# HULK: Large-scale Hierarchical Coordination under Continual and Uncertain Temporal Tasks

Qingyuan Luo[1], Jie Li[2] and Meng Guo[1]

***Abstract*— Multi-agent systems can be extremely efficient when working concurrently and collaboratively, e.g., for delivery, surveillance, search and rescue. Coordination of such teams often involves two aspects: (i) selecting appropriate subteams for different tasks in various areas; (ii) coordinating agents in the subteams to execute the associated subtasks. Existing work often assumes that the tasks are static and known beforehand, where an integer program can be formulated and solved offline. However, in many applications, the team-wise tasks are generated online continually by external requests; and the amount of subtasks within each task is uncertain (e.g., the number of packages to deliver, and victims to rescue). The aforementioned offline solution becomes inadequate as it would require constant re-computation for the whole team and global communication to broadcast the results. Thus, this work tackles the large-scale coordination problem under continual and uncertain temporal tasks, specified as temporal logic formulas over collaborative actions. The proposed hierarchical framework (HULK) consists of two interleaved layers: the rolling assignment of currently-known tasks to subteams within a certain horizon, and the dynamic coordination within a sub-team given the detected subtasks during online execution. Thus, the coordination is performed hierarchically at different granularities and triggering conditions, to improve the computational efficiency and robustness. It is validated rigorously over large-scale heterogeneous systems under various temporal tasks and environment uncertainties.**

## I. Introduction

Fleets of heterogeneous robots, such as ground vehicles and aerial vehicles, are deployed to accomplish tasks that are otherwise too inefficient or even infeasible for a single robot [1]. Not only the overall efficiency of the team can be significantly improved by allowing the robots to move and act concurrently [2], [3]; but also the capabilities of the team can be greatly extended by enabling multiple robots to directly collaborate on a task [4], [5]. However, the optimal coordination of a large-scale multi-agent system to accomplish the desired task is well-known to be hard, especially to fulfill the spatially distributed subtasks in the right order at the right time. The set of possible task assignments are often combinatorial with respect to the number of robots and the length of tasks [1], [2], [6]. Commonly such a team-wise task is specified beforehand and remains unchanged, of which the solutions are derived offline and thus static. A particularly challenging scenario is when the system operates indefinitely, i.e., new tasks are released or canceled *dynamically* and *continually* by external demand, thus requiring the agents to change their task plans frequently. The aforementioned methods become inadequate as the sequence of tasks is infinite and their specifications are unknown beforehand. Recursive application of the static methods in a naive way leads to not only intractable computation complexity, but also inconsistent or even oscillatory assignments.

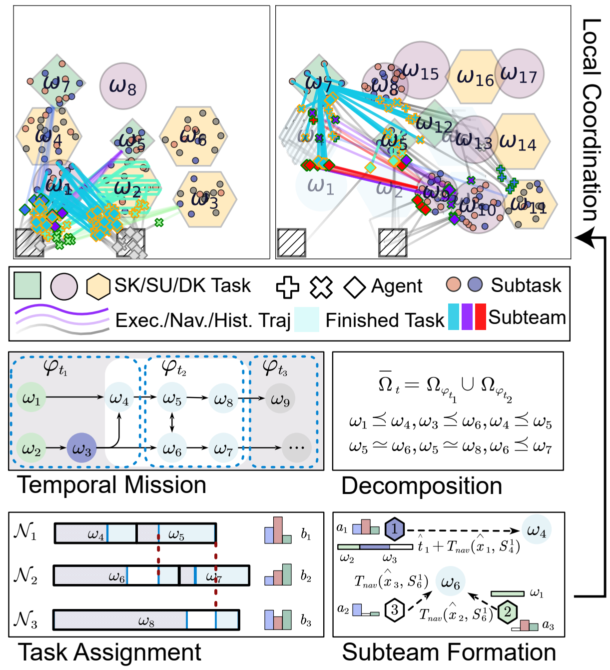


Fig. 1. Overall framework, including: snapshots of online execution of **three** types of tasks, which contains numerous subtasks (**top**); the posets associated with missions that are released online (**middle**); and the receding-horizon assignment of tasks to optimized formation of teams (**bottom**).

### A. Related Work

Task planning refers to the process of first decomposing this task into sub-tasks and then assigning them to the team, see [7], [8], [9] for comprehensive surveys. Different optimization criteria can be chosen, such as MinSUM [8]; and MinMAX [10]. The tasks can be specified in various forms, such as in the multi-vehicle routing problem [9], the job-shop problem [11]; and the coalition formation problem [12]. Existing methods can be categorized into centralized methods such as mixed integer linear programming (MILP) [7] and search-based methods [13]; and decentralized methods such as market-based methods [14] and distributed constraint optimization (DCOP) [15]. However, since many task planning problems are in general NP-hard

The authors are with [1]the Department of Mechanics and Engineering Science, College of Engineering, 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 62203017, T2121002, U2241214; 2030-Key Project under Grant 2020AAA0108200; and by the Fundamental Research Funds for the central universities. Contact: meng.guo@pku.edu.cn.

or even NP-complete [8], meta-heuristic approaches are used to gain computational efficiency, e.g., local search [16] and genetic algorithms [9]. However, most of the aforementioned methods can not be applied directly in this work due to the general task specification as temporal logic formulas over collaborative actions. Additionally, the aforementioned methods often focus on solving a static problem, rather than the continual and dynamic scenario addressed in this work.

Temporal logic formulas can be used to specify complex robotic tasks, such as Probabilistic Computation Tree Logic (PCTL) in [17], Linear Temporal Logics (LTL) in [6], [18], [19], [20], [21], [22], and counting LTL (cLTL) in [23]. Considerable results are developed in the recent years regarding the task assignment problem of team-wise tasks specified as temporal logic formulas. Analogously, they can be categorized into centralized methods and decentralized methods. Centralized methods often put emphases on optimality and completeness, such as the sampling-based search algorithm [6], the simultaneous decomposition and assignment method [18], the MILP formulation [19], [23], [20]. Decentralized methods are more applicable to large-scale multi-agent systems, including the local coordination of local tasks [21], [22], [24], the local assignment under partial workspace [25], the distributed sampling method [26], and the online auction algorithm [27]. However, the scenario where new tasks are released online with uncertain subtasks is rarely addressed in the aforementioned work, as it would require online adaptation for both the global task assignment and the local subtask execution.

### B. Our Method

To tackle these issues, this work put strong emphases on the online coordination algorithm that are essential when the collaborative tasks are released continually, and the amount of subtasks within each task is uncertain. In other words, both the distribution and requirements of the tasks can only be known during online execution. As shown in Fig. 1, the proposed method is based on the hierarchical coordination framework (HULK) that combines the global task assignment and the local subtask coordination. Namely, the global mission specification is first decomposed into collaborative tasks with the associated temporal constraints. Then, a receding-horizon assignment algorithm is applied to assign these collaborative tasks to subteams of the agents, subject to resources requirements, navigation cost and ordering constraints. Afterwards, each subteam follows different local strategies, where subtasks are detected and assigned dynamically during execution. Online adaptation at both levels is triggered by external events and execution status. Efficiency and robustness of the proposed framework is validated rigorously over large-scale heterogeneous systems and three different temporal tasks.

Main contribution of this work is threefold: (I) The proposed hierarchical coordination algorithm is applicable to a wide range of formulations, such as heterogeneous agents, temporal tasks and collaborative actions; (II) It is robust to varying distribution of continual tasks, contingent agent failures and uncertain subtasks; and (III) It is computationally efficient and scalable to large-scale systems.

## II. Problem Description

### A. Multi-agent Systems

Consider a team of $N$ agents that share the common workspace $\mathcal{W} \subset \mathbb{R}^3$. Each agent $i \in \mathcal{N} \triangleq \{1, \cdots, N\}$ is described by its position $x_i \in \mathcal{W}_i$ and its action $a_i \in \mathcal{A}_i$, where $\mathcal{W}_i \subseteq \mathcal{W}$ is the allowed workspace; and $\mathcal{A}_i$ is the set of primitive actions such as surveillance, delivery, capture and defense. Each agent can navigate freely in the workspace via a reference velocity $v_i \in \mathbb{R}^3$. Thus, the local plan of an agent is given by a sequence of timed goal positions and performed actions, i.e., $\tau_i \triangleq (t_i^1, g_i^1, a_i^1)(t_i^2, g_i^2, a_i^2)\cdots$ with $t_i^\ell \geqslant 0$, $g_i^\ell \in \mathcal{W}_i$, $a_i^\ell \in \mathcal{A}_i$ being the time instant, goal position and action, $\forall \ell \geqslant 1$. In other words, under this local plan, agent $i \in \mathcal{N}$ should navigate to $g_i^\ell$ with velocity $v_i$ and start performing $a_i^\ell$ from time $t_i^\ell$, for all $\ell \geqslant 1$.

### B. Mission Specifications

At time $t \geqslant 0$, a temporal mission is released (e.g., by a human operator or triggered by an event) to the fleet, denoted by $\varphi_t \triangleq \text{sc-LTL}\big(\boldsymbol{\omega}_t\big)$, where: (I) $\boldsymbol{\omega}_t \triangleq \{\omega_1, \cdots, \omega_{M_t}\}$ is the set of collaborative tasks within the mission. Each collaborative task $\omega_m \in \boldsymbol{\omega}_t$ is denoted by:

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

where $S_m \subset \mathcal{W}$ is the area that the task should be accomplished; $(n_j, a_j, s_j)$ is a subtask that requires *at least* $n_j$ agents performing action $a_j$ collaboratively at location $s_j \in S_m$; $J_m > 0$ is the total number of subtasks within the task $\omega_m$; and the estimated duration of each subtask is given by function $\eta_m : \mathbb{N} \times \mathcal{A} \times 2^{\mathcal{N}} \to \mathbb{R}^+$, i.e., $\eta_m(n_j, a_j, \mathcal{N}_j)$ returns the duration if the subteam of agents $\mathcal{N}_j \subset \mathcal{N}$ is assigned to provide the required action $a_j$ at location $s_j$. Thus, task $\omega_m$ is accomplished after each subtask is completed; (II) The tasks $\omega_m$ are nested following the syntax of Linear Temporal Logic (LTL) [28], e.g., via $\varphi \triangleq \top \mid p \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid \bigcirc\varphi \mid \varphi_1 \mathsf{U} \varphi_2$, where $\top \triangleq \texttt{True}$, $p \in AP$, $\bigcirc$ (*next*), $\mathsf{U}$ (*until*) and $\bot \triangleq \neg\top$. or other derived operators like $\square$ (*always*), $\Diamond$ (*eventually*), $\Rightarrow$ (*implication*). The full semantics and syntax of syntactic co-safe LTL (sc-LTL) are omitted here for brevity, see e.g., [28].

Consequently, the temporal mission $\varphi_t$ is satisfied if all collaborative tasks are accomplished, while the resulting trace of their temporal ordering satisfies the sc-LTL formula $\varphi_t$ via the satisfaction relation $\models$ from [28]. Lastly, the accumulated missions up to time $t \geqslant 0$ are denoted by $\boldsymbol{\varphi}_t \triangleq \{\varphi_{t_\ell}, \forall t_\ell \leqslant t\}$. It is worth mentioning that the tasks $\{\omega_m\}$ in (3) can be *uncertain*, i.e., the exact number of subtasks $J_m$ and their locations $\{s_j\}$ are unknown at the time of release. This might be due to the partial observability or dynamic nature of the environment.

**Remark 1.** The definition of collaborative task in (3) differs from the notion of cLTL [23] in two aspects: (I) Both the location and number of subtasks are uncertain; (II) The duration of all subtasks can vary depending on the assigned agents, i.e., instead of being instantaneous [29]. ■

**Remark 2.** The duration function $\eta_m(\cdot)$ typically *saturates* as the number of participants increases, i.e., the marginal benefits diminishes as also adopted in [30], [31]. ■

### C. Problem Statement

Given the above model, the overall objective is to synthesize the collective plans $\{\tau_i\}$ such that the average response of each mission is minimized, i.e.,

$$\mathbf{min}_{\{\tau_i\}} \frac{\sum_{\varphi_{t_\ell} \in \boldsymbol{\varphi}_t} (t^{\text{f}}_\ell - t_\ell)}{|\boldsymbol{\varphi}_t|}, \tag{2}$$

where $0 \leqslant t_\ell \leqslant t^{\text{f}}_\ell$ are the time instants when the mission $\varphi_{t_\ell} \in \boldsymbol{\varphi}_t$ is released and accomplished, respectively.

## III. Proposed Solution

The proposed solution consists of two main components: (I) The receding-horizon task planning algorithm that assigns tasks to subteams of agents given the global mission specification and constraints on the resources; (II) The local coordination algorithm that assigns subtasks to agents during online execution. The synergy and adaptation of both components are triggered by external events and execution status.

### A. Receding-horizon Task Assignment

*1) Decomposition of Temporal Tasks:* Given the mission specification $\varphi$, the NBA associated with $\varphi$ is denoted by $\mathcal{B} = (Q,\, Q_0,\, \Sigma,\, \delta,\, Q_F)$, of which the notation follows [28]. Based on $\mathcal{B}$, the associated tasks and their partial temporal constraints can be computed based on our earlier work [29], [32], as posets over tasks, i.e.,

$$\Omega_\varphi \triangleq \big\{(\Omega,\, \preceq,\, \simeq)\big\}, \tag{3}$$

where $\Omega \triangleq \{\omega_1, \cdots, \omega_M\} \subset \Sigma$ is a set of tasks; the partial ordering constraints $\preceq,\, \simeq \subset \Omega \times \Omega$ such that: (I) $\omega_{m_1} \preceq \omega_{m_2}$ if task $\omega_{m_1}$ should be accomplished before task $\omega_{m_2}$ is started; (II) $\omega_{m_1} \simeq \omega_{m_2}$ if tasks $\omega_{m_1}$ and $\omega_{m_2}$ should start at the same time. Simply speaking, the posets are abstracted from the mission automaton as the set of possible ways to satisfy the mission, as the set of tasks involved and their temporal relations. This allows more parallel execution during assignment thus improving efficiency. Detailed algorithms and completeness analyses can be found in [29].

As the missions can be specified online, the set of missions by time $t > 0$ is given by $\boldsymbol{\varphi}_t$, each of which can be decomposed into the partially-ordered tasks. Consequently, the set of all *known and unfinished* tasks by time $t$ is given by: $\overline{\Omega}_t \triangleq \bigcup_{\varphi_i \in \boldsymbol{\varphi}_t} \Omega_{\varphi_i}$, where $\Omega_{\varphi_i}$ is the partially-ordered tasks associated with mission $\varphi_i \in \boldsymbol{\varphi}_t$ as defined in (3). Note that the tasks from different missions are assumed to be independent. Moreover, given $\overline{\Omega}_t$, a directed acyclic *task graph* $\mathcal{G}_t$ can be constructed for these tasks based on their partial ordering, i.e., each node in the graph represents a task in $\overline{\Omega}_t$, and the edges represent the precedence and concurrence constraints.

**Algorithm 1:** Task Assign. and Subteam Format.

**Input:** Robots $\mathcal{N}$, tasks $\boldsymbol{\varphi}_t$, horizon $H$.
**Output:** Assignment $\{\nu^\star_{K^\star}\}$, subteams $\{\mathcal{N}_k\}$, and local plans $\{\xi_i\}$.

```
/* Task Assignment */
Compute posets Ω̄_t for φ_t by (3);
Build task graph G_t given Ω̄_t;
Initialize V* = ∅;
for K ∈ H do
    Compute optimal assignment ν*_K via (7);
    Add ν*_K to V*;
Choose best ν*_{K*} among V* by (7);
/* Subteam Formation */
Compute cost matrix {t_ik} by (8);
Formulate and solve MILP problem to find {b_ik};
Compute local plans {ξ_i} by (9);
```

*2) Capacity-based Task Assignment for Subteams:* Given the temporal constraints on simultaneous execution and the objective to minimize response time, the agents need to be divided into subteams for parallel execution. However, due to limitations in the number and capacity of the agents, it is not feasible to create a subteam for each individual task, rather a subteam is assigned a sequence of tasks based on agent capacity and the requirements of the tasks.

Denote by $\nu \triangleq \{\mathcal{C}_k,\, k = 1, \cdots, K\}$ the set of partial assignments along with the capacity constraints for each subteam, where each local assignment is given by:

$$\mathcal{C}_k \triangleq \Big( (\omega^0_k, \cdots, \omega^{L_k}_k),\, \big\{(m^j_k,\, a^j_k),\, a^j_k \in \mathcal{A}\big\} \Big), \tag{4}$$

of which the first part is the sequence of tasks assigned to the $k$-th subteam, $L_k$ is the total number of tasks in the sequence, while the second is the minimum number of agents $m^j_k$ to perform action $a^j_k$. Note that the number of subteams $K$ is not pre-defined and to be optimized.

Consider that $H > 0$ tasks should be assigned within the task graph $\mathcal{G}_t$. A search-based algorithm is proposed to determine the optimal subteam assignment $\nu^\star$. As summarized in Alg. 1, starting from the root node as the empty assignment $\nu_0$, the selected node is expanded by adding the next feasible task to any of the subteam. In particular, the assignment for the set of tasks that are currently being executed remains unchanged, while the rest of the tasks within $\mathcal{G}_t$ can be added if their preceding tasks are fulfilled or being executed. More importantly, after adding $\omega^{L_k}_k$ to the subteam $k$, the capacity constraint is updated as follows:

$$m^j_k \triangleq \mathbf{max}_{a^j_k \in \omega^\ell_k \in \mathcal{C}_k} \{n_j\},\ \forall a^j_k \in \mathcal{A}; \tag{5}$$

i.e., the maximum number of agents $m^j_k$ required for each action $a^j_k$ across the sequence of task $\omega^\ell_k$. The search is terminated when the number of assigned tasks for the fleet reaches the horizon $H$, or the capacity constraints are violated, i.e.,

$$\sum_{k \in \mathcal{K}} m^j_k \leqslant \sum_{i \in \mathcal{N}} \mathbb{1}(a^j_k \in \mathcal{A}_i),\ \forall a^j_k \in \mathcal{A}; \tag{6}$$

where the left-side is the resources required by the assignment, and the right-side is the complete capacity of the fleet. Lastly, the optimal assignment is selected from the complete tree by evaluating the overall quality of each node, i.e.,

$$\xi(\nu) \triangleq \eta_{L_k}(\nu, \overline{\Omega}_t) + \max_{k \in \mathcal{K}} \{t_{\rm e}(\omega_k^{L_k})\};$$
$$t_{\rm e}(\omega_k^\ell) \triangleq \Big(\max_{\omega_j \in \texttt{Pre}(\omega_k^\ell)} \{t_{\rm e}(\omega_j)\}\Big) + T_{\texttt{nav}}(S_k^{\ell-1}, S_k^\ell), \tag{7}$$

where $t_{\rm e}(\omega_k^\ell)$ is the estimated ending time of $\omega_k^\ell \in \mathcal{C}_k$; $\texttt{Pre}(\omega_k^\ell)$ is set of preceding tasks in the task graph $\mathcal{G}_t$; $T_{\texttt{nav}}(S_k^{\ell-1}, S_k^\ell)$ is the estimated navigation time from the previous task region $S_k^{\ell-1}$ to the current task region $S_k^\ell$; and $\eta(\nu, \Omega_t)$ is the estimated progress achieved by the assignment $\nu$ w.r.t. the unfinished tasks. Denote by $\nu_K^\star$ the optimal assignment for $K$ subteams. The same search procedure is repeated for different choices of $K \in \mathcal{H} \triangleq \{1, \cdots, H\}$, for which the set of optimal assignments is given by $\mathcal{V}^\star \triangleq \{\nu_K^\star, K \in \mathcal{H}\}$. Within this set, the same measure as in (7) is adopted to select the best choice of $K$ as $K^\star$ and the associated assignment $\nu_{K^\star}^\star$. It is worth noting that the subteams $\{\mathcal{C}_k\} \in \nu_{K^\star}^\star$ only specify the constraints on capacity, rather than specific agents.

**Remark 3.** Note that various heuristics can be applied to prune the search space, e.g., elimination of symmetric nodes; removing branches if its root node is worse than the current best node. More numeric details are given in the Sec. IV. ■

*3) Redundancy-aware Subteam Formation:* Given the optimal task assignment for subteams $\nu_{K^\star}^\star$, the actual formation of each subteam should be determined, i.e., to find the set of subteams $\overline{\mathcal{N}} \triangleq \{\mathcal{N}_1, \cdots, \mathcal{N}_{K^\star}\}$, where $\mathcal{N}_{k_1} \bigcap \mathcal{N}_{k_2} = \varnothing$, $\forall k_1 \neq k_2$ and $\mathcal{N}_{k_1}, \mathcal{N}_{k_2} \subset \mathcal{N}$. In other words, the robots in $\mathcal{N}_k$ are assigned to the subteam $\mathcal{C}_k$ within $\nu_{K^\star}^\star$.

This can be done in two steps: (I) The lower and upper bound for each task is determined by the capacity constraints; (II) A constrained min-max assignment problem is formulated for the $N$ robots and the first task of $K$ subteams, i.e., the estimated starting time if robot $i \in \mathcal{N}$ participates in the first task of subteam $\mathcal{C}_k$ is given by:

$$t_{ik} \triangleq \widehat{t}_i + T_{\texttt{nav}}(\widehat{x}_i, S_k^1), \tag{8}$$

where $\widehat{t}_i$ and $\widehat{x}_i$ are the expected time and position when robot $i$ becomes available after executing its current task (if any); $T_{\texttt{nav}}$ is the estimated duration to navigate to the task region $S_k^1$. It can be formulated as a mixed-integer linear programming (MILP) problem over the possible robot-task pairings with boolean variables $\{b_{ik}\}$, which can be solved efficiently by existing solvers such as `GLOP` [33]. Thus, the local *task plan* of each robot $i \in \mathcal{N}_k$ is given by:

$$\xi_i \triangleq (S_k^1, \omega_k^1)(S_k^2, \omega_k^2) \cdots (S_k^{L_k}, \omega_k^{L_k}), \; \forall i \in \mathcal{N}_k; \tag{9}$$

as a timed sequence of tasks, where $\omega_k^\ell \in \Omega$ is the assigned $\ell$-th task of subteam $\mathcal{N}_k \in \overline{\mathcal{N}}$; $S_k^\ell \subset \mathcal{W}$ is the associated region; and $L_k > 0$ is the total length. Note that the total number of tasks $\overline{\Omega}_t$ is much larger than the number of subteams, and continually expanded online.

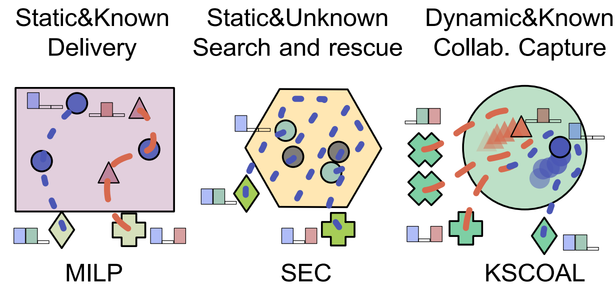


Fig. 2. Illustration of three types of local tasks described in Sec. III-B, and the associated coordination strategy.

## B. Local Task Coordination

Once the local task plans are derived, each robot $i \in \mathcal{N}$ starts executing its $\ell$-th task $(S_k^\ell, \omega_k^\ell)$, i.e., to navigate to region $S_k^\ell$ and perform the task $\omega_k^\ell$. The required subtasks from (3) are denoted by $\mathcal{J}_k^\ell \triangleq \{(n_j, a_j, s_j), j = 1, \cdots, J_k^\ell\}$. Consequently, all agents in $\mathcal{N}_k$ should collaboratively fulfill these subtasks to minimize the overall duration. More specifically, the local *action plan* of each agent $i \in \mathcal{N}_k$ for task $\omega_k^\ell$ is given by $\tau_i = (t_i^1, g_i^1, a_i^1)(t_i^2, g_i^2, a_i^2)\cdots$, as the sequence of timed goal, positions and actions. The overall objective of the local coordination is to optimize the collective plans $\boldsymbol{\tau}_k^\ell \triangleq \{\tau_i, i \in \mathcal{N}_k\}$ such that the makespan of $\omega_k^\ell$, denoted by $T_k^\ell$, is minimized, i.e., $\mathbf{min}_{\boldsymbol{\tau}_k^\ell} \{T_k^\ell\}$. However, as specified earlier, there are uncertainties regarding the total number of subtasks $J_k^\ell$ and their locations $\{s_j\}$ within the task $\omega_k^\ell$. Thus, depending on the characteristics of the tasks and workspace, ***three*** different local coordination strategies are adopted, as shown in Fig. 2.

*1) Static and Known Tasks:* As the first case, consider that the locations and the number of subtasks are all known and static. For instance, the task of "delivery" often consists of several locations to visit in a region and collaboratively deliver some objects, which are often known beforehand according to orders. In this case, variants of the multi-vehicle routing problem can be formulated as a MILP by enforcing the collaborative actions at each location. The key is that the constraints are formulated according to the navigation model of each agent and the duration function $\eta_k^\ell$ from (3) given the assignment variables. The exact formulation is omitted here due to limited space. Denote by $\boldsymbol{\tau}_k^{\ell,\star}$ the resulting local plans, which can be then sent to all agents.

*2) Static and Unknown Tasks:* For the second case, the number and location of the subtasks are unknown or uncertain, but the subtasks remain static and immobile during execution. For instance, for the task of "search and rescue", the exact number of victims within the region is unknown and can only be determined during online execution. Thus, a simultaneous exploration and coordination (SEC) method is proposed. To begin with, a collaborative exploration strategy is adopted for the subteam $\mathcal{N}_k^\ell$ to explore the region $S_k^\ell$ for potential subtasks, e.g., frontiers-based [34] and sampling-based [35]. Without loss of generality, the set of exploration subtasks at time $t > 0$ is associated with the points to visit in the region, denoted by $\mathcal{J}_t^{\rm e}$. Moreover, numerous collaborative subtasks in $\mathcal{J}_k^\ell$ are detected at time $t$ along with its location and the required number of agents, denoted by $\mathcal{J}_t^{\mathrm{c}}$. Consequently, the set of known subtasks that has not been fulfilled is denoted by $\mathcal{J}_t \triangleq \mathcal{J}_t^{\mathrm{e}} \cup \mathcal{J}_t^{\mathrm{c}}$. Since the subtasks in $\mathcal{J}_t$ are constantly changing, a rolling assignment algorithm similar to Alg. 1 is adopted, i.e., to assign subtasks within $\mathcal{J}_t$ in small batches via the optimal algorithm described in the first case. This procedure continues until the region is fully explored and all detected subtasks are completed.

*3) Dynamic and Known Tasks:* For the third case, the total number of the subtasks and their locations are known, but the subtasks are dynamic and mobile during execution. For instance, the task of "collaborative capture" often requires the agents to form subteams in order to surround and capture numerous moving targets. In this case, the previous two strategies are not suitable as the motion of subtasks would quickly render the current plans highly suboptimal or even infeasible. Thus, a dynamic coalition formation (DCF) method is adopted for this case, as proposed in our earlier work [31]. Particularly, each agent $i \in \mathcal{N}_k^\ell$ only decides the next subtask to perform, along with other agents as a coalition, i.e., $i \in \mathcal{N}_j \in \widehat{\mathcal{N}}_t$, where $\widehat{\mathcal{N}}_t \triangleq \{\mathcal{N}_j,\, j \in \mathcal{J}_k^\ell\}$ is the coalition scheme with all coalitions at time $t > 0$; it holds that $\mathcal{N}_{j_1} \cap \mathcal{N}_{j_2} = \varnothing$ and $\cup_{j \in \mathcal{J}_k^\ell} \mathcal{N}_j \subseteq \mathcal{N}_k^\ell$. It has been proven in [31] that the DCF method converges to a K-serial stable (KSS) coalition scheme after a finite number of distributed coordination. The detailed algorithm is omitted here due to limited space. Afterwards, the agents would complete the assigned subtask as coalitions and the coalition scheme is updated each time a subtask is completed.

**Remark 4.** The case of dynamic and unknown tasks is not considered since: (I) Without knowing the total number and locations of subtasks, it is difficult to determine whether the current task is completed; (II) A combination of the strategy for the second and third cases above would suffice. ■

### C. Overall Framework

*1) Online Execution and Adaptation:* Initially at $t = 0$, given the initially-known workspace and mission descriptions, the missions are decomposed into tasks, based on which the set of local teams are formed as $\overline{\mathcal{N}}$. These tasks are assigned to the teams with a given horizon $H$ and a redundancy $\rho$ via Alg. 1, yielding the local plan $\xi_k$ for each team $\mathcal{N}_k \in \overline{\mathcal{N}}$. Afterwards, the teams start executing the task $(S_k^\ell, \omega_k^\ell) \in \xi_k$ by navigating to the desired region $S_k^\ell$ and performing the task $\omega_k^\ell$. Depending on the exact type of task $\omega_k^\ell$, the set of subtasks $\mathcal{J}_k^\ell$ contained within $\omega_k^\ell$ is executed by team $\mathcal{N}_k$ following one of the three local coordination strategies to derive local action plans $\{\tau_i\}$. Note that all teams are executed concurrently, and all agents within the same team are also acting in parallel.

The conditions for replanning are designed as follows: (I) If more than half of the assigned $H$ tasks are accomplished; (II) If new missions are specified; or (III) If the local coordination of certain subteams returns infeasible. During replanning, the task assignment and subteam formation are updated by calling Alg. 1 given the current system state.

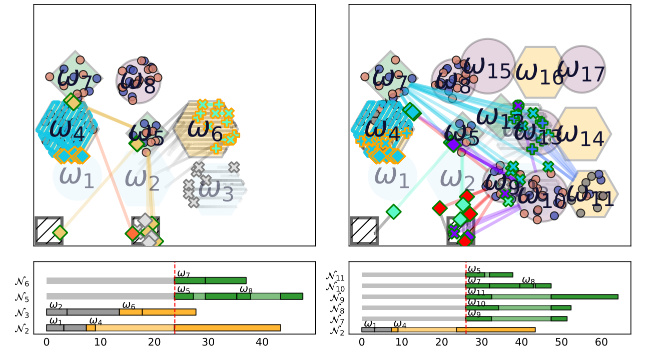

Fig. 3. Snapshots of simulation at $t = 23s$ (**Left**) and $t = 62s$ (**Right**) when new missions are released and replanning occurs.

However, the tasks that are currently being executed can not be preempted, which is essential when there are significantly more tasks than the number of subteams.

*2) Complexity Analysis:* In each iteration of Alg. 1, since $H$ tasks are assigned, the complexity reaches $\mathcal{O}(H2^H)$. During subteam formation, $H \cdot N$ Boolean variables are introduced to indicate the membership of agents in subteams. Regarding different types of tasks, $|\mathcal{N}_k|(J_k^\ell)^2$ integer variables are introduced to solve the MILP for the static and known tasks; the planning complexity for the static and unknown tasks is $\mathcal{O}((J_k^\ell)^3)$, similar to Alg. 1; and the complexity for the dynamic and unknown tasks is $\mathcal{O}((N_{\omega_k} J_k^\ell)(2J_k^\ell + N_{\omega_k})|\mathcal{N}_k|)$, where $N_{\omega_k}$ is the upper bound of agent number for each coalition [31].

## IV. Numerical Experiments

To numerical validations, the proposed method is implemented in `Python3` and tested on a laptop with an Intel Core i5-12500H CPU. The solver `GLOP` [33] is adopted for integer optimization. Simulation videos can be found in the supplementary files.

### A. System Description

As shown in Fig. 3, the simulated fleet consists of $N = 80$ heterogeneous agents in an open environment with map size 30m × 25m. The agents fall into 3 kinds with varying capabilities: 20 Type-A agents capable of perception and delivery, 30 Type-B agents capable of perception and grasping, and 30 Type-C agents capable of delivery and grasping. Initially, the agents are distributed evenly at two bases. All agents adhere to first-order dynamics and have a maximum speed of 1.5m/s in simulation.

Moreover, there are $|\boldsymbol{\varphi}_t| = 4$ missions released at random time instants, with an average interval $\mu = 30s$ with a standard deviation of $\sigma = 10s$. The sc-LTL missions follow a template of $\varphi_i = \Diamond(\varphi_{\texttt{del}} \wedge \Diamond\varphi_{\texttt{surv}}) \wedge (\neg\varphi_{\texttt{cap}} \mathcal{U} \varphi_{\texttt{surv}})$, where the 3 types of tasks are: "delivery" task, requiring delivery or grasping for 2 different subtasks; "surveillance" task, requiring perception; and "dynamic capture" task, requiring delivery or grasping for 2 different subtasks. Delivery tasks have in average 13 subtasks, and 15 subtasks for surveillance tasks with a probability of 0.5 to be unknown. The capture tasks have around 17 dynamic targets with

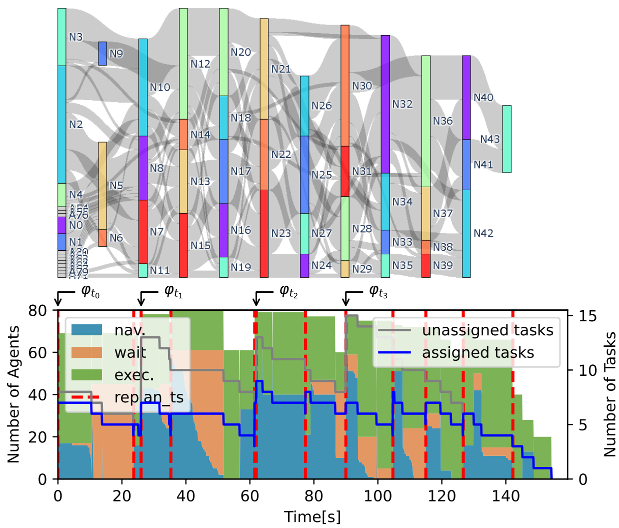

Fig. 4. **Top**: the number of subteams and their composition; **Bottom**: the status of agents in different modes and the number of tasks.

speed 0.5m/s inside the region. The planning horizon is set to $H = 6$ and replanning conditions follow the III-C.

### B. Results

As shown in Fig. 1 and 3, the first mission is known initially, it takes $0.2s$ for the method from [29] to compute its posets, yielding 8 tasks in total. Among these tasks, 8 pairs follow the "$\preceq$" relation and 1 for the "$\simeq$" relation. Moreover, it takes $1.1s$ for Alg. 1 to determine that subteams are required with a predicted makespan of $41s$. The subteams consist of maximum 25 agents and minimum 4 agents. For the first task of each subteam, the completion time is in average $15.9s$ and in total 51 subtasks are detected. The average planning time for 3 types of tasks is $0.3s$, $0.2s$ and $0.5s$, respectively. After 3 tasks are completed at $t = 23.6s$, replanning is triggered, yielding 3 new and 1 old subteams for the remaining tasks. At $t = 26s$, another mission is released, which contains 9 tasks with 10 relations. Thus, replanning is triggered, yielding 3 subteams and a predicted makespan of $48.1s$, with a planning time of $0.13s$. The procedure continues with new missions released at $62s$ and $90s$, which leads to a **total number of** 30 **tasks and** 492 **subtasks**. The complete mission is accomplished at $154.3s$, during which 12 replannings are triggered. As shown in Fig. 3 and 4, the agents switch among navigation, waiting for collaboration, and task execution, of which the trajectories depend heavily on the type of tasks. Fig. 4 shows that the number of subteams and their composition change *dynamically* online, along with the number of tasks.

### C. Comparisons

The proposed method is compared against **six** baselines: (I) **C-MILP-1**, where a complete MILP is formulated for all agents $\mathcal{N}$ and tasks $\overline{\Omega}_t$ similar to [7], [19], i.e., without the subteam formation in Alg. 1; (II) **C-MILP-2**, which formulates a complete MILP directly for all agents $\mathcal{N}$ and subtasks $\{\mathcal{J}_k^\ell\}$, i.e., without the hierarchical scheme; (III) **SAMP-1**, where a sampling-based planner from [6] is adopted for all agents and tasks; (IV) **SAMP-2**, which applies the sampling-based planner directly to subtasks; (V) **Inf-H**, which is the same as our method but with a infinite horizon $H$, i.e., all known tasks are assigned in Alg. 1; (VI) **Greedy**, which assigns maximum one task to each subteam, i.e., without the horizon $H$. As summarized in Table I. the proposed method excels at almost all metrics including response time, planning time and success rate, compared with **C-MILP-1,2** and **SAMP-1,2**. Particularly, via the proposed hierarchical solution, the planning time is 50 times lower than **C-MILP-1** and **SAMP-1** that directly assign agents to tasks. Moreover, the methods **C-MILP-2** and **SAMP-2** often leads to unsuccessful executions without considering uncertainties in subtasks. Lastly, the maximum planning time for **Inf-H** can be prohibitively long ($\geqslant$ 15min), while **Greedy** deploys 65% more agents in travelling than our methods.

TABLE I
COMPARISON WITH BASELINES

| Env. N | Env. M | Env. J | Methods | Resp. Time[s] | Ave/Max Plan Time[s] | T/W/X Agents | Succ. Rate[%] |
|---|---|---|---|---|---|---|---|
| 80 | 30 | 492 | **Ours** | 62.7 | 1.4/4.9 | 20/11/37 | 100 |
| | | | C-MILP-1 | 88.5 | 100/192 | 16/6/26 | 100 |
| | | | C-MILP-2 | 28.3 | 19/64 | 12/10/25 | 86 |
| | | | SAMP-1 | 108.3 | 54/92 | 19/8/24 | 100 |
| | | | SAMP-2 | 40.7 | 5.1/16 | 23/5/20 | 85 |
| | | | Inf-H | 77.5 | $>$ 15min | 24/17/31 | 100 |
| | | | Greedy | 112.1 | 0.3/0.6 | 33/9/30 | 100 |

TABLE II
SCALABILITY ANALYSIS

| Env. N | Env. M | Env. J | $\alpha$ | Resp. Time[s] | Ave/Max Plan Time [s] | T/W/X Agents | Succ. Rate[%] |
|---|---|---|---|---|---|---|---|
| 120 | 50 | 824 | 0.05 | 95 | 1.6/4.6 | 33/10/39 | 100 |
| | | | 0.1 | 102 | 1.7/4.9 | 26/9/35 | 100 |
| 150 | 80 | 1319 | 0.05 | 153 | 1.8/5.1 | 37/12/37 | 100 |
| | | | 0.1 | 165 | 2.1/6.2 | 12/11/34 | 97 |

### D. Generalization

For further validation, the fleet size is further increased and the agents can fail with a probability of $\alpha$. (I) ***Scalability***: As summarized in Table II, when the fleet size is increased to 120 and 150, while the number of tasks to 50 and 80, at $\alpha = 0.05$, the average planning time is increased by 12.5% from $1.6s$ to $1.8s$, while the maximum planning time increases from $4.6s$ to 5.1. Moreover, the average response time decreases from 95 to 153, as the average number of deployed agents increases from 82 to 86. (II) ***Failure Recovery***: When $\alpha = 0.05$, the success rate remains 100% for 150 agents even when the task number reaches 80. However, the success rate drops to 97% when the failure rate reaches 0.1, due the limits of remaining agent capacities.

## V. CONCLUSION

This work proposes a hierarchical coordination framework (HULK) that combines the global task assignment and the local subtask coordination, under continual and uncertain collaborative tasks. Future work includes human interaction and motion constraints.

# REFERENCES


[1] T. Arai, E. Pagello, L. E. Parker *et al.*, "Advances in multi-robot systems," *IEEE Transactions on robotics and automation*, vol. 18, no. 5, pp. 655–661, 2002.

[2] P. Toth and D. Vigo, "An overview of vehicle routing problems," *The vehicle routing problem*, pp. 1–26, 2002.

[3] O. M. Cliff, R. Fitch, S. Sukkarieh, D. L. Saunders, and R. Heinsohn, "Online localization of radio-tagged wildlife with an autonomous aerial robot system," in *Robotics: Science and Systems*, 2015.

[4] J. Fink, M. A. Hsieh, and V. Kumar, "Multi-robot manipulation via caging in environments with obstacles," in *2008 IEEE International Conference on Robotics and Automation*. IEEE, 2008, pp. 1471–1476.

[5] 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.

[6] 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.

[7] 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.

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

[9] A. Khamis, A. Hussein, and A. Elmogy, "Multi-robot task allocation: A review of the state-of-the-art," *Cooperative Robots and Sensor Networks 2015*, pp. 31–51, 2015.

[10] E. Nunes and M. Gini, "Multi-robot auctions for allocation of tasks with temporal constraints," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 29, no. 1, 2015.

[11] P. Brucker, B. Jurisch, and B. Sievers, "A branch and bound algorithm for the job-shop scheduling problem," *Discrete applied mathematics*, vol. 49, no. 1-3, pp. 107–127, 1994.

[12] R. Massin, C. J. Le Martret, and P. Ciblat, "A coalition formation game for distributed node clustering in mobile ad hoc networks," *IEEE Transactions on Wireless Communications*, vol. 16, no. 6, pp. 3940–3952, 2017.

[13] R. Fukasawa, H. Longo, J. Lysgaard, M. P. De Aragão, M. Reis, E. Uchoa, and R. F. Werneck, "Robust branch-and-cut-and-price for the capacitated vehicle routing problem," *Mathematical programming*, vol. 106, no. 3, pp. 491–511, 2006.

[14] L. Luo, N. Chakraborty, and K. Sycara, "Distributed algorithms for multirobot task assignment with task deadline constraints," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 3, pp. 876–888, 2015.

[15] J. C. Boerkoel Jr, L. R. Planken, R. J. Wilcox, and J. A. Shah, "Distributed algorithms for incrementally maintaining multiagent simple temporal networks," in *International Conference on Automated Planning and Scheduling*, 2013.

[16] H. H. Hoos and T. Stützle, *Stochastic local search: Foundations and applications*. Elsevier, 2004.

[17] M. Lahijanian, S. B. Andersson, and C. Belta, "Temporal logic motion planning and control with probabilistic satisfaction guarantees," *IEEE Transactions on Robotics*, vol. 28, no. 2, pp. 396–409, 2011.

[18] 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.

[19] X. Luo and M. M. Zavlanos, "Temporal logic task allocation in heterogeneous multi-robot systems," *arXiv preprint arXiv:2101.05694*, 2021.

[20] A. M. Jones, K. Leahy, C. Vasile, S. Sadraddini, Z. Serlin, R. Tron, and C. Belta, "Scratchs: Scalable and robust algorithms for task-based coordination from high-level specifications," in *Proc. Int. Symp. Robot. Res.*, 2019, pp. 1–16.

[21] M. Guo and D. V. Dimarogonas, "Multi-agent plan reconfiguration under local ltl specifications," *The International Journal of Robotics Research*, vol. 34, no. 2, pp. 218–235, 2015.

[22] J. Tumova and D. V. Dimarogonas, "Multi-agent planning under local ltl specifications and event-based synchronization," *Automatica*, vol. 70, pp. 239–248, 2016.

[23] 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.

[24] M. Guo and D. V. Dimarogonas, "Task and motion coordination for heterogeneous multiagent systems with loosely coupled local tasks," *IEEE Transactions on Automation Science and Engineering*, vol. 14, no. 2, pp. 797–808, 2016.

[25] C. Menghi, S. Garcia, P. Pelliccione, and J. Tumova, "Multi-robot ltl planning under uncertainty," in *International Symposium on Formal Methods*. Springer, 2018, pp. 399–417.

[26] Y. Kantaros and M. M. Zavlanos, "Distributed optimal control synthesis for multi-robot systems under global temporal tasks," in *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, 2018, pp. 162–173.

[27] P. Schillinger, "Specification decomposition and formal behavior generation in multi-robot systems," Ph.D. dissertation, KTH Royal Institute of Technology, 2019.

[28] C. Baier and J.-P. Katoen, *Principles of model checking*. MIT press, 2008.

[29] 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.

[30] K. R. Apt and A. Witzel, "A generic approach to coalition formation," *International game theory review*, vol. 11, no. 03, pp. 347–367, 2009.

[31] J. Chen, Z. Tang, and M. Guo, "Accelerated k-serial stable coalition for dynamic capture and resource defense," *IEEE Robotics and Automation Letters*, vol. 9, no. 1, pp. 443–450, 2024.

[32] Z. Liu, M. Guo, W. Bao, and Z. Li, "Fast and adaptive multi-agent planning under collaborative temporal logic tasks via poset products," *Research*, vol. 7, p. 0337, 2024.

[33] G. L. O. Solver, `https://developers.google.com/optimization/lp`.

[34] 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.

[35] D. Duberg and P. Jensfelt, "Ufoexplorer: Fast and scalable sampling-based exploration with a graph-based planning structure," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 2487–2494, 2022.