Tackling the Curse of Dimensionality in Large-scale Multi-agent LTL Task Planning via Poset Product

Zesen Liu, Meng Guo and Zhongkui Li

Abstract—Linear Temporal Logic (LTL) formulas have been used to describe complex tasks for multi-agent systems, with both spatial and temporal constraints. However, since the planning complexity grows exponentially with the number of agents and the length of the task formula, existing applications are mostly limited to small artificial cases. To address this issue, a new planning algorithm is proposed for task formulas specified as sc-LTL formulas. It avoids two common bottlenecks in the model-checking-based planning methods, i.e., (i) the direct translation of the complete task formula to the associated Büchi automaton; and (ii) the synchronized product between the Büchi automaton and the transition models of all agents. In particular, each conjuncted sub-formula is first converted to the associated R-posets as an abstraction of the temporal dependencies among the subtasks. Then, an efficient algorithm is proposed to compute the product of these R-posets, which retains their dependencies and resolves potential conflicts. Furthermore, the proposed approach is applied to dynamic scenes where new tasks are generated online. It is capable of deriving the first valid plan with a polynomial time and memory complexity w.r.t. the system size and the formula length. Our method can plan for task formulas with a length of more than 60 and a system with more than 35 agents, while most existing methods fail at the formula length of 20. The proposed method is validated on large fleets of service robots in both simulation and hardware experiments.

I. Introduction

Multi-agent systems can be extremely efficient when working concurrently and collaboratively. Linear Temporal Logic (LTL) in [1] has been widely used to specify complex tasks due to its formal syntax and rich expressiveness in various applications, e.g., transportation [2], maintenance [3] and services [4]. The standard framework is based on the model-checking algorithm [1]: First, the task formulas are converted to a Deterministic Robin Automaton (DRA) or Nondeterministic Büchi Automaton (NBA), via off-the-shelf tools such as SPIN [5] and LTL2BA [6]. Second, a product automaton is created between the automaton of formula and the models of all agents, such as weighted finite transition systems (wFTS) [1], Markov decision processes [7] or Petri nets [8]. Last, certain graph search or optimization procedures are applied to find a valid and accepting plan within the product automaton, such as nested-Dijkstra search [4], integer programs [9], auction [10] or sampling-based [11].
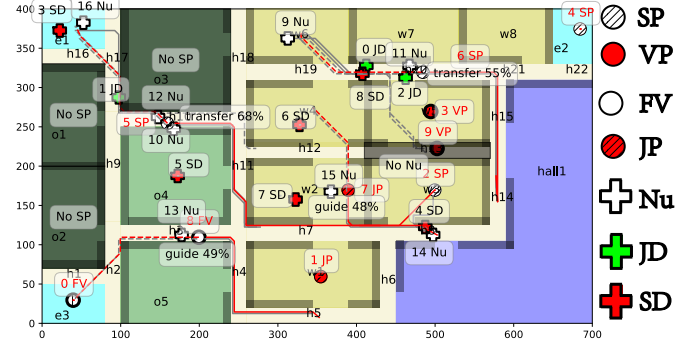
Fig. 1. Snapshots of the simulated hospital environment, where a fleet of 17 robots collaboratively guide visitors, disinfect rooms, and transport patients.

Thus, the fundamental step of the aforementioned methods is to translate the task formula into the associated automaton. This translation may lead a double exponential size in terms of the formula length as shown in [6]. The only exceptions are GR(1) formulas [12], of which the associated automaton can be derived in polynomial time but only for limited cases. In fact, for general LTL formulas with length more than 25, it takes more than 2 hours and $13GB$ memory to compute the associated NBA via LTL2BA. Although recent methods have greatly reduced the planning complexity in other aspects, the simulated examples are still limited by this translation process. For instance, the sampling-based method [11] avoids creating the complete product automaton via sampling, of which the largest simulated case has 400 agents and the task formula has only length 14. The planning method [4] decomposes the resulting task automaton into independent sub-tasks for parallel execution. The simulated case scales up to 6 robots and a task formula of length 18. Moreover, other existing works such as [13], [14], [15], [16] mostly have a formula length around $6 - 10$. This limitation hinders greatly its application to more complex robotic missions.

Furthermore, this drawback becomes even more apparent within dynamic scenes where contingent tasks, specified as LTL formulas are triggered by external observations and released to the whole team online. In such cases, most existing approaches compute the automaton associated with the new task, of which the synchronized product with the current automaton is derived, see [15], [16]. Thus, the size of the resulting automaton equals to the product of *all* automata associated with the contingent tasks, which is clearly a combinatorial blow-up. Consequently, the amount of contingent tasks that can be handled by the aforementioned approaches is limited to hand-picked examples.

To overcome this curse of dimensionality, we propose a

new paradigm that is fundamentally different from the model-checking-based method. First, for an LTL formula that consists of numerous sub-formulas of smaller size, we calculate the R-posets of each sub-formula as a set of subtasks and their partial temporal constraints. Then, an efficient algorithm is proposed to compute the product of R-posets associated with each sub-formula. The resulting product of R-posets is demonstrated to be complete in the sense that it retains all subtasks from each R-poset along with their partial orderings and resolves potential conflicts. Given this product, a task assignment algorithm called the time-bound contract net (TBCN) is proposed to assign collaborative subtasks to the agents, subject to the partial ordering constraints. Last but not least, the same algorithm is applied online to dynamic scenes where contingent tasks are triggered and released by online observations. It is shown formally that the proposed method has a polynomial time and memory complexity to derive the first valid plan w.r.t. the system size and formula length. Extensive large-scale simulations and experiments are conducted for a hospital environment where service robots react to online requests such as collaborative transportation, cleaning and monitoring.

Main contribution of this work is three-fold: (i) a systematic method is proposed to tackle task formulas with length more than 30, which overcomes the limitation of existing translation tools that can only process formulas of length less than 20 in reasonable time; (ii) an efficient algorithm is proposed to decompose and integrate contingent tasks that are released online, which not only avoids a complete re-computation of the task automaton but also ensures a polynomial complexity to derive the first valid plan; (iii) the proposed task assignment algorithm, TBCN, is fully compatible with both static and dynamic scenarios with interactive objects.

II. PRELIMINARIES

A. Linear Temporal Logic and Büchi Automaton

The basic ingredients of Linear Temporal Logic (LTL) formulas are a set of atomic propositions AP in addition to several Boolean and temporal operators. Atomic propositions are Boolean variables that can be either true or false. The syntax of LTL, as given in [1], is defined as: $\varphi \triangleq \top \mid p \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid \bigcirc \varphi \mid \varphi_1 \mathsf{U} \varphi_2$, where $\top \triangleq \texttt{True}$, $p \in AP$, $\bigcirc$ (*next*), U (*until*) and $\bot \triangleq \neg\top$. The derivations of other operators, such as $\square$ (*always*), $\Diamond$ (*eventually*), $\Rightarrow$ (*implication*) are omitted here for brevity. A complete description of the semantics and syntax of LTL can be found in [1]. Moreover, for a given LTL formula φ, there exists a Nondeterministic Büchi Automaton (NBA) as follows:

Definition 1. A NBA $\mathcal{A} \triangleq (S, \Sigma, \delta, (S_0, S_F))$ is a 4-tuple, where S are the states; $\Sigma = AP$; $\delta : S \times \Sigma \to 2^S$ are transition relations; $S_0, S_F \subseteq S$ are initial and accepting states. ∎

An infinite word w over the alphabet 2^{AP} is defined as an infinite sequence $W = \sigma_1\sigma_2\cdots, \sigma_i \in 2^{AP}$. The language of φ is defined as the set of words that satisfy φ, namely, $\mathcal{L}(\varphi) = Words(\varphi) = \{W \mid W \models \varphi\}$ and $\models$ is the satisfaction relation. Additionally, the resulting *run* of w within $\mathcal{A}$ is an infinite sequence $\rho = s_0 s_1 s_2 \cdots$ such that $s_0 \in S_0$, and $s_i \in S$, $s_{i+1} \in$

$\delta(s_i, \sigma_i)$ hold for all index $i \geqslant 0$. A run is called *accepting* if it holds that $\inf(\rho) \cap S_F \neq \varnothing$, where $\inf(\rho)$ is the set of states that appear in ρ infinitely often. A special class of LTL formula called *syntactically co-safe* formulas (sc-LTL) [1], [17], which can be satisfied by a set of finite sequence of words. They only contain the temporal operators $\bigcirc$, U and $\Diamond$ and are written in positive normal form where the negation operator $\neg$ is not allowed before the temporal operators.

B. Partially Ordered Set

As proposed in our previous work [3], a relaxed partially ordered set (R-poset) over an NBA $\mathcal{B}_\varphi$ is a 3-tuple: $P_\varphi = (\Omega_\varphi, \preceq_\varphi, \neq_\varphi)$, where Ω_φ is the set of subtasks and $\preceq_\varphi, \neq_\varphi$ are the partial relations defined as follows.

Definition 2 (Subtasks)**.** [3] The set of *subtasks* is given by $\Omega_\varphi = \{(\ell, \sigma_\ell, \sigma_\ell^s), \forall \ell = 0, \dots, L\}$, where ℓ is the index of subtask ω_ℓ; $\sigma_\ell \subseteq \Sigma$ are the transition labels; $\sigma_\ell^s \subseteq \Sigma$ are the self-loop labels from Def. 1 . ∎

Definition 3 (Partial Relations)**.** [3] Given two subtasks in $\omega_h, \omega_\ell \in \Omega_\varphi$, the following two types of relations are defined: (i) "less equal": $\preceq_\varphi \subseteq \Omega_\varphi \times \Omega_\varphi$. If $(\omega_h, \omega_\ell) \in \preceq_\varphi$ or equivalently $\omega_h \preceq_\varphi \omega_\ell$, then ω_ℓ can only be *started* after ω_h is started; (ii) "opposed": $\neq_\varphi \subseteq 2^{\Omega_\varphi}$. If $\{\omega_h, \cdots, \omega_\ell\} \in \neq_\varphi$ or equivalently $\omega_h \neq_\varphi \cdots \omega_\ell$, subtasks in $\omega_h, \cdots, \omega_\ell$ cannot all be executed simultaneously. ∎

III. PROBLEM FORMULATION

A. Collaborative Multi-agent Systems

Consider a workspace $W \subset \mathbb{R}^2$ with M regions of interest denoted as $\mathcal{W} \triangleq \{W_1, \cdots, W_M\}$, where $W_m \in W$. Furthermore, there is a group of agents denoted by $\mathcal{N} \triangleq \{1, \cdots, N\}$ with different types $\boldsymbol{L} \triangleq \{1, \cdots, L\}$. More specifically, each agent $n \in \mathcal{N}$ belongs to only one type $l = M_{type}(n)$, where $M_{type} : \mathcal{N} \to \boldsymbol{L}$. Each type of agents $l \in \boldsymbol{L}$ is capable of providing a set of different actions denoted by $\mathcal{A}_l$. The set of all actions is denoted as $\mathcal{A}_a = \bigcup_{l \in L} \mathcal{A}_l = \{a_1, \cdots, a_{n_c}\}$. Without loss of generality, the agents can navigate between each region via the same transition graph, i.e., $\mathcal{G} = (\mathcal{W}, \to_\mathcal{G}, d_\mathcal{G})$, where $\to \subseteq \mathcal{W} \times \mathcal{W}$ represents the allowed transitions; and $d_\mathcal{G} : \to_\mathcal{G} \to \mathbb{R}_+$ maps each transition to its duration.

Moreover, there is a set of interactive objects $\mathcal{O} \triangleq \{o_1, \cdots, o_U\}$ with several types $\mathcal{T} \triangleq \{T_1, \cdots, T_H\}$ scattered across the workspace W. These objects are *interactive* and can be transported by the agents from one region to another. An interactive object $o_u \in \mathcal{O}$ is described by a three-tuple:

$$o_u \triangleq (T_{h_u}, t_u, W_u^p), \tag{1}$$

where $T_{h_u} \in \mathcal{T}$ is the type of object; $t_u \in R^+$ is the time when o_u appears in workspace W; $W_u^p : R^+ \to \mathcal{W}$ is a function that $W_u^p(t)$ returns the region of o_u at time $t \geqslant t_u$; and $W_u^p(t_u) \subseteq W$ is the initial region. Additionally, new objects appear in W over time and are then added to the set $\mathcal{O}$. With a slightly abusive of notation, we denote the set of initial objects $\mathcal{O}_{in}$ that already exist in W and the set of online objects $\mathcal{O}_{on}$ that are added during execution, i.e., $\mathcal{O} = \mathcal{O}_{in} \cup \mathcal{O}_{on}$.
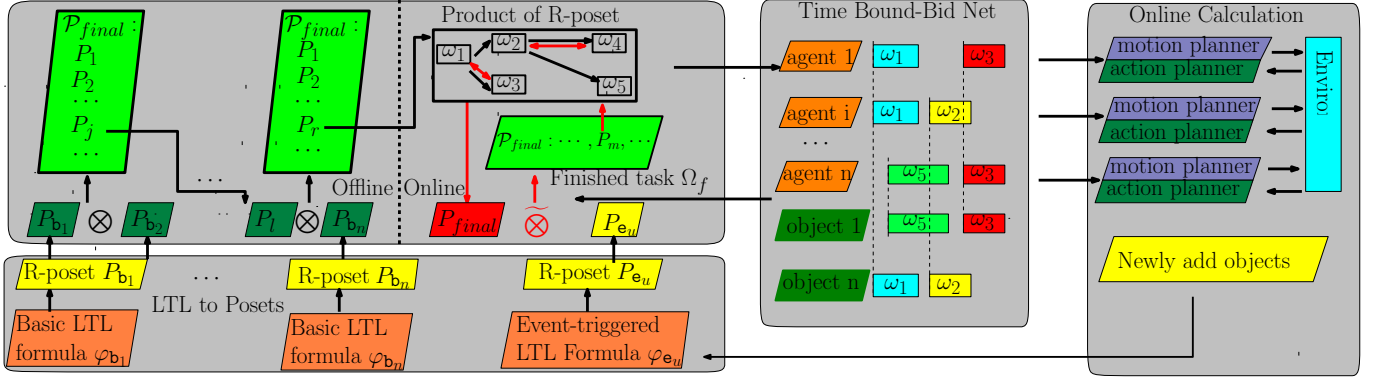
Fig. 2. Framework of the proposed method. We firstly change each sub-formula into sets of R-posets with the method in Sec. IV-A . Then, we repeatedly use Alg. 1 to generate the final R-poset which satisfies all formulas of Φ_{b}. Thirdly, a Time-Bound Bid Net method 2 is given to determine the task sequence of each agent and object. Finally, we generate the event-triggered formula and make an online product of R-posets for online adaption if a new object is found.

To interact with the objects, the agents can provide a series of collaborative behaviors $\mathcal{C} \triangleq \{C_1, \ldots, C_K\}$. A *collaborative behavior* $C_k \in \mathcal{C}$ is a tuple defined as follows:

$$C_k \triangleq (o_{u_k}, \{(a_i, n_i),\ i \in \boldsymbol{L}_k\}) \qquad (2)$$

where $o_{u_k} \in \mathcal{O} \cup \{\varnothing\}$ is the interactive object if any; $a_i \in \mathcal{A}_a$ is the set of cooperative actions required; $0 < n_i \leqslant N$ is the number of agents to provide the action a_i; and $\boldsymbol{L}_k$ is the set of action indices associated with the behavior C_k. Also, d_k denotes the execution time of C_k.

Remark 1. A behavior can only be executed if the required object is at the desired region. Since objects can only be transported by the agents, it is essential for the planning process to find the correct order of these transporation behaviors. Related works [2], [14] build a transition system to model the interaction between objects and agents, the size of which grows exponentially with the number of agents and objects. ∎

Remark 2. The behavior $\mathcal{C}$ above is similar to the atomic unit of capability temporal logic (CaTL) in [9], [18], which only models static scenarios without interactive objects. ∎

B. Task Specifications

Consider the following two types of atomic propositions: (i) p_m^l is *true* when any agent of type $l \in \boldsymbol{L}$ is in region $W_m \in \mathcal{W}$; p_m^r is *true* when any object of type $r \in \mathcal{T}$ is in region $W_m \in \mathcal{W}$; Let $\mathbf{p} \triangleq \{p_m^l, \forall W_m \in \mathcal{W}, l \in L\} \bigcup \{p_m^r, \forall W_m \in \mathcal{W}, r \in \mathcal{R}\}$. (ii) $(C_k)_{k_1,k_2}^{u_k}$ is *true* when the collaborative behavior C_k in (2) is executed with object o_{u_k}, starting from region W_{k_1} and ending in region W_{k_2}. Let $\mathbf{c} \triangleq \{(C_k)_{k_1,k_2}^{u_k}, \forall C_k \in \mathcal{C}, o_{u_k} \in \mathcal{O}, \forall W_{k_1}, W_{k_2} \in \mathcal{W}\}$. Given these propositions, the team-wise task specification is specified as a sc-LTL formula over $\{\mathbf{p}, \mathbf{c}\}$:

$$\varphi = (\bigwedge_{i \in \mathcal{I}} \varphi_{\mathrm{b}_i}) \wedge (\bigwedge_{o_u \in \mathcal{O}_{on}} \varphi_{\mathrm{e}_u}), \qquad (3)$$

where $\{\varphi_{\mathrm{b}_i}, i = 1, \cdots, I\}, \{\varphi_{\mathrm{e}_u}, o_u \in \mathcal{O}_{on}\}$ are two sets of sc-LTL formulas over $\{\mathbf{p}, \mathbf{c}\}$. The φ_{b_i} is specified in advance while φ_{e_u} is generated online when a new object o_u is added to $\mathcal{O}_{on}$. The detailed syntax of sc-LTL is introduced in Sec. II-A.

To satisfy the LTL formula φ, the complete action sequence of all agents is defined as:

$$\mathcal{J} = (J_1, J_2, \cdots, J_N), \qquad (4)$$

where $J_n \in \mathcal{J}$ is the sequence of $(t_k, \mathcal{C}_k, a_k)$, which means that agent n executes behavior $\mathcal{C}_k$ by providing the collaborative service $a_k \in \mathcal{A}_a$ at time t_k. In turn, the sequences of actions for an interactive object is defined as:

$$\mathcal{J}^o = (J_1^o, J_2^o, \cdots, J_U^o), \qquad (5)$$

where $J_u^o = (t_k, \mathcal{C}_k) \cdots$ is the sequence of tasks associated with object $o_u \in \mathcal{O}$. The task pair $(t_k, \mathcal{C}_k)$ is added to J_u^o if object o_u joins behavior $\mathcal{C}_k$ at time t_k. Assume that the duration of formula $\varphi_{\mathrm{b}_i}, \varphi_{\mathrm{e}_i}$ from being published to being satisfied is given by D_i, the average efficiency is defined by $\eta \triangleq \frac{\sum_{C_k \in J} |\boldsymbol{L}_k| d_k}{D_i}$ as the average duration of all subtasks.

C. Problem Statement

Problem 1. Given the sc-LTL formula φ defined in (3), synthesize and update the motion and action sequence of agents $\mathcal{J}$ and objects $\mathcal{J}_o$ for each agent $n \in \mathcal{N}$ to satisfy φ and maximize execution efficient η. ∎

Although maximizing the task efficiency of a multi-agent system is a classical problem, the combination of interactive objects, long formulas and contingent tasks imposes new challenges in terms of exponential complexity [6], [14] and online adaptation [15].

IV. PROPOSED SOLUTION

As shown in Fig. 2, the proposed solution consists of four main components: i) LTL to R-posets, where the R-posets associated with each known formula in $\varphi_{\mathrm{b}_i} \in \Phi_{\mathrm{b}}$ is generated in parallel; ii) Product of R-posets, where the product of existing R-posets is computed incrementally; iii) Task assignment, where subtasks are assigned to the agents given the temporal and spatial constraints specified in the R-posets; iv) Online adaptation, where the product of R-posets is updated given the contingent tasks, and the assignment is adjusted accordingly.

A. LTL to R-posets

To avoid the exponential complexity when translating the formula φ into NBA, the sub-formulas $\varphi_{b_i} \in \Phi_b$ are converted first to R-posets by the method $\texttt{compute\_poset}(\cdot)$ proposed in our earlier work [3]. Particularly, each sub-formula $\varphi_{b_i} \in \Phi_b$ is converted into a set of R-posets, denoted by $\mathcal{P}_{b_i} = \{P_j^{b_i}\}$, where $P_j^{b_i} = (\Omega_j, \preceq_j, \neq_j)$ follows the notations from Def. 3; $\Omega_j = (\ell, \sigma_\ell, \sigma_\ell^s)$ is the set of subtasks each of which consists of the index ℓ, the subtask labels σ_ℓ and the self-loop labels σ_ℓ^s as defined in Def. 2. It is proven that a word is *accepting* if it satisfies all the constraints imposed by $\mathcal{P}_{b_i}$.

Definition 4 (Language of R-poset). [3] Given a word $w = \sigma_1' \sigma_2' \cdots$ satisfying R-poset $P = (\Omega, \preceq, \neq)$, denoted as $w \models P$, it holds that: i) given $\omega_{i_1} = (i_1, \sigma_{i_1}, \sigma_{i_1}^s) \in \Omega$, there exist j_1 with $\sigma_{i_1} \subseteq \sigma_{j_1}'$ and $\sigma_{i_1}^s \cap \sigma_{m_1}' = \varnothing, \forall m_1 < j_1$; ii) $\forall (\omega_{i_1}, \omega_{i_2}) \in \preceq, \exists j_1 \leqslant j_2, \sigma_{i_2} \subseteq \sigma_{j_2}', \forall m_2 < j_2, \sigma_{j_2}^s \subseteq \sigma_{m_2}'$; iii) $\forall (\omega_{i_1}, \cdots, \omega_{i_n}) \in \neq, \exists \ell \leqslant n, \sigma_{i_\ell} \nsubseteq \sigma_{j_1}'$. Language of R-poset P is the set of all word w that satisfies P, denoted by $\mathcal{L}(P) \triangleq \{w | w \models P\}$. ∎

Assuming that $\mathcal{P}_{b_i} = \{P_1^{b_i}, P_2^{b_i}, \cdots\}$ is the set of all possible R-posets, it is shown in Lemma 3 of our previous work [3] that: (i) $\mathcal{L}(P_j^{b_i}) \subseteq \mathcal{L}(\mathcal{B}_{b_i}), P_j^{b_i} \in \mathcal{P}_\varphi^{b_i}$; (ii) $\bigcup_{P_i^{b_i} \in \mathcal{P}_\varphi^{b_i}} \mathcal{L}(P_i) = \mathcal{L}(\mathcal{B}_{b_i})$. In other words, the R-posets contain the necessary information for subsequent steps.

B. Product of R-posets

To generate a complete R-poset for the whole task Φ_b, we define the product of R-posets as follows:

Definition 5 (Product of R-posets). The product of two posets P_1, P_2 is defined as a set of R-posets $\mathcal{P}^r \triangleq \{P_1', P_2', \cdots\}$, denoted as $\mathcal{P}^r = P_1 \bigotimes P_2$ where P_i' satisfies two conditions: (i) $w \in \mathcal{L}(P_1) \cap \mathcal{L}(P_2), \forall w \in \mathcal{L}(P_i'), \forall P_i' \in \mathcal{P}^r$; (ii) $w \in \bigcup_{P_i' \in \mathcal{P}^r} \mathcal{L}(P'), \forall w \in \mathcal{L}(P_1) \cap \mathcal{L}(P_2)$. ∎

Thus, to get the R-poset of formula $\varphi_1 \wedge \varphi_2$, there is no need to generate the NBA of $\varphi_1 \wedge \varphi_2$. Instead, we only generate the NBA of φ_1 and φ_2 to find the R-poset P_1 of φ_1 and P_2 of φ_2. Then, we generate a set of R-posets $\mathcal{P}^r$ by P_1, P_2, which satisfies the two conditions in Def.5. Here we present an anytime algorithm $\texttt{Poset\_Prod}(\cdot)$ as in Alg. 1. It generates the product of two R-posets $P_1 = (\Omega_1, \preceq_1, \neq_1), P_2 = (\Omega_2, \preceq_2, \neq_2)$ in the following two steps.

(i) **Composition of substasks**. In this step, we generate all possible combinations of subtasks which can satisfy both Ω_1 and Ω_2 in Line 1-7. Namely, a set of subtasks $\Omega' = \{\omega_1', \cdots, \omega_{n'}'\}$ satisfies Ω_1 if for each $\omega_i^1 = (i, \sigma_i^1, \sigma_i^{s1}) \in \Omega_1$, there is a subtask $\omega_j' = (j, \sigma_j', \sigma_j^{s'}) \in \Omega'$ satisfying $\sigma_i^1 \subseteq \sigma_j'$ and $\sigma_i^{s1} \subseteq \sigma_j^{s'}$, or $\omega_j' \models \omega_i^1$ for brevity. Since $\Omega' = \Omega_1$ in Line 1, Ω' clearly satisfies Ω_1 with $\forall \omega_i^1 \in \Omega_1, \omega_i' \models \omega_i^1$. Then, a mapping $M_\Omega : \Omega_2 \to \Omega'$ is defined to store the satisfying relationship from Ω_2 to Ω', and $\mathcal{D}(M_\Omega) \in \Omega_2$ is the domain of M_Ω, $\mathcal{R}(M_\Omega)$ is the range of M_Ω. As all relations are unknown, M_Ω, $\mathcal{D}(M_\Omega)$ and $\mathcal{R}(M_\Omega)$ are initially set to empty. Namely, if $\omega_j' \models \omega_i^2$, we will store $M_\Omega(\omega_i^2) = \omega_j'$ and $M_\Omega^{-1}(\omega_j') = \omega_i^2$, and add ω_i^2 into $\mathcal{D}(M_\Omega)$, ω_j' into $R(M_\Omega)$, where M_Ω^{-1} is the inverse function of M_Ω. Then,

---

Algorithm 1: $\texttt{Poset\_Prod}(\cdot)$: Calculate the product between two R-posets

Input : R-poset P_1, R-poset P_2, Time budget t_b
Output: Product R-poset set $\mathcal{P}_{final}$.

1   $\mathcal{P}_{final} = \{\}, que = [(\Omega = \Omega_1, M_\Omega)]$
2   **while** $|que| > 0$ *and* $t < t_b$ **do**
3      $\Omega', M_\Omega' = que.pop()$
4   **for** $\omega_i^2 \in \Omega_2 / R(M_\Omega')$ **do**
5   **for** $\omega_j^1 \in \Omega_1 / D(M_\Omega'), \omega_i^2 \models \omega_j^1$ *or* $\omega_j^1 \models \omega_i^2$ **do**
6   Create $\hat{\Omega}', \hat{M}_\Omega'$ by (6), add them to que
7   Create $\hat{\Omega}', \hat{M}_\Omega'$ by (7), add them to que
8   **for** $(\Omega, M_\Omega) \in que$ with $|D(M_\Omega)| = |\Omega_2|$ **do**
9   Remove (Ω, M_Ω), $\preceq = \preceq_{\varphi_1} \cup M_\Omega(\preceq_{\varphi_2})$
10   Update Ω by (9),(10)
11   Find potential ordering. Update $\preceq, \Omega$
12   **if** *no conflicts between* $\Omega, \preceq$ **then**
13   calculate $\neq$, add $\{\Omega, \preceq, \neq\}$ into $\mathcal{P}_{final}$
14   update t, **if** $t < t_b$ **then return** $\mathcal{P}_{final}$

15 **return** $\mathcal{P}_{final}$

---

in Lines 3-7, we use a depth-first-search (DFS) to construct all possible sets of subtasks $\hat{\Omega}'$ and corresponding $\hat{M}_\Omega$. In Line 4-5, we check all possible combinations of unrecorded subtask $(\omega_j^1, \omega_i^2), \omega_j^1 \in \Omega_1 / R(M_\Omega'), \omega_i^2 \in \Omega_2 / D(M_\Omega')$ whether $\omega_j^1 \models \omega_i^2$ or $\omega_i^2 \models \omega_j^1$ holds. If so, we can create a new set of subtasks $\hat{\Omega}'$ and a new mapping function $\hat{M}_\Omega'$ based on Ω', M_Ω' in Line 6 as follows:

$$\hat{\Omega}' = \Omega', \hat{\omega}_j' = (j, \sigma_j^1 \cup \sigma_i^2, \sigma_j^{s1} \cup \sigma_i^{s2}),$$
$$\hat{M}_\Omega' = M_\Omega', \hat{M}_\Omega'(\omega_i^2) = \omega_j', \tag{6}$$

which means the subtask ω_j' in Ω' can be executed to satisfy ω_j^1, ω_i^2. Moreover, for each subtasks $\omega_i^2 \in \Omega_2 / R(M_\Omega')$, we can always create a set of subtasks $\hat{\Omega}'$ and the corresponding mapping function $\hat{M}_\Omega'$ in Line 7 by appending ω_i^2 into $\hat{\Omega}'$ as $\hat{\omega}_j'$ such that $\hat{\omega}_j' \models \omega_i^2$ holds, i.e.,

$$\hat{\Omega}' = \Omega', j = |\Omega'| + 1, \hat{\omega}_j' = \omega_i^2;$$
$$\hat{M}_\Omega' = M_\Omega', \hat{M}_\Omega'(\omega_i^2) = \omega_j', \tag{7}$$

which means the subtask ω_j' can be executed to satisfy ω_i^2.

This step ends if the time budget t_b or the search sequence que exhausted as in Line 2. Once $|D(M_\Omega)| = |\Omega_2|$ as in Line 8, it means the Ω' satisfying Ω_1, Ω_2 is already found. In this case, the next step is triggered.

(ii) **Relation calculation**. In Lines 8 - 13, given the set of subtasks Ω and the mapping function M_Ω, we calculate the partial relations among them and build a product R-poset P. Firstly, in Line 9, we construct the "less-equal" constraint $\preceq$ as follows:

$$\preceq = \preceq_1 \cup \{(M_\Omega(\omega_{\ell_1}^2), M_\Omega(\omega_{\ell_2}^2)) | (\omega_{\ell_1}^2, \omega_{\ell_2}^2) \in \preceq_2\}, \tag{8}$$

which inherits the both less equal relations $\preceq_1$ in P_1 and $\preceq_2$ in P_2. Then, we update Ω in Line 10 to consider the constraints imposed by the self-loop labels in other subtasks.

Specifically, if a new relation (ω_i, ω_j) is added to $\preceq$ by $(M_\Omega^{-1}(\omega_i^2), M_\Omega^{-1}(\omega_j^2)) \in \preceq_2$ while $(\omega_i^1, \omega_j^1) \notin \preceq_1$ holds, ω_i is required to be executed before ω_j although (ω_i^1, ω_j^1) does not belong to $\preceq_1$ of P_1. In this case, σ_i and σ_i^s are updated to guarantee the satisfaction of the self-loop labels σ_j^s before executing σ_j. For each subtask ω_i, the sets of newly-added suf-subtasks from $\preceq_1, \preceq_2$ are defined as S_p^1, S_p^2, i.e.,

$$S_p^1 = \{\omega_j | (\omega_i, \omega_j) \in \preceq, (\omega_i^1, \omega_j^1) \notin \preceq_1\},$$
$$S_p^2 = \{M_\Omega^{-1}(\omega_j) | (\omega_i, \omega_j) \in \preceq, (M_\Omega^{-1}(\omega_i^2), M_\Omega^{-1}(\omega_j^2)) \notin \preceq_2\}, \quad (9)$$

in which are the subtasks that should be executed after ω_i although $\preceq_1$ or $\preceq_2$ does not require. Thus, the action labels σ_i and self-loop labels σ_i^s in $\omega_i = (i, \sigma_i, \sigma_i^s)$ are updated accordingly as follows:

$$\hat{\sigma} = \bigcup_{\omega_\ell^1 \in S_p^1} \sigma_\ell^{s1} \cup \bigcup_{\omega_\ell^2 \in S_p^2} \sigma_\ell^{s2}, \sigma_i^s = \hat{\sigma} \cup \sigma_i^s, \sigma_i = \hat{\sigma} \cup \sigma_i, \quad (10)$$

in which σ_i^s and σ_i should be executed under the additional labels $\hat{\sigma}$ thus the self-loop labels of S_p^1, S_p^2 are satisfied. In Line 11, we find the potential ordering by checking whether a subtask σ_i^s is in conflicts with another subtask σ_j while $(\omega_i, \omega_j) \notin \preceq$. If so, an additional ordering (ω_i, ω_j) will be added to $\preceq$ and Ω is updated following (9) and (10). Regarding the set of subtasks Ω that have no conflicts in $\preceq$, we can generate its "not-equal" relations $\neq$ by a simple combination as $\neq_1 \cup \left\{\{M_\Omega(\omega_{\ell_i})\} | \{\omega_{\ell_i}\} \in \neq_2\right\}$. Lastly, the resulting poset $P_i = (\Omega, \preceq, \neq)$ is added to $\mathcal{P}_{final}$.

As shown in Fig. 2, the above procedure is applied recursively for each pair of R-posets to compute the complete set of R-posets for formula Φ_b. Namely, the first product is computed as $\mathcal{P}_{final} = P_{b_1} \otimes P_{b_2}$ in the first round. Then, we fetch the best R-poset P_{final} in $\mathcal{P}_{final}$ according to the efficiency measure proposed in our earlier work. It is multiplied with the next P_{b_i} as $\mathcal{P}_{final} = P_{final} \otimes P_{b_i}$. This procedure is repeated until time budget exhausted or all possible R-posets are found. Thus, the anytime property of calculating the R-posets of φ is ensured, so that we can quickly get a P_{final} satisfies all P_{b_i} and then continue to get more R-posets.

C. Task Assignment

The LTL formulas are converted into a R-poset $P_c = (\Omega_c, \preceq_c, \neq_c)$, where each subtask $\omega_i \in \Omega_c$ represents a collaborative behavior C_j. Thus, we can redefine the action sequence of each agent $J_n \in \mathcal{J}$ as $J_n = [(t_k, \omega_k, a_k), \cdots]$ and the action sequence of each object $J_u^o \in \mathcal{J}_o$ as $J_u^o = [(t_k, \omega_k, a_k), \cdots]$. in which we replace the cooperative behavior C_k with ω_k since $C_k \in \sigma_k$.

As summarized in Alg. 2, we propose a sub-optimal algorithm called Time Bound Contract Net (TBCN) for the task assignment. Compared with the classical Contract Net method [19], the main differences are: (i) the assigned subtasks should satisfy the temporal constraints in $\preceq_c, \neq_c$; (ii) the cooperative task should be fulfilled by multiple agents; and (iii) interactive objects should be considered as an additional constraints. To begin with, we initialize the set of unassigned subtasks as $\Omega_u = \Omega_c$, the set of assigned subtasks $\Omega_{as} = \{\}$, and the task sequence of agents $\mathcal{J}$ and objects $\mathcal{J}_o$ as Line 1.

Algorithm 2: `TW_ContractNet`$(\cdot)$: Online task assignment by time bound contract net.

Input : R-poset $P_c = (\Omega_c, \preceq_c, \neq_c)$
Output: Updated plans $\mathcal{J}, \mathcal{J}_o$

1   $\Omega_u = \Omega_c, \Omega_{as} = \{\}, \mathcal{J} = [], \mathcal{J}_o = []$.
2   **while** $|\Omega_u| > 0$ **do**
3   Get feasible subtask Ω_{fe} based on Ω_{as}, P_c by (11)
4   **for** $\omega_i \in \Omega_{fe}$ **do**
5   Get time bounds and generate $\mathcal{J}^i, \mathcal{J}_o^i$
6   Choose ω_i, update $\Omega_u, \Omega_{as}, \mathcal{J} = \mathcal{J}^i, \mathcal{J}_o = \mathcal{J}_o^i$
7   **return** $\mathcal{J}, \mathcal{J}_o$

For the ordering constraints $\preceq_c$, if $(\omega_j, \omega_i) \in \preceq_c$, assigning $(t_{k_j}, \omega_j, a_{k_j})$ to a task sequence $J_i = \cdots (t_{k_i}, \omega_i, a_{k_i})$ will violate such constraints. Thus, we add a mechanism to avoid creating such cases, i.e., only assign the set of feasible subtasks Ω_{fe} based on Ω_{as} in Line 3 :

$$\Omega_{fe} = \{\omega_i | \omega_i \in \Omega_u, \forall (\omega_j, \omega_i) \in \preceq_c, \omega_j \in \Omega_{as}\}, \quad (11)$$

in which the subtasks may lead to unfeasible action sequences being eliminated.

Then, in Line 5 , any constraint of a subtask $\omega_i \in \Omega_{fe}$ is considered as a time instance $t_1 \in \mathbb{R}^+$ which means such constraint can be satisfied after t_1. Without loss of generality, we assume $\sigma_i = \{(C_k)_{k_1, k_2}^{u_k}\}$ meaning that the agents need to execute the behavior C_k from region W_{k_1} to region W_{k_2} using object u_k. Here, we use three kinds of time bounds to manage the biding process: the global time bound t_i^ω, the object time bound $t_{u_k}^o$ and the set of local time bounds $\boldsymbol{T_s}$. The global time bound t_i^ω is the time instance that the ordering constraints $\preceq_c$ and conflict constraints $\neq_c$ will be satisfied if behavior $(C_k)_{k_1, k_2}^{u_k}$ is executed after t_i^ω. For the required object u_k, assuming its participated last task is $\mathcal{J}_{u_k}^o[-1] = (t_\ell, \omega_\ell)$, the object time bound $t_{u_k}^o$ should satisfy that:

$$W_u^p(t) = W_{k_1}, t \geqslant t_\ell + d_\ell, , \forall t \geqslant t_{u_k}^o, \quad (12)$$

which means the object u_k will be at the starting region W_{k_1} of the current behavior $(C_k)_{k_1, k_2}^{u_k}$ and ready for it after $t_{u_k}^o$. Additionally, we set $t_{u_k}^o = \infty$ if the object is not at W_{k_1} after the action sequence $\mathcal{J}_{u_k}^o$, and we set $t_{u_k}^o = 0$ if the behavior $(C_k)_{k_1, k_2}^{u_k}$ does not require object as $u_k = \varnothing$. The set of local time bound is defined as $\boldsymbol{T_s} = \{(\mathcal{A}_n, t_n^a)_{n \in \mathcal{N}}\}$, where $\mathcal{A}_n$ is the set of actions agent n can provide for behavior $(C_k)_{k_1, k_2}^{u_k}$, and t_n^a is the earliest time agent n can arrive region W_{k_1}. $(\mathcal{A}_n, t_n^a)$ means agent n can begin behavior $(C_k)_{k_1, k_2}^{u_k}$ after time t_n^a by providing one of action $a_\ell \in \mathcal{A}_n$. Using these time bounds, we can determine the agents and their providing actions and generate a new party assignment $\mathcal{J}^i, \mathcal{J}_o^i$ to minimize the ending time of subtask ω_i. The efficiency η of each assignment $\mathcal{J}^i, \mathcal{J}_o^i, \omega_i \in \Omega_u$ is calculated in Line 6, and the subtask with max efficiency will be chosen. Afterwards, the chosen subtask is removed from Ω_{un} and added to Ω_{as}. The action sequences $\mathcal{J}, \mathcal{J}_o$ are updated accordingly as $\mathcal{J} = \mathcal{J}^i, \mathcal{J}_o = \mathcal{J}_o^i$ for the next iteration.

Remark 3. As this algorithm can only assign the subtask in Ω of R-poset $P_{final} = (\Omega, \leq, \neq)$, the R-poset should follow an assumption: There exists an action sequence order under the ordering constraints $\leq$ and conflicts constraints $\neq$ that can transform and interact with the objects in order. That means the formula should not require a behavior to transfer an object at region W_1 before it has been moved there beforehand.

D. Online Adaption

As there are objects added to the workspace online, the agents are required to update their action sequences to satisfy the contingent formulas. The online adaptation has two important parts: the **adaptation of R-posets** and the **adaptation of task sequences**. The **adaptation of R-posets** is to update the R-poset P_{final} based on P_{final} and the R-poset P_{e_u} of contingent formula φ_{e_u}, instead of recomputing the R-poset associated with the whole formula from scratch. The definition of online product between on existing R-poset P_1 and contingent P_2 is given as:

Definition 6 (Online Product of R-posets). Given a finite word w_0, the online product of two R-posets P_1, P_2 is defined as a set of R-posets $\mathcal{P}^r = \{P'_1, P'_2, \cdots\}$, denoted as $\mathcal{P}^r = P_1 \widetilde{\otimes} P_2$ where P'_i satisfies two conditions: (i) if $w_0w \in \mathcal{L}(P_1), w \in \mathcal{L}(P_2)$, then $w_0w \in \bigcup_{P'_i \in \mathcal{P}^r} \mathcal{L}(P')$; (ii) if $w_0w \in \mathcal{L}(P'_i), P'_i \in \mathcal{P}^r$, then $w_ow \in \mathcal{L}(P_1), w \in \mathcal{L}(P_2)$. ∎

The difference between $\otimes$ and $\widetilde{\otimes}$ is that the finished word w_0 only effects the R-poset P_1 but not the R-poset P_2 in $\widetilde{\otimes}$. Thus, once a new formula φ_{e_u} is added, we will collect the set of finished behaviers $\Omega_f \subseteq \Omega_1$ from current word w_0, which cannot effect the R-poset P_{e_u} of φ_{e_u}. The method to compute the online product is similar to the Alg. 1, with two minor modifications. Namely, the set of finished subtasks Ω_f is added as an additional input and the condition of the *for* loop in Line 5 is changed to: **for** $\omega_i^1 \in \Omega_1/D(M')/\Omega_f$, with $\omega_i^2 \models \omega_j^1$ or $\omega_j^1 \models \omega_i^2$ holds. Since ω_i^1 is already accomplished before P_2 is proposed, there is no need to require another subtask to satisfy ω_i^1 again.

In the **adaption of task sequences**, the task sequences of agents and objects $\mathcal{J}, \mathcal{J}_o$ are updated given the updated final R-poset $P_{final} = (\Omega, \leq, \neq)$. First, we compute a set of essential conflict subtasks Ω_{ec}, defined as:

$$\Omega_{ec} = \{\omega_i | \omega_i \leq \omega_j, t_i \geq t_j, \forall \omega_i, \omega_j \in \Omega_1/\Omega_f\} \cup \\ \{\omega_i | \omega_i \neq \omega_j, t_j \leq t_i \leq t_j + d_j, \forall \omega_i, \omega_j \in \Omega_1/\Omega_f\}, \quad (13)$$

in which is the subtask conflicts the updated ordering constraints $\leq$ and conflict constraints $\neq$. Then, we compute the set of subtasks Ω_{conf} that should be removed from $\mathcal{J}, \mathcal{J}_o$:

$$\Omega_{conf} = \{\omega_j | \omega_i \leq \omega_j, \forall \omega_i \in \Omega_1/\Omega_f, \forall \omega_j \in \Omega_{ec}\} \cup \Omega_{ec}, \quad (14)$$

in which are the subtasks in Ω_{ec} and the subtasks whose pre-subtasks will be removed. Given Ω_{conf} above, we can use `TW_ContractNet(·)` to generate a new task sequence for each robot and object. After removing all the subtasks in Ω_{conf} from $\mathcal{J}, \mathcal{J}_o$. It is worth mentioning that the step of initialization in Line 1 is changed to: $\Omega_{as} = \Omega_1/\Omega_{conf}, \Omega_u = \Omega_c/\Omega_{as}$ and $\mathcal{J}, \mathcal{J}_o$ are the previous solution with Ω_{conf} been removed.

E. Correctness, Completeness and Complexity Analysis

Theorem 1 (Correctness). *Given two R-posets* $P_{\varphi_1} = (\Omega_{\varphi_1}, \leq_{\varphi_1}, \neq_{\varphi_1}), P_{\varphi_2} = (\Omega_{\varphi_2}, \leq_{\varphi_2}, \neq_{\varphi_2})$ *generated from* $\mathcal{B}_{\varphi_1}, \mathcal{B}_{\varphi_2}$, *we have* $\mathcal{L}(P_j) \subseteq \mathcal{L}(P_{\varphi_1}) \cap \mathcal{L}(P_{\varphi_2})$, *where* $P_j \in \mathcal{P}_{final}$, $\mathcal{P}_{final}$ *is calculated by Alg.1 .* ∎

Proof. If a *word* $w = \sigma'_1 \sigma'_2 \cdots$ satisfies $P_j = (\Omega_j, \leq_j, \neq_j), P_j \in \mathcal{P}_{final}$, it satisfies the three conditions mentioned in Def. 4. In first condition, due to Line 6-7 of Alg. 1, we can infer that for any $\omega_n^1 = (n, \sigma_n^1, \sigma_n^{s1}) \in \Omega_{\varphi_1}$, there exists $\omega_n = (n, \sigma_n, \sigma_n^s) \in \Omega_j$, with $\sigma_n^1 \subseteq \sigma_n, \sigma_n^{s1} \subseteq \sigma_n^s$. Thus, we have $\sigma_{i_1}^{s1} \subseteq \sigma_{i_1} \subseteq \sigma'_{\ell_1}$ and $\sigma_{i_1}^{s1} \cap \sigma'_{m_1} = \varnothing, \forall m_1 < \ell_1$, which indicates that w satisfies P_{φ_1} for condition 1. For the second condition, due to Line 9 of Alg. 1, we have $\leq_{\varphi_i} \subseteq \leq_j$. Thus, we can infer that w satisfies P_{φ_1} for the second condition: Any $(\omega_{i_1}^1, \omega_{i_2}^1) \in \leq_{\varphi_1}$, we have $(\omega_{i_1}, \omega_{i_2}) \in \leq_j$, thus $\exists \ell_1 \leq \ell_2, \sigma_{i_2}^1 \subseteq \sigma_{i_2} \subseteq \sigma'_{\ell_2}$, and $\forall m_2 < \ell_2, \sigma_{\ell_2}^{s1} \subseteq \sigma_{\ell_2}^s \subseteq \sigma'_{m_2}$. Additionally, for the last condition, as the word w satisfied the $\neq_j$ order of P_j. We have $\neq_{\varphi_1} \subseteq \neq_j$ due to Line 13. Thus, the word w also satisfied the third condition. In the end, we can conclude that w satisfies P_{φ_1}. □

Theorem 2 (Completeness). *Given two R-posets* $P_{\varphi_1}, P_{\varphi_2}$ *getting from* $\mathcal{B}_{\varphi_1}, \mathcal{B}_{\varphi_2}$, *with enough time budget, Alg.1 returns a set* $\mathcal{P}_{final}$ *consisting of all final product, and its language* $\mathcal{L}(\mathcal{P}_{final}) = \bigcup_{P_i \in \mathcal{P}_{final}} \mathcal{L}(P_i)$ *is equal to* $\mathcal{L}(\mathcal{P}_{final}) = \mathcal{L}(P_{\varphi_1}) \bigcap \mathcal{L}(P_{\varphi_2})$. ∎

Proof. Due to Theorem 1, it holds that $\mathcal{L}(\mathcal{P}_{final}) \subseteq \mathcal{L}(P_{\varphi_1}) \bigcap \mathcal{L}(P_{\varphi_2})$. Thus, we only need to show that $\mathcal{L}(P_{\varphi_1}) \bigcap \mathcal{L}(P_{\varphi_2}) \subseteq \mathcal{L}(\mathcal{P}_{final})$. Given a word $w = \sigma'_1 \sigma'_2 \cdots$ and $w \in \mathcal{L}(P_{\varphi_1}) \bigcap \mathcal{L}(P_{\varphi_2})$, w satisfies the first condition in Def. 4 for both P_1 and P_2 that: $\forall \omega_{i_1}^1 = (i_1, \sigma_{i_1}^1, \sigma_{i_1}^{s1}) \in \Omega_{\varphi_1}$, there exists σ'_{j_1} with $\sigma_{i_1}^1 \subseteq \sigma'_{j_1}$ and $\sigma_{j_1}^{s1} \subseteq \sigma'_{m_1}, \forall m_1 < j_1$; $\forall \omega_{i_2}^2 = (i_2, \sigma_{i_2}^2, \sigma_{i_2}^{s2}) \in \Omega_{\varphi_2}$, there exists σ'_{j_2} with $\sigma_{i_2}^2 \subseteq \sigma'_{j_2}$ and $\sigma_{j_2}^{s2} \subseteq \sigma'_{m_2}, \forall m_2 < j_2$. If $j_1 = j_2$, the step in Line 6 of Alg.1 will generate a subtask $\omega_{i_1} \in \Omega_j$ with $\omega_{i_1} \models \omega_{i_1}^1, \omega_{i_2}^2$, and $\sigma_{i_1} \subseteq \sigma'_{j_1}, \forall m_3 < j_1, \sigma_{j_1}^s \subseteq \sigma'_{m_3}$. If $j_1 \neq j_2$, the step in Line 7 will generate $\omega_{M_\Omega(i_2)} \models \omega_{i_2}^2$, with $\sigma_{M_\Omega(i_2)} \subseteq \sigma'_{j_2}, \forall m_4 < j_2, \sigma_{M_\Omega(i_2)}^s \subseteq \sigma'_{m_4}$. Thus, there exists $P_j \in \mathcal{P}_{final}$ that satisfies the first condition. For the second condition, $\leq_j$ consists of two parts generated by Line 9 and Line 11. Line 9 guarantees that $w \in \mathcal{L}(P_{\varphi_1}) \cap \mathcal{L}(P_{\varphi_2})$. Line 11 guarantees that w does not conflict the self-loop constraints of $P_{\varphi_1}, P_{\varphi_2}$. Thus, the second condition is satisfied. Regarding the third condition, since $\neq_j = \neq_1 \cap M_\Omega(\neq_2)$ holds, $\neq_j$ is naturally satisfied by w. In conclusion, for any $w \in \mathcal{L}(P_{\varphi_1}) \cap \mathcal{L}(P_{\varphi_2})$, $w \in \mathcal{L}(\mathcal{P}_{final})$ holds and vice versa. Thus, $\mathcal{L}(\mathcal{P}_{final})$ is equal to $\mathcal{L}(P_{\varphi_1}) \bigcap \mathcal{L}(P_{\varphi_2})$. □

Given a set of sub-formulas $\{\varphi_{b_i}, i \leq m\}$ and $\max_i |\varphi_{b_i}| = n$, the complexity of calculating the NBA of the conjunction φ is $O(mn \cdot 2^{mn})$, see [6]. Via our method, the complexity of computing the product of any two sub-formulas is $O(n^3 \cdot 2^n)$, while the complexity of computing the first R-poset is only $O(n^3)$. Thus, the overall complexity to compute the complete R-posets of φ is $O(n^{3m} \cdot 2^{mn})$, while deriving the first solution has a polynomial complexity of $O(m^2 n^3)$.

V. EXPERIMENT

In this section, we show the numerical results in a simulated hospital environment with different types of patients. The agents need to assist the personnel to treat patients, deliver goods and surveil, as specified by sc-LTL formulas online. The proposed approach is implemented in *Python3* on top of Robot Operating System (ROS). All benchmarks are run on a laptop with 16 cores, 2.6 GHZ processors, and 32 GB of RAM.

A. Environment Setup

As showed in Fig. I, the numerical study simulates in a hospital environment, which consists of the wards $w_1 \cdots w_8$, the operating rooms $o_1 \cdots o_5$, the hall $hall_1$, the exits $e_1 \cdots e_3$ and the hallways $h_1 \cdots h_{22}$. The multi-agent system is employed with 3 Junior Doctors (JD), 6 Senior Doctors (SD), and 8 Nurses (Nu), and 4 types of patients are treated as interactive objects, including *Family Visitors* (FV), *Vomiting Patients* (VP), *Senior patients* (SP) and *Junior Patients* (JP). The collaborative behaviors and their labels are "primary operate (P)", "advance operate (A)", "disinfect ground (D)", "transfer patient (T)", "collect information (C)", "guide visitor (G)", "supply medicine (M)", "sterilize radiation (R)".

We define a group of basic formulas Φ_b as the requirement of existing patients $\mathcal{O}_{in} = \{1,2,3,4,5\}$: $\varphi_{b_1} = \Diamond D_{w7,w7}^\varnothing \wedge \Diamond(C_{w7,w7}^\varnothing \wedge \neg M_{w7,w7}^\varnothing \wedge \Diamond M_{w7,w7}^\varnothing)$; $\varphi_{b_2} = \Diamond(C_{w7,w7}^\varnothing \wedge \neg G_{w7,e3}^1 \wedge \Diamond G_{w7,e3}^1) \wedge \neg D_{w7,w7}^\varnothing \mathsf{U} C_{w7,w7}^\varnothing$; $\varphi_{b_3} = \Diamond(T_{w1,o1}^2 \wedge \neg P_{o1,o1}^2 \wedge \Diamond(P_{o1,o1}^2 \wedge \neg R_{o1,o1}^\varnothing \wedge \Diamond T_{o1,w1}^2 \wedge \Diamond R_{o1,o1}^\varnothing))$; $\varphi_{b_4} = \Diamond(C_{w3,w3}^\varnothing \wedge \Diamond(T_{w3,o4}^3 \wedge \Diamond A_{o4,o4}^3 \wedge \neg R_{o4,o4}^\varnothing \wedge \Diamond(T_{o4,w3}^3 \wedge \neg A_{o4,o4}^3 \wedge \Diamond C_{w3,w3}^\varnothing) \wedge \Diamond(R_{o4,o4}^\varnothing \wedge \Diamond D_{o4,o4}^\varnothing)))$; $\varphi_{b_6} = \Diamond(T_{h5,w5}^4 \wedge \Diamond C_{w5,w5}^\varnothing \wedge \Diamond M_{w5,w5}^\varnothing) \wedge \neg M_{w5,w5}^\varnothing \mathsf{U} C_{w5,w5}^\varnothing$; $\varphi_{b_7} = \Diamond(C_{w3,w3}^\varnothing \wedge \neg T_{w3,e2}^5 \wedge \Diamond D_{w3,w3}^\varnothing \wedge \Diamond T_{w3,e2}^5)$.

During execution, if an object u is added to $\mathcal{O}$ at region W_i with the goal region W_j, the corresponding contingent task formula φ_{e_u} is set differently as: $\Diamond(T_{i,i}^u \wedge \Diamond R_{j,j}^\varnothing \wedge \Diamond D_{i,j}^\varnothing) \wedge \neg Nu_i \mathsf{U} R_{i,i}^\varnothing$ with object type VP; $\Diamond G_{i,j}^u \wedge \Diamond R_{i,i}^\varnothing \wedge \neg JD_i \mathsf{U} R_{i,i}^\varnothing$ with object type JP; $\Diamond(T_{i,j}^u \wedge (\bigwedge_{\ell=1\cdots3} \neg SP_{o_\ell}) \wedge \Diamond M_{j,j}^\varnothing \wedge \Diamond C_{j,j}^\varnothing)$ with type SP; $\Diamond G_{i,j}^u \wedge (\bigwedge_{\ell=1\cdots5} \neg FV_{o_\ell})$ with type FV. Specially, the $\neg FV_{o_\ell}$ is the set of propositions in **p**, i.e., the objects of type FV should not enter o_ℓ.

B. Results

To begin with, the R-posets of φ_{b_1} and φ_{b_2} are computed as $\{P_{b_1}\}, \{P_{b_2}, P'_{b_2}\}$, showed in Fig. 3. The subtask ω_i consists of a part (in blue) representing the self-loop labels σ_i^s and a part (in green) representing the action labels σ_i. The directed black arrows represent the $\preceq$ relation and the bidirectional red arrows for the $\neq$ relation. P_{b_2} is chosen first to compute the product $P_{b_1} \otimes P_{b_2}$, as it has less subtasks and ordering constraints compared with P'_{b_2}. P_{f1}, P_{f2} are the results of $P_{b_1} \otimes P_{b_2}$ showed in Fig. 3. We choose P_{f2} for the next iteration instead of P_{f1} for the same reason as P_{b_2}. More specifically, there are two subtasks $\omega_2^1 \in P_{b_1}$ and $\omega_1^2 \in P_{b_2}$ with the same requirement $C_{w7,w7}^\varnothing$ in the action labels, where their self-loop constraints have no conflicts. Thus, we can choose $\omega_2 \in P_{f2}$
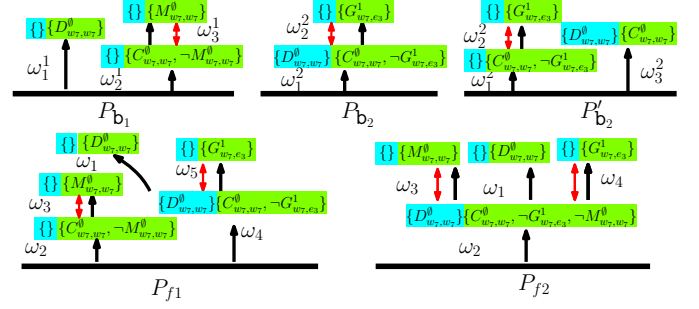
Fig. 3. Illustrations of intermediate R-posets. P_{b_1} is the R-poset of φ_{b_1}; P_{b_2}, P'_{b_2} are the R-posets of φ_{b_2}. $P_{b_1} \otimes P_{b_2} = \{P_{f1}, P_{f2}\}$.
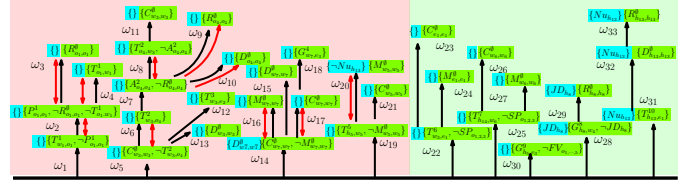
Fig. 4. The final R-poset P_{final} after the completion of all tasks.

which satisfies both $\omega_2^1 \in P_{b_1}$ and $\omega_1^2 \in P_{b_2}$ with a mapping function $M(\omega_1^2) = \omega_2, M(\omega_2^2) = \omega_4$, generated from Line 6 in Alg. 1. Otherwise, we can keep these two tasks and as Line 7 in Alg. 1. This case is shown in P_{f1} with the mapping function $M(\omega_1^2) = \omega_4, M(\omega_2^2) = \omega_5$. Moreover, the conflicts between the self-loop labels σ_i^s and the task labels σ_j are checked in Line 11. Consequently, (ω_2, ω_1) is added to $\preceq$ without conflicts between σ_1 and σ_2^s. As shown in Fig. 4 showed, the final R-poset consists of 21 subtasks generated offline and 12 contingent subtasks added online. The length of the task formula φ is 33. The first R-poset is computed in $3.5s$ but the complete R-posets is not exhausted even after $11h$ of computation. This signifies the importance of an anytime algorithm to compute the R-posets.

During the task assignment and execution, there exists 5 objects in $\mathcal{O}_{in}$, and then 5 additional objects are added in $\mathcal{O}_{on}$ at 4 different time instances as highlighted in the gantt graph of the complete system in Fig. 5. It can bee seen that all subtasks follow the specified ordering constraints in the R-posets. For instances, although the agents $6, 14$ have arrived region $w7$ earlier than $100s$ for their first subtask $\sigma_{16} = \{M_{w7,w7}^\varnothing\}$ (in blue), they have to wait until that the subtask $\sigma_{14} = \{C_{w7,w7}^\varnothing\}$ has been fulfilled by agent 10 (in brown), due to the constraints that $(\omega_{14}, \omega_{16}) \in \preceq$, $\{\omega_{14}, \omega_{16}\} \in \neq$. It is worth noting that task ω_7 (in green) cannot be executed at $380s$ before subtask ω_6 (in purple) is finished as the required object 3 has not been transferred to region o_4. Furthermore, most tasks are executed in parallel without much constraints for efficiency. Comparing to sequential execution, our method can reduce the completion time from $2013.5s$ to $815s$.

After execution, the complete formula φ has a length of 62, which can not be converted to NBA within $5h$. Instead, our algorithm of online adaptation is triggered 4 times, which takes $10.96s$ in total. In contrast, to compute the products offline will take $32.47s$. The trajectories of agents are shown

Fig. 5. The Gantt graphs during the task execution. The black lines indicate that the additional tasks are added at $40s, 180s, 215s, 400s$, and the green blocks represent contingent tasks added online.

Fig. 6. Final R-poset P_{final} in the hardware experiment.

Fig. 7. Gantt graph of task execution in the hardware experiment: the actual execution (**Left**) and the planned execution (**Right**).

Fig. 8. Trajectories of agents in grey and objects in red (**Left**); a snapshot during execution (**Right**).

in Fig. I as gray lines while the trajectories for objects are shown in red. It is worth noting that the paths respect both the environment and temporal constraints. For instance, region h_{13} is in shadow and agents of type Nu can not enter due to the label constraints $\neg Nu_{h_{13}}$. Moreover, the $Transfer$ behavior is collaborative and requires two agents with $carry$ function to move an object of type SP to the target region. Thus, the regions o_1, o_2, o_3 are forbidden during the execution of $Transfer$, due to the constraints $\bigwedge_{\ell=1\cdots3} \neg SP$ caused by the newly-added object SP.

C. Hardware Experiment

We further test the proposed method on hardware within a simplified hospital environment. The multi-agent system consists of 10 differential drive mobile robots, with 3 JD in green, 3 SD in yellow and 4 Nu in blue. The basic formulas in Φ_b are given by: $\varphi_{b_1} = \Diamond(C^{\varnothing}_{w3,w3} \wedge \Diamond(T^1_{w3,o4} \wedge \Diamond(A^1_{o4,o4} \wedge \neg R^1_{o4,o4} \wedge \Diamond(T^1_{o4,w3} \wedge \neg A^1_{o4,o4} \wedge \Diamond C^{\varnothing}_{w3,w3})) \wedge \Diamond R^{\varnothing}_{o4,o4}));$ $\varphi_{b_2} = \Diamond(C^{\varnothing}_{w3,w3} \wedge \neg T^2_{w3,e2} \wedge \Diamond D^{\varnothing}_{w3,w3} \wedge \Diamond T^2_{w3,e2}); \varphi_{b_3} = \Diamond(T^3_{h5,w2} \wedge \Diamond C^{\varnothing}_{w2,w2} \wedge \Diamond M^{\varnothing}_{w2,w2}); \varphi_{b_4} = \neg M^{\varnothing}_{w2,w2} U C^{\varnothing}_{w2,w2}$.

As shown in Fig. 6, the final R-poset consists of 14 subtasks of which the part in red background is calculated offline, while the part in yellow is generated online. Subtask ω_1 is generated by Line 6 in Alg. 1 as the first $C^{\varnothing}_{w3,w3}$ of φ_{b_1} and the $C^{\varnothing}_{w3,w3}$ of φ_{b_2}. A new partial relation is added via Line 11 due to the conflicts between $\neg M^{\varnothing}_{w2,w2}$ in φ_{b_4} and the $M^{\varnothing}_{w2,w2}$ in φ_{b_3}. The complete R-poset product is derived in $3.1s$ while the task assignment takes $2.7s$ in average. As shown in Fig. 7, the Gantt graph is updated twice at $5s$ and $100s$ during execution. The proposed TDCN method in Alg. 2 generates a complete assignment based on the given model. However, the movement of agents during real execution requires more time due to drifting and collision avoidance. Nonetheless, the proposed

method can adapt to these fluctuations and still satisfy the ordering constraints. The final agent and object trajectories are shown in Fig. 8. Simulation and experiment videos are provided in the supplementary files.

D. Scalability Analysis

To further validate the scalability of the proposed method against existing methods, we evaluate the time and memory complexity for computation both first R-poset and complete R-posets, given a set of task formulas with lengths between 5 to 40. All conversions from the LTL formula to NBA is via LTL2BA, against four baselines: (i) the direct translation; (ii) the first R-poset by products; (iii) the complete R-posets by [3]; (iv) the complete R-posets by products; (v) the decomposition algorithm from [20]. As shown in Fig. 9, the computation time and memory both increase quickly with the length of formulas. Most algorithms run out of memory or time when the length exceeds 20. In comparison, our method can always generate the first valid R-poset in a short time, although the complete product is also infeasible for formulas with length more than 25. It is consistent with our complexity analysis in Sec. IV-E. Moreover, the computation time is recorded for different number of agents and objects. As shown in Fig. 9, the computation time only increases slightly as the number of agents systems while the number of objects has a larger impact (all under $7s$). This indicates that the proposed assignment algorithm scales well with the system size.

VI. CONCLUSION

In this letter, a new task planning method is proposed for multi-agent system under LTL tasks, to tackle the computational complexity for long and complex task formulas. The direct translation of the complete task formula into the associated NBA is avoided, by instead computing R-posets and their products. The task assignment algorithm can ensure
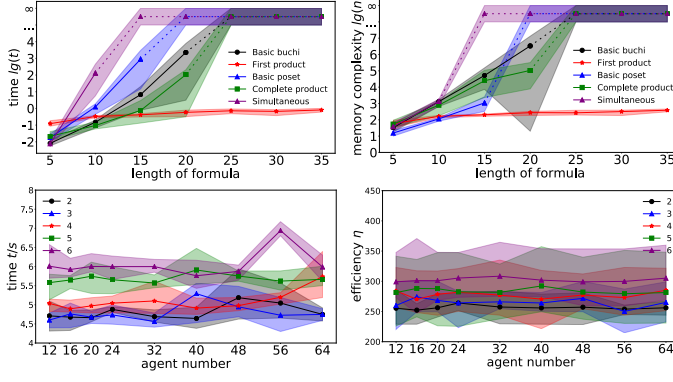
Fig. 9. The computation complexity (**Upper Left**) and memory complexity (**Upper Right**); The computation time (**Lower Left**) and execution efficiency (**Lower Right**) with different number of agents and objects.

the temporal and collaborative constraints within the R-posets. The proposed methods scale well to long task formulas that are generated online and large system sizes.

REFERENCES

[1] Katoen and Joost-Pieter, *Principles of model checking*. Principles of model checking, 2008.

[2] C. K. Verginis, Y. Kantaros, and D. V. Dimarogonas, "Planning and control of multi-robot-object systems under temporal logic tasks and uncertain dynamics," *arXiv preprint arXiv:2204.11783*, 2022.

[3] Z. Liu, M. Guo, and Z. Li, "Time minimization and online synchronization for multi-agent systems under collaborative temporal tasks," *arXiv preprint arXiv:2208.07756*, 2022.

[4] 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.

[5] M. Ben-Ari, "A primer on model checking," *ACM Inroads*, vol. 1, no. 1, pp. 40–47, 2010.

[6] P. Gastin and D. Oddoux, "Fast ltl to büchi automata translation," in *Computer Aided Verification*, G. Berry, H. Comon, and A. Finkel, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 53–65.

[7] X. Ding, S. L. Smith, C. Belta, and D. Rus, "Optimal control of markov decision processes with linear temporal logic constraints," *IEEE Transactions on Automatic Control*, vol. 59, no. 5, pp. 1244–1257, 2014.

[8] M. Kloetzer and C. Mahulea, "Accomplish multi-robot tasks via petri net models," in *2015 IEEE International Conference on Automation Science and Engineering (CASE)*, 2015, pp. 304–309.

[9] K. Leahy, Z. Serlin, C.-I. Vasile, A. Schoer, A. M. Jones, R. Tron, and C. Belta, "Scalable and robust algorithms for task-based coordination from high-level specifications (scratches)," *IEEE Transactions on Robotics*, vol. 38, no. 4, pp. 2516–2535, 2022.

[10] P. Schillinger, M. Bürger, and D. V. Dimarogonas, "Hierarchical ltl-task mdps for multi-agent coordination through auctioning and learning," *The international journal of robotics research*, 2019.

[11] Y. Kantaros and M. M. Zavlanos, "A temporal logic optimal controlsynthesis algorithm for large-scale multi-robot systems," *The International Journal of Robotics Research*, vol. 39, no. 7, p. 027836492091392, 2020.

[12] N. Piterman, A. Pnueli, and Y. Sa'ar, "Synthesis of reactive(1) designs," in *Verification, Model Checking, and Abstract Interpretation*, E. A. Emerson and K. S. Namjoshi, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 364–380.

[13] V. Vasilopoulos, Y. Kantaros, G. J. Pappas, and D. E. Koditschek, "Reactive planning for mobile manipulation tasks in unexplored semantic environments," in *International Conference on Robotics and Automation*, 2021.

[14] C. K. Verginis and D. V. Dimarogonas, "Multi-agent motion planning and object transportation under high level goals," in *IFAC World Congress*, 2018.

[15] B. Lacerda, D. Parker, and N. Hawes, "Optimal and dynamic planning for markov decision processes with co-safe ltl specifications," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 1511–1516.

[16] S. Feyzabadi and S. Carpin, "Multi-objective planning with multiple high level task specifications," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 5483–5490.

[17] C. Belta, B. Yordanov, and E. A. Gol, *Formal methods for discrete-time dynamical systems*. Springer, 2017, vol. 15.

[18] X. Luo and M. M. Zavlanos, "Temporal logic task allocation in heterogeneous multirobot systems," *IEEE Transactions on Robotics*, vol. 38, no. 6, pp. 3602–3621, 2022.

[19] Smith, "The contract net protocol: High-level communication and control in a distributed problem solver," *IEEE Transactions on Computers*, vol. C-29, no. 12, pp. 1104–1113, 1980.

[20] F. Faruq, D. Parker, B. Laccrda, and N. Hawes, "Simultaneous task allocation and planning under uncertainty," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3559–3564.