

Article

Robustness-Aware Physical AI for Multi-Functional Humanoid Robot Team Concurrency Control Under Imperfect Digital Twin Information

Rubab Anwar  and Won-Tae Kim * 

Computer Science and Engineering, Computer Science Department, Korea University of Technology and Education, Cheonan 31253, Republic of Korea; rubabanwar@koreatech.ac.kr

* Correspondence: wtkim@koreatech.ac.kr; Tel.: +82-41-560-1485

Abstract

Humanoid robots are emerging as flexible robotic resources for autonomous manufacturing systems, where different types of tasks must be assigned to suitable robots while shared production resources are coordinated effectively. However, realistic manufacturing environments involve dynamic task arrivals, event-driven priority changes, heterogeneous robot capabilities, and shared-resource contention. In addition, digital twin-based scheduling may rely on state information that is delayed or uncertain, which can reduce the reliability of scheduling decisions. To address this issue, this paper extends the previously proposed deep reinforcement learning-based concurrency control (DRLCC) framework for robustness-aware scheduling and shared-resource control of a multi-functional humanoid robot team. The extended framework integrates capability-aware task assignment, feasibility-based action masking, and priority ceiling protocol (PCP)-based shared-resource coordination under delayed and uncertain digital twin observations. The framework is evaluated in a humanoid-based autonomous manufacturing scenario using performance indicators including task completion, urgent-task delay, resource contention, humanoid utilization, and robustness degradation under imperfect state feedback. Compared with the greedy ceiling-based baseline, DRLCC reduces high-priority task delay by approximately 10.9%, priority inversions by 42.1%, average waiting time by 73.8%, average block count by 74.0%, and temporary infeasible events by 73.5%, while maintaining comparable humanoid utilization. The robustness analysis further shows that overall task-completion performance remains stable under imperfect digital twin feedback, although coordination-level metrics are more sensitive to observation uncertainty and delay. These results suggest that the extended DRLCC framework can support robustness-aware humanoid robot team scheduling in autonomous manufacturing environments where digital twin observations are delayed or uncertain.



Academic Editors: Tihomir Opetuk and Davor Kolar

Received: 29 May 2026

Revised: 7 July 2026

Accepted: 21 July 2026

Published: 23 July 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

Keywords: autonomous manufacturing; multi-functional humanoid robots; digital twin; deep reinforcement learning; capability-aware scheduling; shared-resource coordination; robustness-aware scheduling

1. Introduction

Autonomous manufacturing systems are increasingly moving toward flexible, adaptive, and human-centered production environments, where robotic resources are expected to support multiple types of operations rather than execute only fixed and repetitive tasks.

In this context, humanoid robots have attracted growing research attention because their human-like morphology may allow them to operate in workspaces, tools, and assembly settings originally designed for human workers [1]. Related studies on human–robot collaboration also highlight the increasing role of robotic systems in manufacturing applications such as assembly, inspection, and production support [2]. However, introducing multifunctional humanoid robots into autonomous manufacturing introduces a new coordination challenge: tasks must be assigned to humanoids with suitable capabilities, while shared production resources are used safely and efficiently. In this study, physical AI refers to learning-based decision-making for embodied humanoid robots that interact with tasks, resources, and operational constraints in a physical manufacturing environment.

In humanoid-based autonomous manufacturing, scheduling is not limited to selecting the next task for execution; it also requires matching each task with a humanoid that has the required functional capability. Dynamic manufacturing environments may involve urgent job arrivals, changing priorities, resource disturbances, and uncertain execution conditions, which make static scheduling rules less suitable for real-time decision-making [3]. Heterogeneous robot teams further increase this complexity because robots may differ in mobility, manipulation ability, payload capacity, sensing functions, and task-specific skills [4]. Therefore, an effective scheduling framework for multi-functional humanoid teams should jointly consider task priority, humanoid capability, resource availability, and shared-resource conflicts.

Digital twins provide an important foundation for data-driven and real-time manufacturing scheduling by linking the physical production system with its virtual representation, and they have been widely discussed as key enablers for smart manufacturing and digital twin shop-floor integration [5,6]. Prior studies have used digital twin frameworks to support production scheduling under uncertainty [7] and to combine digital twin environments with deep reinforcement learning for dynamic workshop scheduling [8]. However, in practical autonomous manufacturing systems, digital twin feedback may not always represent the exact current physical state because sensing, communication, synchronization, and estimation processes can introduce delay or noise. As a result, a scheduling policy that performs well with perfect state information may become less reliable when decisions are made using imperfect digital twin observations. Figure 1 summarizes the problem setting considered in this study, including the humanoid-based manufacturing environment, shared production resources, and imperfect digital twin observations used for scheduling.

Although prior studies have advanced reinforcement learning-based scheduling [3], digital twin-enabled production control [7,8], heterogeneous robot allocation [4], and feasibility-aware decision-making [9], the combined problem considered in this paper remains insufficiently explored. In particular, there is still a need to examine how a learning-based scheduling framework can coordinate multi-functional humanoid robot teams while considering capability constraints, shared-resource conflicts, and imperfect digital twin observations.

This paper builds on the previously proposed Deep Reinforcement Learning-based Concurrency Control (DRLCC) framework [10], which integrates a Proximal Policy Optimization (PPO)-based adaptive scheduling layer [11] with a dynamic Priority Ceiling Protocol (PCP) layer [12] for resource coordination in federated digital twin manufacturing systems. The original DRLCC framework mainly focuses on priority-aware scheduling, shared-resource control, and priority-inversion mitigation by combining reinforcement learning with PCP-based concurrency logic [10]. In contrast, the present study does not treat priority inversion as the primary contribution. Instead, it extends DRLCC to a more realistic humanoid-based autonomous manufacturing scenario, where multi-functional

humanoid robots have different task capabilities and scheduling decisions may be made using delayed or uncertain digital twin state information.

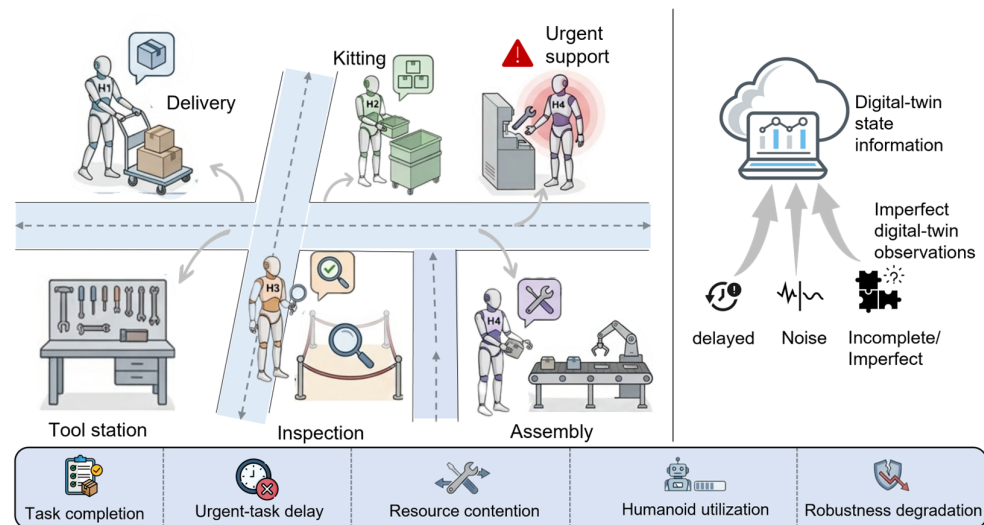


Figure 1. Problem setting for humanoid-based autonomous manufacturing under imperfect digital twin information. The scenario includes multi-functional humanoids, shared lanes, task-specific work areas, and imperfect digital twin observations used for scheduling.

Compared with the previous DRLCC framework, the novelty of this study is not limited to changing the training data or application scenario. The present work reformulates DRLCC for a humanoid-based autonomous manufacturing setting by introducing heterogeneous humanoid capability profiles, task-dependent feasibility constraints, capability-aware action masking, and robustness evaluation under delayed and uncertain digital twin observations. In the previous formulation, the main focus was priority-aware concurrency control and priority-inversion mitigation in a federated digital twin manufacturing environment. In contrast, the present study focuses on whether a PPO-PCP scheduling architecture can support reliable task-level coordination of multi-functional humanoid robot teams when the scheduling state is imperfect. Therefore, the contribution lies in extending DRLCC from general shared-resource scheduling to robustness-aware humanoid-team coordination under capability, resource, and digital twin uncertainty constraints.

The main aim of this paper is to examine whether the previously proposed DRLCC framework can be extended to robustly coordinate multi-functional humanoid robot teams under imperfect digital twin information. The main contributions are fourfold. First, the study reformulates the DRLCC framework for humanoid-based autonomous manufacturing by modeling multi-functional humanoid robots with different task-execution capabilities. Second, it introduces capability-aware task assignment and feasibility-based action masking to prevent invalid humanoid-task selections during PPO-based scheduling. Third, it evaluates DRLCC under imperfect digital twin information by injecting delayed and uncertain state observations into the scheduling process. Fourth, it analyzes the extended framework using performance indicators related to task completion, urgent-task delay, shared-resource contention, humanoid utilization, and robustness degradation.

This study focuses on task-level scheduling, humanoid-task assignment, and shared-resource coordination; low-level bipedal locomotion, whole-body motion planning, and motion-control execution are outside the scope of the present work.

2. Related Work

2.1. Humanoid Robots and Human–Robot Collaboration

Humanoid robots and human–robot collaboration are becoming increasingly relevant in manufacturing as industries move toward more flexible, adaptive, and human-centered production systems. Malik et al. [1] discussed how intelligent humanoid robots can be integrated into manufacturing assembly cells, particularly to support labor-intensive tasks and address worker skill limitations while maintaining human–robot collaboration safety. Dhanda et al. [2] reviewed human–robot collaboration in manufacturing and identified assembly, inspection, and production support as key application areas. In Industry 5.0., Zafar et al. [13] further showed that collaborative robotics, digital twins, and intelligent support technologies are closely connected in smart manufacturing. These studies show the growing importance of humanoid and collaborative robots, but they mainly focus on robot integration, collaboration, and enabling technologies.

2.2. Heterogeneous Multi-Robot Task Allocation

Heterogeneous multi-robot task allocation is also relevant to this study because multi-functional humanoids may have different abilities for delivery, inspection, kitting, assembly, maintenance, and urgent-support tasks. Dai et al. [4] proposed a reinforcement learning-based framework for heterogeneous multi-robot task allocation and scheduling. Lamon et al. [14] studied capability-aware role allocation for industrial assembly tasks by considering agent characteristics and task requirements. Notomista et al. [15] introduced a resilient and energy-aware task-allocation framework for heterogeneous multi-robot systems, while Zhang et al. [16] addressed dynamic and prioritized task scheduling using deep reinforcement learning. These studies show that robot heterogeneity, task priority, and capability-aware allocation are important for multi-robot coordination.

2.3. Reinforcement Learning for Manufacturing Scheduling

Reinforcement learning has become an important approach for dynamic manufacturing scheduling because it can learn adaptive decision policies through interaction with changing production environments. Ngwu et al. [3] reviewed reinforcement learning-based methods for dynamic job-shop scheduling and highlighted their potential for handling complex, uncertain, and continuously changing manufacturing conditions. Zhang et al. [17] proposed a deep reinforcement learning method for learning priority dispatching rules in job-shop scheduling using a graph neural network representation of the scheduling state. Luo [18] addressed dynamic flexible job-shop scheduling with new job insertions, showing the relevance of learning-based scheduling when jobs arrive dynamically. Wang et al. [19] further studied dynamic job-shop scheduling in smart manufacturing and adopted a PPO-based deep reinforcement learning approach to adapt scheduling decisions under uncertain shop-floor conditions. These studies demonstrate the potential of reinforcement learning for adaptive manufacturing scheduling, but most of them focus on job–machine scheduling or dispatching decisions.

2.4. Digital Twins for Production Scheduling

Digital twins are widely used in smart manufacturing to connect physical production systems with virtual models and decision-support tools. Kritzinger et al. [5] clarified the different levels of digital representation in manufacturing, including digital models, digital shadows, and digital twins. Tao and Zhang [6] introduced the Digital Twin Shop-Floor concept, which links physical shop-floor resources, virtual models, services, and digital twin data. For scheduling applications, Negri et al. [7] proposed a field-synchronized digital twin framework for production scheduling under uncertainty, while Geng et al. [8] combined

digital twin-based scheduling with multi-agent deep reinforcement learning. Macchi et al. [20] also discussed the role of digital twins in production scheduling and control. Although these studies show that digital twins can support scheduling, synchronization, and decision-making, fewer works explicitly evaluate how scheduling performance changes when digital twin information is delayed, uncertain, or incomplete.

2.5. Feasibility Constraints and Shared-Resource Control

Feasibility constraints are essential in reinforcement learning-based scheduling because some actions may be invalid in a given system state. In humanoid-based manufacturing, this means that a task should only be assigned to a humanoid with the required capability, and shared resources should be accessed in a controlled manner. Huang and Ontañón [9] studied invalid action masking in policy-gradient algorithms, while Hou et al. [21] further examined practical masking methods for preventing impossible or unavailable actions. Building on the same PPO backbone as the original DRLCC framework [10], the present study applies feasibility-aware action selection to prevent invalid assignments of humanoid tasks. For shared-resource coordination, Sha et al. [12] introduced priority inheritance and priority ceiling protocols to manage priority inversion in real-time systems. These studies provide useful foundations for feasibility-aware decision-making and resource coordination.

2.6. Research Gap Summary

Overall, these studies show important progress in humanoid and collaborative robotics for manufacturing, reinforcement learning-based scheduling, digital twin-enabled production scheduling, heterogeneous multi-robot task allocation, invalid action masking, and PCP-based resource synchronization. However, these areas are often studied separately. Existing works do not fully address the combined problem of multi-functional humanoid capability constraints, dynamic priority-aware scheduling, PCP-based shared-resource coordination, and robustness under delayed or uncertain digital twin observations. Therefore, this study extends the previously proposed DRLCC framework to examine whether it can reliably coordinate a heterogeneous humanoid robot team in autonomous manufacturing when scheduling decisions are made using imperfect digital twin information.

3. Materials and Methods

3.1. Problem Formulation

This study considers the problem of scheduling a heterogeneous team of multi-functional humanoid robots in an autonomous manufacturing environment with dynamic job arrivals, shared-resource contention, and imperfect digital twin feedback. The scheduling problem is formulated as a Markov decision process (MDP), defined by the tuple

$$M = (S, A, R, P, \gamma) \quad (1)$$

where S denotes the state space, A denotes the discrete action space, R denotes the reward function, P denotes the transition dynamics governed by the discrete-event simulation environment, and $\gamma \in [0, 1]$ is the discount factor.

At each decision step t , the scheduling agent receives an observed digital twin state $\hat{s}_t \in S$, selects an action $a_t \in A$, and receives a scalar reward $r_t = R(\hat{s}_t, a_t)$. The observed state $\hat{s}_t$ may differ from the true physical state s_t because of digital twin delay or observation uncertainty. In this study, the action corresponds to selecting a humanoid robot for the current task:

$$a_t = h_i, h_i \in \mathcal{H} \quad (2)$$

where $\mathcal{H} = \{h_1, h_2, \dots, h_6\}$ is the set of available humanoid robots. Since the current task is selected from the priority-aware task queue before humanoid assignment, the policy output represents the humanoid-selection decision for that task.

The objective of the scheduler is to learn a policy $\pi_\theta(a|\hat{s}_t)$ that maximizes the expected cumulative discounted return:

$$J(\theta) = \mathbb{E}_{\pi_\theta} \left[\sum_{t=0}^T \gamma^t r_t \right] \quad (3)$$

while satisfying humanoid-task feasibility and shared-resource control requirements. Unlike the original DRLCC framework, which mainly focused on priority-aware concurrency control, the present formulation extends the scheduling problem to a multi-functional humanoid robot team operating under imperfect digital twin information.

3.2. Manufacturing Environment and System Model

The simulation environment represents a single-zone autonomous manufacturing system in which multiple humanoid robots execute dynamically arriving jobs while sharing limited production resources. The purpose of this environment is not to reproduce an entire factory, but to provide a controlled proof-of-concept setting for evaluating capability-aware scheduling, PCP-based shared-resource control, and robustness under imperfect digital twin observations.

The manufacturing scenario is designed as a task-level abstraction rather than a model of one specific factory or sector. The considered task types, including delivery, kitting, inspection, assembly, maintenance, and urgent support, represent common manufacturing support operations that may appear in different industrial sectors. Therefore, the framework is intended to be transferable at the scheduling and shared-resource coordination level. However, sector-specific deployment would require calibration of task-arrival patterns, task durations, humanoid capability profiles, payload limits, resource layouts, and digital twin uncertainty models using real production data from the target factory.

The system consists of three main elements: jobs, humanoid robots, and shared production resources. Jobs arrive dynamically and contain task-specific requirements. Humanoid robots differ in their task-execution capabilities, payload capacities, battery states, and availability. Shared resources represent physical areas or stations that cannot be accessed by multiple robots simultaneously without control.

3.2.1. Job Model

Jobs arrive dynamically during the simulation according to a Poisson arrival process. Let $J = \{j_1, j_2, \dots, j_N\}$ denote the set of jobs generated within one simulation episode. Each job j is characterized by the following attributes:

- Priority level: $p_j \in \{0, 1, 2\}$, where 0, 1, and 2 represent low-, medium-, and high-priority jobs, respectively. The priority level is sampled with probabilities $\{0.4, 0.3, 0.3\}$.
- Task sequence: each job consists of $n_j \in \{1, 2, 3\}$ subtasks drawn uniformly at random. The duration of each subtask is sampled from a Gamma distribution with shape $k = 2$ and scale $\theta = 1$, giving a mean duration of 2.0 time units.
- Task type: $T_j \in \{\text{delivery}, \text{kitting}, \text{assembly}, \text{inspection}, \text{maintenance}\}$, drawn uniformly at random and fixed for all subtasks of the job.
- Load weight: $w_j \sim \mathcal{U}(50, 200)$, representing the physical payload requirement of the job.

Jobs are maintained in a priority-aware queue and processed through separate scheduling coroutines. A job is considered complete when all of its subtasks have been executed. The simulation horizon is set to $T_{sim} = 2000$ time units.

The task structure considered in this study includes both sequential and interleaved characteristics. Within each job, subtasks are executed sequentially because a job is completed only after all of its subtasks have been processed. However, at the system level, multiple jobs arrive dynamically and are processed through separate scheduling coroutines; therefore, tasks from different jobs can be interleaved in the shared manufacturing environment. This interleaving creates competition for humanoid availability and shared production resources. The proposed DRLCC framework does not require all tasks to follow one fixed global sequence. Instead, it requires that each task has a defined task type, priority level, load requirement, and associated shared-resource requirement so that feasibility checking, capability-aware humanoid assignment, and PCP-based shared-resource coordination can be applied. The main job-generation and simulation parameters are summarized in Table 1.

Table 1. Job-generation and simulation parameters used in the humanoid-based autonomous manufacturing environment.

Parameter	Description	Value
λ	Job-arrival rate	0.4
ρ_1, ρ_2, ρ_3	Low-, medium-, and high-priority probabilities	0.4, 0.3, 0.3
α	Gamma shape parameter	2
β	Gamma scale parameter	1
T_{sim}	Simulation horizon	2000
n_j	Number of subtasks per job	1–3
w_j	Job load-weight range	50–200

3.2.2. Humanoid Robot Model

The manufacturing environment includes $N = 6$ heterogeneous humanoid robots

$$\mathcal{H} = \{h_1, h_2, h_3, h_4, h_5, h_6\} \quad (4)$$

Each humanoid robot h_i is characterized by the following attributes:

- Capability set: C_i representing the task types that humanoid h_i is qualified to perform.
- Payload capacity: W_i , representing the maximum load weight that humanoid h_i can handle.
- Battery level: $b_i \in [0, 1]$, initialized to 1.0 at the beginning of each episode. The battery level decreases during task execution and increases when the humanoid enters charging mode.
- Execution status: each humanoid is either idle, executing a task, or charging.
- Dynamic ceiling: c_i , representing the priority ceiling value used by the PCP-based shared-resource coordination mechanism.

Table 2 defines the capability assignment for the humanoid robot team. Humanoid h_4 acts as the all-rounder capable of executing all six task types, while the remaining humanoids have specialized subsets of capabilities.

Table 2. Capability assignment for the heterogeneous humanoid robot team.

Humanoid	Capabilities
H1	Delivery, kitting
H2	Inspection, urgent support
H3	Maintenance, delivery
H4	Delivery, kitting, inspection, maintenance, urgent support, assembly
H5	Kitting, inspection
H6	Delivery, maintenance, urgent support, assembly

3.2.3. Shared Resources

The manufacturing environment includes a set of shared production resources that must be accessed exclusively during task execution. These resources represent physical areas or stations in the manufacturing zone where simultaneous access by multiple humanoids may cause resource contention. The shared resources considered in this study are shared lanes, the tool station, the inspection area, and the assembly station.

Each task type is associated with one required shared resource, as summarized in Table 3. Delivery and urgent-support tasks require access to the shared lanes, kitting and maintenance tasks require the tool station, inspection tasks require the inspection area, and assembly tasks require the assembly station.

Table 3. Mapping between task types and required shared production resources.

Task Type	Required Shared Resource
Delivery	Shared lanes
Kitting	Tool station
Inspection	Inspection area
Assembly	Assembly station
Maintenance	Tool station
Urgent support	Shared lanes

Because multiple humanoids may request the same shared resource during execution, resource contention can occur. Therefore, after the scheduler selects a humanoid for a task, access to the required resource is regulated by the PCP-based shared-resource coordination module described in Section 3.4.5.

3.3. Digital Twin Observation and State Management Module

The digital twin observation and state-management module provides the state information used by the extended DRLCC scheduler. At each decision step t , the scheduler receives an observed digital twin state $\hat{s}_t$, which represents humanoid status, task-queue information, resource occupancy, and priority-related conditions. The observed state may differ from the true physical system state s_t because of observation uncertainty, synchronization delay, or both. To evaluate robustness, four feedback conditions are considered: ideal, uncertain, delayed, and delayed–uncertain observations.

3.3.1. Observation Uncertainty

Observation uncertainty represents the difference between the actual physical state and the digital twin state available to the scheduler. In this study, uncertainty is applied to the observed state using a proportional Gaussian perturbation:

$$\hat{s}_{t,i} = s_{t,i} + \epsilon_{t,i} \quad (5)$$

$$\epsilon_{t,i} \sim \mathcal{N}(0, (\eta | s_{t,i}|)^2) \quad (6)$$

where $s_{t,i}$ is the i -th feature of the true system state, $\hat{s}_{t,i}$ is the corresponding observed digital twin feature, $\epsilon_{t,i}$ is the uncertainty term, and $\eta = 0.03$ is the uncertainty level.

For humanoid-level observations, battery level and payload capacity are also represented as uncertain digital twin estimates. These estimated values are used during feasibility checking and post-selection verification to evaluate whether the scheduler remains reliable when the digital twin state does not perfectly match the physical system state.

3.3.2. Delayed Observation

Observation delay represents the case where the scheduler receives an older digital twin state instead of the current physical state. To model this condition, a delay buffer is maintained for the state vector. At decision step t , the observed state is given by

$$\hat{s}_t = s_{t-d} \quad (7)$$

where d is the delay length. In the simulation, $d = 2$, meaning that the scheduler receives the state from two decision steps earlier. A fixed scheduling latency of 50 ms is also applied between state observation and action execution, representing the delay associated with digital twin query and decision dispatch.

3.3.3. Digital Twin Observation Condition

Based on the uncertainty and delay models, four digital twin observation conditions are considered. The ideal-feedback condition uses the current system state and serves as the reference case. The uncertain-feedback condition applies observation uncertainty to the current state. The delayed-feedback condition uses a previous state without uncertainty. The delayed–uncertain-feedback condition combines both effects by applying uncertainty to delayed state information. The four observation conditions used to evaluate scheduling robustness are summarized in Table 4.

Table 4. Digital twin observation conditions used for robustness evaluation.

Condition	State Used by Scheduler	Purpose
Ideal feedback	Current system state	Reference condition
Uncertain feedback	Current state with observation uncertainty	Tests sensitivity to uncertain state information
Delayed feedback	Previously observed system state	Tests sensitivity to synchronization delay
Delayed–uncertain feedback	Delayed state with observation uncertainty	Tests robustness under combined digital twin imperfection

The uncertainty and delay settings used in this study are intended to represent common digital twin imperfections rather than reproduce one specific factory dataset. In practical digital twin-based manufacturing systems, scheduling information may differ from the physical state because of sensor noise, communication latency, synchronization delay, state-estimation error, or incomplete updates from physical resources. Therefore, proportional Gaussian perturbation is used to model uncertainty in continuous state features, while the delay buffer and fixed scheduling latency model outdated state feedback and decision-dispatch delay. These settings provide a controlled way to evaluate whether the scheduling policy remains reliable when the digital twin state is imperfect. However, they do not guarantee exact equivalence with every real factory environment. The values should

be calibrated using field data in a deployment scenario. Thus, the present simulation should be interpreted as a proof-of-concept robustness evaluation, and real-data calibration is identified as an important requirement for future industrial validation.

3.4. Proposed Robustness-Aware DRLCC Framework

3.4.1. Framework Overview

This study extends the previously proposed deep reinforcement learning-based concurrency control (DRLCC) [10] framework into a robustness-aware scheduling and shared-resource coordination framework for a multi-functional humanoid robot team. The original DRLCC framework combined a PPO-based adaptive scheduling layer with a PCP-based concurrency control layer to support priority-aware scheduling and shared-resource coordination. In this study, the framework is reformulated for a humanoid-based autonomous manufacturing scenario in which multiple humanoid robots have different task-execution capabilities and must coordinate access to shared production resources under imperfect digital twin observations.

As shown in Figure 2, the proposed framework consists of the physical manufacturing environment and four main functional modules: the manufacturing context preparation module, the digital twin observation and state-management module, the extended DRLCC scheduling module, and the PCP-based shared-resource coordination module. The manufacturing context preparation module defines task types, humanoid capability profiles, shared-resource requirements, and task priorities. The digital twin observation and state-management module provides the observed scheduling state, which may include observation uncertainty, synchronization delay, or both. Based on this observed state, the extended DRLCC scheduling module selects feasible humanoid-task assignments using PPO-based scheduling and capability-aware action masking. The selected assignment is then passed to the PCP-based shared-resource coordination module, which grants or blocks access to shared resources according to priority ceiling logic.

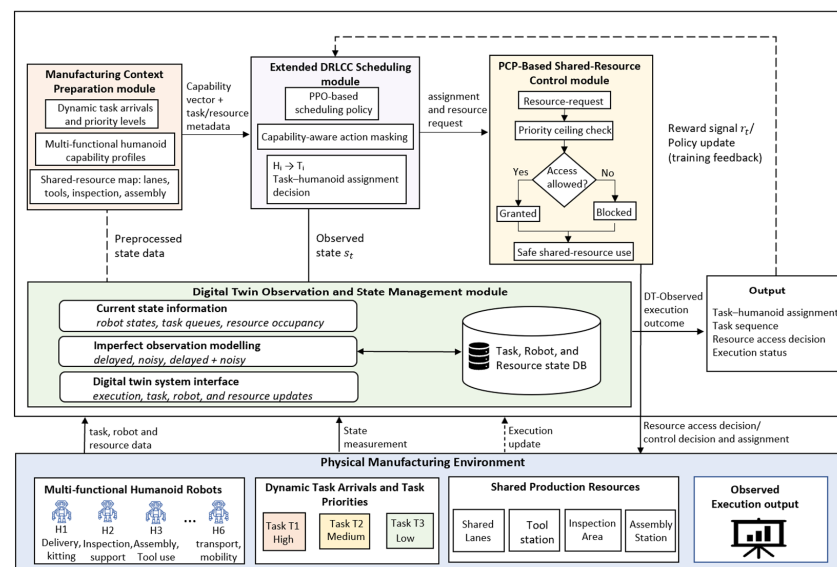


Figure 2. Overview of the proposed robustness-aware DRLCC framework for humanoid-based autonomous manufacturing under imperfect digital twin information. The framework integrates digital twin observation modeling, PPO-based scheduling, capability-aware action masking, and PCP-based shared-resource coordination.

3.4.2. State Representation

At each decision step t , the extended DRLCC scheduler receives the observed digital twin state $\hat{s}_t$. The state representation includes humanoid-level features and task-queue-

level features. For each of the six humanoid robots, four features are included: occupancy status, current job priority, battery level, and PCP ceiling value. These humanoid-level features contribute $6 \times 4 = 24$ dimensions.

The remaining eight dimensions describe the task queue, including queue length, priority counts, and waiting-time statistics. Therefore, the complete observed state vector is represented as:

$$\hat{s}_t \in \mathbb{R}^{32} \quad (8)$$

Before being passed to the PPO policy network, the state vector is normalized to improve training stability. The same state representation is used under ideal, uncertain, delayed, and delayed-uncertain digital twin feedback conditions.

3.4.3. Action Space and Capability-Aware Masking

The scheduling action corresponds to selecting one humanoid robot for the current task. Since the task is already selected from the priority-aware queue, the PPO policy only determines which humanoid should execute that task. Therefore, the discrete action space is defined as:

$$A = \{1, 2, \dots, |\mathcal{H}|\} \quad (9)$$

where each action index represents one humanoid robot in the team. The selected action forms the task-humanoid assignment (j, h_i) , where task j is assigned to a humanoid h_i .

Before action selection, infeasible humanoids are removed using capability-aware action masking. For a task j , the feasible humanoid set is defined as:

$$\mathcal{F}_j = \{h_i \in \mathcal{H} \mid \tau_j \in C_i, W_i \geq w_j, b_i \geq b_{\min}, h_i \text{ is available}\} \quad (10)$$

where τ_j is the task type, C_i is the capability set of a humanoid h_i , W_i is the payload capacity, w_j is the task load weight, b_i is the battery level, and $b_{\min}$ is the minimum battery threshold.

The binary action mask is then defined as

$$m_i = \begin{cases} 1, & \text{if } h_i \in \mathcal{F}_j, \\ 0, & \text{otherwise} \end{cases} \quad (11)$$

The mask is applied to the PPO action logits before the softmax operation, so infeasible humanoids receive near-zero selection probability. If no feasible humanoid is available, the task remains waiting, and the corresponding waiting or contention metrics are updated.

3.4.4. Robust Post-Selection Feasibility Verification

After the PPO policy selects a humanoid, a post-selection feasibility check is performed before the assignment is committed for execution. This step is required because delayed or uncertain digital twin observations may not fully match the current physical system state. The selected humanoid-task assignment is accepted only if the humanoid is available, has the required task capability, satisfies the payload and battery constraints, and remains feasible under the current system state. Otherwise, the assignment is rejected, the task remains in the waiting queue, and the corresponding infeasible-assignment metric is updated.

3.4.5. PCP-Based Shared-Resource Coordination

Following the PCP-based resource-control logic used in the original DRLCC framework [10] and the priority ceiling protocol introduced by Sha et al. [12], the PCP-based shared-resource coordination module determines whether the required resource can be accessed safely after the extended DRLCC scheduler selects a humanoid-task assignment.

In the proposed framework, the PPO policy selects a suitable humanoid, while the PCP regulates access to shared production resources, such as shared lanes, the tool station, the inspection area, and the assembly station. This separation prevents the learning-based scheduler from directly controlling low-level resource-locking decisions.

Each humanoid maintains a dynamic priority ceiling based on the highest-priority waiting task for which it is a feasible candidate. The ceiling values are updated at each scheduling step and are used to decide whether a selected assignment can proceed without causing unsafe resource contention or priority-related conflicts.

If the required resource is available and the PCP condition is satisfied, the task is allowed to execute and the resource remains locked until task completion. Otherwise, the assignment is temporarily blocked, and the corresponding blocking, contention, or priority-inversion metrics are updated. In this way, PCP acts as a deterministic safety layer within the extended DRLCC framework, while the PPO scheduler focuses on adaptive humanoid-task assignment under imperfect digital twin observations.

3.4.6. Reward Feedback and Policy Update

After task execution, blocking, or assignment rejection, the environment returns a reward signal to guide the PPO-based scheduler. The reward function is designed to encourage successful task completion and high-priority task responsiveness while penalizing waiting time, repeated blocking, priority inversion, and infeasible assignments.

The reward components used in the extended DRLCC scheduler are summarized in Table 5. Positive rewards are assigned for successful task completion, with higher emphasis on urgent or high-priority jobs. Negative rewards are assigned when a task experiences excessive waiting, resource blocking, priority inversion, or temporary infeasibility. The resulting transition is stored and later used to update the actor–critic policy during PPO training.

Table 5. Reward components used in the extended DRLCC scheduler.

Reward Component	Purpose
Task-completion reward	Rewards successful task completion
Urgent-task reward	Gives higher reward to high-priority jobs
Waiting-time penalty	Penalizes priority-weighted task waiting time
Blocking penalty	Penalizes repeated blocking during shared-resource access
Priority-inversion penalty	Penalizes priority-related scheduling conflicts
Infeasible-assignment penalty	Penalizes invalid or unavailable humanoid-task assignments

The overall scheduling and shared-resource coordination procedure, adapted from the PPO–PCP structure of the original DRLCC framework [10], is summarized in Algorithm 1.

The PPO-specific training configuration used to optimize the scheduling policy is described in Section 3.5.

Algorithm 1 Robustness-aware DRLCC scheduling and shared-resource coordination procedure.

Require: Episodes E , jobs J , seed set S , delay length d , uncertainty level η

Ensure: Trained policies $\{\pi_{\theta,s}^*\}_{s \in S}$ and per-seed experimental logs

```

1: procedure EXTENDEDRLCCTRAINING ( $E, J, S, d, \eta$ )
2:   for  $s \in S$  do
3:     Set random seed  $s$ 
4:     Initialize PPO actor–critic networks  $(\pi_{\theta}, V_{\theta})$ 
5:     Initialize PPO memory  $\mathcal{M}$  and metric buffer  $\mathcal{B}$ 
6:     for  $e = 1$  to  $E$  do
7:       Initialize environment, humanoids, queue, and resources
8:       Define humanoid capability sets
9:       Initialize PCP-based resource controller
10:      Initialize digital twin buffer and episode metrics
11:      while unfinished tasks remain do
12:        Obtain true system state  $s_t$ 
13:        Generate observed digital twin state  $\hat{s}_t$  using  $d$  and  $\eta$ 
14:        Select current task  $j$  from the waiting queue
15:        Construct capability-aware action mask  $m_t$ 
16:        if no feasible humanoid is available then
17:          Update waiting and contention metrics
18:          Continue to next decision step
19:        end if
20:        Select feasible humanoid  $h_i \sim \pi_{\theta}(\cdot | \hat{s}_t, m_t)$ 
21:        Form assignment  $a_t = (j, h_i)$ 
22:        Check shared-resource access using PCP rules
23:        if access is granted then
24:          Execute task and update system states
25:        else
26:          Block assignment and update contention metrics
27:        end if
28:        Compute reward  $r_t$ 
29:        Observe next digital twin state  $\hat{s}_t + 1$ 
30:        Store transition  $\hat{s}_t, a_t, r_t, \hat{s}_t + 1, m_t$  in  $\mathcal{M}$ 
31:        if PPO update condition is satisfied then
32:          Update actor–critic parameters  $(\theta, \varnothing)$  using PPO
33:          Clear PPO memory  $\mathcal{M}$ 
34:        end if
35:      end while
36:      Record performance and robustness metrics
37:      Append episode metrics to  $\mathcal{B}$ 
38:    end for
39:    Save  $\mathcal{B}$  and best trained policy  $\pi_{\theta,s}^*$ 
40:  end for
41:  return  $\{\pi_{\theta,s}^*\}_{s \in S}$ 
42: end procedure

```

3.5. PPO Training Procedure

The scheduling policy is trained using proximal policy optimization (PPO) [11], following the actor–critic training structure adopted in the original DRLCC framework [10]. The policy network selects one humanoid from the feasible action set, while the value network estimates the state value for policy improvement. Both networks use two fully connected hidden layers with 256 units and ReLU activation. During training and inference, the capability-aware action mask is applied to the policy logits before action selection so that infeasible humanoids receive near-zero selection probability.

The actor and critic networks are optimized using Adam with a learning rate of 1×10^{-4} . Generalized advantage estimation is used with $\lambda = 0.95$, the discount factor is set to $\gamma = 0.95$, the PPO clipping threshold is set to 0.1, and the policy is updated for six epochs with a mini-batch size of 32. The main PPO training hyperparameters are summarized in Table 6.

Table 6. PPO training hyperparameters.

Parameter	Value	Parameter	Value
Discount factor, γ	0.95	Hidden layers	2
GAE parameter, λ	0.95	Entropy coefficient	0.01
Learning rate	1×10^{-4}	Hidden units per layer	256
PPO clipping threshold	0.1	Activation function	ReLU
Mini-batch size	32	Optimizer	Adam
Update epochs	6	Gradient clipping	0.5

3.6. Simulation Environment and Experimental Configuration

All experiments are implemented in a discrete-event simulation environment. At the beginning of each episode, the humanoid states, job queue, shared resources, digital twin delay buffer, and performance counters are initialized. During simulation, jobs arrive dynamically according to the job-generation model described in Section 3.2.1. For each scheduling decision, the extended DRLCC scheduler selects a feasible humanoid-task assignment, the PCP-based coordination module checks shared-resource access, and the digital twin state is updated after task execution, blocking, or assignment rejection.

Each episode runs for a simulation horizon of $T_{sim} = 2000$ time units. The experiments are repeated using six random seeds to account for stochastic variation in job arrivals, task durations, humanoid states, and scheduling decisions. For each training run, the policy is trained for 500 episodes, and the final 50 episodes are used to compute the reported performance metrics. This final-window averaging is used to evaluate the converged scheduling behavior rather than early-stage exploration performance.

3.7. Baseline Methods

To evaluate the effectiveness of the proposed robustness-aware DRLCC framework, its performance is compared against three non-learning baselines and two additional learning-based baselines. The non-learning baselines are selected to represent different levels of scheduling capability: random feasible assignment tests the effect of basic capability filtering; greedy battery-based assignment represents a simple readiness-oriented heuristic; and greedy ceiling-based assignment represents a rule-based method that considers priority ceilings and resource awareness. To further address learning-based comparison, DQN + PCP and A2C + PCP baselines are also evaluated.

The random feasible assignment method selects one humanoid randomly from the feasible candidate set. The greedy battery-based method selects the feasible humanoid with

the highest available battery level. The greedy ceiling-based method selects a feasible humanoid using priority ceiling and shared-resource feasibility information. The DQN + PCP and A2C + PCP baselines use the same PCP-based shared-resource coordination logic as DRLCC, but replace the PPO learning backbone with DQN-based and A2C-based policies. The proposed extended DRLCC method uses PPO-based masked scheduling with PCP-based shared-resource coordination. All methods are evaluated under the same job-generation process, humanoid capability constraints, shared-resource model, and digital twin feedback conditions.

The purpose of the baseline comparison is to evaluate whether the proposed PPO-PCP scheduling framework improves task-level humanoid assignment and shared-resource coordination compared with rule-based scheduling policies and alternative DRL backbones under the same capability, priority, and digital twin feedback constraints. PPO was adopted as the learning backbone because this study extends the previously proposed PPO-based DRLCC framework and because PPO provides a stable policy-gradient structure for discrete humanoid-selection decisions with feasibility masking. The purpose of this study is not to claim that PPO is universally superior to all DRL algorithms, but to evaluate whether the DRLCC (PPO-PCP) architecture can be extended to robustness-aware humanoid-team coordination under imperfect digital twin information. Broader hyperparameter-tuned benchmarking of DRL backbones remains an important direction for the future. The baseline methods used for performance comparison are summarized in Table 7.

Table 7. Baseline methods used for performance comparison.

Method	Description
Random feasible assignment	Randomly selects one humanoid from the feasible candidate set
Greedy battery-based assignment	Selects the feasible humanoid with the highest available battery
Greedy ceiling-based assignment	Selects a feasible humanoid using priority ceiling/resource-awareness
DQN + PCP	Uses DQN-based humanoid selection with PCP-based shared-resource coordination
A2C + PCP	Uses A2C-based humanoid selection with PCP-based shared-resource coordination
Proposed extended DRLCC	Uses PPO-based masked scheduling with PCP-based shared-resource control

3.8. Evaluation Metrics and Robustness Analysis

The proposed framework is evaluated using metrics for task-completion efficiency, urgent-task responsiveness, shared-resource coordination, humanoid utilization, and robustness to imperfect digital twin feedback. These metrics are selected to examine whether the extended DRLCC framework can maintain reliable scheduling and coordination performance when the observed digital twin state is delayed, uncertain, or both.

The main evaluation metrics are summarized in Table 8. Makespan measures the overall time required to complete the workload. Urgent-task delay evaluates the responsiveness of the scheduler to high-priority jobs. Priority inversion count measures priority-related scheduling conflicts. Average waiting time and average block count evaluate queue delay and shared-resource blocking behavior. Temporary infeasible events measure invalid or unavailable assignment attempts, while humanoid utilization measures the active usage of the humanoid team during execution.

Table 8. Evaluation metrics used for performance and robustness analysis.

Metric	Purpose
Makespan	Measures overall task-completion efficiency
Urgent-task delay	Measures responsiveness to high-priority tasks
Priority inversion count	Measures priority-related scheduling conflicts
Average waiting time	Measures queue delay before assignment
Average block count	Measures repeated blocking during execution
Temporary infeasible events	Measures invalid or unavailable assignment attempts
Humanoid utilization	Measures robot usage during execution
Robustness degradation	Measures performance change under imperfect feedback

To quantify robustness, the performance under each imperfect digital twin feedback condition is compared with the clean-feedback condition. For lower-is-better metrics, robustness degradation is calculated as:

$$D_M = \frac{M_{\text{imperfect}} - M_{\text{ideal}}}{M_{\text{ideal}}} \times 100\% \quad (12)$$

For higher-is-better metrics, such as humanoid utilization, degradation is calculated as:

$$D_M = \frac{M_{\text{ideal}} - M_{\text{imperfect}}}{M_{\text{ideal}}} \times 100\% \quad (13)$$

Here, M_{ideal} denotes the metric value under ideal digital twin feedback, and $M_{\text{imperfect}}$ denotes the corresponding value under uncertain, delayed, or delayed–uncertain feedback. A smaller degradation value indicates that the scheduling framework is less sensitive to imperfect digital twin observations.

4. Results and Discussion

This section evaluates the proposed robustness-aware DRLCC framework using two main experiments and an additional learning-based baseline comparison. First, DRLCC is compared with non-learning baseline methods under delayed–uncertain digital twin feedback. Second, DQN + PCP and A2C + PCP are evaluated as additional learning-based baselines under the same delayed–uncertain digital twin feedback condition. Third, an ablation analysis evaluates the effect of digital twin uncertainty, delay, and their combined influence on scheduling and shared-resource coordination.

4.1. Comparison with Non-Learning Baselines

To evaluate the effectiveness of the proposed extended DRLCC framework, its performance was compared with three non-learning baseline methods: random feasible assignment, greedy battery-based assignment, and greedy ceiling-based assignment. All methods were evaluated using the same job-generation process, humanoid capability constraints, shared-resource configuration, and delayed–uncertain digital twin feedback condition.

Table 9 presents the baseline comparison results. The proposed DRLCC scheduler achieves the lowest makespan among the methods compared while maintaining humanoid utilization comparable to that of the baseline policies. This indicates that the learned scheduling policy improves task flow without relying on excessive utilization of specific humanoids.

Table 9. Baseline comparison under delayed–uncertain digital twin feedback.

Method	Makespan	Urgent-Task Delay	Priority Inversions	Avg. Wait Time	Avg. Block Count	Infeasible Events	Humanoid Utilization
DRLCC	253.73 ± 4.8	4.41 ± 0.04	4.05 ± 0.38	0.16 ± 0.01	0.32 ± 0.02	32.66 ± 2.74	0.26 ± 0.004
Random Feasible with no RL	256.33 ± 4.9	5.79 ± 0.03	19.93 ± 0.93	1.20 ± 0.05	2.40 ± 0.10	248.2 ± 10.90	0.26 ± 0.007
Greedy Battery	257.28 ± 4.6	4.91 ± 0.07	7.35 ± 0.57	0.62 ± 0.01	1.25 ± 0.03	125.4 ± 2.98	0.26 ± 0.004
Greedy Ceiling	257.12 ± 4.4	4.95 ± 0.07	6.99 ± 0.32	0.61 ± 0.01	1.23 ± 0.03	123.2 ± 3.50	0.26 ± 0.005

Note: Bold values indicate the best performance among the compared methods.

As shown in Table 9, DRLCC achieves the lowest makespan and maintains humanoid utilization comparable to the baseline methods. Compared with the greedy ceiling-based baseline, DRLCC reduces high-priority task delay from 4.95 to 4.41, priority inversions from 6.99 to 4.05, average waiting time from 0.61 to 0.16, average block count from 1.23 to 0.32, and temporary infeasible events from 123.2 to 32.66. These results show that DRLCC improves coordination-level performance under delayed–uncertain digital twin feedback.

The random feasible baseline performs worst in priority inversions, waiting time, blocking, and infeasible events, indicating that capability filtering alone is insufficient. The greedy battery-based and greedy ceiling-based methods improve over random assignment, but they remain less effective than DRLCC because they rely on fixed rule-based decisions. Overall, the results indicate that PPO-based adaptive scheduling with capability-aware masking and PCP-based resource coordination provides more reliable humanoid-task assignment under imperfect digital twin observations.

Figure 3 visually supports the baseline comparison reported in Table 9. While the makespan values remain relatively close across methods, DRLCC shows a clearer advantage in coordination-sensitive metrics, including average waiting time, average block count, and temporary infeasible events. This indicates that the main benefit of the proposed framework is not only faster task completion but also more reliable humanoid-task assignment and shared-resource coordination under delayed–uncertain digital twin feedback.

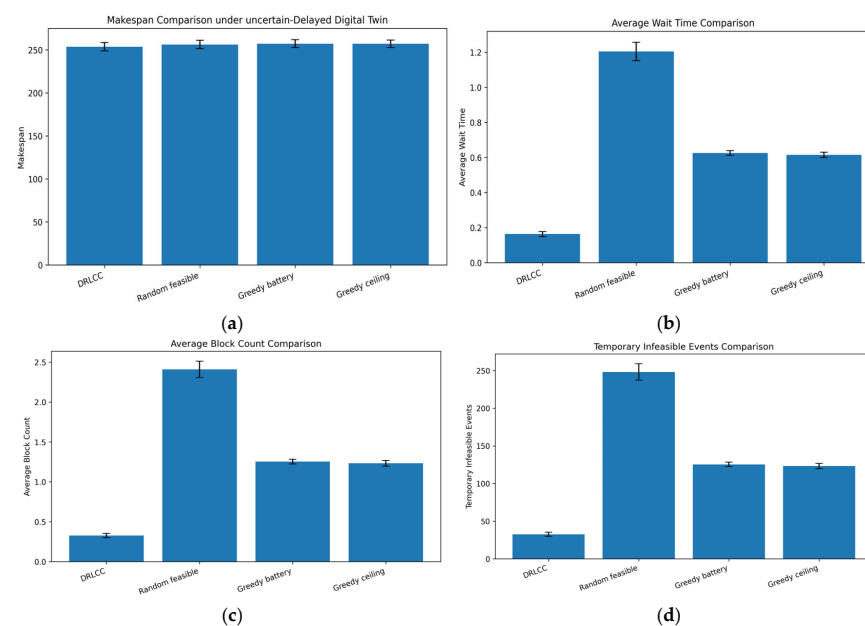


Figure 3. Baseline comparison under delayed–uncertain digital twin feedback: (a) makespan, (b) average waiting time, (c) average block count, and (d) temporary infeasible events. Error bars indicate standard deviation across random seeds.

To further compare the proposed method with learning-based scheduling baselines, DQN + PCP and A2C + PCP were evaluated under the same delayed–uncertain digital twin feedback condition. These baselines use the same PCP-based shared-resource coordination logic but replace the PPO learning backbone with DQN-based and A2C-based policies, respectively. As shown in Table 10, A2C + PCP achieves slightly lower makespan and urgent-task delay, while the proposed DRLCC achieves fewer priority inversions, lower average waiting time, lower average block count, fewer temporary infeasible events, and slightly higher humanoid utilization. These results indicate that DRLCC provides a stronger balance for priority-aware and coordination-sensitive scheduling metrics, while remaining competitive with alternative DRL baselines.

Table 10. Learning-based baseline comparison under delayed–uncertain digital twin feedback.

Method	Makespan	Urgent-Task Delay	Priority Inversions	Avg. Wait Time	Avg. Block Count	Temporary Infeasible Events	Humanoid Utilization
DRLCC	253.739 ± 4.817	4.410 ± 0.045	4.057 ± 0.381	0.163 ± 0.014	0.327 ± 0.027	32.667 ± 2.742	0.265 ± 0.004
DQN + PCP	255.172 ± 3.832	4.378 ± 0.079	4.123 ± 0.265	0.176 ± 0.014	0.354 ± 0.028	35.400 ± 2.802	0.264 ± 0.003
A2C + PCP	253.513 ± 4.753	4.353 ± 0.054	4.183 ± 0.459	0.174 ± 0.013	0.348 ± 0.025	34.830 ± 2.540	0.263 ± 0.005

4.2. Robustness of DRLCC Under Digital Twin Delay and Uncertainty

To evaluate the robustness of the proposed extended DRLCC framework, the trained scheduler was tested under four digital twin feedback conditions: ideal feedback, uncertain feedback, delayed feedback, and delayed–uncertain feedback. Ideal feedback is used as the reference condition, where the scheduler receives the current system state without observation disturbance. The remaining conditions evaluate the effect of observation uncertainty, synchronization delay, and their combined influence on scheduling and shared-resource coordination.

As shown in Table 11, DRLCC maintains stable production-level performance across the four feedback conditions. Makespan and humanoid utilization remain close to the ideal-feedback case, indicating that imperfect digital twin feedback does not significantly disrupt the overall task-completion flow.

Table 11. Performance of DRLCC under different digital twin feedback conditions.

Observation Condition	Makespan	Urgent-Task Delay	Priority Inversions	Avg. Wait Time	Avg. Block Count	Temporary Infeasible Events	Humanoid Utilization
Ideal feedback	253.37 ± 4.7	4.26 ± 0.05	4.15 ± 0.40	0.15 ± 0.008	0.30 ± 0.01	30.58 ± 1.73	0.265 ± 0.005
Uncertain feedback	253.46 ± 4.6	4.39 ± 0.05	4.37 ± 0.70	0.17 ± 0.011	0.35 ± 0.02	35.13 ± 2.25	0.265 ± 0.005
Delayed feedback	253.25 ± 4.7	4.33 ± 0.06	4.07 ± 0.64	0.14 ± 0.01	0.29 ± 0.02	29.50 ± 2.03	0.265 ± 0.005
Delayed–uncertain feedback	255.10 ± 4.83	4.41 ± 0.04	4.05 ± 0.38	0.16 ± 0.01	0.32 ± 0.02	32.66 ± 2.74	0.264 ± 0.004

Figure 4 further shows that production-level performance remains stable, while coordination-sensitive metrics change more noticeably under imperfect feedback. Uncertain feedback increases waiting time, blocking, and temporary infeasible events compared with ideal feedback, whereas delayed feedback alone causes only limited changes. These

results suggest that the proposed framework is more robust to short observation delays than to uncertainty in the observed digital twin state.

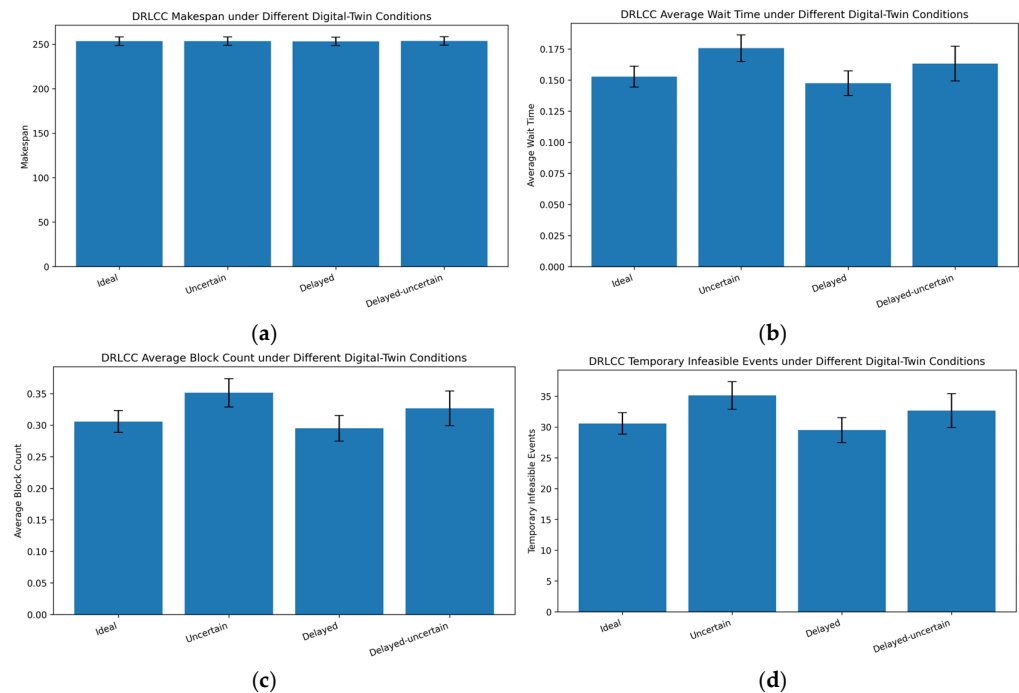


Figure 4. DRLCC performance under different digital twin feedback conditions: (a) makespan, (b) urgent-task delay, (c) average waiting time, and (d) temporary infeasible events. Error bars indicate standard deviation across random seeds.

4.3. Robustness Degradation and Coordination Analysis

To further quantify the effect of imperfect digital twin feedback, robustness degradation was calculated by comparing each imperfect-feedback condition with the ideal-feedback condition. This analysis shows how much each metric changes when the scheduler receives uncertain, delayed, or delayed–uncertain observations.

As shown in Table 12, makespan degradation remains very small across all imperfect-feedback conditions, indicating that DRLCC preserves overall task-completion performance even when the digital twin observation is imperfect. Delayed feedback alone produces almost no degradation in most metrics, suggesting that the scheduler can tolerate a short observation delay when the state information remains accurate.

Table 12. Robustness degradation of DRLCC under imperfect digital twin feedback relative to clean feedback.

Twin Condition	Makespan Degradation	HP Delay Degradation	Priority Inversion Degradation	Wait Time Degradation	Block Count Degradation	Infeasible Event Degradation
Uncertain	0.03%	3.04%	5.21%	14.98%	14.85%	14.85%
Delayed	0.00%	1.57%	0.00%	0.00%	0.00%	0.00%
Delayed–uncertain	0.14%	3.44%	0.00%	6.85%	6.80%	6.80%

In contrast, uncertain feedback produces clearer degradation in coordination-related metrics, including high-priority task delay, average waiting time, average block count, and temporary infeasible events. The delayed–uncertain condition also causes degradation, but the increase remains limited in this experiment. This suggests that the post-selection feasibility verification and PCP-based shared-resource coordination help reduce the effect of combined delay and uncertainty.

Overall, the degradation analysis shows that the extended DRLCC framework is robust at the production level, while coordination-level metrics remain more sensitive to imperfect digital twin information.

4.4. Discussion

The results indicate that the proposed robustness-aware DRLCC framework is effective for humanoid robot team scheduling under imperfect digital twin feedback. The baseline comparison shows that DRLCC improves coordination-related metrics compared with random feasible assignment and greedy rule-based methods, particularly in waiting time, blocking frequency, and temporary infeasible events. This suggests that PPO-based adaptive scheduling provides better assignment decisions than fixed heuristic policies when the observed digital twin state is delayed or uncertain.

The robustness analysis further shows that production-level metrics, especially makespan and humanoid utilization, remain stable across different digital twin feedback conditions. However, coordination-level metrics are more sensitive to imperfect observations. This means that delayed or uncertain digital twin information may not prevent task completion, but it can still affect assignment quality, resource blocking, and priority-related coordination behavior.

The additional learning-based comparison further shows that DQN + PCP and A2C + PCP achieve performance close to the proposed PPO-based DRLCC, confirming that DRL-based scheduling with PCP coordination is effective under delayed–uncertain digital twin feedback. However, DRLCC provides a stronger balance in coordination-sensitive metrics, including priority inversions, average waiting time, average block count, temporary infeasible events, and humanoid utilization. A broader hyperparameter-tuned benchmark across additional DRL backbones, digital twin conditions, and manufacturing layouts remains an important direction for the future.

These findings support the use of feasibility-based action masking, post-selection verification, and PCP-based shared-resource coordination as safety layers around the learning-based scheduler. The PPO policy improves adaptive humanoid-task assignment, while the feasibility and PCP layers reduce invalid or unsafe execution decisions when digital twin observations do not fully match the current physical state. The main simulation assumptions, corresponding real manufacturing considerations, and implications for deployment are summarized in Table 13.

This analysis clarifies that the present results should be interpreted as a controlled simulation-based validation of task-level scheduling and shared-resource coordination rather than as a complete real-world humanoid deployment study. The current framework evaluates whether the DRLCC policy remains stable when capability constraints, shared-resource contention, and imperfect digital twin feedback are present. For industrial deployment, the simulation parameters should be calibrated using real production logs, measured digital twin synchronization errors, sensor uncertainty, network latency, and humanoid execution data. Therefore, the proposed framework provides a proof-of-concept foundation, while real-data calibration and physical or high-fidelity digital twin validation remain necessary future steps.

The present study is limited to a controlled simulation environment with a single-zone manufacturing scenario and predefined humanoid capability profiles. Future work should evaluate the framework in larger multi-zone manufacturing systems, include more realistic humanoid motion and task-execution constraints, and investigate adaptive methods for estimating and compensating digital twin delay and uncertainty.

Table 13. Simulation-to-real gap analysis for industrial deployment.

Simulation Assumption	Real Manufacturing Consideration	Implication for Deployment
Single-zone manufacturing environment	Real factories may contain multiple zones, routing paths, work cells, and cross-zone dependencies	The framework should be validated in larger multi-zone layouts before deployment
Predefined humanoid capability profiles	Real humanoid capabilities may change because of tool condition, payload variation, battery status, calibration error, or task difficulty	Capability profiles should be updated using real execution data
Synthetic job arrivals and task durations	Real production orders may follow shift schedules, urgent insertions, machine availability, and demand fluctuations	Job-generation parameters should be calibrated using factory logs
Proportional Gaussian observation uncertainty	Real digital twin errors may be non-Gaussian, sensor-specific, or caused by missing/incomplete updates	Noise models should be replaced or calibrated using measured sensor and digital twin data
Fixed delay buffer and 50 ms scheduling latency	Real delays may vary depending on network load, database update frequency, sensor synchronization, and controller response time	Delay compensation should be adaptive and validated under measured communication latency
Task-level scheduling focus	Real humanoid deployment also requires locomotion, motion planning, manipulation, safety monitoring, and low-level control	The proposed scheduler should be integrated with robot execution and safety-control layers

5. Conclusions

This paper presented a robustness-aware extension of the DRLCC framework for scheduling and shared-resource coordination of a multi-functional humanoid robot team under imperfect digital twin information. The framework integrates PPO-based humanoid selection, capability-aware action masking, post-selection feasibility verification, and PCP-based shared-resource coordination. The baseline comparison showed that DRLCC improves coordination-level performance compared with random feasible and greedy rule-based methods under delayed-uncertain digital twin feedback. The robustness analysis showed that makespan and humanoid utilization remain stable across different digital twin feedback conditions, while coordination-level metrics are more sensitive to uncertainty and delay. Future work will extend the framework to larger multi-zone manufacturing environments, more realistic humanoid execution models, and adaptive methods for compensating digital twin delay and uncertainty.

Future work may also integrate the proposed task-level DRLCC scheduler with low-level adaptive interaction-control methods. For example, impedance learning for human-guided robots in contact with unknown environments has shown that robot impedance parameters can be adapted to support stable physical interaction during contact-rich tasks [22]. Such methods are complementary to the present framework. While the proposed DRLCC scheduler determines which humanoid should execute a task and how shared resources should be coordinated, impedance-learning-based control could improve the execution reliability of the selected humanoid during contact-intensive manufacturing operations such as assembly, maintenance, or tool handling. Therefore, integrating task-level scheduling with adaptive impedance control is a promising direction for improving the practical performance of humanoid robots in real manufacturing environments.

Author Contributions: Conceptualization, R.A. and W.-T.K.; methodology, R.A.; software, R.A.; validation, R.A. and W.-T.K.; investigation, R.A. and W.-T.K.; writing—original draft preparation, R.A.; writing—review and editing, R.A. and W.-T.K.; visualization, R.A.; supervision, W.-T.K.; project administration, W.-T.K. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the National Research Foundation of Korea (NRF), grant funded by the Korean government (MSIT) (RS-2026-25490993). This work was also partly supported by the Korea Evaluation Institute of Industrial Technology (KEIT), grant funded by the Korean government (MOTIE) (RS-2025-02307650, Development and Practice of an On-device AI Functionality and Performance Testing Framework based on NPU).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The simulation data (SimPy event logs and metrics) generated and analyzed during this study will be made available by the corresponding author upon reasonable request.

Acknowledgments: During the preparation of this manuscript, the authors used OpenAI ChatGPT and Grammarly solely for superficial language editing, including grammar, spelling, sentence structure, punctuation, and readability. These tools were not used to generate research data, figures, study design, methodology, analysis, interpretation of results, or scientific conclusions. The authors reviewed and edited all outputs and take full responsibility for the content of this publication.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Malik, A.A.; Masood, T.; Brem, A. Intelligent Humanoid Robots in Manufacturing: Addressing the Worker Shortage and Skill Gaps in Assembly Cells. In Proceedings of the Companion of the 2024 ACM/IEEE International Conference on Human-Robot Interaction, Boulder, CO, USA, 11–14 March 2024; pp. 20–27.
2. Dhanda, M.; Rogers, B.; Hall, S.; Dekoninck, E.; Dhokia, V. Reviewing human-robot collaboration in manufacturing: Opportunities and challenges in the context of Industry 5.0. *Robot. Comput.-Integr. Manuf.* **2025**, *93*, 102937. [\[CrossRef\]](#)
3. Ngwu, C.; Liu, Y.; Wu, R. Reinforcement learning in dynamic job shop scheduling: A comprehensive review of AI-driven approaches in modern manufacturing. *J. Intell. Manuf.* **2026**, *37*, 1093–1108.
4. Dai, W.; Rai, U.; Chiun, J.; Cao, Y.; Sartoretti, G. Heterogeneous multi-robot task allocation and scheduling via reinforcement learning. *IEEE Robot. Autom. Lett.* **2025**, *10*, 2654–2661. [\[CrossRef\]](#)
5. Kritzinger, W.; Karner, M.; Traar, G.; Henjes, J.; Sihm, W. Digital Twin in manufacturing: A categorical literature review and classification. *IFAC-Pap.* **2018**, *51*, 1016–1022. [\[CrossRef\]](#)
6. Tao, F.; Zhang, M. Digital Twin Shop-Floor: A New Shop-Floor Paradigm Towards Smart Manufacturing. *IEEE Access* **2017**, *5*, 20418–20427. [\[CrossRef\]](#)
7. Negri, E.; Pandhare, V.; Cattaneo, L.; Singh, J.; Macchi, M.; Lee, J. Field-synchronized Digital Twin framework for production scheduling with uncertainty. *J. Intell. Manuf.* **2021**, *32*, 1207–1228.
8. Geng, S.; Huang, S.; Guo, Y.; Qian, W.; Fang, W.; Zhang, L.; Wang, S. Digital twin driven dynamic scheduling of discrete manufacturing workshop with transportation resource constraint using multi-agent deep reinforcement learning. *Robot. Comput.-Integr. Manuf.* **2025**, *95*, 103042. [\[CrossRef\]](#)
9. Huang, S.; Ontañón, S. A closer look at invalid action masking in policy gradient algorithms. In Proceedings of the Thirty-Fifth International Florida Artificial Intelligence Research Society Conference, Jensen Beach, FL, USA, 15–18 May 2022.
10. Anwar, R.; Kwon, J.-W.; Kim, W.-T. A Deep Reinforcement Learning-Based Concurrency Control of Federated Digital Twin for Software-Defined Manufacturing Systems. *Appl. Sci.* **2025**, *15*, 8245. [\[CrossRef\]](#)
11. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal Policy Optimization Algorithms. *arXiv* **2017**, arXiv:1707.06347.
12. Sha, L.; Rajkumar, R.; Lehoczy, J.P. Priority Inheritance Protocols: An Approach to Real-Time Synchronization. *IEEE Trans. Comput.* **1990**, *39*, 1175–1185. [\[CrossRef\]](#)
13. Zafar, M.H.; Langås, E.F.; Sanfilippo, F. Exploring the synergies between collaborative robotics, digital twins, augmentation, and Industry 5.0 for smart manufacturing: A state-of-the-art review. *Robot. Comput.-Integr. Manuf.* **2024**, *89*, 102769. [\[CrossRef\]](#)
14. Lamon, E.; De Franco, A.; Peternel, L.; Ajoudani, A. A capability-aware role allocation approach to industrial assembly tasks. *IEEE Robot. Autom. Lett.* **2019**, *4*, 3378–3385. [\[CrossRef\]](#)

15. Notomista, G.; Mayya, S.; Emam, Y.; Kroninger, C.; Bohannon, A.; Hutchinson, S.; Egerstedt, M. A resilient and energy-aware task allocation framework for heterogeneous multi-robot systems. *IEEE Trans. Robot.* **2022**, *38*, 159–179. [[CrossRef](#)]
16. Zhang, J.; Jia, Q.; Zhang, S.; Chen, G. Dynamic and prioritized task scheduling of heterogeneous multi-robot systems using deep reinforcement learning. *Neurocomputing* **2025**, *638*, 130184. [[CrossRef](#)]
17. Zhang, C.; Song, W.; Cao, Z.; Zhang, J.; Tan, P.S.; Xu, C. Learning to dispatch for job shop scheduling via deep reinforcement learning. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 1621–1632.
18. Luo, S. Dynamic scheduling for flexible job shop with new job insertions by deep reinforcement learning. *Appl. Soft Comput.* **2020**, *91*, 106208. [[CrossRef](#)]
19. Wang, L.; Hu, X.; Wang, Y.; Xu, S.; Ma, S.; Yang, K.; Liu, Z.; Wang, W. Dynamic job-shop scheduling in smart manufacturing using deep reinforcement learning. *Comput. Netw.* **2021**, *190*, 107969. [[CrossRef](#)]
20. Macchi, M.; Ragazzini, L.; Negri, E. A conceptual framework for Digital Twins in production scheduling and control. *IFAC-Pap.* **2023**, *56*, 5661–5666. [[CrossRef](#)]
21. Hou, Y.; Liu, Z.; Zhang, X.; Zhang, Y.; Chen, X. Exploring the Use of Invalid Action Masking in Reinforcement Learning: A Comparative Study of On-Policy and Off-Policy Algorithms. *Appl. Sci.* **2023**, *13*, 8283. [[CrossRef](#)]
22. Xing, X.; Burdet, E.; Si, W.; Yang, C.; Li, Y. Impedance Learning for Human-Guided Robots in Contact With Unknown Environments. *IEEE Trans. Robot.* **2023**, *39*, 3705–3721. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.