Adaptive Mobile Robot Scheduling in Multiproduct Flexible Manufacturing Systems Using Reinforcement Learning

Muhammad Waseem
Department of Mechanical and Aerospace Engineering,
University of Virginia,
Charlottesville, VA 22903
e-mail: kqr5pu@virginia.edu

Qing Chang[1]
Mem. ASME
Professor
Department of Mechanical and Aerospace Engineering,
University of Virginia,
Charlottesville, VA 22903
e-mail: qc9nq@virginia.edu

The integration of mobile robots in material handling in flexible manufacturing systems (FMS) is made possible by the recent advancements in Industry 4.0 and industrial artificial intelligence. However, effectively scheduling these robots in real-time remains a challenge due to the constantly changing, complex, and uncertain nature of the shop floor environment. Therefore, this paper studies the robot scheduling problem for a multiproduct FMS using a mobile robot for loading/unloading parts among machines and buffers. The problem is formulated as a Markov Decision Process, and the Q-learning algorithm is used to find an optimal policy for the robot's movements in handling different product types. The performance of the system is evaluated using a reward function based on permanent production loss and the cost of demand dissatisfaction. The proposed approach is validated through a numerical case study that compares the proposed policy to a similar policy with different reward function and the first-come-first-served policy, showing a significant improvement in production throughput of approximately 23%. [DOI: 10.1115/1.4062941]

Keywords: multiproduct production, flexible manufacturing, mobile robot scheduling, permanent production loss, demand dissatisfaction, Markov decision process, reinforcement learning, computer-integrated manufacturing, control and automation, materials handling, modeling and simulation, production systems optimization, robotics and flexible tooling

1 Introduction

The growing demand for mass customization presents a significant challenge for manufacturers, who must have robust and self-adjusting systems in order to meet these changing requirements. Flexibility is a key characteristic of modern manufacturing systems that enables them to respond quickly to changes in market demand, product design, and other issues such as machine failures [1,2]. The concept of flexibility has evolved over time, starting with the introduction of flexible manufacturing systems (FMS) in the 1980s [3]. Although there is no consensus on a precise definition of flexibility, recent literature suggests that an FMS is capable of producing a variety of products and adapting to changes in market demand, product design, and random disruptions like machine failures [4].

Conventional manufacturing lines, also known as serial production lines, consist of a series of machines that process parts in fixed paths [5]. These lines are suitable for high-volume production as each part is processed by all the machines following a fixed schedule and sequence. However, they are not flexible enough to handle variations in demand or random disturbances within the system. This lack of flexibility can make it difficult for these lines to adapt to changes in market conditions or uncertainties, such as machine failures.

Satisfying customer demands heavily relies on production scheduling and control. This paper focuses on making prompt decisions concerning production sequences and routes in response to real-time customer orders and unexpected disruptions. The challenge is that material flow is contingent on robot movements, and the robot schedule is intricately connected to system modeling, which makes analytical solutions to the robot assignment problem either difficult or impossible. Moreover, the large search space makes search-based heuristics impractical for real-time scheduling. Given the problem's stochastic and NP-hard nature, mathematical optimization techniques are unlikely to be effective under these circumstances.

This paper seeks to address the aforementioned challenges by identifying an optimal control policy for the real-time allocation of a mobile robot in a multiproduct FMS to meet market demands for different product types and minimize the overall system cost. The paper presents the following contributions:

(1) Developing a comprehensive mathematical framework for a robot-assisted multiproduct FMS, which accounts for both control input (i.e., robot assignments) and random disruptions, and identifies dynamic system properties such as permanent production loss (PPL) and the cost of unsatisfied demand.

[1]Corresponding author.

Manuscript received January 25, 2023; final manuscript received July 5, 2023; published online July 31, 2023. Assoc. Editor: Pai Zheng.

(2) Formulating the robot scheduling problem as a reinforcement learning (RL) problem and leveraging system properties, i.e., PPL and cost of unsatisfied demand, in the reward function. The problem is then solved using the Q-learning algorithm to demonstrate the effectiveness of the control scheme.

The rest of the paper is organized as follows: Sec. 2 addresses the relevant literature. Section 3 discusses the system description. Section 4 addresses dynamic system modelling and performance evaluation. In Sec. 5, the control problem is formulated, and Q-learning is discussed after which the mobile robot problem is formulated. Section 6 presents a case study to validate the proposed policy. Discussion and real-world implementation are addressed in Sec. 7. Finally, the conclusions and future work are discussed in Sec. 8.

2 Literature Review

FMS is a diverse field, and therefore, a significant amount of research is available in the literature [6] addressing its modelling, analysis, and control. The level of flexibility that an FMS can provide is contingent on the system's design and nature, and scheduling plays a critical role in its control. Recent literature has employed various approaches, including Petri-net [7], RL [8], and deep reinforcement learning (DRL) [9], among others, to manage FMS. In one study, Hu et al. [7] used a petri-net in combination with DRL and graph convolution networks (GCN) to manage a dynamic FMS. However, they only used a general FMS with shared resources. In another study, Windmann et al. [10] employed a dynamic programming-based model to address routing problems in FMS. They also attempted to generalize the model for variable parametric settings, but it may not be suitable for different environments.

In a multiproduct FMS, the output of production is contingent on robot scheduling, which is intricately linked with system modeling. This means that the system cannot be represented in a closed form [11], posing challenges for dynamic modeling of the system. Data-driven models that capture real-time information through sensors and other channels can be employed to model such a system [12,13]. While real-time information can be crucial in updating the system and meeting market demand, real-time control becomes more complex and cannot be solved via conventional optimization methods. Consequently, RL-based approaches are used to tackle such issues.

Real-time data-driven methods have also received attention in the literature, such as a method developed by [14] to estimate the effects of random machine downtime on system performance. They use opportunity window (OW), PPL, or demand satisfaction as performance metrics. OW is the longest time a machine can be down without affecting the system output, whereas PPL is defined as the unrecoverable loss due to disruptions on the production line. However, these methods generally focus on single-product systems.

While addressing a multiproduct system, Long et al. [15] compared the productivity of an aircraft assembly line using seven different predictive models and simulated three product types. Using performance evaluation and machine learning methods, they found that data-driven models may struggle due to a lack of historical data and suggested using transfer learning to improve model predictions. Similarly, Mueller et al. [16], in an attempt to satisfy small customized demands, optimized the process plans for a multiproduct system. However, their approach is only applicable to a serial production line and cannot be extended to production systems with intermittent flow paths.

Demand satisfaction/dissatisfaction is another important factor that can be used to evaluate the performance of a production system. The demand for different product types can vary, so it is important to maintain a balance in production. Therefore, Kuck et al. [17] used a forecasting model based on dynamic systems to predict the market demand. However, the forecasting models are not very robust and may not generate good results in a different environment. To address the forecasting error and improve its accuracy, Wang et al. [18] used individual and moving-range control charts to propose a mechanism for error compensation. They tested their model using a manufacturing dataset and found a significant improvement. However, it is not guaranteed that it generates similar results in other scenarios, and demand may change abruptly leading to customer dissatisfaction. Therefore, to increase customer satisfaction, Ouaret et al. [19] studied random demand and replacement policies for production lines with deteriorating machines. They used a semi-Markov decision process to determine whether machines should be replaced or repaired based on product quality and demand satisfaction. Overall, these methods improved demand satisfaction in their respective environments, but the developed models are based on strict assumptions and may not be generally applicable. Therefore, a real-time control scheme that considers market demand and system performance and can maintain a balance in the production of different product types is needed.

For real-time scheduling, a predefined policy can be utilized e.g., the first-come-first-served (FCFS) policy [20]. FCFS is a simple scheduling method, where the tasks are stored in a queue and performed in the order they are received. However, it is not effective at evaluating the efficiency of a production line. Therefore, a new scheduling policy that integrates the mobile robot and the system is needed. Scheduling problems can be solved using analytical or heuristic methods. Analytical methods aim to find an optimal solution for an objective function subject to certain constraints, but many scheduling problems, which belong to the class of NP-hard problems, are difficult to solve using this approach [21]. Heuristic methods, such as search-oriented heuristics and learning-based heuristics, can be used to address these challenges. Search-oriented heuristics find a good policy from a set of options, but the performance of these methods depends on the search space [22]. Learning-based heuristics, such as RL, have received significant attention in the literature [23,24]. It allows an agent to learn from its interactions with the environment and improve based on a set reward function, rather than following a predetermined model like in supervised learning. Q-Learning is a popular RL approach and is widely regarded as the most common RL algorithm. Many existing RL methods are based on Q-learning, which has been extensively explored and discussed in the literature [25], and its convergence has been demonstrated. Consequently, this study also employs the Q-learning approach to manage robot actions. Although it can be sometimes computationally expensive, particularly for large state-space systems, it is advantageous to analyze a new system using a well-defined method that can be used to prove the concept of the proposed framework and serve as a benchmark for comparing other approaches in the future.

In this paper, a multiproduct FMS is considered, where a mobile robot is utilized to transport materials and partially finished parts between machines in order to produce multiple different products. This system is designed to be responsive to changing market demand and resilient to disruptions such as machine failures. To fully utilize the potential of this system, intelligent scheduling and control of the mobile robot are essential. However, effectively scheduling and controlling the mobile robot in this system is a challenging task, as it requires taking into account the status of the machines and the utilization of buffers. While scheduling has been extensively studied in various contexts, such as for controlling process cycle times and performing maintenance in conventional production systems [26,27], the scheduling of mobile robots in multiproduct FMS has not been thoroughly addressed in the current literature. This is due in part to the lack of dynamic models for these complex systems, as well as the NP-hard nature of the scheduling problem and the added challenge of real-time control in such systems.

3 System Description

A multiproduct FMS with M machines, $M-1$ intermediate buffers, and a mobile robot for loading/unloading parts, is considered. As shown in Fig. 1, a machine is denoted as S_i, $i=1, 2, \ldots, M$ and is represented by a rectangle, and a buffer denoted as B_i, $i=2, 3, \ldots, M$ is represented by a circle. The production line receives

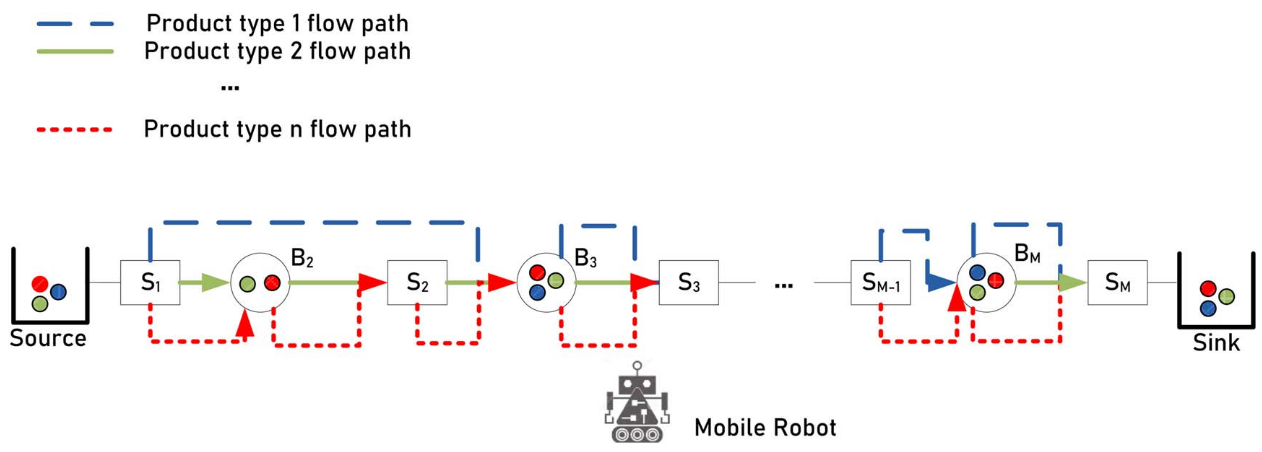

Fig. 1 A general multiproduct FMS with *n* product types, handled by a mobile robot

raw products from the source, and the finished products are stored in the sink. The small circles with different patterns and colors in the source, buffers, and sink represent different product types. Production line processes l types of products. After a machine completes its processing, the robot unloads the partially processed product to the corresponding downstream buffer based on product type and loads another part from the upstream buffer. In addition, different flowline styles between stations and buffers represent the material flows of different product types. There are n types of products shown with each flowline representing an individual product type l, $l = 1, 2, \ldots, n$.

The following notations are used in this paper:

(1) S_i, $i = 1, 2, \ldots, M$ represents the ith machine.
(2) B_i, $i = 2, 3, \ldots, M$ represents the ith buffer. With the abuse of notation, it is also denoted as the capacity of the ith buffer.
(3) $\mathcal{P} = \begin{bmatrix} P_{11} & P_{12} & \ldots & P_{1M} \\ P_{21} & P_{22} & \ldots & P_{2M} \\ \vdots & \vdots & \vdots & \vdots \\ P_{K1} & P_{K2} & \ldots & P_{KM} \end{bmatrix}$, where P_{li}, $l = 1, 2, \ldots, K$ and $i = 1, 2, \ldots, M$ denotes whether product type l is processed at the machine S_i. P_{li} has only two values: 1 or 0. A value of 1 denotes the presence of operation by machine S_i on product type l, and 0 denotes the absence of operation by machine S_i on product type l.
(4) $p_{li}(t)$ denotes the quantity of product type l in buffer B_i at time t.
(5) T_i, $i = 1, 2, \ldots, M$ denotes the processing time of the ith machine.
(6) T^i_{load} and T^i_{unload} denote the designed robot loading/unloading time per part for the ith machine.
(7) T_{travel} is the travel time of a mobile robot between two adjacent machines. $T^{ij}_{travel} = |j - i| \times T_{travel}$, $i, j = 1, 2, \ldots, M$, and $i \neq j$ represent the traveling time of a mobile robot from machine S_i to machine S_j. Machine S_i denotes the current position of the robot while the machine S_j is the final position of the robot.
(8) $\vec{b_i}(t) = [p_{1i}(t) \ldots p_{li}(t) \ldots p_{Ki}(t)]'$ is the ith buffer level at time t, representing the quantity of each product type l, $l = 1, 2, \ldots, K$.
(9) $\vec{e_i} = (j, t_i, d_i)$, denotes the ith disruption event that machine S_j is down at the time t_i for a duration of d_i.
(10) $\vec{e}_{vi} = (j, t_{vi}, d_{vi})$, denotes the ith virtual disruption event that machine S_j is waiting for the robot to load/unload at the time t_{vi} for a duration of d_{vi}.
(11) $MTBF_i$ and $MTTR_i$ represent the mean time between failure and mean time to repair of machine S_i, respectively.
(12) S_{M^*} denotes the slowest machine of the production line, while $T_{M^*} + T^{M^*}_{load} + T^{M^*}_{unload} + T^{iM^*}_{travel}$ represents the cycle time of the slowest machine.
(13) $D_l(t)$ is the market demand of product type l at time t.

The following assumptions are adopted in this paper:

(1) The term "part" and "product" in this paper are used interchangeably.
(2) The system is capable of processing l types of products. Each product type follows a unique sequence of operations, for example, $S_1 \rightarrow S_2 \rightarrow S_4$ and $S_1 \rightarrow S_3 \rightarrow S_4$ are two different product types.
(3) Each part must pass through the first and the last machine. In other words, machines S_1 and S_M are always critical.
(4) Each machine must process at least one type of product in the system.
(5) The intermediate buffer has only those product types that are to be processed by the immediate next machine.
(6) The mobile robot carries the material handling task of loading/unloading parts among machines and buffers based on the unique sequence of each product type.
(7) Each machine S_i operates at a rated speed of $1/T_i + T^i_{load} + T^i_{unload} + T^{ji}_{travel}$.
(8) A machine is said to be blocked if it is operational and its downstream buffer is full. However, a machine is said to be starved if it is operational and its upstream buffer is empty.

4 Dynamic System Modeling and Performance Evaluation

Modeling and analyzing systems are necessary for production control. The author's previous work has built a data-enabled mathematical model for multiproduct production lines [28]. To keep the paper self-contained, we briefly introduce the model and conclusion without going into technical details. Based on this model, our multiproduct FMS is operated under control inputs and random disruptions. The buffer level of each buffer is treated as a system state, and there are two sources of external disturbances: random disruption events, such as machine failures, and material shortages due to the robot's unavailability to load and unload parts to certain machines. These material shortages are referred to as "virtual disruptions," and are defined as $\vec{e}_{vi} = (j, t_{vi}, d_{vi})$, $j = 1, 2, \ldots, M$, $i \in \mathbb{Z}^+$, where S_j is the machine, t_{vi} is the time, and d_{vi} is the duration of the material shortage disturbance.

With the prior discussion, the dynamic system can be represented by the following state-space equation:

$$\dot{\boldsymbol{b}}(t) = \boldsymbol{F}(\boldsymbol{b}(t), \boldsymbol{U}(t), \boldsymbol{W}(t)) \tag{1}$$

$$\boldsymbol{Y}(t) = \boldsymbol{H}(\boldsymbol{b}(t)) \tag{2}$$

In the context of this system, the parameters are defined as $\boldsymbol{b}(t) = [\vec{b}_2(t), \vec{b}_3(t), \ldots, \vec{b}_M(t)]'$ represents the buffer levels at time t. It is a vector of all product types and $\vec{b}_i(t) = [p_{1i}(t) \ldots p_{li}(t) \ldots p_{Ki}(t)]'$.

$\boldsymbol{W}(t) = [w_1(t), w_2(t), \ldots w_M(t)]'$ represents the disturbances at time t, where $w_j(t)$ describes whether S_j suffers from a disruption at time t. If $\exists \vec{e}_k \in \boldsymbol{E}\, s.t. \vec{e}_k = (i, t_k, d_k)$ and $t \in [t_k, t_k + d_k]$, then $w_i(t) = 1$; otherwise, $w_i(t) = 0$. Define $\theta_i(t)$ as the status of a machine S_i at time t, i.e., $\theta_i(t) = 1 - w_i(t)$. A machine S_i is up at time t when $w_i(t) = 0$ and down when $w_i(t) = 1$.

$\boldsymbol{Y}(t) = [\vec{Y}_1(t), \vec{Y}_2(t), \ldots, \vec{Y}_M(t)]'$ denotes the system output at time t, where $\vec{Y}_i(t) = [p_{1i}(t) \ldots p_{li}(t) \ldots p_{Ki}(t)]'$, $i = 1, \ldots, M$, and $l = 1, 2, \ldots, K$ denotes the production count of the machine S_i for all product types up to time t.

$\boldsymbol{F}(*) = [f_1(*), f_2(*), \cdots, f_M(*)]'$, where $f_j(*)$ denotes the dynamic function of a machine S_j;

$\boldsymbol{H}(*) = [H_1(*), H_2(*), \ldots, H_M(*)]'$, where $H_j(*)$ is the observation function of a machine S_j;

$\boldsymbol{U}(t) = [u_{11}(t), u_{12}(t), \ldots, u_{1M}(t), u_{21}(t), u_{22}(t), \ldots, u_{lM}(t)]$ is the control input, where $u_{lj}(t)$ describes whether the robot is assigned to the machine S_j for loading/unloading of product type l. $u_{lj}(t) = 1$ when the robot is assigned to S_j for product type l, otherwise $u_{lj}(t) = 0$.

The system's state, i.e., buffer level, depends on the control input $U(t)$ and the machine status $\theta_i(t)$. The amount of individual product types in each buffer at time t is $\vec{\Gamma}^l_{ij}(t) = \vec{Y}_i(t) - \vec{Y}_j(t) = [\Gamma^1_{ij}(t) \ldots \Gamma^l_{ij}(t) \ldots \Gamma^K_{ij}(t)]'$, which follows the conservation of flow

$$\vec{\Gamma}^l_{ij}(t) = \begin{cases} \sum_{n=i+1}^{j} [p_{1n}(t) \ldots p_{ln}(t) \ldots p_{Kn}(t)]' - \sum_{n=i+1}^{j} [p_{1n}(0) \ldots p_{ln}(0) \ldots p_{Kn}(0)]', & i < j \\ 0 & i = j \\ \sum_{n=j+1}^{i} [p_{1n}(0) \ldots p_{ln}(0) \ldots p_{Kn}(0)]' - \sum_{n=j+1}^{i} [p_{1n}(t) \ldots p_{ln}(t) \ldots p_{Kn}(t)]', & i > j \end{cases} \quad (3)$$

where $p_{ln}(t)$ is the quantity of product type l in buffer n at time t. $\vec{\Gamma}^l_{ij}(t)$ denotes the sum of all product types l between machine S_i and S_j at time t. $\vec{\Gamma}_{ij}(t)$ has an upper limit at which a machine S_i is either starved if its upstream buffer b_i is empty or blocked if its downstream buffer b_j is full. Let this boundary be denoted as β_{ij}, we have

$$\beta_{ij} = \begin{cases} \sum_{n=i+1}^{j} B_n - \sum_{n=i+1}^{j} [p_{1n}(0) + \cdots + p_{ln}(0) + \cdots + p_{Kn}(0)], & i < j \\ 0 & i = j \\ \sum_{n=j+1}^{i} [p_{1n}(0) + \cdots + p_{ln}(0) + \cdots + p_{Kn}(0)], & i > j \end{cases} \quad (4)$$

where B_n is the buffer capacity of buffer n.

Therefore, a machine's output depends on the status of its upstream and downstream buffers. If it is not starved or blocked, it operates at its designated speed. Otherwise, it is constrained by the corresponding upstream or downstream machines.

Let $\varphi(t) = [\varphi_{ij}(t)]_{M \times M}$ be a matrix used to indicate the interactions among machines S_i and S_j at time t as follows:

$$\varphi_{ij}(t) = \begin{cases} 1, & \text{if } \Gamma_{ij}(t) = \beta_{ij}, \quad i \neq j \\ \infty, & \text{otherwise} \end{cases} \quad (5)$$

where $\varphi_{ij}(t)$ denotes if the machine S_i is starved or blocked by machine S_j. If machine S_i is constrained by machine S_j, then machine S_i must operate at the operating speed of the machine S_j

$$v_i(t) = \min\left\{\frac{\varphi_{ij}(t)\theta_j(t)}{T_j + T^{ij}_{travel}}, \frac{\theta_i(t)}{T_i + T^{ji}_{travel}}\right\} \quad (6)$$

where $v_i(t)$ is the operating speed of machine S_i at time t. Extending Eq. (6) to a whole production line, it can be represented as follows:

$$v_i(t) = \min\left\{\begin{array}{c} \dfrac{\varphi_{i1}(t)\theta_1(t)}{T_1 + T^{j1}_{travel}}, \\ \dfrac{\varphi_{i2}(t)\theta_2(t)}{T_2 + T^{j2}_{travel}}, \\ \vdots \\ \dfrac{\theta_i(t)}{T_i + T^{ji}_{travel}}, \\ \vdots \\ \dfrac{\varphi_{iM}(t)\theta_M(t)}{T_M + T^{jM}_{travel}} \end{array}\right\} \quad (7)$$

The change rate of $\vec{b}_i(t)$ is the speed difference between S_i and S_{i-1}.

$$\dot{\vec{b}}_i(t) = \{[v_i(t)][P_{1i} \ldots P_{li} \ldots P_{Ki}]' - [v_{i-1}(t)] [P_{1(i-1)} \ldots P_{l(i-1)} \ldots P_{K(i-1)}]'\} \odot [P_{1i} \ldots P_{li} \ldots P_{Ki}]' \quad (8)$$

$$= F_i(\boldsymbol{b}(t), \boldsymbol{W}(t))$$

where $\odot$ represents the Hadamard product, and P_{li}, $i = 1, 2, \ldots, M$, $l = 1, 2, \ldots, K$ denotes the relation of machine S_i and product type l. It is 1 if a machine S_i processes product type l otherwise 0.

The output of the system at time t is given as follows:

$$\boldsymbol{Y}(t) = \vec{Y}_M(t) = [p_{1M}(t) \ldots p_{lM}(t) \ldots p_{KM}(t)] = H_i(\boldsymbol{b}(t)) \quad (9)$$

4.1 Permanent Production Loss Evaluation.

To solve the problem of robot scheduling, the system needs to be completely understood. The authors will extend and adapt their earlier work on OW and PPL [14]. To keep this paper self-contained, only a summary of the basic concepts will be provided without including detailed proof. The PPL of a system can only be evaluated when it is subject to disruptions. The concept of OW is used to measure the status of a system and its resilience to disruption events [14,29].

OW of a machine S_j, denoted as $OW_j(T_d)$, is the longest possible downtime that can be taken on the machine S_j at time T_d without causing PPL at the end-of-line machine S_M. It can be defined as

$$OW_j(T_d) = \sup\left\{d \geq 0 : s.t.\ \exists T^*(d), \int_0^T S_M(t)dt = \int_0^T \tilde{S}_M(t; \vec{e})dt,\ \forall T \geq T^*(d)\right\} \quad (10)$$

where $\int_0^T S_M(t;\vec{e})dt$ and $\int_0^T S_M(t)dt$ represent the production count of the last machine S_M with and without disruption event $\vec{e}=(M, T_d, d)$, respectively.

Based on our previous studies, the OW of a machine S_j along a serial production line with finite buffers is defined as the time it takes for the buffers between machine S_j and the slowest machine S_{M^*} to become empty ($j<M^*$) or full ($j>M^*$) when S_j is forced down.

It has also been proven that the system will experience PPL only if the OW of a machine is exceeded by the duration of the disruption event on the machine. This kind of PPL resulting from an extended disturbance is also permanent for all other machines on the line. If a disruption event $\vec{e_i}=(j, t_i, d_i,)$ lasts longer than its corresponding opportunity window, $OW_j(t)$, then for any machine S_j, there exists $T^* \geq t+d$, depending on the relative position of the machine S_j to the slowest machine, S_{m^*} such that:

$$\int_0^T S_j(t')dt' - \int_0^T S_j(t', \vec{e})dt' = \frac{(d-OW_j(t))}{T_{M^*}}, \quad \forall T \geq T^* \tag{11}$$

In this paper, the concepts of OW and PPL are extended to multiproduct flexible production lines. The PPL is utilized as a quantitative measure of the impact of both real and virtual disruption events on the system. Real disruptions are disruptions that occur randomly, such as machine failures, while virtual disruptions are artificially created, such as elongated waiting times for a machine due to the assignment of a robot ($\boldsymbol{U}(t)$). The goal of the system is to control the assignment of the robot ($\boldsymbol{U}(t)$) in order to minimize the impact of these disruptions on the system, as measured by PPL.

The amount of PPL is evaluated by taking the difference between the actual output of a production system (real production) and the output of an ideal production system, which is a theoretical system that does not experience any disruptions (either real or virtual) and always has a robot available to load and unload parts when needed. Let the output of an ideal production system be denoted as $Y_{ideal}(T)$ and real output of the system be denoted as $Y(T)$, the amount of PPL during a time period $[0, T]$ can be defined as $PPL(T)=Y_{ideal}(T)-Y(T)$. It is noted that the output of the last machine is used to represent the overall system output for both real system and ideal production system.

Now to derive the PPL, a critical machine is defined. A machine S_i is critical if it processes all product types of the system. For example, if a machine S_i is critical, then $\begin{bmatrix}P_{1i} & P_{2i} & \dots & P_{Ki}\end{bmatrix}'=\begin{bmatrix}1 & 1 & \dots & 1\end{bmatrix}'$. The ith critical machine is denoted as S_{ic}, whereas $(T_{ic}+T_{load}^{ic}+T_{unload}^{ic}+T_{travel}^{j,ic})$ represents the cycle time of the ith critical machine. Similarly, the slowest critical machine of the production line is referred to as S_{sc}, and its cycle time is the longest among all critical machines, represented by $(T_{sc}+T_{load}^{sc}+T_{unload}^{sc}+T_{travel}^{j,sc})$.

Based on our previous research [14], the system will experience PPL only if the OW of the slowest critical machine S_{sc} is exceeded by any disruption event. Therefore, the PPL during a time $[0, T]$ depends on the status of S_{sc} and can be defined as

$$PPL(T)=\frac{D_{sc}(T)}{T_r^{sc}+T_{load}^{sc}+T_{unload}^{sc}+T_{travel}^{j,sc}} \tag{12}$$

S_{sc} may stop due to random disruption events or virtual disruptions, such as the unavailability of the robot or starvation/blockage caused by random disruptions on other critical machines. As discussed earlier [30], virtual disruptions are dependent on the system and are considered an endogenous factor of the system. These disruptions can be controlled by optimizing the scheduling policy of the mobile robot. On the other hand, a random disturbance is considered an exogenous factor of the system, which is caused by uncontrollable random disruption events. Therefore, this paper focuses on endogenous factors of the system (i.e., robot scheduling) and uses RL Q-learning to develop an optimum policy.

Besides, to estimate the impact of individual machine downtime on the whole production line, our previous work [28] discussed sensitivity analysis based on our performance metric, i.e., PPL attribution to each machine along a multiproduct FMS.

4.2 Demand Dissatisfaction. Demand $D_l(t)$ is the customer order for a product type l at time t. Besides PPL, demand is another important factor to analyze system behavior. A demand $D_l(t)$ is satisfied if the market's order is fulfilled within the required time $[0, T]$, otherwise dissatisfied. The market demand for each product type l varies over time, making it challenging to satisfy the demand for each product within due time, as the system may experience disruptions.

As previously mentioned, PPL is caused by disruptions on the critical machine S_{ic}. Therefore, if there is an extended disruption event on a non-critical machine S_i, there may be dissatisfaction with the demand for the product type l that passes through the disrupted machine S_i. This paper considers demand dissatisfaction to be equivalent to permanent production loss.

A demand $D_l(t)$ is satisfied for a product type l during the time $[0, t]$ if the actual production of product type l, i.e., $Y_l(t)$, is not exceeded by the market demand $D_l(t)$. Otherwise, there is a demand dissatisfaction for the product type l, and the associated cost, referred to as delay cost, can be evaluated. On the other hand, the cost for the excess production, referred to as the cost of surfeit, can be also calculated. Together, these costs can be written as

$$CT=\sum_{l=1}^{K}\omega_d^l \max\{0, D_l(t)-Y_l(t)\}+\sum_{l=1}^{K}\omega_s^l \max\{0, Y_l(t)-D_l(t)\} \tag{13}$$

where ω_d^l and ω_s^l are the delay cost rate and the surfeit cost rate for product type l, respectively.

5 Control Problem Formulation and Solution

The control input $\boldsymbol{U}(t)$ determines where the robot should be assigned and what product it should load or unload. $\boldsymbol{U}(t)$ needs to be updated based on the current state of the system. However, $\boldsymbol{U}(t)$ also affects the system model because different product types follow different paths, leading to changes in the system's state. As a result, $\boldsymbol{U}(t)$ is linked to the system's state $\boldsymbol{b}(t)$, which complicates the control design. Furthermore, it is not possible to use classic control methods because there is no general closed-form representation of the system.

To address this issue, this paper proposes a learning-based control scheme. Since the control problem involves a robot making sequential decisions about which machine to assign to and what product to handle, it is suitable to formulate it using a Markov Decision Process (MDP). An MDP is based on a tuple $\langle S, A, P, R, \gamma\rangle$ where S denotes the agent's Markov states, A denotes a set of actions the agent can take in the state $s \epsilon S$, P is the transition probability from state s to s' given action a is taken, R is the reward, the agent receives after transitioning from s to s' by taking an action a, and $\gamma\epsilon[0, 1]$ is the discount factor. Since previous research has demonstrated that model-free methods are effective in solving such problems [27,31], a model-free RL-based control scheme is developed in this paper.

5.1 Problem Formulation. The control input $\boldsymbol{U}(t)$ aims to minimize the cumulative production loss PPL and total cost CT by dynamically scheduling the robot to meet the real-time demand for each product type. An RL framework is used to model this problem. The agent (i.e., the mobile robot) takes an action a_t at time t, which in this case is the loading/unloading of a product type l at machine S_i. This action causes a transition from state s_t to s_{t+1}, and the agent receives a reward r_{t+1} for taking the action.

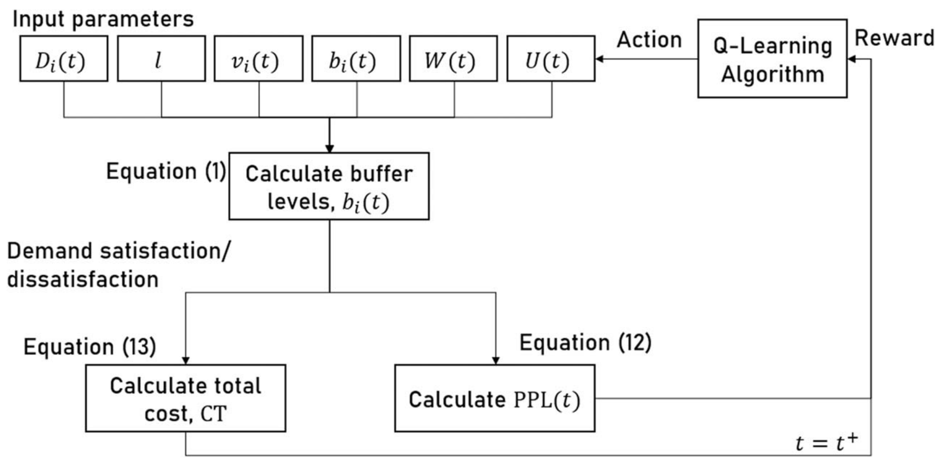

Fig. 2 A complete working structure of multiproduct FMS controlled by Q-learning algorithm

To meet the demand for each product type l and minimize the PPL over a period, an optimal policy π^* must be defined.

$$\pi^* = \arg\min_{\pi} (PPL + CT) \tag{14}$$

Thus, the robot scheduling problem can be formulated as: Given the current system state $s_t \in S$, find a feasible control action $a_t \in A$ that minimizes PPL (i.e., maximizes the production output) while satisfying the market demand for each product type.

The states, actions, and rewards for this problem are discussed below. The transition probability is not addressed because a model-free RL approach is used.

States

The system state consists of the buffer status (i.e., the buffer levels for each product type at each time t), the machine's status (i.e., up, or down), and the remaining process time for each machine for the current part. Therefore, the system state at time t, $s_t \in S$ can be represented as

$$s_t = \{\vec{b}_1, \vec{b}_2, \ldots, \vec{b}_{M-1}, \theta_1, \theta_2, \ldots, \theta_M, t_{r1}, t_{r2}, \ldots, t_{rM}\} \tag{15}$$

where $\vec{b}_i$, $i = 1, 2, \ldots, M-1$ denotes the buffer level for each product type; $\theta_i = 1$ if S_i is up otherwise 0; t_{rj} denotes the amount of remaining process time for the machine S_j to finish the loaded part.

Actions

The robot needs to be assigned to a machine S_j, $j = 1, 2, \ldots, M$ for a product type l. Define A as a set of robot's actions and $a_t \in A$. Then, the actions can be represented as

$$a_t = [a_{11}, a_{12}, \ldots, a_{1M}, a_{21}, a_{22} \ldots, a_{lM}] \tag{16}$$

where a_t represents the robot assignment at time t, and a_{lj} denotes the assignment of the robot to the machine S_j for loading/unloading product type l.

Reward

The reward function for the robot's action is calculated as a linear combination of the PPL and the net cost of demand dissatisfaction (CT) as

$$r = -PPL - CT \tag{17}$$

The robot is trying to maximize production of a specific product type (l) in order to avoid the cost of delay (CoD), as dictated by market demand ($D_l(t)$). However, the robot must also take into account the cost of surfeit (CoS), which is the cost associated with consistently assigning similar product types that are not in high demand by customers. This balance between CoD and CoS allows the robot not only to increase total production but also to focus on assigning the required product types to the machines. This reward function encourages the robot to take actions that minimize the PPL and CT.

5.2 Solve the Problem Using Q-Learning.

The solution to an MDP problem is based on a policy π, where the agent tries to maximize its reward by taking an optimal action in each state s. Q-Learning can be used to optimize the action-selection policy for any given MDP. It is a model-free algorithm, meaning that the agent relies on estimates of the expected response of the environment rather than following a model of the environment. The Q-learning algorithm is based on a simple value-iteration update and uses the following equation:

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha\left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)\right] \tag{18}$$

where γ is the discount factor ($0 \le \gamma \le 1$) used to balance immediate and future rewards. The learning rate α ($0 < \alpha \le 1$) determines how much the new value is accepted relative to the old value. The ε-greedy algorithm ($0 \le \varepsilon \le 1$) is used to balance exploration and exploitation, ensuring that the agent visits all state-action pairs.

The Q-table is regularly updated, leading to the approximation of optimal action-value functions Q^*. After Q converges to Q^* and the agent has learned completely, the optimal policy π^* can be directly constructed from Q^*

$$\pi^*(s) = \arg\max_a Q^*(s, a) \tag{19}$$

In Fig. 2, the flow structure of the multiproduct system is depicted. The model inputs include the buffer level $\vec{b}_i(t)$, random disruptions $w(t)$, and control action $u(t)$. The updated buffer levels and PPL are calculated using Eqs. (1) and (12), respectively. The market demand is compared to the sink, and CT is calculated using Eq. (13). The calculated PPL and CT are used to estimate the reward, and the Q-learning algorithm generates a new action which is applied to the system model.

6 Case Study

To demonstrate the effectiveness of the proposed method for solving the robot assignment problem, extensive numerical studies were conducted. The study involves evaluating 20 distinct system configurations, and for each configuration, three policies are compared. The first policy, labeled RL-PPLCT, is based on the proposed method and employs the reward function as defined in Eq. (17), i.e., $r = -PPL - CT$. The second policy, labeled RL-PPL, employs RL but with a different reward setting, $r = -PPL$. The third policy, labeled First-come-first-serve (FCFS), is a simple scheduling policy commonly found in many real-world applications.

To evaluate the performance of each policy, three metrics are utilized: (1) production count, (2) demand satisfaction/dissatisfaction, and (3) estimated profit. Based on the results of the numerical case study, it can be concluded that the proposed RL-PPLCT policy outperformed the other two policies in all three metrics, with statistical significance.

6.1 Test Environment. Twenty different system configurations are constructed by selecting randomly and equiprobably from the following sets:

$$\textit{No. of machines}, M \in \{3, 4, \ldots, 10\}$$

$$\textit{No. of robots}, R \in \{1, 2, 3, 4, 5, 6\}$$

$$\textit{Processing time}, T_i \in [50, 90]\textit{sec}, i = 1, 2, \ldots, M$$

$$\textit{Loading time}, T^i_{load} \in [5, 25]\textit{sec}, i = 1, 2, \ldots, M$$

$$\textit{Unloading time}, T^i_{unload} \in [5, 25]\textit{sec}, i = 1, 2, \ldots, M$$

$$\textit{MTBF}_i \in [10, 25]\textit{mins}, i = 1, 2, \ldots, M$$

$$\textit{MTTR}_i \in [2, 6]\textit{mins}, i = 1, 2, \ldots, M$$

$$\textit{Buffer capacity}, B_i \in [10, 50], i = 1, 2, \ldots, M - 1$$

$$\textit{No. of product types}, l \in \{2, 3, 4, 5, 6\}$$

$$\textit{Weekly demand}, D_l \in [50, 250], l = 1, 2, \ldots, K$$

All intervals presented an increase with an integer value of 1. Random disruptions are generated assuming a geometric distribution using mean time between failures ($MTBF_i$) and mean time to repair ($MTTR_i$).

For this case study, the system parameters and data are collected from a refrigerator and motor-bike manufacturing lines. To maintain confidentiality, the mockup system parameters are shown in Tables 1–6.

To provide further details on the method and performance evaluation, two system configurations are presented in this paper. Configuration-1 consists of four machines, three buffers, three product types, and one robot, as shown in Fig. 3. Configuration-2 consists of five machines, four buffers, three product types, and one robot, as shown in Fig. 4. The robot assignment policy is trained once, and then, the trained policy based on the proposed method is evaluated through online execution and compared with RL-PPL and FCFS for both configurations. For each configuration, the production count, demand dissatisfaction, and estimated profit under the proposed policy are compared with those of RL-PPL and FCFS.

6.2 Offline Training of the Mobile Robot. The mobile robot agent is trained for 20 episodes with each episode representing 10 h a day, 5 days a week. As shown in Fig. 5, it takes almost 800 h for both RL-PPLCT and RL-PPL policies to get stabilized. However, the overall reward is higher under the proposed RL-PPLCT policy. The parameters used during the training are the following: ε-greedy set to 0.2, the discount factor γ to 0.95, and the learning rate α to 0.1. The $Q(s, a)$ table is initialized as zero and is continuously updated with each episode.

6.3 Online Execution of the Proposed Policy. Once trained, the policies from RL-PPLCT and RL-PPL are executed on the two system configurations mentioned above, and three performance metrics are utilized to compare the three policies: RL-PPLCT, RL-PPL, and FCFS.

6.3.1 Configuration-1. In Configuration-1, Figs. 6(*a*)–6(*c*) compare the production accumulation of product types 1, 2, and 3, respectively, under the proposed RL-PPLCT and FCFS for a three-week duration. Figure 6(*a*) shows an average difference of 50 units per week between both policies, with production under the proposed method being less. It takes more time for the proposed

Table 1 Parameters for the machines (Configuration-1)

Parameters	S_1	S_2	S_3	S_4
Processing time, T_r (s)	60	55	70	65
Loading time, T_l (s)	15	15	15	15
Unloading time, T_u (s)	10	10	10	10
MTBF (min)	16	14	22	20
MTTR (min)	5	3	4	4

Table 2 Parameters for the machines (Configuration-2)

Parameters	S_1	S_2	S_3	S_4	S_5
Processing time, T_r (s)	70	50	90	60	55
Loading time, T_l (s)	10	10	10	10	10
Unloading time, T_u (s)	10	10	10	10	10
MTBF (min)	19	16	25	18	20
MTTR (min)	3	4	2	4	3

Table 3 Parameters for the three buffers (Configuration-1)

Parameters	B_1	B_2	B_3
Buffer capacity	25	25	25
Initial buffer level	0	0	0

Table 4 Parameters for the four buffers (Configuration-2)

Parameters	B_1	B_2	B_3	B_4
Buffer capacity	20	20	20	20
Initial buffer level	0	0	0	0

Table 5 Parameters for the three product types (Configuration-1)

Parameters	Product type 1	Product type 2	Product type 3
Week 1 demand	200	200	200
Week 2 demand	250	250	250
Week 3 demand	180	180	180
Profit ($/part)	2	2	2

Table 6 Parameters for the three product types (Configuration-2)

Parameters	Product type 1	Product type 2	Product type 3
Weekly demand	70	95	155
Profit ($/part)	2	2	2

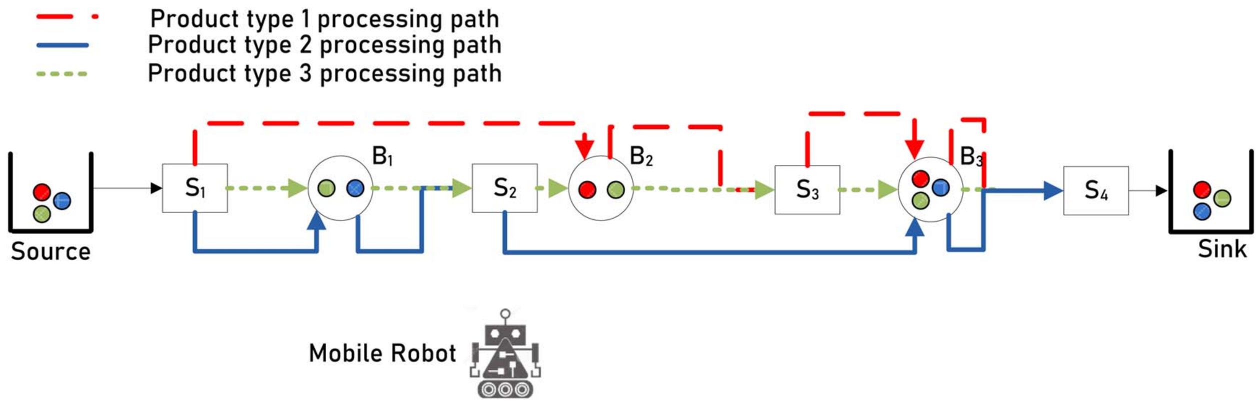
Product type 1 processing path
Product type 2 processing path
Product type 3 processing path
Source
S₁
B₁
S₂
B₂
S₃
B₃
S₄
Sink
Mobile Robot

Fig. 3 Configuration-1: A multiproduct FMS with four machines, three buffers, and three product types handled by a mobile robot

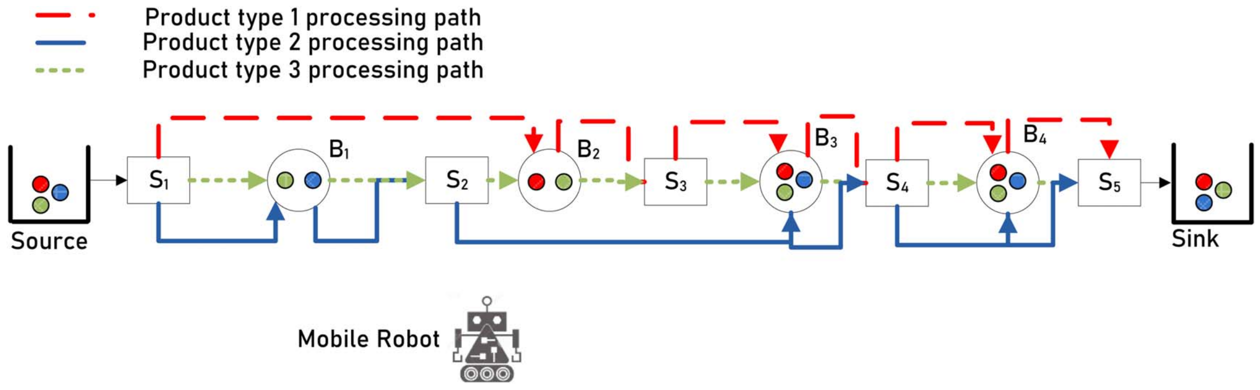
Product type 1 processing path
Product type 2 processing path
Product type 3 processing path
Source
S₁
B₁
S₂
B₂
S₃
B₃
S₄
B₄
S₅
Sink
Mobile Robot

Fig. 4 Configuration-2: A multiproduct FMS with five machines, four buffers, and three product types handled by a mobile robot

RL-PPLCT policy to satisfy the market demand for product type 1 compared to FCFS, due to the reward setting that restricts the robot from continuously assigning product type 1. On the other hand, production for product types 2 and 3 is increased under the proposed policy, with an average increase of 70 and 40 units per week, respectively. In Fig. 6(*b*), FCFS policy fails to satisfy the market demand for product type 2 in the second and third weeks, while RL-PPLCT policy catches up with the market demand quickly. Similarly, in Fig. 6(*c*), although both policies satisfy the market demand, the proposed policy significantly reduces the demand satisfaction time.

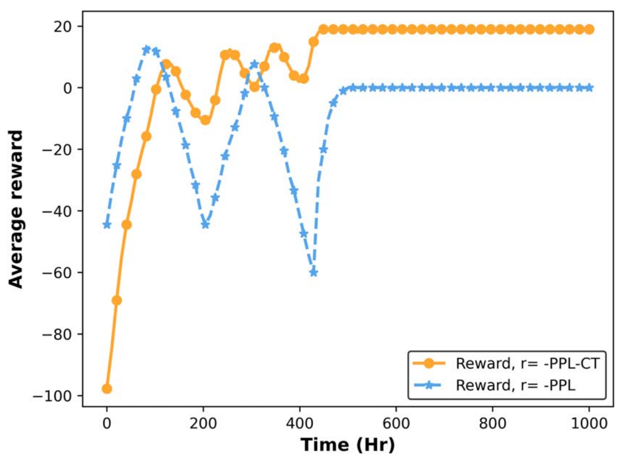

Fig. 5 Training data, reward versus time

Figure 7 presents a comparison of net profit under FCFS, RL-PPL, and RL-PPLCT for individual product types and cumulative production. The dotted bar indicates the profit under the FCFS policy, whereas the bars with line and circle patterns represent the profit under RL-PPL and RL-PPLCT, respectively. While considering the corresponding market demand of each product type, the proposed method reduces the production of product type 1 but increases the production of product types 2 and 3, resulting in an increase in net profit of $408 and $360, respectively. The average net profit for total production increases by $177 under the proposed method, demonstrating its effectiveness. The 95% confidence interval also indicates a significant difference in net profits.

Figure 8 illustrates a comparison of the overall production count for one week among the proposed RL-PPLCT, RL-PPL, and FCFS policies for Configuration-1. The results show that the proposed policy outperforms the FCFS policy by increasing the production for each product type, with an average increase of 50 units for product type 1, 32 units for product type 2, and almost 46 units for product type 3 over the course of 20 episodes, which represent one week. The 95% confidence intervals for the RL-PPLCT did not overlap with those of the FCFS policy, indicating that the RL-PPLCT policy performs better than the FCFS policy. Overall, production is increased by almost 128 units in a single week, showing the effectiveness of the proposed policy.

In summary, the proposed policy demonstrated a statistically significant increase in performance compared to FCFS and RL-PPL, as evidenced by higher net profit, production, and demand satisfaction. Although there was a slight reduction in the production of product type 1, the proposed policy achieved a more balanced distribution of production across all product types, resulting in better overall performance. The results of the case study highlight the effectiveness of the RL-PPLCT policy in improving manufacturing system

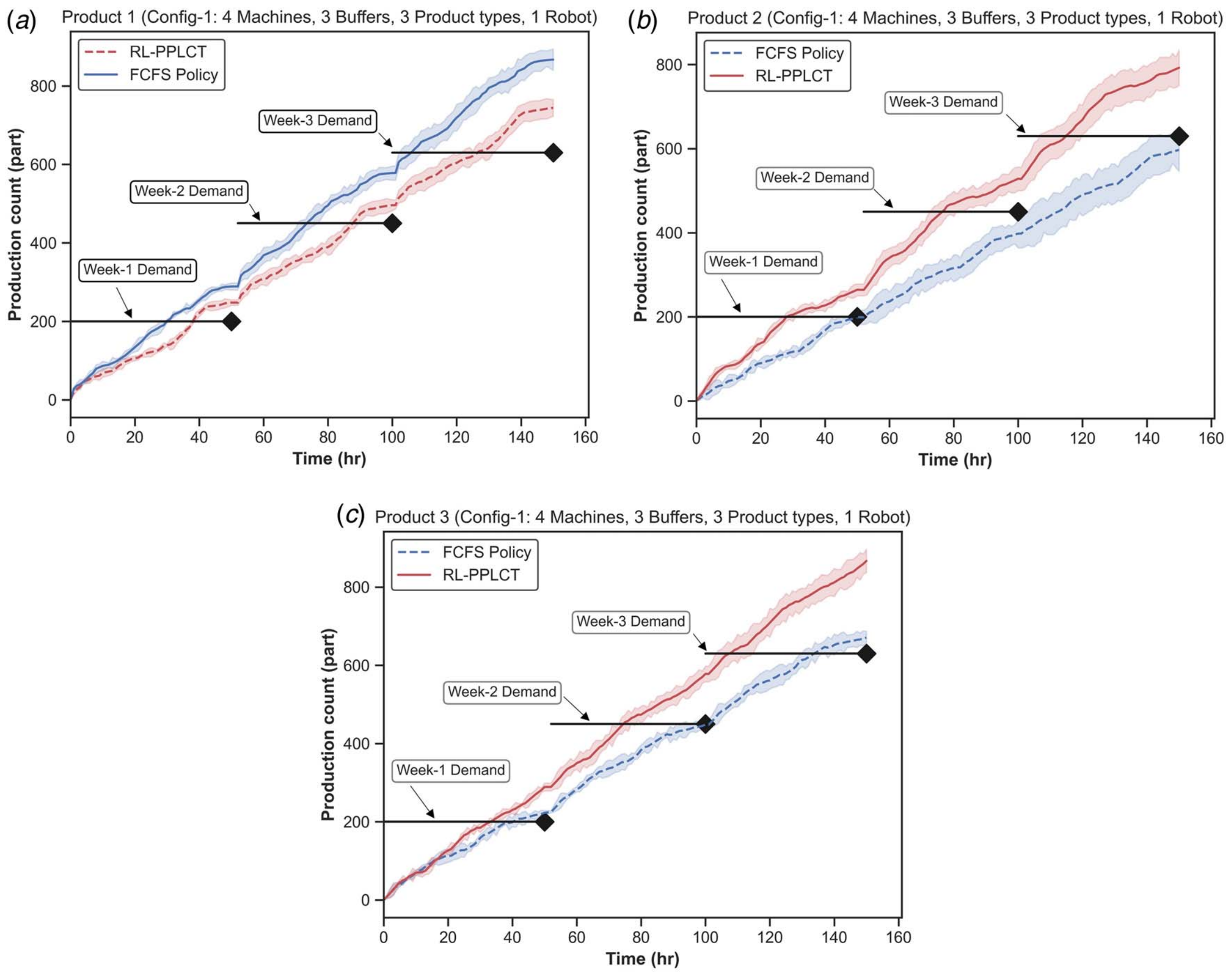

Fig. 6 Comparison of production trajectory under FCFS and RL-PPLCT for (*a*) Product type 1, (*b*) Product type 2, and (c) Product type 3

performance and its potential application in other manufacturing systems.

6.3.2 Configuration-2. For Configuration-2, we only compare the net production count and estimated profit for one week. The production trajectories over time are not shown in this case and the discussion is not very detailed, as it follows the same concept as Configuration-1.

Figure 9 illustrates a comparison of the overall production count for one week among the proposed RL-PPLCT, RL-PPL, and FCFS

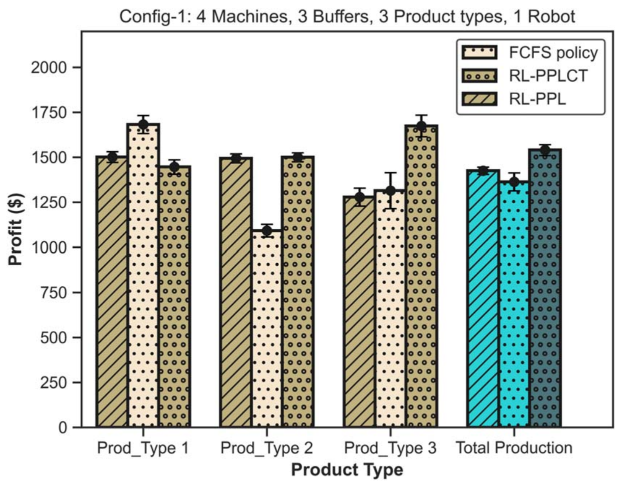

Fig. 7 Comparison of estimated profit under FCFS, RL-PPL, and RL-PPLCT for configuration-1

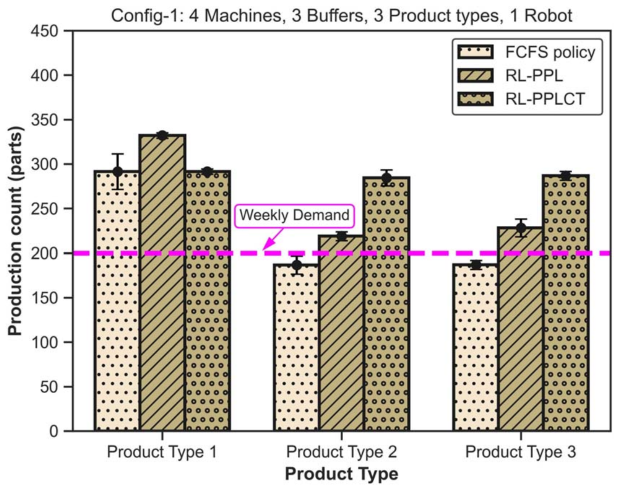

Fig. 8 Comparison of net production under FCFS, RL-PPL, and RL-PPLCT for configuration-1

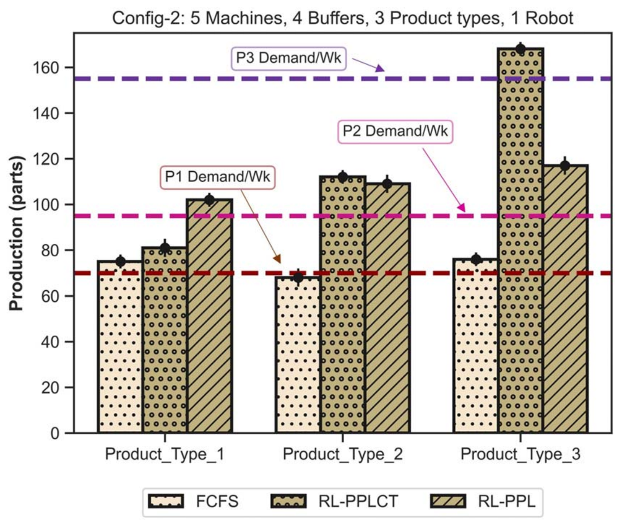

Fig. 9 Comparison of net production under FCFS, RL-PPL, and RL-PPLCT for configuration-2

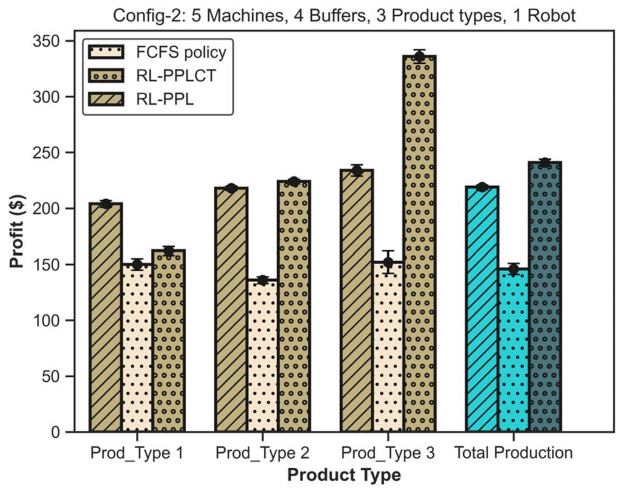

Fig. 10 Comparison of estimated profit under FCFS, RL-PPL, and RL-PPLCT for configuration-2

policies for Configuration-2. It can be observed that only the proposed policy is able to meet the market demand on a weekly basis. This is due to the reward function, which takes into account both PPL and CT. As a result, the policy is adjusted to ensure that the slowest critical machine S_{sc} is not stopped and the market demand is met. In contrast, the RL-PPL policy does not meet the corresponding market demand, as it only considers PPL and ensures that S_{sc} is not stopped without updating based on the specific market demand of individual product types.

In Fig. 10, the net profit for individual product types and total production is compared among the proposed RL-PPLCT, RL-PPL, and FCFS for Configuration-2. The proposed policy results in the highest net profit of almost \$241, compared to RL-PPL with \$219 and FCFS with \$146. The proposed RL-PPLCT adjusts the production according to the market demand and produces a larger quantity of product type 3 compared to product types 1 and 2, resulting in demand satisfaction. Therefore, the proposed policy outperforms the RL-PPL and FCFS policies.

7 Discussion and Real-World Implementation

The case study presented in this paper demonstrates the effectiveness of using reinforcement learning (RL)-based control policy for controlling multiproduct FMS in realistic settings. Apart from the reported results, the study offers valuable insights for further research and practical implementation. First, incorporating domain knowledge into reinforcement learning is crucial. The study reveals that the reward function solely based on the production maximization, i.e., reducing PPL, does not provide significant results, as it only ensures to keep the machines processing products irrespective of their types. The performance metrics like market demand along with PPL can enhance the system's performance. Therefore, it is imperative for the academicians and practitioners to gain a deep understanding of the domain and identify relevant performance metrics that can be integrated into the reward function.

Second, the algorithm used in this study is Q-learning, which provides good results when the state-space is not too huge. However, in real-world scenarios, it depends on the environment where the method is applied. Therefore, the state-space and action space need to be analyzed before using this approach. In the future, complicated real-world problems having large state space, advanced algorithms, e.g., policy gradient, Deep Neural Networks (DNN), etc., will be explored.

Finally, since the proposed method takes real-time information from the environment, the output is dependent on the input data. Therefore, it is recommended that the data must be pre-processed and removed any noise available before feeding to the model, as the real-time data are collected through different sensors and there is always a high probability to have some noise in the data.

8 Conclusion and Future Work

This paper studies a multiproduct flexible manufacturing system that utilizes a mobile robot for handling the parts among machines and buffers. A data-enabled model is developed and using PPL and demand dissatisfaction as performance metrics, a fast recursive algorithm calculates the system's real-time state. The challenge of real-time control of the robot in this system, which is impacted by the modeling of the system and the scheduling of the robot, is addressed by formulating it as a reinforcement learning (RL) problem with a reward function based on PPL and the cost of demand dissatisfaction. A case study is used to demonstrate the effectiveness of the learned policy using the Q-learning algorithm. The Q-learning algorithm is used to learn a policy under two different reward settings, and the resulting policies are compared with a first-come, first-served (FCFS) control policy. Simulation results show that the policy learned with the cumulative reward function outperforms the FCFS policy and the policy learned with the reward function considering only PPL. The proposed policy increases overall production throughput by approximately 23% per week while satisfying the market demand for individual product types. Therefore, RL is found to be effective in reducing real-time production loss in a multiproduct FMS while meeting market demand.

In this study, a simple production line was used to demonstrate the proposed method. As such, a straightforward Q-learning approach was sufficient to address the robot control problem. However, in more complex environments that feature a larger number of machines, buffers, or product types, state-of-the-art algorithms such as deep neural networks (DNNs) and policy gradient algorithms may be more appropriate. Therefore, it would be valuable to investigate the use of these and other RL algorithms in future research and compare their performance.

Funding Data

- This work was funded by the U.S. National Science Foundation (NSF) (Grant No. CMMI 1853454).

Conflict of Interest

There are no conflicts of interest.

Data Availability Statement

The authors attest that all data for this study are included in the paper.

References

[1] Bavelos, A. C., Kousi, N., Gkournelos, C., Lotsaris, K., Aivaliotis, S., Michalos, G., and Makris, S., 2021, "Enabling Flexibility in Manufacturing by Integrating Shopfloor and Process Perception for Mobile Robot Workers," Appl. Sci., **11**(9), p. 3985.

[2] Romero, M., Guédria, W., Panetto, H., and Barafort, B., 2020, "Towards a Characterisation of Smart Systems: A Systematic Literature Review," Comput. Ind., **120**, p. 103224.

[3] Browne, J., Dubois, D., Rathmill, K., Sethi, S. P., and Stecke, K. E., 1984, "Classification of Flexible Manufacturing Systems," FMS Mag., **1**(1), pp. 114–117.

[4] Bhatta, K., Huang, J., and Chang, Q., 2022, "Dynamic Robot Assignment for Flexible Serial Production Systems," IEEE Rob. Autom. Lett., **7**(3), pp. 7303–7310.

[5] Yadav, A., and Jayswal, S., 2018, "Modelling of Flexible Manufacturing System: A Review," Int. J. Prod. Res., **56**(7), pp. 2464–2487.

[6] Javaid, M., Haleem, A., Singh, R. P., and Suman, R., 2022, "Enabling Flexible Manufacturing System (FMS) Through the Applications of Industry 4.0 Technologies," Internet Things Cyber-Phys. Syst., **2**(1), pp. 49–62.

[7] Hu, L., Liu, Z., Hu, W., Wang, Y., Tan, J., and Wu, F., 2020, "Petri-Net-Based Dynamic Scheduling of Flexible Manufacturing System via Deep Reinforcement Learning With Graph Convolutional Network," J. Manuf. Syst., **55**, pp. 1–14.

[8] Ayvaz, S., and Alpay, K., 2021, "Predictive Maintenance System for Production Lines in Manufacturing: A Machine Learning Approach Using IoT Data in Real-Time," Expert Syst. Appl., **173**, p. 114598.

[9] Xia, K., Sacco, C., Kirkpatrick, M., Saidy, C., Nguyen, L., Kircaliali, A., and Harik, R., 2021, "A Digital Twin to Train Deep Reinforcement Learning Agent for Smart Manufacturing Plants: Environment, Interfaces and Intelligence," J. Manuf. Syst., **58**, pp. 210–230.

[10] Windmann, S., Balzereit, K., and Niggemann, O., 2019, "Model-Based Routing in Flexible Manufacturing Systems," at-Automatisierungstechnik, **67**(2), pp. 95–112.

[11] Li, J., and Meerkov, S. M., 2008, *Production Systems Engineering*, Springer Science & Business Media, Warren, MI.

[12] Rossit, D. A., and Tohmé, F., 2022, "(Data-Driven) Knowledge Representation in Industry 4.0 Scheduling Problems," Int. J. Comput. Integr. Manuf., **35**(10–11), pp. 1–16.

[13] Miranda, J., Ponce, P., Molina, A., and Wright, P., 2019, "Sensing, Smart and Sustainable Technologies for Agri-Food 4.0," Comput. Ind., **108**, pp. 21–36.

[14] Zou, J., Chang, Q., Arinez, J., Xiao, G., and Lei, Y., 2017, "Dynamic Production System Diagnosis and Prognosis Using Model-Based Data-Driven Method," Expert Syst. Appl., **80**, pp. 200–209.

[15] Long, T., Li, Y., and Chen, J., 2022, "Productivity Prediction in Aircraft Final Assembly Lines: Comparisons and Insights in Different Productivity Ranges," J. Manuf. Syst., **62**, pp. 377–389.

[16] Mueller-Zhang, Z., Antonino, P. O., and Kuhn, T., 2021, "Integrated Planning and Scheduling for Customized Production Using Digital Twins and Reinforcement Learning," IFAC-PapersOnLine, **54**(1), pp. 408–413.

[17] Kück, M., and Freitag, M., 2021, "Forecasting of Customer Demands for Production Planning by Local k-Nearest Neighbor Models," Int. J. Prod. Econ., **231**, p. 107837.

[18] Wang, C.-C., Chang, H.-T., and Chien, C.-H., 2022, "Hybrid LSTM-ARMA Demand-Forecasting Model Based on Error Compensation for Integrated Circuit Tray Manufacturing," Mathematics, **10**(13), p. 2158.

[19] Ouaret, S., 2022, "Production Control Problem With Semi-Markov Jump Under Stochastic Demands and Deteriorating Inventories," Appl. Math. Model., **107**, pp. 85–102.

[20] John, T. M., and Millum, J., 2020, "First Come, First Served?," Ethics, **130**(2), pp. 179–207.

[21] Kim, M., and Park, J., 2021, "Learning Collaborative Policies to Solve NP-Hard Routing Problems," Adv. Neural Inf. Process. Syst., **34**, pp. 10418–10430.

[22] Kuhpfahl, J., and Bierwirth, C., 2016, "A Study on Local Search Neighborhoods for the Job Shop Scheduling Problem With Total Weighted Tardiness Objective," Comput. Oper. Res., **66**, pp. 44–57.

[23] Zhang, T., and Mo, H., 2021, "Reinforcement Learning for Robot Research: A Comprehensive Review and Open Issues," Int. J. Adv. Rob. Syst., **18**(3), p. 17298814211007305.

[24] Dalzochio, J., Kunst, R., Pignaton, E., Binotto, A., Sanyal, S., Favilla, J., and Barbosa, J., 2020, "Machine Learning and Reasoning for Predictive Maintenance in Industry 4.0: Current Status and Challenges," Comput. Ind., **123**, p. 103298.

[25] Clifton, J., and Laber, E., 2020, "Q-learning: Theory and Applications," Annu. Rev. Stat. Appl., **7**(1), pp. 279–301.

[26] Li, C., Huang, J., and Chang, Q., 2021, "Data-Enabled Permanent Production Loss Analysis for Serial Production Systems With Variable Cycle Time Machines," IEEE Rob. Autom. Lett., **6**(4), pp. 6418–6425.

[27] Huang, J., Chang, Q., and Arinez, J., 2020, "Deep Reinforcement Learning Based Preventive Maintenance Policy for Serial Production Lines," Expert Syst. Appl., **160**, p. 113701.

[28] Waseem, M., Li, C., and Chang, Q., 2023, "Dynamic Modeling and Analysis of Multi-Product Flexible Production Line," Int. J. Comput. Integr. Manuf., **34**, pp. 1–15.

[29] Zou, J., Chang, Q., Arinez, J., and Xiao, G., 2017, "Data-Driven Modeling and Real-Time Distributed Control for Energy Efficient Manufacturing Systems," Energy, **127**, pp. 247–257.

[30] Ou, X., Chang, Q., Arinez, J., and Zou, J., 2018, "Gantry Work Cell Scheduling Through Reinforcement Learning With Knowledge-Guided Reward Setting," IEEE Access, **6**(9), pp. 14699–14709.

[31] Ou, X., Chang, Q., and Chakraborty, N., 2019, "Simulation Study on Reward Function of Reinforcement Learning in Gantry Work Cell Scheduling," J. Manuf. Syst., **50**, pp. 1–8.