideal Configuration Mimicking(ICM)

Ideal Configuration Mimicking uses the optimization problem from (2.22) to find an ideal clean configuration for the current number of working robots in the system. This means that the constraint of $\sum_{i=1}^{w} \mathbf{M}_{i,1} = r$ is replaced by $\sum_{i=1}^{w} \mathbf{M}_{i,1} = \sum_{i=1}^{w} \hat{\mathbf{M}}_{i,1}(t)$ where, $\hat{\mathbf{M}}(t)$ is the working configuration of the system. Every time there is a disruption event in a robot, the number of working robots decrease and a new ideal clean configuration is found. However, this control scheme fails when the number of working robots is less than the number of workstations because the definition of the ideal clean case dictates that each workstation needs to have at-least one robot working.

Consider $\mathbf{M}_c - \hat{\mathbf{M}} = \boldsymbol{\alpha}$, where the vector $\boldsymbol{\alpha}$ refers to change in the number of robots from the ideal clean configuration. A positive element of $\boldsymbol{\alpha}$ denotes that workstation W_i needs to gain α_i

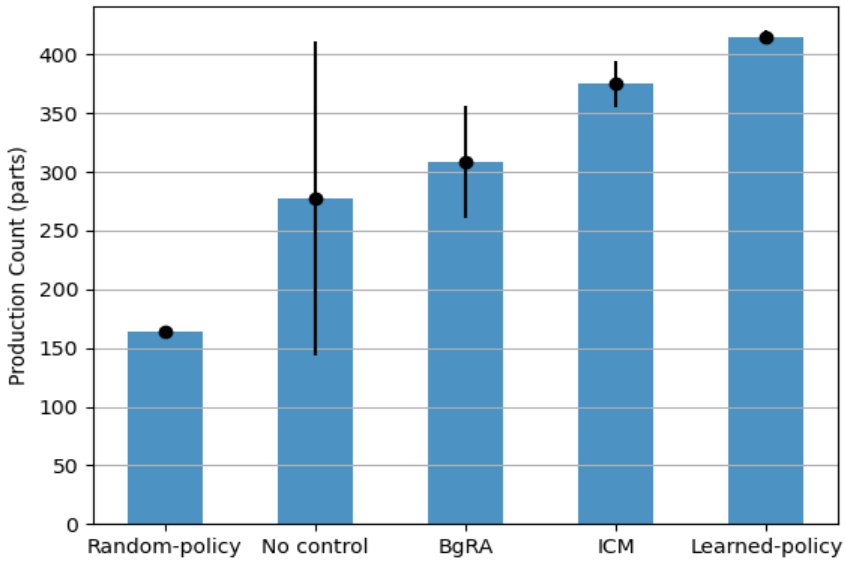

Figure 2.5: Learned policy compared with other policies. Each bar shows mean and 95% confidence interval

number of robots and a negative element denotes that workstation W_i needs to loose α_i number of robots to get to the ideal clean case configuration. Implementation of this control scheme is as follows:

$$u(x_{|\alpha_i|} \in_R \mathscr{R}_i) = j \; \forall \; \alpha_i < 0 \text{ and } \alpha_j > 0. \tag{2.35}$$

For this control strategy, control action is taken every time the number of working robots change.

Comparison

The performance of the learned policy was evaluated through a one-week test duration consisting of 20 iterations, and the results are shown in Fig. 2.5. The average production counts of each policy were compared, and it was found that the learned policy outperformed all of its competitors, including the random control policy, no control policy, BgRa policy, and ICM policy. The learned policy had an average improvement over the random policy of 254.3 parts, over the no control policy of 143.6 parts, over the BgRa policy of 60.3 parts, and over the ICM policy of 36.6 parts.

The error bars on the graph represent the 95% confidence interval of each policy, and it was

observed that the learned policy's interval did not overlap with any of the other policies, indicating that it was objectively better than the rest. The no-control policy had a large confidence interval due to its sensitivity to disruption events, and even its highest mean value was less than that of the learned policy. The random policy had the worst performance, but it had a low variance due to the coupling of effective disruption events with the robot assignment, which led to consistently worse performance.

In contrast, the learned policy was robust, with an average production of 420.5 parts per week, indicating its effectiveness in improving the production count of the FMS system while ensuring resilience in the face of random disruption events. The ICM policy was found to be inconsistent in performance, as indicated by its confidence interval. These results highlight the potential of the proposed control framework and the learned policy to significantly enhance the efficiency of the FMS system.

2.7 Summary

The chapter proposes a new approach to flexible manufacturing systems (FMS) by utilizing mobile multi-skilled robots capable of adapting to random disruptions. The concepts of Opportunity Window and Permanent Production Loss are extended to such a system, which allows for the development of a control framework using Markov Decision Processes (MDP) formulation.

By applying reinforcement learning to solve the problem, a policy is trained that outperformed several heuristic policies in a one-week test duration. The results showed that the proposed control framework could significantly improve the production count of the FMS system while ensuring robustness in the face of random disruption events.

The study's findings and methodology could have significant implications for the manufacturing industry. As FMS systems become increasingly complex, incorporating mobile robots that can adapt to disruption events and optimizing their control policies could enhance production efficiency and reduce downtime. Further exploration of more optimal assignment policies using state-of-the-art methods could lead to even more significant improvements in production performance.

2.8 Related Work

Part of the results presented in this chapter have been published in [39, 40]

Chapter 3

Integrating Robot Assignment, Tool Change and Preventive Maintenance

3.1 Background

Production control, tool change, and maintenance scheduling are important components of manufacturing systems that have been studied extensively in isolation. For example, Mourtzis and Doukas [41] examined production control methods to optimize machine utilization, while Ben Mansour et al. [42] focused on tool change scheduling to minimize production time. Maintenance scheduling has also been studied, with researchers investigating various maintenance policies such as preventive, corrective, and condition-based maintenance [43, 44].

However, despite these individual efforts, there is a lack of research that integrates all three components of production control, tool change, and maintenance scheduling [45, 46]. Babu and Prasad [45] conducted a review of the literature on tool change and maintenance scheduling in flexible manufacturing systems, and identified the need for further research on the integration of these two components with production control. Gebennini et al. [46] also conducted a review of the literature on the integration of maintenance, production planning, and control in manufacturing

systems, and found that most studies focused on the integration of only two components, with limited research on the integration of all three.

Thus, there is a need for further investigation into the integration of these three components in manufacturing systems to optimize production efficiency and reduce downtime [46, 47]. Mustaffa et al. [47] proposed a framework for integrated maintenance, production planning, and control that takes into account the dependencies and interactions between these three components. The framework includes decision-making models for maintenance scheduling, production planning, and tool change scheduling, as well as optimization algorithms to minimize downtime and production costs.

Other researchers have also proposed methods for integrating production control, tool change, and maintenance scheduling. For example, Abdellatif et al. [48] proposed an integrated framework that considers the impact of tool wear and machine availability on production scheduling and tool change planning. Li et al. [49] proposed a model that integrates production scheduling, tool change planning, and maintenance scheduling for a multi-product manufacturing system.

Since production control, tool change, and maintenance scheduling are closely interrelated in a manufacturing system, focusing on only one aspect may not effectively improve the system's efficiency. Therefore, in this chapter, we will explore the integration of these aspects and compare the effectiveness of control strategies that incorporate all three components against those that do not.

3.2 System Description and Notation

The system under consideration is very similar to the one from Chapter 2 but here, quality control mechanisms are placed after each workstation for the removal of defective parts. These mechanisms are denoted as $Q_i (i = 1, \ldots, w)$. Furthermore, every robot is assumed to have all tools required to work in each workstation. The system is illustrated in Fig. 3.1. Here are the notations, which are different from/have not been previously introduced in Chapter 2:

1. $\boldsymbol{\theta} = [\theta_1(t), \theta_2(t), \ldots, \theta_r(t)]^T$ is the vector of working status of all robots. Each element, θ_j is 1 if R_j is working and 0 otherwise. A robot is considered *working* in workstation W_i if it

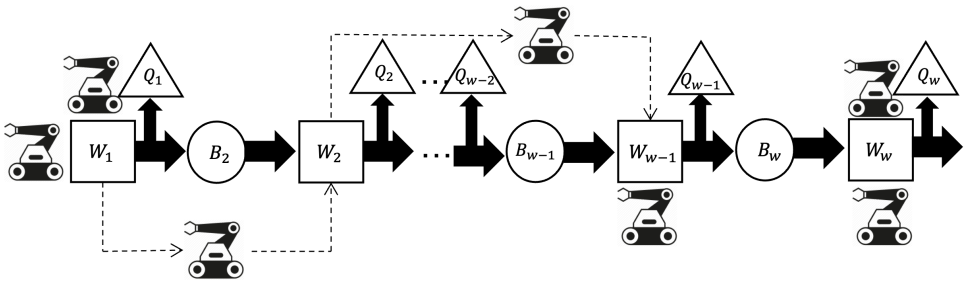

Figure 3.1: Mobile multi-skilled robot operated FMS with added quality control mechanisms

has arrived to it's assigned workstation and is not currently undergoing maintenance;

2. Each robot has one tool for each workstation making w tools per robot. State of robot R_j's i^{th} tool at time t is denoted as $x_j^i(t)$, $i = 1,2,\ldots,w \; \forall \; j = 1,2,3\ldots,r$;

3. Age of each robot at time t is represented by $a_j(t)$ where $j = 1,2,\ldots,r$. There is a prescribed age for each robot which upon reaching, the robot will fail, we denote this age by A_j;

4. With abuse of notation, $Q_i(t)$ represents the number of parts that have been discarded from W_i, for $i = 1,\ldots,w$;

5. $MTBF_j$ and $MTTR_j$ represents the mean time between failures and mean time to repair for robot R_j. This applies for random disruption events;

The following assumptions are made:

1. All parameters and quantities discussed in the paper are in discrete time. An argument of t, is adopted for ease of notation and corresponds to the value of the parameter at discrete time t. Time advances with a small timestep δt;

2. The total number of robots in the system remains constant, i.e, $\sum_{i=1}^{w} m_i(t) = r, \forall t$.

3. At any given time, the number of robots working in W_i cannot exceed an upper bound, $\hat{m}_i{}^{[U]}$, i.e, $\hat{m}_i(t) \leq \hat{m}_i{}^{[U]}, \forall t$;

4. The system has a unique and constant ideal clean configuration, i.e, the number of robots in each workstation that results in the best performance is constant and unique, This number is represented by $\hat{m}_{i,c}$ for the workstation W_i [39].

5. A workstation is *slowed down* if the number of robots working on it at time t is less than the number of robots in the ideal clean configuration, i.e, $\hat{m}_i(t) < \hat{m}_{i,c}$ and it is *down* if there are no robots working, i.e, $\hat{m}_i(t) = 0$ for $i = 1, 2, ..., w$;

6. A workstation is blocked (partially blocked) if it is operational, its immediate downstream buffer is full and the subsequent downstream workstation is down (slowed down);

7. A workstation is starved (partially starved) if it is operational, it's immediate upstream buffer is empty and the subsequent upstream workstation is down (slowed down);

8. The first workstation W_1 is never starved and the last machine W_w is never blocked since the performance of this isolated production line is to be studied;

3.3 System Model

As in chapter 2, the dynamics of a manufacturing system can be modeled using a state space equation:

$$\dot{\mathbf{X}}(t) = \mathbf{F}\big(\mathbf{X}(t), \mathbf{Q}(t), \hat{\mathbf{u}}(t), \boldsymbol{\theta}(t)\big) \tag{3.1}$$

where each component of Eq. (3.1) is defined as:

- $\mathbf{F}(*)$ is a function that defines the rate of part production for all workstations at any time t.
- $\mathbf{X}(t) = [X_1(t), \ldots, X_w(t)]^T$ is the system state, such that $X_i(t)$ is the cumulative production count of workstation W_i up to time t.
- $\mathbf{Q}(t) = [Q_1(t), \ldots, Q_w(t)]^T$, such that $Q_i(t)$ is the cumulative defective product count from workstation W_i at time t.
- $\hat{\mathbf{u}}(t)$ is vector of control inputs of all robots at time t.
- $\boldsymbol{\theta}(t)$ is the vector of all robot status' at time t.

3.3.1 Control Inputs

The control system of the manufacturing process involves various types of actions such as assignment, tool change, maintenance, and combined tool change with maintenance. To represent these actions, we introduce the following sets: $\mathscr{U}$ represents the set of assignment actions, $\mathscr{C}$ represents the set of tool change actions, $\mathscr{M}$ represents the set of maintenance actions, and $\mathscr{N} = \mathscr{C} \times \mathscr{M}$ represents the set of combined tool change and maintenance actions. The control input is then defined as a vector $\hat{\mathbf{u}}$ consisting of elements $\hat{u}_i \in \mathscr{U} \cup \mathscr{C} \cup \mathscr{M} \cup \mathscr{N}$ for $i = 1, 2, \ldots r$.

To assign robots to the workstations, we define the set of assignment actions as $\mathscr{U} = 1, 2, \ldots, w$, where w is the number of workstations. Each robot is characterized by its time remaining for arrival at time t, denoted by $\tau_i(t)$. Additionally, a constant time ψ is introduced to represent the time taken by a robot to move between any two workstations. To take an assignment action, the robot must be active and have already arrived at its assigned workstation.

Tool change actions are necessary to replace the worn-out tools that affect the quality of the manufactured products. There can be multiple tool wear levels, and the probability of transition between these levels depends on the current tool state. We represent the state of the i^{th} tool of robot R_j at time t by $x_j^i(t) \in 1, 2, \ldots, n^t$, where n^t is the number of possible tool states. We can use a probabilistic model to express the transition probabilities between the different tool states.

$$p(x_j^i(t)|x_j^i(t-\delta t)) \tag{3.2}$$

Experiments on the tool wear during different processes can be used to obtain this model. Changing the tool is a binary operation, i.e, you either change the tool or you don't. Thus, the set $\mathscr{C} = \{1\}$ is sufficient to model the tool change action. Upon taking this action, the tool state is reset to the least worn state.

Maintenance actions and their effects, on the other hand, are more complex to model. There are different models of maintenance effects in literature [50], but in this study, we consider the Kijima III model [51]. This model assumes a "virtual age" of the robot, which increases with time and after a certain threshold, the robot is considered broken, or no longer usable. In our case, a corrective maintenance action (CM) is automatically evoked to restore the virtual age of the robot

to 0. This can be thought of as a replacement of the robot once it is broken.

In addition to CM, we can schedule preventive maintenance actions on the robots (PM) that restore the virtual age of the robot to somewhere between 0 and the current age. The extent to which PM can restore the health of the robot is controlled using the "recovery factor", $0 \leq r < 1$, which is multiplied by the current age to get the new virtual age of the robot. This implies that a recovery factor of 0 means perfect maintenance since the age is reduced to zero, and the lower the recovery factor, the better the reduction and vice versa. We define the set of maintenance actions as $\mathscr{M} = r_k$, where $k = 1, 2, \ldots, l$, and l is the number of maintenance levels.

Finally, we can perform a combined tool change and robot maintenance, which is defined using a Cartesian product of the previous action types, i.e., $\mathscr{N} = \mathscr{C} \times \mathscr{M}$. In addition to aging, each robot R_j is subject to random disruption events which occur with an exponential distribution with a mean of $MTBF_j$.

Performing tool change or maintenance actions results in downtimes, which can affect the overall efficiency of the system. Specifically, the time required to change the i^{th} tool of any robot is represented by $Time_i^c$ for $i = 1, 2, \ldots, w$. On the other hand, the time required for Preventive Maintenance (PM) of level k on robot R_j is denoted by $Time_j^{PM,k}$, where $k = 1, 2, \ldots, l$ and $j = 1, 2, \ldots, r$. Here, l represents the total number of PM levels. Similarly, the time required for Corrective Maintenance (CM) on robot R_j is represented by $Time_j^{CM}$ for $j = 1, 2, \ldots, r$. It is worth noting that the time required for combined tool change and robot maintenance is equivalent to that of robot maintenance since both are assumed to be carried out simultaneously.

3.3.2 Model Derivation

The model derivation for the most part is similar to how it is done in Chapter 1 but since tool states are modelled discretely, the model is in discrete time. Furthermore, the addition of quality mechanisms changes some portions of the proof. Consider Eq. (3.1). For a small value of δt, forward Euler discretization can be applied to get a discrete version of the equation:

$$\mathbf{X}(t+\delta t) - \mathbf{X}(t) = \mathbf{F}\big(\mathbf{X}(t), \mathbf{Q}(t), \hat{\mathbf{u}}(t), \boldsymbol{\theta}(t)\big)\,\delta t \tag{3.3}$$

The number of robots assigned to workstation W_i at time t, $m_i(t)$ can be defined as,

$$m_i(t) = \sum_{j=1}^{r} \mathbb{1}_{u_j(t)=i} \tag{3.4}$$

where $\mathbb{1}_{u_j(t)=i}$ is a binary variable which indicates if robot j is assigned to workstation i or not.

From among all the assigned robots, the robots that are down or have not arrived can then be subtracted to find the number of working robots, i.e, $\hat{m}_i$,

$$\hat{m}_i(t) = m_i(t) - \sum_{j=1}^{r} \big(1-\theta_j(t)\big)\mathbb{1}_{u_j(t)=i} - \sum_{j=1}^{r} \big(1-\delta_k(\tau_i(t),0)\big)\mathbb{1}_{u_j(t)=i} \tag{3.5}$$

where, $\delta_k(x,y)$ is the Kronecker delta function, i.e., $\delta_k(x,y) = 1$ if $x = y$ and 0 otherwise.

The difference in accumulated *defective* production counts between workstation W_i and W_j within a time period $[0,t]$ $\forall i = 1,2,\ldots,w$ and $j = 1,2,\ldots,w$ is represented by,

$$Q_{ij}(t) = \begin{cases} \sum_{k=j}^{i} Q_k(0) - \sum_{k=j}^{i} Q_k(t) & i > j \\ \sum_{k=i}^{j} Q_k(t) - \sum_{k=i}^{j} Q_k(0) & i < j \end{cases} \tag{3.6}$$

The difference in accumulated production counts between two workstations W_i and W_j in a time period of $[0,1]$ is represented by $\mu_{ij}(t)$. Thus, using the conservation of flow and Eq. (3.6), the difference in accumulated production of *non-defective* parts between two workstations, W_i and W_j, $\forall i,j \in 1,2,\ldots w, i \neq j$ within a time period $[0,t]$ can be defined as:

$$\mu_{ij}(t) - Q_{ij}(t) = \begin{cases} \sum_{k=j+1}^{i} b_k(0) - \sum_{k=j+1}^{i} b_k(t) & i > j \\ \sum_{k=i+1}^{j} b_k(t) - \sum_{k=i+1}^{j} b_k(0) & i < j \end{cases} \tag{3.7}$$

$\mu_{ij}(t) - Q_{ij}(t)$ is bounded by an upper limit which we represent as β_{ij}. $\mu_{ij}(t) - Q_{ij}(t) = \beta_{ij}$ implies that W_i is either blocked or starved depending on its relative location from W_j. If W_j is downstream of W_i, then W_i is blocked and if W_j is upstream of W_i, W_i is starved.

$$\beta_{ij} = \begin{cases} \sum_{k=j+1}^{i} b_k(0) & i > j \\ \sum_{k=i+1}^{j} B_k - \sum_{k=i+1}^{j} b_k(0) & i < j \end{cases} \tag{3.8}$$

One important implication of this boundary is that if $\mu_{ij}(t) - Q_{ij}(t) = \beta_{i,j}$ and $\frac{\hat{m}_i}{T_i}(t) > \frac{\hat{m}_j}{T_j}(t)$, W_j will constrain the speed of W_i. Therefore, an operational workstation either works at it's own speed or at the speed of its constraining workstation. This can be mathematically represented as:

$$X_i(t+\delta t) - X_i(t) = min\Big\{\zeta\Big(\big(X_i(t) - X_j(t) - Q_{ij}(t)\big) - \beta_{ij}(t), \frac{\hat{m}_j(t)}{T_j}\Big), \frac{\hat{m}_i(t)}{T_i}\Big\}\delta t \tag{3.9}$$

where,

$$\zeta(u,v) = \begin{cases} +\infty \text{ if } u < 0 \\ v \text{ if } u = 0 \end{cases}$$

Comparing to all workstations in the system,we have,

$$\begin{aligned} X_i(t+\delta t) - X_i(t) = min \Bigg\{ & \zeta\Big(\big(X_i(t) - X_1(t) - Q_{i1}(t)\big) - \beta_{i1}(t), \tfrac{\hat{m}_1(t)}{T_1}\Big) \\ & \zeta\Big(\big(X_i(t) - X_2(t) - Q_{i2}(t)\big) - \beta_{i2}(t), \tfrac{\hat{m}_w(t)}{T_2}\Big), \\ & \qquad \vdots \\ & \tfrac{\hat{m}_i(t)}{T_i} \qquad\qquad * \delta t \\ & \qquad \vdots \\ & \zeta\Big(\big(X_i(t) - X_w(t) - Q_{ij}(t)\big) - \beta_{ij}(t), \tfrac{\hat{m}_w(t)}{T_w}\Big) \Bigg\} \\ & = f_i(X_i(t), Q_i(t), \hat{m}_i(t)) * \delta t \end{aligned} \tag{3.10}$$

$\hat{m}_i(t)$ is a function of $u_i(t)$, $\theta_i(t)$ and $\tau_i(t)$ from Eq. (3.5). Since $u_i(t)$ and $\tau_i(t)$ are dependent on $\hat{\mathbf{u}}(t)$, we can write,

$$f_i(X_i(t), Q_i(t), \hat{m}_i(t)) = f_i(X_i(t), Q_i(t), \hat{u}_i(t), \theta_i(t)) \tag{3.11}$$

Extending the idea to all workstations, we have,

$$\mathbf{X}(t+\delta t)-\mathbf{X}(t)=\begin{cases} f_1(X_1(t),Q_1(t),\hat{u}_1(t),\theta_1(t)) \\ f_2(X_2(t),Q_2(t),\hat{u}_2(t),\theta_2(t)) \\ \vdots \\ f_w(X_w(t),Q_w(t),\hat{u}_w(t),\theta_w(t)) \end{cases} * \delta t \tag{3.12}$$

$$= \mathbf{F}\big(\mathbf{X}(t),\mathbf{Q}(t),\hat{\mathbf{u}}(t),\boldsymbol{\theta}(t))\big)\delta t$$

At every workstation, the defective parts are taken out of circulation. The quality of each product is modeled as two discrete states, compliant(1) or defective(0), with a probability of transition conditioned on the state of the tool processing it. The quality of any product produced in workstation W_i by robot R_j at time t is denoted by $q_j^i(t) \in \{0,1\}$, and is given in the following probabilistic form:

$$p(q_j^i(t)|x_j^i(t)). \tag{3.13}$$

The worst tool state among all working in a workstation is used to determine quality. Consider $\hat{x}_i(t)$ such that,

$$\hat{x}_i(t) = max\big(x_j^i(t)\big)\ i=1,2,\ldots,w,\ j=1,2,\ldots,r \quad s.t\ u_j(t)=i, \tag{3.14}$$

$\hat{x}_i(t)$ represents the worst tool state among all robots working in workstation W_i at time t. Therefore, quality of products from W_i can be modeled using the following probability distribution:

$$p(q_i(t)|\hat{x}_i(t)). \tag{3.15}$$

So, we can write the total number of defective parts at time t is,

$$Q_i(t+\delta t) = Q_i(t) + \sum_{k=1}^{\lfloor X_i(t+\delta t)-X_i(t)+rem_i(t)\rfloor} \Big(1-\big[k^0 q_i \sim p\big(q_i(t)|\hat{x}_i(t)\big)\big]\Big) \tag{3.16}$$

where $\lfloor * \rfloor$ is the floor operator which disregards numbers beyond the decimal converting the value

to an integer and $\sim$ implies individual sampling from the distribution. This is done because only fully completed parts are checked for quality. The value that is excluded is the remainder and is taken into account in the next time step.

$$rem_i(t+\delta t) = X_i(t+\delta t) - X_i(t) + rem_i(t) - (\lfloor X_i(t+\delta t) - X_i(t) + rem_i(t) \rfloor) \tag{3.17}$$

The accumulated non-defective parts produced at each workstation until time t represented as $\tilde{PC}_i(t)$ can then be calculated as,

$$\tilde{PC}_i(t) = X_i(t) - Q_i(t) \tag{3.18}$$

The output of the line is the number of good parts at the end-of-the-line workstation,

$$\mathbf{Y}(t) = \tilde{PC}_w(t) \tag{3.19}$$

and the buffer levels can be updated,

$$b_{i+1}(t+\delta t) = \tilde{PC}_i(t) - X_{i+1}(t) + b_{i+1}(0) \tag{3.20}$$

Given consistent initial conditions, the model can be used to compute production counts of the system at any time recursively.

3.4 Control Problem

The system being analyzed in this chapter is also highly complex and cannot be efficiently handled by classical control or operational research methods due to its lack of a closed form representation. Therefore, an MDP formulation and learning methods are required. However, the MDP framework assumes that all agents have complete knowledge of the environment, which is often not the case in reality. Many features may be hidden or inaccessible to agents, and the current state of one agent may not be visible to another. Thus, the MDP framework is inadequate for this type of problem.

To address this limitation, the Decentralized-Partially Observable Markov Decision Process (Dec-POMDP) framework is used, which allows for decentralized control policies while accounting for partial observability of the environment. To solve the problem, a Multi-Agent Reinforcement Learning (MARL) algorithm is utilized. A brief description of the framework and algorithm used follows.

3.4.1 Dec-POMDP and MARL

The Decentralized Partially Observable Markov Decision Process (Dec-POMDP) framework is used to model multi-agent decision-making problems in partially observable environments. In Dec-POMDP, each agent only has partial observations of the environment and must use its own observations and actions to estimate the current state of the environment. Dec-POMDP can be thought of as a generalization of the POMDP framework to multi-agent problems [52]. The main challenge in Dec-POMDP is finding a decentralized control policy that maximizes the expected reward of the system over time.

The Dec-POMDP framework is based on the following components: a set of agents, a set of actions for each agent, a set of observations for each agent, a state space, a reward function, and a transition probability function. The state space is hidden from the agents, and each agent observes a noisy version of the state. The transition probability function defines the probability of transitioning from one state to another given the joint action of all agents. The reward function defines the reward for each agent as a function of the state and joint action. The goal in Dec-POMDP is to find a decentralized control policy that maximizes the expected reward over time [52].

Mathematically, a Dec-POMDP can be represented using a tuple $(\mathscr{S}, \mathscr{A}_1, \ldots, \mathscr{A}_n, P, R_1, \ldots, R_n, O_1, \ldots, O_n, \gamma)$, where $\mathscr{S}$ is the set of states, $\mathscr{A}_i$ is the set of actions available to agent i, $P(s'|s, a_1, \ldots, a_n)$ is the transition probability function, $R_i(s, a_1, \ldots, a_n)$ is the reward function for agent i, $O_i(o|s, a_1, \ldots, a_n)$ is the observation probability function for agent i, and $\gamma \in [0, 1]$ is the discount factor. The goal is to find a joint policy $\pi(a_1, \ldots, a_n|s)$ that maximizes the expected discounted sum of rewards.

Multi-Agent Reinforcement Learning (MARL) is a class of learning algorithms that can be used to solve Dec-POMDP problems. MARL algorithms can learn decentralized control policies that maximize the expected reward over time through trial and error interactions with the environment. In MARL, each agent maintains its own value function or policy, and agents may learn from each other's experiences. MARL algorithms can be divided into two categories: centralized training with decentralized execution (CTDE) and fully decentralized training and execution (FDTE) [53].

In CTDE, a centralized critic is trained using the joint experience of all agents, and each agent maintains its own actor policy that maps its observations to actions. During execution, each agent uses its own policy to select actions based on its observations. In FDTE, each agent maintains its own value function or policy and learns independently from its own experiences. The main challenge in FDTE is ensuring coordination among agents since each agent learns independently. Recent advances in MARL, such as multi-agent actor-critic methods and centralized training with decentralized execution using multi-agent reinforcement learning with communication (MARL-Comm), have shown promising results in solving large-scale Dec-POMDP problems [54, 55].

3.4.2 Problem Formulation

Since the problem is formulated as a Dec-POMDP the definitions for the observations, actions, and reward are given below. Since the algorithm to be used is a model-free MARL algorithm, transition probabilities are not needed.:

- **Observation and State** Certain variables from the model can be used to define the observation and state of the system. The observation for each robot R_i is given by:

$$\mathbf{o}_i = [u_i, d_i, \tau_i, a_i, t_i^{rep}, \hat{\mathbf{x}}_\mathbf{j}^\mathbf{i}, b_j, b_{j+1}, \hat{m}_1, \hat{m}_2, \ldots, \hat{m}_w] \tag{3.21}$$

  such that robot R_i is working in workstation W_j, i.e, $u_j(t) = i$. The state of the system is a concatenation of all robot observations, i.e.,

$$\mathbf{S} = [\mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_r] \tag{3.22}$$

- **Action:**

  Each robot can take ne of three kinds of actions as described in 3.3.1. The previous actions has to be finished for a new action to be chosen by the robot. The action space can be represented as,

$$\mathscr{A} = \{1, 2, \ldots, w, (0,1), (0,2), \ldots, (0,l), (1,0), (1,1), (1,2), \ldots, (1,l)\}, \tag{3.23}$$

  where the first w elements are assignment actions and the remainder are maintenance/tool change actions in the form: (tool change, robot maintenance) with a value of 0 implying no tool change/no robot maintenance.

- **Reward**:

  In multi-agent systems, a single global reward determines the quality of the joint actions taken by all the agents. The reward function must be designed carefully to ensure that the system behaves in the desired manner. For instance, in a mobile multi-robot operated Flexible Manufacturing System (FMS), the reward setting was discussed in [56], where it was found that a reward based on the Permanent Production Loss (PPL) was the most effective for controlling the system.

$$\mathscr{R}(t) = -PPL(t) * w_{PPL} + (\tilde{PC}_w(t) - Q_w(t))(1 - w_{PPL}). \tag{3.24}$$

  where $PPL(t)$ is the Permanent Production Loss of the system which was defined in chapter 2. In the current context, w_{PPL} represents a weighting factor that determines the relative importance of PPL over part quality. The parameter w_{PPL} is tuned to optimize the system's performance.

3.4.3 Value Decomposition Actor Critic (VDAC)

Compared to single agent reinforcement learning algorithms, multi-agent reinforcement learning (MARL) algorithms can handle large action spaces effectively. In addition, Actor-Critic MARL algorithms are well-suited for medium to large state spaces, unlike value-based algorithms such as SARSA, Q-learning, or Deep Q-learning, which are commonly used. However, because these algorithms prescribe a global reward, it is difficult to measure the contribution of each agent's actions to the system performance, which can lead to sub-optimal performance. To address this problem, the concept of difference reward was introduced [57]. The difference reward for each agent is the change in the global reward if the action of that agent is changed to a default action. The more difference reward an agent has, the more it contributes to the performance. It is important to note that there is a monotonic relationship between the agent's local rewards shaped by the difference rewards and the global reward. Unfortunately, implementing such a solution directly is challenging because it requires executing the same episode multiple times with different action settings, which quickly becomes infeasible.

However, inspired by the idea of difference rewards, the Value Decomposition Actor Critic (VDAC) algorithm was proposed [58]. VDAC is an on-policy multi-agent actor-critic algorithm that uses a Recursive Neural Network (RNN) with one fully connected neural network as the actor network and a hypernet with positive weights as the critic network. The RNN extracts sequential information from the local observation trajectory of each robot in the form of hidden states. It takes the previous hidden state and final embeddings from the HGNN as inputs and outputs a new hidden state. A SoftMax activation on the hidden state generates a multinomial stochastic policy for each individual robot. A fully connected neural network also uses this hidden state to generate a local value function.

The local value function, along with the global state of the system, is used by the critic network to compute the global value function, which is used to obtain the gradient for optimization. The critic uses a hypernet with the global state as input to generate weights and biases. These weights and biases are then used on the local value functions to compute the global value function. The weights generated by the hypernet are always positive to enforce the monotonic relationship be-

tween the local value function and the global value function. The critic network is only used for training and is disconnected when executing the learned policies. This type of learning paradigm is referred to as centralized training and decentralized execution, allowing for faster training.

The algorithm is chosen for three main reasons. Firstly, it handles the credit assignment issue in MARL well by enforcing a monotonic relationship between local and global state values. Secondly, it is suitable for problems with medium to large state spaces and action spaces. Since the problem at hand is integrated control, the state and action space is fairly large and depends on the number of workstations and robots considered. Thirdly, it allows for running parallel episodes, which helps reduce training time and effectively utilizes available computing resources. The algorithm has shown excellent results in benchmarking with Starcraft II and a maintenance scheduling problem [26].

3.5 Comparative Study between integrated and non-integrated control policies

Before comparing the effectiveness of the integrated policy against non-integrated policies, it is important to first establish what we mean by non-integrated policies. In this context, we consider two non-integrated policies, each of which only accounts for a subset of the system parameters. Specifically, one policy only conditions on the system parameters, while the other policy only considers maintenance and quality parameters. These policies are similar to controlling a manufacturing system without accounting for the interdependence of production control, maintenance scheduling, and product quality. The observation definitions for agents trained on these policies are as follows:

- System Model: System model only includes features that pertain to the system but not maintenance scheduling or product quality. The observations of each robots for this model is:

$$\mathbf{o}_i = [u_i, d_i, b_j, b_{j+1}, \hat{m}_1, \hat{m}_2, \ldots, \hat{m}_w] \tag{3.25}$$

	R_1	R_2	R_3	R_4	R_5	R_6
$Time_j^{PM,1}$	9	9	9	9	9	9
$Time_j^{PM,2}$	5	5	5	5	5	5
$Time_j^{PM,3}$	4	4	4	4	4	4
$Time_j^{CM}$	15	18	16	13	14	13
A_j	1000	800	900	1000	950	875
$MTBF_j$	200	300	400	250	200	300
$MTTR_j$	10	9	11	12	11	13

	W_1	W_2	W_3
T_i	1	0.9	1.2
$Time_i^c$	2	3	1

	B_2	B_3
B_i	6	8

Table 3.1: Parameters for Mobile multi-skilled robot operated FMS

where j is the index of the workstation the robot R_j is working in.

- Maintenance and Quality Model: This model only includes parameters that pertain to quality and maintenance but not system level parameters. The observations of each robots for this model is:

$$\mathbf{o}_i = [\tau_i, a_i, t_i^{rep}, \hat{\mathbf{x}}_\mathbf{j}^\mathbf{i}, m_j] \tag{3.26}$$

In order to compare the effectiveness of the integrated and non-integrated policies, it is crucial to establish appropriate metrics for evaluation. In this study, four metrics have been chosen for scoring each policy. These include the number of completed compliant parts, which is measured from the last workstation, as well as the total number of compliant and defective parts from all workstations. The Permanent Production Loss is also included as a metric. These metrics enable a comprehensive evaluation of the performance of the different policies.

A multi-skilled robot operated FMS is considered, consisting of $w = 3$ workstations, $r = 6$ robots, and 2 intermediate buffers. The system parameters for the manufacturing system are presented in Table 3.1. To enhance the system's robustness, the initial buffer level $b_i (i = 2, 3, \ldots w)$ is chosen randomly between 0 and B_i. Moreover, 4 tool states are taken into account, i.e., $n^t = 4$.

The transition model for the tool states is defined as follows:

$$p(x_j^i(t)|x_j^i(t-\delta t)) = \begin{bmatrix} 0.85 & 0.12 & 0.02 & 0.01 \\ 0.00 & 0.76 & 0.16 & 0.08 \\ 0.00 & 0.00 & 0.70 & 0.3 \\ 0.00 & 0.00 & 0.00 & 1.00 \end{bmatrix}$$

The product quality model considered in the experiment is as follows:

$$p(q_j^i(t) = 0|x_j^i(t)) = \begin{bmatrix} 0.01 & 0.38 & 0.67 & 0.99 \\ 0.01 & 0.50 & 0.73 & 0.99 \\ 0.01 & 0.40 & 0.59 & 0.99 \end{bmatrix}$$

The tool states for each episode are initialized randomly, and the maintenance levels considered are categorized into three: level 1 ($k = 1, r = 0$), level 2 ($k = 2, r = 0.3$), and level 3 ($k = 3, r = 0.6$). Table 3.1 presents the downtime resulting from maintenance actions of different levels. The initial age of each robot $a_j (j = 1, 2, \ldots, r)$ is randomly selected between 0 and A_j. To simulate unexpected disruptions, an exponential distribution with parameters $MTBF_j$ and $MTTR_j$ is utilized. The time required for robots to move between consecutive workstations is assumed to be $\psi = 1$.

The training process of the model involves running multiple episodes until the policy has converged. In order to expedite the training process, 20 episodes are executed in parallel. Each episode corresponds to a 10-hour workday, with a time step (δt) of 1 minute. To evaluate the effectiveness of the policy, the model is tested after every 10,000 steps (or 16.6 episodes) with 96 test episodes. The test results include several metrics such as the average and standard deviation. During training, the actor network is set to have a learning rate of 0.001, while the critic network has a learning rate of 0.0005. A batch size of 20 is utilized for the training process.

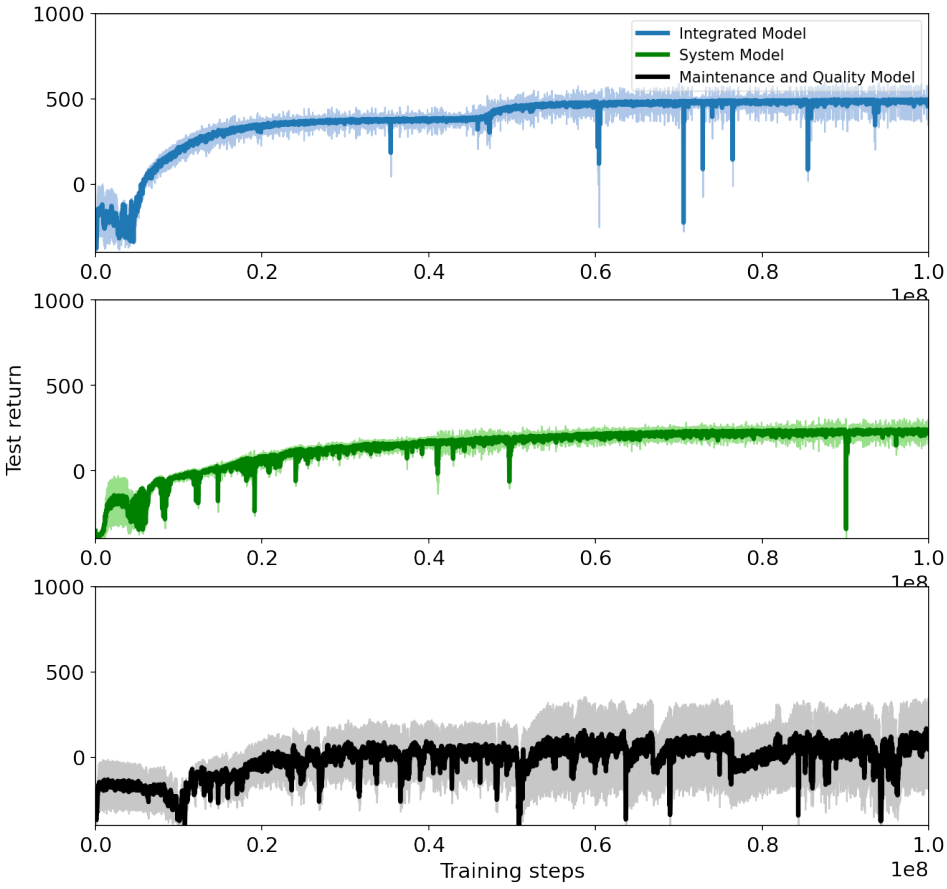

Figure 3.2: Average test return vs. training steps. *Top*: Integrated model, *Middle*: System Model, *Bottom*: Maintenance and Quality Model

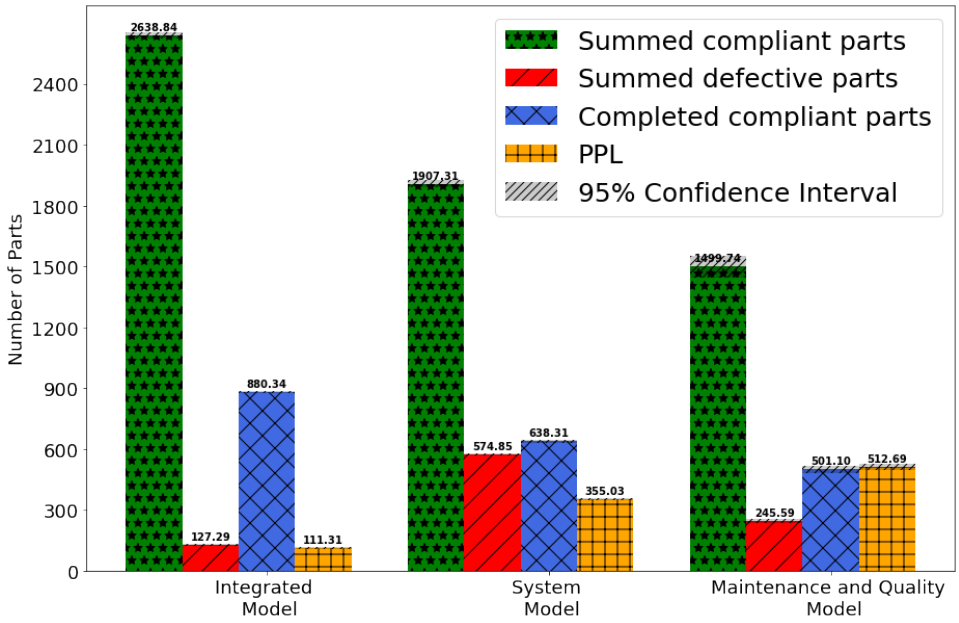

Figure 3.3: Comparison of performance metrics among different models

The convergence behavior of the integrated model and system model is illustrated in Fig. 3.2, which shows that both models converge at approximately 5×10^7 timesteps with a stable return. On the other hand, the maintenance and quality model converges at 3×10^7 timesteps, but its test return has a high standard deviation due to the exclusion of essential parameters. In addition, the results displayed in Fig. 3.3 indicate that the integrated model surpasses the other two models in terms of performance. Specifically, it achieves 880.34 compliant parts production, 2638.84 summed compliant part production, 127.29 summed defective part production, and a PPL of 111.31. This is a substantial improvement compared to the system model and maintenance and quality model, with an increase of 242.03 and 379.24 in complete compliant parts, and 731.53 and 1139.1 in summed compliant parts, respectively. Moreover, the defective part production and PPL decrease by 447.56, 118.3 and 243.72, 401.38, respectively, compared to the system model and maintenance and quality model. Statistical analysis demonstrates that the integrated model's performance increase is significant because the 95% confidence intervals of the different models do not overlap. The integrated model's success can be attributed to its inclusion of all essential features, resulting in a highly effective control strategy.

The results emphasize the importance of using a comprehensive mathematical model that con-

siders various factors that influence intricate manufacturing systems. The study also showcases that the suggested model can result in reliable training and efficient control strategies.

3.6 Summary

In summary, the chapter presented a comprehensive mathematical model for a mobile multi-skilled robot operated manufacturing system. The model integrates three essential control aspects, namely robot assignment, maintenance scheduling, and product quality control. Through a case study, the effectiveness of the control formulation based on the developed model was demonstrated. The study compared the performance of the integrated model-based control with two other control policies. One control policy was based solely on system parameters, while the other focused only on maintenance and quality parameters. The results showed that the integrated model-based control policy outperformed both of the other control policies in terms of compliant part production, defective part production, and permanent production loss.

The findings of this chapter highlight the importance of integrating production control, equipment maintenance, and quality control in modern manufacturing systems. The proposed model can be used to achieve higher productivity and efficiency in complex manufacturing systems. By considering multiple factors that impact manufacturing systems, the model can lead to stable training and effective control strategies, ultimately improving the overall performance of manufacturing systems.

Chapter 4

Concluding Remarks

4.1 Summary of Scientific Contribution

This thesis makes significant scientific contributions in the field of manufacturing system modeling and control. With the increasing advancements in robot technology, the use of cobots in manufacturing is becoming more prevalent, making it imperative for scientists to develop appropriate frameworks for analyzing such systems. Furthermore, since each manufacturing system is unique, it is challenging to generalize already developed models to new systems.

The thesis starts by developing a mathematical model for a flexible manufacturing system (FMS) that is manned by mobile robots. The model is then used to develop domain-specific real-time performance metrics that can be used for control. The thesis also demonstrates the effectiveness of a policy that incorporates such metrics, showing that it outperforms other policies that do not take these metrics into account.

After the initial model is developed, the thesis adds two other crucial aspects of manufacturing: maintenance scheduling and tool change. This creates a more holistic model that is more in tune with real-life manufacturing systems. A comparison is then made between models that only consider system-level information and models that only consider process-level information. It is seen that the integrated model significantly outperforms the rest.

Overall, this thesis has contributed significantly to the development of comprehensive models for flexible manufacturing systems that use mobile robots. By deriving domain-specific real-time performance metrics, the thesis has established an effective real-time control strategy that can be used to improve overall system performance. The addition of maintenance scheduling and tool change to the model creates a more realistic model of manufacturing systems.

4.2 Directions for Future Research

This thesis opens up several exciting directions for future research. Firstly, the model could be generalized to accurately represent robot collaborative behavior during tasks. The current models make simplifying assumptions about the relationship between workstation cycle time and the number of robots operating, which may not always be accurate. Addressing this complex problem requires a deep understanding of the tasks and their constraints, as well as an accurate representation of the robot's capabilities. Further research in this area would provide a more accurate representation of manufacturing systems and enhance their effectiveness.

Secondly, there is an opportunity use state-of-the art machine learning algorithms such as decision transformers and graph based ML algorithms to further optimize the control policies.

Finally, the scalability of the control problem needs to be studied to accommodate the addition of more robots and workstations. The current approach of retraining the entire system after each small change may not be sustainable in the long run. To address this issue, transfer learning may provide a solution. Overall, this thesis provides a solid foundation for future research in the area of flexible manufacturing systems and their control.

List of Figures

List of Tables

Bibliography

[1] Frederick S Hillier and Gerald J Lieberman. *Introduction to Operations Research*. McGraw-Hill, 2010.

[2] A Gunasekaran and EW Ngai. "The future of operations management: an outlook and analysis". In: *International Journal of Production Economics* 101.1 (2004), pp. 1–18. DOI: 10.1016/j.ijpe.2005.06.009.

[3] George Bernard Dantzig. *Linear programming and extensions*. Princeton university press, 1998.

[4] James Browne. "Flexible manufacturing systems". In: *International Journal of Production Research* 22.3 (1984), pp. 359–369. DOI: 10.1080/00207548408942594.

[5] Tianbiao Yang and Chunsheng Yang. "Flexibility in manufacturing: A survey". In: *International Journal of Production Research* 39 (2001), pp. 2685–2710. DOI: 10.1080/00207540110056906.

[6] M. M. Tseng and C. C. Tan. "Flexibility in Manufacturing: A Survey". In: *International Journal of Operations and Production Management* 18 (1998), pp. 322–351. DOI: 10.1108/01443579810199563.

[7] Y. Kim and C. Lee. "Framework to measure manufacturing flexibility". In: *International Journal of Operations and Production Management* 20 (2000), pp. 548–564. DOI: 10.1108/01443570010312662.

[8] Quanmin Zhu and Ming Zhou. “Review of the state-of-the-art of flexible production systems”. In: *Computers and Industrial Engineering* 54 (2008), pp. 1–23. DOI: 10.1016/j.cie.2007.07.010.

[9] Benoit Montreuil and Jean-Marc Frayret. “Flexible manufacturing systems: an overview”. In: *International Journal of Production Research* 34.3 (1996), pp. 537–554.

[10] ML Meena et al. “Multi-tasking in robotics—A review”. In: *Robotics and Autonomous Systems* 62.6 (2014), pp. 887–898.

[11] Martina Calzavara et al. “Walking worker vs fixed worker assembly considering the impact of components exposure on assembly time and Energy Expenditure”. In: *The International Journal of Advanced Manufacturing Technology* 112.9-10 (2021), 2971–2988. DOI: 10.1007/s00170-020-06438-9.

[12] Antonella Meneghetti and Alessandra Toninelli. “Walking workers in flexible manufacturing systems: a literature review”. In: *The International Journal of Advanced Manufacturing Technology* 26.11-12 (2005), pp. 1292–1306.

[13] Renato Cesar De Amorim and Eugenio Azevedo De Souza. “Using simulation to analyze the performance of a flexible manufacturing system”. In: *Journal of Manufacturing Systems* 32.1 (2013), pp. 160–171. DOI: 10.1016/j.jmsy.2012.05.002.

[14] Duncan Shaw. “Analysis of manufacturing system performance using simulation and visualisation techniques”. In: *The International Journal of Advanced Manufacturing Technology* 32.11-12 (2007), pp. 1199–1217.

[15] Antonio Arreola-Risa and Jose Luis Diaz-Gomez. “Production analysis of Bernoulli unreliable machines with limited buffer space”. In: *International Journal of Production Economics* 107.1 (2007), pp. 93–107. DOI: 10.1016/j.ijpe.2006.08.001.

[16] Shouren Huang et al. “Dynamic compensation robot with a new high-speed vision system for flexible manufacturing”. In: *The International Journal of Advanced Manufacturing Technology* 95.9 (2018-04), pp. 4523–4533. ISSN: 0268-3768, 1433-3015. DOI: 10.1007/

s00170-017-1491-7. URL: http://link.springer.com/10.1007/s00170-017-1491-7 (visited on 11/13/2021).

[17] Chen Li, Jing Huang, and Qing Chang. "Data-Enabled Permanent Production Loss Analysis for Serial Production Systems With Variable Cycle Time Machines". In: *IEEE Robotics and Automation Letters* 6.4 (2021), pp. 6418–6425. ISSN: 2377-3766, 2377-3774. DOI: 10.1109/LRA.2021.3093012. URL: https://ieeexplore.ieee.org/document/9466325/ (visited on 11/13/2021).

[18] Jing Zou et al. "Dynamic production system diagnosis and prognosis using model-based data-driven method". In: *Expert Systems with Applications* 80 (2017), pp. 200–209. ISSN: 09574174. DOI: 10.1016/j.eswa.2017.03.025. URL: https://linkinghub.elsevier.com/retrieve/pii/S095741741730180X (visited on 11/13/2021).

[19] Hui Zhang, Peng Jiang, and Jianxin Wang. "Collaborative robots for manufacturing: A review". In: *Engineering Applications of Artificial Intelligence* 91 (2020), p. 103578.

[20] Li Tang, Wei Liu, and Kejie Huang. "A review of mobile robot localization". In: *Journal of Field Robotics* 34.1 (2017), pp. 1–25.

[21] Chen Yu et al. "A framework for task allocation and scheduling in human-robot collaboration environment". In: *Robotics and Computer-Integrated Manufacturing* 68 (2021), p. 102046.

[22] Xinyi Luo et al. "Multi-robot system for quality inspection of manufacturing processes: A review". In: *Robotics and Computer-Integrated Manufacturing* 62 (2020), p. 101881.

[23] Jingjing Chen et al. "Deep learning-based robotic manipulation for flexible manufacturing". In: *Robotics and Computer-Integrated Manufacturing* 67 (2021), p. 101990.

[24] Zhen Li, Ying Cheng, and Yanming Li. "Intelligent robotic system for welding". In: *International Journal of Advanced Manufacturing Technology* 112.1-2 (2021), pp. 143–155.

[25] Jing Zou et al. "Data-driven modeling and real-time distributed control for energy efficient manufacturing systems". In: *Energy* 127 (2017), pp. 247–257. ISSN: 03605442. DOI:

10.1016/j.energy.2017.03.123. URL: https://linkinghub.elsevier.com/retrieve/pii/S0360544217305157 (visited on 11/13/2021).

[26] Jianyu Su et al. "Deep multi-agent reinforcement learning for multi-level preventive maintenance in manufacturing systems". In: *Expert Systems with Applications* 192 (2022), p. 116323.

[27] Jing Zou et al. "Event-Based Modeling and Analysis of Sensor Enabled Networked Manufacturing Systems". In: *IEEE Transactions on Automation Science and Engineering* 15.4 (2018), pp. 1930–1945. ISSN: 1545-5955, 1558-3783. DOI: 10.1109/TASE.2018.2861837. URL: https://ieeexplore.ieee.org/document/8437242/ (visited on 11/13/2021).

[28] Xinyan Ou, Qing Chang, and Nilanjan Chakraborty. "Simulation study on reward function of reinforcement learning in gantry work cell scheduling". In: *Journal of Manufacturing Systems* 50 (2019), pp. 1–8. ISSN: 0278-6125. DOI: https://doi.org/10.1016/j.jmsy.2018.11.005. URL: https://www.sciencedirect.com/science/article/pii/S0278612518304503.

[29] Jing Huang, Qing Chang, and Jorge Arinez. "Deep reinforcement learning based preventive maintenance policy for serial production lines". In: *Expert Systems with Applications* 160 (2020), p. 113701. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2020.113701. URL: https://www.sciencedirect.com/science/article/pii/S095741742030525X.

[30] Martin L Puterman. *Markov decision processes: Discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[31] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[32] Christopher JCH Watkins and Peter Dayan. "Q-learning". In: *Machine learning* 8.3-4 (1992), pp. 279–292.

[33] Gavin A Rummery and Mahesan Niranjan. "On-line Q-learning using connectionist systems". In: *Technical Report CUED/F-INFENG/TR 166* 4 (1994), p. 6.

[34] Hado van Hasselt, Arthur Guez, and David Silver. *Deep Reinforcement Learning with Double Q-learning*. 2015. arXiv: 1509.06461 [cs.LG].

[35] Ronald J Williams. "Simple statistical gradient-following algorithms for connectionist reinforcement learning". In: *Machine learning* 8.3-4 (1992), pp. 229–256.

[36] Vijay R Konda and John N Tsitsiklis. "Actor-critic algorithms". In: *Advances in neural information processing systems*. 2000, pp. 1008–1014.

[37] John Schulman et al. "Proximal policy optimization algorithms". In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org. 2017, pp. 3191–3199.

[38] Volodymyr Mnih et al. "Human-level control through deep reinforcement learning". In: *Nature* 518.7540 (2015), pp. 529–533.

[39] Kshitij Bhatta, Jing Huang, and Qing Chang. "Dynamic Robot Assignment for Flexible Serial Production Systems". In: *IEEE Robotics and Automation Letters* 7.3 (2022), pp. 7303–7310. DOI: 10.1109/LRA.2022.3182822.

[40] Kshitij Bhatta, Chen Li, and Qing Chang. "Production Loss Analysis in Mobile Multi-skilled Robot Operated Flexible Serial Production Systems". In: *Manufacturing Letters* 33 (2022). 50th SME North American Manufacturing Research Conference (NMRC 50,2022), pp. 835–842. ISSN: 2213-8463. DOI: https://doi.org/10.1016/j.mfglet.2022.07.103. URL: https://www.sciencedirect.com/science/article/pii/S2213846322001341.

[41] D. Mourtzis and M. Doukas. "Production control methods for flexible manufacturing systems: A review". In: *Journal of Intelligent Manufacturing* 24.6 (2013), pp. 1113–1129.

[42] R. Ben Mansour, R. Erol, and O. Battaïa. "Tool change scheduling for multi-job multi-tool systems with stochastic processing times". In: *International Journal of Production Research* 56.16 (2018), pp. 5426–5446.

[43] Harwinder Singh et al. "Comprehensive review of condition-based maintenance optimization models for manufacturing systems". In: *Journal of Quality in Maintenance Engineering* 22.2 (2016), pp. 136–172.

[44] Y. Nakamura and M. Saito. “A review on maintenance optimization methods for deteriorating systems”. In: *Journal of Quality in Maintenance Engineering* 26.3 (2020), pp. 603–629.

[45] P. S. Babu and V. R. Prasad. “Tool change and maintenance scheduling in flexible manufacturing system: A review”. In: *International Journal of Advanced Manufacturing Technology* 78.5-8 (2015), pp. 1005–1018.

[46] E. Gebennini et al. “Integration of maintenance, production planning and control in manufacturing systems: A review”. In: *Journal of Manufacturing Systems* 56 (2020), pp. 24–38.

[47] N. A. Mustaffa et al. “A framework for integrated maintenance, production planning, and control”. In: *Journal of Manufacturing Systems* 60 (2021), pp. 105–123.

[48] Nabil Abdellatif, Ashraf W Labib, and Tariq Masood. “Integrated production planning, scheduling, and maintenance: A review of the recent literature”. In: *International Journal of Production Research* 58.10 (2020), pp. 2937–2961.

[49] Xinli Li, Wenbin Li, and Changyuan Li. “Integrated production scheduling, tool change planning, and maintenance scheduling for a multi-product manufacturing system”. In: *Journal of Intelligent Manufacturing* (2021), pp. 1–17.

[50] Amir Ghaffari and Yanfeng Ouyang. “A review of maintenance models for deteriorating systems”. In: *Reliability Engineering & System Safety* 145 (2016), pp. 60–78.

[51] Masaaki Kijima. “Some results for repairable systems with general repair”. en. In: *J. Appl. Probab.* 26.1 (Mar. 1989), pp. 89–102.

[52] Frans A Oliehoek et al. “The Decentralized Partially Observable Markov Decision Process”. In: *Synthesis Lectures on Artificial Intelligence and Machine Learning* 6.6 (2012), pp. 1–117.

[53] Lucian Busoniu et al. *Comprehensive survey on multiagent reinforcement learning*. Springer, 2008.

[54] Ryan Lowe et al. “Multi-agent actor-critic for mixed cooperative-competitive environments”. In: *Neural Information Processing Systems* (2017), pp. 6379–6390.

[55] Peter Sunehag et al. “Value-Decomposition Multi-Agent Actor-Critics”. In: *arXiv preprint arXiv:1801.04087* (2018).

[56] Kshitij Bhatta and Qing Chang. “An integrated control strategy for simultaneous robot assignment, Tool Change and preventive maintenance scheduling using heterogeneous graph neural network”. In: *SSRN Electronic Journal* (2022). DOI: 10.2139/ssrn.4269017.

[57] Jakob Foerster et al. *Counterfactual Multi-Agent Policy Gradients*. 10.48550/ARXIV.1705.08926. 2017. DOI: 10.48550/ARXIV.1705.08926. URL: https://arxiv.org/abs/1705.08926.

[58] Jianyu Su, Stephen Adams, and Peter Beling. “Value-Decomposition Multi-Agent Actor-Critics”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 35.13 (2021), pp. 11352–11360. DOI: 10.1609/aaai.v35i13.17353. URL: https://ojs.aaai.org/index.php/AAAI/article/view/17353.