Research
A SCIENCE PARTNER JOURNAL

RESEARCH ARTICLE

Fast and Adaptive Multi-Agent Planning under Collaborative Temporal Logic Tasks via Poset Products

Zesen Liu[1], Meng Guo[1], Weimin Bao[2], and Zhongkui Li[1]*

[1]Department of Mechanics and Engineering Science, College of Engineering, Peking University, Beijing 100871, China. [2]Science and Technology Commission of China Aerospace Science and Technology Corporation, Beijing 100048, China.

*Address correspondence to: zhongkli@pku.edu.cn

Citation: Liu Z, Guo M, Bao W, Li Z. Fast and Adaptive Multi-Agent Planning under Collaborative Temporal Logic Tasks via Poset Products. *Research* 2024;7:Article 0337. https://doi.org/10.34133/research.0337

Submitted 2 January 2024
Accepted 17 February 2024
Published 20 March 2024

Efficient coordination and planning is essential for large-scale multi-agent systems that collaborate in a shared dynamic environment. Heuristic search methods or learning-based approaches often lack the guarantee on correctness and performance. Moreover, when the collaborative tasks contain both spatial and temporal requirements, e.g., as linear temporal logic (LTL) formulas, formal methods provide a verifiable framework for task planning. However, since the planning complexity grows exponentially with the number of agents and the length of the task formula, existing studies are mostly limited to small artificial cases. To address this issue, a new planning paradigm is proposed in this work for system-wide temporal task formulas that are released online and continually. It avoids two common bottlenecks in the traditional methods, i.e., (a) the direct translation of the complete task formula to the associated Büchi automaton and (b) the synchronized product between the Büchi automaton and the transition models of all agents. Instead, an adaptive planning algorithm is proposed, which computes the product of relaxed partially ordered sets (R-posets) on-the-fly and assigns these subtasks to the agents subject to the ordering constraints. It is shown that the first valid plan can be derived with a polynomial time and memory complexity with respect to the system size and the formula length. Our method can take into account task formulas with a length of more than 400 and a fleet with more than 400 agents, while most existing methods fail at the formula length of 25 within a reasonable duration. The proposed method is validated on large fleets of service robots in both simulation and hardware experiments.

Introduction

Recent advances in computation, perception, and communication allow the deployment of autonomous robots in large, remote, and hazardous environments, e.g., to assist service staff in hospitals [1], to maintain offshore drilling platforms [2], and to monitor and assist construction sites [3]. Furthermore, fleets of heterogeneous robots, such as unmanned ground vehicles and unmanned aerial vehicles, are deployed to accomplish tasks that are otherwise too inefficient or even infeasible for a single robot [4]. Not only the overall efficiency of the team can be largely improved by allowing the robots to move and act concurrently [5] but also the capabilities of the team can be greatly extended by enabling multiple robots to directly collaborate on a task [6]. Recent works have demonstrated such potentials preliminary for simple tasks such as collaborative exploration [7], formation control [8], object transportation [9], and pursuer–evader games [10]. The task planning problem for multi-robot systems is in general NP-hard [11], due to the inherent combinatorial nature of robot-task assignment and various constraints such as capabilities and deadlines. The standard approach is to formulate a mixed integer linear programs (MILPs) over the integer variables and constraints [12]. While being sound and optimal, these methods are applicable to only small-scale systems. Thus, extensive work can be found on designing meta-heuristic algorithms for finding sufficiently good solutions in a reasonable time, e.g., genetic algorithms [13], colony optimization [14,15], particle swarm optimization [16,17], learning-based algorithms [18–20], or large language models [21,22]. However, these methods often lack a formal guarantee on the correctness and quality of the planning results.

Moreover, to specify more complex tasks, many recent works propose to use formal languages such as linear temporal logic (LTL) [23], computation tree logic (CTL) [24], and signal temporal logic (STL) [25] as an intuitive yet powerful way to describe both spatial and temporal requirements on global [26] or local [27] behaviors. Notably, the works in [28–30] formulate MILP by a central planning unit given different system models and task constraints; the works in [31–35] instead propose various search algorithms over the state or solution space of the whole system. However, the aforementioned planning methods are often executed offline for a set of predefined static tasks. A particularly challenging scenario is when the system operates indefinitely, i.e., new tasks are released or canceled dynamically and continually by external demand [36], or certain target features related to the tasks can change location during run time [10]. This would require the fleet to adaptively change their task plans online to modify existing assignments and incorporate new

tasks. Thus, the aforementioned methods become inadequate as the sequence of tasks is infinite and their specifications are unknown beforehand. Recursive application of the centralized methods in a naive way leads to not only intractable computation complexity but also inconsistent or even oscillatory assignments. Thus, an efficient and adaptive planning scheme is essential for multi-robot systems that collaborate in a dynamic environment [37–39] or an unknown environment [40].

Related work

The standard framework for planning under temporal tasks is based on the model-checking algorithm [23]: 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 [41] and LTL2BA [42]. Second, a product automaton is created between the automaton of formula and the models of all agents, such as weighted finite transition systems (wFTSs) [23], Markov decision processes (MDPs) [43], or petri nets [44]. 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 [32], integer programs [45], auction [46], or sampling-based [33,47].

Thus, the fundamental step of all aforementioned methods is to translate the task formula into the associated automaton. This translation may lead a double-exponential size w.r.t. the formula length as shown in [42]. The only exceptions are GR(1) formulas [48], 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 h and 13 GB memory to compute the associated NBA via LTL2BA. Although recent methods have greatly reduced the planning complexity in other aspects, the length of considered task specifications remains limited due to this translation process. For instance, the sampling-based method [33,47] avoids creating the complete product automaton via rapidly exploring random tree (RRT) sampling, of which the largest simulated case has 400 agents and the task formula has a maximum length of 14. The planning method [32] decomposes the resulting task automaton into independent subtasks for parallel execution. The simulated case scales up to 100 robots and a task formula of maximum length 18. Moreover, other existing works such as [49–52] mostly consider task formulas of length around 6 to 10. This limitation hinders its application to more complex robotic missions.

This drawback becomes even more apparent for 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, e.g., [51,52]. 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 handpicked examples.

Our contribution

To overcome this curse of dimensionality in the size of tasks and agents, we propose a new paradigm that is fundamentally different from the model checking-based methods. First, for a syntactically co-safe LTL (sc-LTL) formula that is a conjunction of numerous subformulas, we calculate the R-posets of each subformula 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 subformula. The resulting product of R-posets is 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.

The main contribution of this work is threefold: (a) a systematic method is proposed to tackle task formulas with length more than 400, which overcomes the limitation of existing translation tools that can only process formulas of length less than 25 in reasonable time; (b) 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; (c) the proposed task assignment algorithm is fully compatible with both static and dynamic scenarios with interactive objects.

Problem statement

Consider a multi-agent system with heterogeneous capabilities, a series of sc-LTL task formulas that are released online, and a set of interactive objects in the dynamic environment. The objective is to generate a task plan for the system online such that these tasks are satisfied with high efficiency.

For instance, as shown in the "Numerical simulation" section, a fleet of heterogeneous service robots is deployed in the hospital environment. Different tasks such as transportation of goods or patients, cleaning, and maintenance are released online continuously. Many of such tasks contain numerous subtasks with ordering constraints that require direct collaboration of different robots. An efficient coordination algorithm is proposed such that these subtasks are assigned and fulfilled online in a timely manner.

Results

In this section, we present the proposed solution briefly, where we first give a definition of task specification and introduce the core method of computing the R-poset product. Then, an efficient assignment algorithm is introduced under these posets with temporal and spatial constraints. The simulation and hardware experiment results are presented against several strong baselines for different sizes of fleets and task complexities. Technical details and derivations can be found in Materials and Methods.

Task specification and R-poset product

We briefly introduce the syntax of LTL used for task specifications. The basic ingredients of 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 is defined as $\varphi \triangleq \top \mid p \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \wedge \varphi_2 \mid \neg \varphi \mid \bigcirc \varphi \mid \varphi_1 \ U\varphi_2 \mid \square \varphi \mid \diamond \varphi,$

where $\top \triangleq$ True; $p \in AP$ is the alphabet; $\vee$ (conjunction), $\wedge$ (disjunction), and $\neg$ (negation) are the logical operators; $\diamond$ (eventually), $\bigcirc$ (next), and U (until) are the temporal operators; and $\square$ (always) and $\Rightarrow$ (implication) are the derived operators. A special class of LTL formula called sc-LTL [23,53] only contains the temporal operators $\bigcirc$, U, and $\diamond$. A complete description of the semantics and syntax of LTL can be found in [23].

Given a series of sc-LTL formulas that are released online, most existing methods would combine the formulas with $\wedge$ (conjunction) and convert them into an NBA. It results in a graph structure that consists of states, transitions, guards, inital states, and final states. However, since its size is double exponential to the length of formula φ as proven in [42], it quickly becomes intractable with more task formulas. Instead, we compute the product of R-posets associated with these formulas, called the Poset-prod (denoted by $\otimes$). The R-poset $P = (\Omega, \preceq, \neq)$ is a high-level abstraction of NBA, proposed in our earlier work [54], which consists of a set of subtasks Ω and their partial relations as less equal $\preceq$ and conflict $\neq$. It has been proven therein that if the subtasks are executed under the partial relations, the resulting traces satisfy the NBA. Once a new task formula is released, it is transformed into a new R-poset P_2 and its product with the current R-poset P_1 is computed. More specifically, given P_1, P_2, the Poset-prod returns a new set of R-posets $\mathcal{P} = P_1 \otimes P_2$ that satisfies both P_1 and P_2, which is computed by iterating the following two procedures: Task composition and Relation update. The first procedure is to create a group of subtasks as a combination between the subtasks in P_2 and the subtasks of P_1 that have not been executed. A depth-first-search (DFS) algorithm is proposed to gradually add the subtasks along with the corresponding mapping function. The second procedure is to calculate the partial relations between the subtasks such that the ordering constraints among the subtasks in the composed product are consistent without conflicts. Consequently, the overall poset is given by $P_{final} = (\Omega_f, \preceq_f, \neq_f)$, which is iteratively computed each time a new poset is added.

The proposed Poset-prod method outperforms the traditional method in both computational efficiency and performance. This improvement becomes even more pronounced when the number and length of subformulas increase. This is because the time of generating NBAs of $\varphi_1, \varphi_2, \cdots$ grows linearly, while the time of converting $\varphi_1 \wedge \varphi_2 \wedge \cdots$ into NBA grows exponentially. Moreover, for a new added formula, the algorithm updates the final R-poset based on the previous result, which ensures the performance for online cases. Finally, it is an anytime algorithm that the algorithm can calculate an R-poset within linear complexity for multiple subformulas at the expense of optimality. The concrete complexity analysis of Poset-pord is shown in Discussion, and the definition of R-poset and label is shown in Materials and Methods.

To give an example, consider four subformulas φ_1, φ_2, φ_3, φ_4 as shown in Fig. 1. Their NBAs $\mathcal{B}_1, \cdots, \mathcal{B}_4$ can be transformed into four R-posets $P_1, \cdots, P_4$. The first round of Poset-prod is between P_1 and P_2 as $P_1 \otimes P_2 = \{P_{f1}, P_{f2}\}$. Then, the following rounds of Poset-prod are performed between the results of the previous round and the next R-poset as $P_{f1} \otimes P_3$, of which the first solution is denoted by P_{f3}. At the expense of some optimality, the algorithm can go to next round before all results of $P_{f2} \otimes P_3$ are generated, as it is an anytime algorithm. Finally, we can derive P_{f4} by computing $P_{f3} \otimes P_4$ as the final R-poset, which satisfies $P_1, \cdots, P_4$. Note that the time to generate the NBA associated with φ_1, $\wedge_{i=1}^{2} \varphi_i$, $\wedge_{i=1}^{3} \varphi_i$, and $\wedge_{i=1}^{4} \varphi_i$ grows significantly with a duration of 0.058 s, 0.076 s, 1.56 s, and 25.56 s via LTL2BA [42]. By contrast, the time to compute these products $P_1 \otimes P_2$, $P_1 \otimes P_2 \otimes P_3$, $P_1 \otimes P_2 \otimes P_3 \otimes P_4$ grows slower with a duration of 0.199 s, 0.279 s, and 0.385 s. Thus, our method can deal with a much larger number of tasks online. Detailed comparisons between our method and traditional methods can be found in the "Scalability analysis and comparisons" section.

Online subtask assignment under complexity constraints

Once the set of R-posets that satisfies all subformulas is generated, a series of subtasks in the R-poset should be assigned to the agents under several complexity constraints, including temporal constraints from R-poset, objects constraints, and cooperative constraints. Similar to the classical contract net method [55], we propose an efficient and suboptimal assignment algorithm called time bound contract net (TBCN). The main difference is that all these constraints are represented uniformly by time bounds in our methods, and an example is shown in Fig. 2. The final assignment is a group of timed sequence of robot actions $\mathcal{J} = [J_1, \cdots, J_n]$, where $J_n = [(t_k, \omega_k, a_k), \cdots]$ indicates that agent n will execute action a_k at time t_k to satisfy subtask ω_k such that the newly released tasks are fulfilled.

TBCN consists of three steps: initialization, computation of feasible subtasks, and online bidding. As the partial constraints of final R-poset P_{final} might be changed after computing the product, the subtasks that are conflicting with the updated partial orders are removed in initialization. Then, the last two steps are iterated: the set of subtasks whose partial orders are satisfied given the current assignment $\mathcal{J}$ in computation of feasible subtasks. Consequently, a linear program is formulated and solved in the online bidding, to choose one subtask from the feasible set. Thus, the resulting execution time and action plans are added to $\mathcal{J}$.

Numerical simulation

As shown in Fig. 4, the hospital environment consists of the wards, the operating rooms, the hall, the exits, and the hallways. The multi-agent system is employed with three junior doctors, six senior doctors, and eight nurses, and four types of patients are treated as interactive objects including family visitors, vomiting patients, senior patients, and junior patients. The detailed mappings between agents, action, objects, and their labels are shown in Table S1. Furthermore, various types of tasks are considered, such as "Go the rounds of the wards and provide medicine," "Check and record the patient status," and "Prepare and perform an operation on a patient." Some tasks are released online under certain conditions, for instance, when a patient vomits at a region, the task "Take the patient into ward and check his status. Meanwhile, the doctors should not enter this region until it has been cleaned." The sc-LTL formula associated with the complete task is given by $\varphi = \varphi_{b1} \wedge \cdots \varphi_{b_6} \wedge \varphi_{VP} \wedge \varphi_{JP} \wedge \varphi_{SP}^1 \wedge \varphi_{SP}^2 \wedge \varphi_{FV}$, which has a total length of 62. Detailed description of formulas are shown in Table S2.

As shown in Fig. 3, each subtask ω_i consists of the constraints before execution (in blue) and during execution (in green). The directed black arrows denote the ordering constraints, while the bidirectional red arrows for the conflicting constraints. The mapping from the subtasks within the input R-posets to the subtasks within the final R-poset is shown in brackets as $\omega_1'(\omega_1'): \omega_1' \rightarrow \omega_1$.

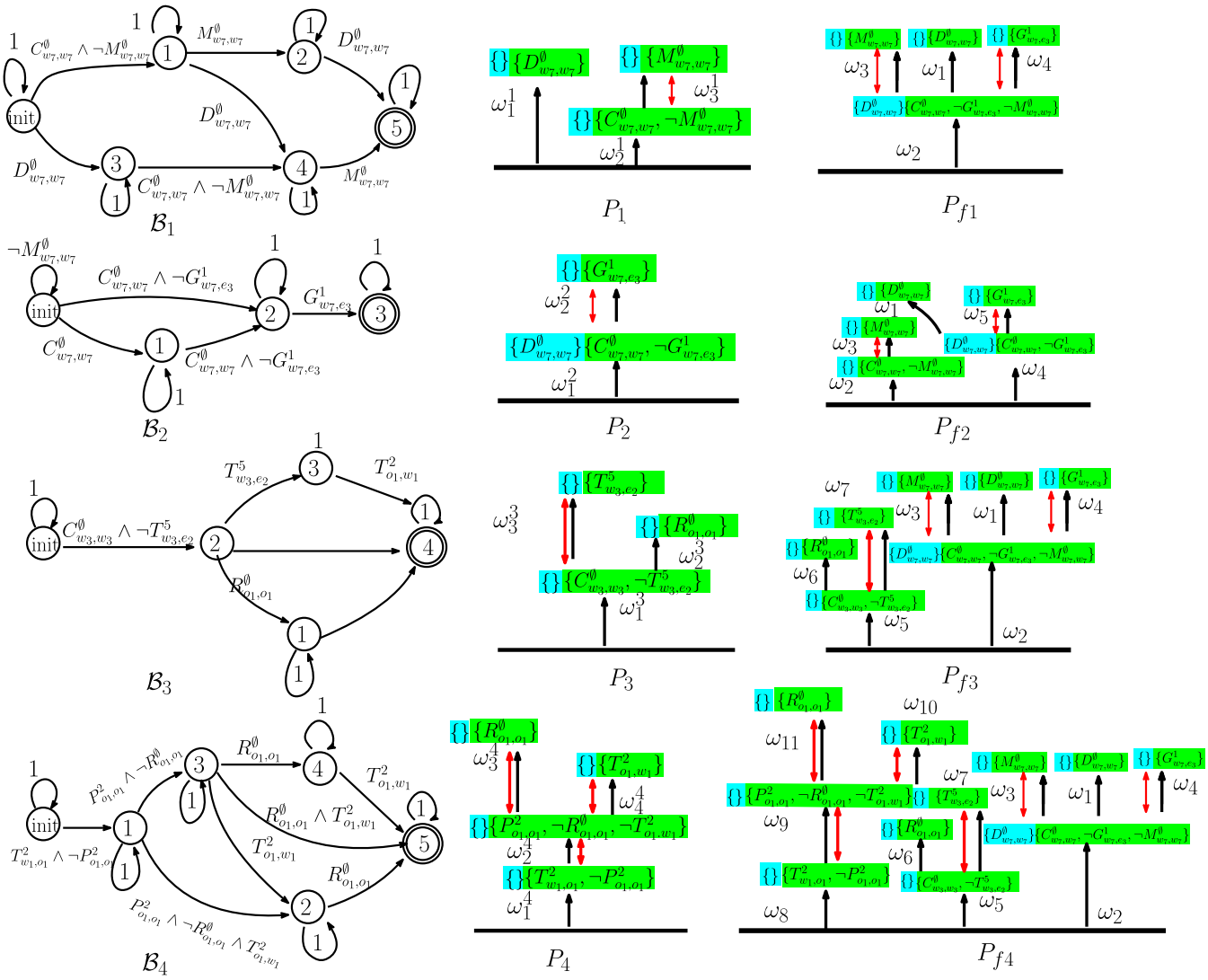

Fig. 1. Example of Poset-prod associated with four formulas $\varphi_1 = \diamond D^{\emptyset}_{w7,w7} \wedge \diamond \left(C^{\emptyset}_{w7,w7} \wedge \neg M^{\emptyset}_{w7,w7} \wedge \diamond M^{\emptyset}_{w7,w7}\right)$, $\varphi_2 = \diamond \left(C^{\emptyset}_{w7,w7} \wedge \neg G^{1}_{w7,e3} \wedge \diamond G^{1}_{w7,e3}\right) \wedge \neg D^{\emptyset}_{w7,w7} U C^{\emptyset}_{w7,w7}$, $\varphi_3 = \diamond \left(C^{\emptyset}_{w3,w3} \wedge \neg T^{5}_{w3,e2} \wedge \diamond D^{\emptyset}_{w3,w3} \wedge \diamond T^{5}_{w3,e2}\right)$, and $\varphi_4 = \diamond \left(T^{2}_{w1,o1} \wedge \neg P^{2}_{o1,o1} \wedge \diamond \left(P^{2}_{o1,o1} \wedge \neg R^{\emptyset}_{o1,o1} \wedge \diamond T^{2}_{o1,w1} \wedge \diamond R^{\emptyset}_{o1,o1}\right)\right)$. Left: $\mathcal{B}_1, \cdots, \mathcal{B}_4$ are the NBAs of formula $\varphi_1, \cdots, \varphi_4$. Middle: $P_1, \cdots, P_4$ are the R-posets of $\mathcal{B}_1, \cdots, \mathcal{B}_4$. Right: $\mathcal{B}_1, \cdots, \mathcal{B}_4 P_1 \otimes P_2 = \{P_{f1}, P_{f2}\}$, $P_{f_3} \in P_{f_1} \otimes P_3$ and $P_{f_4} \in P_{f_3} \otimes P_4$.

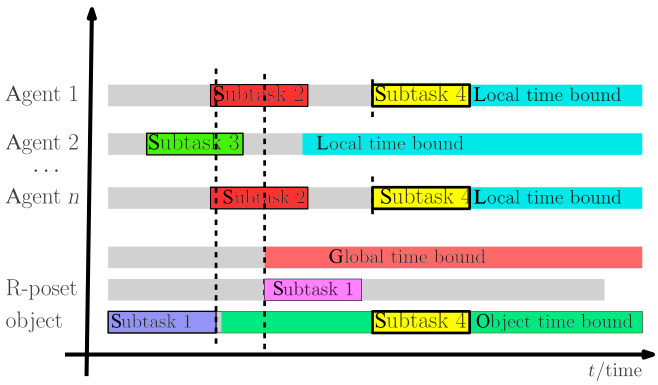

Fig. 2. The bidding process of assigning subtask 4 under the global time bound (when the partial relations in R-poset are satisfied), the time bound for object (when the object is reachable), and local time bounds (when the agents get ready).

Note that most of R-posets on the right are directly incorporated into the final R-posets on the left, such as $P_{b_1}, \cdots, P_{b_6}, P_{u_1}, \cdots, P_{u_5}$. This is due to the fact that their subtasks are independent without ordering constraints, which allows for parallel execution. Additionally, some subtasks on the left represent the same subtasks on the right. For example, $\omega_1^{'}$ of P_{b_2} and $\omega_1^{'}$ of P_{b_2} representing both ω_5, since their ordering constraints (in green) are identical. The same action can be performed to satisfy multiple subtasks on the left, which improves the overall efficiency. Furthermore, all subtasks on the right satisfy the partial relations between the corresponding subtasks on the left, while additional constraints are added if there are conflicts among the ordering constraints. For instance, action $D^{\emptyset}_{w_7,w_7}$ of ω_{15} should not be executed before the subtask ω_{14}. Thus, an additional ordering constraint is added such that subtask ω_{15} should be executed after ω_{14}. These properties guarantee that each subtask within the R-posets on the left can be executed, as their partial relations are satisfied. Consequently, the final R-poset satisfies all R-posets on the left. Note that the complete formula has a total length of 62, of which the NBA takes more than 1 h to compute. On the contrary, the first final R-poset is computed within 10.96 s.

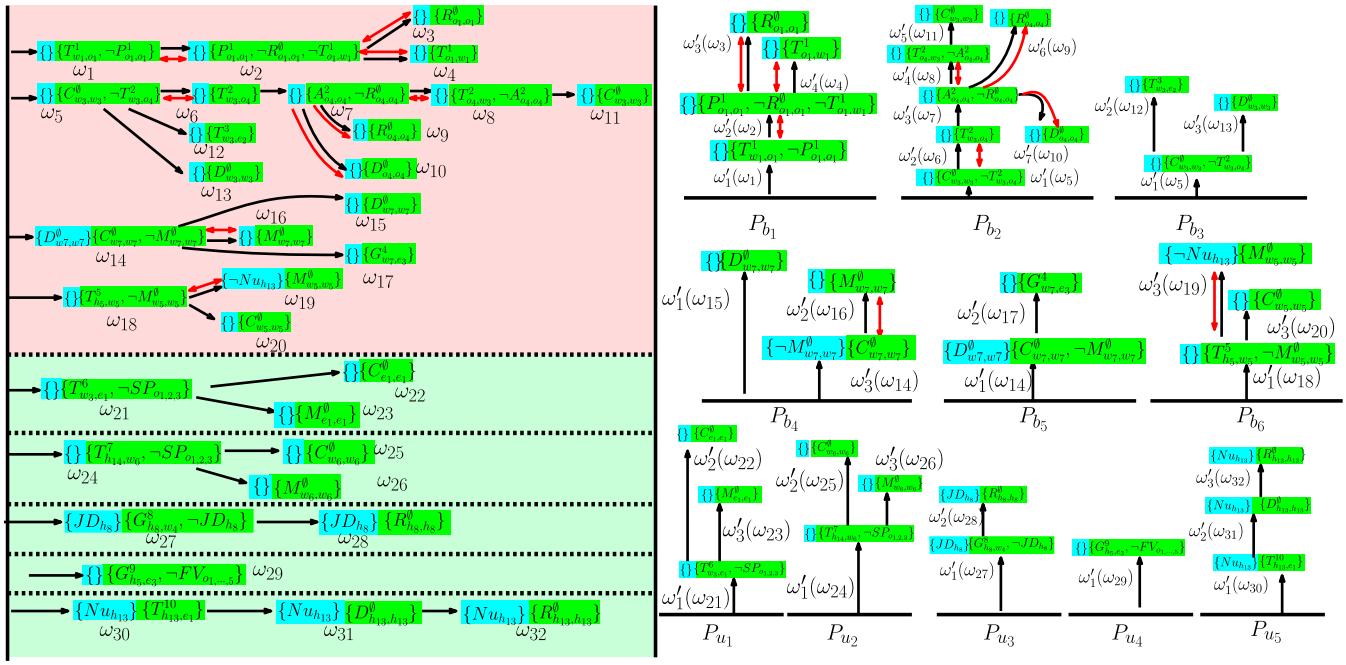

Fig. 3. Illustration of computing the product posets. Left: Final R-poset computed from the initial subformulas and online subformulas (dashed lines). Right: $P_{b_1}, \cdots, P_{b_6}$ are the R-posets associated with the initial subformulas, and $P_{u_1}, \cdots, P_{u_5}$ are associated with the online subformulas.

The results of task assignment are shown in Fig. 4, which include the updates at 40, 180, 215, and 400 s with five additional objects added. Then, after modifying the current R-posets to accommodate these formulas, the method TBCN is executed to assign the new subtasks within the final R-poset. The trajectories of agents and objects are shown, with certain regions that are not allowed to enter. For instance, when a patient vomits at region h13 in Fig. 4C, the nurses cannot enter until other agents have cleaned this region. These constraints ensure that both their actions and their trajectories satisfy the R-poset. In addition, once a new object is added, a new formula is released and then the R-poset is updated. All subtasks satisfy the ordering constraints in the R-posets. For instances, as shown in Fig. 4D, although agents 6, 14 have arrived in region w7 before 100 s to collaborate on the subtask $\sigma_{16} = \left\{ M^{\emptyset}_{w7,w7} \right\}$ (in blue), they have to wait until that agent 10 has fulfilled the subtask $\sigma_{14} = \left\{ C^{\emptyset}_{w7,w7} \right\}$, due to the ordering constraints that $(\omega_{14},\omega_{16}) \in \preceq$, $\{\omega_{14},\omega_{16}\} \in \neq$. The constraints introduced by objects are also satisfied, e.g., task ω_7 (in green) cannot be executed at 380 s before subtask ω_6 (in purple) has been finished, since the required object 3 has not been transferred to region o_4. Last but not least, most tasks are executed in parallel mostly with a total completion time of 815 s, which is much shorter than 2013.5 s if the subtasks are executed sequentially.

Scalability analysis and comparisons

First, we show how the computation time of the proposed task assignment method TBCN varies under different numbers of agents and subtasks. Since the number of subtasks cannot be directly determined, we run the TBCN with a large number of formulas, of which the length ranges from 20 to 80. The number of subtasks and the associated computation time are recorded. As shown in Fig. 5, the average computation time only increases slightly as the number of agents is increased from 12 to 400, while the computation time increases considerably if the number of subtasks is increased from 22 to 34. This is due to the fact that new subtasks would introduce additional temporal constraints in the assignment procedure. The execution efficiency η from Eq. 6 decreases as the number of agent increases, while η increases slightly as the number of subtasks grows. The highest η is about 39% where most subtasks are executed in parallel with only minimum waiting time for task synchronization.

Second, to further validate the scalability of the proposed method against existing methods, we evaluate the time and memory cost to compute both the first R-poset and the complete R-posets by the proposed Poset-prod, given the task formulas of different lengths. The conversion from an LTL formula to NBA is via LTL2BA. The following three baselines are considered: (a) the direct translation from the complete formula to NBA [42], in which the complete formula is the conjunction of all subformulas; (b) the decomposition set-based algorithm [56], which decomposes the NBA into independent subtasks; (c) the sampling-based method [33,47], which generates a product automaton much smaller than the complete one by sampling the product states of NBA and weighted transition systems (WTS) via RRT. Each method is tested three times with five formulas of the same length ranging from 5 to 400. As shown in Fig. 6, both the time and memory cost increase drastically with the formula length for all methods except the proposed Poset-prod to compute the first R-poset. In particular, when the formula length exceeds 15, the decomposition algorithm runs out of memory or time. The sampling-based method cannot generate a solution when the formula length exceeds 20. Since all the baseline methods require the translation to NBA first, it becomes intractable as the formula length exceeds 25. In contrast, the proposed method of Poset-prod can generate all R-posets with a formula length of 70 and the first R-poset even when the length reaches 400, which is consistent with our analyses in Discussion.

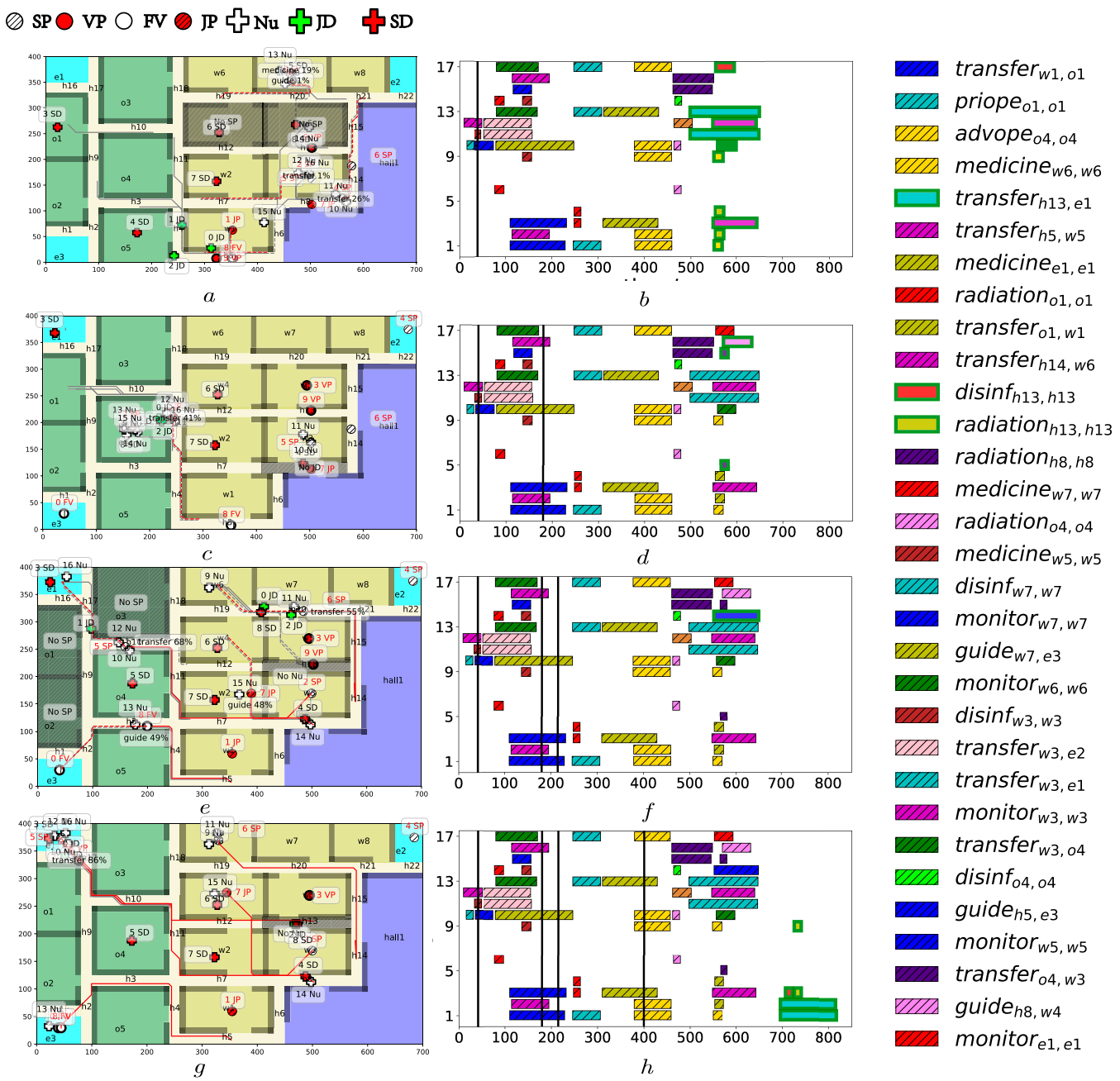

Fig. 4. (A, C, E,and G): Snapshot of agent trajectories at 40, 180, 215, and 400 s when new tasks are added online. (B, D, F, and H): Gantt graph of task assignment at these time instants, as highlighted in green boxes.

Hardware experiment

We further tested the proposed method on hardware within a simplified hospital environment. The multi-agent system consists of 10 differential-driven mobile robots, with 3 *JD* in green, 3 *SD* in yellow, and 4 *Nu* in blue. The required tasks include "prepare and execute an advanced operation for patient 1 at operating room *o4*." Similar to the "Numerical simulation" section, there are event-triggered tasks released online such as "when a patient vomits, take him to the ward, do not enter this region until it is cleaned." The exact task description and formulas are shown in Table S3.

In summary, the system is required with six subformulas, and the final formula is $\varphi = \varphi_{b1} \wedge \cdots \wedge \varphi_{b_4} \varphi_{FV} \wedge \varphi_{JP}$, with the total length 27; thus, it cannot be translated into an NBA directly. The agent trajectories within 185 s are shown in Fig. 7. As shown in the right of Fig. 7, the final R-poset consists of 14 subtasks, where the part in red is calculated offline and the part in yellow is generated online. Most subtasks are executed in parallel, meaning that the R-posets can be satisfied with a high efficiency. The complete R-poset product is derived in 3.1 s, and the task assignment method TBCN is activated three times, of which the average planing time is 2.7 s. As shown in the left of Fig. 8, the Gantt graph is updated twice at 5 and 100 s during execution, and subtasks that are released online are marked by green boxes. It is worth noting that the agent movement during real execution requires more time due to motion uncertainty, communication delay, drifting, and collision avoidance. Nonetheless, the proposed method can adapt to these

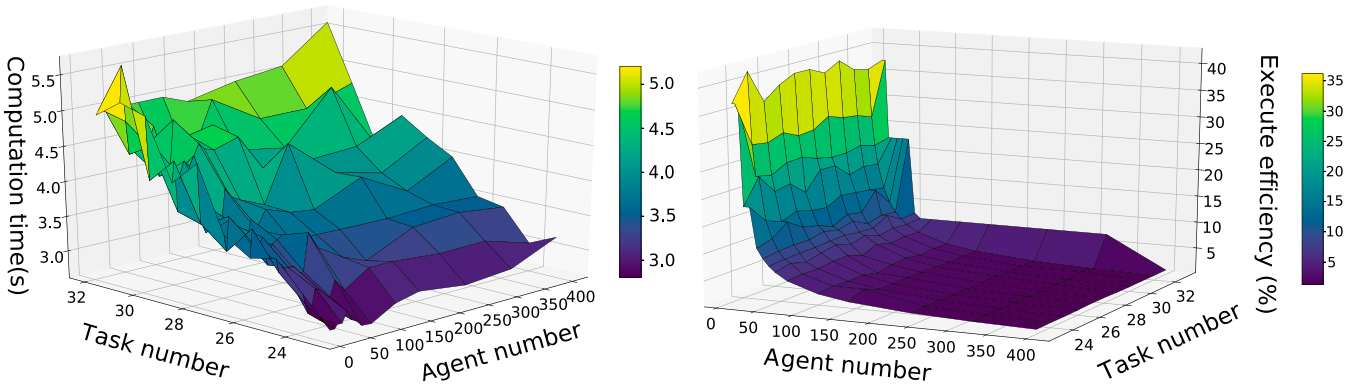

Fig. 5. The computation time (left) and the execution efficiency η (right) with respect to different number of agents and subtasks.

fluctuations and still satisfy the specified tasks, as shown in the Gantt graph of the overall execution in Fig. 8. Experiment videos are provided in the Supplementary Materials.

Discussion

In this work, we propose Poset-prod, an efficient online task planning algorithm for multi-agent systems where tasks are released dynamically and constantly online. It consists of a systematic method to convert the temporal tasks into their equivalent R-posets, of which their products are computed online. Given these R-posets, an anytime task assignment algorithm is proposed to adapt the local plans of robots online such that the overall safety and correctness are ensured. The overall framework is shown to be fast and efficient, thus particularly suitable for large-scale multi-agent systems collaborating in dynamic environments.

The most significant advantage of Poset-prod is that the complexity of obtaining the first solution only grows linearly with the length of formulas. Given a set of formulas $\{\varphi_i, i \leq m\}$ and $\max_i |\varphi_i| = n$, deriving the first solution has a polynomial complexity of $O(m^2n^3)$. Despite that the overall complexity to compute the complete R-posets of $\{\varphi_i\}$ is $O(n^{5m})$, it is still much smaller than the complexity of calculating the NBA of the conjunction φ via LTL2BA [42], which is $O(mn \cdot 2^{mn})$. Thus, our method can plan for task formulas with a length of about 400 within about 50 s, while most existing methods fail at the formula length of 25.

Future work involves two directions: (a) extending the sc-LTL task formulas to general LTL and other languages such as CTL [24] and STL [25]. It remains unclear how general LTL formulas with always operators can be incorporated in the R-poset, especially with the prefix–suffix structure, and (b) considering unknown and uncertain environments that are modeled as MDPs. In this case, a reactive high-level plan is essential to take into account all possible environment behaviors.

Materials and Methods

In this section, we provide the knowledge of LTL in the "Preliminaries" section, the definition of alphabets and objective function in the "Problem formulation" section, and algorithm details in the "Approach" section.

Preliminaries

As mentioned in the "Task specification and R-poset product" section of Results, the basic ingredients of LTL formulas are a set of atomic propositions AP in addition to several Boolean and temporal operators. For a given LTL formula φ, there exists an NBA as follows:

Definition 1 An NBA $\mathcal{A} \triangleq (S, \Sigma, \delta, (S_0, S_F))$ is a four-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, $\mathcal{L}(\varphi) = Words(\varphi) = \{W \mid W \vDash \varphi\}$ and $\vDash$ is the satisfaction relation. Additionally, the resulting run of w within $\mathcal{A}$ is an infinite sequence $\rho = s_0s_1s_2\cdots$ 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 $i \geq 0$. A run is called accepting

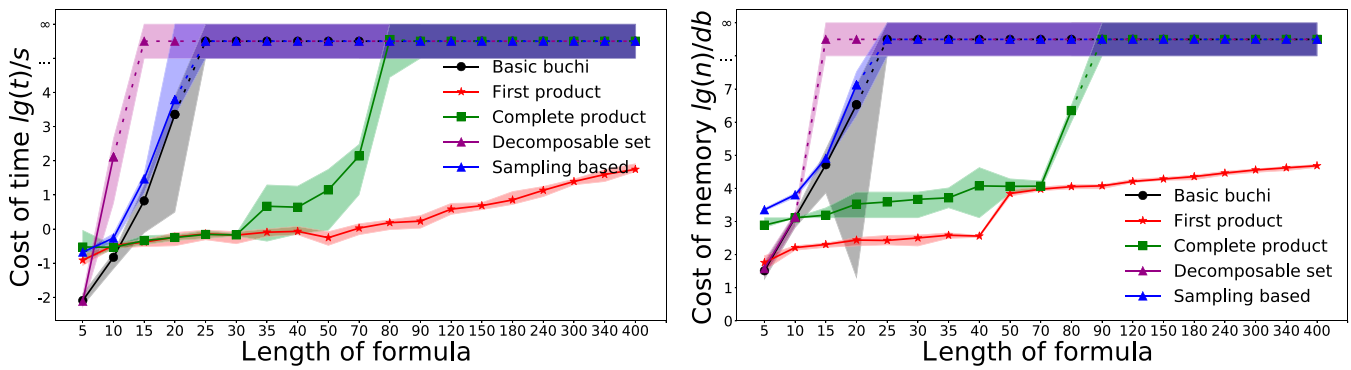

Fig. 6. The comparison of the computation time (left) and memory (right) by different methods.

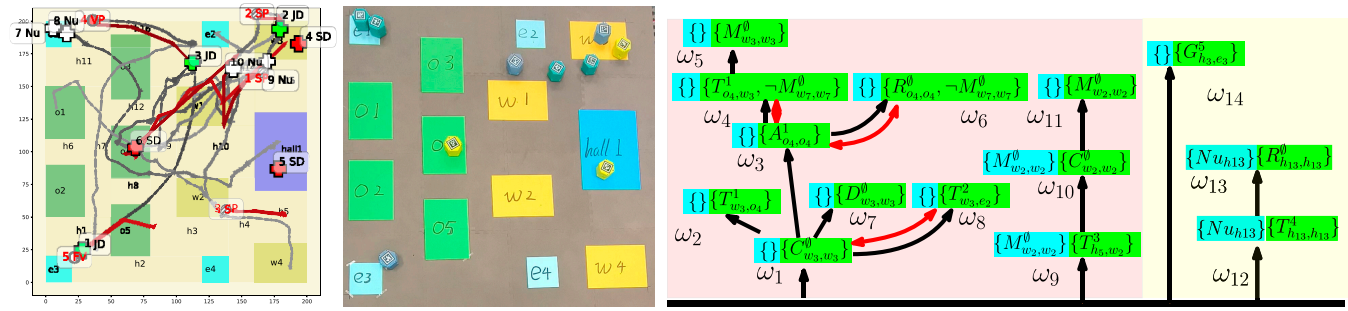

Fig. 7. The trajectories of agents in experiment (left), experimental snapshot (middle), and the final R-poset (right).

if it holds that $\inf(\rho) \cap S_F \neq \emptyset$, where $\inf(\rho)$ is the set of states that appear in ρ infinitely often. A special class of LTL formula is called sc-LTL [23,53], 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. A relaxed partially ordered set (R-poset) over an NBA $\mathcal{B}_\varphi$ is defined as follows:

Definition 2 (R-poset) [54] R-poset is a three-tuple: $P_\varphi = (\Omega_\varphi, \preceq_\varphi, \neq_\varphi)$: $\Omega_\varphi = \left\{ \left(\ell, \sigma_\ell, \sigma_\ell^s\right), \forall \ell = 0, \ldots, L \right\}$ is the set of subtasks, 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 Definition 1.

$\preceq_\varphi \subseteq \Omega_\varphi \times \Omega_\varphi$ is the "less equal" relation: 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. $\neq_\varphi \subseteq 2^{\Omega}{}_\varphi$ is the "opposed" relation: If $\{\omega_h, \cdots, \omega_\ell\} \in \neq_\varphi$ or equivalently $\omega_h \neq_\varphi \cdots \neq_\varphi \omega_\ell$, subtasks in $\omega_h, \cdots, \omega_\ell$ cannot all be executed simultaneously.

A word is accepting if it satisfies all the constraints imposed by P_φ. Additionally, the word of R-poset will satisfy the NBA as $Words(P_\varphi) \subset Words(\varphi)$.

Definition 3 (Language of R-poset) [54] Given a word $w = \sigma'_1 \sigma'_2 \cdots$ satisfying R-poset $P_\varphi = (\Omega_\varphi, \preceq_\varphi, \neq_\varphi)$, denoted as $w \vDash P_\varphi$, it holds that (a) given $\omega_{i_1} = \left(i_i, \sigma_{i_1}, \sigma^s_{i_1}\right) \in \Omega_\varphi$, there exist j_1 with $\sigma_{i_1} \subseteq \sigma'_{j_1}$ and $\sigma^s_{j_1} \cap \sigma'_{m_1} = \emptyset, \forall m_1 < j_1$; (b) $\forall \left(\omega_{i_1}, \omega_{i_2}\right) \in \preceq_\varphi, \exists j_1 \leq j_2, \sigma_{i_2} \subseteq \sigma'_{j_2}, \forall m_2 < j_2, \sigma^s_{j_2} \subseteq \sigma'_{m_2} = \emptyset$; (c) $\forall \left(\omega_{i_1}, \ldots, \omega_{i_n}\right) \in \neq_\varphi, \exists \ell \leq n, \sigma_{i_\ell} \subseteq \sigma'_{j_1}$. Language of R-poset P_φ is the set of all word w that satisfies P_φ, denoted by $\mathcal{L}\left(P_\varphi\right) \triangleq \left\{ w \mid w \vDash P_\varphi \right\}$.

Assuming that $\mathcal{P}_{b_i} = \left\{ P_1^{b_i}, P_2^{b_i}, \cdots \right\}$ is the set of all possible R-posets of NBA $\mathcal{B}_{b_i}$, it is shown in Lemma 3 of our previous work [54] that (a) $\mathcal{L}\left(P_j^{b_i}\right) \subseteq \mathcal{L}\left(\mathcal{B}_{b_i}\right), P_j^{b_i} \in \mathcal{P}_\varphi^{b_i}$ and (b) $\bigcup_{P_j^{b_i} \in \mathcal{P}_\varphi^{b_i}} \mathcal{L}\left(P_i\right) = \mathcal{L}\left(\mathcal{B}_{b_i}\right)$. In other words, the R-posets contain the necessary information for subsequent steps.

Problem formulation

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 $\mathcal{A}_a = \cup_{l \in L} \mathcal{A}_l = \left\{ a_1, \cdots, a_{n_c} \right\}$. Without loss of generality, the agents can navigate between each region via the same transition graph, i.e., $\mathcal{G} = \left(\mathcal{W}, \to_\mathcal{G}, d_\mathcal{G}\right)$ where $\to_\mathcal{G} \subseteq \mathcal{W} \times \mathcal{W}$ 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:

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

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 $t \geq t_u$; and $W_u^P\left(t_u\right) \subseteq W$ 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

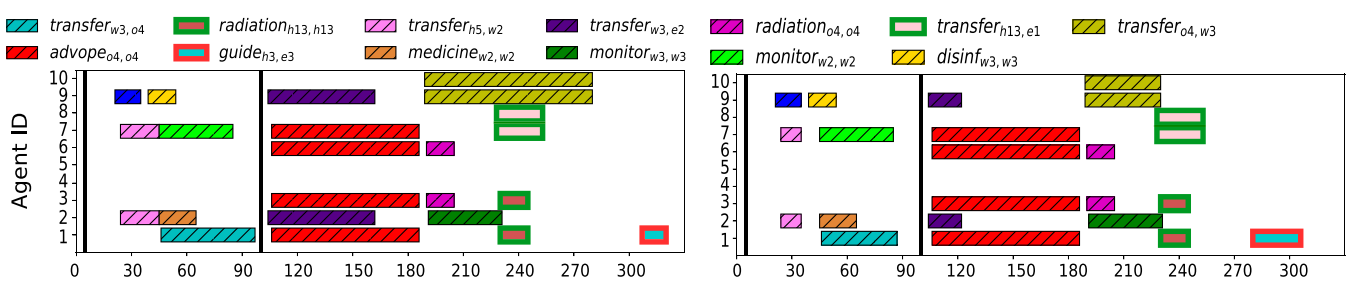

Fig. 8. The Gantt graph of the planned execution (left) and the Gantt graph of the actual execution (right).

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., $\mathcal{O} = \mathcal{O}_{in} \cup \mathcal{O}_{on}$.

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

$$C_k \triangleq \left(o_{u_k}, \{(a_i, n_i), i \in L_k\}\right) \tag{2}$$

where $o_{u_k} \in \mathcal{O} \cup \{\emptyset\}$ is the interactive object if any; $a_i \in \mathcal{A}_a$ is the set of cooperative actions required; $0 < n_i \leq N$ 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.

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 transportation behaviors. Related works [50,57] 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.

Task specifications

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

$$\varphi = \left(\bigwedge_{i \in I} \varphi_{b_i}\right) \wedge \left(\bigwedge_{o_u \in \mathcal{O}_{on}} \varphi_{e_u}\right), \tag{3}$$

where $\{\varphi_{b_i}, i = 1, \ldots, I\}$, $\{\varphi_{e_u}, o_u \in \mathcal{O}_{on}\}$ are two sets of sc-LTL formulas over $\{\mathbf{p}, \mathbf{c}\}$. φ_{bi} is specified in advance, while φ_{eu} is generated online when a new object o_u is added to $\mathcal{O}_{on}$.

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

$$\mathcal{J} = \left(J_1, \ J_2, \ \cdots, \ J_N\right), \tag{4}$$

where $J_n \in \mathcal{J}$ is the sequence of $\left(t_k, \ C_k, \ a_k\right)$, which means that agent n executes behavior 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:

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

where $J_u^o = \left(t_k, \ C_k\right) \cdots$ is the sequence of tasks associated with object $o_u \in \mathcal{O}$. The task pair $(t_k, \ C_k)$ is added to J_u^o if object o_u joins behavior C_k at time t_k. Assuming that the duration of formula φ_{bi}, φ_{ei} from being published to being satisfied is given by D_i, the average efficiency is defined as

$$\eta \triangleq \frac{\sum_{C_k \in J} |\boldsymbol{L}_k| \, d_k}{D_i}, \tag{6}$$

which is the percentage of time when actions are performed.

Problem 1 Given the sc-LTL formula φ defined in Eq. 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 [42,50] and online adaptation [51].

Approach

As shown in Fig. 9, when a new R-poset is generated, the proposed solution realizes the requirement through two main components: (a) product of R-posets, where the product of existing R-posets is computed incrementally, and (b) task assignment, where subtasks are assigned to the agents given the temporal and spatial constraints specified in the R-poset.

Product of R-posets

As the first two steps shown in Fig. 9, when a new formula $\varphi_i \in \Phi_b, \Phi_u$ is added, it will be transformed into NBA first. Then, an R-poset P_i is generated by the method proposed in our previous work [54]. The other R-poset P_{final} involved in calculation is the previous result of Poset-prod, which will be updated after this round calculation. With P_{final} as $P_1 = (\Omega_1, \preceq_1, \neq_1)$, P_i as $P_2 = (\Omega_2, \preceq_2, \neq_2)$, we define product of R-posets as follows:

Definition 4 (Product of R-posets) Given a finite word w_0, the 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 $\mathcal{P}^r = P_1 \otimes P_2$ where P_i' satisfies two conditions: (a) if $w_0 w \in \mathcal{L}(P_1), w \in \mathcal{L}(P_2)$, then $w_0 w \in \bigcup_{P_i' \in \mathcal{P}^r} \mathcal{L}(P')$; (b) if $w_0 w \in \mathcal{L}(P_i'), P_i' \in \mathcal{P}^r$, then $w_0 w \in \mathcal{L}(P_1), w \in \mathcal{L}(P_2)$.

The word w_0 is already executed, containing the finished subtasks Ω_{finish}^1 of P_1. Thus, w_0 can not influence P_2 whose subtasks will be executed in the future. Specially, if the new formula is offline as $\varphi_i \in \Phi_b$ and the agents have not started to execute subtasks, w_0 will be set as empty. As shown in Fig. 9, Poset-prod consists of the following two steps.

(a) Task composition. In this step, we generate all possible combinations of subtasks that can satisfy both Ω_1 and Ω_2. 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 $\omega_j' = \left(j, \sigma_j', \sigma_j^{s\prime}\right) \in \Omega'$ satisfying $\sigma_i^1 \subseteq \sigma_j'$ and $\sigma_i^{s1} \subseteq \sigma_j^{s\prime}$, or $\omega_j' \vDash \omega_i^1$ for brevity. Thus, we set $\Omega' = \Omega_1$ first and Ω' clearly satisfies Ω_1 with $\forall \omega_i^1 \in \Omega_1, \omega_i' \vDash \omega_i^1$. Then, a mapping function $M_\Omega : \Omega_2 \rightarrow \Omega'$ 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 and $\mathcal{R}(M_\Omega)$ 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 $\omega_j' \vDash \omega_i^2$, 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 $\mathcal{D}(M_\Omega)$, and ω_j' into $R(M_\Omega)$, where M_Ω^{-1} is the inverse function of M_Ω.

DFS [58] is used to add the subtasks of Ω_2 to Ω' in order, and these mapping relations will be recorded in M_Ω. The search sequences of DFS can be initialized as $que = [(\Omega' = \Omega_1, M_\Omega = \emptyset)]$. Then, during the circle, we will fetch the first

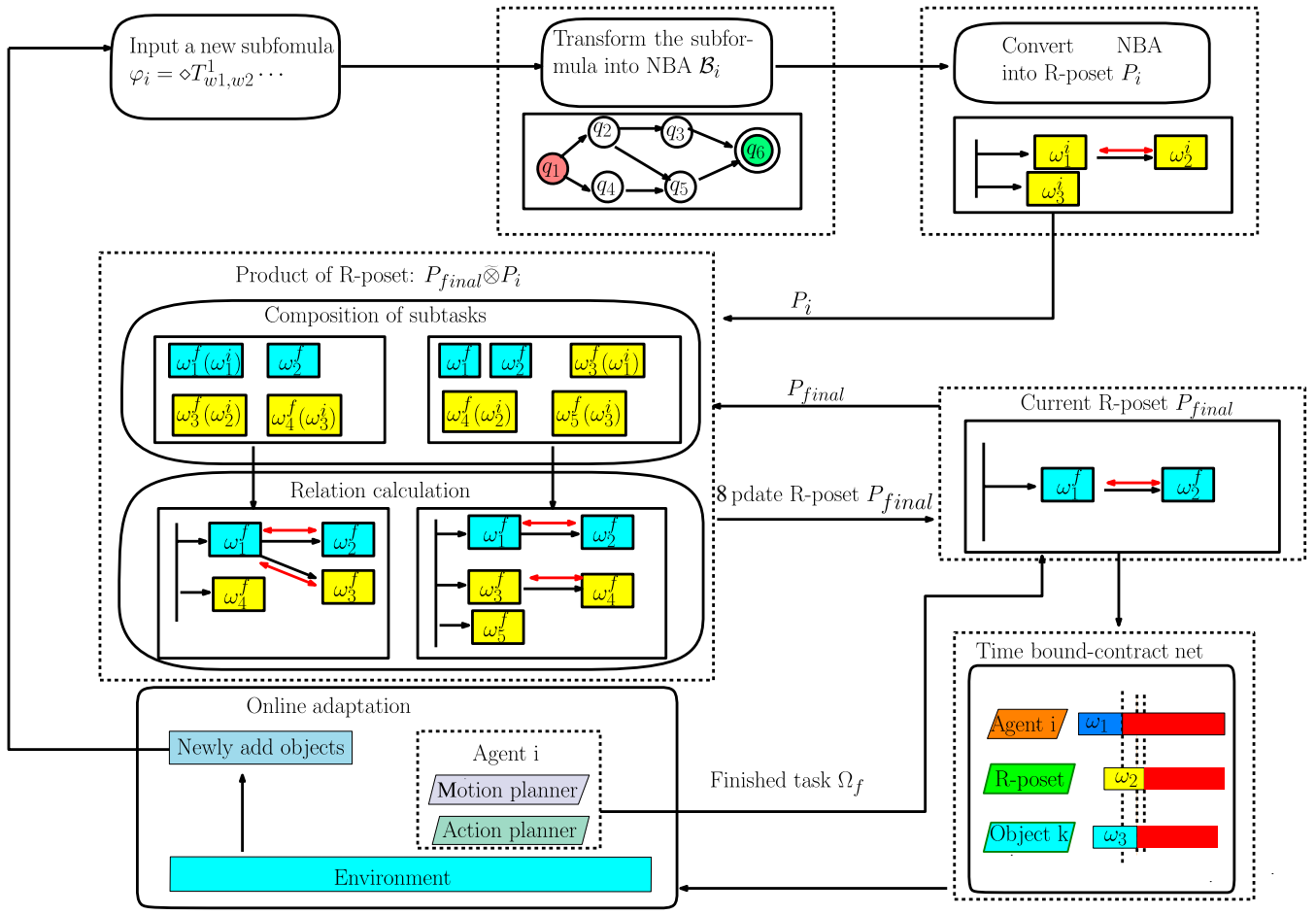

Fig. 9. Framework of the proposed method. For a new input formula φ_i, we first change it into an R-poset P_i with the method in the "Preliminaries" section. Then, Poset-prod between P_i and P_{final} is calculated as in the "Product of R-posets" section. Third, after the R-poset P_{final} is updated, the TBCN method in the "Task assignment" section determines and adjusts the task sequence of each agent and object. New tasks are released when the agents execute their local plans and detect new objects online.

node of que as (Ω', M'_Ω). The next unmixed subtasks in Ω_2 are $\omega_i^2, i = |M'_\Omega| + 1$. Any subtask $\omega_j^1 \in \Omega_1 / R(M'_\Omega) / \Omega^1_{finish}$ with $\omega_j^1 \vDash \omega_i^2$ or $\omega_i^2 \vDash \omega_j^1$ can create a new combination based on (Ω', M'_Ω):

$$\begin{aligned} \hat{\Omega}' &= \Omega', \ \hat{\omega}'_j = \left(j, \sigma_j^1 \cup \sigma_i^2, \ \sigma_j^{s1} \cup \sigma_i^{s2}\right), \\ \hat{M}'_\Omega &= M'_\Omega, \ \hat{M}'_\Omega\left(\omega_i^2\right) = \omega'_j, \end{aligned} \tag{7}$$

which means that the subtask ω'_j in Ω' can satisfy both ω_j^1, ω_i^2. Moreover, for the subtask ω_i^2, we can always create a set of subtasks $\hat{\Omega}'$ and the corresponding mapping function $\hat{M}'_\Omega$ by appending ω_i^2 into $\hat{\Omega}'$ as $\hat{\omega}'_j$ such that $\hat{\omega}'_j \vDash \omega_i^2$ holds, i.e.,

$$\begin{aligned} &\hat{\Omega}' = \Omega', j = |\Omega'| + 1, \hat{\omega}'_j = \omega_i^2; \\ &\hat{M}'_\Omega = M'_\Omega, \hat{M}'_\Omega\left(\omega_i^2\right) = \omega'_j, \end{aligned} \tag{8}$$

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.

Once $|D(M'_\Omega)| = |\Omega_2|$, a Ω' satisfying Ω_1, Ω_2 is already found. In this case, the next step is triggered. As shown in Fig. 9, one of found combination is $M^1_\Omega(1) = 1, M^1_\Omega(2) = 3, M^1_\Omega(3) = 4, \omega_1^f$ is created by (7), and ω_3^f, ω_4^f is created by (8).

(b) Relation update. 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. First, we construct the "less-equal" constraint $\preceq$ as follows:

$$\preceq = \preceq_1 \cup \left\{ \left(M_\Omega\left(\omega^2_{\ell_1}\right), M_\Omega\left(\omega^2_{\ell_2}\right)\right) | \left(\omega^2_{\ell_1}, \omega^2_{\ell_2}\right) \in \preceq_2 \right\}, \tag{9}$$

which inherits both less equal relations $\preceq_1$ in P_1 and $\preceq_2$ in P_2. Then, we update Ω' 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 $(M_\Omega^{-1}(\omega_i), M_\Omega^{-1}(\omega_j)) \in \preceq_2$ while $(\omega_i^1, \omega_j^1) \in \preceq_1$ 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 newly added suf-subtasks from $\preceq_1, \preceq_2$ are defined as S_p^1, S_p^2, i.e.,

$$S_p^1 = \left\{ \omega_j | \left(\omega_i, \omega_j\right) \in \preceq, \left(\omega_i^1, \omega_j^1\right) \in \preceq_1 \right\},$$
$$S_p^2 = \left\{ M_\Omega^{-1}\left(\omega_j\right) | \left(\omega_i, \omega_j\right) \in \preceq, \left(M_\Omega^{-1}\left(\omega_i^2\right), M_\Omega^{-1}\left(\omega_j^2\right)\right) \in \preceq_2 \right\}, \tag{10}$$

where S_p^1, S_p^2 are the subtasks that should be executed after ω_i. 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:

$$\hat{\sigma} = \bigcup_{\omega_\ell^1 \in S_p^1} \sigma_\ell^{s1} \cup \bigcup_{\omega_\ell^2 \in S_p^2} \sigma_\ell^{s2}, \sigma_i^s = \hat{\sigma} \cup \sigma_i^s, \sigma_i = \hat{\sigma} \cup \sigma_i, \tag{11}$$

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.

Finally, we find the potential ordering by checking whether a subtask σ_i^s is in conflict with another subtask while $(\omega_i^1,\ \omega_j^1) \in \preceq_1$. If so, an additional ordering will be added to $\preceq$ as:

$$\preceq = \preceq \cup \left\{ (\omega_i,\ \omega_j) | \sigma_j \vDash \sigma_i^s \right\} \tag{12}$$

Then, Ω will be updated following Eqs. 10 and 11. Regarding the set of subtasks Ω that have no conflicts in $\preceq$, its "not-equal" relations $\neq$ are generated by a simple combination as:

$$\neq = \neq_1 \cup \left\{ \left\{ M_\Omega\left(\omega_{\ell_i}\right) \right\} | \left\{ \omega_{\ell_i} \right\} \in \neq_2 \right\}. \tag{13}$$

The resulting poset $P_i = (\Omega,\ \preceq, \neq)$ is added to $\mathcal{P}_{final}$.

As shown in Fig. 9, Relation update gets two R-posets and the partial relations of each R-poset are succeeded from P_i, P_{final}. Due to the anytime property, the two-step procedure can be repeated until all possible R-posets are found or just ended when the first R-poset is found.

Task assignment

To satisfy the final R-poset $P_{final} = (\Omega_f, \preceq_f, \neq_f)$, the subtasks Ω_f should be executed under the partial orders of $\preceq_f$, $\neq_f$. Each subtask $\omega_i \in \Omega_c$ represents a collaborative behavior C_j. Thus, we can redefine the action sequence of each agent $J_u^o \in \mathcal{J}_o$ as $J_u^o = [(t_k,\ \omega_k, a_k),\ \cdots]$ 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 $C_k \in \sigma_k$.

We propose a suboptimal algorithm called time bound contract net (TBCN) to generate and adapt the action sequence of agents and objects. Compared with the classical contract net method [55], the main differences are as follows: (a) the partial order $\preceq_f$, $\neq_f$ of R-poset might be changed when new formula is added, (b) the assigned subtasks should satisfy the partial orders in $\preceq_f$, $\neq_f$, (c) the cooperative task should be fulfilled by multiple agents, and (d) interactive objects should be considered as an additional constraints. TBCN solves these differences with three steps: First, we design a cancellation mechanism in the initialization to adapt to the changes of R-poset mentioned in (a). Second, the partial orders in (b) are guaranteed by only assigning feasible subtasks but not all unassigned subtasks in each loop. Third, the constraints mentioned in (c) and (d) are considered as a time bound $t_1 \in \mathbb{R}^+$ in the bidding process.

(a) Initialization: Once the R-poset P_{final} is updated by Posetprod, we first collect the action sequence $\mathcal{J}, \mathcal{J}_o$ of previous solution and the set of finished subtasks Ω_{finish} from executing word w_0. Specially, all of them will be empty if it is the first round. Then, a set of essential conflict subtasks Ω_{ec} is defined to collect the subtasks that might conflict the updated partial orders $\preceq_f, \neq_f$:

$$\begin{aligned} \Omega_{ec} = & \left\{ \omega_i | \omega_i \preceq_f \omega_j, t_i \geq t_j, \forall \omega_i, \omega_j \in \Omega_1 / \Omega_{finish} \right\} \cup \\ & \{ \omega_i | \omega_i \neq_f \omega_j, t_j \leq t_i \leq t_j + d_j, \\ & \forall \omega_i, \omega_j \in \Omega_1 / \Omega_{finish} \}. \end{aligned} \tag{14}$$

Then, we compute the set of subtasks Ω_{conf} that should be removed from $\mathcal{J}, \mathcal{J}_o$:

$$\begin{aligned} \Omega_{conf} = & \{ \omega_j | \omega_i \preceq_f \omega_j, \forall \omega_i \in \Omega_1 / \Omega_{finish}, \\ & \forall \omega_j \in \Omega_{ec} \} \cup \Omega_{ec}, \end{aligned} \tag{15}$$

in which Ω_{conf} are the subtasks in Ω_{ec} and the subtasks whose pre-subtasks will be removed. With the action sequences $\mathcal{J}, \mathcal{J}_o$ removing all the subtasks in Ω_{conf}, we can initialize the set of assigned subtasks $\Omega_{as} = \{ \omega_i | \forall (t_i,\ \omega_i,\ a_i) \in \mathcal{J} \}$ and the set of unassigned subtasks $\Omega_u = \Omega_f / \Omega_{as}$.

(b) Computation of feasible subtasks: After initialization, the algorithm starts a loop to assigned the subtasks in Ω_u: getting a set of feasible subtasks Ω_{fe}; calculating their time bounds; choosing the best subtask. For the ordering constraints $\preceq_c$, if $(\omega_j, \omega_i) \in \preceq_c$, assigning $(t_{kj}, \omega_j, a_{kj})$ to a task sequence $J_i = \cdots (t_{ki}, \omega_i, a_{ki})$ will violate such constraints. Thus, Ω_{fe} based on current Ω_{as} is defined as:

$$\Omega_{fe} = \left\{ \omega_i | \omega_i \in \Omega_u, \forall \left(\omega_j, \omega_i\right) \in \preceq_f, \omega_j \in \Omega_{as} \right\}, \tag{16}$$

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

(c) Online bidding: Then, we will try assigning each subtask in Ω_{fe} and only choose the one with the best result. Without loss of generality, we assume that subtask $\omega_i \in \Omega_{fe}$ requires a label $\left(C_k\right)_{k_1,k_2}^{u_k}$, which means that the agents need to execute the behavior C_k from region W_{k1} to region W_{k2} using object u_k. Any constraint mentioned in (b), (c), and (d) can be considered as a time bound $t_1 \in \mathbb{R}^+$, which means that such constraint can be satisfied after t_1. Here, we use three kinds of time bounds: the global time bound t_i^ω, the object time bound $t_{u_k}^o$, and the set of local time bounds $\boldsymbol{Ts}$. The global time bound t_i^ω is the time instance that the ordering constraints $\preceq_f$ and conflict constraints $\neq_f$ will be satisfied if behavior $\left(C_k\right)_{k_1,k_2}^{u_k}$ is executed after t_i^ω:

$$\begin{aligned} & t_i^\omega \geq t_j, \qquad \forall (j,i) \in \preceq_f, \omega_j \in \Omega_{as}, \\ & t_i^\omega \geq t_\ell + d_\ell, \ \forall \{ \omega_i, \omega_\ell, \cdots \} \in \neq_f, \omega_\ell \in \Omega_{as}. \end{aligned} \tag{17}$$

For the required object u_k, assuming its participated last task is $\mathcal{J}_{u_k}^o[-1] = \left(t_\ell, \omega_\ell\right)$ the object time bound $t_{u_k}^o$ should satisfy that:

$$W_u^p(t) = W_{k_1}, t \geq t_\ell + d_\ell, , \forall t \geq t_{u_k}^o, \tag{18}$$

which means that the object u_k will be at the starting region W_{k1} of the current behavior $\left(C_k\right)_{k_1,k_2}^{u_k}$ and ready for it after $t_{u_k}^o$. Additionally, we set $t_{u_k}^o = \infty = \infty$ if the object is not at W_{k1} after the action sequence $\mathcal{J}_{u_k}^o$, and we set $t_{u_k}^o = 0$ if the behavior

$(C_k)^{u_k}_{k_1,k_2}$ does not require object as $u_k = \varnothing$. The set of local time bound is defined as

$$T_s = \{(\mathcal{A}_n, t^a_n) \mid \mathcal{A}_n = \mathcal{A}_{M_{type}(n)} \cap \mathcal{A}_{C_k}, \\ t^a_n = t_\ell + d_\ell + d_{\mathcal{G}}(k_\ell, k_1), \forall n \in \mathcal{N}\}, \quad (19)$$

where $\mathcal{A}_n$ is the set of actions that agent n can provide for behavior C_k, t^a_n is the earliest time agent n can arrive region W_{k1}, $t_\ell + d_\ell$ is the time when the last subtask $\omega_\ell \in J_n - [1]$ has finished, $d_{\mathcal{G}}(k_\ell, k_1)$ is the cost of moving, and $W_{k\ell}$ is the goal region of ω_ℓ. $(\mathcal{A}_n,\ t^a_n)$ means agent n can begin behavior $(C_k)^{u_k}_{k_1,k_2}$ after time t^a_n 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 $\mathcal{J}^i, \mathcal{J}^i_o$ to minimize the ending time of subtask ω_i. The efficiency η of each assignment $\mathcal{J}^i, \mathcal{J}^i_o, \omega_i \in \Omega_u$ is calculated, and the subtask with maximum efficiency will be chosen. Afterward, 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 $\mathcal{J} = \mathcal{J}^i, \mathcal{J}_o = \mathcal{J}^i_o$ for the next iteration.

Correctness and completeness analysis

Theorem 1 (Correctness) Given two R-posets $P_1 = (\Omega_1, \preceq_1, \neq_1)$, $P_2 = (\Omega_2, \preceq_2, \neq_2)$ generated from $\mathcal{B}_1, \mathcal{B}_2$, we have $\mathcal{L}(P_j) \subseteq \mathcal{L}(P_1) \cap \mathcal{L}(P_2)$, where $P_j \in \mathcal{P}_{final}$, $\mathcal{P}_{final} = P_1 \otimes P_2$.

Proof 1 If a word $w = \sigma'_1\sigma'_2\cdots$ satisfies $P_j = (\Omega_j, \preceq_j, \neq_j)$, it satisfies the three conditions mentioned in Definition 3. In first condition, due to the step Task composition of Poset-prod, we can infer that for any $P_j \in \mathcal{P}_{final}$ there exists $\omega^1_n = (n, \sigma^1_n, \sigma^{s1}_n) \in \Omega_1$, with $\omega_n = (n, \sigma_n, \sigma^s_n) \in \Omega_j$. Thus, we have $\sigma^1_n \subseteq \sigma_n, \sigma^{s1}_n \subseteq \sigma^s_n$ and $\sigma^{s1}_{i_1} \cap \sigma'_{m_1} = \emptyset, \forall m_1 < \ell_1$, which indicates that w satisfies P_1 for condition 1. For the second condition, due to Eq. 9 in step Relation update, we have $\preceq_i \subseteq \preceq_j$. Thus, we can infer that w satisfies P_1 for the second condition: Any $\sigma^{s1}_{i_1} \cap \sigma'_{m_1} = \emptyset, \forall m_1 < \ell_1$, we have $(\omega_{i1}, \omega_{i2}) \in \preceq_j$, thus $(\omega^1_{i_1}, \omega^1_{i_2}) \in \preceq_1$ and $\exists \ell_1 \le \ell_2$,. Additionally, for the last condition, as the word w satisfied the $\neq_j$ order of P_j. We have $\neq_1 \subseteq \neq_j$ due to Eq. 13. Thus, the word w also satisfied the third condition. In the end, we can conclude that w satisfies P_1. In the same way, we can proof the w also satisfies P_2. Thus, $\sigma^1_{i_2} \subseteq \sigma_{i_2} \subseteq \sigma'_{\ell_2}$

Theorem 2 (Completeness) Given two R-posets P_1, P_2 getting from $\mathcal{B}_1, \mathcal{B}_2$, with enough time budget, Poset-prod 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 $\mathcal{L}(\mathcal{P}_{final}) = \mathcal{L}(P_1) \bigcap \mathcal{L}(P_2)$.

Proof 2 Due to Theorem 1, it holds that $\mathcal{L}(\mathcal{P}_{final}) \subseteq \mathcal{L}(P_1) \bigcap \mathcal{L}(P_2)$. Thus, we only need to show that $\mathcal{L}(P_1) \bigcap \mathcal{L}(P_2) \subseteq \mathcal{L}(\mathcal{P}_{final})$. Given a word $w = \sigma'_1\sigma'_2\cdots$ and $w \in \mathcal{L}(P_1) \bigcap \mathcal{L}(P_2)$, w satisfies the first condition in Definition 3 for both P_1 and P_2 that: $\forall \omega^1_{i_1} = (i_1, \sigma^1_{i_1}, \sigma^{s1}_{i_1}) \in \Omega_1$, there exists σ'_{j1} with $\sigma^1_{i_1} \subseteq \sigma'_{j_1}$ and $\sigma^{s1}_{j_1} \subseteq \sigma'_{m_1}, \forall m_1 < j_1$; $\forall \omega^2_{i_2} = (i_2, \sigma^2_{i_2}, \sigma^{s2}_{i_2}) \in \Omega_2$, there exists σ'_{j2} with $\sigma^2_{i_2} \subseteq \sigma'_{j_2}$ and $\sigma^{s2}_{j_2} \subseteq \sigma'_{m_2}, \forall m_2 < j_2$. If $j_1 = j_2$, the step in Eq. 8 of Task composition will generate a subtask $\omega_{i1} \in \Omega_j$ with $\omega_{i_1} \vDash \omega^1_{i_1}, \omega^2_{i_2}$, and $\sigma_{i_1} \subseteq \sigma'_{j_1}, \forall m_3 < j_1, \sigma^s_{j_1} \subseteq \sigma'_{m_3}$. If $j_1 \neq j_2$, the (7) will generate $\omega_{M_\Omega(i_2)} \vDash \omega^2_{i_2}$, with $\sigma_{M_\Omega(i_2)} \subseteq \sigma'_{j_2}, \forall m_4 < j_2, \sigma^s_{M_\Omega(i_2)} \subseteq \sigma'_{m_4}$. Thus, there exists $P_j \in \mathcal{P}_{final}$ that satisfies the first condition. For the second condition, $\preceq_j$ consisting of two parts generated by Eqs. 9 and 12 guarantees that $w \in \mathcal{L}(P_1) \cap \mathcal{L}(P_2)$. Moreover, Eqs. 10 and 11 guarantee that w does not conflict the self-loop constraints of P_1, P_2. Thus, the second condition is satisfied. Regarding the third condition, since $\neq_j = \neq_1 \cap M_\Omega(\neq_2)$ holds in Eq. 13, $\neq_j$ is naturally satisfied by w. In conclusion, for any $w \in \mathcal{L}(P_1) \cap \mathcal{L}(P_2)$, $w \in \mathcal{L}(P_{final})$ holds and vice versa. Thus, $\mathcal{L}(\mathcal{P}_{final})$ is equal to $\mathcal{L}(P_1) \bigcap \mathcal{L}(P_2)$.

Acknowledgments

Funding: This work was supported by the National Key R&D Program of China under grants 2022ZD0116401 and 2022ZD0116400 and the National Natural Science Foundation of China under grants U2241214, 62373008, 62203017, and T2121002.

Author contributions: Z. Li and M.G. initiated the idea. Z. Liu designed the algorithms and conducted the experiments. Z. Liu and M.G. wrote the first draft of the manuscript. Z. Liu, M.G., W.B., and Z. Li commented and revised the manuscript. Z. Li and W.B. supervised the work. All authors have read, commented, and approved the final manuscript.

Competing interests: The authors declare that they have no competing interests.

Data Availability

The authors confirm that the data supporting the findings of this study are available within the article.

Supplementary Materials

Movie S1
Tables S1 to S3

References

1. Jemal H, Kechaou Z, Ayed MB, Alimi AM. A multi agent system for hospital organization. *Intl J Mach Learn Comput*. 2015;5(1):51–56.
2. Cliff OM, Fitch R, Sukkarieh S, Saunders DL, Heinsohn R. Online localization of radio tagged wildlife with an autonomous aerial robot system. Paper presented at: Robotics: Science and Systems; 2015 Jul; Rome, Italy.
3. Zhang C, Hammad A, Bahnassi H. Collaborative multi-agent systems for construction equipment based on real-time field data capturing. *J Inform Technol Constr (ITcon)*. 2009;14:204–228.
4. Arai T, Pagello E, Parker LE. Advances in multi-robot systems. *IEEE Trans Robot Autom*. 2002;18:655–661.
5. Toth P, Vigo D. An overview of vehicle routing problems. In: *The vehicle routing problem*. Philadelphia, PA: SIAM; 2002. p. 1–26.

6. Fink J, Hsieh MA, Kumar V. Multi-robot manipulation via caging in environments with obstacles. Paper presented at: 2008 IEEE International Conference on Robotics and Automation. 2008; Pasadena, CA, USA.
7. Arm P, Waibel G, Preisig J, Tuna T, Zhou R, Bickel V, Ligeza G, Miki T, Kehl F, Kolvenbach H, et al. Scientific exploration of challenging planetary analog environments with a team of legged robots. Science. *Robotics*. 2023;8(80):eade9548.
8. Varava A, Hang K, Kragic D, Pokorny FT. Herding by caging: A topological approach towards guiding moving agents via mobile robots. Paper presented at: Robotics: Science and Systems; 2017; Cambridge, MA, USA.
9. Ozkan-Aydin Y, Goldman DI. Self-reconfigurable multilegged robot swarms collectively accomplish challenging terradynamic tasks. *Sci Robot*. 2021;6(56):eabf1628.
10. Ruan W, Duan H, Sun Y, Yuan W, Xia J. Multiplayer reach–avoid differential games in 3D space inspired by Harris' hawks' cooperative hunting tactics. *Research*. 2023;6:0246.
11. Kartik S, Murthy SR, C. Task allocation algorithms for maximizing reliability of distributed computing systems. *IEEE Trans. Comput*. 1997;46(6):719–724.
12. Agrawal P, Varakantham P, Yeoh W. Scalable greedy algorithms for task/resource constrained multi-agent stochastic planning. Paper presented at: Proceedings of the 25th International Joint Conference on Artificial Intelligence. 2016 July 9; New York.
13. Keshanchi B, Souri A, Navimipour NJ. An improved genetic algorithm for task scheduling in the cloud environments using the priority queues: Formal verification, simulation, and statistical testing. *J Syst Softw*. 2017;124:1–21.
14. Li J, Zhang R, Yang Y. Multi-AUV autonomous task planning based on the scroll time domain quantum bee colony optimization algorithm in uncertain environment. *PLOS ONE*. 2017;12(11):Article e0188291.
15. Wu H, Xiao R. Flexible wolf pack algorithm for dynamic multidimensional knapsack problems. *Research*. 2020;2020:1762107.
16. Yan M, Yuan H, Xu J, Yu Y, Jin L. Task allocation and route planning of multiple UAVs in a marine environment based on an improved particle swarm optimization algorithm. *EURASIP J Adv Signal Process*. 2021;1–23.
17. Biswas S, Anavatti SG, Garratt MA. Particle swarm optimization based co-operative task assignment and path planning for multi-agent system. Poster presented at: 2017 IEEE Symposium Series on Computational Intelligence (SSCI). 2017; Honolulu, Hawaii, USA.
18. Wells AM, Dantam NT, Shrivastava A, Kavraki LE. Learning feasibility for task and motion planning in tabletop environments. *IEEE Robot Autom Lett*. 2019;4(99):1255–1262.
19. Omidshafiei S, Pazis J, Amato C, How JP, Vian J. Deep decentralized multi-task multi-agent reinforcement learning under partial observability. Paper presented at: Proceedings of the 34th International Conference on Machine Learning. 2017; Sydney, Australia.
20. Liu M, Ma H, Li J, Koenig S. Task and path planning for multi-agent pickup and delivery. Paper presented at: Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS). 2019; Montreal, QC, Canada.
21. Zhang H, Du W, Shan J. Building cooperative embodied agents modularly with large language models. arXiv. 2023. arXiv:2307.02485.
22. Ruan J, Chen Y, Zhang B. Tptu: Task planning and tool usage of large language model based ai agents. arXiv. 2023. arXiv:2308.03427.
23. Baier C, Ketoen JP. *Principles of model checking*. Cambridge, MA: MIT Press; 2008.
24. Koymans R. Specifying real-time properties with metric temporal logic. *Real-Time Syst*. 1990;2:255–299.
25. Maler O, Nickovic D. Monitoring temporal properties of continuous signals. In: *International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems*. Heidelberg, Germany: Springer; 2004. p. 152–166.
26. Luo X, Kantaros Y, Zavlanos MM. An abstraction-free method for multirobot temporal logic optimal control synthesis. *IEEE Trans Robot*. 2021;37(5):1487–1507.
27. Guo M, Dimarogonas DV. Task and motion coordination for heterogeneous multiagent systems with loosely coupled local tasks. *IEEE Trans Autom Sci Eng*. 2016;14(2):797–808.
28. Luo X, Zavlanos MM. Temporal logic task allocation in heterogeneous multi-robot systems. *IEEE Trans Robot*. 2022;38(6):3602–3621.
29. Sahin YE, Nilsson P, Ozay N. Multirobot coordination with counting temporal logics. *IEEE Trans Robot*. 2019;36(4):1189–1206.
30. Jones AM, Leahy K, Vasile C. ScRATCHS: Scalable and robust algorithms for task-based coordination from high-level specifications. *Proc Int Symp Robot Res*. 2019;38(4):1–16.
31. Schillinger P, Bürger M, and Dimarogonas DV. Decomposition of finite LTL specifications for efficient multi-agent planning. Paper presented at: International Symposium on Distributed Autonomous Robotic Systems;2016; London, UK.
32. Schillinger P, Burger M, Dimarogonas DV. Simultaneous task allocation and planning for temporal logic goals in heterogeneous multi-robot systems. *Int J Robot Res*. 2018;37(7):818–838.
33. Kantaros Y, Zavlanos MM. Stylus*: A temporal logic optimal control synthesis algorithm for large-scale multi-robot systems. *Int J Robot Res*. 2020;39:812–836.
34. Yu X, Yin X, Li S, Li Z. Security-preserving multi-agent coordination for complex temporal logic tasks. *Control Eng Pract*. 2022;123(1):Article 105130.
35. Lli L, Chen Z, Wang H, Kan Z. Fast task allocation of heterogeneous robots with temporal logic and inter-task constraints. *IEEE Robot Autom Lett*. 2023;8(8): 4991–4998.
36. Bonnet J, Gleizes MP, Kaddoum E, Rainjonneau S, Flandin G. Multi-satellite mission planning using a self-adaptive multi-agent system. Paper presented at: 2015 IEEE 9th International Conference on Self-Adaptive and Self-Organizing Systems. 2015; Cambridge, MA, USA.
37. Yang Q, Luo Z, Song W, Parasuraman R. Self-reactive planning of multi-robots with dynamic task assignments. Paper presented at: 2019 International Symposium on Multi-Robot and Multi-Agent Systems (MRS). 2019; Boston, MA, USA.
38. Faroni M, Umbrico A, Beschi M, Orlandini A, Cesta A, Pedrocchi N. Optimal task and motion planning and execution for multiagent systems in dynamic environments. *IEEE Trans Cyber*. 2023;1–12.
39. Choudhury S, Gupta JK, Kochenderfer MJ, Sadigh D, Bohg J. Dynamic multi-robot taskallocation under uncertainty and temporal constraints. *Auton Robots*. 2022;46: 231–247.

40. Tian D, Fang H, Yang Q, Guo Z, Cui J, Liang W, Wu Y. Two-phase motion planning under signal temporal logic specifications in partially unknown environments. *IEEE Trans Ind Electron*. 2023;70(7):7113–7121.
41. Ben-Ari M. A primer on model checking. *ACM Inroads*. 2010;1(1):40–47.
42. Gastin P, Oddoux D. Fast LTL to Bu¨chi Automata Translation. Paper presented at: Proceedings of the 13th International Conference on Computer Aided Verification. 2002; Copenhagen, Denmark.
43. Ding X, Smith SL, Belta C, Rus D. Optimal control of Markov decision processes with linear temporal logic constraints. *IEEE Trans Automat Contr*. 2014;59(5):1244–1257.
44. Kloetzer M, Mahulea C. Accomplish multi-robot tasks via Petri net models. Paper presented at: 2015 IEEE International Conference on Automation Science and Engineering (CASE). 2015; Gothenburg, Sweden.
45. Leahy K, Serlin Z, Vasile CI, Schoer A, Jones AM, Tron R, Belta C. Scalable and robust algorithms for task-based coordination from high-level specifications (ScRATCHeS). *IEEE Trans Robot*. 2022;38(4):2516–2535.
46. Schillinger P, Burger M, Dimarogonas DV. Hierarchical LTL-task mdps for multi-agent coordination through auctioning and learning. *Intl J Robot Res*. 2019;153:104085.
47. Kantaros Y, Zavlanos MM. Sampling-based optimal control synthesis for multirobot systems under global temporal tasks. *IEEE Trans Automat Contr*. 2018;64(5):1916–1931.
48. Piterman N, Pnueli A, Sa'ar Y. Synthesis of reactive(1) designs. *J Comput Syst Sci*. 2006;78(3):364–380.
49. Vasilopoulos V, Kantaros Y, Pappas GJ, Koditschek DE. Reactive planning for mobile manipulation tasks in unexplored semantic environments. Paper presented at: International Conference on Robotics and Automation; 2021; Xi'an, China.
50. Verginis CK, Dimarogonas DV. *Multi-agent motion planning and object transportation under high level goals*. IFAC World Congress; Sydney, Australia, 2018.
51. Lacerda B, Parker D, Hawes N. Optimal and dynamic planning for Markov decision processes with co-safe LTL specifications. Paper presented at: 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems. 2014; Chicago, IL, USA.
52. Feyzabadi S, Carpin S. Multi-objective planning with multiple high level task specifications. Paper presented at: 2016 IEEE International Conference on Robotics and Automation (ICRA). 2016; Stockholm, Sweden.
53. Belta C, Yordanov B, Gol EA. *Formal methods for discrete-time dynamical systems*. Heidelbery, Germany: Springer; 2017.
54. Liu Z, Guo M, Li Z. Time minimization and online synchronization for multi-agent systems under collaborative temporal logic tasks. *Automatica*. 2024;159: Article 111377.
55. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Trans. Comput*. 1980;C-29(12):1104–1113.
56. Faruq F, Parker D, Laccrda B, and Hawes N. Simultaneous task allocation and planning under uncertainty. Paper presented at: 2018 IEEE/RSJ international conference on intelligent robots and systems (IROS). 2018; Madrid, Spain.
57. Verginis CK, Dimarogonas DV. Multi-agent motion planning and object transportation under high level goals. *IFAC-PapersOnLine*. 2017;50:15816–15821.
58. Kozen DC, Kozen DC. Depth-first and breadth-first search. *Design Anal Algorithms*. 1992;19–24.