# Predictive Path Coordination of Collaborative Transportation Multirobot System in a Smart Factory

Zixiang Nie, *Member, IEEE*, and Kwang-Cheng Chen, *Fellow, IEEE*

***Abstract*—Smart factories employ intelligent transportaton systems such as autonomous mobile robots (AMRs) to support real-time adjusted production flows for agile and flexible production. While decentralized transportation task execution provides a scalable multirobot system (MRS) for a smart factory, new coordination challenges arise in implementing such a system. Transportation-MRS collaborates with production-MRS to accommodate just-in-time (JIT) production, leading to nonstationary transportation tasks that transportation-MRS must learn and adapt to. Also, decentralized operation on a shared shop floor means that one robot cannot factor in peer robots' task execution planning, leading to competitive collisions. Meanwhile, predictive coordination with communication among multiple learning and adapting intelligent robots is still an open problem. On top of identifying the aforementioned challenges, this article first proposes a multifloor transportation graph model to discretize transportation task execution and allow real-time adjustment of transportation paths toward collision-free. We introduce a unique collaborative multi-intelligent robot system approach taking each robot as a cyber–physical agent with automated artificial intelligence (AI) workflow. First, it includes a novel multiagent reinforcement learning (MARL) algorithm, where each robot predictively plans collision-avoidant paths. Second, we introduce a token-passing mechanism to resolve inevitable competitive collisions due to nonstationary tasks. The proposed approach innovatively uses the multifloor model as a domain model for planning. By allowing competitive collision to occur and resolve, a robot only needs to learn and adapt to uncertain parts of the environment—nonstationary tasks and peer robots' paths. Computational experiments show that our approach is both sample-efficient and computationally efficient. The transportation-MRS quickly reaches near-optimal performance levels, which are empirically shown to scale with the number of robots involved.**

***Index Terms*—Collaborative–competitive environment, decentralized multiagent reinforcement learning (MARL), intelligent transportation systems, multirobot system (MRS), smart factory.**

## I. Introduction

SMART factories driven by multirobot systems (MRSs) that are modeled as cyber–physical systems consisting of a transportation-MRS and a production-MRS, enable flexible, agile, and efficient productivity and resource utilization, with scalability and resilience [1].

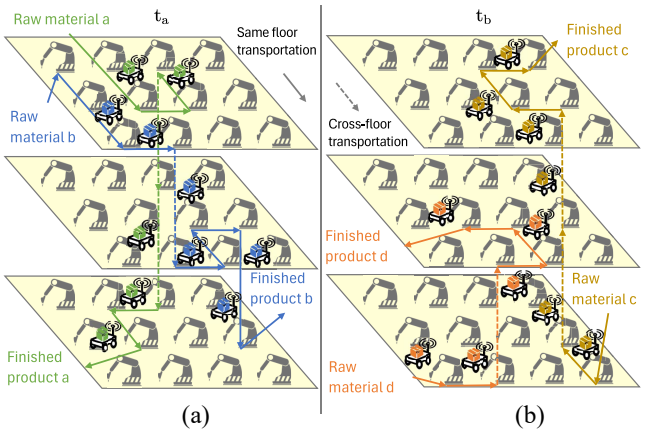



Fig. 1. Smart factory transportation is formulated as discrete navigations of a transportation-MRS in a multifloor environment. (a) Production flows at t=$t_a$. (b) Production flows at t=$t_b$.

An artificial intelligence (AI)-enabled transportation-MRS decentralized operates in a shared multifloor environment to achieve agile and flexible production in a smart factory [2], [3] as depicted in Fig. 1. Edge computing performs real-time multirobot task allocation (MRTA) [4] and dynamically assigns production tasks to production-MRS according to production demands and dynamic transportation tasks to transportation-MRS. Since the shared shop floor, competitive collisions occur when multiple robots are located at the same location and intend to take the same navigation step, which can be resolved or avoided toward collision-free. Transportation-MRS accomplishes transportation tasks assigned by edge computing via predictive collision-avoidant path planning, discrete navigations with real-time adjustments, and competitive collision resolution toward collision-free operation. This operation paradigm allows production flows to change from $a$ and $b$ to $c$ and $d$ between time $t_a$ and $t_b$, and transport-MRS autonomously adapts to the changes.

In contrast to conventional industrial autonomous transportation, techniques, such as conveyor belts, assembly lines, and autonomous guided vehicles (AGVs), are designed to accommodate static production flows. Once such systems are tuned to satisfy productivity and energy efficiency, they lose adaptability to dynamic changes in just-in-time (JIT)

Manuscript received 27 October 2023; revised 19 March 2024; accepted 9 July 2024. Date of publication 6 August 2024; date of current version 18 September 2024. This article was recommended by Associate Editor J. Wang. *(Corresponding author: Zixiang Nie.)*

The authors are with the Department of Electrical Engineering, University of South Florida, Tampa, FL 33620 USA (e-mail: znie@usf.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TSMC.2024.3431222.

Digital Object Identifier 10.1109/TSMC.2024.3431222

production demands [5]. Moreover, unlike warehouse logistics with autonomous mobile robots (AMRs), transportation tasks in smart factories do not begin or end at a single fixed location and have a precise due time because they are parts of production flows [6]. Literature, such as [7] and [8], formulate smart factory transportation as multipath scheduling that optimizes delay and energy consumption. Yet, they cannot address adaptation to transportation tasks that support dynamic production flows in a smart factory. Also, [9] and [10] formulate the path-following control for industrial AGVs but cannot address the path constraints from the shared shop floor and therefore are not applicable to smart factories.

Despite the benefits of scalability from decentralized operation, applying AI-enabled transportation-MRS in a smart factory faces new challenges in coordination.

1) A smart factory accommodating JIT production introduces nonstationary transportation tasks to transportation-MRS. The task execution must adapt to the transportation tasks with constantly new statistic characteristics in order to maintain delay and energy consumption performance. Although the Markov decision process (MDP) is widely adapted to model sequential decision problems such as transportation, its dimension increases with the introduction of more robots. Furthermore, common multidimensional MDPs cannot account for the nonstationary transportation tasks and real-time adjusted execution because of the various task execution times. Literature, such as [6], [11], and [12], model discrete navigation as MDP but lack illustration about optimizing transportation task performance.
2) Decentralized path planning cannot guarantee collision-free task execution, since one robot's path planning cannot factor in the plans of other robots. This is particularly challenging in real-time, end-to-end transportation tasks where replanning is required when competitive collision, deviation, and deadlocks happen. Furthermore, due to the expanded dimension of the shop floor, predicting all robots' path plans for coordination is not a scalable smart factory solution even for edge computing. Literature, such as [13] and [14], address end-to-end multiagent pickup and delivery problems but without agent-based modeling, which means those methods' time complexity grows exponentially with increasing robot numbers and thus difficult to optimize.
3) On top of discrete navigation enabled by AI decision making, a communication-based collaboration is needed to automate and regulate information collection and exchange from task completion and collision resolutions to optimize performance. Literature, such as [15] and [16], propose such mechanisms in urban traffic scenarios while the effective, scalable mechanism for a multifloor smart factory is still an open problem.

This article proposes a unique coordination strategy enabled by decentralized cyber–physical AI agents [17] as transportation-MRS, depicted in Fig. 2. This strategy allows competitive collisions to occur and resolve, which can lead to a larger delay in task execution. Meanwhile, MRS learn from the collision resolution, so that they can take optimal or near-optimal plans to avoid competitive collisions in a predictive manner, as illustrated by $t_a$ to $t_c$ in Fig. 2. We implement such a strategy as a multiagent reinforcement learning (MARL) framework in a collaborative–competitive environment [18], optimizing the average task execution delay and energy consumption assisted by the multifloor model. Literature, such as [19], [20], and [21], also adopt reinforcement learning frameworks to resolve multiagent path planning in dynamic, nonindustrial environments, but they are yet to address the optimization of nonstationary transportation tasks in a multifloor smart factory.

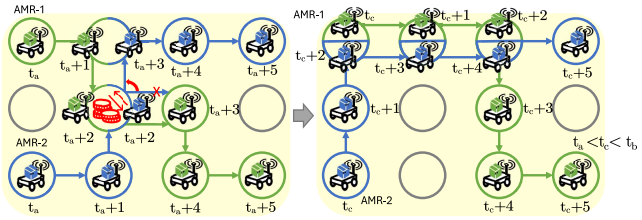



Fig. 2. Proposed coordination strategy allows transportation-MRS to learn, adapt, and optimize performance through planning collision-avoidant paths, real-time adjusting transportation paths, and resolving competitive collisions.

To highlight the technical contributions, this article offers a novel modeling of smart factory transportation by treating it as a multirobot discrete navigation problem within a multifloor domain model, complete with path constraints and nonstationary tasks. It enables real-time adaptivity and collision-free task executions. To address the challenges introduced by decentralized operation, we introduce a unique collaborative multi-intelligent robot system approach to realize the proposed coordination strategy. In this approach, each robot is given greedy path planning based on the domain model, and a predictive collision-avoidant plan selection is implemented as the actor-mixed-critics MARL algorithm. The MARL architecture is thoughtfully designed, featuring a shared, centralized, collaboratively trained global critic, along with distributed local critics for each robot. Additionally, we introduce a token-passing mechanism to resolve competitive collisions, which automates inter-robot communication to ensure collision-free operations. By allowing competitive collision to occur and resolve, transportation-MRS learn and adapt toward optimizing collective performance without explicitly predicting nonstationary tasks and peer robots' plans. Through computational experiments across various scales of transportation-MRS and shop floor sizes, we demonstrate that our collaborative multi-intelligent robot system achieves optimization of both delay and energy consumption.

Section II in this article details the transportation-MRS model, graphical multifloor model, and transportation-MRS discrete navigation problem. Section III illustrates the technical implementation that addresses smart factory transportation, including the multiagent actor-mixed-critics algorithm and token-passing-based collision resolution. Section IV shows the results of the computational experiment in terms of effectiveness, optimality, and scalability. Section V summarizes this article.

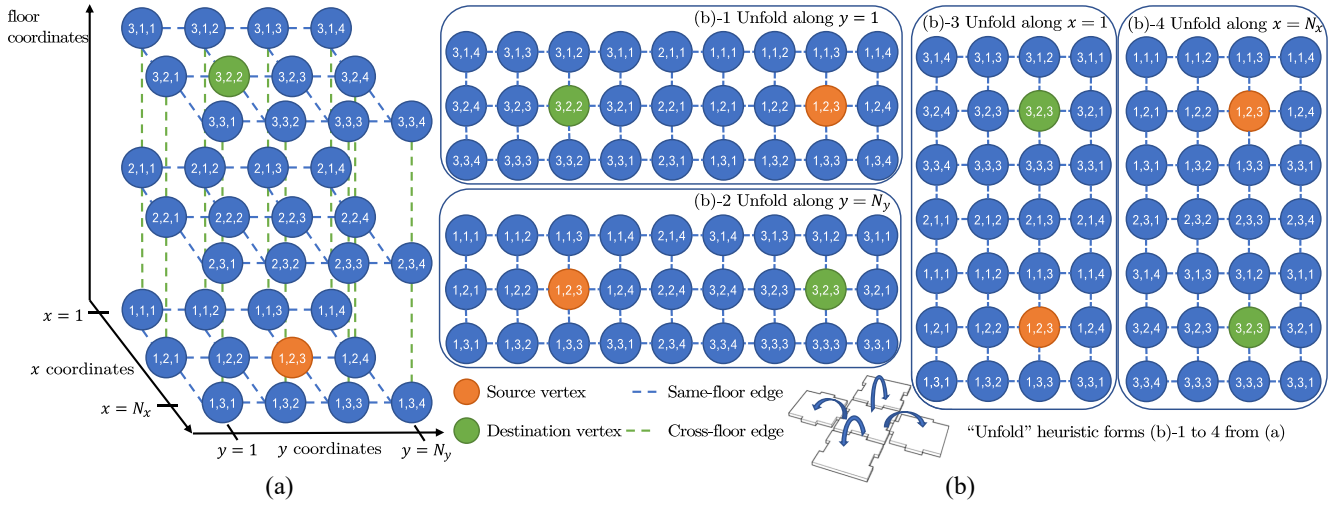



Fig. 3. Smart factory multifloor model for transportation. (a) Multifloor model. (b) Illustration of the "unfold" heuristic.

## II. Smart Factory Transportation Model

### A. Transportation-MRS in Multifloor Smart Factory

A multifloor smart factory transportation environment is characterized by an undirected graph, denoted as $G_{\text{floor}}$. It is defined by the tuple $(N_f, N_x, N_y)$, where each element represents the maximum value of the floor, $x$, and $y$ coordinates, respectively. A robot may either move from one vertex to another linked vertex or remain at the current vertex, termed an "action" or a "navigation step" throughout this article. This model spans a Euclidean space, specifically:

1) $N_f$ represents the number of floors, with values ranging from 1 to $N_f$.
2) Each individual floor is structured as an $N_x \times N_y$ lattice graph (also known as a square grid graph). Vertices within this graph correspond to either a production robot or a production cell. The $x$ coordinates span from 1 to $N_x$, while the $y$ coordinates range from 1 to $N_y$.

The edges of $G_{\text{floor}}$ characterize feasible navigation steps between adjacent vertices, whether through the floor, $x$, or $y$ coordinates. Given the nature of adjacency, these edges are inherently undirected. When navigating within the same floor, adjacent vertices in the $x$ or $y$ direction are interconnected, consistent with the properties of a lattice graph. For navigation between floors, typically facilitated by UAVs or elevators which come with their own constraints [22], only vertices adjacent in floor coordinates and located on, the sides of the floor are connected. This is further elaborated by (1).

Essentially, $G_{\text{floor}}$ is a cyclic graph, characterized by the presence of multiple graph cycles. Moreover, all edges within this graph bear weights, indicative of the energy consumption associated with navigation. The energy consumption is associated with the navigation steps along one of the coordinates, denoted $\varepsilon_f, \varepsilon_x, \varepsilon_y$.

The formal definition of $G_{\text{floor}}$ is presented in (1), where the vertices are denoted as $v_{i,j,k}$, in which the subscripts are sequenced by the floor, $x$, and $y$ coordinates. Additionally, a vertex positioned on the side, expressed as $v_{i,j_{\text{side}},k_{\text{side}}}$, is located at the side of the $i$th floor, as elaborated in (1c) and (1e). Fig. 3(a) provides a visualization of an example $G_{\text{floor}}$ characterized by $(N_f = 3, N_x = 3, N_y = 4)$. If a robot navigates from vertex $v_{1,2,3}$ to $v_{3,2,2}$, it would necessarily traverse the vertices situated on the sides of all three floors to facilitate cross-floor navigation

$$G_{\text{floor}} = (\mathbf{V}_{\text{floor}}, \mathbf{E}_{\text{floor}}) \tag{1a}$$

$$\mathbf{V}_{\text{floor}} = \{v_{i,j,k}\}, i = 1, \ldots, N_f, j = 1, \ldots, N_v, k = 1, \ldots, N_h \tag{1b}$$

$$\mathbf{E}_{\text{floor}} = \left\{\left(v_{i,j,k}, v_{i,j',k'}\right)\right\} \cup \left\{\left(v_{i,j_{\text{side}},k_{\text{side}}}, v_{i',j_{\text{side}},k_{\text{side}}}\right)\right\} \tag{1c}$$

$$j' = j \pm 1, k' = k \pm 1 \tag{1d}$$

$$j_{\text{side}} = 1, N_v, k_{\text{side}} = 1, N_h, i' = i \pm 1. \tag{1e}$$

A transportation-MRS comprises $K$ robots, hereafter referred to as $K$-robots, operating with discrete synchronous navigation in each timeslot $t$. Every navigation step consumes one timeslot, independent of energy consumption. This synchronized approach promotes real-time adjustments that ensure collision-free operations, as corroborated by prior studies [1], [4], [23]. Given that industrial robots may have various durations of navigation steps, the timeslot is designed to be flexible, accounting for longer navigation steps like cross-floor navigation. The multifloor model acts both as the environment for the robot system and as a knowledge base for intelligent decision making, given that both time and energy consumption metrics for navigation are predefined.

### B. Nonstationary Transportation Tasks

Despite all the navigation steps being synchronized, robots in the transportation-MRS execute transportation tasks in an end-to-end sequence, in which a new task is assigned upon completion of the prior one. This leads to asynchrony at the task execution level. As discussed in Section I, JIT production demands necessitate reconfigurations of both MRSs, specifically, changes in production and transportation tasks. Nonetheless, between these reconfigurations, the transportation-MRS operates with a static set of tasks that reflect production tasks and flows. This dynamic nature

means transportation task assignments for robots are inherently nonstationary and challenging to predict. They might be reconfigured with a different set of tasks to serve adjusted production flows, reconfigured to support ongoing flows, or assigned a completely new task due to reconfiguration.

To model such nonstationarity, let $M^t$ represent the total number of transportation tasks, adjustable at reconfiguration intervals denoted by $t = t_1, t_2, \ldots$. During each reconfiguration, a fresh set of $M^t$ tasks, $\text{TS}^t$ is computed, comprising pairs of pickup vertex $v^m_{f_p,x_p,y_p}$ and delivery vertex $v^m_{f_d,x_d,y_d}$ as defined in (2a). This task generation process is stochastic, characterized by (2b). $h_{\text{DU}}$ represents a discrete uniform distribution, designating two vertices from $G_{\text{floor}}$ for pickup and delivery purposes. Thus, according to the law of small numbers [24], each task set $\text{TS}^t$ is unique and nonstationary

$$\text{TS}^t = \left\{\left(v^1_{f_p,x_p,y_p}, v^2_{f_d,x_d,y_d}\right), \ldots, \left(v^m_{f_p,x_p,y_p}, v^m_{f_d,x_d,y_d}\right) \ldots, \left(v^{M^t}_{f_p,x_p,y_p}, v^{M^t}_{f_d,x_d,y_d}\right)\right\} \tag{2a}$$

$$\left(v^m_{f_p,x_p,y_p}, v^m_{f_d,x_d,y_d}\right) = h_{\text{DU}}(2; \mathbf{V}_{\text{floor}}) \tag{2b}$$

$$m = 1, 2, \ldots, M^t \tag{2c}$$

$$t = t_1, t_2, \ldots. \tag{2d}$$

A transportation task, denoted as $\mathsf{task}^m$ for $m = 1, 2, \ldots, M^t$, when assigned to robot-$k$, is characterized by the pickup vertex $v^m_{f_p,x_p,y_p}$, delivery vertex $v^m_{f_d,x_d,y_d}$, along with associated pickup and delivery due times, $\tau^m_{\text{due,pickup}}$ and $\tau^m_{\text{due,deliver}}$, respectively, as defined in (3). Given that robot-$k$'s current location at the time of assignment is $v_{f_k,x_k,y_k}$, it navigates to $v^m_{f_p,x_p,y_p}$ for pickup and subsequently moves to $v^m_{f_d,x_d,y_d}$ for delivery. The $A^*$ algorithm, represented by $h_{A^*}$, computes the shortest feasible delay for both paths. However, since robots might not always opt for the shortest path, a margin parameter $\beta$ is introduced to adjust due times appropriately

$$\mathsf{task}^m = \left(v^m_{f_p,x_p,y_p}, v^m_{f_d,x_d,y_d}, \tau^m_{\text{due,pickup}}, \tau^m_{\text{due,deliver}}\right) \tag{3a}$$

$$\tau^m_{\text{due,pickup}} = (1+\beta)\cdot h_{A^*}\left(v_{f_k,x_k,y_k}, v^m_{f_p,x_p,y_p}\right) \tag{3b}$$

$$\tau^m_{\text{due,deliver}} = (1+\beta)\cdot h_{A^*}\left(v_{f_p,x_p,y_p}, v^m_{f_d,x_d,y_d}\right). \tag{3c}$$

In summary, the set of transportation tasks is updated in the reconfiguration timeslot (occurring every several hundred to thousand timeslots [25]), as delineated by (2). Task assignments are made upon task completion by a robot, a necessity for adaptable smart factories. While efficient scheduling and assignment of production and transportation tasks are important for the practice of our method, these elements are not central to our proposed approach. For such challenges, real-time MRTA in smart factories as explored by [4] is proposed and can be integrated with our methodology. In this work, we assign tasks in the set of tasks to robots without bias, so that all tasks are executed equitably.

### C. Task Execution as Adaptive Discrete Navigations

As illustrated in Section II-B, smart factory robots are assigned new, nonstationary transportation tasks upon completion of the last one. Also, the time or timeslots of task executions (referred to as delay) vary since they depend on the robots' locations when tasks are assigned. Therefore, on top of path planning, robots must collaboratively and adaptively coordinate their path plans toward collision-free and optimized delay.

A navigation step taken by robot-$k$ in timeslot $t$ is denoted by action $a^t_k$, which takes robot-$k$ from vertex $v_{f,x,y}$ to a linked adjacent vertex $v_{f',x',y'}$ in $G_{\text{floor}}$, represented by shorthand denotation (4a). Equations (4b)–(4h) formally define the actions empirically "west," "south," "east," "north," "stay," "up-floor," and "down-floor," respectively, and their numerical representations. The discrete action space for robots is thus $\mathcal{A} = \{0, 1, 2, 3, 4, 5, 6\}$.

Consider the modeling of the above two sections, robot-$k$ located at $v_{f_0,x_0,y_0}$ is assigned $\mathsf{task}^m$ defined by (3) at timeslot $t$, navigates to $v_{f_p,x_p,y_p}$ with $\tau^m_{k,\text{pickup}}$ timeslots to pick up, then navigates to $v_{f_d,x_d,y_d}$ with $\tau^m_{k,\text{deliver}}$ timeslots to deliver, which is a task execution. This procedure produces two walks in $G_{\text{floor}}$, respectively, $\boldsymbol{\rho}^m_{k,\text{pickup}}$ and $\boldsymbol{\rho}^m_{k,\text{deliver}}$, defined by (5a) and (5b)

$$v_{f,x,y} \xrightarrow{a^t_k} v_{f',x',y'} \tag{4a}$$

$$a^t_k = 0 := f' = f, x' = x, y' = y - 1 \tag{4b}$$

$$a^t_k = 1 := f' = f, x' = x + 1, y' = y \tag{4c}$$

$$a^t_k = 2 := f' = f, x' = x, y' = y + 1 \tag{4d}$$

$$a^t_k = 3 := f' = f, x' = x - 1, y' = y \tag{4e}$$

$$a^t_k = 4 := f' = f, x' = x, y' = y \tag{4f}$$

$$a^t_k = 5 := f' = f + 1, x' = x - 1, y' = y \tag{4g}$$

$$a^t_k = 6 := f' = f - 1, x' = x - 1, y' = y. \tag{4h}$$

Therefore, the delay (total timeslots, completion time, or makespan in some literature) and the energy consumption of completing $\mathsf{task}^m$ by robot-$k$ are given by (5c) and (5d), respectively. As defined in Section II-A, each navigation step takes one timeslot, the delay is the number of vertices in walks $\boldsymbol{\rho}^m_{k,\text{pickup}}$ and $\boldsymbol{\rho}^m_{k,\text{deliver}}$ minus 1, which is also the number of actions that makeup each walk. Particularly in this section that emphasizes graphical modeling, we use the term "walk" instead of "path" since the positive possibility of robot-$k$ taking the same vertex multiple times in a walk. In the rest of this article, the term "path" is in the context of transportation instead of graph theory

$$\boldsymbol{\rho}^m_{k,\text{pickup}} = \left(v_{f_0,x_0,y_0} \xrightarrow{a^t_k} v_{f_1,x_1,y_1} \xrightarrow{a^{t+1}_k} v_{f_2,x_2,y_2} \cdots \cdots \xrightarrow{a^{t+\tau_{\text{pickup}}}_k} v^m_{f_p,x_p,y_p}\right) \tag{5a}$$

$$\boldsymbol{\rho}^m_{k,\text{deliver}} = \left(v^m_{f_p,x_p,y_p} \xrightarrow{a^{t+\tau_{\text{pickup}}+1}_k} v_{f_3,x_3,y_3} \cdots \cdots \xrightarrow{a^{t+\tau_{\text{pickup}}+\tau_{\text{deliver}}}_k} v^m_{f_d,x_d,y_d}\right) \tag{5b}$$

$$\tau_k^m = \tau_{k,\text{pickup}}^m + \tau_{k\text{deliver}}^m \tag{5c}$$

$$\mathcal{E}_k^m = h_{\text{weight}}\left(\boldsymbol{\rho}_{k,\text{pickup}}^m\right) + h_{\text{weight}}\left(\boldsymbol{\rho}_{k,\text{deliver}}^m\right). \tag{5d}$$

Competitive collision in task execution occurs when the robot set $K_{f,x,y}^t$ is located at $v_{f,x,y}$ at time $t$. Given all planned actions $a_k^t, k \in K_{f,x,y}^t$, the set of actions does not exist as (6) since there are duplicate elements. Thanks to time synchronization, competitive collisions can be resolved by token-passing, detailed in Section III-C, which may lead robots to take the "stay" action and extend the delay

$$\nexists\left\{a_k^t | \forall a_k^t, k \in K_{f,x,y}^t\right\}. \tag{6}$$

Delay and energy consumption are two major performance indicators for smart factory transportation [26] are thus adopted to evaluate the performance of transportation-MRS given by (7a) under the proposed multifloor model, task model, and formulated discrete navigation problem. Equation (7a) gives the evaluation function, including the delay and energy consumption averaged by all completed tasks and by all robots. A Cobb–Douglas utility function [27] is adopted with $\alpha$ indicating the preference between delay and energy. Equation (7b) suggests the delay and energy are derived from the task execution paths $\boldsymbol{\rho}_{k,\text{pickup}}^m$, $\boldsymbol{\rho}_{k,\text{deliver}}^m$, and those paths must fulfill $\mathsf{task}^m$

$$\text{minimize} \quad \frac{\sum_{k=1}^{K}\sum_{t=1}^{T}\sum_{m=1}^{M^t}\left(\tau_k^m\right)^{\alpha}\left(\mathcal{E}_k^m\right)^{1-\alpha}}{K\sum_{t=1}^{T}M^t} \tag{7a}$$

$$\text{s.t.} \quad \exists \quad \tau_k^m, \mathcal{E}_k^m \leftarrow \left(\mathsf{task}^m, \boldsymbol{\rho}_{k,\text{pickup}}^m, \boldsymbol{\rho}_{k,\text{deliver}}^m\right) \tag{7b}$$

$$m = 1, 2, \ldots, M^t. \tag{7c}$$

The goal of the MRS coordination, referred to as the collective objective, is to minimize the value of the evaluation in (7a). Task executions along with their delay and energy consumption are contributed by the discrete navigation of all robots in transportation-MRS, which is a multiagent optimization problem that addresses the first challenge in Section I.

## III. Collaborative Multi-Intelligent Robot System With Domain Model for Path Coordination

In this work, we propose a collaborative multi-intelligent robot system approach that employs a multifloor transportation model as its domain model for planning. This approach seeks to collaboratively and predictively coordinate the paths of robots throughout the entire task execution, aiming to complete tasks within due time and optimize the collective objective (7). Our approach has two key components to realize the strategy proposed in Section I.

*Collaborative Predictive Collision Avoidance:* A multiagent actor-mixed-critics algorithm is proposed to allow transportation-MRS to perform collaborative and predictive planning with the multifloor model, avoiding competitive collisions. It addresses the second challenge in Section I. The algorithm has an architecture, including a shared global critic module, supplemented by distributed local critic and actor modules associated with each robot.

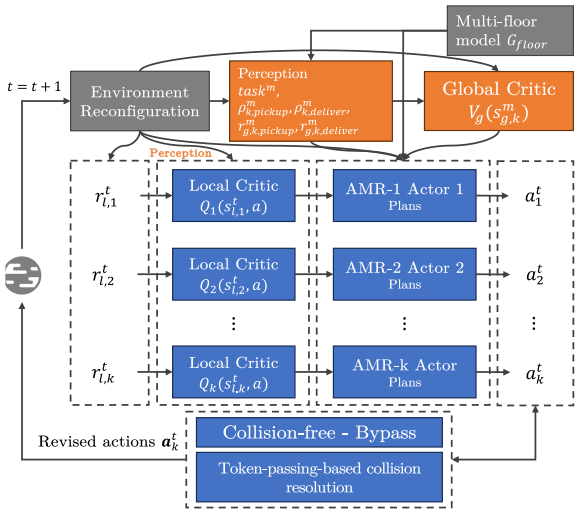



Fig. 4. Architecture of the multiagent actor-mixed-critics with the multifloor model as a domain model. Each robot possesses an actor module for planning upon receiving a transportation task assignment, utilizing predictions from both the shared global critic and its own local critic to decide a collision-avoidant path to execute. Given that this planning does not consider the plans of other robots, competitive collisions, when they arise, are resolved through token-passing among the colliding robots. These robots perceive their environment with the aid of the multifloor model, generating global states, global rewards, local states, and local rewards.

*Token-Passing-Based Collision Resolution:* If the aforementioned predictive planning fails to avoid competitive collisions, the colliding robots initiate a token-passing-based resolution. The resolution is collaborative, predictive, and distributed which addresses the third challenge in Section I. It is primarily facilitated by the individual robots' local critic and actor modules.

The architecture of our proposed decentralized, off-policy multiagent actor-mixed-critics algorithm is illustrated in Fig. 4. In this intelligent cyber–physical agent system approach, robots lack direct control or predictive capacity over the multifloor model, transportation tasks, and task reconfigurations and thus necessitate learning and adaptation.

Instead, upon receiving task assignments, denoted as $\mathsf{task}^m$, each robot's actor module performs planning in accordance with the multifloor model $G_{\text{floor}}$, its local critic $Q_k$, and the global critic $V_g$ modules. The outcome plans are either collision-free or entail minimal collisions and are optimized to achieve optimal or near-optimal delays and thus denoted as collision-avoidant.

To maintain clarity in this section, we persist in using $g$ and $l$ as subscripts for variables related to global and local critics, respectively, and the critic denotation does not distinguish the timeslots. A deeper exploration into the intricacies of critics and actors will be presented in subsequent sections.

### A. Learning Intelligent Agents With Global Critic and Local Critics

While the delay in task execution can be obtained only upon completion of a task, the global critic operates by taking pickup or delivery paths equivalently [referred to as paths or sequences of actions, as defined by (5)] as its

input. Subsequently, it predicts a global reward, which mirrors the contribution of the path to timely task completion and the collective objective given by (7). Conversely, the local critic predicts individual robots' actions within the action space, focusing on collision avoidance based on the current location. Consequently, these two critics operate with distinct temporal resolutions for both prediction and training, and these differences will be elaborated upon separately within this section.

*1) Shared Global Critic:* The global critic denoted as $V_g$, is a state-value function shared among all robots and is thus hosted via edge computing. This critic aims to predict the global reward corresponding to the contribution of a pickup or delivery path sourced from either a plan or a task execution, to the timely task completion and the collective objective. This is perceived through the assigned task, $\mathsf{task}^m$, and its corresponding path, $\boldsymbol{\rho}_k^m$. For the sake of simplicity, the paths are not differentiated based on pickup or delivery, as the global critic treats them equivalently.

The combined perception of $\mathsf{task}^m$ and $\boldsymbol{\rho}_k^m$ manifests as a global state $s_{g,k}^m$, distinguishing between transportation task $m$ and the specific robot-$k$. This state is represented by a numerical vector comprising six elements.

1) The initial four elements pertain to the embedded and encoded path $\boldsymbol{\rho}_k^m$.
2) The fifth element signifies the energy consumption as elaborated in (5d).
3) The sixth element denotes the remaining timeslots to execute $\mathsf{task}^m$. For planning purposes, this is equivalent to the due time as defined by (3); otherwise, it represents $\tau_{\text{due}}^m - \tau_k^m$.

Given that the count of vertices or actions within a pickup or delivery path can fluctuate based on planning or task execution specifics, embedding the path is important to establish a fixed-length numerical vector suitable for the global critic's input. Moreover, to accommodate real-time adaptive task execution, dimension reduction becomes necessary to reduce the computational complexity of the training and prediction of the global critic. Thus, a random walk-based path embedding and a pretrainable autoencoder are introduced.

A random walk-based path embedding algorithm inspired by [28] is introduced. $n_{\text{walks}}$ random walks start from a fixed set of reference vertices $\mathbf{N}_{\text{ref}}$ are performed with $n_{\text{steps}}$ steps. Similar to [28], return parameter $p_{\text{walks}}$ and in-out parameter $q_{\text{walks}}$ are implemented so that search biases are $(1/p_{\text{walks}})$, 1, $(1/q_{\text{walks}})$. These three elements are then normalized to the probabilities of returning to a previously visited vertex, visiting a local vertex (breadth-first search), and visiting a deeper vertex (depth-first search), respectively. The embedding outputs a vector where each element is the number of vertices of random walks from the corresponding reference vertex in $\mathbf{N}_{\text{ref}}$ shared with $\boldsymbol{\rho}_k^m$. This is detailed in Algorithm 1. The function `node2vecWalk` is adopted from [28]. Thus, the embedding space shares dimension with $\mathbf{N}_{\text{ref}}$ and there are $\mathbf{N}_{\text{ref}} \times n_{\text{walks}} \times n_{\text{steps}}$ possibilities. Fig. 5(a) visualizes two reference vertices taking three random walks with three steps that output embedding $(2, 5)$.

**Algorithm 1:** Random Walk-Based Path Embedding

```
def PathEmbd(G_floor, N_ref, n_walks, n_steps, p_walks, q_walks, ρ_k^m):
    Result: Embed_vector
1   Embed_vector ← ∅
2   for v in N_ref:
3       Walks ← node2vecWalk
          (G_floor, n_walks, n_steps, p_walks, q_walks)
4       counter = 0
5       for walk in Walks:
6           for vertex in ρ_k^m:
7               if vertex in walk:
8                   counter+ = 1
9       Embed_vector[v] ← counter
```

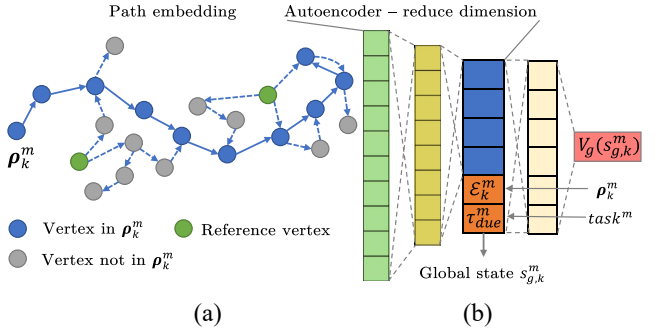



Fig. 5. (a) Example of path embedding $(\mathbf{N}_{\text{ref}}, n_{\text{walks}}, n_{\text{steps}}) = (2, 3, 3)$. (b) Example of autoencoder taking an embedded path with $\mathbf{N}_{\text{ref}} = 10$, as well as the six-element global state vector.

Autoencoder is an unsupervised machine learning technique that achieves dimension reduction by learning an efficient encoding and decoding scheme that allows it to reconstruct the input data from the reduced representation. As visualized in Fig. 5(b), the encoder part is implemented to reduce the dimension of path embedding from $\mathbf{N}_{\text{ref}}$ to four.

The time complexity of our proposed path embedding is scalable based on several factors: the number of reference vertices $\mathbf{N}_{\text{ref}}$, the count of random walks $n_{\text{walks}}$, and the step quantity $n_{\text{steps}}$. In our implementation, the autoencoder undergoes offline training using randomly generated and embedded paths, and thus, the time spent on this training has been excluded from the time complexity calculations.

In contrast to the centralized critic methods commonly found in the literature, the global reward in our approach is agent-specific. Due to inherent decentralization, it becomes challenging to predict the precise timeslot for task completion across all robots. As a result, the global reward is allocated in each timeslot. As defined in (8), robot-$k$ receives a larger reward of 10 when either a pickup or delivery is accomplished within the due time. In contrast, an overdue task receives a diminished reward of 5. Until either the pickup or delivery task is completed, robot-$k$ registers a global reward of 0

$$r_{g,k}^t = \begin{cases} 10, & \text{picked up or delivered and } \tau_k^m \le \tau_{\text{due}}^m \\ 5, & \text{picked up or delivered and } \tau_k^m > \tau_{\text{due}}^m \\ 0, & \text{otherwise.} \end{cases} \quad (8)$$

*2) Collaborative Training of Shared Global Critic:* The edge computing, which hosts the global critic, is not a cyber–physical agent and thus lacks the capability for task

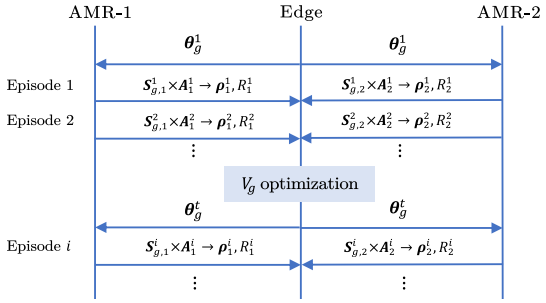



Fig. 6. Example of decentralized sample collection with two robots.

execution. As a result, transitions for training the global critic are collaboratively collected by all robots. Considering the execution of $\mathsf{task}^m$ as an episode, each robot compiles two global critic transitions—one from the pickup path and the other from the delivery path. By the episode's completion, two paths $\boldsymbol{\rho}_k^m$ and two cumulative returns $R_k^m$ [as delineated by (9a)] are collected, thus constituting two global state vectors and their corresponding global rewards. Essentially, being a state-value function, the global critic does not necessitate actions within its transitions. For the global critic, we utilize a neural network as a function approximator, characterized by its weights $\boldsymbol{\theta}_g^t$.

As illustrated in Fig. 6, the edge computing system, via robot-to-infrastructure (R2I) communications, consistently updates $\boldsymbol{\theta}_{\text{global}}$ for all robots. In turn, at the completion of each episode, every robot transmits two transitions to the edge. Consequently, the incorporation of additional robots directly augments the number of transitions available for training the global critic, rendering the training both scalable and efficient. The optimization of the global critic employs gradient ascent, as defined in (9b), utilizing mean-squared loss and experience replay with unified sampling [29]. The trajectory (or path) $\boldsymbol{\rho}_k^m$ outlined in (9a) accounts for the sequential global rewards $r_{g,k}^t$ achieved at each step

$$J(V_g) = \mathop{\mathbb{E}}_{\boldsymbol{\rho}_k^m \sim V_g}\left[R_k^m\right] = \mathop{\mathbb{E}}_{\boldsymbol{\rho}_k^m \sim V_g}\left[\sum_{t}^{t+\tau_k^m} r_{g,k}^t\right] \tag{9a}$$

$$\boldsymbol{\theta}_g^{t+1} = \boldsymbol{\theta}_g^t + lr_g \nabla_{\boldsymbol{\theta}} J(V_g). \tag{9b}$$

*3) Distributed Local Critics:* Each robot individually hosts and trains its local critics, a state–action value function in a distributed manner. Specifically, the local critic $Q_k$ directs robot-$k$ toward avoiding competitive collisions by predicting the penalty (termed as the local reward) for all possible actions within its action space $\mathcal{A}$. These predictions are based on prior experience of competitive collisions. The local state $s_{l,k}^t$ is constructed as an 8-element vector, directly relating to robot-$k$'s location at time $t$, represented as $v_{f_k,x_k,y_k}^t$. This local state formulation facilitates better generalization of the correlation between the local state and actions. Formally, $s_{l,k}^t = (f_k, x_k, y_k, s_{\text{west},k}, s_{\text{south},k}, s_{\text{east},k}, s_{\text{north},k}, s_{\text{cross},k})$, in which five binaries $s_{\text{west},k}, s_{\text{south},k}, s_{\text{east},k}, s_{\text{north},k}, s_{\text{cross},k}$ indicate either pickup location $v_{f_p,x_p,y_p}^m$ or delivery location $v_{f_d,x_d,y_d}^m$ located on left (on the $y$-axis), down (on the $x$-axis), right (in the $y$-axis), up (in the $x$-axis), and cross-floor (in floor axis) relative to $v_{f_k,x_k,y_k}^t$, respectively. The relative location is not exclusive; for example, $v_{1,3,1}$ is located on both the right side and the down side of $v_{3,2,2}$. Thus, given $G_{\text{floor}}$, the local state has the state space of $\Omega_l = N_f \cdot N_x \cdot N_y \cdot 2^4$.

Local rewards, as defined by (10), are determined based on the scenario where $K_{f_k,x_k,y_k}^t$ robots are located at $v_{f_k,x_k,y_k}$ at time $t$. Robots that execute an action conflicting with another robot or when the count of robots at a single location exceeds 3 are subjected to a penalty in the form of local rewards. Notably, when three robots are located at the same location, there is a higher probability of collision. Under the assumption that robots act randomly, the probability of collision is computed as $1 - \binom{5}{3} \cdot (1/5)^3 = 0.92$. This statistic is particularly appropriate for vertices without cross-floor neighbors

$$r_{l,k}^t = \begin{cases} -1, & \text{action cause competitive collision as (6)} \\ & \text{or } K_{f_k,x_k,y_k}^t > 3 \\ 0, & \text{otherwise.} \end{cases} \tag{10}$$

Therefore, by predicting the local reward using the optimal state–action value function $Q_k$, as elaborated in (11), the actions that avoid competitive collisions will have a larger value. In this context, $\gamma_l$ denotes the discount factor, while $s_{l,k}' \sim \boldsymbol{\rho}_k^m$ and $a'$ represent abbreviated forms of the local state and subsequent action in the next timeslot, respectively, adhering to the path $\boldsymbol{\rho}_k^m$ treated as a trajectory. In our implementation, the target network technique [30] is employed to approximate and refine $Q_k^*$, using the smooth L1 loss and a time-dependent learning rate $lr_l^t$. Again, experience replay with unified sampling is adopted to optimize local critics with fixed-length FIFO replay buffers

$$Q_k^*(s_{l,k}, a_k) = \mathop{\mathbb{E}}_{s_{l,k}' \sim \boldsymbol{\rho}_k^m}\left[r_{l,k} + \gamma_l \max_{a_k'} Q^*\left(s_{l,k}', a_k'\right)\right]. \tag{11}$$

Two fine-tuning techniques are introduced to address the challenges of distribution shift due to nonstationary tasks. First, for adaptivity, the replay buffers for local critics are reset with the task reconfiguration. The reason is that transitions in the replay buffer may be outdated and no longer reflect the stochastic characteristics of the current task set after task reconfigurations. This technique ensures that training is always performed with transitions reflecting the current set of tasks. Second, the learning rates $lr_l^t$ defined in (11) have exponential decay [31] with the discounting factor $\gamma_{\text{decay}}$ and are scheduled to reset to their initial value in the timeslots for task reconfiguration. This also explains the smaller local reward numerical values in (10) than the global reward numerical values in (8).

### *B. Distributed Actors and Predictive Collision Avoidance With Domain Model*

Actor modules are distributed across robots, enabling real-time local navigation decisions. This decentralization serves to mitigate the limitations associated with wireless latency and potential errors inherent in centralized approaches. As illustrated in Fig. 7, the actor modules operate based on the perceived task assignment. They incorporate predictions from the shared global critic $V_g$, the robot's local critic $Q_k$, the

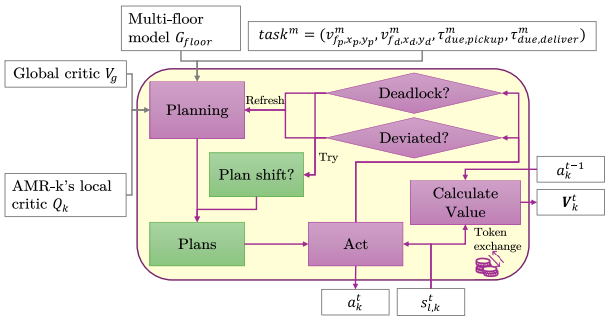



Fig. 7. Actor algorithm on robots.

shared multifloor model $G_{\text{floor}}$, and the transportation task $\mathsf{task}^m$ to ensure decision making grounded in fundamental reasoning. Notably, the off-policy optimization employed in both the global and local critics allows the actor module to perform planning and prediction for decision making at any timeslot. Algorithm 2 details the three associated algorithms in the actor module illustrated in Fig. 7.

*1) Planning With Domain Model:* An actor module is initialized using the domain model $G_{\text{floor}}$ and a predetermined plan number, denoted by $\kappa_N$, to improve computational efficiency. Upon assignment of $\mathsf{task}^m$ to robot-$k$, it employs a multipath planning method, which is an enhancement of Yen's algorithm as presented by [32]. This method is outlined as `Planning` in Algorithm 2. The heuristic termed "unfold" is graphically demonstrated in Fig. 3(b). This heuristic is instrumental in generating four acyclic subgraphs from the cyclic graph $G_{\text{floor}}$. This transformation is essential because Yen's algorithm is inherently designed for acyclic graphs. To elaborate further, when both the current location of the robot, $v_{f_k,x_k,y_k}$, and its target location, $v_{f_t,x_t,y_t}$ (which could either be a pickup vertex with coordinates $f_t = f_s, x_t = x_s, y_t = y_s$ or a delivery vertex $f_t = f_d, x_t = x_d, y_t = y_d$) reside on the same floor, the actor algorithm exclusively utilizes the subgraph of this floor for multipath planning, as depicted in lines 3–5 of Algorithm 2. Conversely, if the robot's current and target locations are spread across two distinct floors, the actor algorithm incorporates these two floors and vertices that bridge these floors at locations $x = 1$, $x = N_x$, $y = 1$, and $y = N_y$. The algorithm thus generates four acyclic subgraphs for multipath planning, as exhibited by the `Unfold` method in Algorithm 2 and formally by (12). The planning with domain model adds a greedy heuristic to the actor module since Yen's algorithm gives the shortest plans first, which makes sure robots always execute tasks with optimal or near-optimal path plans

$$\mathbf{V}_{\text{sub}} = \{(f_i, x_i, y_i) \in \mathbf{V}_{\text{floor}} | f_i = f_k \text{ or } f_i = f_t\} \tag{12a}$$

$$\mathbf{V}_{\text{sub1}} = \mathbf{V}_{\text{sub}} \cup \{(f_i, x_i, y_i) \in \mathbf{V}_{\text{floor}} | x_i = 1\} \tag{12b}$$

$$\mathbf{E}_{\text{sub1}} = \big\{((f_i, x_i, y_i), (f_j, x_j, y_j)) \in \mathbf{E}_{\text{floor}} | (f_i, x_i, y_i), (f_j, x_j, y_j) \in \mathbf{V}_{\text{sub1}}\big\} \tag{12c}$$

$$G_{\text{sub1}} = (\mathbf{V}_{\text{sub1}}, \mathbf{E}_{\text{sub1}}) \tag{12d}$$

$$\mathbf{V}_{\text{sub2}} = \mathbf{V}_{\text{sub}} \cup \{(f_i, x_i, y_i) \in \mathbf{V}_{\text{floor}} | x_i = N_x\} \tag{12e}$$

$$\ldots. \tag{12f}$$

*2) Predictive Collision Avoidance:* Next, the actor module selects $\kappa_k$ plans from a total of $\kappa_N$ plans. The selection is

**Algorithm 2:** Actor Module

```
def Planning(v_{f_k,x_k,y_k}, G_floor, task^m, V_g, Q_k, κ_N, κ_k):
    Result: plans
1   v_{f_t,x_t,y_t} ← task^m
2   if f_k == f_t:
3       V_sub = {(f_i, x_i, y_i) ∈ V_floor | f_i = f_k}
4       E_sub = {((f_i, x_i, y_i), (f_j, x_j, y_j)) ∈ E_floor | f_i = f_j = f_k}
5       G_sub1 = (V_sub, E_sub)
6   else:
7       {G_sub1, ..., G_sub4} ← Unfold (G_floor, f_k, f_t)
8   {ρ_k^m, ...} ← Yen's_algorithm
      ({G_sub, ...}, v_{f_k,x_k,y_k}, v_{f_t,x_t,y_t}, κ_N)
9   values_global ← V_g(s_{g,k}^m ← PathEmbd({ρ_k^m, ...}))
10  values_local_mean ← Mean (Q_k({s_l^k, a_k, ...} ← {ρ_k^m, ...}))
11  A_k({ρ_k^m, ...}) ← α_g·values_global+values_local_mean
12  plans ← argsort (κ_k, A_k({ρ_k^m, ...}))

def Calculate_Value(a_k^t, a_k^{t-1}, s_{l,k}^t, Q_k):
    Result: V_k^t
13  values_local ← Q_k(s_{l,k}^t, a), a ∈ A
14  values_local ← argsort (values_local)
15  for a in A:
16      if a == Reverse (a_k^{t-1}):
17          values_local(a) = 0
18      elif a == a_k^t:
19          values_local(a) = 6
20      elif a meet cross_floor_condition:
21          values_local(a) = 0
22  V_k^t ← Map (values_local, [0, 1, 2, 3, 4, 5, 6])

def Act(plan, v_{f_k,x_k,y_k}, G_floor, task^m, V_g, Q_k, κ_N, κ_k):
    Result: a_k^t
23  if plan[−10:].count(plan[−1]) > 2:
24      if plan[−10:].count(plan[−2]) > 2:
25          switch_flag ←Switch_plan (v_{f_k,x_k,y_k})
26          if ¬switch_flag:
27              if Random > 0.5:
28                  return a_k^t = 4
29  if v_{f_k,x_k,y_k} ∉ plan:
30      switch_flag ←Switch_plan (v_{f_k,x_k,y_k})
31      if ¬switch_flag:
32          Planning (v_{f_k,x_k,y_k}, G_floor, task^m, V_g, Q_k, κ_N, κ_k)
33  plan ← Get_plan (v_{f_k,x_k,y_k})
34  a_k^t ← plan
```

based on the highest advantage values computed by $A_k$, as defined in (13). This function is introduced after considering both the global and local critic evaluations of robot-$k$. The generalized advantage function, as presented by [33], offers a relative measurement of actions within the action space. In contrast, (13) provides a more nuanced representation. It encapsulates the measurement of the collective objective (7a) on the relative influence of plans on aspects, such as timely execution, delay mitigation, energy efficiency, and collision avoidance.

Drawing inspiration from TD($\lambda$) [34], for a given path plan $\boldsymbol{\rho}_k^m$, the formulation of (13) incorporates $V_g$, which predicts the additive expected return of global rewards from $\boldsymbol{\rho}_k^m$, as detailed in Section III-A1. Simultaneously, it also integrates $Q_k$, which predicts the expected values of all actions constituting $\boldsymbol{\rho}_k^m$, as detailed in Section III-A3. Nevertheless, due to the variable number of actions included within each $\boldsymbol{\rho}_k^m$, there exists an inherent bias. To mitigate this, the expected local reward values predicted by $Q_k$ are averaged over the number of actions,

denoted as $i$. In light of this, it becomes crucial to decide on an optimal coefficient, denoted as $\alpha_g$, that harmonizes the global critic value with the local critic value. This coefficient should be determined empirically, considering both the global and local reward functions

$$A_k(\boldsymbol{\rho}_k^m) = \alpha_g V_g\left(s_{g,k}^m \leftarrow \texttt{PathEmbd}(\boldsymbol{\rho}_k^m)\right) + \frac{1}{i}\sum_{s_l^k, a_k \in \boldsymbol{\rho}_k^m} Q_k\left(s_l^k, a_k \leftarrow \boldsymbol{\rho}_k^m\right). \tag{13}$$

Consequently, the actor finalizes its decision with the plan possessing the maximum advantage value, designating it as the primary policy. The remaining $\kappa_k - 1$ plans serve as alternative plans. This procedure is succinctly outlined in lines 9–12 of Algorithm 2. The formulation in (13) stands as our proposition for collaborative, predictive collision avoidance. This is primarily because all intelligent robots contribute to the training of shared global critic and the planning with domain model is predictive. However, the inherent bias in (13) is beyond the scope of this article. Our focus is not to propose it as a universal advantage function, but rather seeking near-optimal performance of the smart factory transportation problem.

*3) Deviations and Deadlocks:* As briefed in Section I, due to the path constraint, competitive collisions can occur when robots perform distributed planning. The distributed collision resolution can lead to deviation from the path plans and even deadlocks [14]. Two mechanisms are involved in the actor module described by lines 23–32 in Algorithm 2 to address these issues. First, plan switching is performed when deviation or deadlocks are detected. The actor looks for the current location $v_{f_k,x_k,y_k}$ in $\kappa_k - 1$ alternative plans and switches to the plan that includes $v_{f_k,x_k,y_k}$ ensured by function `Switch_plan`. Replanning is the second mechanism that re-executes multi-floor planning in Section III-B1 in case no policy contains $v_{f_k,x_k,y_k}$. The actor has 50% to take "stay" action and another 50% for replanning to avoid frequent replanning. in which the probability can be empirically adjusted toward better performance.

### C. Collision Resolution With Token-Passing

Under the proposed collaborative multi-intelligent robot system approach, token-passing-based collision resolution is introduced when competitive collisions are not avoided predictively. This resolution approach ingeniously addresses the balancing between collaboration and competition (competitive collision) via automated intelligent agents' communications.

Under the transportation-MRS proposed in Section II-A, although task execution is asynchronous, discrete navigation steps are synchronous among robots. When competitive collisions occur, the colliding robots resolve the collision in a distributed manner to ensure collision-free paths in task executions.

The token-passing-based collision resolution is inspired by the Dutch auction. In this setup, each robot maintains a numerical value stored locally termed $\mathsf{token}_k^t$, which effectively functions as a "currency" facilitating the action-claiming process. Every robot starts with an initial allocation of tokens, represented by the positive integer $\mathsf{token}_k^0$. Referring to Fig. 8, consider a scenario where a competitive collision is anticipated among a set of robots, exemplified by robot-1, 2, 3, 4, at a given time $t$. First, each robot leverages its actor module to compute a value vector $V_k^t$, corresponding to the action space. The computation of this vector is elaborated in Algorithm 2 under the function `Calculate_Value` (spanning lines 13–22). It ensures actions that avoid competitive collisions are assigned larger values with lines 15–21 incorporating domain-specific knowledge. A planned action, as outlined in lines 18 and 19, is assigned a value of 6. In contrast, unplanned cross-floor actions, identified by the `cross_floor_condition`, alongside the action that reverses the last action, are assigned a value of 0. The `Map` function makes sure all the integer elements in $V_k^t$ are in the range of $[0, 6]$, coherent with the action space $\mathcal{A}$.

Next, robots exchange the value vector and token numbers, as depicted in Fig. 7. This communication results in a consolidated value vector, denoted as $\mathbf{V}_{k,a}^t$, and token vector $\mathbf{token}^t$ for the distributed execution of Algorithm 3. Lines 2 and 4 emphasize that robots holding a larger quantity of tokens are given priority, allowing them to stake a claim on their most valued actions before others. In contrast, robots with fewer tokens find themselves relegated to choosing from the remaining options in `Action_set`. In situations where

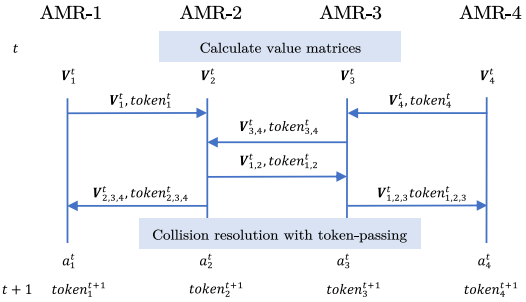



Fig. 8. Example of distributed collision resolution among three robots.

**Algorithm 3:** Collision Resolution With Token-Passing

**Data:** $\mathcal{A}, \mathbf{a}_n^t = (a_k^t, \dots), \mathbf{V}_{k,a}^t, \mathbf{token}^t = (\mathsf{token}_k^t, \dots)$

1 $\mathbf{X}^t, \texttt{prices} \leftarrow \emptyset$
2 `robot_set` $\leftarrow$ `argsort` (**token**)
3 `Action_set` $\leftarrow \mathcal{A}$
4 **for** $k$ **in** *robot_set*:
5  `desired_actions_k` $\leftarrow$ `argsort` $(\mathbf{V}_{k,a}^t)$
6  **for** $a_k$ **in** *desired_actions_k*:
7  **if** $a_k$ **in** *Action_set*:
8  $\mathbf{X}^t \leftarrow k, a_k$
9  `price` $\leftarrow \mathbf{V}_{k,a}^t$
10  **if** *price* $\geq \mathsf{token}_k$:
11  `Action_set.remove`$(a_k)$
12  `prices` $\leftarrow$ `price`
13  break
14  **if** $k$ *not* **in** $\mathbf{X}^t$:
15  $\mathbf{X}^t \leftarrow k, a_k = 4$
16  `price` $\leftarrow 0$
17 $\mathbf{token}^{t+1} \leftarrow$ `Token_exchange` $(\mathbf{X}^{n,t}, \texttt{prices}, \mathbf{token})$

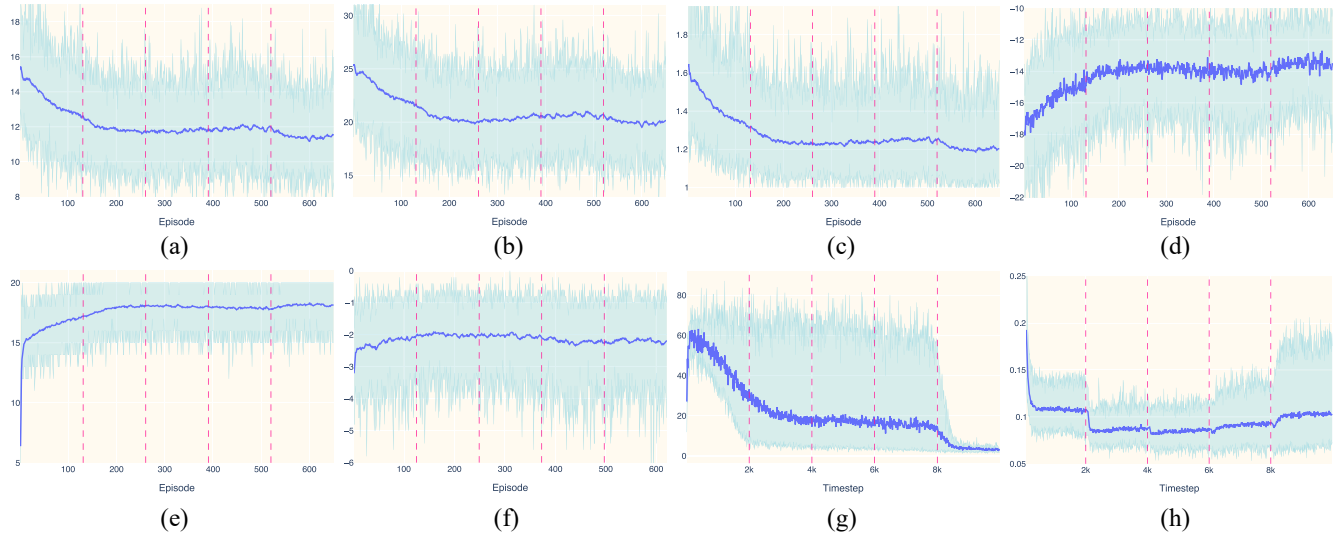



Fig. 9. Performance and training results with a 31-robot system in multifloor model defined by $(3, 3, 4)$. (a) Delay. (b) Energy consumption. (c) STR. (d) Collective objective (negative). (e) Global reward. (f) Local reward. (g) MSE loss of the global critic. (h) Smooth L1 loss of local critics.

tokens are sparse, as depicted in lines 7–13, some robots may be unable to claim any action. Line 5 underscores that robots will only opt for actions assigned a value exceeding 0, interpreting actions with a value of 0 as being "undesirable." Robots unable to claim an action are defaulted to the "stay" action, which does not cost any token, as detailed in lines 14–16. Lastly, the procedure `Token_exchange` ensures that the tokens spent by each robot are equally reallocated to other robots. Thus, the total number of tokens in the system remains constant, posing a multiplayer zero-sum game. Upon resolution, all robots engage in an exchange of derived outcomes, represented by $\mathbf{X}^t$, and their updated token counts, denoted as $\mathbf{token}^t$.

The resolution of collisions is executed in a distributed manner among the set of colliding robots. This is facilitated by a proximity-based ad hoc wireless network [35], which enhances scalability and optimizes both communication efficiency and energy consumption. In situations where multiple robots demonstrate a preference for the same actions and have equal token counts, a synchronized random seed is employed. This synchronization ensures that the outcomes, represented by $\mathbf{X}^t$, remain consistent across the robots. The collision resolution process not only contributes to the local reward, as detailed in Section III-A3 but also makes the necessary modifications to the initial actions proposed by the actor-mixed-critics algorithm. This ensures adherence to the path constraints. Furthermore, this resolution introduces a degree of variability in the critic transitions, promoting their continuity and smoothness.

## IV. Computational Experiments

To demonstrate the effectiveness, near-optimal performance, and scalability of the proposed approach, computational experiments are conducted on a set of robot numbers and a set of multifloor models, where $K \in \{13, 17, 19, 23, 29, 31\}$ and $(N_f, N_x, N_y) \in \{(2, 4, 5), (2, 5, 5), (2, 5, 6), (3, 3, 4), (3, 4, 4), (3, 4, 5)\}$. All the experiments are conducted on a single MacBook Air laptop with an 8-core M2 CPU (max clock speed of 3.5 GHz) and 16 GB of memory, to empirically demonstrate the significant potential for real-time application. We implement our approach with Python 3.8, custom autoencoder and critic networks with PyTorch 1.13, and the multifloor model with Gym 0.26.2 and NetworkX 2.8.8.

### A. Effectiveness

The experiments with 31 robots and multifloor model with $(3, 3, 4)$ (36 vertices) are carried out over 10 000 timesteps to demonstrate the effectiveness and near-optimal performance. This is because it has the most number of robots and the least number of vertices in all experiments, which naturally cause the most competitive collisions under the path constraint. During 10 000 timesteps, the task set reconfigures at timesteps of 2000, 4000, 6000, and 8000 simulating the nonstationary tasks in smart factories, which is unknown to all robots. To make the results comparable over different task sets, we control the task set with tasks that have an optimal delivery path delay of 6 so that all task sets are equivalently challenging for robots. Each experiment is repeated five times with distinct seed sets for the environment and each robot. Using the parameters in Table I, the results display the curves averaged by the number of robots and five repetitions and their range for five repetitions with shades.

The results of an experiment with 31 robots in a multifloor model defined by $(3, 3, 4)$ are shown in Fig. 9 and 11. While Fig. 9(a)–(f) have the $x$-axis of episodes (completed tasks), Figs. 11(b), 9(g) to 11(a) have the $x$-axis of timesteps (timeslots). The reason is that the delay, energy consumption, and collective objective are defined after completed tasks and both global rewards and local rewards are defined to optimize those performances. The vertical dashed lines show the timeslots of task reconfigurations (changes in task set). Since the number of timeslots (timesteps) in each episode is

TABLE I
PARAMETER VALUES OF EXPERIMENTS

| *Parameter* | *Value* | *Parameter* | *Value* |
|---|---|---|---|
| $N_f, N_x, N_y, K$ | $3, 3, 4, 31, 7$ | $p_{walks}, q_{walks}$ | $1, 1$ |
| $\varepsilon_f, \varepsilon_x, \varepsilon_y$ | $3, 1, 2$ | $\kappa_k$ | $5$ |
| $\beta$ | $0.2$ | $\epsilon$ | $1$ |
| $\alpha$ | $0.7$ | $\mathsf{token}_k^0$ | $10$ |
| $\kappa_N$ | $240$ | Training period | 10 timeslots |
| $lr_g, lr_l^k$ | $0.001, 0.0005$ | $V_g$ hidden size | $[6, 6]$ |
| $\gamma_l, \gamma_{decay}$ | $0.98, 0.90$ | $Q_k$ hidden size | $[8]$ |
| $\alpha_{global}$ | $0.1$ | $V_g$ buffer size | $5000$ |
| $n_{\text{walks}}, n_{\text{steps}}$ | $3, 3$ | $Q_k$ buffer size | $1000$ |

varied, for those with the $x$-axis of episodes, the timeslots of task reconfigurations are approximated.

The shortest time ratio (STR) defined by (14) demonstrates near-optimality performance, whose result is shown in Fig. 9(c). STR calculates the average delay ratio of each completed task ($\tau_k^m$) over the shortest possible delay ($\tau_{\text{shortest}}^m$), in which $\tau_{\text{shortest}}^m$ has a definition similar to $\tau_{\text{due}}^m$ in (3) but with the margin parameter $\beta = 0$. Also, similar to (3), $v_{f_k,x_k,y_k}$ is the location of the robot-$k$ as $\mathsf{task}^m$ is assigned to it. When STR is closer to 1, more tasks are completed with optimal paths and vice versa

$$\text{STR}^T = \frac{1}{K\sum_{t=1}^{T} M^t} \sum_{t=1}^{T} \frac{\tau_k^m}{\tau_{\text{shortest}}^m} \tag{14a}$$

$$\exists \quad \tau_k^m \leftarrow \left(\mathsf{task}^m, \boldsymbol{\rho}_{k,\text{pickup}}^m, \boldsymbol{\rho}_{k,\text{deliver}}^m\right) \tag{14b}$$

$$\tau_{\text{shortest}}^m = h_{A^*}\left(v_{f_k,x_k,y_k}, v_{f_p,x_p,y_p}^m\right) + h_{A^*}\left(v_{f_p,x_p,y_p}, v_{f_d,x_d,y_d}^m\right) \tag{14c}$$

$$m = 1, 2, \ldots, M^T. \tag{14d}$$

As illustrated in Section III-B, although planning with the domain model ensures near-optimality, robots adaptively avoid competitive collisions to coordinate task execution paths. In the first 200 episodes, as shown in Fig. 9(a)–(d), delay, energy consumption, and STR decrease significantly. Meanwhile, the collective objective shows significant improvement. In the episodes that follow episode 200, all four performances fluctuate by a small margin even though the task sets change subtly. This suggests that within the first 200 episodes, all robots learned to choose collision-avoidant plans and adapted to task reconfigurations to maintain performance. In terms of near-optimality, after 200 episodes, the STR remains around 1.23 on average, which means the average delay is 23% larger than the ideal optimal delay. This is achieved by 31 robots operating in a $G_{\text{floor}}$ with 36 vertices ($3 \times 3 \times 4$) so that on average one robot has the probability of 0.86 of sharing location with another robot.

Fig. 9(e) illustrates the average global rewards, which is consistent with the collective objective in Fig. 9(d). The reason is that the global critic and the global reward are introduced to optimize the delay and energy consumption in task execution. Fig. 9(h) depicts the average local reward that reflects the average number of competitive collisions encountered by a robot in every episode. In the first 20 episodes, there is a significant improvement from approximately $-3$ to $-2$, and subsequent episodes continue around $-2$. Again, this

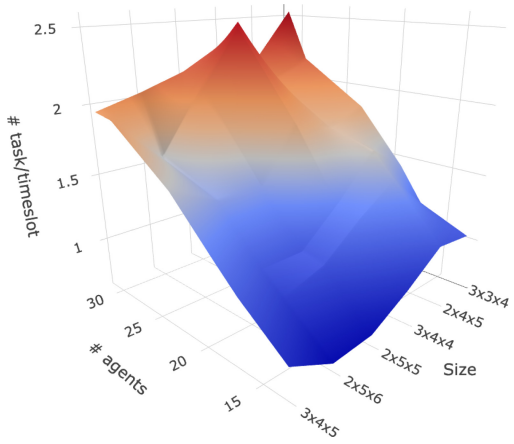



Fig. 10. Empirical scalability is demonstrated by CTPT.

demonstrates that when 31 robots operate on a $G_{\text{floor}}$ with 36 vertices, two competitive collisions are expected per episode.

Fig. 9(g) and (h) depicts the MSE loss of the global critic and the average smooth L1 loss of the local critics, respectively. After approximately 9000 timeslots, the global critic loss continues to decline and is close to 3. Even the global rewards have a rather wider range ($[0, 10]$), and the model is trained on all 31 robot transitions, resulting in a good convergence signature. Nonetheless, the average local critic loss indicated that reconfigurations make the coordination of the robot path plan challenging. After a dramatic reduction from 0.25 to 0.11 in the first 400 timeslots, each reconfiguration (shown by vertical dot lines) causes shocks to the average local critic curve. This is because reconfiguration disrupts the coordination gained by robots and introduces a distribution shift, requiring them to achieve new coordination collaboratively and adaptively. For local critics, reconfiguration results in optimal weight shifts. Thus, rapid increases or decreases in loss are observed, which is then optimized by the replay buffer reset fine-tuning mechanism.

### B. Scalability

The scalability of the proposed algorithm is empirically demonstrated by the aforementioned six $K$ values and six multifloor model definitions, which produce 36 parameter combinations and are depicted by Fig. 10. Since adaptivity and near-optimality are demonstrated by 31 robots, each parameter combination is trained for 2000 timeslots without reconfigurations to demonstrate scalability.

The $x$-axis in Fig. 10 is ordered by the total number of vertices. The $z$-axis shows the number of completed tasks per timeslot (CTPT), given by (15). It indicates that increasing the number of robots steadily increases the completion of transportation tasks, even if the shop floor may be more "crowded" and more probable competitive collisions. Additionally, enlarging the multifloor model by vertex number may lead to longer delays in task execution. Although the optimal delivery delay is fixed at 6, the pickup delay increases with the enlargement of the shop floor. Meanwhile, it leads to less probable competitive collisions. Thus, a small task completion decrease is observed

with enlarging the multifloor model by vertex number

$$\mathrm{CTPT}^T = \frac{K^2 \sum_{t=1}^{T} M^t}{\sum_{t=1}^{T} \sum_{k=1}^{K} \sum_{m=1}^{M^t} \tau_k^m}. \tag{15}$$

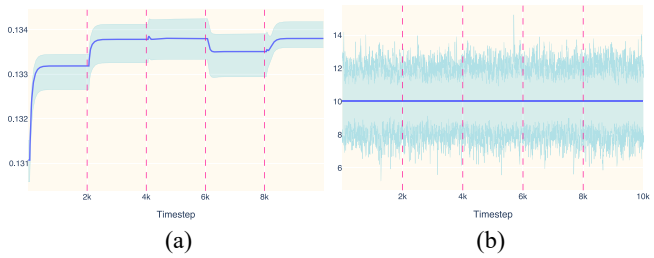
### C. Discussion

*1) Scalability and Time Complexity:* The proposed approach demonstrates robust scalability with respect to the number of robots. Several key characteristics contribute to this scalability: the decentralization allows the training and predicting of local critics and actors to be independent of the number of robots. For both kinds of critic, the prediction and training are quadratic to the number of neurons in the hidden layers and adjustable through the experience replay technique. The path embedding is linear to the number of reference vertices $\mathbf{N}_{\text{ref}}$, the number of walks $n_{\text{walks}}$, and the number of steps in each walk $n_{\text{steps}}$. The autoencoder is trained offline and the encoding has a quadratic time complexity of the number of reference vertices $\mathbf{N}_{\text{ref}}$. The $k$-shortest path algorithm employed in the actors has a time complexity of $\mathcal{O}(\kappa_N|\mathbf{V}_{\text{floor}}|(|\mathbf{E}_{\text{floor}}| + |\mathbf{V}_{\text{floor}}|\log|\mathbf{V}_{\text{floor}}|))$, which can be adjusted by $\kappa_N$. The token-passing-based collision resolution is also independent of the number of robots. Consequently, all the components present in the proposed approach are designed to be independent of the total number of robots, $K$. Furthermore, for algorithms that may exhibit more-than-linear complexity, there exists the flexibility to adjust their computational demands based on specific implementation requirements.

Although the $k$-shortest path algorithm does not have linear time complexity with the expansion of the multifloor model, from the results of size $(2, 4, 5)$, the benefit of enlarging the multifloor model on the floor axis is empirically demonstrated, as $(3, 4, 4)$ and $(3, 4, 4)$ both have a larger CTPT value. To determine the layout and scale of the robots in a smart factory, it is more efficient to add new floors than to increase the number of production robots on each floor, given the delay that can be tolerated for the production flows. Thanks to the proposed approach, introducing more robots to fulfill better efficiency also becomes an option under the path constraint.

Regarding the communication overhead of the proposed approach, each robot executes a two-hop communication protocol to the edge computing and receives the updated global critic weights in return, which is a linear communication overhead. Two-hop communication regulates collision resolution in the conceptual proximity-based ad hoc wireless network with ultralow latency communication [36]. The communication complexity indeed increases with the number of robots and subsequently the number of competitive collisions. However, the proposed collision-avoidant approach can effectively mitigate this overhead. Meanwhile, the smart factory wireless architecture can be evolved with advanced techniques such as multilink access that allows the effective coordination and utilization of wireless resources across the shop floors or geographical zones, significantly reducing the latency associated with multiple access [37].

*2) Collaboration:* The proposed approach achieves collision-avoidant paths in a predictive manner in time-varying transportation tasks. To demonstrate this, taking local critics as distributions over $\mathcal{A}$ given $s_{l,k}^t \in \Omega_l$, the variance of KL divergence $\mathrm{Var}_{\mathrm{KLDiv}}^t$ is calculated by (16). $D_{\mathrm{KL}}(Q_i^t \| Q_j^t)$ is the KL divergence of local critics of robot-$i$ and robot-$j$ at timeslot $t$. Greater values of $\mathrm{Var}_{\mathrm{KLDiv}}^t$ suggest that, given the same local state, local critics of all 31 robots tend to take different actions and vice versa. Since local critics are encouraging robots to avoid competitive collisions (taking different actions), Fig. 11(a) displays a significant increase in the first 400 timeslots. However, reconfigurations break coordination, and robots need to balance between acting "too different" and "similar." Similarly to the fluctuated average local critic loss, each reconfiguration causes a sudden increase or decrease in $\mathrm{Var}_{\mathrm{KLDiv}}^t$




(a) (b)

Fig. 11. Additional experiment results with a 31-robot system in multifloor model defined by $(3, 3, 4)$. (a) Variance of KL divergence. (b) Tokens (time average).

$$\begin{aligned}\mathrm{Var}_{\mathrm{KLDiv}}^t = \mathrm{Var}\big(&D_{\mathrm{KL}}\big(Q_1^t \| Q_1^t\big), \ldots \\ &D_{\mathrm{KL}}\big(Q_1^t \| Q_K^t\big), \ldots \\ &D_{\mathrm{KL}}\big(Q_K^t \| Q_K^t\big)\big).\end{aligned} \tag{16}$$

Furthermore, the implemented token-passing-based collision resolution treats competitive collision as multiplayer zero-sum games [38] since Fig. 11(b) shows the average token number remains the same all the time, which means one robot's pay is other robots' gain. The modeling and utility optimization are out of the scope of this article since the proposed approach is not game-theoretic.

*3) Answer to the Challenges:* The results confirm that intelligent robot systems execute adaptive, collision-avoidant, near-optimal paths, optimizing both delay and energy usage in a collaborative and predictive manner. The collective objective definition, supported by Fig. 9(a)–(d), addresses the first challenge of Section I, demonstrating optimized outcomes and the near-optimality of delay. The second challenge is tackled by the decentralized actor-mixed-critics approach leveraging robots' mobile computing. The third challenge is met through collaborative collision resolution emphasizing the communication among robots. The robots are capable of autonomous perception, reasoning, planning, and decision making facilitated by the graphical domain knowledge model—"multifloor transportation model" to avoid collisions and communicate to resolve collisions. Consequently, numerical analyses and discussions affirm the effectiveness and scalability of the proposed approach, positioning it as a solution to smart factory transportation challenges that emphasize adaptability and resource efficiency. Although the MDP is widely employed

in the literature for multiagent sequential decision making, its application in modeling smart factory transportation remains open. The dynamism of reconfiguration timeslots, $t_1, t_2, \ldots,$ and the fluctuating nature of transportation tasks, $M^t$, are challenging to predict based on executive decisions and varying customer demands. Such dynamics either violate the Markov property, as highlighted in [39, Ch. 2.2.3], or result in a high-dimensional MDP, challenging the mobile computing capacities of robots, as discussed in [40]. Comparisons with general MDP-model methods, including QMIX [41] and MADDPG [42], are conducted, and observed struggle with our multifloor model. Potential adaptations for these methods remain unexplored and subject to future research.

## V. Conclusion

This research introduces a novel smart factory transportation problem, including nonstationary transportation tasks, a multifloor transportation model, and decentralized intelligent robot operation. The problem is formulated as transportation-MRS discrete navigations and robots are modeled as a cyber–physical AI agent system. To address the challenges of smart factory transportation, utilizing automated AI cognitions, including reasoning, predicting, planning, and decision making, as well as mobile computing, and wireless communications, a multiagent actor-mixed-critics algorithm and a wireless-enabled, token-passing-based collision resolution are proposed. This approach takes a multifloor model as a stepping stone to AI planning to ensure near-optimality with a short learning curve. The algorithmic design considers time complexity and scalability, which is essential for MRS-driven smart factories. Experiments demonstrate the adaptability to nonstationary transportation tasks and collaborative, predictive path coordination among robots, both of which contribute to the system's performance. Empirical scalability demonstrates that, although adding more robots to the system increases the competition, the MRS can still autonomously, and adaptively optimize the system performance. Our immediate future work is focused on systematically analyzing scalability beyond the initial empirical demonstrations. In the longer term, our aim is to achieve full integration with production demands and the operational framework of production robots. This step is crucial to develop more realistic models of production flows. By addressing these areas, we aim to enhance the practical applicability and efficiency of our approach in real-world smart factory settings.

## References


[1] K. C. Chen, S. C. Lin, J. H. Hsiao, C. H. Liu, A. F. Molisch, and G. P. Fettweis, "Wireless networked multirobot systems in smart factories," *Proc. IEEE*, vol. 109, no. 4, pp. 468–494, Apr. 2021.

[2] Y. Chen, F. Zhao, and Y. Lou, "Interactive model predictive control for robot navigation in dense crowds," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 52, no. 4, pp. 2289–2301, Apr. 2022.

[3] Y. Liu et al., "Dynamic lane-changing trajectory planning for autonomous vehicles based on discrete global trajectory," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 7, pp. 8513–8527, Jul. 2022.

[4] Z. Nie and K.-C. Chen, "Hypergraphical real-time multirobot task allocation in a smart factory," *IEEE Trans. Ind. Informat.*, vol. 18, no. 9, pp. 6047–6056, Sep. 2022.

[5] S. K. Jagatheesaperumal, M. Rahouti, K. Ahmad, A. Al-Fuqaha, and M. Guizani, "The duo of artificial intelligence and big data for industry 4.0: Applications, techniques, challenges, and future research directions," *IEEE Internet Things J.*, vol. 9, no. 15, pp. 12861–12885, Aug. 2022.

[6] A. Balachandran, S. Anil Lal, and P. Sreedharan, "Autonomous navigation of an AMR using deep reinforcement learning in a warehouse environment," in *Proc. IEEE 2nd Mysore Sub Sect. Int. Conf. (MysuruCon)*, 2022, pp. 1–5.

[7] S. S. Abosuliman and A. O. Almagrabi, "Routing and scheduling of intelligent autonomous vehicles in industrial logistics systems," *Soft Comput.*, vol. 25, pp. 11975–11988, Sep. 2021.

[8] W. Xia, J. Goh, C. A. Cortes, Y. Lu, and X. Xu, "Decentralized coordination of autonomous agvs for flexible factory automation in the context of industry 4.0," in *Proc. IEEE 16th Int. Conf. Autom. Sci. Eng. (CASE)*, 2020, pp. 488–493.

[9] Z. Liu, Y. Zhang, C. Yuan, and J. Luo, "Adaptive path following control of unmanned surface vehicles considering environmental disturbances and system constraints," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 51, no. 1, pp. 339–353, Jan. 2021.

[10] J. Zhao, W. Li, C. Hu, G. Guo, Z. Xie, and P. K. Wong, "Robust gain-scheduling path following control of autonomous vehicles considering stochastic network-induced delay," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 12, pp. 23324–23333, Dec. 2022.

[11] M. Pei, H. An, B. Liu, and C. Wang, "An improved Dyna-Q algorithm for mobile robot path planning in unknown dynamic environment," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 52, no. 7, pp. 4415–4425, Jul. 2022.

[12] K. V. Sagar and J. Jerald, "Real-time automated guided vehicles scheduling with Markov decision process and double Q-learning algorithm," *Mater. Today, Proc.*, vol. 64, pp. 279–284, Jan. 2022.

[13] Z. Chen, J. Alonso-Mora, X. Bai, D. D. Harabor, and P. J. Stuckey, "Integrated task assignment and path planning for capacitated multi-agent pickup and delivery," *IEEE Robot. Autom. Lett.*, vol. 6, no. 3, pp. 5816–5823, Jul. 2021.

[14] T. Yamauchi, Y. Miyashita, and T. Sugawara, "Standby-based deadlock avoidance method for multi-agent pickup and delivery tasks," in *Proc. 21st Int. Conf. Auton. Agents Multiagent Syst.*, 2022, pp. 1427–1435.

[15] M. Kneissl, A. K. Madhusudhanan, A. Molin, H. Esen, and S. Hirche, "A multi-vehicle control framework with application to automated valet parking," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 9, pp. 5697–5707, Sep. 2021.

[16] M. Autili, L. Chen, C. Englund, C. Pompilio, and M. Tivoli, "Cooperative intelligent transport systems: Choreography-based urban traffic coordination," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 4, pp. 2088–2099, Apr. 2021.

[17] I. Ahmed, G. Jeon, and F. Piccialli, "From artificial intelligence to explainable artificial intelligence in industry 4.0: A survey on what, how, and where," *IEEE Trans. Ind. Informat.*, vol. 18, no. 8, pp. 5031–5042, Aug. 2022.

[18] H. Ghorbel et al., "SOON: Social network of machines to optimize task scheduling in smart manufacturing," in *Proc. IEEE 32nd Annu. Int. Symp. Pers., Indoor Mobile Radio Commun. (PIMRC)*, 2021, pp. 1–6.

[19] H. Shi, J. Li, M. Liang, M. Hwang, K.-S. Hwang, and Y.-Y. Hsu, "Path planning of randomly scattering waypoints for wafer probing based on deep attention mechanism," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 53, no. 1, pp. 529–541, Jan. 2023.

[20] R. Reijnen, Y. Zhang, W. Nuijten, C. Senaras, and M. Goldak Altgassen, "Combining deep reinforcement learning with search heuristics for solving multi-agent path finding in segment-based layouts," in *Proc. IEEE Symp. Series Compu. Intell. (SSCI)*, 2020, pp. 2647–2654.

[21] Z. Liu, B. Chen, H. Zhou, G. Koushik, M. Hebert, and D. Zhao, "MAPPER: Multi-agent path planning with evolutionary reinforcement learning in mixed dynamic environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2020, pp. 11748–11754.

[22] X. Liu, H. Zhang, J. Lin, X. Chen, Q. Chen, and N. Mao, "A queuing network model for solving facility layout problem in multifloor flow shop," *IEEE Access*, vol. 10, pp. 61326–61341, 2022.

[23] Z. Nie and K.-C. Chen, "Distributed coordination by social learning in the multi-robot systems of a smart factory," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, 2021, pp. 1–6.

[24] M. Rabin, "Inference by believers in the law of small numbers," *Quart. J. Econ.*, vol. 117, no. 3, pp. 775–816, 2002.

[25] D. Xia, C. Jiang, J. Wan, J. Jin, V. C. Leung, and M. Martínez-García, "Heterogeneous network access and fusion in smart factory: A survey," *ACM Comput. Surv.*, vol. 55, no. 6, pp. 1–31, 2022.

[26] Y. Du, J. Li, C. Li, and P. Duan, "A reinforcement learning approach for flexible job shop scheduling problem with crane transportation and setup times," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 4, pp. 5695–5709, Apr. 2024.
[27] Indrawati, F. M. Puspita, E. Yuliza, E. Susanti, S. Octarina, and I. Lestari, "Information services financing scheme model with marginal costs and supervisory costs for modified Cobb–Douglas and linear utility functions," in *Proc. 5th Int. Seminar Res. Inf. Technol. Intell. Syst. (ISRITI)*, 2022, pp. 799–804.
[28] A. Grover and J. Leskovec, "Node2vec: Scalable feature learning for networks," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, 2016, pp. 855–864.
[29] W. Fedus et al., "Revisiting fundamentals of experience replay," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 3061–3071.
[30] J. Fan, Z. Wang, Y. Xie, and Z. Yang, "A theoretical analysis of deep Q-learning," in *Proc. Learn. Dyn. Control*, 2020, pp. 486–489.
[31] N. Agarwal, S. Goel, and C. Zhang, "Acceleration via fractal learning rate schedules," in *Proc. 38th Int. Conf. Mach. Learn.*, 2021, pp. 87–99. [Online]. Available: https://proceedings.mlr.press/v139/agarwal21a.html
[32] A. Al Zoobi, D. Coudert, and N. Nisse, "Space and time trade-off for the k shortest simple paths problem," in *Proc. 18th Int. Symp. Exp. Algorithms*, 2020, p. 13. [Online]. Available: https://hal.inria.fr/hal-02865918
[33] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2016, pp. 1–14.
[34] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.
[35] R. Mirsky et al., "A survey of Ad Hoc teamwork: Definitions, methods, and open problems," in *Proc. Eur. Conf. Multiagent Syst.*, 2022, pp. 1–19.
[36] K.-C. Chen, T. Zhang, R. D. Gitlin, and G. Fettweis, "Ultra-low latency mobile networking," *IEEE Netw.*, vol. 33, no. 2, pp. 181–187, Mar./Apr. 2019.
[37] Z. Nie, K.-C. Chen, and Y. Alanezi, "Socially networked multi-robot system of time-sensitive multi-link access in a smart factory," in *Proc. IEEE Int. Conf. Commun.*, 2023, pp. 4918–4923.
[38] L. Zhao and K.-C. Chen, "The game theoretic consensus in a networked multi-agent system," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, 2018, pp. 1–7.
[39] W. B. Powell, *Reinforcement Learning and Stochastic Optimization: A Unified Framework for Sequential Decisions*. Hoboken, NJ, USA: Wiley, 2022.
[40] B. Yuan, J. Wang, P. Wu, and X. Qing, "IoT malware classification based on lightweight convolutional neural networks," *IEEE Internet Things J.*, vol. 9, no. 5, pp. 3770–3783, Mar. 2022.
[41] T. Rashid, M. Samvelyan, C. S. De Witt, G. Farquhar, J. Foerster, and S. Whiteson, "Monotonic value function factorisation for deep multi-agent reinforcement learning," *J. Mach. Learn. Res.*, vol. 21, no. 1, pp. 7234–7284, 2020.
[42] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch, "Multi-agent actor–critic for mixed cooperative–competitive environments," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 6382–6393.

**Zixiang Nie** (Member, IEEE) received the B.E. degree in electronic information engineering from the Beijing University of Technology, Beijing, China, in 2016, the M.S. degree in electrical and computer engineering from the University of Florida, Gainesville, FL, USA, in 2019, and the Ph.D. degree in electrical engineering from the University of South Florida, Tampa, FL, USA, in 2023.

He started a co-op at the Nokia Cloud Network Services, Dallas, TX, USA, from May 2022 to August 2023 and working full-time as a Data Scientist since 2024. His research involves autonomous AI agent system solutions to wireless networked multiagent systems in the context of smart factories and his research interests include domain-specific artificial intelligence, autonomous machine intelligence, multiagent systems, and multiagent reinforcement learning.

**Kwang-Cheng Chen** (Fellow, IEEE) received the B.S. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, in 1983, and the M.S. and Ph.D. degrees in electrical engineering from the University of Maryland at College Park, College Park, MD, USA, in 1987 and 1989, respectively.

His early career from 1987 to 1998 included positions with SSE, COMSAT, IBM, New York, NY, USA, and National Tsing Hua University, Hsinchu, Taiwan. He served as a Distinguished Professor with National Taiwan University from 1998 to 2016, taking various leadership roles. Since 2016, he has been a Professor of Electrical Engineering with the University of South Florida, Tampa, FL, USA. His research encompasses wireless networks, multirobot systems, IoT, social networks, data analytics, and cybersecurity.

Prof. Chen's accolades include the 2011 IEEE COMSOC WTC Recognition Award, the 2014 IEEE Jack Neubauer Memorial Award, and the 2014 IEEE COMSOC AP Outstanding Paper Award. He actively participates in IEEE conference organization and journal editorship. He has also made important contributions to numerous wireless standards, such as IEEE 802, Bluetooth, LTE, 5G-NR, and ITU-T FG ML5G.