2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

UMBRELLA: Uncertainty-aware Multi-robot Reactive Coordination under Dynamic Temporal Logic Tasks

Qisheng Zhao[1], Meng Guo[1], Hengxuan Du[1], Lars Lindemann[2], and Zhongkui Li[1]

***Abstract*— Multi-robot systems can be extremely efficient for accomplishing team-wise tasks by acting concurrently and collaboratively. However, most existing methods either assume static task features or simply replan when environmental changes occur. This paper addresses the challenging problem of coordinating multi-robot systems for collaborative tasks involving dynamic and moving targets. We explicitly model the uncertainty in target motion prediction via Conformal Prediction (CP), while respecting the spatial-temporal constraints specified by Linear Temporal Logic (LTL). The proposed framework (UMBRELLA) combines the Monte Carlo Tree Search (MCTS) over partial plans with uncertainty-aware rollouts, and introduces a CP-based metric to guide and accelerate the search. The objective is to minimize the Conditional Value at Risk (CVaR) of the average makespan. For tasks released online, a receding-horizon planning scheme dynamically adjusts the assignments based on updated task specifications and motion predictions. Spatial and temporal constraints among the tasks are always ensured, and only partial synchronization is required for the collaborative tasks during online execution. Extensive large-scale simulations and hardware experiments demonstrate substantial reductions in both the average makespan and its variance by 23% and 71%, compared with static baselines.**

I. Introduction

Recent advances in computation, perception and communication enable the deployment of autonomous robots in large, remote and hazardous environments, such as offshore drilling platforms [1] and construction sites [2]. Concurrent motion and actions can greatly improve the team efficiency [3], [4], while direct collaboration on a single task further extends system capabilities [5]. To specify complex tasks beyond simple sequential visiting, many studies employ formal languages such as Linear Temporal Logic (LTL) formulas [6], as an intuitive yet powerful way to describe both spatial and temporal requirements on the team behavior [7], [8]. However, while tasks are commonly defined over static features such as regions and landmarks, many real-world applications involve dynamic targets, e.g., monitoring animal flocks or tracking moving vehicles [9]. These scenarios pose particular challenges for traditional offline methods that are designed for tasks over static features [10]–[12], as the future motion of targets can significantly impact the performance and even correctness of the overall plan.

The authors are with: [1]School of Advanced Manufacturing and Robotics, Peking University, Beijing 100871, China; and [2]Automatic Control Laboratory, ETH Zürich, Zürich 8092, Switzerland. This work was supported by the National Natural Science Foundation of China under grants 62425301, U2241214, T2121002. Corresponding author: Zhongkui Li (zhongkui.li@pku.edu.cn).

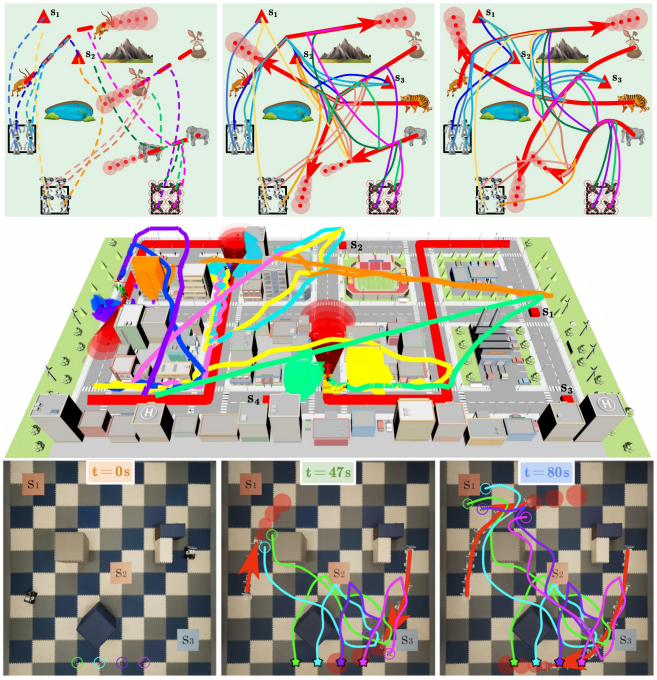

Fig. 1. **Top:** Task plans with 12 robots coordinating to track 4 dynamic targets across 12 tasks in two scenes (Scene-1: left and middle; Scene-2: right). **Middle:** ROS simulation with 8 robots and 3 dynamic targets executing 10 tasks. **Bottom:** Hardware experiments with 4 robots and 2 dynamic targets performing 7 tasks, showing snapshots at different times.

A. Related Work

Extensive work has addressed task planning for team-wise temporal-logic specifications. Centralized methods are often used to ensure optimality and completeness. A sampling-based method in [7] avoids synchronized products of individual models. The work in [13] decomposes team tasks into subtasks and assigns them to individual robots. Works [10], [11] formulate task constraints and assignments as integer optimization programs. However, these works typically assume static propositions and operate offline.

For dynamic workspaces, [14] performs local replanning for LTL tasks using updated workspace models. A reactive planning framework for heterogeneous robots is introduced in [15], allowing dynamic adaptation to collaborative temporal logic missions through local reallocation and path adjustment, minimizing task violations. A sampling-based coordination method in [16] addresses environments with uncertain semantic labels and known target motions. To handle online temporal tasks, [12] proposes computing products of partially-ordered subtasks and reassigning them after each task update. However, for tasks defined over dynamically

moving features, the aforementioned methods would mostly plan over the current system state, neglecting their future motions and more importantly, the inherent uncertainties.

Furthermore, Conformal Prediction (CP) has recently been used in robotics to quantify uncertainties in prediction [17]–[19]. In [17], CP is integrated with model predictive control to ensure motion safety for simple navigation tasks in dynamic environments. Temporal correlations are predicted using CP in [18] to enhance long-horizon performance for single-robot navigation. CP-based trajectory predictors for uncontrollable dynamic agents are applied to Signal Temporal Logic (STL) control in [19]. However, the planning problem for multi-robot collaborative tasks involving dynamic and uncertain targets remains an open challenge.

B. Our Method

This work proposes **UMBRELLA**, an online planning framework for multi-robot coordination under dynamic temporal tasks. It integrates CP-based motion prediction with MCTS for task assignment. Predicted target motions and quantified uncertainty are used after "Expansion" to filter child nodes, and incorporated into "Simulation" to evaluate partial plans under the Conditional Value at Risk (CVaR) metric. To handle online updates of target motions and newly released tasks, a receding-horizon scheme is adopted to dynamically adjust task assignments. The framework ensures LTL-specified spatial-temporal constraints while accounting for target motion uncertainty, requiring only partial synchronization during collaborative task execution. Compared with offline baselines assuming static workspaces or simple periodic replanning, our method significantly reduces both the mean and variance of the average makespan.

The main contribution is two-fold: (I) the incorporation of CP-based prediction into online multi-robot coordination under complex temporal tasks associated with dynamic targets; (II) a substantial reduction in the average makespan for temporal tasks specified offline and released online, with a 23% and 71% decrease in mean and variance, respectively.

II. Problem Description

A. Robots and Targets

Consider a team of N autonomous robots operating within a workspace $\mathcal{W} \subset \mathbb{R}^3$. The state of robot $n \in \mathcal{N} \triangleq \{1, \cdots, N\}$ at time t is $x_t \in \mathcal{W}$. The maximum velocity of robot n is denoted by v_n. Each robot n can execute actions from a set $\mathcal{A}_n$. Let $\mathcal{A} \triangleq \bigcup_{n\in\mathcal{N}} \mathcal{A}_n$. The team is preprogrammed with collaborations $\mathcal{C} \triangleq \{C_1, \cdots, C_k\}$, where each collaboration $C_k \in \mathcal{C}$ consists of a list of actions that must be executed by different robots, denoted by: $C_k \triangleq [a_1, a_2, \cdots, a_{\ell_k}]$, where $\ell_k > 0$ is the number of required actions, and $a_\ell \in \mathcal{A}$ for $\forall \ell = 1, \cdots, \ell_k$. Each action $a_\ell \in C_k$ should be performed by a capable robot n, i.e., $a_\ell \in \mathcal{A}_n$. Each collaboration has a fixed duration $\rho : \mathcal{C} \to \mathbb{R}_+$.

Moreover, the workspace contains M dynamic targets with unknown trajectories. The position of target $m \in \mathcal{M} \triangleq \{1, \cdots, M\}$ at time t is modeled as a random variable $Y_{t,m} \in \mathbb{R}^2$, with the joint target state $Y_t \triangleq (Y_{t,1}, \cdots, Y_{t,M}) \in \mathbb{R}^{2M}$. Their trajectories are assumed to follow an unknown distribution $\mathcal{D}$, i.e., $(Y_0, Y_1, \cdots) \sim \mathcal{D}$. A total of $\bar{K}$ independent trajectory samples $Y^{(i)} \triangleq (Y_0^{(i)}, Y_1^{(i)}, \cdots)$ are available, partitioned into a calibration set $D_{\texttt{cal}} \triangleq \{Y^{(1)}, \cdots, Y^{(K)}\}$ and a training set $D_{\texttt{tra}} \triangleq \{Y^{(K+1)}, \cdots, Y^{(\bar{K})}\}$. The distribution $\mathcal{D}$ is independent of robot behaviors, i.e., it does not depend on the team state x. Perfect observations of target positions and velocities are continuously available, denoted by $Y_{0:t} \triangleq (Y_0, \cdots, Y_t)$ and $V_{0:t} \triangleq (V_0, \cdots, V_t)$, where $V_t \triangleq (V_{t,1}, \cdots, V_{t,M})$. A centralized scheme aggregates all measurements and fuses them into real-time estimates of target states. For simplicity, static task regions and obstacles are modeled as targets with zero velocity.

B. Task Specification

Consider two types of atomic propositions: (I) p_n^m is *true* if the distance between robot $n \in \mathcal{N}$ and target $m \in \mathcal{M}$ is below a given threshold. Let $\mathbf{p} \triangleq \{p_n^m, \forall m \in \mathcal{M}, n \in \mathcal{N}\}$; (II) c_k^m is *true* if collaboration C_k is executed on target m. Let $\mathbf{c} \triangleq \{c_k^m, \forall m \in \mathcal{M}, \forall C_k \in \mathcal{C}, \forall a^c \in C_k\}$. Thus, the complete set of propositions is denoted by $AP \triangleq \mathbf{p} \cup \mathbf{c}$.

Given these propositions, a team-wise *task* is represented as a syntactically co-safe LTL (sc-LTL) formula: $\varphi_i = \text{sc-LTL}(AP)$. The overall task specification is represented as a set of LTL formulas, which consists of two parts:

$$\varphi \triangleq \varphi_{\texttt{static}} \bigwedge_{\bar{e}\in\bar{E}} \square \left(\varphi_{\texttt{obs}}^{\bar{e}} \to \Diamond \varphi_{\texttt{rep}}^{\bar{e}}\right), \tag{1}$$

where $\varphi_{\texttt{static}}$ is a predefined set of sc-LTL formulas; $\bar{E}$ is the set of events triggered by online observations; $\varphi_{\texttt{obs}}^{\bar{e}}$ is the propositional condition for event $\bar{e} \in \bar{E}$; and $\varphi_{\texttt{rep}}^{\bar{e}}$ is the corresponding response task, also specified as sc-LTL. Specifically, if $\varphi_{\texttt{obs}}^{\bar{e}}$ holds, then $\varphi_{\texttt{rep}}^{\bar{e}}$ must eventually be satisfied. We adopt the standard LTL syntax [6]: $\varphi \triangleq \top \mid p \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid \bigcirc \varphi \mid \varphi_1 \mathsf{U} \varphi_2$, where $\top \triangleq \texttt{True}$, $p \in AP$, $\bigcirc$ (*next*), U (*until*) and $\bot \triangleq \neg\top$. sc-LTL [20] is a fragment of LTL restricted to operators $\bigcirc$, U, and $\Diamond$ (eventually), written in positive normal form without the negation operator $\neg$ preceding temporal operators.

Moreover, the task plan for the robot team is defined as

$$\Pi_{\mathcal{N}} \triangleq (\pi_1, \pi_2, \cdots, \pi_N), \tag{2}$$

where $\pi_n \triangleq (t_n^1, a_n^1, m_n^1)(t_n^2, a_n^2, m_n^2) \cdots (t_n^{K_n}, a_n^{K_n}, m_n^{K_n})$ is the timed sequence of actions and corresponding targets for robot $n \in \mathcal{N}$. Each action $a_n^k \in \mathcal{A}_n$ is executed on target $m_n^k \in \mathcal{M}$ at time $t_n^k > 0$, where $k \in \mathcal{K}_n \triangleq \{1, \cdots, K_n\}$. Given $\Pi_{\mathcal{N}}$, the induced trace is given by the sequence of propositions satisfied by the actions, i.e., $w_\Pi = \sigma_1\sigma_2\cdots\sigma_L$ where $\sigma_\ell \in 2^{AP}$. The language of φ is $\mathcal{L}_\varphi \triangleq \{\boldsymbol{w} \mid w \models \varphi\}$, where $\models$ is the satisfaction relation. Since sc-LTL formulas admit satisfaction by finite traces [6], [20], the plan satisfies φ if $w_\Pi \in \mathcal{L}_\varphi$, denoted by $\Pi_{\mathcal{N}} \models \varphi$. The *makespan* T_φ is defined as the minimal time to generate a trace satisfying φ. As the tasks assigned to the robot system consist of initially issued tasks and online triggered tasks during execution, for a sequence of tasks $\varphi(t) \triangleq \{\varphi_1, \ldots, \varphi_L\}$ received up to time t, the *average makespan* is defined as: $\overline{T}_\varphi \triangleq \frac{1}{L}\sum_{\ell=1}^{L} T_{\varphi_\ell}$.

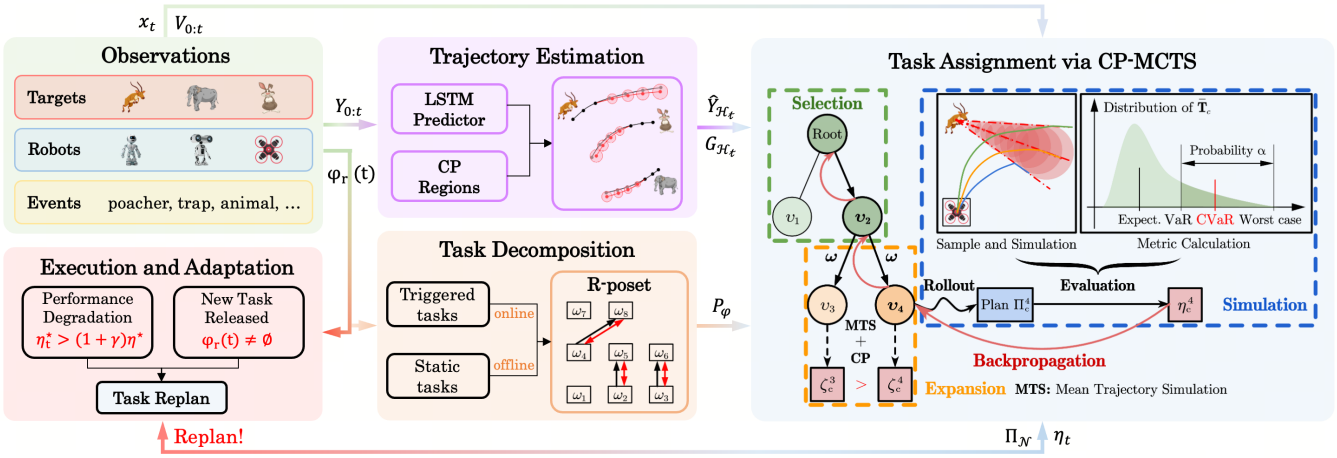

Fig. 2. Overview of the proposed framework, consisting of four main components: (i) trajectory estimation via LSTM and CP, (ii) task decomposition into an R-poset, (iii) CP-MCTS for uncertainty-aware assignment, and (iv) online execution and receding-horizon adaptation. In the R-poset illustration, precedence and mutual-exclusion relations are marked by black and red arrows, respectively.

C. Problem Statement

Given task formulas $\varphi(t)$ and observations $\{Y_{0:t}, V_{0:t}\}$, the overall objective is to synthesize the team-wise plan $\Pi_{\mathcal{N}}$ to satisfy $\varphi(t)$ and minimize the average makespan $\overline{T}_{\varphi}$.

Example 1: Consider a fleet of UAVs and ground robots (GRs) deployed for wildlife monitoring in a workspace labeled with `poacher`, `trap` and `animal`. An offline collaborative task φ_1 is specified as:

$$\begin{aligned} \varphi_1 &= \varphi_{\mathtt{p-s_1}} \wedge \varphi_{\mathtt{mf-a}},\ \varphi_{\mathtt{p-s_1}} = \Diamond \mathtt{patrol_{s_1}}, \\ \varphi_{\mathtt{mf-a}} &= \Diamond(\mathtt{monitor_a} \wedge \neg \mathtt{film_a} \wedge \Diamond \mathtt{film_a}), \end{aligned} \tag{3}$$

which requires to patrol region $\mathtt{s}_1$, monitor and film antelopes in sequence. GRs must avoid obstacles in the workspace. The reactive protocol specifies: $(\mathtt{poacher} \rightarrow \Diamond \mathtt{arrest})$, where an "arrest" task is triggered upon detecting a poacher, and $(\mathtt{trap} \wedge \mathtt{animal} \rightarrow \Diamond \mathtt{rescue})$, triggers a "rescue" task when an animal is trapped. ■

III. Proposed Solution

As illustrated in Fig. 2, the proposed solution consists of four components: the dynamic target trajectory estimation in Sec. III-A; the derivation of the relaxed partially-ordered set (R-poset) in Sec. III-B; the uncertainty-aware task assignment via CP-MCTS in Sec. III-C; and a receding-horizon planning scheme for real-time adaptation in Sec. III-D.

A. Trajectory Estimation of Dynamic Targets

1) Trajectory Predictor: Given a task horizon T_{φ} and the history of target observations $Y_{0:t}$, we train an independent trajectory predictor for each target based on the training dataset $D_{\mathtt{tra}}$. Let $D^m_{\mathtt{tra}} \subset D_{\mathtt{tra}}$ collect trajectories of target m, $Y^{(i)}_{0:T,m} \triangleq (Y^{(i)}_{0,m}, \cdots, Y^{(i)}_{t,m}, Y^{(i)}_{t+1,m}, \cdots, Y^{(i)}_{T_{\varphi},m})$ denotes the i-th trajectory in $D^m_{\mathtt{tra}}$. The trajectory predictor is defined as $\Upsilon_m : \mathbb{R}^{2(t+1)} \rightarrow \mathbb{R}^{2(T_{\varphi}-t)}$ that estimates the future states of target m as $\widehat{Y}_{\mathcal{H}_t,m} \triangleq \Upsilon_m(Y_{0:t,m}), \mathcal{H}_t \triangleq \{t+1, \cdots, T_{\varphi}\}$, where $\Upsilon_m(Y_{0:t,m}) \triangleq (\widehat{Y}_{t+1|t,m}, \cdots, \widehat{Y}_{T_{\varphi}|t,m})$. Let $\boldsymbol{\Upsilon} \triangleq (\Upsilon_1, \cdots, \Upsilon_M)$. In principle, any trajectory predictor can be employed, including long short-term memory (LSTM) networks [21], recurrent neural networks (RNN) [22], and gated recurrent units (GRU) [23]. In this work, a LSTM network is adopted for each target and trained by minimizing the following loss function:

$$\min_{\Upsilon_m} \frac{1}{|D^m_{\mathtt{tra}}|} \sum_{i=1}^{|D^m_{\mathtt{tra}}|} \left\| Y^{(i)}_{\mathcal{H}_t,m} - \Upsilon_m(Y^{(i)}_{0:t,m}) \right\|^2,$$

which is the Mean Squared Error (MSE) over the predicted trajectory for target m. Stacking per-target predictions gives $\widehat{Y}_{\mathcal{H}_t} \triangleq (\widehat{Y}_{\mathcal{H}_t,1}, \cdots, \widehat{Y}_{\mathcal{H}_t,M})$. Furthermore, based on the velocity measurements $V_{0:t}$, we define $v^*_m(t) \triangleq \max\{|V_{\varsigma,m}| : \varsigma \leqslant t\}$ as the maximum historical velocity of target m up to time t. This velocity bound serves as an input parameter for the subsequent planning module.

2) Conformal Prediction Regions: The CP framework can construct regions around predicted trajectories that contain the true trajectory with high probability, see [24] for detailed descriptions. More specifically, we adopt the method in [17] to construct valid prediction regions for each target independently. Given observations $Y_{0:t}$ at time t, the trajectory predictors $\boldsymbol{\Upsilon}$ generate predictions $\widehat{Y}_{\mathcal{H}_t}$ for the task horizon T_{φ}. For each target $m \in \mathcal{M}$, given a failure probability $\delta \in (0,1)$, the prediction regions $G_{\mathcal{H}_t,m} \triangleq (G_{t+1|t,m}, \cdots, G_{T_{\varphi}|t,m})$ are constructed such that:

$$\Pr\Big(\left\| Y_{h,m} - \widehat{Y}_{h|t,m} \right\| \leqslant G_{h|t,m}, \forall h \in \mathcal{H}_t \Big) \geqslant 1 - \delta, \tag{4}$$

where $G_{h|t,m}$ denotes the h-step prediction error for target m at time t. The *nonconformity score* is defined as

$$R^{(i)}_{h|t,m} \triangleq \left\| Y^{(i)}_{h,m} - \widehat{Y}^{(i)}_{h|t,m} \right\|, \tag{5}$$

for calibration trajectories $Y^{(i)} \in D^m_{\mathtt{cal}}$, where $D^m_{\mathtt{cal}}$ is the calibration dataset for target m. Specifically, predictions $\widehat{Y}^{(i)}_{h|t,m}$ are computed for each $Y^{(i)} \in D^m_{\mathtt{cal}}$, and the corresponding nonconformity scores $R^{(i)}_{h|t,m}$ are calculated. These scores are sorted in non-decreasing order after adding $R^{(|D^m_{\mathtt{cal}}|+1)}_{h|t,m} \triangleq \infty$. The error bound $G_{h|t,m}$ is then chosen as the p-th smallest score, where $p = \lceil (|D^m_{\mathtt{cal}}| + 1)(1 - \bar{\delta}) \rceil$. Finally, the prediction regions for all targets are aggregated as: $G_{\mathcal{H}_t} \triangleq (G_{h|t,1}, \cdots, G_{h|t,M})$.

B. R-posets for Task Formulas

To efficiently capture the temporal constraints embedded in a sc-LTL formula φ, we adopt the notion of relaxed partially ordered sets (R-posets) as proposed in [25].

Given φ, we first translate it into a Nondeterministic Büchi Automaton (NBA) $\mathcal{A}_\varphi = (S, \Sigma, \delta, S_0, S_F)$, where S is the set of states; $\Sigma = AP$ is the alphabet; $\delta : S \times \Sigma \to 2^S$ is the transition relation; $S_0, S_F \subseteq S$ denote the initial and accepting states. Along a satisfying run of $\mathcal{A}_\varphi$, a *subtask* ω is defined as the minimal symbol enabling a transition between states. Each subtask represents a unit of progress, and the collection of all subtasks forms Ω_φ.

Definition 1 (R-poset): An R-poset over φ is defined as the triple: $P_\varphi = (\Omega_\varphi, \preceq_\varphi, \neq_\varphi)$: (I) Ω_φ is the set of subtasks; (II) $\preceq_\varphi \subseteq \Omega_\varphi \times \Omega_\varphi$ is the precedence relation: if $(\omega_1, \omega_2) \in \preceq_\varphi$, then ω_2 cannot start before ω_1 starts; (III) $\neq_\varphi \subseteq 2^{\Omega_\varphi}$ is the mutual exclusion relation: subtasks in the same set cannot be executed simultaneously. ■

Although an R-poset is not unique for a given φ, the set of all possible R-posets P_φ is as expressive as the original NBA. Any plan consistent with an R-poset must satisfy φ, since Words$(P_\varphi) \subset$ Words(φ). In practice, we construct P_φ using the algorithm in [25], denoted as `Compute_poset`$(\cdot)$. An example of an R-poset is illustrated in Fig. 2.

C. Uncertainty-Aware Task Assignment

Given the estimation of target motions $(\widehat{Y}_{\mathcal{H}_t}, G_{\mathcal{H}_t})$ and the final R-poset $P_\varphi = (\Omega_\varphi, \preceq_\varphi, \neq_\varphi)$, the objective is to find an efficient assignment of all subtasks in Ω_φ given the robot team $\mathcal{N}$ such that all partial orders in $\preceq_\varphi$, $\neq_\varphi$ are respected and the average makespan of all tasks $\overline{T}_\varphi$ is minimized.

1) CP-based Monte Carlo Tree Search: Monte Carlo Tree Search (MCTS) is a well-known heuristic search algorithm for solving complex planning problems in dynamic scenes. Built upon this algorithm, this work introduces CP-based Monte Carlo Tree Search (CP-MCTS). As summarized in Alg. 1 and Fig. 2, it is a centralized task planning algorithm designed to efficiently handle complex temporal tasks associated with dynamic targets. It repeats four stages until the time budget expires: selection, expansion, simulation, and backpropagation. Notably, as the number of robots and tasks increases, the nodes generated during the "Expansion" phase grow rapidly. If "Simulation" is performed for all expanded nodes, the algorithm becomes biased toward breadth-first search, neglecting depth exploration and thus reducing efficiency. To mitigate this, a CP-based metric is introduced to efficiently evaluate and select the most promising child nodes, which are then advanced to the "Simulation" stage.

2) Selection and Expansion: Each node in the search tree represents a partial assignment of subtasks, i.e., $\nu \triangleq (\tau_1, \tau_2, \cdots, \tau_N)$, where τ_n is the ordered sequence of subtasks assigned to robot $n \in \mathcal{N}$.

During "Selection", the Upper Confidence Bound applied to Trees (UCT) [26] is used to choose nodes for "Expansion". The UCT value is computed as $(\overline{\xi}_i + Q \times \sqrt{\frac{lnB}{b_i}})$, where $\overline{\xi}_i$ is the estimated value of the i-th child node, b_i is its visit count, B is the visit count of the current node, and $Q > 0$ is a parameter that balances exploration and exploitation. Starting from the root, at each level the child with the highest UCT value is recursively selected until a leaf node is reached.

Algorithm 1: CP-based MCTS (`CP-MCTS`)

Input : Robots $\mathcal{N}$, poset P_φ, duration func. ρ, time budget t_b, target estimations $\widehat{Y}_{\mathcal{H}_t}, G_{\mathcal{H}_t}$.
Output: Plan $\Pi_{\mathrm{c}}^\star$, average makespan $\eta_{\mathrm{c}}^\star$.

Initialize root node ν_0, $\eta_{\mathrm{c}}^\star \leftarrow \infty$;
while $time < t_b$ **do**
  Leaf node $\nu \leftarrow$ `Selection`(ν_0);
  Child nodes $\{\nu_+\} \leftarrow$ `Expansion`(ν, P_φ);
  Filtered nodes $\{\nu_{\mathrm{s}}\}$ via ζ in (7);
  for $\nu_s \in \{\nu_s\}$ **do**
    `/* Simulation=Rollout+Eval */`
    $\Pi_{\mathrm{c}} \leftarrow$ `Rollout`$(\nu_{\mathrm{s}}, P_\varphi, \rho, \widehat{Y}_{\mathcal{H}_t}, x_t)$;
    $\eta_{\mathrm{c}} \leftarrow$ `Eval`$(\Pi_{\mathrm{c}}, P_\varphi, \rho, \widehat{Y}_{\mathcal{H}_t}, G_{\mathcal{H}_t}, x_t)$;
    if $\eta_c < \eta_c^\star$ **then**
      $\eta_{\mathrm{c}}^\star \leftarrow \eta_{\mathrm{c}}$, $\Pi_{\mathrm{c}}^\star \leftarrow \Pi_{\mathrm{c}}$;
    `Backpropagate`$(\nu_{\mathrm{s}}, \xi_{\mathrm{c}})$;

During "Expansion", unless a terminal state with a complete assignment is reached, child nodes are generated from the leaf node by assigning the *next* subtask to the robot team. Let $\Omega_\nu \triangleq \{\omega \in \tau_n, \forall n \in \mathcal{N}\}$ be the set of subtasks already assigned in node ν, and $\Omega_\nu^- \triangleq \Omega_\varphi \backslash \Omega_\nu$ the remaining ones. To ensure feasibility, the *next* subtask ω is chosen from $\Omega_\nu^{\mathrm{a}} \triangleq \{\omega \,|\, \omega \in \Omega_\nu^-, \omega' \in \Omega_\nu, \forall \omega' \in \mathrm{Pre}(\omega)\}$, where Pre$(\omega)$ represents the set of subtasks that must be completed before ω according to R-poset P_φ. In other words, a subtask ω_i cannot be assigned to node ν if some ω_j with $(\omega_j, \omega_i) \in \preceq_\varphi$ has not yet been assigned. When a subtask $\omega \in \Omega_\nu^+$ is selected, a child node ν^+ is created by assigning ω to a robot group $\mathcal{I}_\omega$, i.e., by appending it to the local plan τ_n of each robot $n \in \mathcal{I}_\omega$. For child node ν^+, define *key subtasks* $\Omega_{\nu^+}^{\mathrm{s}} \subset \Omega_{\nu^+}$ as: $\Omega_{\nu^+}^{\mathrm{s}} \triangleq \{\omega \,|\, \omega = \tau_n[0], \omega \notin \tau_{n'}[1:], \forall n, n' \in \mathcal{N}\}$, where $\Omega_{\nu^+}^{\mathrm{s}}$ contains the first-to-execute subtasks across robots, independent of others. Stepwise simulation is then performed based on the predicted target trajectories $\widehat{Y}_{\mathcal{H}_t}$, as described in the sequel, resulting in the predicted completion time $\widehat{T}_\omega$ for each assigned subtask $\omega \in \Omega_{\nu^+}$.

3) Simulation: The "Simulation" procedure consists of three steps: (I) *rollout* to complete an assignment; (II) sample the makespan distribution; (III) evaluate the plan based on this distribution. First, a *rollout* policy is applied recursively from the selected child node until all subtasks are assigned. In each iteration, the *next* subtask is selected as in "Expansion" and assigned to a robot group either randomly or greedily according to stepwise simulation results. To enhance *rollout* diversity, we use a random factor $\epsilon \in [0, 1]$: with probability ϵ, a feasible robot group is chosen uniformly at random; with probability $1-\epsilon$, the group expected to initiate this subtask earliest is selected. Once a complete plan Π_{c} is obtained, a sampling-based method derives its average makespan distribution $\overline{\mathbf{T}}_{\mathrm{c}}$ via stepwise simulation with $z \in \mathbb{N}$ samples drawn from the prediction regions $\widehat{Y}_{\mathcal{H}_t}, G_{\mathcal{H}_t}$.

Definition 2 (VaR and CVaR): The value at risk (VaR) at risk level $\alpha \in (0,1]$ is defined as $\mathrm{VaR}_\alpha(\Pi_\mathrm{c}) \triangleq \mathbf{inf}\{\rho \in \mathbb{R}, \mathrm{Prob}(\overline{\mathbf{T}}_\mathrm{c} \geqslant \rho) \geqslant \alpha\}$, i.e., the α-quantile of the distribution $\overline{\mathbf{T}}_\mathrm{c}$. The associated conditional value at risk (CVaR) is defined as: $\mathrm{CVaR}_\alpha(\Pi_\mathrm{c}) \triangleq \mathbb{E}\big[\overline{\mathbf{T}}_\mathrm{c} | \overline{\mathbf{T}}_\mathrm{c} \geqslant \mathrm{VaR}_\alpha(\Pi_\mathrm{c})\big]$, as the expected value of the worst α-quantile of $\overline{\mathbf{T}}_\mathrm{c}$. ■

As shown in Fig. 2, VaR captures the maximum loss at a given confidence level, while CVaR assesses the expected loss beyond the VaR threshold, providing a more informative measure of tail risk. Accordingly, we use $\eta_\mathrm{c} \triangleq \mathrm{CVaR}_\alpha(\Pi_\mathrm{c})$ as the evaluation metric for each plan based on its average makespan distribution. The procedure of deriving the distribution and computing the CVaR is encapsulated as $\texttt{Eval}(\cdot)$. During the search, the best plan $\Pi_\mathrm{c}^\star$ and its minimal risk value $\eta_\mathrm{c}^\star$ are maintained. For each new candidate Π_c, if $\eta_\mathrm{c} < \eta_\mathrm{c}^\star$, both $\Pi_\mathrm{c}^\star$ and $\eta_\mathrm{c}^\star$ are updated.

4) Backpropagation: To support the tree search, we define a normalized performance measure:

$$\xi_\mathrm{c} \triangleq 2 - \frac{\eta_\mathrm{c}}{\eta_\mathrm{c}^\star}, \tag{6}$$

which facilitates comparison across different branches of the search tree. During "Backpropagation", the evaluation values and visit counts of nodes are propagated and updated along the path from the selected node to the root.

Lemma 1: Given an expanded node ν^+, the completion times of *key subtasks* in $\Omega^\mathrm{s}_{\nu^+}$ satisfy that

$$\Pr\Big(T_\omega \leqslant \widehat{T}_\omega + \frac{G_{\widehat{T}_\omega,m}}{\mathbf{min}_{n\in\mathcal{I}_\omega}\{v_n\} - v_m^\star}, \forall \omega \in \Omega^\mathrm{s}_{\nu^+}\Big) \geqslant (1-\delta)^{|\Omega^\mathrm{s}_{\nu^+}|},$$

where T_ω is the actual completion time of subtask $\omega \in \Omega^\mathrm{s}_{\nu^+}$; m is the associated target; $\mathcal{I}_\omega$ is the set of robots executing ω; v_n and $v_m^\star$ are the velocities of robot n and target m.

Proof: From the simulation results, all robots $n \in \mathcal{I}_\omega$ assigned to subtask $\omega \in \Omega^\mathrm{s}_{\nu^+}$ would reach the predicted position $\widehat{Y}_{\widehat{T}_\omega,m}$ of target m by time instance $\widehat{T}_\omega$. By the probability guarantee in (4), it holds that

$$\Pr\Big(\Big\|Y_{\widehat{T}_\omega,m} - \widehat{Y}_{\widehat{T}_\omega,m}\Big\| \leqslant G_{\widehat{T}_\omega,m}, \forall \omega \in \Omega^\mathrm{s}_{\nu^+}\Big) \geqslant (1-\delta)^{|\Omega^\mathrm{s}_{\nu^+}|}.$$

The additional delay required to reach target m is upper-bounded by $\widehat{T}^\mathrm{e}_\omega = G_{\widehat{T}_\omega,m}/(\mathbf{min}_{n\in\mathcal{I}_\omega}\{v_n\} - v_m^\star)$, this completes the proof. ■

The actual completion time of a first executed subtask T_ω is probabilistically bounded by the predicted time $\widehat{T}_\omega$ plus an uncertainty term determined by the prediction region $G_{\widehat{T}_\omega,m}$ and the velocity difference between the slowest robot and the target. Hence, with confidence level $(1-\delta)$, each *key subtask* completion time can be reliably estimated. The evaluation metric for a child node ν^+ is then defined as:

$$\zeta_{\nu^+} \triangleq \frac{\sum_{i=1}^{|\Omega_{\nu^+}|} \widehat{T}_{\omega_i} + \sum_{j=1}^{|\Omega^\mathrm{s}_{\nu^+}|} \widehat{T}^\mathrm{e}_{\omega_j}}{|\Omega_{\nu^+}|}, \tag{7}$$

which approximates the average completion time of assigned subtasks with uncertainty adjustment. After "Expansion", for each *next* subtask, the child node with the smallest ζ is selected to enter "Simulation", forming the filtered set $\{\nu_\mathrm{s}\}$.

Algorithm 2: Online Dynamic Task Assignment

Input : Robots $\mathcal{N}$, task formula $\varphi(t)$, duration func. ρ, trajectory predictors $\boldsymbol{\Upsilon}$, prediction regions $G_{\mathcal{H}_t}$, observations $\{Y_{0:t}, V_{0:t}\}$.
Output: Assignment $\Pi^\star$.

Initialize $P_\varphi \leftarrow \texttt{Compute\_poset}(\varphi(0))$, $\Omega_\iota \leftarrow \varnothing$;
Initialize $\Pi^\star$, $\eta^\star \leftarrow \texttt{CP-MCTS}(P_\varphi, \rho, \hat{Y}_{\mathcal{H}_0}, G_{\mathcal{H}_0}, x_0)$;
while *not terminated* **do**
  Each robot $n \in \mathcal{N}$ applies π_n;
  Sense x_t and Y_t;
  Obtain predictions $\hat{Y}_{\mathcal{H}_t}$;
  Update $\varphi(t)$ by (8);
  Update P_φ by $\texttt{Compute\_poset}(\varphi(t))$;
  if $\varphi_\mathrm{r}(t)$ **then**
    $\Pi^\star$, $\eta^\star \leftarrow \texttt{CP-MCTS}(P_\varphi, \rho, \hat{Y}_{\mathcal{H}_t}, G_{\mathcal{H}_t}, x_t)$;
  else
    $\eta^\star \leftarrow \eta^\star - 1$;
    if $\Omega_\mathrm{c}(t) \neq \varnothing$ **then**
      Update $\eta^\star$ by (9);
      $\Omega_\iota \leftarrow \Omega_\iota \cup \Omega_\mathrm{c}(t)$;
    $\eta_t^\star \leftarrow \texttt{Eval}(\Pi^\star, P_\varphi, \rho, \hat{Y}_{\mathcal{H}_t}, G_{\mathcal{H}_t}, x_t)$;
    if $\eta_t^\star > (1+\gamma)\eta^\star$ **then**
      $\widehat{\Pi}_t$, $\widehat{\eta}_t \leftarrow \texttt{CP-MCTS}(P_\varphi, \rho, \hat{Y}_{\mathcal{H}_t}, G_{\mathcal{H}_t}, x_t)$;
      if $\widehat{\eta}_t < \eta_t^\star$ **then**
        $\Pi^\star \leftarrow \widehat{\Pi}_t$, $\eta^\star \leftarrow \widehat{\eta}_t$;
  $t \leftarrow t+1$;

Remark 1: For a child node ν^+, if a subtask $\omega \in \Omega^\mathrm{s}_{\nu^+}$ is assigned to a robot $n \in \mathcal{I}_\omega$ with velocity $v_n \leqslant v_m^\star$, the robot may be unable to complete the subtask. In this case, the predicted completion time is $\widehat{T}_\omega = \infty$, which yields $\zeta_{\nu^+} = \infty$; thus the node is excluded from the "Simulation" step. ■

D. Online Execution and Adaptation

1) Online Execution and Adaptation: As illustrated in Fig. 2 and summarized in Alg. 2, the planning scheme consists of two stages: initial planning and online adaptation. Following the initial plan, each robot $n \in \mathcal{N}$ executes its local plan π_n by navigating to the targets specified by the assigned subtasks and then performing the corresponding actions. When executing action $a_n^{K_n}$ on target $m_n^{K_n}$, the robot n must wait for collaborators if $a_n^{K_n}$ belongs to a multi-robot collaboration C_k (i.e., $|C_k| > 1$). Such partial synchronization is essential to handle uncertainties in navigation and task execution times. If the action is non-collaborative, the robot proceeds independently without waiting.

At each time step $t > 0$, the robot states x_t and target observations Y_t are updated, and new predictions $\hat{Y}_{\mathcal{H}_t}$ are generated from $Y_{0:t}$ using the trajectory predictors $\boldsymbol{\Upsilon}$. Moreover, the task formula $\varphi(t)$ is updated by integrating triggered reactive task formula, i.e.,

$$\varphi(t) \triangleq \varphi(t^-) \wedge \varphi_\mathrm{r}(t), \tag{8}$$

where $\varphi(t^-)$ is the previous formula and $\varphi_\mathrm{r}(t) \triangleq \bigwedge_{\bar{e}\in\bar{E}} \Diamond\varphi^{\bar{e}}_{\mathrm{rep}}(t)$ is the reactive tasks released at time t. The current R-poset P_φ is updated accordingly. Replanning is

triggered under two conditions: (I) new reactive tasks appear, i.e., $\varphi_{\mathtt{r}}(t) \neq \varnothing$, or (II) the current plan's performance degrades significantly, i.e., $\eta_t^\star > (1+\gamma)\eta^\star$, where $\gamma \in (0,1)$ is the replanning triggering ratio. Here, the performance metric $\eta_t^\star$ is computed via $\mathtt{Eval}(\cdot)$ given the latest observations. For completed tasks, the expected value $\eta^\star$ is updated as

$$\eta_+^\star \triangleq \frac{\big(|\varphi(t)| - |\Omega_\iota| + |\Omega_{\mathtt{c}}(t)|\big)}{|\varphi(t)| - |\Omega_\iota|}\eta_-^\star, \tag{9}$$

where $\eta_+^\star$ and $\eta_-^\star$ denote the updated and previous values; $|\varphi(t)|$ is the total number of tasks; $|\Omega_\iota|$ is the number of completed tasks; and $|\Omega_{\mathtt{c}}(t)|$ is the number of tasks finished at time t. When replanning is triggered, CP-MCTS generates a candidate plan $\widehat{\Pi}_t$ with metric $\widehat{\eta}_t$. The system adopts this new plan only if it offers improved performance as $\widehat{\eta}_t < \eta_t^\star$. Otherwise, the current plan $\Pi^\star$ is maintained and the metric $\eta^\star$ is monotonically decreased to reflect progress. This procedure repeats until the system terminates.

2) Complexity Analysis: The computational complexity of Alg. 2 is analyzed as follows. Generating a R-poset has worst-case complexity $\mathcal{O}(J^2)$, where J is the number of subtasks, bounded by the number of edges in the NBA. For task assignment, the worst-case search space is $\mathcal{O}(J! \cdot N^J)$, as subtask orderings are combinatorial and assignments grow exponentially with the number of robots N. In contrast, the *rollout* process remains $\mathcal{O}(J \cdot N)$, as subtasks are assigned randomly or greedily. The complexity of stepwise simulation is $\mathcal{O}(\frac{zT \cdot (J+N)}{\Delta t})$, where z is the number of samples, T is the makespan and Δt is the time step.

3) Generalization: There are two notable extensions of the proposed scheme. *(I) Robot Failures*. In the event of robot failures, a modified CP-MCTS is employed to generate a new plan. The root node is updated to exclude failed robots and include completed subtasks. If a failure occurs during execution, the affected subtask is rescheduled. The standard CP-MCTS procedure is then applied to update the assignment. *(II) Dynamic Task Priority*. When tasks are associated with different priorities α_ℓ, the optimization objective shifts from the average makespan $\overline{T}_\varphi$ to the weighted makespan $\widetilde{T}_\varphi \triangleq \sum_{\ell=1}^{L_t} \alpha_\ell T_{\varphi_\ell}$, which emphasizes high-priority tasks.

IV. Numerical Experiments

This section presents numerical validations of the proposed method over large-scale multi-robot systems. The implementation is in `Python3` and tested on a laptop with an AMD Ryzen 9 7845HX CPU. Simulation and experiment videos are available in the supplementary material.

A. Scenario Description

For the wildlife protection scenario in Fig. 1, four types of animals: antelope, rabbit, elephant, and tiger, move within an $800m \times 800m$ outdoor area containing obstacles such as reservoirs and hills. A total of 2000 synthetic trajectories are collected for the four types of animals as the training and calibration datasets (1000 each), where initial and target positions are specified and Gaussian noise is added to sampled waypoints. Four robots of each type are deployed:

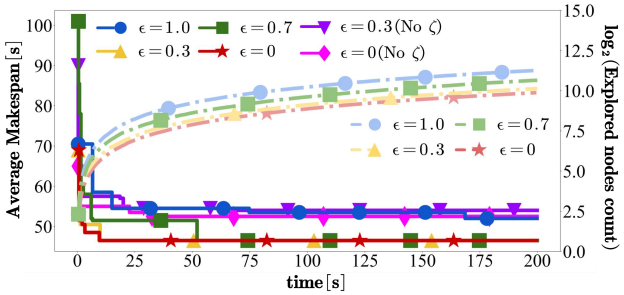

Fig. 3. The average makespan and the number of explored nodes with different random factors ϵ and expansion strategies (with or without CP-based metric ζ in (7)) during initial planning in Scene-1.

fast GRs $G_{\mathtt{f}}$ to monitor, patrol and arrest; slow GRs $G_{\mathtt{s}}$ to monitor, patrol and rescue; and UAVs U to monitor and film. GRs can navigate safely around obstacles as a team. Tasks are specified by the following LTL formulas, which require to monitor, film, and rescue animals, while also patrolling and arresting poachers in designated areas:

$$\varphi_{\mathtt{static}} = \varphi_{\mathtt{p-s_1}} \wedge \varphi_{\mathtt{p-s_2}} \wedge \varphi_{\mathtt{mf-a}} \wedge \varphi_{\mathtt{mf-r}} \wedge \varphi_{\mathtt{mf-e}},$$
$$\varphi_{\mathtt{r}}(50) = \varphi_{\mathtt{mf-t}},\ \varphi_{\mathtt{r}}(90) = \Diamond \mathtt{arrest}_{\mathtt{s_3}} \wedge \Diamond \mathtt{rescue}_{\mathtt{e}},$$

where $\varphi_{\mathtt{p}}$, $\varphi_{\mathtt{mf}}$ follow the same structure as in (3). Two scenes with different animal motion patterns are considered (Fig. 1): **Scene-1**: targets exhibit piecewise uniform linear motion; **Scene-2**: targets follow general smooth trajectories. The main parameters are: CP failure probability $\delta = 0.15$, CP-MCTS time budget $t_b = 10s$, exploration factor $Q = 1.5$, random factor $\epsilon = 0.3$, number of samples $z = 50$, risk level $\alpha = 0.05$, and replanning triggering ratio $\gamma = 0.2$.

B. Results

The results are shown in Figs. 1 and 4. At $t = 0$, trajectory prediction takes 0.05s, R-posets computation for 8 static tasks takes 0.03s, and the "Simulation" at each node averages 0.25s with $z = 50$ samples. The metric ζ_{ν^+} in (7) ranges from 24.5s to 72.3s, and a plan with average makespan $\eta_0^\star = 48$s is generated after exploring 118 nodes within $t_b = 10$s. At $t = 43s$, replanning is triggered due to performance degradation, with $\eta_{43}^\star = 14.2$s exceeding the threshold $(1+\gamma)\eta^\star = 13.7$s for $\gamma = 0.2$. After exploring 140 nodes, the new plan yields $\widehat{\eta}_{43} = 16.4$s, which does not improve performance; thus, the current plan is retained. At $t = 50s$, the appearance of a new target `tiger` and task $\varphi_{\mathtt{mf-t}}$ triggers replanning with 4 subtasks, where 389 nodes are explored, producing $\eta_{50}^\star = 28.2$s. At $t = 90s$, reactive tasks $\mathtt{arrest}_{\mathtt{s_3}}$ and $\mathtt{rescue}_{\mathtt{e}}$ trigger replanning with 3 unfinished subtasks, exploring 155 nodes and yielding $\eta_{90}^\star = 40.3$s. The system terminates at $t = 147$s, with an average makespan of 67.6s over 12 subtasks. Fig. 3 illustrates convergence and node exploration at $t = 0$. The algorithm is deemed converged when the best plan remains unchanged for more than 100s. As the random factor ϵ increases, more nodes are explored, but convergence time grows. With heuristic-based robot selection during *rollout*, convergence is achieved within 10s for $\epsilon = 0$ and $\epsilon = 0.3$. Moreover, using ζ_{ν^+} as in (7) to filter child nodes accelerates tree exploration and improves performance.

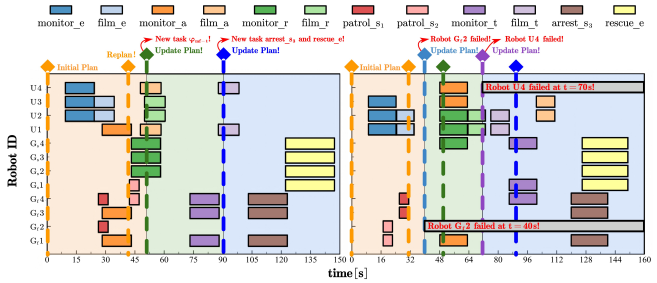

Fig. 4. **Left:** Gantt chart of Scene-1, where replanning happens when the predicted value $\eta_t^\star$ exceeds a threshold (orange line) and new tasks are triggered (green and blue lines). **Right:** Gantt chart of Scene-1 where two robots fail at $40s$ and $70s$ (in grey), respectively.

C. Comparisons

To evaluate the effectiveness of our method (**Ours**), we compare against **six** baselines: Mixed Integer Linear Programming (**MILP**): task decomposition and assignment via integer optimization [10]; Branch and Bound (**BnB**): a search-based method from [25]; No Trajectory Prediction (**NTP**) and No Uncertainty (**NU**): Ours without trajectory prediction or without uncertainty regions, respectively; **Ours-G**: Ours with a GRU-based trajectory predictor; **Ours-P10**: Ours with periodic replanning every 10s instead of event-triggered replanning. All methods are first evaluated for task allocation at $t = 0$ in Scene-1, executed over 500 target trajectory samples. Metrics include the mean and variance of average makespan and the solution time. They are further tested online in Scene-1 and Scene-2 with 500 samples, together with a clairvoyant strategy (**CS**) that has access to future trajectories and applies NU for assignment. Static methods (MILP, BnB, NTP) model each dynamic target as fixed at its most recently observed position during planning.

As shown in Table I, Ours outperforms static methods with reductions of 16.5%, 24.6%, and 22.5% in mean makespan and 64.9%, 67.3%, and 74.8% in variance across Scene-1 ($t = 0$), Scene-1, and Scene-2, respectively. BnB finds the first feasible solution quickly but converges slowly due to random exploration. MILP is the most time-consuming because of its exponential complexity. Ours requires more computation than NTP and BnB due to sampling and step-wise simulation, but achieves more reliable outcomes.

Although NU achieves a 6.7% lower mean makespan than Ours in Scene-1 at $t = 0$, it incurs $1.2\times$ higher variance and performs worse in online settings, confirming that Ours offers more risk-averse assignments via CP. Ours-G performs comparably to Ours, showing adaptability to different predictors. Ours-P10 increases mean makespan by 10.2% and variance by 4.2% relative to Ours, underscoring the benefit of event-triggered replanning. Finally, Ours remains within $6.0\% \sim 7.5\%$ of CS in mean makespan with only minor variance increases, indicating performance close to CS without prior knowledge of target motion.

D. Generalization

1) Scalability Analysis: Scalability is evaluated with respect to the number of robots N, tasks F, and targets M, as summarized in Table II. With fixed $F = 8$ and $M = 3$, increasing N from 12 to 36 raises the computation time for the first solution from 1.22s to 3.81s, while the convergence time remains around 4.7s. With fixed $M = 3$ and $N = 12$, increasing F from 8 to 16 extends convergence time from 4.7s to 15.5s, whereas the time for the first solution grows only modestly from 1.22s to 3.93s, owing to the polynomial complexity of *rollout*. With fixed $F = 16$ and $N = 12$, increasing M from 3 to 9 causes fluctuations in the first-solution time, but convergence time stays around 15.0s.

2) Robot Failure: In Scene-1, ground robot $G_{\mathtt{f}}2$ fails at $t = 40s$, and UAV $U4$ at $t = 70s$. As shown in the Gantt chart of Fig. 4, the failure of $G_{\mathtt{f}}2$ during antelope monitoring triggers replanning, where $U3$ replaces it. At $t = 70s$, the failure of UAV $U4$ during antelope filming leads to another update, assigning $U2$ and $U3$ to complete the task. Despite these failures, the team successfully completes all tasks by $t = 151s$, with an average makespan of $74.9s$.

3) Dynamic Task Priority: In Scene-1, the antelope-related task $\varphi_{\mathtt{mf-a}}$ is assigned a priority coefficient of 0.3, while other tasks are given 0.1. When optimizing for the average makespan $\overline{T}_\varphi$, the completion times for monitoring and filming the antelope are $T_{\mathtt{m-a}} = 92s$ and $T_{\mathtt{f-a}} = 112s$. Under weighted makespan $\widetilde{T}_\varphi$ minimization, these times decrease to $82s$ and $102s$, respectively, each reduced by $10s$.

4) High-fidelity ROS Simulation: In the second scenario (Figs. 1 and 5), three types of suspects: on foot $\mathtt{f}$, by bike $\mathtt{b}$, and by car $\mathtt{c}$ flee through city roads. Two types of robots, four of each, are deployed: UAVs $U_{\mathtt{t}}$ for tracking suspects and scouting, and UAVs $U_{\mathtt{i}}$ for interception and scouting. Tasks are given by $\varphi_{\mathtt{static}} = \varphi_{\mathtt{s-s_1}} \wedge \varphi_{\mathtt{s-s_2}} \wedge \varphi_{\mathtt{ti-f}} \wedge \varphi_{\mathtt{ti-b}} \wedge \varphi_{\mathtt{ti-c}}$, $\varphi_{\mathtt{r}}(40) = \varphi_{\mathtt{s-s_3}}$, $\varphi_{\mathtt{r}}(80) =$

TABLE I
COMPARISON AGAINST BASELINES

Method	Scene-1 ($\mathbf{t = 0}$)		Scene-1	Scene-2
	$\mathbf{M}(\overline{\mathbf{T}}_\phi)$, $\mathbf{V}(\overline{\mathbf{T}}_\phi)$[a]	$\mathbf{t_p}[s]$[b]	$\mathbf{M}(\overline{\mathbf{T}}_\phi)$, $\mathbf{V}(\overline{\mathbf{T}}_\phi)$	$\mathbf{M}(\overline{\mathbf{T}}_\phi)$, $\mathbf{V}(\overline{\mathbf{T}}_\phi)$
Ours	48.0, **3.25**	1.22, 4.70	67.6, 5.32	68.5, 4.34
NU	**44.8**, 7.05	0.30, 2.74	72.5, 14.32	71.8, 12.59
NTP	56.2, 10.05	0.25, **0.53**	81.9, 12.41	83.0, 17.21
MILP	59.1, 8.79	3.24, 905.6	94.9, 16.33	94.8, 18.12
BnB	57.1, 8.91	**0.06**, 35.22	92.0, 20.13	87.5, 16.24
Ours-G	49.9, 3.98	1.31, 5.21	66.9, 6.12	70.2, 5.43
Ours-P10	/	/	79.3, 11.45	77.8, 12.67
CS	/	/	**63.8**, **3.24**	**65.3**, **2.96**

[a] $\mathbf{M}(\overline{\mathbf{T}}_\phi)$, $\mathbf{V}(\overline{\mathbf{T}}_\phi)$: mean and variance of average makespan $\overline{T}_\phi$.
[b] The solution time $\mathbf{t_p}$ is measured by two timestamps: (i) when the first solution is returned; (ii) when the algorithm converges.
[c] Best values are in **bold**; second-best are underlined.

TABLE II
SCALABILITY ANALYSIS RESULTS

$(\mathbf{G_f}, \mathbf{G_s}, \mathbf{U})$[a]	$\mathbf{t_p}$ $(\mathbf{M = 3})[s]$			$\mathbf{t_p}(\mathbf{F = 16})[s]$	
	$\mathbf{F = 8}$	$\mathbf{F = 12}$	$\mathbf{F = 16}$	$\mathbf{M = 6}$	$\mathbf{M = 9}$
(4, 4, 4)	1.22, 4.7	2.03, 8.2	3.93, 15.5	2.04, 13.2	1.63, 15.6
(8, 8, 8)	2.71, 4.1	5.15, 9.2	7.64, 18.1	5.29, 12.1	7.59, 19.1
(12, 12, 12)	3.81, 4.9	6.19, 9.5	9.39, 18.3	8.35, 12.8	9.59, 16.3

[a] $(\mathbf{G_f}, \mathbf{G_s}, \mathbf{U})$: the number of three types of robots.

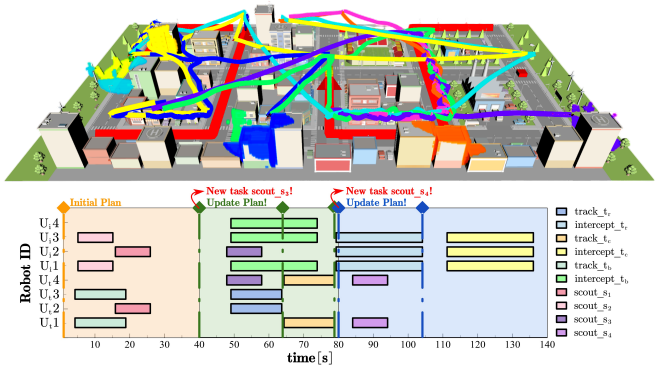

Fig. 5. ROS simulation results. **Top**: Robot and target trajectories. **Bottom**: Gantt chart of the execution timeline with task allocation and replanning.

$\varphi_{s-s_4}, \varphi_{s-s_1} = \Diamond \texttt{scout}_{s_1}$, $\varphi_{ti-f} = \Diamond(\texttt{track}_f \wedge \neg \texttt{intercept}_f \wedge \Diamond \texttt{intercept}_f)$. Task assignments are updated at $t = 40s$ and $t = 80s$ when new reactive tasks appear. Additional replanning occurs at $t = 64s$, $t = 79s$, and $t = 104s$, though the plan remains unchanged. The system terminates at $t = 136s$, completing all 10 tasks with an average makespan of $66.9s$.

E. Hardware Experiments

For further validation with hardware, we built a setup similar to the second scenario with 4 robots (UAV: $2U_t$, $2U_i$) and 2 targets (UGV: a_1, a_2) in a $4.95m \times 4.95m$ workspace, with the motion capture system OptiTrack providing their global states. Each robot communicates wirelessly with a control PC via ROS1. Mature navigation controllers are adopted and omitted for brevity. The tasks are specified as: $\varphi_{static} = \varphi_{s-s_1} \wedge \varphi_{s-s_2} \wedge \varphi_{ti-a_1} \wedge \varphi_{ti-a_2}$, $\varphi_r(80) = \varphi_{s-s_3}$. The algorithm uses the same parameters as in simulation. Assignments are updated at $t = 47s$ (due to performance degradation) and $t = 80s$ (due to a new task). The system terminates at $t = 113s$, completing all 7 tasks with an average makespan of $64.7s$. The resulting trajectory, snapshots, and Gantt chart are shown in Figs. 1 and 6.

V. Conclusion

This paper has presented **UMBRELLA**, an online multi-robot coordination framework for collaborative temporal tasks with dynamic targets. It achieves substantial reductions in both the mean and variance of the average makespan, while guaranteeing satisfaction of spatial-temporal task specifications. Future work includes intention-aware prediction for dynamic targets and finer motion constraints for robots.

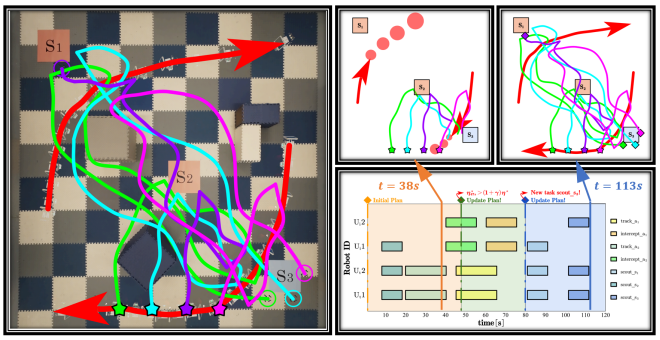

Fig. 6. Hardware experiment results. **Left:** Recorded robot and target trajectories. **Top-right:** Example trajectory snapshots highlighting robot-target interactions. **Bottom-right:** Gantt chart of the execution timeline.

References

[1] A. Shukla and H. Karki, "Application of robotics in offshore oil and gas industry—a review part ii," *Robotics Auton. Syst.*, vol. 75, pp. 508–524, 2016.

[2] T. Bock, "Construction robotics," *Auton. Robots*, vol. 22, no. 3, pp. 201–209, 2007.

[3] P. Toth and D. Vigo, "An overview of vehicle routing problems," *The Vehicle Routing Problem*, pp. 1–26, 2002.

[4] 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 Sci. Syst.*, 2015.

[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 Sci. Syst.*, 2017, pp. 696–700.

[6] C. Baier and J.-P. Katoen, *Principles of Model Checking*. MIT Press, 2008.

[7] Y. Kantaros and M. M. Zavlanos, "Stylus*: A temporal logic optimal control synthesis algorithm for large-scale multi-robot systems," *Int. J. Robot. Res.*, vol. 39, no. 7, pp. 812–836, 2020.

[8] M. Guo and M. M. Zavlanos, "Multirobot data gathering under buffer constraints and intermittent communication," *IEEE Trans. Robot.*, vol. 34, no. 4, pp. 1082–1097, 2018.

[9] C. Robin and S. Lacroix, "Multi-robot target detection and tracking: taxonomy and survey," *Auton. Robots*, vol. 40, pp. 729–760, 2016.

[10] X. Luo and M. M. Zavlanos, "Temporal logic task allocation in heterogeneous multi-robot systems," *IEEE Trans. Robot.*, vol. 38, no. 6, pp. 3602–3621, 2022.

[11] Y. E. Sahin, P. Nilsson, and N. Ozay, "Multirobot coordination with counting temporal logics," *IEEE Trans. Robot.*, vol. 36, no. 4, pp. 1189–1206, 2019.

[12] 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.

[13] P. Schillinger, M. Bürger, and D. V. Dimarogonas, "Simultaneous task allocation and planning for temporal logic goals in heterogeneous multi-robot systems," *Int. J. Robot. Res.*, vol. 37, no. 7, pp. 818–838, 2018.

[14] M. Guo and D. V. Dimarogonas, "Multi-agent plan reconfiguration under local ltl specifications," *Int. J. Robot. Res.*, vol. 34, no. 2, pp. 218–235, 2015.

[15] Y. Zhang, S. Kalluraya, G. J. Pappas, and Y. Kantaros, "Reactive planning for teams of heterogeneous robots with dynamic collaborative temporal logic missions," in *IEEE Conf. Decis. Control*, 2024, pp. 1599–1606.

[16] S. Kalluraya, G. J. Pappas, and Y. Kantaros, "Multi-robot mission planning in dynamic semantic environments," in *IEEE Int. Conf. Robot. Autom.*, 2023, pp. 1630–1637.

[17] L. Lindemann, M. Cleaveland, G. Shim, and G. J. Pappas, "Safe planning in dynamic environments using conformal prediction," *IEEE Robot. Autom. Lett.*, vol. 8, no. 8, pp. 5116–5123, 2023.

[18] S. Tonkens, S. Sun, R. Yu, and S. Herbert, "Scalable safe long-horizon planning in dynamic environments leveraging conformal prediction and temporal correlations," in *IEEE Int. Conf. Robot. Autom.*, 2023.

[19] X. Yu, Y. Zhao, X. Yin, and L. Lindemann, "Signal temporal logic control synthesis among uncontrollable dynamic agents with conformal prediction," *arXiv:2312.04242*, 2023.

[20] C. Belta, B. Yordanov, and E. A. Gol, *Formal Methods for Discrete-Time Dynamical Systems*. Springer, 2017, vol. 15.

[21] S. Hochreiter, "Long short-term memory," *Neural Comput.*, 1997.

[22] L. R. Medsker, L. Jain, *et al.*, "Recurrent neural networks," *Design and Applications*, vol. 5, no. 64-67, p. 2, 2001.

[23] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," *arXiv preprint arXiv:1409.1259*, 2014.

[24] A. N. Angelopoulos and S. Bates, "A gentle introduction to conformal prediction and distribution-free uncertainty quantification," *arXiv:2107.07511*, 2021.

[25] 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.

[26] L. Kocsis and C. Szepesvári, "Bandit based monte-carlo planning," in *Eur. Conf. Mach. Learn.*, 2006, pp. 282–293.