

From Real-Time Planning to Reliable Execution: Scalable Coordination for Heterogeneous Multi-Robot Fleets in Industrial Environments

Bo Cao, Zhe Liu, and Hesheng Wang, *Senior Member, IEEE*

Abstract—With the increasing deployment of heterogeneous robot fleets in industrial environments, efficient coordination remains a critical challenge. Real-time path planning must simultaneously accommodate high robot densities and heterogeneous motion capabilities, while communication delays, execution uncertainties, and other disturbances may cause robots to deviate from the temporal assumptions underlying planned paths. Such deviations can lead to excessive waiting and congestion propagation across the fleet. This paper presents SCALE, a reactive on-line coordination framework that enables real-time planning while maintaining robust execution. Within this framework, we introduce a motion-induced conflict reduction mechanism to support the online generation of feasible paths for online conflict resolution. To mitigate the effects of disturbances, we further design a generalized Conjugate Action-Precedence Hypergraph (CAPH) that adaptively adjusts precedence relations among robots. Extensive validation experiments, together with a three-day deployment in a warehouse, demonstrate the practical feasibility and effectiveness of the proposed method.

Index Terms—Heterogeneous multi-robot systems, multi-robot path planning (MRPP), robust execution, planning and improving while executing

I. INTRODUCTION

INTELLIGENT manufacturing systems and warehousing logistics are increasingly operated by unmanned robot fleets, including automated guided vehicles (AGVs) such as Kiva-like robots, autonomous forklifts, mobile manipulators, and large stacker robots. Compared with homogeneous mobile-robot teams, such fleets differ substantially in footprint and kinematic behavior. Coordinating them in dense workspaces is therefore not merely a path-planning problem: the system must generate feasible routes in real time, avoid collisions and deadlocks in limited space, and maintain robust execution despite communication latency, tracking errors, and uncertain waiting caused by humans or third-party facilities.

In today’s industrial multi-robot deployments, coordination is commonly built on well-designed topological roadmaps or traffic zones. Collision avoidance and deadlock prevention are typically implemented by engineering-oriented methods such as zone control [1], Petri-net supervisors [2], glued-node reservation [3], or other handcrafted traffic rules. These methods are useful in specific scenarios because their safety logic is explicit and easy to certify. However, when dozens of tasks are involved, scheduling can take tens of seconds, which is not efficient enough for industrial systems. More

importantly, for large-scale robot fleets, such strategies tend to be conservative: zones, guidepath segments, or reserved vertices often have to be allocated according to worst-case occupancy, reducing space utilization and degrading system efficiency in high-density scenarios [3]. Their performance also strongly depends on manually designed layouts and rules, which severely limits their generality.

Recent advances in multi-agent path finding (MAPF) offer the possibility of more general planning. Representative algorithms, such as MAPF-LNS2 [4], LaCAM, and its improved variant LaCAM* [5], [6], have demonstrated strong scalability and solution quality in MAPF benchmarks. Nevertheless, most MAPF solvers rely on a graph-agent abstraction, in which each agent occupies a vertex or traverses an edge at a discrete time step. This creates a substantial gap when applied to real-world heterogeneous robots: the occupied space of a robot is determined not only by its vertex position, but also by its rigid-body footprint, orientation, and kinematic constraints. Although some approaches [7], [8] partially relax the point-agent assumption, they still struggle to preserve the real-time capability and scalability of the original algorithms. Moreover, in industrial environments, common disturbances, including occasional communication failures and human-robot interactions, can cause path execution to deviate from nominal discrete timing, compromising both efficiency and safety. Robust MAPF methods reserve temporal margins [9] or adjust precedence relations during execution [10], [11]. However, these methods still characterize inter-robot interactions primarily through pairwise vertex-occupancy conflicts or simplified footprint representations, limiting their applicability to heterogeneous robots.

To address the challenges above, we present **SCALE**, a Scalable Coordination Architecture with Latency-resilient Execution for large-scale heterogeneous robot fleets. It adopts an online reactive planning-execution architecture that groups potentially conflicting robots, generates locally feasible paths, and releases safe path segments through an execution authorization module under external disturbances. The main contributions of this work are as follows: **1)** a generalized coordination framework for large-scale heterogeneous robot fleets; **2)** a motion-induced conflict reduction mechanism that enables real-time, kinematically feasible multi-robot path planning (MRPP); **3)** a generalized structure called Conjugate Action-Precedence Hypergraph (CAPH) for robust and latency-resilient asynchronous execution; and **4)** extensive

simulations and real-world deployments in industrial scenarios with AGVs and forklifts, demonstrating sustained collision- and deadlock-free operation and a more than 30% improvement in throughput. Compared with traditional scheduling methods that separate route planning from execution control, SCALE models fleet-level path planning and asynchronous cooperative execution as an integrated online process, enabling scalable planning and improving while executing under unavoidable industrial uncertainties.



Fig. 1: A typical industrial warehouse involving heterogeneous robot fleets.

II. RELATED WORK

A. Scheduling of Industrial Multi-Robot Fleets

Multi-robot scheduling has been extensively studied. Existing industrial approaches often rely on handcrafted heuristics tailored to specific scenarios. For example, [12] adopts purely rule-based scheduling, while [1], [2] require predefined constraints under particular road-network structures. These methods can be efficient but are limited in generality. [3] provides a unified representation of inter-robot conflicts, and [13] proposes a more general unified scheduling framework; however, both lack scalability. Recently, MAPF-based methods have shown strong potential for coordination of large-scale robot fleets. [14], [15] enable real-time planning for hundreds of physical robots, but they typically simplify robot dynamics and may require structured roadmaps or well-formed conditions to ensure feasible planning. In contrast, the paper proposes a general and scalable lifelong scheduling framework for heterogeneous robot fleets.

B. Multi-Robot Path Planning

Multi-robot path planning provides robot fleets with a degree of general planning capability. Recent progress in MAPF under the point-agent assumption has substantially improved the scalability of planning algorithms. For example, [4] employs large neighborhood search to enhance the scalability of classical search-based methods such as [16]–[18]. Another line of work, including [5], [6], [19], seeks feasible solutions through priority inheritance among agents, enabling planning for tens of thousands of agents within seconds. Several studies, such as [7], [20], [21], have attempted to incorporate more realistic assumptions into discrete planning, while [22] extends LaCAM to continuous motion-space search. However, they cannot perform real-time planning, which is unacceptable in industrial applications. Some of them also adopt overly idealized robot models, such as representing robots of different sizes as disks, leading to a substantial gap between their assumptions and real-world physical constraints.

C. Robust Execution of Planned Paths

For robust collaborative path execution, some approaches reserve temporal slack during planning so that bounded delays can be safely accommodated, as in k -robust and p -robust MAPF [9], [23]. In [24], stable execution of multi-robot plans is achieved by optimizing robot velocities. These formulations grow increasingly conservative as the delay margin increases and are generally inadequate when the system occasionally experiences extreme crashes or large delays. Other works focus on precedence relations among actions of robots. For example, ADG [10] enforces a fixed ordering of execution admission, while time-independent planning works on a globally synchronized clock [25]. More recently, SADG [11] extends ADG with switchable dependencies, adjusting the execution order to reduce unnecessary waiting. In addition, methods such as [26] couple planning and execution, where a committed path segment is executed while the remaining plan is improved online. However, these methods still primarily reason over pairwise topological dependencies or simplified vehicle models. For heterogeneous robots, a more realistic and efficient method is required to handle unavoidable disturbances.

III. COORDINATING HETEROGENEOUS ROBOT FLEETS

We formulate the coordination of heterogeneous robot fleets from two coupled perspectives. The first is *fleet-level path planning*, which computes a kinematically feasible path for each robot. The second is *robust path execution*, which regulates the release of the planned paths through action-level precedence constraints. Let the shared coordination space be represented by an undirected graph $G = (V, E)$, where V is a finite set of vertices and $E \subseteq V \times V$ is the set of topological adjacencies. The robot fleet is denoted by $\mathcal{R} = \{r_1, r_2, \dots, r_N\}$. Each robot $i \in \mathcal{R}$ has a type $\tau_i \in \mathcal{T}$, which determines physical properties like its footprint and turning characteristics. For robot i , a path is defined as

$$p_i = ((v_i^1, \rho_i^1), (v_i^2, \rho_i^2), \dots, (v_i^{L_i}, \rho_i^{L_i})) \in (V \times \mathbb{N})^{L_i},$$

where $v_i^k \in V$ is the k -th vertex and $\rho_i^k \in \mathbb{N}$ is its nominal precedence rank. A smaller rank indicates earlier execution. Unlike the discrete time index in step MAPF, ρ_i^k is not a discrete timestamp but an ordering over vertices. Every consecutive vertex pair for a robot corresponds to an edge transition: $\{v_i^k, v_i^{k+1}\} \in E$ for all $k = 1, \dots, L_i - 1$. If p_i satisfies the kinematics constraints of robot i , it induces an action sequence $\sigma_i = (a_i^1, a_i^2, \dots, a_i^{L_i-1})$, where $a_i^k = (i, k, v_i^k, v_i^{k+1}, \gamma_i^k)$ denotes the continuous motion from v_i^k to v_i^{k+1} . Here $\gamma_i^k(\lambda) \rightarrow SE(2)$ denotes the trajectory corresponding to the motion, and $\lambda \in [0, 1]$ is a normalized parameter. The fleet-level path plan is $\mathcal{P} := \{p_i\}_{i \in \mathcal{R}}$, and the induced action set is $\mathcal{A}(\mathcal{P}) := \bigcup_{i \in \mathcal{R}} \{a_i^k \mid k = 1, \dots, L_i - 1\}$.

A. Heterogeneous Multi-Robot Path Planning

Given start vertices $\mathbf{v}^s = (v_1^s, \dots, v_N^s)$ and goal vertices $\mathbf{v}^g = (v_1^g, \dots, v_N^g)$, the heterogeneous multi-robot path planning problem is to find a plan $\mathcal{P} = \{p_i\}_{i \in \mathcal{R}}$ such that $v_i^1 = v_i^s$ and $v_i^{L_i} = v_i^g$ without any conflicts. Each path must be rank-consistent and feasible under the conflict model defined below.

Classical MAPF considers two topological conflicts: vertex conflicts and swap conflicts. Following this convention, for two actions a_i^k and a_j^ℓ with $i \neq j$, we set $\text{TC}(a_i^k, a_j^\ell) = 1$ if either $v_i^{k+1} = v_j^{\ell+1}$, which denotes two actions entering the same vertex, or $v_i^k = v_j^{\ell+1}$ and $v_i^{k+1} = v_j^\ell$, which denotes two actions traversing the same edge in opposite directions. Obviously, topological conflicts are insufficient for robots. Let $\mathcal{B}_{\tau_i} \subset \mathbb{R}^2$ denote the body-frame footprint of robot type τ_i . For action a_i^k , let $\gamma_i^k(\lambda) = (x_i^k(\lambda), y_i^k(\lambda), \theta_i^k(\lambda))$ be its continuous pose, and let $R(\theta)$ be the planar rotation matrix. The swept region of a_i^k is defined as

$$\mathcal{W}_\epsilon(a_i^k) := \bigcup_{\lambda \in [0,1]} \left(R(\theta_i^k(\lambda)) \mathcal{B}_{\tau_i} + \begin{bmatrix} x_i^k(\lambda) \\ y_i^k(\lambda) \end{bmatrix} \right) \oplus \mathbb{B}_\epsilon, \quad (1)$$

where $\oplus$ denotes the Minkowski sum and $\mathbb{B}_\epsilon = \{\pi \in \mathbb{R}^2 \mid \|\pi\|_2 \leq \epsilon\}$ is a safety margin with radius $\epsilon \geq 0$. A motion-induced conflict is defined as $\text{MC}(a_i^k, a_j^\ell) = 1 \iff \mathcal{W}_\epsilon(a_i^k) \cap \mathcal{W}_\epsilon(a_j^\ell) \neq \emptyset, i \neq j$. This definition captures collisions caused by robots with different footprints and nonholonomic constraints. The overall conflict model is $\text{C}(a_i^k, a_j^\ell) = 1$ if $\text{MC}(a_i^k, a_j^\ell) = 1$ or $\text{TC}(a_i^k, a_j^\ell) = 1$.

B. Hybrid Action-Precedence Graph for MRPP

Due to asynchronous execution and external disturbances in practice, the planned paths cannot always be executed in strict accordance with the discrete rank ordering. Drawing on the action-dependency in ADG [10], we introduce a generalized representation, termed Hybrid Action-Precedence Graph (HAPG). It defines action-level dependencies based on physical interactions among robots rather than vertex occupancy.

Definition III.1 (Hybrid Action-Precedence Graph). Given a set of paths $\mathcal{P}$ and its induced action set $\mathcal{A}$, the HAPG is a directed graph $\mathcal{G}_a = (\mathcal{A}, \mathcal{E})$, where $\mathcal{E} \subseteq \mathcal{A} \times \mathcal{A}$. A directed edge $(b, c) \in \mathcal{E}$ indicates that c can be executed only after b has been completed. There are two kinds of edges: $\mathcal{E} = \mathcal{E}_{\text{self}} \cup \mathcal{E}_{\text{inter}}$, where

$$\mathcal{E}_{\text{self}} := \{(a_i^k, a_i^{k+1}) \mid i \in \mathcal{R}, k = 1, \dots, L_i - 2\},$$

$$\mathcal{E}_{\text{inter}} := \{(a_i^k, a_j^\ell) \mid i \neq j, \text{C}(a_i^k, a_j^\ell) = 1, a_i^k \prec_\rho a_j^\ell\}.$$

Here $\mathcal{E}_{\text{self}}$ preserves the path ordering of each robot, whereas $\mathcal{E}_{\text{inter}}$ assigns precedence relations to conflicting actions. Specifically, $a_i^k \prec_\rho a_j^\ell$ if $\rho_i^k \leq \rho_j^\ell \wedge \text{C}(a_i^k, a_j^\ell) = 1$. See Algorithm 1 for details.

Let $\{\text{staged}, \text{in-progress}, \text{completed}\}$ be the state set. The action state is initialized as $\xi(a) = \text{staged}$. For a , define $\text{Pred}(a) = \{a' \in \mathcal{A} \mid (a', a) \in \mathcal{E}\}$. The action can enter in-progress only when every $a' \in \text{Pred}(a)$ satisfies $\xi(a') = \text{completed}$, and turns to completed after the action is finished.

Lemma III.2. Let $\mathcal{P}$ be a feasible rank-consistent MRPP plan. If each action is admitted only after all its HAPG predecessors are completed, then the HAPG execution is collision-free and deadlock-free.

Proof. For any two inter-robot actions b and c , $\text{C}(b, c) = 0$ indicates the absence of collision. If $\text{C}(b, c) = 1$, HAPG includes an inter-robot precedence edge between them: one action can start only after the other has completed, so the

Algorithm 1 Hybrid Action-Precedence Graph Construction

Input: Fleet-level path plan $\mathcal{P} = \{p_i\}_{i \in \mathcal{R}}$
Output: Hybrid action-precedence graph $\mathcal{G}_a = (\mathcal{A}, \mathcal{E})$

```

1:  $\mathcal{A} \leftarrow \emptyset, \mathcal{E} \leftarrow \emptyset$ 
2: for all  $i \in \mathcal{R}$  do
3:   for  $k = 1$  to  $L_i - 1$  do
4:      $a_i^k = (i, k, v_i^k, v_i^{k+1}, \gamma_i^k)$ 
5:      $\mathcal{A} \leftarrow \mathcal{A} \cup \{a_i^k\}$ 
6:     if  $k > 1$  then
7:        $\mathcal{E} \leftarrow \mathcal{E} \cup \{(a_i^{k-1}, a_i^k)\}$ 
8:     end if
9:   end for
10: end for
11: for all  $a_i^k, a_j^\ell \in \mathcal{A}$  with  $i < j$  do
12:   if  $\text{C}(a_i^k, a_j^\ell) = 1$  then
13:      $\mathcal{E} \leftarrow (a_i^k, a_j^\ell)$  if  $a_i^k \prec_\rho a_j^\ell$ ; otherwise  $\mathcal{E} \leftarrow (a_j^\ell, a_i^k)$ 
14:   end if
15: end for
```

conflicting swept regions never overlap in time. The self edges serialize the actions of each robot. So the execution is collision-free. Rank consistent paths induces a strict ordering among all actions. Therefore every edge in $\mathcal{E}$ points from an earlier action to a later action in this order. As long as the path plan is feasible, no action can precede itself in the induced temporal ordering. Thus HAPG is acyclic, which means it is deadlock-free. $\square$

IV. MAIN APPROACHES

The overall coordination framework proposed is illustrated in Fig. 2. It consists of two main modules: an online reactive path planning module and a resilience-guaranteed path execution module. The former generates feasible path sequences for each robot over a graph-based road network, while the latter determines the subsequent paths authorization locally.

In industrial robot fleets scheduling, paths generated by edge-side devices are not dispatched to robots as complete routes. Dynamic task arrivals, heterogeneous kinematic behaviors, and asynchronous execution make long-horizon path commitment difficult to control in practice. We therefore adopt a bounded path-release strategy: in each scheduling round, at most K steps of the planned path are committed to each robot, where K is an empirical commitment horizon. A released prefix is required to be executable to its end without collision or deadlock. For candidate local groups whose paths are bounded by this horizon, the execution authorization layer further uses CAPH in Sec. IV-B to release paths under action-precedence constraints. The next planning step starts from the terminal vertex of the last released path for each robot. This strategy is the basis for efficient and controllable scheduling of large-scale industrial robot fleets.

A. Online Reactive Path Conflict Resolution

1) *Potential Conflict Robot Group Identification:* The online planner does not replan all robots jointly in every round. It first extracts robot groups whose unreleased paths may require

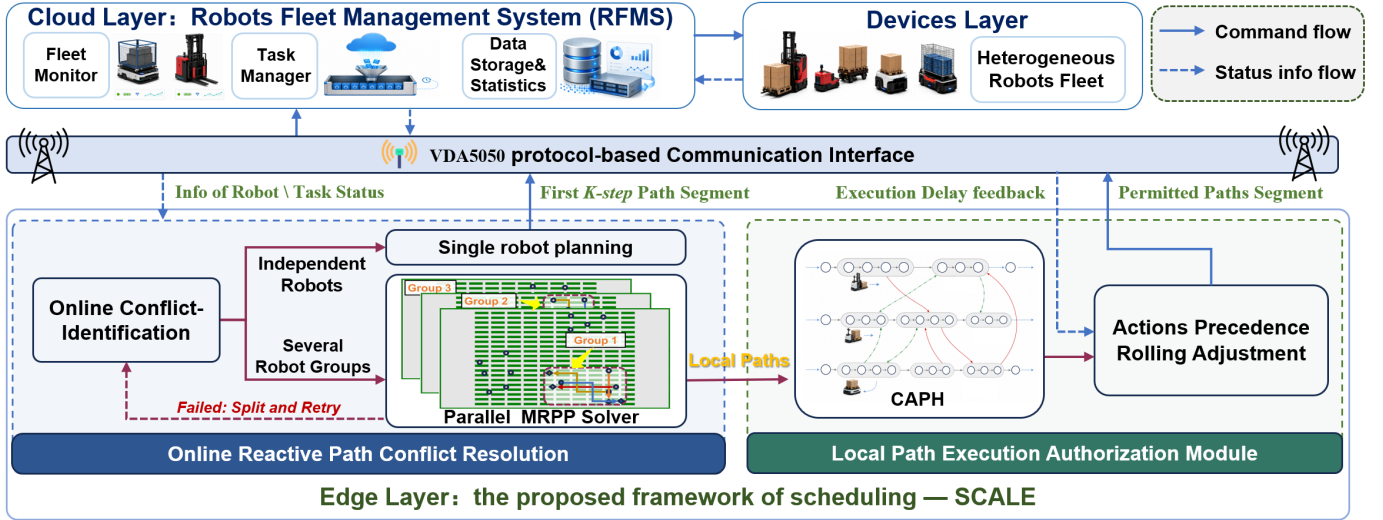


Fig. 2: Overall coordination architecture of the proposed method.

repair to handle local path conflicts, while the remaining independent robots follow single-robot planning. Let $\mathcal{R}_{\text{idle}}$ denote idle robots and $\mathcal{R}_{\text{act}} = \mathcal{R} \setminus \mathcal{R}_{\text{idle}}$ denote robots with active tasks. For each $i \in \mathcal{R}_{\text{act}}$, let $\bar{p}_i$ be the unreleased segment of the path planned in the previous scheduling round. Only the first K actions induced by $\bar{p}_i$ are considered, forming the horizon action set $\mathcal{A}_{i,K}$; if $\bar{p}_i$ is shorter than K , its terminal vertex is treated as persistent occupation. Thus, the conflict source is the path information that has already been planned but has not yet been released for execution. The horizon action sets inherit partial execution dependencies from the CAPH constructed in the previous scheduling process, too. To determine whether all the unreleased actions may block each other, we incrementally augment the dependency structure using the currently planned paths of all robots, rather than considering each tentative group in isolation, and then check whether a directed deadlock cycle is formed. Let $\mathcal{C}_K$ denote the set of detected cycles, as defined in Sec. IV-B. For a cycle $Z \in \mathcal{C}_K$, define the involved robot set as $\mathcal{R}(Z) := \{i \in \mathcal{R} \mid \exists a_i^k \in Z\}$. The initial groups are obtained by merging all overlapping cycle-induced sets: $\mathbb{G} = \text{Merge}(\{\mathcal{R}(Z) \mid Z \in \mathcal{C}_K\})$, where $\text{Merge}(\cdot)$ operates on robot sets: two cycle-induced sets are unioned whenever they share at least one robot, i.e., $\mathcal{R}(Z_1) \cap \mathcal{R}(Z_2) \neq \emptyset$. Hence, multiple deadlock cycles coupled through the same robot are assigned to one local planning group. Let v_i^{cur} be the current occupied roadmap vertex of robot i , and let $\bar{a}_i(v)$ denote its zero-motion occupation at vertex v . The same horizon is then checked against idle robots. If an idle robot j satisfies $C(a_i^k, \bar{a}_j(v_j^{\text{cur}})) = 1$ for any $a_i^k \in \mathcal{A}_{i,K}$, it is added to the group containing i .

Once a group $\mathcal{S} \in \mathbb{G}$ has been identified, local goals are assigned to guide the robots. Let $\mathcal{U}_{\mathcal{S}}$ denote the vertices set containing only the path vertices of the cyclic actions in $\mathcal{S}$:

$$\mathcal{U}_{\mathcal{S}} := \bigcup_{a_i^k(v_i^k, v_i^{k+1}) \in \mathcal{S}, \quad Z \in \mathcal{C}_K} \{v_i^k, v_i^{k+1}\}.$$

The local coordination region is obtained by breadth-first search on the roadmap graph: $\mathcal{V}_{\mathcal{S}} = \text{BFS}_{K'}(\mathcal{U}_{\mathcal{S}})$, where K' is the depth of expansion. For an active robot, the task goal

is directly used when

$\Phi_i := [v_i^g \in \mathcal{V}_{\mathcal{S}} \wedge C(\bar{a}_i(v_i^g), \bar{a}_j(v_j^{\text{cur}})) = 0, \forall j \in \mathcal{S} \setminus \{i\}]$, otherwise the local goal is selected along its current single-robot path. Define two selection predicates:

$$\begin{aligned} \text{Goal}_i(v) &:= [v \notin \mathcal{V}_{\mathcal{S}} \wedge C(\bar{a}_i(v), \bar{a}_j(v_j^{\text{cur}})) = 0, \\ &\quad \forall j \in \mathcal{S} \setminus \{i\}], \\ \text{Free}_i(v) &:= [v \notin \mathcal{V}_{\mathcal{S}} \wedge C(\bar{a}_i(v), a_m^q) = 0, \\ &\quad \forall m \in \mathcal{S} \cap \mathcal{R}_{\text{act}}, \forall a_m^q \in \mathcal{A}_{m,K}]. \end{aligned}$$

Then the strategy for local goals assignment is

$$g_i^{\text{loc}} = \begin{cases} v_i^g, & i \in \mathcal{R}_{\text{act}} \text{ and } \Phi_i, \\ \text{First}_{p_i^{\text{sr}}} \{v \in V \mid \text{Goal}_i(v)\}, & i \in \mathcal{R}_{\text{act}} \text{ and } \neg \Phi_i, \\ \arg \min_{\substack{v \in V \\ \text{Free}_i(v)}} l(v_i^{\text{cur}}, v), & i \in \mathcal{R}_{\text{idle}}. \end{cases}$$

Here $\text{Goal}_i(v)$ indicates that vertex v is outside the cycle-expanded region and can be statically occupied by robot i without conflicting with the current positions of all other robots in the group, whereas $\text{Free}_i(v)$ indicates that an idle robot can occupy v without conflicting with the active horizon actions. In addition, p_i^{sr} is the current single-robot path toward the task goal, $\text{First}_{p_i^{\text{sr}}}(\cdot)$ returns the first vertex along this path satisfying the specified condition, and $l(\cdot, \cdot)$ is the Manhattan distance on the roadmap. This assignment keeps active robots moving toward their task destinations while providing the local MRPP solver with local goals for conflict resolution; idle robots are assigned nearby avoidance vertices to avoid unnecessary blocking.

2) Independent Robot Path Planning: Robots that are not assigned to any group are planned independently. This single-robot planning step serves as a lightweight path planning and congestion-avoidance procedure for robots whose current decisions have no immediate influence on others. Inspired by traffic-flow-based optimization [27], we construct a guidance graph from the current planned paths, so that independent planning can account for the near-future congestion. Since paths are released under action-precedence constraints, only the path segments corresponding to actions that are not yet

completed should contribute to the guidance cost. Accordingly, for each robot path $p_i \in \mathcal{P}_t$, let $\hat{p}_i^t \subseteq E$ denote the relevant set of edges. The flow count of $(\mu, \nu) \in E$ is:

$$\varrho(\mu, \nu) = \sum_{p_i \in \mathcal{P}} \mathbf{1}\{(\mu, \nu) \in \hat{p}_i\}, \quad (2)$$

which represents the number of paths that pass through (μ, ν) . The vertex-level congestion and contraflow weight are estimated as $\kappa(\nu) = \sum_{\mu \in \mathcal{N}(\nu)} \varrho(\mu, \nu)$, $\chi(\mu, \nu) = \varrho(\mu, \nu)\varrho(\nu, \mu)$. These quantities are incorporated into the edge cost: $c(\mu, \nu) = (\chi(\mu, \nu), d(\mu, \nu) + \beta\kappa(\nu))$, where $d(\mu, \nu)$ denotes the edge length and $\beta > 0$ is a scaling coefficient. We thus construct the guided graph $G^g = (V, E, c)$, on which A* [28] is applied. In this way, severe opposite-direction traffic is avoided with highest priority, while usual congestion is penalized as a secondary criterion. The path planned for an independent robot is not necessarily the shortest in distance, but rather a low-congestion path.

3) Search-Integrated Conflict Discretization for Multi-Robot Path Planning: The motion-induced conflict model in Sec. III captures the physical interaction between heterogeneous robots through their swept footprints. For an MRPP algorithm, directly evaluating this continuous model during searching would require repeated pairwise polygon-overlap evaluations over candidate actions, which is inefficient for real-time planning. We introduce Search-Integrated Conflict Discretization (SICD), a reduction mechanism that embeds continuous-motion conflict detection into the discrete search process. Instead of invoking a geometric collision detector for every robot pair, it associates each candidate action with a graph-level occupancy set and uses the incremental nature of search to discard infeasible joint decisions.

Consider a single searching round in which the planner evaluates a transition from the current fleet state D^t to a tentative successor state D^{t+1} . For robot i , the corresponding continuous action is $a_i^t = (i, d_i^t, d_i^{t+1}, \gamma_i^t)$, where $d_i^t, d_i^{t+1} \in V$. Let $X(v) \in \mathbb{R}^2$ denote the planar coordinate of vertex v . SICD maps the swept footprint of a_i^t to an occupancy set:

$$\mathcal{O}(a_i^t) = \left\{ v \in \mathcal{N}_G(\gamma_i^t) \mid X(v) \in \widehat{W}_\epsilon(a_i^t) \right\}, \quad (3)$$

where $\mathcal{N}_G(\gamma_i^t)$ denotes the trajectory-wise roadmap neighborhood pre-generated along γ_i^t , and $\widehat{W}_\epsilon(a_i^t)$ is the sampled occupied region of the action. In each search round, the planner incrementally evaluates all candidate one-step movements and determines whether they can be incorporated into the same state transition. Then we embed the occupancy set $\mathcal{O}(a_i^t)$ of each action into a bitmap $\mathbf{b}(a_i^t) \in \{0, 1\}^{|V|}$, where each entry is a binary occupancy variable and the component associated with vertex v is one if and only if $v \in \mathcal{O}(a_i^t)$. Let $\mathbf{B}$ denote the accumulated bitmap obtained from all actions that have already been accepted in the current round. Then the newly considered action a_i^t is compatible if and only if $(\mathbf{B} \& \mathbf{b}(a_i^t)) = \mathbf{0}$. Here $\&$ denotes bitwise AND. If the condition holds, the accumulated occupancy is updated by the bitwise OR operation $\mathbf{B} \leftarrow \mathbf{B} \vee \mathbf{b}(a_i^t)$; otherwise the current partial joint decision is identified as conflicting. Since the bitmap entries are binary variables, both the intersection and the update can be implemented by efficient word-level bit oper-

ations. Therefore, SICD converts repeated continuous sweeping collision detection into incremental bitmap set-intersection while preserving the footprint-awareness of the underlying kinematic model. It should be noted that the mechanism is designed for collision detection on the roadmap graph during planning, while complementary geometric collision avoidance during continuous execution is enforced by the execution layer, as discussed in Sec. IV-B. Moreover, SICD is not tied to a particular solver and can be integrated into any discrete-time search-based MRPP method. The specific details of the mechanism are shown in Algorithm 2.

Algorithm 2 SICD collision detection for a state transition

Input: Current fleet state D^t , tentative successor state D^{t+1}

```

1: function SICD( $D^t, D^{t+1}$ )
2:    $\mathbf{B} \leftarrow \mathbf{0}$  ▷ occupied discretized footprint bitmap
3:   for all  $i \in \mathcal{R}$  do
4:     if  $D_i^{t+1}$  is undefined then
5:       continue
6:     end if
7:      $a_i^t \leftarrow (i, D_i^t, D_i^{t+1}, \gamma_i^t)$ 
8:      $\mathbf{b}_i \leftarrow \mathbf{b}(a_i^t)$ 
9:     if  $(\mathbf{B} \& \mathbf{b}_i) \neq \mathbf{0}$  then
10:      return true ▷ collision detected
11:    end if
12:     $\mathbf{B} \leftarrow \mathbf{B} \vee \mathbf{b}_i$ 
13:  end for
14:  return false ▷ collision free
15: end function
```

Complexity Analysis. Let w_m be the machine word size. A single SICD query performs one bitmap intersection over $\lceil |V|/w_m \rceil$ words; if the action is accepted, one bitmap union of the same size is applied. Therefore, one transition detection costs $O(\lceil |V|/w_m \rceil) \cdot O(1)$, where the $O(1)$ is the complexity of bitmap-level operation. For a collective transition with N robots, the worst-case round complexity is $O(N\lceil |V|/w_m \rceil)$. By comparison, a direct polygon-overlap (PO) scheme repeatedly evaluates continuous geometry for robot pairs and has $O(N^2 L c_{\text{poly}})$ worst-case complexity per search round, where N is the number of robots, L is the number of footprint samples along an action, and c_{poly} denotes the cost of one polygon-overlap predicate. Since c_{poly} involves floating-point geometric predicates on polygonal footprints, frequent PO calls introduce substantial and redundant computation during incremental searching. SICD reduces this burden by replacing pairwise continuous overlap detections with one bitmap query against the accumulated occupancy for each candidate action.

Remark IV.1. When local MRPP fails to find a feasible solution for a group within the time limit, the group is split. The planner then reduces the commitment horizon K and relaxes the timeout threshold before retrying the resolution process. This fallback strategy allows the online conflict-resolution process to iteratively pursue a feasible resolution.

B. Conjugate Action-Precedence Hypergraph

To achieve latency-resilient path execution among heterogeneous robots, we introduce the Conjugate Action-Precedence

Hypergraph (CAPH), which accommodates delays caused by external uncertainty. It preserves the original action-level dependencies of HAPG while lifting selected inter-robot dependencies to a capsule-level representation, termed the Conjugate Precedence Capsule (CPC). This representation reduces unnecessary waiting through dependency switching.

1) **CPC Construction:** We first introduce the action capsule, the basic component of CAPH:

Definition IV.2 (Action Capsule). For robot i , let $\sigma_i = (a_i^1, \dots, a_i^{L_i-1})$ be its action sequence. An action capsule is a contiguous action segment $Q_i^{p:q} := \langle a_i^p, \dots, a_i^q \rangle$, $1 \leq p \leq q \leq L_i - 1$. For each action a_i^k , let Δ_i^k denote its nominal traversal duration, estimated from the length of the corresponding trajectory and the speed of robot type τ_i . The entry and exit times of an action capsule are denoted by $T_s(\cdot)$ and $T_g(\cdot)$, respectively. Then a capsule $Q_i^{k:k+\ell}$ is associated with an entry time $\hat{T}_s(Q_i^{k:k}) \geq \hat{T}_g(Q_i^{k:k-1})$ and an exit time $\hat{T}_g(Q_i^{k:k+\ell}) \geq T_s(Q_i^{k:k}) + \sum_{n=k}^{k+\ell} \Delta_i^n$. An action capsule has the same states as an action: {staged, in-progress, completed}.

The CPC construction process is based on the initial dependencies in HAPG. Each action is first wrapped as a singleton action capsule: $\mathcal{Q}_0 = \{Q_i^{k:k} \mid i \in \mathcal{R}, k = 1, \dots, L_i - 1\}$. Let $\psi(a_i^k) = Q_i^{k:k}$. This mapping preserves the original HAPG dependencies: $(\psi(a), \psi(b)) \iff (a, b) \in \mathcal{E}$. For any pair of robot paths (p_i, p_j) , we define the conflict signature of action a_j^ℓ with respect to p_i as $\Gamma_{i \leftarrow j}(\ell) = \{k \in \{1, \dots, L_i - 1\} \mid C(a_i^k, a_j^\ell) = 1\}$, $\ell = 1, \dots, L_j - 1$. This signature characterizes the local conflict context of a_j^ℓ with respect to the reference path p_i . A non-empty interval $[s, t]$ on p_j is called a maximal constant-signature interval if $\Gamma_{i \leftarrow j}(s) = \Gamma_{i \leftarrow j}(s+1) = \dots = \Gamma_{i \leftarrow j}(t) = \mathcal{I} \neq \emptyset$. This interval induces the capsule $Q_j^{s:t} = \langle a_j^s, \dots, a_j^t \rangle$. Its conjugate capsule on p_i is defined as the minimum contiguous span covering the signature $\mathcal{I}$, namely $Q_i^{p:q}$ with $p = \min \mathcal{I}$ and $q = \max \mathcal{I}$. If the conflict signatures inherited from HAPG between the two capsules do not form bidirectional dependencies, a candidate capsule pair is formed, denoted by $\omega_{ij}^\zeta := (Q_i^{p:q}, Q_j^{s:t})$, where ζ indexes the capsule pair.

Lemma IV.3. For a candidate capsule pair $\omega_{ij}^\zeta = (Q_i^{p:q}, Q_j^{s:t})$ constructed from the mutual conflict signature, all conflicts between $Q_j^{s:t}$ and path p_i are contained in $Q_i^{p:q}$, i.e.,

$$C(a_i^k, a_j^\ell) = 0, \quad \forall \ell \in [s, t], k \notin [p, q].$$

Therefore, after robot i exits $Q_i^{p:q}$, its subsequent actions do not conflict with $Q_j^{s:t}$. The same property holds symmetrically for robot j .

Proof. For any $\ell \in [s, t]$, the maximal constant conflict signature gives $\Gamma_{i \leftarrow j}(\ell) = \mathcal{I}$. By definition, $k \in \mathcal{I}$ if and only if $C(a_i^k, a_j^\ell) = 1$. Since $Q_i^{p:q}$ is the minimum contiguous span covering $\mathcal{I}$, any $k \notin [p, q]$ is not in $\mathcal{I}$, and thus cannot conflict with a_j^ℓ . Applying the same argument to the opposite conflict signature proves the symmetric case. $\square$

Lemma IV.4. Given an action capsule pair $\omega_{ij}^\zeta = (Q_i^{p:q}, Q_j^{s:t})$, it is a CPC if at least one capsule is non-singleton, i.e., $q - p + 1 \geq 2$ or $t - s + 1 \geq 2$, and the binary orientation

$z_\omega \in \{0, 1\}$ satisfies the temporal constraints in Definition IV.2 whenever switching. Specifically, if the two capsules satisfy $\hat{T}_g(Q_i^{p:q}) \leq \hat{T}_s(Q_j^{s:t})$, then the relation is $Q_i^{p:q} \prec Q_j^{s:t}$, and vice versa. The initial CPC orientation is consistent with the physical actions but is constructed at the capsule level.

Proof. Without loss of generality, assume $z_\omega = 0$. Let $n_i = q - p + 1$ and $n_j = t - s + 1$. If $n_i = n_j = 1$, then both capsules are singletons. Since each capsule corresponds to one action, $Q_i^{p:q} = \{a_i^p\}$ and $Q_j^{s:t} = \{a_j^s\}$, switching to $z_\omega = 1$ is equivalent to reversing the two adjacent actions, which is an invalid switch. This corresponds to the following relation between two robots. Now suppose that $\max\{n_i, n_j\} \geq 2$. Initially, when $z_\omega = 0$, the relation is given by $T_g(Q_i^{p:q}) \leq T_s(Q_j^{s:t})$. After switching to $z_\omega = 1$, Lemma IV.3 ensures that no subsequent action outside the paired capsules conflicts with the opposite capsule, and the absence of bidirectional dependencies within the CPC guarantees acyclicity between them. That is, before any action in $Q_i^{p:q}$ starts, all actions in $Q_j^{s:t}$ that appear in $\Gamma_{i \leftarrow j}$ have already been completed, satisfying the constraint $T_g(Q_j^{s:t}) \leq T_s(Q_i^{p:q})$. Hence, ω is a switchable CPC. $\square$

As shown in Fig. 3a, the paths of robots A and B generate the following sequences of action capsule: $\langle Q_A^{1:1}, Q_A^{2:2}, Q_A^{3:3} \rangle, \langle Q_B^{1:1}, \dots, Q_B^{5:5} \rangle$. The conflict signature of robot B with respect to robot A over actions 1 to 4 is given by $\Gamma_{A \leftarrow B}(1, \dots, 4) = \{1, 2\}$, which induces the CPC: $(Q_A^{1:2}, Q_B^{1:4})$. Since $\hat{T}_g(Q_A^{1:2}) \leq \hat{T}_s(Q_B^{1:4})$, its initial orientation is $Q_A^{1:2} \prec Q_B^{1:4}$; hence, actions in $Q_B^{1:4}$ are admitted only after $Q_A^{1:2}$ is completed. In this example, the Kiva-like robot can enter the intersection only after the forklift has left it completely, while this capsule order can still be switched online when both capsules remain staged.

2) **CAPH:** We continue to present the formal definition of CAPH and its online optimization process in this part.

Definition IV.5 (Conjugate Action-Precedence Hypergraph). The CAPH is defined as $\mathcal{H} = (\mathcal{Q}, \Pi_f, \Omega_z)$, where $\mathcal{Q}$ is the set of action capsules, $\Pi_f \subseteq \mathcal{Q} \times \mathcal{Q}$ is the set of fixed capsule precedences, Ω is the set of CPCs, and $\mathbf{z} = \{z_\omega \mid \omega \in \Omega\}$ is the set of binary variables to represent the orientations.

For a given configuration $\mathbf{z}$, CAPH induces an active capsule precedence set $\Pi(\mathbf{z}) = \Pi_f \cup \Omega(\mathbf{z})$, where $\Omega(\mathbf{z})$ contains the dependencies selected by each CPC. During online process, the orientation of a CPC can only be switched when both capsules are in the state of staged.

Definition IV.6 (Inter-robot deadlock cycle in CAPH). Given the active precedence set $\Pi(\mathbf{z})$, a CAPH deadlock cycle exists if there is a sequence of capsules $Q_{\alpha_1}^{b_1:e_1}, Q_{\alpha_2}^{b_2:e_2}, \dots, Q_{\alpha_M}^{b_M:e_M}$ with $M \geq 2$, $\alpha_m \in \mathcal{R}$, and at least two robots such that

$$(Q_{\alpha_m}^{b_m:e_m}, Q_{\alpha_{m+1}}^{b_{m+1}:e_{m+1}}) \in \Pi(\mathbf{z}), \quad m = 1, \dots, M,$$

where $\alpha_{M+1} = \alpha_1$, $b_{M+1} = b_1$, and $e_{M+1} = e_1$. The action-level cycle is

$$Z = \bigcup_{m=1}^M \{a_{\alpha_m}^k \mid k = b_m, \dots, e_m\}.$$

The set of all such action-level cycles within a horizon K is denoted by $\mathcal{C}_K$. The CAPH is deadlock-free if $\mathcal{C}_K = \emptyset$.

3) Rolling Adjustment of CAPH: Inspired by the receding-horizon schedule re-ordering in SADG [11], we formulate the rolling adjustment as a MILP to optimize CAPH. Since paths are released at most K steps in each scheduling cycle, the optimization only involves the unreleased capsules within this bounded horizon and is therefore lightweight. Let $\mathcal{Q}_K \subseteq \mathcal{Q}$ be the set of staged capsules, $\Omega_K \subseteq \Omega_{\mathbf{z}}$ be the CPCs, and $\Pi_{f,K}$ be the fixed precedences. Let z_{ω}^0 denote the current orientation of CPC ω , and let $\hat{\Delta}(Q)$ be the remaining execution duration of capsule Q .

The MILP of rolling adjustment is formulated as:

$$\begin{aligned} \min_{\mathbf{z}_K, \mathbf{T}_s, \mathbf{T}_g} \quad & \sum_{Q \in \mathcal{Q}_K^{\text{end}}} T_g(Q) \\ \text{s.t.} \quad & T_g(Q) \geq T_s(Q) + \hat{\Delta}(Q), \\ & T_s(Q') \geq T_g(Q), \\ & T_s(Q') \geq T_g(Q) - M_B z_{\omega}, \\ & T_s(Q) \geq T_g(Q') - M_B(1 - z_{\omega}). \end{aligned} \quad (4)$$

Here $\mathcal{Q}_K^{\text{end}}$ contains the last horizon capsule of each robot considered in the current release, and M_B is a sufficiently large constant. The objective minimizes the cumulative predicted horizon-exit time. The binary switch variables satisfy $\mathbf{z}_K = \{z_{\omega} \mid \omega \in \Omega_K\}$. $\hat{\Delta}(Q)$ is the calculated remaining duration of capsule Q . The duration constraint is imposed for all $Q \in \mathcal{Q}_K$, the fixed precedence constraint for all $(Q, Q') \in \Pi_{f,K}$, and the two big- M_B constraints for every CPC $\omega = (Q, Q') \in \Omega_K$. A switch with $z_{\omega} \neq z_{\omega}^0$ is admissible only when $\xi(Q) = \xi(Q') = \text{staged}$, preventing dependencies from being reversed after either capsule has already been active. If a switched active precedence set contained a directed capsule cycle, the corresponding start and completion times would require at least one capsule to start after its own completion, contradicting the feasible temporal ordering. Hence the optimized hypergraph satisfies $\mathcal{C}_K(\mathbf{z}_K) = \emptyset$.

Theorem IV.7. *The CAPH-based persistent path execution is collision-free and deadlock-free with paths planned by the MRPP solver proposed in IV-A.*

Proof. For a solvable instance, the MRPP solver returns a feasible ranked paths set for robots. During planning, the SICD mechanism in Sec. IV-A.3 enables MRPP to exclude all roadmap-node occupations induced by swept footprints. The remaining execution safety is then enforced by CAPH. Consider two inter-robot actions a_{α}^k and a_{β}^m . If $C(a_{\alpha}^k, a_{\beta}^m) = 0$, they are non-conflicting under the model in Sec. III. If $C(a_{\alpha}^k, a_{\beta}^m) = 1$, Def. III.1 introduces an action-level precedence between them. CAPH lifts such conflict to fixed capsule precedences or CPC, ensuring non-collision among robots under the precedence constraints. By Lemma IV.4, a valid CPC provides a capsule-level temporal order for the conflicting capsules. Since a capsule is active only after all its active predecessors have been completed, conflicting actions are temporally separated during asynchronous execution. Therefore the CAPH-based execution is collision-free. For deadlock freedom, by Lemma III.2, the HAPG induced by the feasible

MRPP paths is acyclic and admits deadlock-free execution. As initial CAPH is obtained from this HAPG by lifting selected action-level precedences to fixed capsule precedences or initial CPCs, it admits the same feasible temporal ordering among capsules. During rolling adjustment in Sec. IV-B.3, completed or in-progress capsule precedences cannot be reversed, and a CPC can be switched only when both capsules are still staged. Each accepted switch also satisfies the temporal constraints in (4); otherwise a directed capsule cycle would require some capsule to start after its own completion. Thus $\Pi(\mathbf{z})$ remains acyclic after every accepted adjustment. Hence the persistent path execution is deadlock-free. $\square$

As illustrated in Fig. 3b, the example demonstrates a CAPH and its rolling adjustment process. During execution, r_1 and r_2 are delayed due to human-induced dynamic blockages or robot failures. After optimization, r_3 and r_4 are allowed to execute their paths earlier while collision-free operation is preserved.

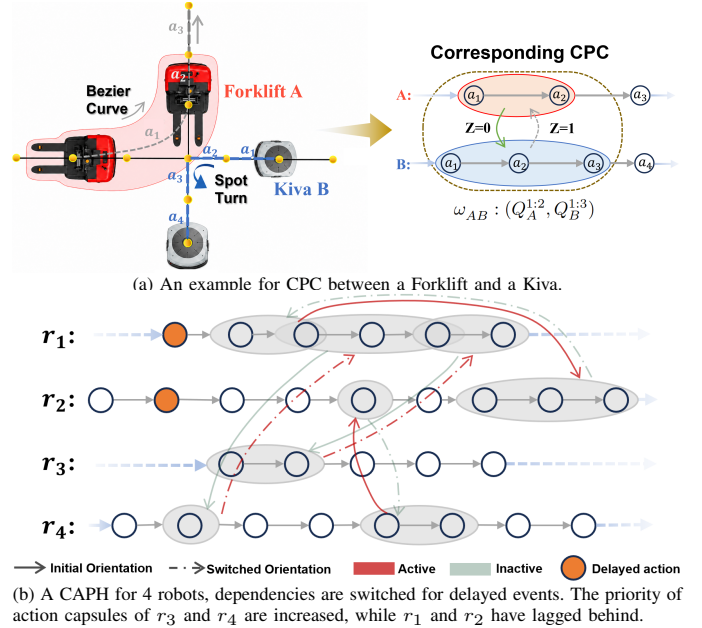


Fig. 3: Illustration of examples of CAPH and its components.

V. EVALUATION

We evaluate the proposed method along three dimensions: real-time MRPP performance, robust path execution under external disturbances, and system-level lifelong coordination. For MRPP, the proposed SICD mechanism is integrated into representative solvers, including LaCAM* [6], PIBT [19], PBS [17], ECBS [16], and CBS [18].¹ These solvers are extended from grids to graph version. For each solver, we compare the original graph-agent implementation without continuous motion-collision checking, direct polygon-overlap (PO) detection, and the proposed SICD-based method. For path execution, except for non-switchable HAPG, CAPH is compared with SG [29], an advanced segment-graph method

¹We build the solvers based on the public implementations of LaCAM* <https://github.com/Keil8/lacam0>, PIBT <https://github.com/Keil8/piibt2>, and PBS/ECBS/CBS <https://github.com/whoenig/libMultiRobotPlanning>.

that models robot motion at the segment level and greedily switches passage priorities. For system-level validation, SCALE is compared with two industrial baselines, CAEH [13] and GNGW [3]. CAEH alleviates congestion through a hierarchical topological architecture and local CBS-based conflict resolution, whereas GNGW uses glued nodes for safe operation and local give-way behaviors for collision avoidance.

To evaluate latency resilience under industrial disturbances, we introduce the uncertainty sources in Table I into the experiments in Secs. V-B and V-C. Packet loss is sampled once per scheduling round; if triggered, the system loses the corresponding status update. Controller delay occurs independently before the start of each robot task, postponing execution by 5 s. Human-factor disturbances are sampled for each robot-task pair with a probability of 2%; once triggered, the affected robot remains stationary for 120 s before resuming operation.

All experiments involve three robot types: forklifts, mobile manipulators, and Kiva robots. Table II summarizes the robot parameters, with the three robot types maintained at a fixed number ratio of 4 : 1 : 5.

All programs are written in C++ and run on a workstation with an Intel Core i9-14900K CPU and 32 GB RAM.

TABLE I: Settings for External Disturbances in Experiments

Item	Probability	Pattern
Packet loss	1% / planning cycle	State update loss
Controller delay	0.5% / robot start	5 s start delay
Human factor	2% / robot-task pair	120 s pause

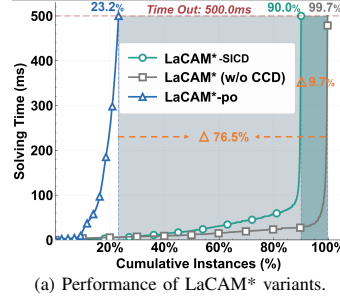
TABLE II: Parameters of Heterogeneous Robots

Parameter	Forklift (FL.)	Mobile Manipulator (M.M.)	Kiva
Unloaded robot size ($m \times m$)	(2.10×0.96)	(0.85×0.85)	(0.76×0.76)
Loaded robot size ($m \times m$)	(2.10×1.20)	(0.85×0.90)	(0.76×0.76)
Max velocity (m/s)	1.6	1.0	1.3
Max acceleration (m/s^2)	0.6	0.5	0.7
Max braking acceleration (m/s^2)	0.8	0.7	0.9
Min turning radius (m)	1.6	0.8	0
Safety radius ϵ (m)	0.3	0.2	0.15
Battery consumption rate (%/h)	15%/h	17%/h	8%/h
Charging rate (%/h)	50%/h	55%/h	80%/h

A. Performance of One-shot MRPP

We first evaluate conflict detection on MRPP instances sampled from the long-term operation in Sec. V-C. Each triggered local MRPP group is recorded, yielding 10,000 instances with 2–20 robots. Fig. 4a reports the cumulative solving-time distribution under a 500 ms timeout. LaCAM* with SICD solves 90.0% of the instances while explicitly considering continuous motion-conflict detection (CCD), only 9.7% lower than the original graph-agent solver without CCD. In contrast, the PO-based variant solves only 23.2%. Fig. 4 further reports the average total detections (TD) and true-positive rate (TPR). Introducing CCD increases the number of checks because swept-volume interactions reveal physically relevant conflicts beyond the assumption of agents. Overall, SICD provides the required physical safety screening while preserving much of the original MRPP search efficiency.

To further verify the applicability of SICD, we evaluate it on three maps from the public *MovingAI MAPF* benchmark². In these maps, each edge is assigned a length of 1.5 m, and



(a) Performance of LaCAM* variants.

N_r	w/o CCD		SICD	
	TD	TPR(%)	TD	TPR(%)
2	26	1.3	344	8.7
4	45	3.2	798	5.8
6	73	3.3	853	15.8
8	95	3.7	1913	11.5
10	122	5.7	1813	9.9
12	144	4.9	1600	7.5
14	205	7.6	2109	8.0
16	297	9.2	2661	13.5
18	274	8.2	3054	11.3
20	277	7.5	5031	12.1

(b) Number of conflicts during search.

Fig. 4: Planning performance on an industrial roadmap graph.

edge curvature is adjusted according to robot turning-radius constraints. For each map, we randomly generate 50 instances under 11 robot-density settings from 0.02 to 0.30, with a 30 s solving timeout. Fig. 5 shows that SICD-based solvers maintain high success rates across different structures of maps. Compared with the original solvers without CCD, the success-rate reduction is generally below 8% when the robot density is below 0.20 and remains within 22% in denser settings. In particular, it largely preserves the original solver efficiency when the number of robots is below 100. By contrast, PO substantially reduces scalability: even advanced solvers such as LaCAM* and PIBT lose the ability to solve in time at low densities. For example, on den312d, CBS with PO fails to solve any instance even at density 0.02. Fig. 6 reports the median solving time over solvable cases, with shaded bands indicating the minimum–maximum range. PO-based solvers spend substantial effort on repeated geometric overlap tests, whereas SICD keeps the overhead close to the original search and remains practical for MRPP.

B. Performance of Path Execution under Uncertainty

We evaluate path execution policies under external disturbances. Statistics from the system-level simulation in Sec. V-C show that local robot groups in SCALE rarely contain more than 20 robots. To further test scalability, we randomly generate solvable cases with group sizes $N_r \in \{10, 20, 30, 40, 50\}$ in the same scenario and sample 50 cases for each size. The disturbance settings follow Table I. Table III shows that CAPH achieves the shortest execution time for all group sizes. Compared with the non-switchable HAPG, CAPH reduces execution time by up to 18.4% when the scale is 50. Since HAPG strictly follows the precedence relations generated by MRPP, a local delay can propagate through fixed dependencies and increase the completion time of the entire group. SG also allows dependency switching, but its execution-time reduction over HAPG remains below 8.0% in all tested cases, indicating that greedy switching over coarse action segments provides limited resilience to external disturbances.

The advantage of CAPH over SG becomes more pronounced as the group size increases. In the 50-robot cases, it reduces the average execution time by about 110s compared with SG. It also achieves this improvement with far fewer dependency switches: its switch count is about one third of SG across all cases. The capsule-level dependency model therefore avoids many ineffective or negative switches globally, reduces redundant switching computation, and helps prevent livelock caused by dependency oscillations.

²<https://movingai.com/benchmarks/mapf>

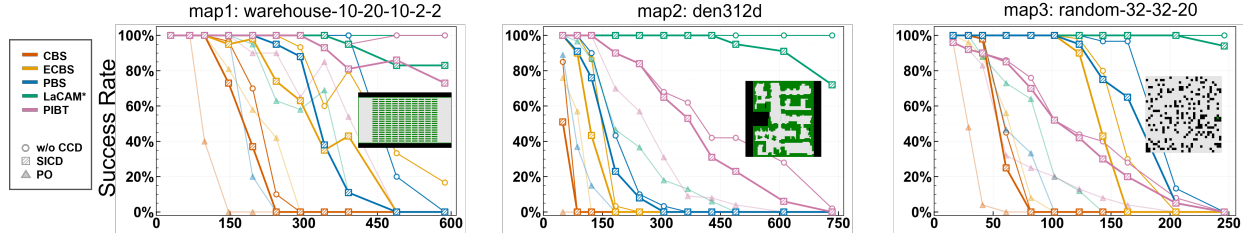


Fig. 5: Success rate versus the number of robots for representative MRPP solvers with and without CCD.

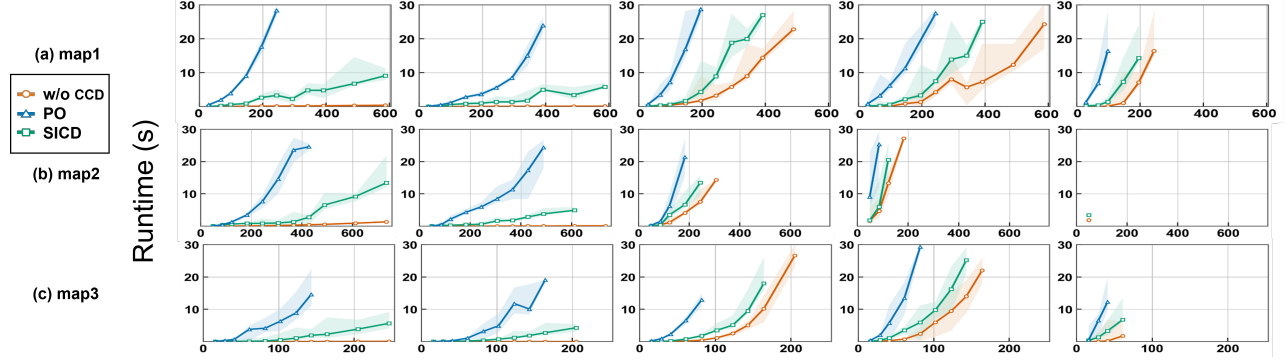


Fig. 6: Computation time of successful MRPP instances in Fig. 5.

TABLE III: Comparison of different execution strategies

N_r	Total Time of Execution / s ($\downarrow$)			Number of Switches	
	HAPG	SG	CAPH	SG	CAPH
10	239.6 (16.9)	237.6 (13.1)	212.3 (9.7)	33.5	10.7
20	395.9 (68.5)	387.4 (74.2)	297.1 (38.1)	116.4	33.5
30	519.0 (90.1)	502.1 (92.2)	427.5 (76.7)	181.3	57.1
40	621.4 (130.5)	581.3 (105.3)	533.8 (91.0)	240.3	106.3
50	793.3 (199.9)	756.1 (168.5)	647.8 (112.5)	292.4	126.8

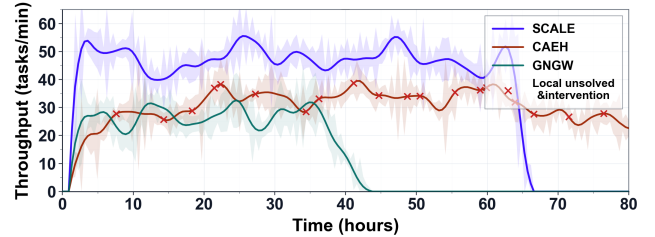
Values in parentheses denote standard deviations.

C. System-level Simulation

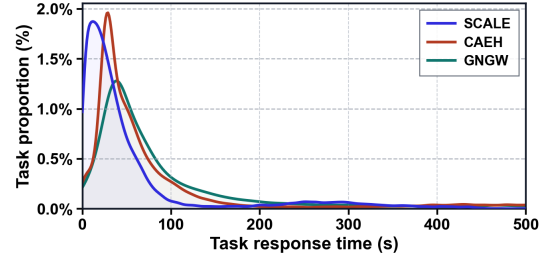
We conduct a lifelong system-level simulation in our real-world industrial project with a fleet of 300 heterogeneous robots. The roadmap graph contains 24,356 nodes, 60109 edges, and 1500 fixed storage locations. Under the external-disturbance settings in Table I and the robot parameters in Table II, the system randomly releases 80 tasks every 60s. Each task includes pickup and drop-off operations, with a 5s operation time. The experiment runs for three days.

The comparison of throughput over time is shown in Fig. 7a. SCALE rapidly increases its throughput during the startup phase and reaches about 50 tasks/min within the first 10h, while CAEH and GNGW remain below about 25 tasks/min in the same period. During the main operating interval from 10h to 60h, SCALE maintains a high throughput level, mostly between 40 and 55 tasks/min and peaking at about 54 tasks/min. In contrast, CAEH stays around 30–35 tasks/min and requires multiple local interventions when unsolved local conflicts occur, as marked by the crosses in the figure. Its average throughput is therefore about 30% lower than that of SCALE. GNGW achieves a short-term peak of about 35 tasks/min, but its throughput starts to collapse after about 30h and becomes nearly zero around 42h because local give-way behaviors cannot recover from accumulated deadlocks. Consequently, GNGW finishes less than 40% of the total tasks, whereas SCALE completes the released tasks and returns all robots to the idle state at the end. Fig. 7b reports the distribution of task response time, measured as the waiting time between

task assignment and the moment when the assigned robot starts moving. SCALE has the most concentrated distribution, with its main peak within about 10–20s and a fast decay after 60s. CAEH and GNGW exhibit right-shifted distributions, with peaks around 30s and 40s, respectively, and visibly heavier long tails caused by postponed task admission under congestion and unresolved local conflicts. The average task response times of SCALE, CAEH, and GNGW are 28s, 51s, and 69s, respectively, meaning that SCALE reduces the average response time by about 45% compared with CAEH and about 59% compared with GNGW. Overall, the proposed method achieves higher sustained throughput and shorter task response time in the lifelong industrial simulation.



(a) Comparison of throughput per minute.



(b) Comparison of response time for tasks.

Fig. 7: Comparison of lifelong performance for running 300 robots for three days under three scheduling frameworks.

D. Industrial Deployment

The deployment validation is conducted in a real-world industrial warehousing scenario with a heterogeneous fleet

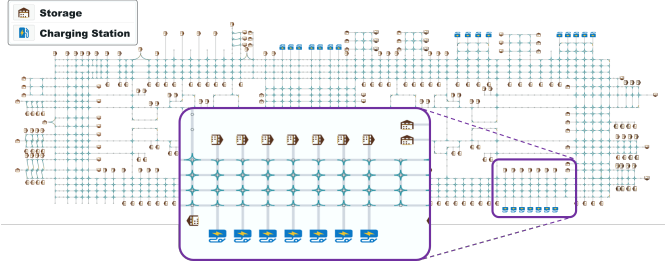


Fig. 8: The roadmap network in a warehousing factory.

of 100 robots; their physical parameters are summarized in Table II. The real-world deployment scene is shown in Fig. 1, and the corresponding roadmap graph is shown in Fig. 8. The site contains multiple storage blocks, long trunk corridors, dense aisle intersections, and several charging areas, which together create frequent merging, crossing, and temporary blocking situations during normal operation. The warehouse roadmap graph contains 5,839 nodes, 19,670 edges, and 274 fixed storage locations. During the monitored multi-day op-

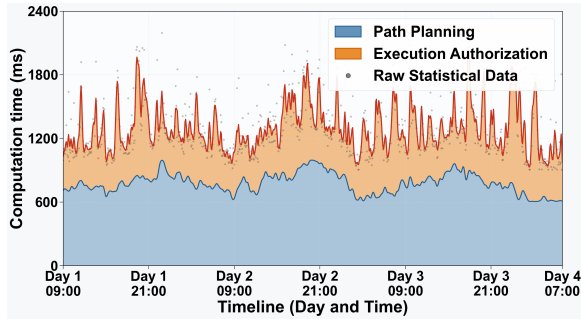


Fig. 9: Computational time distribution in deployment.

eration, the 100 robots complete approximately 30,000 tasks in total, corresponding to more than 100 tasks per robot per day on average. Scheduling is triggered every 10s, and each round processes robot states, updates unfinished path prefixes, resolves newly detected local conflicts, and authorizes executable paths. Fig. 9 shows the edge-side computation-time distribution during operation. The path planning time in each scheduling round remains stable within 700–1800 ms. Path authorization relies on rolling optimization and therefore incurs additional computational overhead, particularly when robot priorities are need to be adjusted in response to external disturbances. Even so, the total edge-side computation remains well below the period between scheduling loops, leaving sufficient time for device-side execution.

VI. CONCLUSION

This paper studies scalable coordination for heterogeneous robot fleets in industrial environments, where system efficiency depends not only on collision-free path planning under realistic constraints, but also on reliable, robust asynchronous path execution under external disturbances. The proposed framework treats planning and execution as a coupled online process. It commits bounded paths to enable controllable scheduling through online conflict resolution, and regulates execution relations via an action-precedence dependency hypergraph, on which the execution order of robots can be receding-horizon optimized in response to external disturbances. In this sense,

the framework bridges graph-based multi-robot planning and the practical requirements of scheduling dense industrial robot fleets. Extensive simulations and real-world factory deployment demonstrate the feasibility and efficiency of the proposed method for large-scale heterogeneous robot fleets. Our future work will focus on extending the pipeline from roadmap graph-based coordination to more general multi-robot trajectory planning, so that continuous-time motion planning and execution can be integrated within a more unified representation, thereby broadening the applicability of multi-robot systems.

REFERENCES

- [1] S. Reveliotis, “An mpc scheme for traffic coordination in open and irreversible, zone-controlled, guidpath-based transport systems,” *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 3, pp. 1528–1542, 2020.
- [2] J. Luo, Y. Wan, W. Wu, and Z. Li, “Optimal petri-net controller for avoiding collisions in a class of automated guided vehicle systems,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 11, pp. 4526–4537, 2019.
- [3] Z. Xing, H. Yue, T. Zhang, W. Wu, and R. Hu, “Collision avoidance and give way of heterogeneous and variable-sized multiple mobile robots based on glued nodes,” *IEEE Transactions on Automation Science and Engineering*, vol. 21, no. 3, pp. 3627–3638, 2023.
- [4] J. Li, Z. Chen, D. Harabor, P. J. Stuckey, and S. Koenig, “Mapf-Ins2: Fast repairing for multi-agent path finding via large neighborhood search,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 9, 2022, pp. 10 256–10 265.
- [5] K. Okumura, “LaCAM: Search-based algorithm for quick multi-agent pathfinding,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 10, 2023, pp. 11 655–11 662.
- [6] K. Okumura, “Improving LaCAM for scalable eventually optimal multi-agent pathfinding,” in *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*, 8 2023, pp. 243–251.
- [7] J. Li, P. Surynek, A. Felner, H. Ma, T. S. Kumar, and S. Koenig, “Multi-agent path finding for large agents,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 7627–7634.
- [8] E. Boyarski, R. Stern, K. S. Yakovlev, and A. Andreychuk, “Improving Continuous-time Conflict Based Search,” *arXiv: Artificial Intelligence*, Jan. 2021.
- [9] Z. Chen, D. D. Harabor, J. Li, and P. J. Stuckey, “Symmetry breaking for k-robust multi-agent path finding,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 14, 2021, pp. 12 267–12 274.
- [10] W. Hönig, S. Kiesel, A. Tinka, J. W. Durham, and N. Ayanian, “Persistent and robust execution of mapf schedules in warehouses,” *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1125–1131, 2019.
- [11] A. Berndt, N. Van Duijkeren, L. Palmieri, A. Kleiner, and T. Keviczky, “Receding horizon re-ordering of multi-agent execution schedules,” *IEEE Transactions on Robotics*, vol. 40, pp. 1356–1372, 2023.
- [12] Y. Sun, N. Zhao, L. Tang, and L. Luo, “Breaking the limit on the number of robots through the conflict-free scheduling in robotic mobile fulfillment systems,” *IEEE Transactions on Automation Science and Engineering*, vol. 22, pp. 7324–7334, 2024.
- [13] A. Bonetti, S. Proia, S. Guidetti, and L. Sabattini, “Agy traffic management in automated industrial plants: An enhanced lifelong multi-agent path finding approach,” in *2024 IEEE 20th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2024, pp. 626–632.
- [14] H. Ma, W. Hönig, T. S. Kumar, N. Ayanian, and S. Koenig, “Lifelong path planning with kinematic constraints for multi-agent pickup and delivery,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 7651–7658.
- [15] J. Li, A. Tinka, S. Kiesel, J. W. Durham, T. S. Kumar, and S. Koenig, “Lifelong multi-agent path finding in large-scale warehouses,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 13, 2021, pp. 11 272–11 281.
- [16] M. Barer, G. Sharon, R. Stern, and A. Felner, “Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem,” in *Proceedings of the Seventh Annual Symposium on Combinatorial Search*, 2014, pp. 19–27.

- [17] H. Ma, D. Harabor, P. J. Stuckey, J. Li, and S. Koenig, "Searching with consistent prioritization for multi-agent path finding," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 7643–7650.
- [18] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," *Artificial intelligence*, vol. 219, pp. 40–66, 2015.
- [19] K. Okumura, M. Machida, X. Défago, and Y. Tamura, "Priority inheritance with backtracking for iterative multi-agent path finding," *Artificial Intelligence*, vol. 310, p. 103752, 2022.
- [20] D. Atzmon, Y. Zax, E. Kivity, L. Avitan, J. Morag, and A. Felner, "Generalizing multi-agent path finding for heterogeneous agents," in *Proceedings of the International Symposium on Combinatorial Search*, vol. 11, no. 1, 2020, pp. 101–105.
- [21] J. Kottlinger, S. Almagor, and M. Lahijanian, "Conflict-based search for multi-robot motion planning with kinodynamic constraints," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 13 494–13 499.
- [22] A. Moldagalieva, K. Okumura, A. Prorok, and W. Hönig, "db-lacm: Fast and scalable multi-robot kinodynamic motion planning with discontinuity-bounded search and lightweight mapf," *arXiv preprint arXiv:2512.06796*, 2025.
- [23] D. Atzmon, R. Stern, A. Felner, N. R. Sturtevant, and S. Koenig, "Probabilistic robust multi-agent path finding," in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 30, 2020, pp. 29–37.
- [24] W. Hönig, T. Kumar, L. Cohen, H. Ma, H. Xu, N. Ayanian, and S. Koenig, "Multi-agent path finding with kinematic constraints," in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 26, 2016, pp. 477–485.
- [25] K. Okumura, Y. Tamura, and X. Défago, "Time-independent planning for multiple moving agents," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 13, pp. 11 299–11 307, 2021.
- [26] Y. Zhang, Z. Chen, D. Harabor, P. Le Bodic, and P. J. Stuckey, "Concurrent planning and execution in lifelong multi-agent path finding with delay probabilities," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, no. 22, 2025, pp. 23 387–23 394.
- [27] Z. Chen, D. Harabor, J. Li, and P. J. Stuckey, "Traffic flow optimisation for lifelong multi-agent path finding," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 18, 2024, pp. 20 674–20 682.
- [28] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.
- [29] A. Coskun, J. M. O’Kane, and M. Valtorta, "Deadlock-free online plan repair in multi-robot coordination with disturbances," in *Proceedings of the Thirty-Fourth International Florida Artificial Intelligence Research Society Conference*. Florida Online Journals, 2021.