HECTOR: Human-centric Hierarchical Coordination and Supervision of Robotic Fleets under Continual Temporal Tasks

Shen Wang[1], Yinhang Luo[1], Jie Li[2] and Meng Guo[1]

***Abstract*—Robotic fleets can be extremely efficient when working concurrently and collaboratively, e.g., for delivery, surveillance, search and rescue. However, it can be demanding or even impractical for an operator to directly control each robot. Thus, autonomy of the fleet and its online interaction with the operator are both essential, particularly in dynamic and partially unknown environments. The operator might need to add new tasks, cancel some tasks, change priorities and modify planning results. How to design the procedure for these interactions and efficient algorithms to fulfill these needs have been mostly neglected in the related literature. Thus, this work proposes a human-centric coordination and supervision scheme (HECTOR) for large-scale robotic fleets under continual and uncertain temporal tasks. It consists of three hierarchical layers: (I) the bidirectional and multimodal protocol of online human-fleet interaction, where the operator interacts with and supervises the whole fleet; (II) the rolling assignment of currently-known tasks to teams within a certain horizon, and (III) the dynamic coordination within a team given the detected subtasks during online execution. The overall mission can be as general as temporal logic formulas over collaborative actions. Such hierarchical structure allows human interaction and supervision at different granularities and triggering conditions, to both improve computational efficiency and reduce human effort. Extensive human-in-the-loop simulations are performed over heterogeneous fleets under various temporal tasks and environmental uncertainties.**

***Note to Practitioners*—This work is motivated by the practical need for an operator to coordinate a large number of heterogeneous and autonomous robots such as unmanned aerial and ground vehicles, to accomplish complex collaborative tasks such as patrol, delivery, search and rescue. Existing approaches often assume that the fleet size is relatively small, and the tasks are known beforehand, while neglecting the human interaction during online execution. This paper proposes a human-centric and hierarchical framework for online coordination and supervision of potentially dozens of robots. It allows the operator to interact online with the fleet via lean communication, e.g., to specify complex temporal missions, approve the formation of teams with assigned tasks, monitor the progress of task execution, potentially add new tasks and remove old tasks, and adjust the priority of some tasks. The hierarchical planning scheme decouples the formation of teams, the task assignment among teams and the task coordination within each team. Human-in-the-loop simulations suggest that the framework can be applied to practical relief missions after disasters, where the operator can coordinate and supervise large-scale fleets efficiently.**

***Index Terms*—Multi-robot systems, human-centric coordination, task assignment, hierarchical planning.**

The authors are with [1] the School of Advanced Manufacturing and Robotics, Peking University, Beijing 100871, China; and [2]National University of Defense Technology, Hunan 410073, China. This work was supported by the National Natural Science Foundation of China (NSFC) under grants U2241214, T2121002. Corresponding author: Meng Guo, meng.guo@pku.edu.cn.

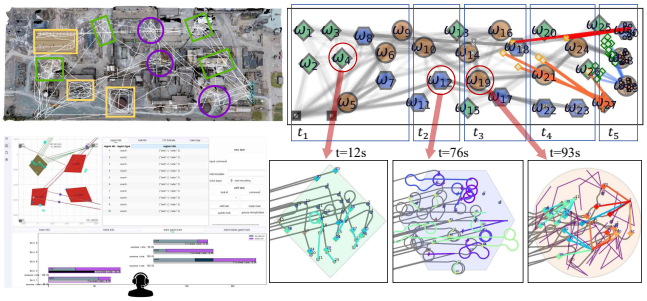


Fig. 1. The considered scenario where a human operator coordinates, interacts, and supervises a large robotic fleet via the proposed framework. Online requests are specified via the designed interface and protocol (**bottom-left**), such as adding or cancelling new temporal tasks, different priorities and deadlines, or direct assignment of certain robots. The fleet reacts autonomously to fulfill different tasks at different regions, both at the global fleet-level (**top**) and the local team-level (**bottom-right**). In the simulated case, 30 tasks and 450 subtasks are accomplished by 80 robots.

I. Introduction

HETEROGENEOUS robotic fleets, combining ground and aerial vehicles, are increasingly adopted for missions that exceed the capabilities of individual robots [1], [2]. Concurrent operation improves efficiency, while collaboration enables functions such as formation, cooperative transport, and coverage [3], [4]. Despite these advantages, coordination of large fleets remains a central challenge, especially for missions with distributed subtasks requiring temporal and spatial constraints [5]. The task allocation problem grows combinatorially with fleet size and mission length [6]. Many existing approaches compute assignments offline for static task sets [7], but such assumptions are impractical for deployments where objectives evolve or are canceled in real time. Dynamic environments demand continual replanning, yet conventional methods rapidly become intractable and unstable, often leading to oscillatory allocations and degraded team performance [8].

Moreover, in practice, robotic fleets rarely operate in full autonomy. As shown in Fig. 1, human operators frequently supervise high-level decision-making, including mission specification, cancellation of outdated objectives, and adjustment of task priorities [1], [9]. During execution, the operator may need to monitor mission progress, inspect robot status, reassign subtasks, or intervene under uncertainty [10]. Efficient procedures for human–fleet interaction are therefore crucial to ensure responsiveness and safety during operation. However, most existing research either emphasizes fully autonomous

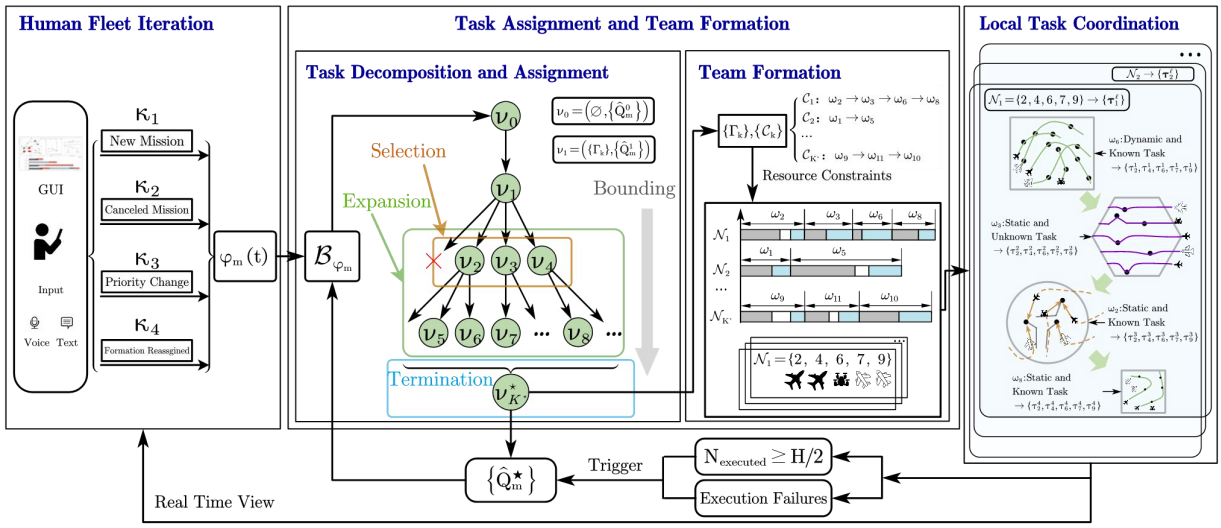


Fig. 2. The proposed human-centric framework for hierarchical coordination and supervision of robotic fleets, consisting of **three** main components: (I) the interaction protocol and interface for four types of online requests (**left**); (II) the automata-guided task assignment and team formation (**middle**); (III) three types of local coordination strategy for different tasks (**right**). Human requests, environment state and execution status are all updated online, for which adaptations for tasks and motions are triggered at different granularities and conditions.

coordination [11] or restricts human–robot interaction to very small teams [9]. Recent efforts highlight scalable approaches to interactive fleet learning [8], but protocols and interfaces for large heterogeneous fleets under dynamic and partially unknown conditions remain largely unexplored.

To address these challenges, this work introduces HECTOR, a unified framework for human-centric supervision and coordination of large-scale fleets in partially unknown and dynamic environments. Team-wise collaborative missions are specified and released online in response to observations during execution, so the distribution and requirements of tasks are revealed only at runtime. As illustrated in Fig. 2, the framework employs a hierarchical hybrid architecture with three layers: (I) a human–fleet interaction layer that converts online observations into team-wise missions; (II) a task-assignment layer that decomposes missions into collaborative tasks and allocates them to composed teams using a receding-horizon strategy; and (III) a local coordination layer that manages intra-team execution through appropriate distributed strategies. The interaction protocol supports bidirectional and multimodal requests between the operator and the fleet, including real-time visualization and supervision. Collaborative missions are expressed as linear temporal logic formulas over parameterized utility functions, providing both generality and formality in specification. The framework is fully online and adapts continuously to operator inputs, external events, and runtime observations. Its efficiency and robustness are validated through practical human-in-the-loop simulations with large heterogeneous systems.

The main contributions of this work are twofold: (I) a flexible and scalable framework enabling operators to interact with and coordinate large-scale robotic fleets in uncertain and dynamic settings; and (II) explicit treatment of practical aspects such as heterogeneous capabilities, parameterized collaboration utilities, temporal task constraints, and variability in the number and distribution of subtasks.

II. Related Work

A. Task Planning for Robotic Fleets

Task planning in robotic fleets addresses the decomposition of missions into subtasks and their allocation across robots or teams. Surveys on multi-robot task allocation (MRTA) [12], [13] have established taxonomies of problem structures, including vehicle routing, scheduling, and coalition formation [14]. Centralized optimization methods such as MILP and heuristic search [15] generate globally near-optimal solutions and allow precise encoding of resource and time constraints, but the combinatorial complexity makes them unsuitable for large fleets or missions with dynamic updates. Decentralized approaches improve scalability and resilience by distributing decision-making, such as market-based auctions [16] that trade computational effort for speed, and distributed optimization frameworks [17] that enable concurrent negotiation among agents. Approximate methods, including evolutionary algorithms [18] and genetic search [19], reduce computational overhead and can adapt to heterogeneous capabilities, but they often sacrifice guarantees of feasibility or temporal optimality. Recent efforts emphasize hierarchical decomposition and reactive coordination to improve scalability under dynamic task streams, including decomposition-based hierarchical planning [20], [21] and real-time reactive allocation under temporal logic [22]. Despite these advances, most approaches assume static and fully known tasks at the planning stage. Efficient solutions that adapt to the continual online task generation, uncertainties in subtask number and distribution, and online operator interventions remain open.

B. Complex Tasks as Temporal Logic Formulas

Temporal logic has emerged as a principled way to specify robotic missions that combine safety, temporal order, and collaboration requirements. Probabilistic temporal logics (PCTL) [23] enable reasoning about uncertainty in outcomes,

TABLE I
COMPARISON WITH RELATED WORK

Method	Task Specification	Online Task	Uncertain/ Unknown Subtasks	Human Interaction	Dynamic Constraints	Local Coordination
STyLuS* [5]	LTL	×	×	×	√	×
ScRATCHeS [21]	CaTL	×	×	×	√	×
DecTree [22]	LTL	√	√	×	×	×
cLTL+ [26]	cLTL	×	×	×	√	×
HieraTL [33]	hLTL	√	√	×	×	×
Flow-based [34]	Ordered	×	√	×	×	×
HULK [35]	LTL	√	√	×	√	√
HECTOR (Ours)	LTL	√	√	√	√	√

while LTL provides a rich language for describing sequential objectives such as repeated patrolling, collaborative capture, or persistent surveillance [5], [21], [24], [25]. Counting LTL (cLTL) [26] extends this by capturing quantitative requirements, for instance requiring a minimum number of robots at specific subtasks. Capability LTL [27], [28] address the time-dependent capabilities of robots during execution. Hierarchical LTL is proposed in [20] to further introduce structured analyses and synthesis to improve efficiency. Centralized formulations ensure completeness and optimality through sampling-based search [5], automaton–system synchronization [24], and MILP encoding of task constraints [21], [25], [26], [29], but they face double-exponential growth with task and fleet size. Decentralized methods address scalability through local coordination [30], decision trees [22], partial-order analyses [31], reactive synthesis [32], and hierarchical decoupling [33]. More recent works emphasize dynamic reactivity by combining logic specifications with scalable heuristics [21], [22], [28]. As summarized in Table I, scalability in complex missions has been addressed through mechanisms such as sampling [5], capacity-based optimization [21], counting constraints [26], decision tree [22], and flow-based construction [34]. Nevertheless, most aforementioned work overlook the online interaction with human operators, and different types of local tasks contained within the temporal mission, which in itself requires coordination and adaptation. Such nested coordination raises key challenges in retaining tractable computational complexity while enabling coherent adaptation across multiple spatial, temporal and organizational scales.

As the most relevant to this work, HULK [35] facilitates continual task planning by modeling missions constraints as posets. However, this approach necessitates reconstruction whenever missions are modified online, limiting its online responsiveness. Furthermore, it fails to account for the interplay between high-level coordination and low-level motion feasibility, as well as the integration of human supervision during execution. The proposed method addresses these shortcomings by incorporating a human interaction protocol and an online adaptation algorithm, further integrating dynamic constraints into motion feasibility. Additionally, it enhances computational efficiency by eliminating the need for poset reconstruction and employing an automata-guided search instead.

C. Human-fleet Interaction

Beyond automated task planning, effective human-swarm collaboration under uncertainty remains a central bottleneck, especially in deciding when and how operators should validate and intervene online [1], [9]. In practice, no universal metric reliably predicts whether an autonomous plan will remain operationally sound once execution begins, so human expertise is essential for validating reasoning quality and mission viability. Human–swarm interaction emphasizes interface design, situational awareness, and scalable mechanisms for directing collectives [36]. Mixed-initiative approaches [37] dynamically shift decision authority between humans and autonomy, supporting conflict resolution and efficient replanning in uncertain missions. Similar approaches are proposed in [38], [39] to compute least violating plans. Scaled-autonomy frameworks [10] extend this by allowing robots to modulate the level of assistance provided to the operator based on task load. Interactive fleet learning [8] integrates demonstration, online supervision, and data aggregation to refine fleet policies over time. However, many existing paradigms provide limited support for online validation and intervention, often restricting operators to static displays or post-hoc plans. In dynamic settings, this can force a fallback to teleoperation, which scales poorly with the fleet size due to increasing workload and latency. Consequently, interaction protocols that enable sparse and high-level oversight while preserving scalable autonomy remain insufficiently addressed.

III. PRELIMINARY

A. Linear Temporal Logic (LTL)

Linear Temporal Logic (LTL) formulas are composed of a set of atomic propositions AP together with Boolean and temporal operators. Atomic propositions are Boolean variables representing elementary facts in the system, which can be either true or false depending on the state. The syntax [40] is given as follows:

$$\varphi \triangleq \top \mid p \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid \bigcirc \varphi \mid \varphi_1 \mathsf{U} \varphi_2,$$

where $\top \triangleq \texttt{True}$, $p \in AP$, $\bigcirc \triangleq$ *next*, $\mathsf{U} \triangleq$ *until*, and $\bot \triangleq \neg\top$. Other operators such as $\Box \triangleq$ *always*, $\Diamond \triangleq$ *eventually*, and $\Rightarrow \triangleq$ *implication* can be derived as abbreviations. LTL is widely used in robotics for specifying high-level missions such as "eventually visit region A and then always avoid region B."

Formally, an infinite word w over the alphabet 2^{AP} is defined as an infinite sequence $W \triangleq \sigma_1\sigma_2\cdots$, with $\sigma_i \in 2^{AP}$. The language of a formula φ is defined as the set of words that satisfy it, namely $\mathcal{L} \triangleq Words(\varphi) = \{W \mid W \models \varphi\}$, where $\models$ denotes the satisfaction relation. A particularly useful subclass is the *co-safe* formulas, which can be satisfied by a finite sequence of words. They involve only the operators $\bigcirc$, U, and $\Diamond$, and are expressed in positive normal form. This property is advantageous in planning problems, since many practical missions can be verified after a finite execution, avoiding the need for reasoning over infinite traces.

B. Nondeterministic Büchi Automaton

Given an LTL formula φ, one can construct an equivalent automaton that accepts exactly the set of words satisfying φ. A common representation is the Nondeterministic Büchi Automaton (NBA), defined as follows.

Definition 1 (NBA)**.** A NBA $\mathcal{B}$ is a 5-tuple $\mathcal{B} \triangleq (Q, Q_0, \Sigma, \delta, Q_F)$, where Q is the set of states; $Q_0 \subseteq Q$ is the set of initial states; $\Sigma = AP$ is the set of alphabets; $\delta : Q \times \Sigma \to 2^Q$ is the nondeterministic transition relation; and $Q_F \subseteq Q$ is the set of *accepting* states. ■

Given an infinite word $w \triangleq \sigma_1\sigma_2\cdots$, the resulting *run* [40] within $\mathcal{B}$ is an infinite sequence $\rho \triangleq q_0q_1q_2\cdots$ such that $q_0 \in Q_0$, $q_i \in Q$, and $q_{i+1} \in \delta(q_i, \sigma_i)$ hold for all $i \geqslant 0$. A run is *accepting* if it visits accepting states infinitely often, i.e., $\mathsf{inf}(\rho) \cap Q_F \neq \varnothing$, where $\mathsf{inf}(\rho)$ is the set of states that appear infinitely often in the sequence of ρ. Such accepting runs are commonly represented in a prefix–suffix form, where the prefix reaches an accepting state, and the suffix is a cycle that revisits this state infinitely. This construction is fundamental in temporal-logic planning, but the size of $\mathcal{B}$ can grow double exponentially with the length of φ, which poses scalability challenges in complex missions.

IV. Problem Formulation

A. Multi-robot Systems

Consider a team of N robots denoted by $\mathcal{N} \triangleq \{1, \cdots, N\}$, operating in a shared workspace $\mathcal{W} \subset \mathbb{R}^3$. Each robot $i \in \mathcal{N}$ is characterized by its position $x_i \in \mathcal{W}$, a reference velocity $v_i \in \mathbb{R}^3$, and a set of primitive actions $\mathcal{A}_i$. A robot can navigate freely in $\mathcal{W}$ according to v_i, and can execute one action from $\mathcal{A}_i$ at a time. The local plan of robot i is expressed as a sequence of timed actions:

$$\tau_i \triangleq (t_i^1,\, p_i^1,\, a_i^1)(t_i^2,\, p_i^2,\, a_i^2)\cdots, \tag{1}$$

where $t_i^\ell \geqslant 0$ denotes the time instant; $p_i^\ell \in \mathcal{W}$ the goal position; and $a_i^\ell \in \mathcal{A}_i$ the action to be executed. Thus, robot i navigates to p_i^ℓ with velocity v_i and initiates action a_i^ℓ from time t_i^ℓ, for all $\ell \geqslant 1$. This representation encodes both mobility and task execution, which allows the fleet to be described as a set of inter-dependent and timed action sequences that must be coordinated at scale.

B. Online Requests from Human Operator

During execution, the operator can adapt the behavior of the fleet through online requests, which may introduce new missions or modify existing ones. Unlike offline formulations that assume static task sets, here the requests are treated as dynamic events that evolve over time and directly affect planning decisions. Four different types of requests are defined with explicit parameters as follows.

(I) A *new mission* can be released at time $t > 0$ defined as $\kappa_1(t) \triangleq (\varphi_t,\, t)$, where

$$\varphi_t \triangleq \text{sc-LTL}(\boldsymbol{\omega}_t), \tag{2}$$

is a syntactically co-safe LTL specification over the set of collaborative tasks $\boldsymbol{\omega}_t \triangleq \{\omega_1, \cdots, \omega_{M_t}\}$. Each collaborative task ω_m is defined as:

$$\omega_m \triangleq \Big(S_m,\, \eta_m,\, \{(n_j,\, a_j,\, s_j), j = 1, \cdots, J_m\}\Big), \tag{3}$$

where $S_m \subset \mathcal{W}$ is the region of execution; (n_j, a_j, s_j) is a subtask requiring at least n_j robots to perform action a_j at location s_j; and J_m is the number of subtasks. The function $\eta_m : \mathbb{N} \times \mathcal{A} \times 2^{\mathcal{N}} \to \mathbb{R}^+$ defines the estimated duration, i.e., $\eta_m(n_j, a_j, \mathcal{N}_j)$ gives the completion time if subteam $\mathcal{N}_j \subseteq \mathcal{N}$ executes action a_j at s_j. The logical composition of tasks follows the syntax in Sec. III-A, and satisfaction is defined through the relation $\models$. Thus, the missions accumulate as $\boldsymbol{\varphi}_t \triangleq \{\varphi_{t_\ell},\ \forall t_\ell \leqslant t\}$. Note that uncertainty is inherent: both the number of subtasks J_m and their locations $\{s_j\}$ may be unknown at release, requiring redundancy in team formation and adaptive coordination during execution.

Remark 1. The task definition in (3) differs from cLTL-based formulations in [26] in two aspects: (I) both task locations and the number of subtasks may be uncertain, and (II) the subtask duration depends on the assigned robots, rather than being instantaneous [31]. These extensions better reflect real-world missions such as search and rescue, where the environment reveals subtasks progressively. ■

Remark 2. The duration function $\eta_m(\cdot)$ typically saturates, where the marginal benefit of adding robots decreases with team size. This property is widely adopted in generic task models [1], [2], [41], yet it is often neglected in temporal-logic planning [5], [20], [21], [22], [25], [28]. Capturing this effect is crucial to avoid over-allocation and to ensure efficient use of heterogeneous resources. ■

(II) Previous missions *can be cancelled* by $\kappa_2(t) \triangleq \{\varphi_{t_\ell}\}$, where $\varphi_{t_\ell} \in \boldsymbol{\varphi}_t$ is the mission that has not yet been completed and should be cancelled. This captures the practical need to revoke outdated or invalid goals without disrupting other ongoing tasks and plans.

(III) *Deadlines and priorities* of existing missions can be modified by $\kappa_3(t) \triangleq \{(\varphi_{t_\ell},\, d_\ell^\star,\, w_\ell^\star)\}$, where $\varphi_{t_\ell} \in \boldsymbol{\varphi}_t$ is the mission to be modified, $d_\ell^\star > 0$ is the updated deadline, and $w_\ell^\star > 0$ is the revised priority. This reflects the ability of the operator to shift focus between competing objectives depending on urgency and available resources.

(IV) *Robots may be reassigned* across missions manually by $\kappa_4(t) \triangleq \{(\mathcal{N}_\ell,\, \varphi_{t_\ell})\}$, where the subset of robots $\mathcal{N}_\ell \subseteq \mathcal{N}$ should be assigned to mission $\varphi_{t_\ell} \in \boldsymbol{\varphi}_t$. This allows explicit operator control over resource allocation in situations where autonomous assignment may not match human intent.

Thus, the evolving set of operator requests up to time $t \geqslant 0$ is denoted by the set below:

$$\mathcal{K}(t) \triangleq \big\{\kappa_1(t), \kappa_2(t), \kappa_3(t), \kappa_4(t)\big\}, \tag{4}$$

where fulfilled requests are removed upon completion. This formulation enables a closed-loop interaction in which operator input and autonomous planning mutually adapt over time.

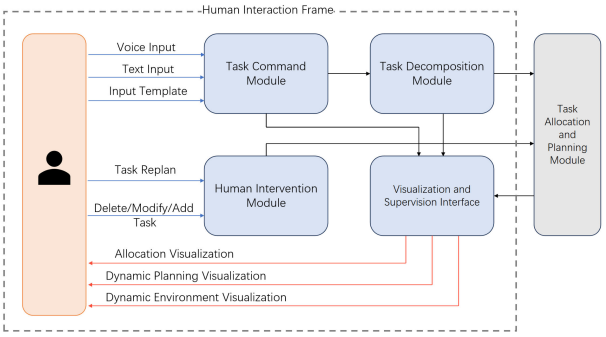


Fig. 3. The proposed protocols for human-fleet interaction: the human command and intervention module (**left**); the task decomposition and planning modules (**right**); and the visualization module (**middle**).

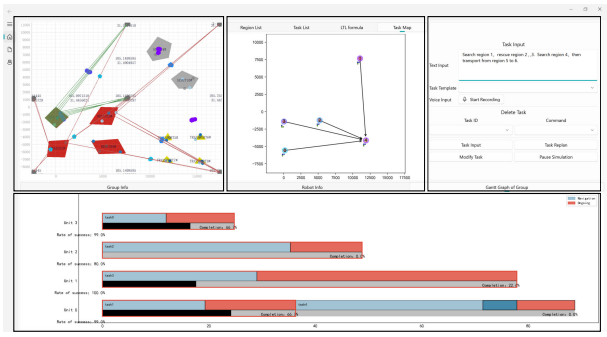


Fig. 4. Integrated interface for human-fleet interaction, which includes scenario visualization (**top-left**), temporal ordering of subtasks (**top-middle**), panel to specify operator requests (**top-right**), local plans and execution progress for teams and robots are visualized online (**bottom**).

C. Problem Statement

Given the mission specifications in (2) and additional operator requests in (4), the overall objective is to synthesize collective plans in (1) such that the average mission response time is minimized, i.e.,

$$\mathbf{min}_{\{\tau_i\}} \frac{\sum_{\varphi_{t_\ell} \in \boldsymbol{\varphi}_t} (t_\ell^{\mathrm{f}} - t_\ell)}{|\boldsymbol{\varphi}_t|}, \tag{5}$$

where t_ℓ and t_ℓ^{f} denote the release and completion times of mission φ_{t_ℓ}. This objective explicitly measures the responsiveness of the fleet, which is critical in dynamic and safety-critical environments such as disaster relief or surveillance. All operator requests in $\mathcal{K}(t)$ must be satisfied during planning and execution as described earlier, ensuring that human interventions are seamlessly integrated into the autonomous coordination framework.

Remark 3. The problem combines temporal missions with online operator requests, via introducing cancellations, priority updates and uncertain subtasks. Centralized MILP [13], [15] and decentralized market-based methods [16], [17] typically assume static tasks, while temporal-logic approaches [5], [24], [26] focus on fixed specifications. Existing frameworks for human–fleet interaction [1], [10] also target simpler settings, leaving the proposed problem insufficiently addressed. The integration of temporal-logic missions, dynamic operator requests, and large-scale multi-robot coordination therefore represents a significant gap that this work aims to fill. ■

V. Proposed Solution

The proposed solution consists of three main components as shown in Fig. 2: (I) the protocol and interface of human-fleet interaction, along with the hierarchical coordination and communication in Sec. V-A; (II) the receding-horizon task planning algorithm that assigns tasks to subteams of robots given the global mission specification and constraints on the resources in Sec. V-B; (III) the local coordination algorithm that assigns subtasks to robots during online execution in Sec. V-C. The synergy and adaptation of these components are triggered by online observations, human requests and execution status. For clarity, a table of key variables in this work is provided in the Appendix.

A. Protocol and Interface of Human-fleet Interaction

This section introduces the communication and coordination framework that enables online human–fleet interaction. It first presents the protocol that maps operator requests to system modules, then describes the graphical interface that supports multimodal inputs and visual supervision, and finally outlines the hierarchical communication structure between the operator, team leaders, and team members.

1) Protocol for Online Requests: The communication protocol links the four operator request types in (4) to the interaction module. As shown in Fig. 3, when a new mission is issued through request κ_1, the input is provided via voice, text, or templates, processed by the "task command module" and forwarded to the "task decomposition module," which generates a sc-LTL specification with spatial-temporal constraints for the planning pipeline. Requests for cancellation κ_2, task modification κ_3, and robot reassignment κ_4 are processed by the "human intervention module," which updates active missions and resources, passing changes to the "task allocation and planning module." Ongoing executions that cannot be interrupted are preserved. The protocol ensures closed-loop communication, where upward flows transmit operator directives to the planning logic and downward flows provide feedback on planning results, execution progress, and environmental changes. This feedback is visualized via the "visualization and supervision interface," displaying allocation maps, task graphs, and execution timelines. This setup ensures that operator interventions are executed and monitored in real-time throughout the mission lifecycle.

2) Design of Graphical Interface for Online Interactions: The graphical interface in Fig. 4 integrates the online interaction protocol with real-time visualization, linking operator inputs to the corresponding modules. On the right panel, the operator can release new missions through text, templates or voice, which are processed by the task command module to fulfill request κ_1. The same panel allows cancellation and priority updates, routed to the human intervention module for requests κ_2 and κ_3. Robot reassignment for request κ_4 is also supported. Responses are reflected in multiple interactive panels: the scene map displays task distribution, robot positions and trajectories; the mission panel shows the decomposed task graph and LTL formulas; and Gantt charts summarize task allocations and execution progress both for teams and

robots. This interface closes the loop between human input, intermediate results and fleet execution, enabling transparent supervision and efficient adaptation.

3) Hierarchical Coordination and Communication: The communication architecture follows a hierarchical structure that mirrors the organization of the fleet. At the top level, the operator interacts with designated team leaders through the task command and intervention modules, issuing high-level missions, modifying specifications, and adjusting deadlines or resource allocations [42]. Each team leader translates these directives into concrete plans for its team, forming subgroups when necessary and allocating robots to subtasks according to capabilities and availability [36]. At the lower level, team members communicate directly with their leader to receive local assignments and report execution status, enabling rapid coordination without overwhelming the operator with low-level details. This structure improves scalability by limiting communication overhead and enhances robustness by isolating local replanning within teams, while maintaining consistency since operator requests are propagated downward and execution feedback is aggregated upward in a structured manner.

B. Task Assignment and Team Formation

1) Simultaneous Task Decomposition and Team Assignment: Existing work on task coordination for multi-robot systems under temporal logic specifications often relies on the synchronized product between the task automaton $\mathcal{B}_{\varphi_t}$ and the global system model, which is constructed as the product of all local robot models [24], [25]. This approach guarantees correctness but suffers from double-exponential growth in computational complexity. To mitigate this challenge, several methods have been proposed, including local coordination [30], decision trees [22], distributed sampling [5], hierarchical decoupling [33], and partial-order analyses [31]. Although these methods improve scalability in static environments, they are not suitable for open-world scenarios where new tasks may be triggered online. In such settings, recomputation of the entire system model would be required after each update, which invalidates previously computed results and leads to inefficiency.

Consider the Büchi automaton $\mathcal{B}_{\varphi_m} \triangleq (Q^m, Q_0^m, \Sigma^m, \delta^m, Q_F^m)$ associated with the mission specification $\varphi_m \in \Phi_t$, where $\Phi_t \triangleq \{\varphi_1, \cdots, \varphi_M\}$ is the set of missions known at time $t > 0$ and $\mathcal{M} \triangleq \{1, \cdots, M\}$. Moreover, the fleet is divided into K teams denoted by $\mathcal{C}_k \subset \mathcal{N}$ and $\mathcal{K} \triangleq \{1, \cdots, K\}$, of which the exact value of K is to be decided. Each team $\mathcal{C}_k$ is defined not by a fixed set of robots but by its composition, i.e., the available number of each robot type with associated capabilities. Denote by $\{\Gamma_k, k \in \mathcal{K}\}$ the local plans of teams $\{\mathcal{C}_k\}$ as the sequences of tasks to be accomplished.

Problem 1. Given the mission specifications $\varphi_m \in \Phi_t$, determine the optimal number of teams K, the composition of each team $\{\mathcal{C}_k\}$, and the local plans $\{\Gamma_k\}$ such that: (I) the tasks can be executed by the assigned team under the temporal-logic constraints and the robot capacity constraints; and (II) the execution time for missions is minimized as in (5). ■

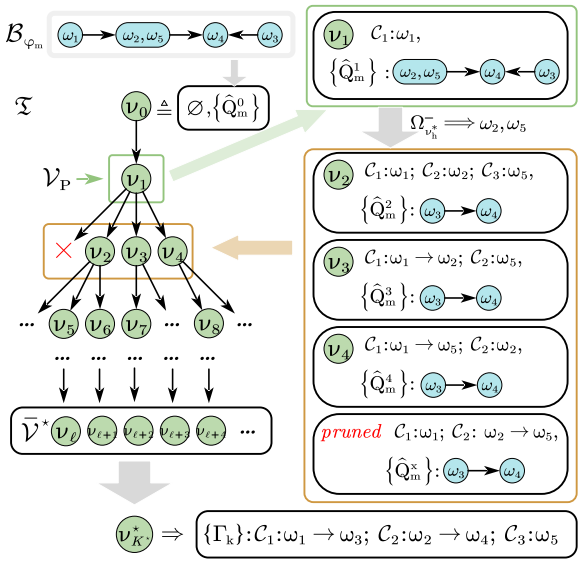


Fig. 5. Illustration of the automaton-guided search tree, where the node expands over not only the local plans of each team $\{\Gamma_k, k \in \mathcal{K}\}$, but also the progress of each mission $\{\hat{Q}_m, m \in \mathcal{M}\}$. The leaf nodes of complete assignments may correspond to different number of teams.

The search structure is organized as a tree $\mathfrak{T} \triangleq (\mathcal{V}, \to)$, where $\mathcal{V} \triangleq \{\nu\}$ is the set of nodes, and $\to \subset \mathcal{V} \times \mathcal{V}$ defines the edges. Each node $\nu \triangleq (\{\Gamma_k, k \in \mathcal{K}\}, \{\hat{Q}_m, m \in \mathcal{M}\})$ contains two components: the partial local plans of all teams $\{\mathcal{C}_k\}$ and the sets of current reachable states in $\mathcal{B}_{\varphi_m}$ for all missions $\varphi_m \in \Phi_t$. The root node is $\nu_0 \triangleq (\varnothing, \{\hat{Q}_m^0, m \in \mathcal{M}\})$. As summarized in Algorithm 1 and illustrated in Fig. 5, the proposed procedure consists of the following four stages.

(I) Selection. A set of $P > 0$ candidate nodes is selected for parallel expansion, i.e., $\mathcal{V}_P \triangleq \{\nu_1^\star, \cdots, \nu_P^\star\}$, where $\nu_p^\star \triangleq$ **argmin-p**$_{\nu \in \mathcal{V}} \{\chi(\nu)\}$ for $p = 1, \cdots, P$. The value function $\chi : \mathcal{V} \to \mathbb{R}^+$ is defined as:

$$\begin{aligned} \chi(\nu) \triangleq & \mathbf{max}_{k \in \mathcal{K}} \{T_k\} + \eta_1 \sum_{k \in \mathcal{K}} C_k \\ & + \eta_2 \sum_{m \in \mathcal{M}} \mathbf{w}_m \min_{q_m \in \hat{Q}_m} \psi(q_m, Q_F^m), \end{aligned} \tag{6}$$

where $\eta_1, \eta_2 > 0$ are weighting parameters; $\mathbf{w}_m$ denotes the preference weight associated with mission φ_m, reflecting human-specified priorities; $T_k > 0$ is the ending time of the local plan Γ_k; $C_k > 0$ is the estimated cost of Γ_k; and $\psi(q_m, Q_F^m) > 0$ returns the length of the shortest path from state q_m to any final state in Q_F^m. The first term measures the makespan of the current assignments, the second term estimates the overall cost, and the third term accounts for the overall progress of missions.

Remark 4. Parameters $\eta_1, \eta_2 > 0$ above regularize the search process, i.e., η_1 penalizes nodes with equal makespan based on cost, while η_2 prioritizes nodes with greater mission progress via ψ. These weights ensure makespan minimization while improving the search efficiency, with their impact primarily on convergence rate rather than optimality. ■

Moreover, given a node $\nu \in \mathcal{V}_P$ and the associated local plans $\{\Gamma_k\}$, the capacity constraints of each team $k \in \mathcal{K}$ under the updated assignment are given by:

$$\mathcal{C}_k \triangleq \Big(\{(\beta_k^j,\, a^j),\, a^j \in \omega_k^j\},\, \forall \omega_k^j \in \Gamma_k\Big), \tag{7}$$

which specifies the minimum number of robots $\beta_k^j > 0$ to perform action $a^j \in \mathcal{A}$ for each assigned task $\omega_k^j \in \Gamma_k$. This requirement provides actions and counts, without binding specific robots, and thereby decouples assignment from scheduling. Moreover, if the capacity constraints above exceed the overall fleet capacity, i.e., the following condition:

$$\sum_{k\in\mathcal{K}} \beta_k^j \leqslant \sum_{i\in\mathcal{N}} \mathbb{1}(a^j \in \mathcal{A}_i),\ \forall a^j \in \mathcal{A}; \tag{8}$$

is violated, where the left-hand side is the required capacity and the right-hand side is the available fleet capacity. Then this node is marked infeasible and excluded from the set $\mathcal{V}_P$.

(II) Expansion. Each selected node $\nu_h^\star \in \mathcal{V}_P$ is expanded by assigning an additional task to one team. The set of candidate tasks is defined as below:

$$\Omega^-_{\nu_h^\star} \triangleq \bigcup_{m\in\mathcal{M}} \Big\{\omega \in \Sigma^m \,\big|\, \exists q_m \in \widehat{Q}_m : \delta^m(q_m,\omega) \in Q^m\Big\}, \tag{9}$$

where ω is an atomic task symbol from the alphabet Σ^m and $\delta^m : Q^m \times \Sigma^m \to 2^{Q^m}$ is the transition function of the Büchi automaton $\mathcal{B}_{\varphi_m}$. Thus, a candidate task must enable a valid state transition for at least one automaton $\mathcal{B}_{\varphi_m}$. For each $\omega \in \Omega^-_{\nu_h^\star}$, a child node is created by augmenting the local plan Γ_k of team $\mathcal{C}_k$ with ω, namely:

$$\nu^+ \triangleq \Big(\{\Gamma_1, \cdots, \Gamma_k^+, \cdots, \Gamma_K\},\, \{\widehat{Q}_m^+\}_{m\in\mathcal{M}}\Big), \tag{10}$$

where Γ_k^+ appends ω to Γ_k; the task set Ω_k within Γ_k is updated by adding ω; and $\widehat{Q}_m^+ \triangleq \{\delta^m(q_m,\omega) \,|\, q_m \in \widehat{Q}_m\}$ is the updated set of reachable states for mission φ_m. Consequently, the edge $(\nu_h^\star,\, \nu^+)$ is inserted into the search tree $\mathfrak{T}$. If no valid and feasible transition exists, then $\Omega^-_{\nu_h^\star} = \varnothing$ and the node cannot be expanded. In addition, a new team $(K+1)$ can be created first by specifying its composition; a candidate task $\omega \in \Omega^-_{\nu_h^\star}$ is then assigned to it, and its capacity constraints $\mathcal{C}_{K+1}$ are computed by (7). Hence, task assignment and team formation emerge naturally during the search.

More importantly, after assigning task ω to team $k \in \mathcal{K}$, denoted by ω_k^ℓ, the associated capacity constraint $\mathcal{C}_k$ for this team is updated as follows:

$$\beta_k^j \triangleq \max_{a^j \in \omega_k^\ell \in \Gamma_k} \{n_j\},\ \forall a^j \in \mathcal{A}; \tag{11}$$

i.e., the maximum number of robots β_k^j required for each action a^j across the sequence of tasks $\omega_k^{L_k}$. The related ending time is updated by:

$$\begin{aligned} t_{\mathrm{e}}(\omega_k^\ell) \triangleq \max_{\omega_j \in \mathtt{Pre}(\omega_k^\ell)} &\big\{t_{\mathrm{e}}(\omega_j)\big\} \\ &+ T_{\mathtt{nav}}(S_k^{\ell-1}, S_k^\ell) + T_{\mathtt{exec}}(\omega_k^\ell), \end{aligned} \tag{12}$$

where $t_{\mathrm{e}}(\omega_k^\ell)$ is the estimated ending time of $\omega_k^\ell \in \Gamma_k$; $\mathtt{Pre}(\omega_k^\ell)$ is the set of assigned tasks; and $T_{\mathtt{nav}}(S_k^{\ell-1}, S_k^\ell)$ is the estimated navigation duration between regions.

Algorithm 1: Simultaneous Task Decomposition and Team Assignment

Input: $\Phi_t = \{\varphi_1, \cdots, \varphi_M\}$, automata $\{\mathcal{B}_{\varphi_m}\}$, fleet $\mathcal{N}$.
Output: Teams K, capacities $\{\mathcal{C}_k\}$, plans $\{\Gamma_k\}$.

Initialize $\mathfrak{T} = (\mathcal{V}, \to)$, $\mathcal{V} = \{\nu_0\}$;
while *not terminated* **do**
  /* **Selection** */
  $\mathcal{V}_P \leftarrow \{\nu_1^\star, \cdots, \nu_P^\star\}$ by $\chi(\nu)$ in (6);
  Initialize $\mathcal{C}_k$ for all $k \in \mathcal{K}$ via (7);
  /* **Expansion** */
  foreach $\nu_h^\star \in \mathcal{V}_P$ **do**
    Obtain $\Omega^-_{\nu_h^\star}$ by (9);
    foreach $\omega \in \Omega^-_{\nu_h^\star}$ **do**
      if *add new team* **then**
        Create $\mathcal{C}_{K+1}$, set $\Gamma_{K+1} = \varnothing$;
        Update $\mathcal{K}$;
      foreach *team* $k \in \mathcal{K}$ **do**
        Generate ν^+ by adding ω to Γ_k as (10);
        Update $\widehat{Q}_m^+$ for all m;
        Add $(\nu_h^\star, \nu^+)$ to $\mathfrak{T}$;
        Update $\mathcal{C}_k$ via (11);
        Update $t_{\mathrm{e}}(\omega_k^\ell)$ via (12);
        if (8) *violated* **then** mark ν^+ infeasible;
  /* **Bounding** */
  foreach $\nu \in \mathcal{V}$ **do**
    Compute $\zeta(\nu)$ by (13);
    Prune dominated nodes by (14);
  Update $\overline{\mathcal{V}}$ by (15);
/* **Termination** */
Compute $\overline{\mathcal{V}}^\star$ by (16), select $\nu_{K^\star}^\star$ by (17);
return Optimal K, $\{\mathcal{C}_k\}$, $\{\Gamma_k\}$;

(III) Bounding. To reduce unnecessary exploration, each node is evaluated through a *performance profile* that retains detailed information about the plans of all teams and the progress of all missions, i.e.,

$$\begin{aligned} \zeta(\nu) \triangleq \Big[&\{T_k\},\, \{\mathcal{C}_k\}, \\ &\Big\{\min_{q_m \in \widehat{Q}_m} \psi_m(q_m,\, Q_F^m), m \in \mathcal{M}\Big\}\Big], \end{aligned} \tag{13}$$

where T_k is the ending time of Γ_k; C_k is the estimated cost of Ω_k; and the last term $\psi_m(\cdot)$ measures the minimum distance from the current automaton states $\widehat{Q}_m$ to the accepting set Q_F^m for each mission φ_m. The profile $\zeta(\nu)$ has dimension $(2K + M)$ and is non-negative. Given two nodes ν_1 and ν_2, node ν_1 is said to *dominate* node ν_2 if the following holds:

$$\zeta(\nu_1) \leqslant \zeta(\nu_2), \quad \text{and} \quad \zeta(\nu_1) \neq \zeta(\nu_2), \tag{14}$$

where the inequality is understood element-wise across the vector in (13). Thus, node ν_1 has no larger makespan or cost for any team, and no less mission progress for any specification, with strict improvement in at least one entry. In this case, node ν_2 is marked as dominated and excluded

from expansion. The set of all non-dominated nodes is the set of frontiers, defined as follows:

$$\overline{\mathcal{V}} \triangleq \{\nu \in \mathcal{V} \,|\, \nexists\, \nu' \in \mathcal{V} : \nu' \text{ dominates } \nu\}, \tag{15}$$

which is updated whenever a new node is added. This bounding procedure maintains only $\overline{\mathcal{V}}$ as candidates for expansion, ensuring that strictly inferior nodes are pruned and that the search focuses on promising branches of the tree $\mathfrak{T}$.

(IV) Termination. The stages of selection, expansion, and bounding are repeated until the computation budget is exhausted or no new non-dominated nodes emerge. The current set of frontier is $\overline{\mathcal{V}}$ from (15), and the subset of complete assignments is given by:

$$\overline{\mathcal{V}}^{\star} \triangleq \{\nu \in \overline{\mathcal{V}} \,|\, (\widehat{Q}_m \cap Q_F^m) \neq \varnothing,\ \forall m \in \mathcal{M}\}, \tag{16}$$

where the accepting set is reached for all missions. Thus, the optimal node is selected among these assignments, i.e.,

$$\nu_{K^\star}^{\star} \triangleq \mathbf{argmin}_{\nu \in \overline{\mathcal{V}}^{\star}} \{\chi(\nu)\}, \tag{17}$$

with $\chi(\nu)$ from (6). The associated capacity constraints $\{\mathcal{C}_k\}$ and the global assignment are specified by the resulting plans $\{\Gamma_k\}$ together with execution times.

Example 1. As illustrated in Fig. 5, given $\omega_1, \omega_2, \omega_3, \omega_4, \omega_5$ in the mission automata $\{\mathcal{B}_{\varphi_m}\}$, the node $\nu_{K^\star}^{\star}$ assigns 5 tasks to 3 subteams and the optimal $K^\star = 3$. Thus the local plans are given by executing ω_3 after ω_1 for team one; ω_4 after ω_2 for team two; and task ω_5 for team three. ■

Last but not least, due to the dynamic nature of the environment, a receding-horizon strategy is adopted for the task assignment. Thus, the search is paused when the number of assigned tasks for the fleet reaches the horizon $H > 0$ as a user-defined hyper-parameter, and is resumed at the next planning cycle as described in the sequel. Note that the horizon H balances the batch-assignment efficiency and the online responsiveness. A small H results in fragmented task sequences and frequent replanning, which under-utilizes its capability to optimize large task sets. Conversely, an excessively large H increases computational redundancy and may induce oscillation in dynamic environments, as long-term commitments are often invalidated by online mission updates. This trade-off is evaluated empirically in Sec. VI.

Remark 5. The framework presented above offers several advantages over existing approaches [5], [24], [25], [33], [31], [35]: (I) it circumvents the explicit construction of the synchronous product between the Büchi automaton, the robot models as transition systems, and the global product, which is prohibitively large in multi-robot settings; (II) the search is both anytime and complete, in contrast to mixed-integer linear programming methods that often require long solving times without intermediate feasible outputs; (III) it is well suited to partially known and dynamic environments where missions are triggered online, since a new specification can be incorporated by introducing its reachable state set $\widehat{Q}_k$ without interfering with existing missions; (IV) the stages of node selection and expansion can be carried out in parallel, enabling efficient scaling to large teams and complex mission sets; (V) the node construction and expansion above differ from the poset-based methods in [31], [35]. Instead of using partial orders, each node ν above maintains the sets of reachable states $\widehat{Q}_m$ for each mission automaton, allowing task decomposition and team assignment simultaneously. This approach eliminates the re-computation for global posets, enabling faster integration of online missions and reducing overhead pre-processing. More numerical comparisons are given in Sec. VI. ■

Correctness and completeness of the simultaneous task decomposition and team assignment algorithm above is provided below. Particularly, it is shown that the completeness and optimality hold in the static and known case with the full horizon. Moreover, in the online receding-horizon case, the algorithm can still ensure correctness and feasibility, while scalability and adaptability are evaluated empirically in Sec. VI. Proofs are provided in the Appendix.

Theorem 1. *Consider an instance of Problem 1 with fleet $\mathcal{N}$ and mission set Φ_t. Suppose all missions are known* a priori *and the planning horizon H is larger than the total number of admissible tasks. Then, the team plans $\{\Gamma_k\}$ generated by Alg. 1 satisfy: (I) all temporal constraints encoded in the mission automata $\{\mathcal{B}_{\varphi_m}\}$ hence all mission specifications; and (II) all capacity constraints in (7)–(8). Moreover, whenever feasible plans exist, Alg. 1 returns the feasible plan that minimizes the makespan objective in (6).*

Lemma 2. *Under online mission release and the receding-horizon replanning with $H < |\Omega|$, Alg. 1 guarantees that the computed partial plan ν is consistent with the active missions. Moreover, each time a new mission is added, the updated partial plan after adaptation can still preserve satisfaction under the event-triggered adaptation scheme.*

2) Capacity-based and Redundancy-aware Team Formation: Given the optimal team–task assignment $\nu_{K^\star}^{\star}$, the composition of each team remains to be determined. In particular, the capacity constraints derived earlier specify only the aggregate requirements for each team, i.e., the minimum numbers of robots with specific capabilities. The actual formation of teams requires allocating individual robots to these abstract capacities while respecting disjoint membership and redundancy margins. This problem is formulated as follows.

Problem 2. Given the assignment $\nu_{K^\star}^{\star}$ and the fleet $\mathcal{N}$, determine the team formation $\overline{\mathcal{N}} = \{\mathcal{N}_1, \cdots, \mathcal{N}_{K^\star}\}$, where $\mathcal{N}_k$ is the set of robots assigned to team $\mathcal{C}_k$. Each team must satisfy the required capacities, and membership must be disjoint, i.e., $\mathcal{N}_{k_1} \cap \mathcal{N}_{k_2} = \varnothing$ for $k_1 \neq k_2$. The objective is to minimize the overall response time as defined in (5), thereby ensuring efficient execution of all assigned tasks. ■

The team formation problem is addressed using a redundancy-aware mixed-integer linear programming (MILP) formulation. This formulation explicitly enforces the capacity constraints of each team, while also incorporating redundancy margins that allow flexibility in resource allocation. The ap-

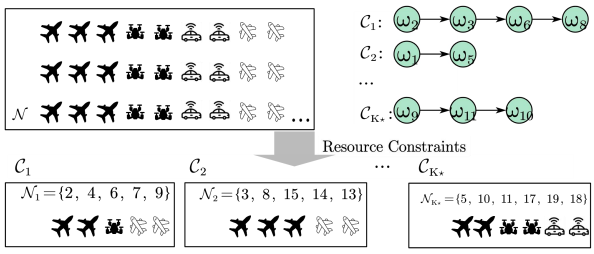


Fig. 6. Illustration of the capacity-based and redundancy-aware team formation. Given the task assignment $\nu^\star_{K^\star}$, the team formation $\{\mathcal{N}_1, \cdots, \mathcal{N}_{K^\star}\}$ are the determined optimally under the redundancy margin in (18).

proach can be summarized in two key components.

(I) *Capacity constraints.* Binary decision variables $b_{ik} \in \{0, 1\}$ are introduced to indicate whether robot $i \in \mathcal{N}$ is assigned to team $\mathcal{C}_k$. For each action $a^j \in \mathcal{A}$ required by team $\mathcal{C}_k$, the number of assigned robots must satisfy a lower bound and an upper bound as follows:

$$\beta^j_k \leqslant \sum_{i \in \mathcal{N}} b_{ik} \mathbb{1}(a_i = a^j) \leqslant \beta^j_k \alpha_j, \tag{18}$$

where β^j_k is the minimum number of robots required to perform action a^j, and $\alpha_j \geqslant 1$ is a *redundancy margin* introduced as in Remark 6. The inequality above ensures that team $\mathcal{C}_k$ is always sufficiently staffed to satisfy the task requirements, while still allowing limited redundancy given available resources within the fleet. Together, these constraints provide a trade-off between feasibility, efficiency and robustness for the team formation.

Remark 6. The margin $\{\alpha_j\}$ above accounts for (i) uncertainty in the effective workload of tasks such as unknown subtasks revealed online, and (ii) robot failure or unavailability during execution. In practice, they are selected as a small safety factor above 1 and increased when subtask-count uncertainty or failure rate is higher. Larger $\{\alpha_j\}$ improves robustness but may reduce efficiency by reserving additional robots; smaller $\{\alpha_j\}$ yields higher utilization but can be less resilient to uncertainty and failures. ■

(II) Min-max objective. The second component is to minimize the response time across all teams. For each robot i potentially assigned to team $\mathcal{C}_k$, the expected arrival time at the first task region S^1_k is $t_{ik} \triangleq \widehat{t}_i + T_{\text{nav}}(\widehat{x}_i, S^1_k)$, where $\widehat{t}_i$ is the time when robot i becomes available and $\widehat{x}_i$ is its position at that time. Once robots are assigned, the execution cost of team $\mathcal{N}_k$ is given by $J(k) \triangleq \mathbf{max}_{i \in \mathcal{N}_k}\{t_{ik}\} + \sum_{\omega \in \Gamma_k} T_{\text{exec}}(\omega)$, where the first term captures the synchronization delay of the slowest robot, and the second term aggregates execution times. The global objective is then $\mathbf{min}\{\mathbf{max}_{k \in \mathcal{K}} J(k)\}$, which minimizes the worst-case response time across all teams. This reflects the fact that overall mission efficiency is determined by the slowest team to complete its tasks.

The above formulation constitutes a MILP problem, which can be solved by off-the-shelf solvers such as `GLOP` [43]. The solution provides the optimal binary assignment $\{b_{ik}\}$, from which the team formation $\overline{\mathcal{N}}$ is derived. Specifically, the team formation is given by $\mathcal{N}_k \triangleq \{i \in \mathcal{N} \mid b_{ik} = 1\}$, which defines the robots in team $\mathcal{C}_k$. In addition, the local plan of each robot $i \in \mathcal{N}_k$ can be generated as a timed sequence of tasks, i.e.,

$$\xi_i \triangleq (S^1_k, \omega^1_k)(S^2_k, \omega^2_k) \cdots (S^{L_k}_k, \omega^{L_k}_k), \ \forall i \in \mathcal{N}_k; \tag{19}$$

where ω^ℓ_k is the ℓ-th task in team $\mathcal{C}_k$; S^ℓ_k is the associated region; and L_k is the number of tasks. This ensures that each robot has a concrete plan consistent with both the task assignment of its team and the global mission specification.

The redundancy-aware team formation is formulated as a MILP with binary membership variables $b_{ik} \in \{0, 1\}$, where $i \in \mathcal{N}$ and $k \in \{1, \cdots, K^\star\}$. This formulation introduces $\mathcal{O}(|\mathcal{N}|K^\star)$ integer variables and $\mathcal{O}(|\mathcal{A}|K^\star)$ capacity constraints in (18). While NP-hard in the worst case, the value of $K^\star$ is small in this context, ensuring that the solve time remains manageable. The best incumbent feasible formation can be returned under a limited planning time. In case of no feasible solutions, a greedy capacity-filling heuristic is employed as a fallback. Note that this team-formation MILP is significantly smaller than the direct robot-to-subtask assignment method without the hierarchical structure [26], [29]. More numerical comparisons can be found in Sec. VI.

Example 2. Fig. 6 shows the team formation induced by the optimal task assignment $\nu^\star_{K^\star}$, i.e., coalitions $\{\mathcal{C}_k\}_{k=1}^{K^\star}$. In particular, the team $\mathcal{N}_1 = \{2, 4, 6, 7, 9\}$ executes $\omega_2\omega_3\omega_6\omega_8$; the team $\mathcal{N}_2 = \{3, 8, 15, 14, 13\}$ executes $\omega_1\omega_5$; and the team $\mathcal{N}_{K^\star} = \{5, 10, 11, 17, 19, 18\}$ executes $\omega_9\omega_{11}\omega_{10}$. ■

C. Local Task and Trajectory Coordination within Teams

Given the optimal assignment $\nu^\star_{K^\star}$, the local task plan $\boldsymbol{\tau}_i$ of each robot $i \in \mathcal{N}_k$ is derived as in (19). Each robot i navigates to region S^ℓ_k to start executing its ℓ-th task $(S^\ell_k, \omega^\ell_k)$. As denoted in (3), subtasks $\mathcal{J}^\ell_k \triangleq \{(n_j, a_j, s_j), j = 1, \cdots, J^\ell_k\}$ must be considered to perform the task ω^ℓ_k. For task ω^ℓ_k assigned to team $\mathcal{N}_k$, the local plan of each robot $i \in \mathcal{N}_k$ is given by $\boldsymbol{\tau}_i \triangleq (t^1_i, p^1_i, a^1_i)(t^2_i, p^2_i, a^2_i) \cdots$, which is a sequence of timed goal positions and actions. The collective plan of the team is given by $\boldsymbol{\tau}^\ell_k \triangleq \{\boldsymbol{\tau}_i \mid i \in \mathcal{N}_k\}$.

However, the assignment of subtasks is interdependent with the optimization of the associated trajectories, as the trajectory affects the time taken for each robot to reach subtask positions and execute subtasks. The optimization problem must consider the trajectory of each robot $\mathbf{x}_i$ alongside subtask assignment, ensuring that both spatial and temporal coordination are optimized simultaneously. Thus, the objective is not only to assign subtasks to the robots but also to determine both the local plans $\{\boldsymbol{\tau}_i\}$ and trajectories $\{\mathbf{x}_i\}$ of all robots in team $\mathcal{N}_k$. This simultaneous coordination guarantees that the overall task completion is time-efficient, considering both the subtask execution and the robot motion. Optimizing trajectories is crucial to minimizing the makespan of task completion.

Problem 3. Given task ω^ℓ_k with area S^ℓ_k, subtasks $\mathcal{J}^\ell_k$ as in (3) and the assigned team $\mathcal{N}_k$, the goal is to determine the optimal local plans $\{\boldsymbol{\tau}_i\}$ and robot trajectories $\{\mathbf{x}_i\}$ for each robot $i \in \mathcal{N}_k$. The objective is to minimize the makespan T^ℓ_k as the

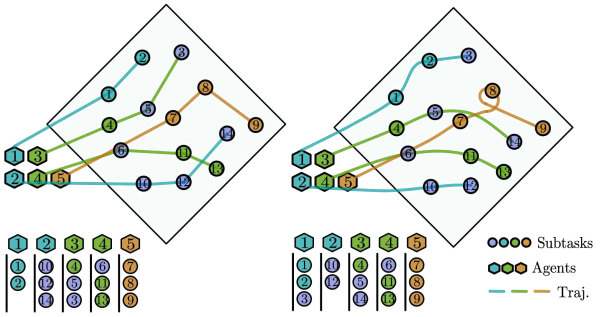


Fig. 7. Local coordination results for the static and known local tasks described in Sec. V-C1. In total 14 delivery subtasks (filled circles) are assigned to 5 robots (filled pentagons), of which the trajectories are shown for fully-actuated (**left**) and non-holonomic (**right**) robots.

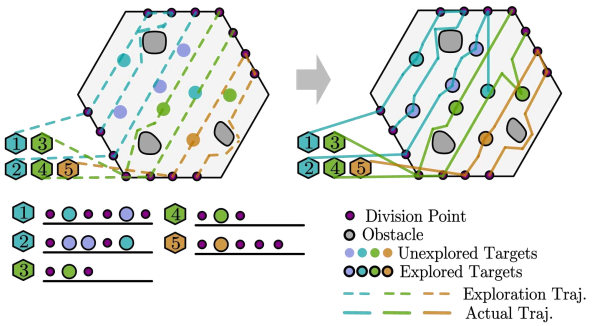


Fig. 8. Execution results for the case of static and unknown subtasks in Sec. V-C2, e.g., the "search and rescue" task. **Left**: The initial trajectories for coverage by 5 robots with unknown targets (in circles); **Right**: The actual trajectories after discovering, assigning and executing 8 rescue subtasks.

collective execution time for task ω_k^ℓ, while simultaneously optimizing trajectories and task assignments. ■

Uncertainty in the number of subtasks J_k^ℓ and their locations $\{s_j\}$ for task ω_k^ℓ requires coordination strategies conditioned on task properties. Static and known tasks reduce to allocation and trajectory optimization for makespan minimization. Static but unknown tasks require exploration with online insertion of discovered subtasks. Dynamic but known tasks require continual online reassignment and trajectory updates. These cases defined by the static–dynamic and known–unknown axes motivate tailored coordination strategies below.

1) Static and Known Subtasks: In the static and known case, both the number of subtasks J_k^ℓ and their locations $\{s_j\}$ are predetermined. This situation often arises in delivery or inspection tasks with fixed points of interest. While the setting is classical, the consideration of robot dynamics is essential: robots with free holonomic motion can move directly between subtasks, whereas robots with non-holonomic constraints must follow feasible trajectories that depend on both position and orientation. Therefore, planning in this case requires different treatments depending on the underlying motion model.

For holonomic robots, the problem reduces to a variant of the multi-vehicle routing problem (MVRP) without return. Robots travel along straight-line segments between subtasks, and the sequence of visits is determined by solving a combinatorial optimization problem, via off-the-shelf solvers such as [43]. The outcome directly specifies the local plans $\{\boldsymbol{\tau}_i\}$, while the trajectories $\{\mathbf{x}_i\}$ are straight-line connections. However, for non-holonomic robots, the problem becomes a hybrid optimization in which task sequencing and feasible trajectories are *optimized jointly*. Each subtask location is augmented with an orientation, and also the robot state. The transition between subtasks is evaluated using motion primitives generated between states, such as Dubins curves [44], [45]. Each primitive is a dynamically feasible trajectory with the associated cost and duration. Thus, the overall objective is given by:

$$J_k^\ell \triangleq \min_{\{\boldsymbol{\tau}_i, \mathbf{x}_i\}} \left\{ \max_{i \in \mathcal{N}_k} \left\{ \sum_{(s_{j_1}, s_{j_2}) \in \boldsymbol{\tau}_i} \min_{\kappa \in \boldsymbol{\kappa}(x_{j_1}, x_{j_2})} \mathsf{Cost}(\kappa) \right\} \right\}, \tag{20}$$

where $\boldsymbol{\kappa}(x_{j_1}, x_{j_2})$ denotes the predefined set of feasible motion primitives connecting the intermediate states x_{j_1} and x_{j_2}. In practice, a hybrid-A* like search method [46] can be adopted to jointly optimize the sequence of subtasks and motion primitives, producing the local plans $\{\boldsymbol{\tau}_i\}$ and trajectories $\{\mathbf{x}_i\}$ that minimize the makespan.

Example 3. As shown in Fig. 7, in total 14 delivery subtasks are assigned to 5 robots under different dynamic constraints. Note that the subtasks require different robot capabilities. The resulting assignments and trajectories are significantly different for holonomic and non-holonomic teams. ■

2) Static and Unknown Subtasks: In the static and unknown case, the subtasks $\mathcal{J}_k^\ell$ of ω_k^ℓ are initially unknown but become fixed once discovered. This setting is common in search-and-rescue, where the number and locations of victims or targets in a region are revealed only during execution. Since subtasks appear online, the static assignment model in the first case is not directly applicable. To address this, we propose a simultaneous exploration and coordination (SEC) method with two components: (I) collaborative exploration of the task area and (II) dynamic assignment of newly discovered subtasks during exploration. During exploration, the team must survey the task region S_k^ℓ to reveal subtasks. Two strategy classes are considered here: (I) *Obstacle-free spaces*: the goal is collaborative coverage, where standard coverage methods, e.g., polygon-based decomposition, apply while accounting for heterogeneous robot velocities, initial positions, and perception radii. (II) *Cluttered or structured workspaces*: robots must plan around obstacles; frontier-based exploration is commonly used [47], with trajectories respecting the robot dynamic constraints. In both cases, coordination avoids redundant search and ensures full coverage. The output is the exploration trajectories $\{\mathbf{x}_i\}$ for all $i \in \mathcal{N}_k$.

Furthermore, once the region begins to be explored and new subtasks are discovered, the task of assigning and fulfilling these subtasks must be done simultaneously with exploration. A minimum-disruption planning method is proposed, where newly discovered subtasks are inserted into the local plans of each robot. The objective is to minimize the increase in makespan due to these insertions. Specifically, once a subtask $j \in \mathcal{J}_k^\ell$ that meets the robot capabilities $a_j \in \mathcal{A}_i$ is discovered, the robot updates its local plan to include the subtask. The task sequence for each robot $i \in \mathcal{N}_k$ is then re-

Algorithm 2: Distributed Local Update for Dynamic and Known Subtasks

Input: Subteam $\mathcal{N}_k$, subtasks $\mathcal{J}_k^\ell$, cost $\chi(\cdot)$.
Output: Updated $\{\boldsymbol{\tau}_i(t)\}$ and $\{\mathbf{x}_i(t)\}$.

repeat at each planning iteration for each robot $i \in \mathcal{N}_k$;
  Collect neighbor memberships and $\{\chi(\mathcal{R}_j)\}$;
  Set $j' \leftarrow \mathbf{argmin}_j\, \chi(\mathcal{R}_j)$;
  if (23) holds **then**;
    Send switch intent to robot j';
  if no higher-priority conflicts;
    Set $j_i \leftarrow j'$ and synchronize;
until no robots can switch;
Update $\{\boldsymbol{\tau}_i(t)\}$ and $\{\mathbf{x}_i(t)\}$ given $\overline{\mathcal{R}}_t$;

optimized to minimize the makespan for the new tasks, i.e.,

$$\min_{\{(\boldsymbol{\tau}_i, \mathbf{x}_i)\}} \Big\{ \sum_{j=1}^{L_k^\ell} \big(\mathsf{Cost}(\boldsymbol{\tau}_i^+ - \mathsf{Cost}(\boldsymbol{\tau}_i))\big) \Big\}, \tag{21}$$

where $\boldsymbol{\tau}_i^+$ denotes the updated local plan after inserting the newly discovered subtasks, and the objective quantifies the resulting increase in execution cost and makespan. Subtasks are assigned online by selecting the robot that yields the smallest incremental cost in (21). The assignments are recomputed either periodically or after substantial progress on the current set. The output is the updated local sequences $\{\boldsymbol{\tau}_i\}$ and corresponding trajectories $\{\mathbf{x}_i\}$. Finally, non-holonomic constraints are handled as in the static and known case by planning over motion primitives to ensure feasible trajectories.

Example 4. As shown in Fig. 8, in total 5 robots are assigned to execute the task with 8 unknown targets. The initial trajectories of exploration minimizes the coverage time by the computed division points. More targets are discovered during exploration and assigned via the proposed scheme. The resulting trajectories not only explore the task region, but also fulfill all discovered subtasks. ■

3) Dynamic and Known Subtasks: For the third case, the set of subtasks J_k^ℓ within ω_k^ℓ and their initial locations s_j are known, but the subtasks are *dynamic* moving during execution. Their real-time positions are assumed to be available through external sensing or tracking systems [48]. Typical examples include collaborative pursuit or capture tasks, where the robots must form coalitions to surround and intercept moving targets. In such settings, static plans quickly become suboptimal or infeasible as the subtasks evolve over time.

A distributed *dynamic coalition formation* (DCF) method is adopted, in which each robot $i \in \mathcal{N}_k$ updates its coalition membership using local information and peer communication [14], [49]. At any time t, each subtask $j \in \mathcal{J}_k^\ell$ is associated with a coalition $\mathcal{R}_j(t) \subseteq \mathcal{N}_k$, and the collection of all coalitions forms a scheme $\overline{\mathcal{R}}_t \triangleq \{\mathcal{R}_j(t),\, j \in \mathcal{J}_k^\ell\}$. The coalitions are disjoint and collectively exhaustive, i.e., $\mathcal{R}_{j_1}(t) \cap \mathcal{R}_{j_2}(t) = \varnothing$ for $j_1 \neq j_2$, and $\bigcup_{j \in \mathcal{J}_k^\ell} \mathcal{R}_j(t) = \mathcal{N}_k$. The local plan $\boldsymbol{\tau}_i(t)$ is determined by its current coalition,

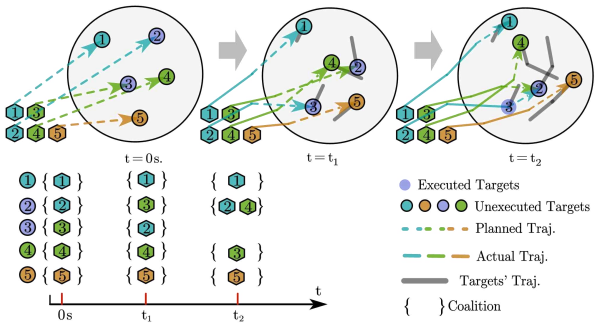


Fig. 9. Execution results of the dynamic and known subtasks in Sec. V-C3. **Top**: the trajectories of 5 robots (filled hexagons) and 5 targets (filled circles) at different snapshots; **Bottom**: the evolution of assigned coalitions for each target. Note that different colors reflect different capacity constraints.

i.e., $\boldsymbol{\tau}_i(t) = (t_i, p_i, a_i)$ for the currently assigned subtask, and its trajectory $\mathbf{x}_i(t)$ follows the coalition decision. To guide updates, each robot maintains local estimates of the coalition costs $\chi(\mathcal{R}_j)$, which depend on the robot–target distances and the velocities. The instantaneous team cost is defined as:

$$\chi(\overline{\mathcal{R}}_t) \triangleq \max_{j \in \mathcal{J}_k^\ell} \chi(\mathcal{R}_j) + \frac{1}{J_k^\ell} \sum_{j \in \mathcal{J}_k^\ell} \chi(\mathcal{R}_j), \tag{22}$$

where the first term reflects the current makespan across all subtasks, and the second term encourages a balanced workload. Based on (22), the local switch rule is stated from the perspective of each robot and is summarized in Alg. 2. At time t, robot i knows its assigned task j_i and the associated coalition $\mathcal{R}_{j_i}$. More importantly, it can estimate $\chi(\mathcal{R}_{j'})$ for nearby subtasks $j' \in \mathcal{J}_k^\ell$. Then, the robot i can evaluate the following condition locally:

$$\begin{aligned} &\max\Big\{\chi(\mathcal{R}_{j'} \bigcup \{i\}),\, \chi(\mathcal{R}_{j_i} \backslash \{i\})\Big\} \\ &< \max\big\{\chi(\mathcal{R}_{j_i}),\, \chi(\mathcal{R}_{j'})\big\}, \end{aligned} \tag{23}$$

where $\mathcal{R}_{j'}$ is the candidate coalition and $\mathcal{R}_{j_i}$ is the current coalition of robot i. If the inequality holds, an intent to switch is announced to neighbors in $\mathcal{R}_{j'}$ and $\mathcal{R}_{j_i}$, a priority rule resolves potential conflicts, and afterwards a synchronization finalizes the change. Robot i then updates its local plan $\boldsymbol{\tau}_i(t)$ and trajectory $\mathbf{x}_i(t)$ to track the new subtask. This local rule uses only the neighborhood costs and memberships, preserves disjoint coalitions, and drives the scheme toward a locally optimal $\overline{\mathcal{R}}_t^\star$, yielding time-varying plans $\{\boldsymbol{\tau}_i(t)\}$ and trajectories $\{\mathbf{x}_i(t)\}$ that continuously adapt to the moving subtasks.

Example 5. As shown in Fig. 9, 5 robots are assigned to 5 moving targets by forming dynamic coalitions online in Alg. 2. Initially, each robot is assigned to one target. However, robots 2 and 3 exchange their targets at t_1. When target 3 is finished at t_3, robots $2, 4$ form into a coalition to execute target 2 while robot 3 executes target 4. ■

Remark 7. The local coordination policy in Sec. V-C relies on inter-robot information exchange over the communication graph induced by the communication radius. Connectivity is the key structural requirement: if the graph is disconnected,

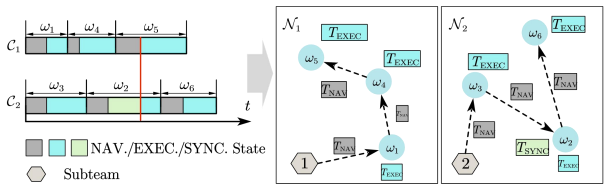


Fig. 10. Illustration of the change of execution status between navigation, execution and synchronization (**left**), for a team $\mathcal{C}_k$ given its local plan Γ_k (**right**) in (17) and (19). Note that the concurrency constraint between tasks ω_2, ω_5 are enforced by the synchronization state (red line).

coordination messages cannot propagate and effective collaboration cannot be realized. Moreover, latency, packet loss, and bandwidth limitations primarily slow information flow, increasing the planning time and causing transient inconsistencies in $\{\boldsymbol{\tau}_i\}$. However, when the graph is connected sufficiently often over time windows, delayed or missed updates are corrected once communication resumes. More realistic models of intermittent communication with joint planning of task and communication events remain our ongoing work. ■

Remark 8. Note that the case of dynamic and unknown tasks is not considered here, due to two reasons: (I) without knowing the total number and locations of subtasks, it is difficult to determine whether the current task is completed, e.g., subtasks may move dynamically from unexplored areas to explored areas; (II) if such a task exists, the coordination strategy would be a combination of the second and third cases above, i.e., to assign the subtasks of exploration and collaboration via dynamic coalition formation. ■

The theorem below presents the correctness guarantee for the local coordination policies proposed above. Detailed proof is attached in the Appendix.

Theorem 3. *For each task ω_k^ℓ and team $\mathcal{N}_k$, the strategies in Sec. V-C ensure that the plans $\{\boldsymbol{\tau}_i\}$ and trajectories $\{\mathbf{x}_i\}$ are feasible, and minimize the local task makespan.*

D. Online Execution, Interaction and Adaptation

1) Online Execution and Receding-horizon Adaptation: Given the initial workspace and mission descriptions, the missions are decomposed into tasks and the local teams $\overline{\mathcal{N}}$ are formed. A finite set of H tasks is then assigned with redundancy by Alg. 1, yielding the local plan Ξ_k for each team $\mathcal{N}_k \in \overline{\mathcal{N}}$. Then, each team executes $(S_k^\ell, \omega_k^\ell) \in \xi_k$ by navigating to S_k^ℓ and performing ω_k^ℓ. Specifically for robots in teams, synchronization is required before execution. Not only the robots within the team should collaborate to perform tasks, but also there are temporal constraints between tasks including precedence or concurrency. This yield three distinctive states during online execution: *navigation*, *execution* and *synchronization*. When execution begins, the subtasks $\mathcal{J}_k^\ell$ are coordinated by three local strategies to derive the action plans $\{\boldsymbol{\tau}_i\}$. Execution is concurrent across teams and parallel within each team. This process continues until all local plans $\{\Xi_k\}$ are completed, or a replanning is triggered.

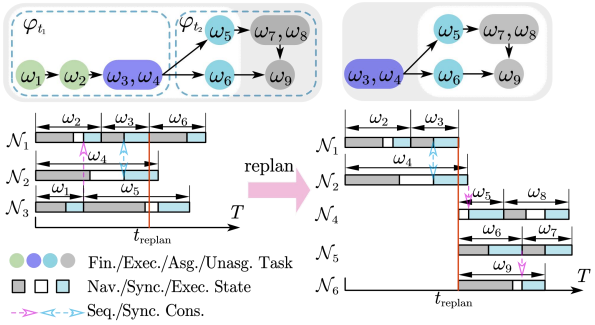


Fig. 11. Illustration of the receding-horizon scheme for coordination and online adaptation in Sec. V-D. Note that the number of teams is increased from 3 (**left**) to 5 after the task assignment and team formation (**right**), and the tasks ω_3, ω_4 are not preempted after replanning.

Example 6. As shown in Fig. 10, teams $\mathcal{N}_1$ and $\mathcal{N}_2$ are constructed from capacities $\mathcal{C}_1$ and $\mathcal{C}_2$ by team formation. After task ω_2, team $\mathcal{N}_2$ should wait for team $\mathcal{N}_1$ to perform task ω_5, due to the concurrency constraint for ω_2, ω_5. ■

Furthermore, a receding-horizon scheme handles replanning for online tasks and operator updates. As shown in Fig. 11, only H tasks are assigned per cycle, and the remaining tasks are considered in the next cycle. Replanning is triggered by: (I) execution progress, when more than half of the planned tasks are accomplished; (II) new mission specifications, when new missions or modifications arrive; and (III) feasibility, when any team reports infeasibility due to execution failures.

Upon any of the triggering conditions, the task assignment and team formation are recomputed by Alg. 1 using the current system state. The candidate pool includes tasks that are assigned but not yet started, unassigned tasks and newly specified tasks, which enables consistent revision of priorities and team compositions. However, tasks currently in execution are not preempted, and the associated teams continue execution without interruption. This non-preemption policy preserves the safety and continuity, which is crucial when the tasks outnumber the required teams. Moreover, the robot states are maintained by rolling forward the execution status: for team $\mathcal{N}_k$ with the current sequence $(\omega_k^0, \cdots, \omega_k^{L_k})$, where $\omega_k^{L_k}$ is ongoing, it holds that $\widehat{t}_i \triangleq t_\in(\omega_k^{L_k})$ and $\widehat{x}_i \triangleq S_k^{L_k}$ for each $i \in \mathcal{N}_k$. This enables consistent future formations under temporal ordering constraints.

Remark 9. Note that the choice of the key parameters including the rolling horizon H, the redundancy margin $\alpha \geqslant 1$, and the replanning trigger, such as the completion ratio of committed tasks or event-driven updates, typically depends on the expected mission volatility, e.g., the arrival and cancellation rate in Φ_t, the uncertainty in task and target estimation, e.g., location and service-time dispersion, and the anticipated robot failure and communication drop rate, which together balance look-ahead optimality, robustness, and responsiveness. Numerical analyses on how their values affect the overall performance are provided in Sec. VI. ■

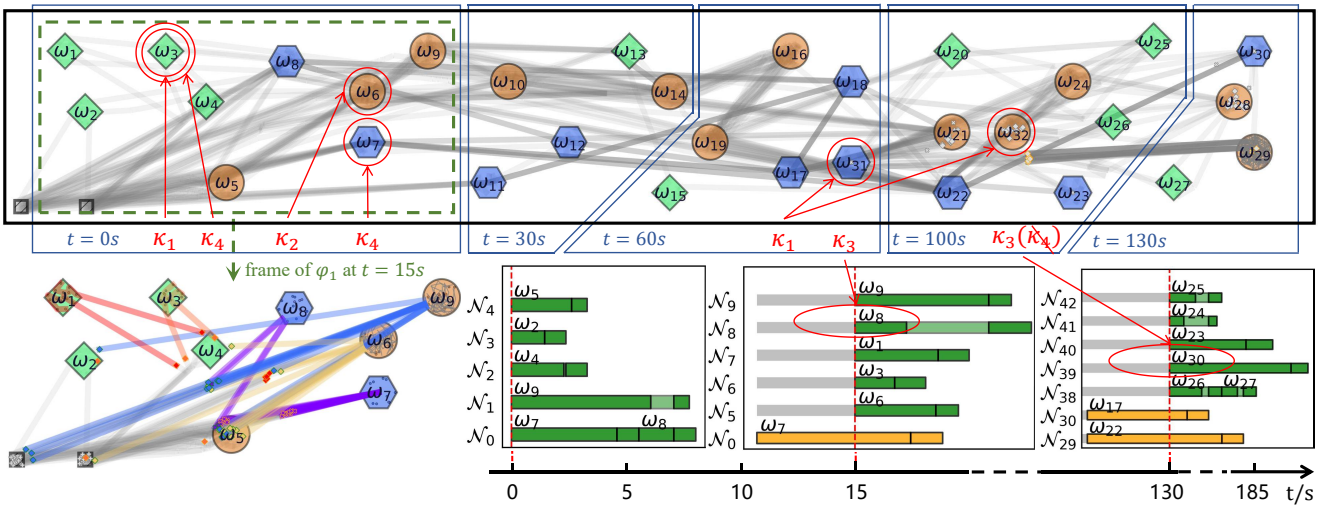


Fig. 12. Simulated human-in-the-loop scenario of 80 robots in dynamic environment. **Top**: in total 32 tasks and 482 subtasks are performed during the mission time of 180.4 s, with the time elapsed from left to right. Clearly κ_1 to κ_4 are issued in the process; **Bottom-left**: the snapshots of execution of tasks at $t = 15$ s, where different markers indicate different types of robots; **Bottom-right**: gantt charts of teams executing assigned tasks at $t = 0$ s, $t = 15$ s and $t = 130$ s. Note that tasks $\omega_1 - \omega_9$ are released at $t = 0$ s, $\omega_{10} - \omega_{14}$ at $t = 30$ s, $\omega_{15} - \omega_{19}$ at $t = 60$ s, $\omega_{20} - \omega_{26}$ at $t = 100$ s, $\omega_{27} - \omega_{30}$ at $t = 130$ s, and ω_{31}, ω_{32} are issued by operator at $t = 110$ s.

2) Fulfillment of Online Human Requests: Online operator requests are routed from the graphic interface including text, voice and templates, following the protocol for online requests in Sec. V-A. These requests are processed by the task decomposition module and the proposed planning stack. Each request induces a bounded update of the search tree $\mathfrak{T}$ and the frontiers $\overline{\mathcal{V}}$, followed by a re-optimization of teams K, capacities $\{\mathcal{C}_k\}$, and plans $\{\Gamma_k\}$ by Alg. 1. The updated plans are forwarded to the team execution as in Sec. V-C, under the non-preemption and receding-horizon scheme in Sec. V-D1, while the visualization interface immediately reflects changes in the task allocation, temporal relations among tasks and Gantt timelines, as summarized in Sec. V-A2.

More specifically, the four types of requests defined in Sec. IV-B are handled as follows. A new mission κ_1 is parsed and decomposed to sc-LTL, yielding $\mathcal{B}_{\varphi_m}$ and $\widehat{Q}_m^0$, with $\Phi_t \leftarrow \Phi_t \cup \{\varphi_m\}$; the planner continues the tree search by evaluating $\chi(\nu)$ in (6), updating $\overline{\mathcal{V}}$ via (15), and re-selecting the complete assignment by (16). A cancellation κ_2 removes φ_m from Φ_t, prunes nodes whose progress depends on $\widehat{Q}_m$, and recomputes $\overline{\mathcal{V}}$, while preserving tasks already in execution. A deadline or priority update κ_3 modifies the value function and costs by re-weighting the progress and adding deadline penalties within $\chi(\nu)$ and C_k of (6) and (7), in addition to bounding functions in (13), thus biasing the selection toward urgent missions. Lastly, a direct robot re-assignment κ_4 changes its fleet $\mathcal{N}$ and states $(\widehat{t}_i, \widehat{x}_i)$. Then, the capacity-based formation with bounds (18) is re-solved with the min–max objective $J(k)$ in (17), producing the revised $\{\mathcal{C}_k\}$ and $\{\Gamma_k\}$. All requests are issued through the same interface and acknowledged by synchronized feedback. Note that request κ_4 imposes linear admissibility constraints on $\{b_{ik}\}$ such as the locked and forbidden assignments, and optional limits on membership changes. These constraints are reflected in the receding-horizon assignment by restricting which capacity allocations are allowed within the tree search. Therefore, feasibility checks are performed across layers.

Remark 10. In practice, an operator may issue contradictory requests, e.g., enforcing immediate completion of a task via κ_3 while restricting the only capable robot via κ_4. In case of conflicting operator requests, if the set of current requests $\mathcal{K}_t$ renders the hard constraints inconsistent, e.g., capacity bounds (18), non-preemption in Sec. V-D1, or priority constraints in $\chi(\nu)$ and C_k in (7), then Alg. 1 and the formation objective $J(k)$ in (17) becomes infeasible. This infeasibility is detected at the module where it arises, thus an explicit warning is issued with the violated constraint class and implicated missions or robots, which prompts the operator to revise $\mathcal{K}_t$ before replanning proceeds. ■

3) Complexity and Scalability Analysis: The computational complexity of the proposed method is analyzed as follows. In each iteration of Alg. 1, selecting a batch of P nodes and expanding up to $B \triangleq (K+1)\,|\Omega^-_{\nu_h^\star}|$ children incurs $\mathcal{O}(PBK)$ for the automaton-state updates $\widehat{Q}_m^+$, together with the capacity and timing updates in (11) and (12); maintaining the non-dominated frontier based on (13)-(15) requires $\mathcal{O}(|\mathcal{V}|(2K+M))$ to reach the completeness condition in (16). The capacity-based team-formation MILP introduces $\mathcal{O}(NK^\star)$ binary variables b_{ik}, which is NP-hard in general but limited by the small number of teams. For the case of static and known tasks, the routing problem with subtour elimination is NP-hard, and its number of variables scales with $|\mathcal{N}_k|(J_k^\ell)^2$. For static and unknown tasks, polygonal coverage generation and assignment are near-linear in the region representation, while rolling insertion of newly discovered subtasks is $\mathcal{O}(|\mathcal{N}_k|)$ per discovery. Lastly, for dynamic and known tasks, the distributed coalition updates are linear in team size and the number of active targets per iteration. Although the local MILPs and coalition updates are NP-hard in the worst case, the hierarchical decomposition confines them to small

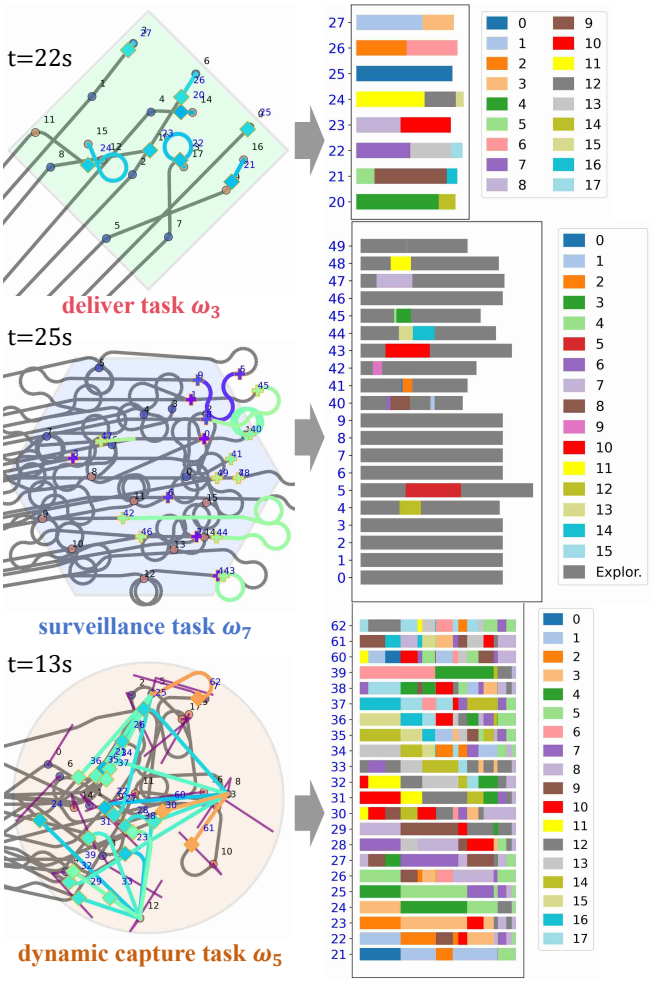


Fig. 13. Final trajectories of local task executions for robots under the dynamic constraints of maximum curvature $\kappa \leqslant 3$. **Top**: delivery task ω_3 at $t = 22\,\mathrm{s}$ for a team of 8 robots and 18 subtasks; **Middle**: surveillance task ω_7 at $t = 25\,\mathrm{s}$ for a team of 20 robots and 16 subtasks; **Bottom**: dynamic capture task ω_5 at $t = 13\,\mathrm{s}$ for a team of 22 robots and 18 subtasks.

teams, enabling event-based replanning with standard solvers. Detailed empirical runtime evaluations of each component and scaling benchmarks are provided in Sec. VI.

VI. Numerical Experiments

For numerical validation, the proposed method is implemented in `Python 3` and tested on a laptop with an Intel Core Ultra 9 285K CPU. The solver `GLOP` [43] is used for integer optimization. Simulation videos can be found in the supplementary files.

A. System Description

As shown in Fig. 12, the simulated fleet consists of $N = 80$ heterogeneous robots operating in an open environment of size $260\,\mathrm{m} \times 42\,\mathrm{m}$. There are three robot types with complementary capabilities: 20 Type-A robots that perform perception and delivery; 20 Type-B robots that perform perception and grasping; and 40 Type-C robots that perform delivery and grasping. Robots are initially distributed evenly over two bases. Unless otherwise specified, all robots follow a first-order dynamics model with a maximum speed of $2.5\,\mathrm{m/s}$ in simulation. To assess the effect of motion feasibility, both robots with curvature constraints and robots without curvature constraints are considered. The curvature limit is set to $3\,\mathrm{m}^{-1}$.

There are $|\boldsymbol{\varphi}_t| = 5$ missions released online at random time instants, with inter-arrival times drawn from a normal distribution with mean $\mu = 32.5\,\mathrm{s}$ and standard deviation $\sigma = 5\,\mathrm{s}$; samples are truncated to positive times. Upon release, mission locations are placed within the workspace according to a spatially uniform distribution. Each mission specified at time t_i follows the sc-LTL template below:

$$\varphi_i \;=\; \Diamond\big(\varphi^i_{\texttt{del}} \wedge \Diamond\, \varphi^i_{\texttt{surv}}\big) \;\wedge\; \big(\neg\, \varphi^i_{\texttt{cap}}\, \mathcal{U}\, \varphi^i_{\texttt{surv}}\big),$$

where the three task types are as follows. A delivery task requires two distinct subtasks to be completed through delivery or grasping actions. A surveillance task requires perception. A dynamic capture task requires two distinct subtasks to be completed through delivery or grasping actions while targets move. Delivery tasks contain on average 13 subtasks. Surveillance tasks contain on average 15 subtasks, and each subtask is initially unknown with probability 0.5. Capture tasks include approximately 17 moving targets with speed $0.5\,\mathrm{m/s}$ within the designated region. The global planning horizon is $H = 6$, and replanning is triggered. Replanning occurs upon sufficient execution progress, upon the release or modification of missions, and upon detected infeasibility. Note that operator commands are issued in real-time throughout the simulation, via the proposed protocol.

B. Results

1) Mission Decomposition and Subteam Formation: As shown in Fig. 12, the first mission contains 9 tasks. At release, tasks ω_1 through ω_4 are delivery, ω_7 and ω_8 are surveillance, and ω_5, ω_6, and ω_9 are capture. The associated NBA is computed in $0.20\,\mathrm{s}$ with 7 states and 18 transitions. Given these task automata, Algorithm 1 optimizes the number of subteams and predicts a makespan of $37.5\,\mathrm{s}$ when $K = 5$. The computation takes $1.36\,\mathrm{s}$. During this process, human intervention request κ_1 for new mission release is integrated in real time, where a new task ω_3 is added to the set. This process confirms the system adaptability to the online inputs of the operator. This layer validates the top-down design: the global planner reasons on precedence to size coalitions before any motion planning. Then the subteams are instantiated with capabilities matched to the next admissible tasks. The resulting formations are: $\mathcal{N}_0$ with 16 Type-B robots for surveillance tasks ω_7 and ω_8, $\mathcal{N}_1$ with 2 Type-B and 14 Type-C robots for capture task ω_9, $\mathcal{N}_2$ with 5 Type-C robots for delivery task ω_4, $\mathcal{N}_3$ with 5 Type-C robots for delivery task ω_2, and $\mathcal{N}_4$ with 2 Type-B and 14 Type-C robots for capture task ω_5. This composition reflects the capability coupling in the specification: surveillance requests perception, capture requests grasping or delivery in addition to coordination, and delivery requests transport actions.

Execution begins with $\mathcal{N}_3$ on ω_2 and $\mathcal{N}_4$ on ω_5. Teams $\mathcal{N}_0$, $\mathcal{N}_1$, and $\mathcal{N}_2$ move to staging locations because the temporal ordering requires ω_2 before ω_4, ω_5 before ω_7, and ω_4 before ω_8. The ordering also allows ω_8 and ω_9 to proceed concurrently. These temporal constraints promote purposeful staging

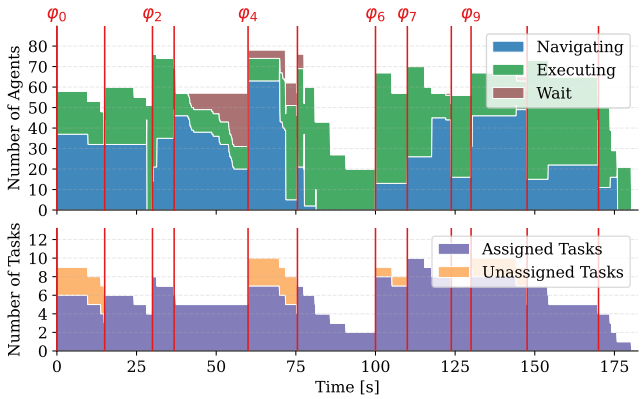


Fig. 14. Evolution of enrolled robots and tasks accomplished, as new missions are specified by $\varphi_0 - \varphi_{10}$. **Top**: the number of enrolled robots with different status, including navigation, waiting, and task execution; **Bottom**: number of remaining tasks that are assigned and unassigned.

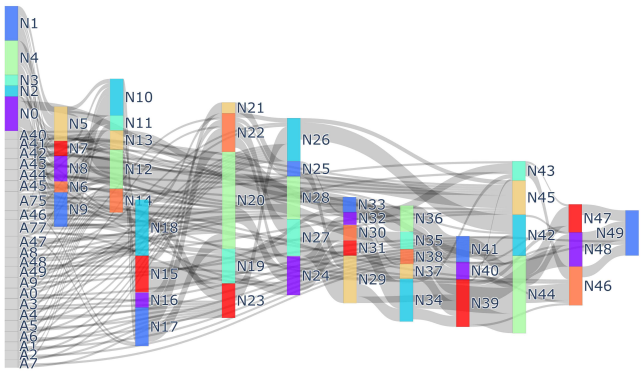


Fig. 15. Sankey diagram of all robots that participated in the overall mission, along with the number of teams and their compositions. Note that subteams are labeled $N_0 - N_{49}$ and robots are labeled $A_0 - A_{79}$.

and prevent premature, infeasible starts. At $t = 15\,\mathrm{s}$, teams $\mathcal{N}_3$, $\mathcal{N}_4$, and $\mathcal{N}_2$ complete ω_2, ω_5, and ω_4, respectively, for a total of 47 finished subtasks. Completion triggers replanning, which costs $0.41\,\mathrm{s}$. At this stage, request κ_3 *for priority change* is issued by the operator to execute ω_8 due to a *deadline adjustment*. Since $\mathcal{N}_0$ is executing ω_7 at that time, the tasks ω_8 and ω_9 are revisited along with the remaining frontier tasks. Team $\mathcal{N}_0$ continues ω_7 without interruption, and new teams $\mathcal{N}_5$ through $\mathcal{N}_9$ are formed to execute ω_6, ω_3, ω_1, ω_8, and ω_9, respectively. This confirms that the online loop preserves continuity for in-progress work while exploiting newly freed resources. At $t = 24\,\mathrm{s}$, the operator issues request κ_2 *for task cancellation*, terminating task ω_6 to reflect the updated priorities. Thus, the respective robots in $\mathcal{N}_5$ are freed up resources to other tasks in the next replanning.

At $t = 30\,\mathrm{s}$, a new mission with 5 tasks and 7 relations is released. The global layer replans in $0.44\,\mathrm{s}$, produces 5 subteams, and updates the predicted makespan to $51.3\,\mathrm{s}$. Note that at $t = 110\,\mathrm{s}$, requests κ_1 are released to add task ω_{31} and ω_{32}. Other missions are released at $t = 60\,\mathrm{s}, 100\,\mathrm{s}, 130\,\mathrm{s}$ increase the overall workload to 32 tasks and 482 subtasks. Specially, conflicting requests are issued by operator at $t = 140\,\mathrm{s}$, where κ_3 requests to raise the priority of task ω_{30} execute, and κ_4 requests to assign 10 Type-C robots to execute ω_{27}. Namely, the available Type-C robots can not meet the capacity requirements of task ω_{30}. Thus, the system shows the conflict to operator and asks which command to execute. The operator chooses κ_3, raising the priority of task ω_{30}. In total, the full mission set finishes at $180.4\,\mathrm{s}$ over 12 replanning events. The mean task response time is $37.0\,\mathrm{s}$, and the mean and max replanning time for 12 events are $0.36\,\mathrm{s}$, $0.87\,\mathrm{s}$. The requests continue to be processed, ensuring that dynamic adjustments are smoothly integrated into the execution timeline.

2) Local task execution: Local execution follows the three strategy classes in Sec. V-C and respects the curvature feasibility shown in Fig. 13. All robot trajectories have curvature below $3\,\mathrm{m}^{-1}$. At $t = 22\,\mathrm{s}$, the delivery task ω_3 includes 18 static and known subtasks. Team $\mathcal{N}_6$ is assigned 8 robots using (20); each robot completes 2 or 3 subtasks under a fixed robot-to-subtask assignment, yielding short traversal and limited coalition changes. During this phase, the operator issues request κ_4 *for robot reassignment*, transferring robot 21 from ω_5 to support team $\mathcal{N}_6$ in completing subtasks $5, 9, 16$ of ω_3 under resource reallocation. This intention enables rapid cross-task resource redistribution under changing priorities.

At $t = 25\,\mathrm{s}$, the surveillance task ω_7 requires exploration with unknown subtasks. Team $\mathcal{N}_0$ follows region-covering trajectories, and newly revealed subtasks are allocated using (21) to minimize route disruption. For example, robot 43 completes subtask 10 and robot 4 completes subtask 12 before resuming exploration. At this point, the operator issues request κ_4 *for robot reassignment*, assigning robot 42 to assist team $\mathcal{N}_0$ in completing subtask 9 of ω_7. This timely intervention reallocates available capacity to the newly revealed workload, improving execution efficiency. In total, 16 subtasks are discovered and completed, validating that exploration and execution should interleave to reduce idle time.

At $t = 13\,\mathrm{s}$, the capture task ω_5 comprises 18 moving subtasks and is executed by team $\mathcal{N}_4$ with 22 robots. Alg. 2 updates local coalitions online as targets move; coalitions dissolve upon subtask completion and robots are reassigned. For instance, robots 60 and 21 complete subtask 0 jointly; robot 60 then completes subtask 10 alone, while robot 21 completes subtask 1 alone. Near completion, robots consolidate into two coalitions to finish subtasks 5 and 8, indicating that additional robots mitigate motion variability and accelerate completion. Across task families, mean response times are $29.39\,\mathrm{s}$ for delivery, $46.37\,\mathrm{s}$ for surveillance, and $35.65\,\mathrm{s}$ for dynamic capture, with local planning times of $0.31\,\mathrm{s}$, $0.22\,\mathrm{s}$, and $0.50\,\mathrm{s}$.

3) Overall Resources Utilization and Adaptation: As can be seen in Fig. 14, the status of task execution takes the largest proportion of robot states for whole progress. The proportion of navigation states is acceptable since the whole scene is in shape of rectangular. Waiting states are kept low due to the scheduling strategies. The unassigned tasks in the lower panel spike to 5 after each release and fall to 0 within $15\,\mathrm{s}$, indicating fast adaptation to new tasks. Moreover, the proposed scheme continually reshapes the composition of each team as the mission evolves, as illustrated in Fig. 15. In total 49 subteams are formed and robots are reused across teams such as robots in N_{43} later are separated out to form new teams N_{46} and N_{47} together with other robots.

TABLE II
COMPARISON WITH BASELINES (N=80, M=30, J= 450)

Method	Resp. Time [s]	Ave. Resp. [s]	Ave/Max Plan [s]	N/W/E Robots	Succ. Rate [%]
Ours	184.4	**34.6**	0.46/0.76	**19/3/27**	**100**
MILP	327.8	99.1	90.3/144.0	23/7/18	100
SAMP-Task	286.0	72.6	0.45/0.74	24/7/15	100
SAMP-Subtask	230.1	45.6	3.8/8.0	16/5/30	86
ScRATCHeS	257.4	64.9	1.25/7.60	13/3/19	100
Hulk(+poset)	194.4	40.5	4.66/7.59	24/8/24	100
Flow	183.5	41.8	1.79/5.98	31/1/34	96
Inf-H	**182.1**	77.1	53.4/415.8	23/3/24	100
Greedy	325.7	115.6	**0.31/0.56**	31/7/30	100

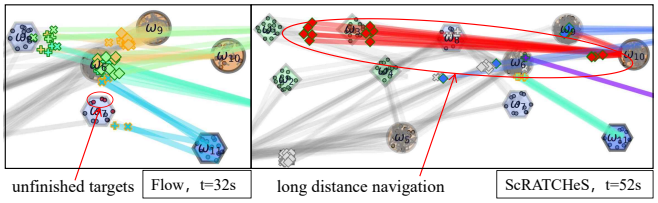


Fig. 16. Snapshots of baselines Flow and ScRATCHeS. **Left**: the subtasks in the red circle are unfinished after t = 32s; **Right**: robots marked in red have long navigation distance due to inefficient assignments.

TABLE III
SCALABILITY AND ROBUSTNESS ANALYSIS

(N, M, J)	Failure ρ	Resp. Time [s]	Ave/Max Plan [s]	N/W/E Robots	Succ. Rate [%]
(120, 50, 750)	0.05	378.2	0.46/0.82	26/5/27	100
	0.10	346.6	0.51/0.79	29/5/25	100
(150, 80, 1202)	0.05	699.1	0.53/0.88	29/3/31	100
	0.10	788.3	0.55/0.90	26/7/36	100
(170, 100, 1509)	0.05	923.1	0.65/1.48	30/6/30	100
	0.10	1120	0.87/1.65	31/3/29	100

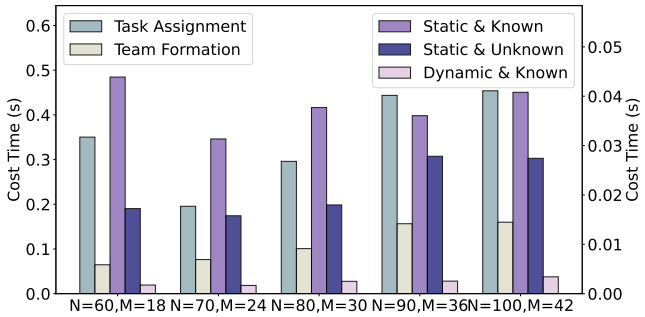


Fig. 17. Computational time analysis across the **five core modules** with varying numbers of robots N and tasks M.

C. Comparisons

The proposed method is compared against **eight** baselines: (I) **MILP**, where a complete MILP is formulated for all robots $\mathcal{N}$ and tasks $\overline{\Omega}_t$, similar to [13], [25], i.e., without the subteam formation; (II) **SAMP-Task**, where a sampling-based planner from [5] is adopted for all robots and tasks; (III) **SAMP-Subtask**, which applies the sampling-based planner directly to subtasks; (IV) **ScRATCHeS**, which formulates a complete MILP for all robots $\mathcal{N}$ and tasks $\overline{\Omega}_t$, with capacity-based temporal logic formulation [21]; (V) **Hulk**, which follows poset abstraction of mission automata and task-graph constrained receding-horizon assignment [35]; (VI) **Flow**, which models precedence-aware coalition task allocation and solves it via network-flow approximations with online re-allocation [34]; (VII) **Inf-H**, which is the same as our method but with an infinite horizon H, i.e., all known tasks are assigned in Alg. 1; (VIII) **Greedy**, which assigns a maximum of one task to each subteam, i.e., without the horizon H. The first six baselines are established methods, while the last two are ablation studies. Note that the replanning conditions for all baselines are identical to the proposed method. The compared metrics include the maximum response time for missions, average response time of missions, the average and maximum planning time, the average number of robots performing navigation, waiting for collaboration, and executing tasks, and the success rate.

As summarized in Table II, the proposed method outperforms all other methods across most metrics, including response time, planning time, and robot scheduling. **Efficiency and Response:** HECTOR achieves a total response time of 184.4 s, which is significantly lower than most methods like MILP at 327.8 s and SAMP-Task at 286.0 s, largely owing to the establishment for subteam. While the Inf-H method exhibits a comparable total response time, its maximum planning time can be prohibitively long at 415.8 s, demonstrating the inefficiency of an infinite horizon. The Flow based method has a execution success rate of 96% though also exhibiting a comparable response time. Meanwhile, HECTOR maintains a highly stable and efficient planning, with the average and max planning time of 0.46/0.76 s. While some methods exhibit comparable computational efficiency in terms of planning time with SAMP-Task at 0.45/0.74 s and Greedy at 0.31/0.56 s, they suffer from inefficient scheduling of robots or prolonged response times. **Reliability and Success Rate:** HECTOR and MILP both achieve a 100% success rate, but HECTOR does so with significantly less computational overhead. As shown in Fig. 16, while ScRATCHeS exhibits excellent robot scheduling (13/3/19), it falls short in terms of response and planning time. In contrast, SAMP-Subtask and Flow exhibit lower success rates at 86% and 96%, as they do not account for uncertainties in subtasks. Although Greedy remains fast, it results in inefficient robot scheduling (31/7/30) and a long response time at 325.7s. **Robustness in Complex Coordination:** While SAMP-Subtask struggles with the increased search space of 450 subtasks, our hierarchical structure effectively decouples the task assignment from intra-team coordination. Though Hulk shows a competitive response time at 194.4 s, HECTOR avoids the mission poset establishment for complex temporal tasks, leading to a 10 times reduction in planning time.

D. Generalization

For further validation, the scalability and robustness of our method are evaluated by increasing the fleet size and introducing robot failures with a probability ρ. The scenarios are scaled from a fleet of 120 robots with 50 tasks (released within 240 s), to 150 robots and 80 tasks (released within 400 s), to 170 robots and 100 tasks (released within 480 s).

(I) *Scalability*. As summarized in Table III, regardless of the failure rate ρ being 0.05 or 0.1, the average and maximum planning time remain consistently below 2.0 s, even as the

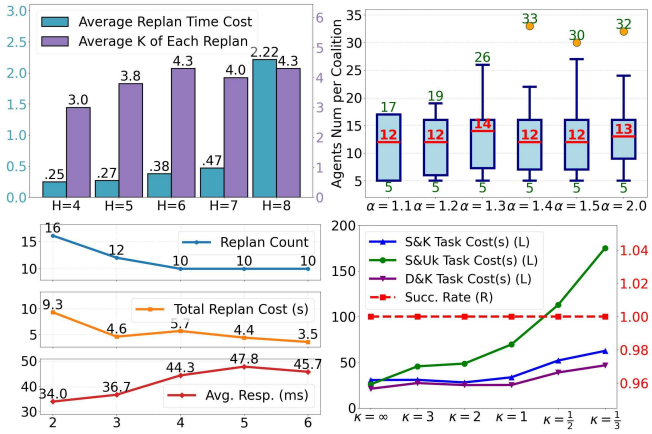


Fig. 18. Parameter sensitivity and performance analysis of the HECTOR framework: parameter horizon H in assignment (**top-left**), redundancy margin α in team formation (**top-right**), the number of finished task in current horizon to trigger the replan (**bottom-left**), and the curvature κ of robots to plan the trajectories in task coordination (**bottom-right**).

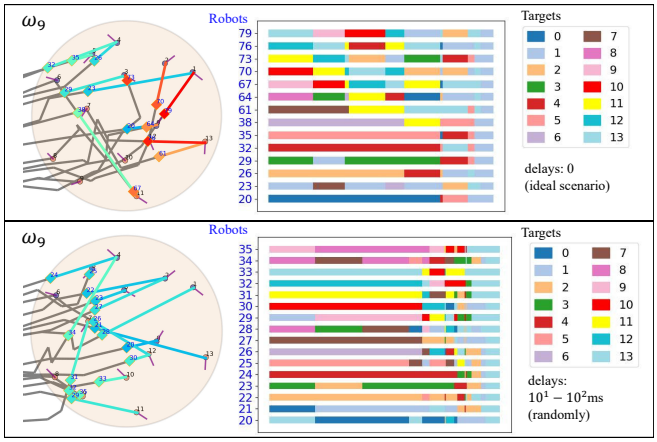


Fig. 19. Illustration of the distributed dynamic coalition formation under different communication delays. **Top**: the ideal scenario without communication delay; **Bottom**: scenario with stochastic delays ranging from 10^1 to 10^2 ms.

problem scale grows to 170 robots and 100 tasks. This efficiency is primarily attributed to our receding-horizon planning with $H = 6$, which prevents computational explosion with increasing fleet size. When the fleet size and the number of tasks increase to 120 and 50, the response time increases by 40% to 57% relative to the total horizon of 240 seconds. Similarly, with a fleet size of 150 and 80 tasks, the response time increases by 75% to 97% relative to the total horizon of 400 seconds, while the response time increases by 92% to 133% relative to the horizon of 480 seconds, clearly indicating that the response time scales with the fleet and task sizes, and demonstrating the long-range scheduling of robots in large-scale scenarios. Furthermore, the average number of deployed robots increases as the fleet size and task complexity grow.

(II) *Failure Recovery*. Even with failure probabilities of $\rho = 0.05$ and 0.1, the success rate remains at 100% for fleets of 170 robots with 100 tasks and 1509 subtasks. To recover from these failures, more robots are recruited and the average response time is further increased. The average planing time is only increased slightly with higher failure rate. This shows that the fleet capacities are sufficient to meet the task requirements, even under more challenging conditions.

(III) *Runtime Decomposition*. Fig. 17 reports the runtime of each main module in the proposed framework, including task assignment, team formation, and local coordination, under different fleet sizes $|\mathcal{N}|$ and the numbers of tasks $|\mathcal{T}_{\text{act}}|$. It can be seen that task assignment consistently requires more computational time than team formation, with times ranging from approximately 0.35s to a maximum of 0.55s as both N and M increase. In contrast, team formation remains relatively stable, typically between 0.05s and a maximum of 0.1s. The dynamic and known scenario exhibits the lowest computational times for local coordination strategies, while the static and known case takes longest time at around 0.4s.

(IV) *Sensitivity to Key Parameters*. As shown in Fig.18, the sensitivity of the proposed scheme with respect to several key parameters are analyzed. In particular, the analysis of the rolling horizon H reveals that the average replanning time increases from 0.25s at $H = 4$ to 2.22s at $H = 8$. The average number of teams during each replan is around 3.9, which remains close across different horizons. Moreover, the replan count decreases to 10 as the triggering condition is increased from 2 to 6. The total replanning time decreases from 9.3s to 3.5s, while the response time is increased from 34.0ms and 47.8ms. Regarding the redundancy margin α, the agents per coalition increase with α, reaching 32 agents at $\alpha = 2.0$, compared to 17 agents at $\alpha = 1.1$. However, the success rate remains 100% across all choice of α. Lastly, as the curvature constraints κ are tightened, the execution costs of local tasks are increased due to tighter dynamic constraints, especially for static and unknown tasks from 25s to 180s. The success rate remains 100% across all curvatures and tasks, demonstrating its versatility, even for non-holonomic teams.

(V) *Communication Constraints*. The performance of the distributed dynamic coalition formation under varying communication latencies is evaluated, as shown in Fig. 19. It can be seen that when the communication latency is increased to a stochastic delay ranging from 10^1 to 10^2 ms, the proposed algorithm can still ensuring successful task completion. Notably, the resulting robot formation, local plans, and trajectories are drastically different. For instance, robots 21 and 20 are assigned to the subtask 0, while subtask 7 is finished by the coalition formed by robots 34 and 25 instead of robots 23 and 61. The overall task execution time is increased slightly from 31.0s to 41.2s.

E. High-fidelity ROS-Simulation

1) Mission and Workspace Setup: To validate the proposed framework under more realistic robot dynamics and environment interactions, a ROS-based simulation is conducted for 12 UAVs performing 10 tasks as part of the disaster relief mission, as shown in Fig. 20. The tasks are released in three stages across 3 different missions. Two UAVs are assigned to the static tasks, including the searching for the workshop and storage tank. Four UAVs are allocated to the dynamic tasks, such as transporting relief packages to moving workers. The static tasks include delivering goods and seeding messages to fixed delivery locations. The request κ_1 *for new task release* is issued to add an additional task, ω_9, for

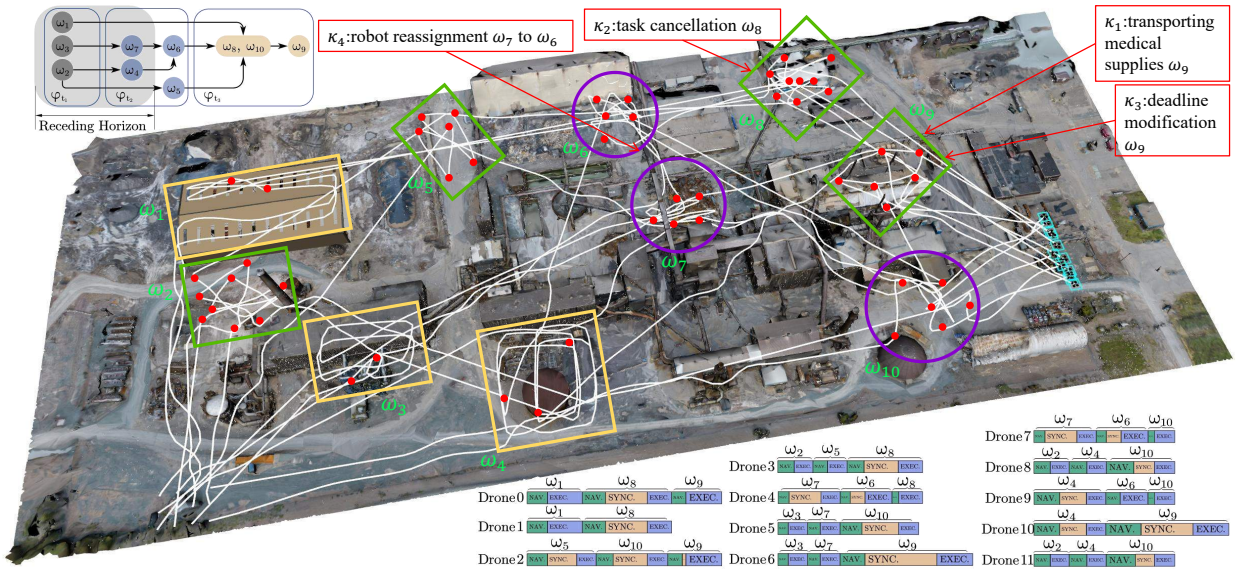


Fig. 20. High-fidelity ROS simulation of the proposed framework under the scene for disaster relief. **Top-left**: 2 UAVs assigned to static known tasks (green), 4 robots to dynamic known tasks (purple), and 6 robots to static unknown tasks (yellow); **Middle**: the task execution trajectories for 10 tasks over different regions and 36 subtasks. **Bottom-right**: the execution timeline of each robot completing assigned tasks in 34.6 s on average.

transporting medical supplies to a new location, prompting a reallocation of resources. When workers make empirical requests to the operator at unexpected locations, the request κ_4 *for robot reassignment* is utilized to reassign UAV 7 from ω_7 to assist team $\mathcal{N}_3$ with ω_6. These interventions ensure the system remains flexible and responsive to dynamic missions.

2) Simulation Results and System Performance: As summarized in Fig. 20, the simulation results highlight the effectiveness of the hierarchical coordination framework, with all 10 tasks and 50 subtasks successfully completed. The system efficiently handles dynamic task reassignments, with 4 replanning events performed in less than 1 s each. The average response time for tasks is 34.6 s, and the mean replanning time for 10 events is 0.46 s. For static tasks, the average response time is 28.9 s, while dynamic capture tasks have a response time of 29 s. The multimodal human-fleet interaction protocol allows real-time operator inputs through the GUI, enabling dynamic task modifications. Leveraging external situational awareness, the operator identifies a critical conflict where a routine supply delivery obstructed urgent rescue operations, request κ_2 *for task cancellation* is triggered to cancel ω_8 due to a priority shift, freeing resources for higher-priority tasks. Additionally, the request κ_3 *for deadline modification* is used to extend deadlines for ω_9 as delays occur due to unforeseen human-induced circumstances. This flexibility ensures seamless task allocation and continuous adaptation to changing task conditions, maintaining smooth execution throughout the mission.

VII. Conclusion

This work presents HECTOR, a hierarchical planning and coordination framework that couples global mission assignment with local subtask and trajectory coordination. It embeds practical human–fleet protocols and a graphic interface for multimodal and online interactions under complex temporal tasks. Future work includes conflicting requests, adversarial settings via game-theoretic reasoning, and tighter integration of communication constraints.

References

[1] J. Ferreira and P. Lima, "A survey of human–multi-robot interaction," *Robotics and Autonomous Systems*, vol. 144, p. 103837, 2021.

[2] M. Križmančić, B. Arbanas, T. Petrović, F. Petrić, and S. Bogdan, "Cooperative aerial–ground multi-robot system for automated construction tasks," *Automation in Construction*, vol. 141, p. 104469, 2022.

[3] A. Varava, K. Hang, D. Kragic, and F. T. Pokorny, "Herding by caging: a topological approach towards guiding moving agents via mobile robots." in *Robotics: Science and Systems*, 2017, pp. 696–700.

[4] E. Tuci *et al.*, "Cooperative object transport in multi-robot systems," *Frontiers in Robotics and AI*, vol. 5, p. 59, 2018.

[5] Y. Kantaros and M. M. Zavlanos, "Stylus*: A temporal logic optimal control synthesis algorithm for large-scale multi-robot systems," *The International Journal of Robotics Research*, vol. 39, no. 7, pp. 812–836, 2020.

[6] S. Choudhury, S. Arora, A. Kapoor, and D. Dey, "Dynamics-aware multi-robot task allocation with deadlines," *Robotics: Science and Systems (RSS)*, 2017.

[7] S. L. Smith and F. Bullo, "Dynamic task allocation in multi-robot systems," *Proceedings of the IEEE*, vol. 94, no. 7, pp. 1272–1286, 2009.

[8] R. Hoque, L. Y. Chen, S. Sharma, K. Dharmarajan, B. Thananjeyan, P. Abbeel, and K. Goldberg, "Fleet-dagger: Interactive robot fleet learning with scalable human supervision," in *Conference on Robot Learning (CoRL)*, 2022.

[9] A. Dahiya *et al.*, "A survey of multi-agent human–robot interaction systems," *Robotics and Autonomous Systems*, vol. 161, p. 104335, 2023.

[10] G. Swamy, S. Reddy, S. Levine, and A. Dragan, "Multi-robot allocation of assistance from a shared uncertain human supervisor," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 6409–6415.

[11] Y. Chen, Y. Sun, and B. Englot, "Decentralized multi-robot task allocation with uncertain task rewards," *Autonomous Robots*, vol. 45, no. 5, pp. 659–680, 2021.

[12] M. Gini, "Multi-robot allocation of tasks with temporal and ordering constraints," in *AAAI Conference on Artificial Intelligence*, 2017.

[13] A. Torreño, E. Onaindia, A. Komenda, and M. Štolba, "Cooperative multi-agent planning: A survey," *ACM Computing Surveys (CSUR)*, vol. 50, no. 6, pp. 1–32, 2017.

[14] W. Dai, A. Bidwai, and G. Sartoretti, "Dynamic coalition formation and routing for multirobot task allocation via reinforcement learning," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 16 567–16 573.

[15] Y. Ji, X. Li, and X. Yang, "Multi-robot task allocation with time and resource constraints," *Robotics and Autonomous Systems*, vol. 136, p. 103711, 2021.
[16] S. Zhao, Y. Shen, and Y. Ding, "Market-based task allocation for multi-robot systems with uncertain task rewards," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3053–3060, 2021.
[17] F. Fioretto, E. Pontelli, and W. Yeoh, "Distributed constraint optimization problems and applications: A survey," *Journal of Artificial Intelligence Research*, vol. 61, pp. 623–698, 2018.
[18] H. Liu, M. Li, and X. Li, "Multi-robot task allocation with evolutionary algorithms: A comparative study," *Applied Soft Computing*, vol. 81, p. 105511, 2019.
[19] J. G. Martin, J. R. D. Frejo, R. A. García, and E. F. Camacho, "Multi-robot task allocation problem with multiple nonlinear criteria using branch and bound and genetic algorithms," *Intelligent Service Robotics*, vol. 14, no. 3, pp. 707–727, 2021.
[20] X. Luo, S. Xu, R. Liu, and C. Liu, "Decomposition-based hierarchical task allocation and planning for multi-robots under hierarchical temporal logic specifications," *IEEE Robotics and Automation Letters*, vol. 9, no. 8, pp. 7182–7189, 2024.
[21] K. Leahy, Z. Serlin, C.-I. Vasile, A. Schoer, A. M. Jones, R. Tron, and C. Belta, "Scalable and robust algorithms for task-based coordination from high-level specifications (scratches)," *IEEE Transactions on Robotics*, vol. 38, no. 4, pp. 2516–2535, 2021.
[22] Z. Chen and Z. Kan, "Real-time reactive task allocation and planning of large heterogeneous multi-robot systems with temporal logic specifications," *The International Journal of Robotics Research*, vol. 44, no. 4, pp. 640–664, 2025.
[23] M. Lahijanian, S. B. Andersson, and C. Belta, "Temporal logic motion planning and control with probabilistic satisfaction guarantees," *IEEE Transactions on Robotics*, vol. 31, no. 3, pp. 546–561, 2015.
[24] P. Schillinger, M. Bürger, and D. V. Dimarogonas, "Simultaneous task allocation and planning for temporal logic goals in heterogeneous multi-robot systems," *The International Journal of Robotics Research*, vol. 37, no. 7, pp. 818–838, 2018.
[25] X. Luo and M. M. Zavlanos, "Temporal logic task allocation in heterogeneous multirobot systems," *IEEE Transactions on Robotics*, vol. 38, no. 6, pp. 3602–3621, 2022.
[26] Y. E. Sahin, P. Nilsson, and N. Ozay, "Multirobot coordination with counting temporal logics," *IEEE Transactions on Robotics*, vol. 36, no. 4, pp. 1189–1206, 2019.
[27] L. Li, Z. Chen, H. Wang, and Z. Kan, "Task allocation of heterogeneous robots under temporal logic specifications with inter-task constraints and variable capabilities," *IEEE Transactions on Automation Science and Engineering*, 2025.
[28] G. A. Cardona and C.-I. Vasile, "Planning for heterogeneous teams of robots with temporal logic, capability, and resource constraints," *The International Journal of Robotics Research*, vol. 43, no. 13, pp. 2089–2111, 2024.
[29] V. Kurtz and H. Lin, "A more scalable mixed-integer encoding for metric temporal logic," *IEEE Control Systems Letters*, vol. 6, pp. 1718–1723, 2021.
[30] L. Lindemann, J. Nowak, L. Schönbächler, M. Guo, J. Tumova, and D. V. Dimarogonas, "Coupled multi-robot systems under linear temporal logic and signal temporal logic tasks," *IEEE Transactions on Control Systems Technology*, vol. 29, no. 2, pp. 858–865, 2019.
[31] Z. Liu, M. Guo, and Z. Li, "Time minimization and online synchronization for multi-agent systems under collaborative temporal logic tasks," *Automatica*, vol. 159, p. 111377, 2024.
[32] Y. Kantaros, S. Kalluraya, Q. Jin, and G. J. Pappas, "Perception-based temporal logic planning in uncertain semantic maps," *IEEE Transactions on Robotics*, vol. 38, no. 4, pp. 2536–2556, 2022.
[33] X. Luo and C. Liu, "Simultaneous task allocation and planning for multi-robots under hierarchical temporal logic specifications," *IEEE Transactions on Robotics*, 2025.
[34] W. Gosrich, S. Agarwal, K. Garg, S. Mayya, M. Malencia, M. Yim, and V. Kumar, "Online multi-robot coordination and cooperation with task precedence relationships," *IEEE Transactions on robotics*, 2025.
[35] Q. Luo, J. Li, and M. Guo, "Hulk: Large-scale hierarchical coordination under continual and uncertain temporal tasks," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2025.
[36] A. Kolling, P. Walker, N. Chakraborty, K. Sycara, and M. Lewis, "Human interaction with robot swarms: A survey," *IEEE Transactions on Human-Machine Systems*, vol. 46, no. 1, pp. 9–26, 2016.
[37] M. Gombolay, R. Wilcox, and J. A. Shah, "Computational design of mixed-initiative human–robot teaming," *Journal of Artificial Intelligence Research*, vol. 57, pp. 389–452, 2017.

TABLE IV
NOMENCLATURE OF KEY VARIABLES AND DEFINITIONS

Term	Definition	Reference
Φ_t	Set of missions known at time $t > 0$.	Sec. V-B
τ_i	Local plan of robot $i \in \mathcal{N}$.	Sec. IV-A
$\mathcal{K} = \{\kappa_{1,2,3,4}\}$	Operator requests.	Sec. IV-B
$\mathcal{B}_{\varphi_m}$	Büchi automaton for mission φ_m.	Sec. V-B
$\widehat{Q}_m$	Set of reachable states in automaton $\mathcal{B}_{\varphi_m}$.	Sec. V-B
$\mathfrak{T}$	Search tree structure defined as $(\mathcal{V}, \rightarrow)$.	Sec. V-B
$\chi(\nu)$	Value function for node selection.	Eq. (6)
$\mathcal{C}_k$	Capacity constraints for team k.	Sec. V-B
β_k^j	Min. robots required to perform action a^j for team k.	Eq. (7)
α_j	Redundancy margin for workload uncertainty and failures.	Sec. V-B
$\zeta(\nu)$	Performance profile for node evaluation and pruning.	Eq. (13)
$\overline{\mathcal{V}}$	Set of non-dominated frontier nodes in the search tree.	Eq. (15)
H	Planning horizon for task assignment.	Sec. V-B
b_{ik}	Robot i assigned to team $\mathcal{N}_k$.	Sec. V-B
$J(k)$	Execution cost of team k.	Sec. V-B
N_k	Robots in team k.	Sec. V-B
$\mathcal{J}_k^\ell$	Set of local tasks for the ℓ-th task of team k.	Sec. V-C
$\mathbf{x}_i$	Trajectory of robot $i \in \mathcal{N}$.	Sec. V-C

[38] J. Tumova, A. Marzinotto, D. V. Dimarogonas, and D. Kragic, "Maximally satisfying ltl action planning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2014, pp. 1503–1510.
[39] A. Ulusoy, S. L. Smith, X. C. Ding, C. Belta, and D. Rus, "Optimality and robustness in multi-robot path planning with temporal logic constraints," *The International Journal of Robotics Research*, vol. 32, no. 8, pp. 889–911, 2013.
[40] C. Baier and J.-P. Katoen, *Principles of model checking*. MIT press, 2008.
[41] K. R. Apt and A. Witzel, "A generic approach to coalition formation," *International game theory review*, vol. 11, no. 03, pp. 347–367, 2009.
[42] Y. You *et al.*, "Human-multi-robot interaction: A survey," *Foundations and Trends in Robotics*, vol. 9, no. 2, pp. 59–170, 2021.
[43] G. L. O. Solver, https://developers.google.com/optimization/lp.
[44] J. Ny, E. Feron, and E. Frazzoli, "On the dubins traveling salesman problem," *IEEE Transactions on Automatic Control*, vol. 57, no. 1, pp. 265–270, 2011.
[45] P. Váňa and J. Faigl, "On the dubins traveling salesman problem with neighborhoods," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 4029–4034.
[46] S. Erke, D. Bin, N. Yiming, Z. Qi, X. Liang, and Z. Dawei, "An improved a-star based path planning algorithm for autonomous land vehicles," *International Journal of Advanced Robotic Systems*, vol. 17, no. 5, 2020.
[47] D. Holz, N. Basilico, F. Amigoni, and S. Behnke, "Evaluating the efficiency of frontier-based exploration strategies," in *International Symposium on Robotics*. VDE, 2010, pp. 1–8.
[48] L. Zhou, V. Tzoumas, G. J. Pappas, and P. Tokekar, "Resilient active target tracking with multiple robots," *IEEE Robotics and Automation Letters*, vol. 4, no. 1, pp. 129–136, 2018.
[49] J. Guerrero and G. Oliver, "Multi-robot coalition formation in real-time scenarios," *Robotics and Autonomous Systems*, vol. 60, no. 10, pp. 1295–1307, 2012.

APPENDIX

A. Proof of Lemmas and Theorems

Proof. **of Theorem 1**. Temporal correctness follows from the update of the reachable state sets $\widehat{Q}_m$ along enabled automaton

transitions $q_m^{\ell+1} \in \delta^m(q_m^\ell, \omega^{\ell+1})$ and from the completeness condition $\widehat{Q}_m \cap Q_F^m \neq \varnothing$ in (16). Any root to leaf path that ends at a node $\nu \in \overline{\mathcal{V}}^\star$ therefore induces, for every $m \in \mathcal{M}$, an accepting run of $\mathcal{B}_{\varphi_m}$, so the joint plans $\{\Gamma_k\}$ satisfy all temporal constraints encoded by $\{\mathcal{B}_{\varphi_m}\}$. Moreover, the capacity feasibility and finite time convergence follow from local pruning and finiteness of the search space. Each expansion step updates $\mathcal{C}_k$ and enforces the fleet constraint $\sum_{k\in\mathcal{K}} \beta_k^j \leqslant \sum_{i\in\mathcal{N}} \mathbb{1}(a^j \in \mathcal{A}_i)$, hence any node in $\overline{\mathcal{V}}^\star$ respects all capacity bounds. The set Φ_t, the automaton state sets Q^m, the alphabets Σ^m, the total number of tasks Ω, and the admissible team compositions are finite, so the set of feasible nodes is finite. Dominance pruning does not remove all representatives of any feasible solution class, and a fair selection rule eventually expands every feasible non-dominated node. For a feasible problem, at least one complete non-dominated node $\nu \in \overline{\mathcal{V}}^\star$ is therefore generated in finite time, which yields plans that satisfy all missions and all capacities. Lastly, given the objective function (6) and enough planning time, the node with the minimum cost as in (5) would be returned, which also satisfies each mission requirement at the accepting states. This completes the proof. □

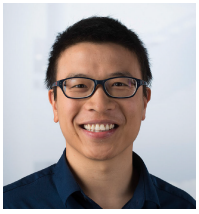
Proof. **of Lemma 2**. Similar to the previous case of known tasks, since each node ν explicitly tracks reachable states $\widehat{Q}_m$ and transitions δ^m, and expansion strictly enforces the capacity bounds in (8), the partial plans are inherently consistent with the mission specification and capability constraints. Moreover, the receding-horizon handover during re-planning uses the final state of H as the next root ν_0, ensuring the execution sequence remains a valid prefix toward the accepting sets $\{Q_F^m\}$. Thus, the accumulated trace of all robots satisfies the mission requirement φ_m once an accepting state $q_k \in Q_F^m$ is reached. However, it is worth noting that global optimality is lost because the limited horizon H and online mission updates prevent the tree search from considering the complete set of all future tasks. □

Proof. **of Theorem 3**. In the static and known case, the task reduces to an MVRP, where the trajectory of each robot $i \in \mathcal{N}_k$ is optimized to minimize the makespan J_k^ℓ. For non-holonomic robots, the problem is augmented with motion primitives $\kappa(x_{j_1}, x_{j_2})$ to ensure feasible trajectories, with the overall objective to minimize J_k^ℓ under trajectory constraints. In the static and unknown case, exploration ensures all subtasks $\mathcal{J}_k^\ell$ are discovered, and the makespan $\mathbf{max}_{i\in\mathcal{N}_k} T_i$ is minimized as new subtasks are dynamically inserted. In the dynamic and known case, coalition updates are driven by the cost function $\chi(\mathcal{R}_j)$ for each coalition $\mathcal{R}_j$, and the switch condition ensures that robots only switch coalitions if it reduces the overall team cost $\chi(\overline{\mathcal{R}}_t)$. The process converges to a locally optimal configuration, minimizing the makespan $\mathbf{max}_{k\in\mathcal{K}} J(k)$. □

Shen Wang received the B.E. degree in robotics engineering from the College of Engineering, Peking University, Beijing, China, in 2025, where he is currently working toward the M.E. degree in mechanical engineering with the School of Advanced Manufacturing and Robotics. His current research interests include multi-robot systems, task planning, and reinforcement learning.

Yinhang Luo received the B.S. degree in statistics from Shandong University, China, in 2021, and the M.S. degree in fluid mechanics from Peking University, China, in 2025. He is currently a Research Assistant at the School of Advanced Manufacturing and Robotics, Peking University. His research interests include multi-robot systems and reinforcement learning.

Jie Li received the B.E. degree in automation and the M.E. and Ph.D. degrees in pattern recognition and intelligent systems from the National University of Defense Technology, China, in 2006, 2008, and 2014, respectively. From 2014 to 2019, he was a Lecturer with the Unmanned Aerial Systems Laboratory. Since 2020, he has been an Assistant Professor with the College of Intelligence Science and Technology, National University of Defense Technology. His research interests include swarm intelligence, emergent behavior, multi-agent cooperation, distributed control, and collective decision-making.



Meng Guo received the M.Sc. degree in system, control, and robotics and the Ph.D. degree in electrical engineering from the KTH Royal Institute of Technology, Sweden, in 2011 and 2016, respectively. He was a Post-Doctoral Associate with the Department of Mechanical Engineering and Materials Science, Duke University, Durham, NC, USA. From 2018 to 2021, he was a Senior Research Scientist in reinforcement learning and planning at Bosch Center for Artificial Intelligence (BCAI), Germany. Since 2022, he has been an Assistant Professor with the School of Advanced Manufacturing and Robotics, Peking University, China. His main research interests include task and motion planning for robotic systems.