Decentralized Multi-agent Coordination under MITL Specifications and Communication Constraints

Wei Wang, Georg Friedrich Schuppe, and Jana Tumova

***Abstract*— We propose a decentralized solution for high-level multi-agent task planning problems in environments with a communication network failure. In particular, we consider that the agents can only sense each other and communicate only within a limited radius, yet, they may need to collaborate to accomplish their tasks. These are given in Metric Interval Temporal Logic (MITL) which enables to capture complex task specifications involving explicit time constraints. To substitute for the lacking communication network, we propose to deploy an *agile robot* that transfers information between the *heavy-duty robots* executing the tasks. We propose an algorithm to decompose each MITL formula into an independent promise for the respective agent and an independent request for others. The agile robot systematically pursues heavy-duty robots to exchange requests. The heavy-duty robots use formal methods-based algorithms to compute path plans satisfying the independent promises and the received requests. While the plan computation for the agents is fully decentralized, the satisfaction of all tasks is guaranteed (if such plans are found). We present a series of illustrative simulation examples motivated by search and rescue scenarios.**

I. Introduction

Search and rescue missions in post-catastrophe environments are an example of where teams of autonomous robots can become truly useful. In such context, next to reliable hardware, sensing, perception, control, and other important aspects, multi-agent coordination to achieve complex tasks is one of the challenges that need to be addressed. In this work, we focus on three particular features of multi-agent coordination: dealing with complex tasks that may include explicit time constraints; dealing with communication network failures; and decentralization that supports robustness and scalability of the team coordination.

We consider teams of heavy-duty ground vehicles that are given tasks such as to visit a set of locations within a given time limit in order to gather information about damage, or to clean debris in collaboration with others. To support rigorous specification of such complex tasks with explicit time constraints, we consider Metric Interval Temporal Logic (MITL) as the specification language. We also consider that the robots can communicate only within each others' sensing radius and that their motion capabilities in a cell-partitioned environment are modeled as a Weighted Transition Systems (WTS). Centralized approaches to multi-agent path planning for WTS under MITL specifications are usually computationally intractable; hence, we aim to design a decentralized approach.

We propose to decompose the MITL specification formulas into fully independent subtasks – *promises* for the respective agent and *requests* for the other agents. We prove that if these are all satisfied, the original, possibly dependent, MITL specifications are all satisfied, too. To facilitate communication between the robots, we deploy an agile robot executing a pursuit-evasion strategy. We formulate conditions under which the agile robot is guaranteed to locate the heavy-duty robots and time bounds by which this happens. Eventually, heavy-duty robots receive all requests from others, update their task specification – which is now fully independent on others. We use UPPAAL [1] to synthesize a plan that provably satisfies the updated task – if such plan exists. Our solution is sound, i.e. if a plan is found for every robot, the plans together guarantee satisfaction of the original dependent MITL tasks while respecting the communication constraints. However, the solution is not complete, i.e. a plan might exist even if it is not found via our method.

1) Related Work: A variety of methods have been used to address multi-agent cooperation and control, including graph-based methods [2], [3], or learning methods [4], [5]. Formal-methods based approaches use temporal logics as a specification language, allowing to formulate complex missions, goals and constraints in a rigorous, rich, yet relatively user-friendly way. They also come with a correct-by-design synthesis algorithms allowing to automatically compute provably satisfying plans or control strategies. In particular, Linear temporal logic (LTL) has been used as a specification language for expressing high-level tasks in a number of recent works focused on multi-agent planning and control [6], [7], [8]. There is extensive literature focused on distributed or decentralized coordination, and/or control of multi-agent systems from temporal logic specifications. For instance, distributed motion coordination was studied in [9], [10], and [11], [12] achieved decentralized multi-agent coordination, but neither of them considered explicit time constraints. MITL [13], offers a rigorous, yet relatively user-friendly way to specify a rich set of various complex requirements and dependencies, including explicit time constraints. Complex timed-bounded MITL tasks are achieved in [14], [15], [16] but with the help of centralized techniques and approaches. On the other hand, multi-agent coordination with limited communication capabilities is considered in [17] and [18]. However, there are no explicit time constraints for the

Wei Wang, Georg Friedrich Schuppe and Jana Tumova are with the Division of Robotics, Perception and Learning, KTH Royal Institute of Technology, Stockholm, Sweden. {wei7, schuppe, tumova}@kth.se

This research has been carried out as part of the TECoSA Vinnova Competence Center at KTH Royal Institute of Technology and in addition been partly supported by the Swedish Research Council (VR) (project no. 2017-05102). The authors are also affiliated with Digital Futures.

tasks. [19] proposed a solution similar to ours for decentralized multi-agent coordination with time and communication constraints, but our method only needs to communicate twice within the sensing radius per agent rather than after each sampling time and can perform more complex and diverse cooperative tasks than navigation and collision avoidance.

2) Contributions: To address multi-agent coordination under time and communication constraints, this paper: (1) proposes a method to decompose time-constrained dependent tasks into independent ones, hence allowing for decentralized coordination; (2) shows how to use an agile robot to replace lacking communication networks; (3) enables complex coordination while explicit communication is limited to several exchanges of MITL formulas.

II. Preliminaries

In this section, we summarize the necessary notation and preliminaries for path synthesis that will be used afterwards.

Let $\mathbb{N}$, $\mathbb{W}$, $\mathbb{Q}_{\geq 0}$, and $\mathbb{R}$ denote the set of natural, whole, non-negative rational, and real numbers, respectively. Given a set S, $|S|$ is defined as its cardinality, $S^n = S \times \ldots \times S$ as n-fold Cartesian product, and 2^S as the set of all subset.

An infinite sequence of time values $\tau(\ell) \in \mathbb{Q}_{\geq 0}$ form a *time sequence* $\tau = \tau(0)\tau(1)\ldots$ satisfying monotonicity: $\tau(\ell) < \tau(\ell+1)$, $\forall \ell \in \mathbb{N}$, and progress: $\forall t \in \mathbb{Q}_{\geq 0}$, $\exists \ell \geq 1$, such that $\tau(\ell) > t$. Given a finite set of atomic propositions *AP*, a *timed word* $\omega = (\omega(0), \tau(0))(\omega(1), \tau(1))\ldots$ is an infinite sequence over the set *AP* where $\omega(0)\omega(1)\ldots$ is an infinite word over 2^{AP}, $\tau(0)\tau(1)\ldots$ is a time sequence with $\tau(\ell) \in \mathbb{Q}_{\geq 0}$, $\ell \in \mathbb{N}$ [13].

As a modeling formalism, we use weighted transition systems to model the dynamics of each robot.

Definition II-.1. *A* weighted transition system *(WTS) is a tuple* $\mathcal{T} = \langle S, S^{init}, Act, \delta, \mathfrak{t}, AP \rangle$, *where* S *is a finite set of states;* $S^{init} \subseteq S$ *is a set of initial states;* Act *is a set of actions; a transition relation is given as* $\delta \subseteq S \times Act \times S$*;* $\mathfrak{t} : \delta \to \mathbb{Q}_{\geq 0}$ *is a map which assigning a positive weight to each transition.*

Definition II-.2. *A* timed run $\rho = (\rho(0), \tau(0))(\rho(1), \tau(1))\ldots$ *of a WTS is an infinite sequence where* $\rho(0) \in S^{init}$, $\tau(0) = 0$, $(\rho(\ell), a(\ell), \rho(\ell+1)) \in \delta$, *and* $\tau(\ell+1) = \tau(\ell) + \mathfrak{t}(\rho(\ell), a(\ell), \rho(\ell+1))$, $\forall \ell \in \mathbb{N}$.

As a specification language, *Metric Interval Temporal Logic (MITL)* can express properties by specifying the time bounds of missions and constraints. The syntax of MITL formula over a set of atomic propositions *AP* is:

$$\varphi := True \mid p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \Diamond_I \varphi \mid \Box_I \varphi \mid \varphi_1 \, \mathcal{U}_I \varphi_2 \quad (1)$$

where $p \in AP$; $\neg$, $\wedge$ indicate the boolean negation and conjunction; $\Diamond$, $\Box$, and $\mathcal{U}$ are the temporal operators denoting $Eventually$, $Always$, and $Until$, respectively; I is a nonsingular timed interval $I = [a, b], a, b \in \mathbb{Q}_{\geq 0} \cup \{\infty\}$ with $a < b$. Here, we set *AP* as the set of states S of a WTS. $s \in AP$ is considered to be true if the WTS is in state s and false otherwise. MITL formulas are interpreted over timed words like the ones produced by a WTS. For instance, the satisfaction relation $(\omega, \ell) \models \Diamond_I \varphi$ says that, at time step $\tau(\ell)$, $\ell \in \mathbb{N}$ in a timed word ω, there has to exists a time step $\tau(\ell') - \tau(\ell) \in I$, $\ell' \geq \ell$, $\ell' \in \mathbb{N}$ satisfying $(\omega, \ell') \models \varphi$. For full semantics, we refer the reader to [19].

III. Problem Formulation

In this section, we focus on modeling the multi-agent system and formalizing the agents' specifications. We state the main problem and outline our approach.

A. System model

Consider a team of $N+1$ robots $\mathcal{R} := \{R_i : i \in [0, N]\}$ with N heavy-duty robots $\{R_i : i \in [1, N]\}$ and one agile light-duty robot R_0 operating in a workspace $\mathcal{W} \subseteq \mathbb{R}^2$ that is partitioned into cells. For simplicity of presentation, we assume that the workspace is a rectangle and partitioned into $W = \mathcal{N}_x \times \mathcal{N}_y$ square cells of equal size. Each cell $w \in \mathcal{W}$ is either obstacle-free and belongs to $\mathcal{W}_{free} \subseteq \mathcal{W}$ or intersects with an obstacle and belongs to $\mathcal{W}_{obs} \subseteq \mathcal{W}$.

Each heavy-duty robot R_i is assigned to a default *work region* $\mathcal{W}_i$, such that $\mathcal{W} := \mathcal{W}_1 \uplus \ldots \uplus \mathcal{W}_N$. The work regions are disjoint and respect the cell partitioning. The cells in work region $\mathcal{W}_i$ are denoted $w_1^{\mathcal{W}_i} \ldots w_{N_i}^{\mathcal{W}_i}$, where N_i is the total number of cells in $\mathcal{W}_i$. Each robot $R_i \in \mathcal{R}$ is abstracted as a point in the center of a cell it occupies and is associated with a distinct *sensing radius* r_i, within which it can sense and communicate with others. Heavy-duty robots consider sensing occlusion due to obstacles in the environment, i.e, all cells behind obstacles within the sensing radius are not sensed. We assume that obstacles do not affect sensing capabilities of the agile robot.

A state s_i of a robot R_i is determined by the cell it occupies. Let $R_{i'}@w_{j'}^{\mathcal{W}_{k'}}$ denotes that robot $R_{i'}$ occupies cell $w_{j'}^{\mathcal{W}_{k'}}$. In each state, a robot can move between the adjacent obstacle-free cells by taking action $North$, $South$, $East$, $West$, or $Idle$ to stay in its current cell. Ultimately, heavy duty robots may move outside their default work regions.

Example III-A.1. *Figure 1 shows a system with two heavy-duty robots* R_1, R_2, *and an agile robot* R_0. R_0 *and* R_1 *are located in* $\mathcal{W}_1$, *with current states* $R_0@w_{4E}^{\mathcal{W}_1}$ *and* $R_1@w_{2G}^{\mathcal{W}_1}$, *respectively. The state of* R_2 *in* $\mathcal{W}_2$ *is* $R_2@w_{8D}^{\mathcal{W}_2}$.

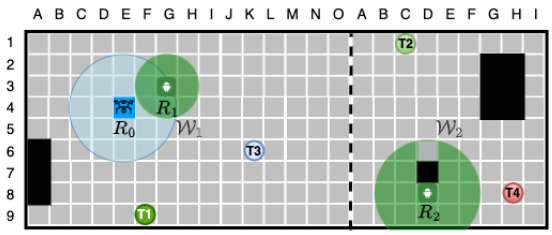


Fig. 1: The circular area is covered by the robot's sensing radius. T_1-T_4 are the cells of interest that need to reached within certain intervals. The solid black cells belong to $\mathcal{W}_{obs}$. Cell $w_{6D}^{\mathcal{W}_2}$ cannot be sensed by R_2 since it is occluded by $w_{7D}^{\mathcal{W}_2}$.

Formally, we use WTS to model the robots:

Definition III-A.1. *$R_i \in \mathcal{R}$, $i \in [0, N]$ is modeled by a WTS $\mathcal{T}_i = \langle S_i, s_i^{init}, Act_i, \delta_i, \mathfrak{t}_i \rangle$, where*

- *S_i is the set of states;*
- *$s_i^{init} \in S_i$ is an initial state;*
- *$Act_i = \{North, South, East, West, Idle\}$;*
- *$\delta_i \subseteq S_i \times Act_i \times S_i$ is the transition relation;*
- *$\mathfrak{t}_i : \delta_i \to \mathbb{Q}_{\geq 0}$ is a map that assign a positive weight to each transition, representing the transition time.*

The movement of a robot is captured by the timed run generated by the WTS (Definition II-.2). The team of heavy-duty robots produces *joint timed run*:

Definition III-A.2 (Joint timed run)**.** *Assuming timed runs $\rho_i = (\rho_i(0), \tau_i(0))(\rho_i(1), \tau_i(1)) \dots$ generated by WTS $\mathcal{T}_i$, $i \in [1, N]$, we define the* joint timed run *as $\rho_{joint} = (\varrho(0), \tau(0))(\varrho(1), \tau(1)) \dots (\varrho(k), \tau(k)) \dots$, where $\varrho(k) = \bigcup_{i \in [1,N]} \rho_i(j_i)$, and j_i is such that $\tau(k) \in [\tau_i(j_i), \tau_i(j_i+1))$.*

Example III-A.2. *The current state in the joint run in Fig. 1 is $(\{R_1@w_{2G}^{\mathcal{W}_1}, R_2@w_{8D}^{\mathcal{W}_2}\}, \tau(k))$ at time $\tau(k)$.*

Remark III-A.1. *For the sake of clarity, we use a grid-world in the examples, but our approach applies to any WTS, obtained e.g. through triangulation [20] or other cell partitioning methods [19], discrete abstraction of a robot's dynamics, and estimate of travel time of between the cells.*

B. Mission Specification

Each heavy-duty robot R_i, $i \in [1, N]$ is assigned a mission to accomplish that involves visiting cells of interests within certain time intervals. These are specified in a fragment of MITL as a formula φ_i:

$$\varphi_i := \varphi_i^{own} \wedge (\bigwedge_{j \neq i} \varphi_j^{collab}) \wedge \varphi_i^{meet} \tag{2}$$

where φ_i is over the states S_i, φ_i^{own} is an MITL formula over S_i that defines tasks to be completed independently by R_i itself. φ_j^{collab} is an MITL formula over S_j that defines tasks to be completed independently by other robots R_j, $j \neq i$. φ_i^{meet} defines tasks for which R_i and other robots R_j, $j \neq i$ need to meet and complete cooperatively:

$$\varphi_i^{meet} := \Diamond_I \psi \mid \Box_I \psi \mid \varphi_{i,1}^{meet} \wedge \varphi_{i,2}^{meet}, \psi := \bigwedge_{j \in [1,N]} p_j \tag{3}$$

where $p_j \in S_j \cup \{true\}$.

Remark III-B.1. *In formal methods-based planning literature, specifications are often defined over atomic propositions instead of states. In this paper, we consider that the state themselves are atomic propositions. This design choice allows us to express missions that we are interested in, while keeping the notation simple. Note however, that all methods presented in this paper are directly applicable to handling specifications over atomic propositions in general.*

Example III-B.1. *An example of mission specifications for the two robots in Fig.1 are: $\varphi_1 = \varphi_1^{own} \wedge \varphi_2^{collab} = \Diamond_{I_1}(R_1@w_{T_1}^{\mathcal{W}_1}) \wedge \Diamond_{I_3}(R_2@w_{T_3}^{\mathcal{W}_1})$, $\varphi_2 = \varphi_2^{own} \wedge \varphi_2^{meet} = \Box_{I_2}(R_2@w_{T_2}^{\mathcal{W}_2}) \wedge \Diamond_{I_4}(R_2@w_{T_4}^{\mathcal{W}_2} \wedge R_1@w_{T_4}^{\mathcal{W}_2})$. φ_1 indicates that R_1 should reach T_1 within interval I_1 and it requires R_2 to reach T_3 within I_3. φ_2 represents that R_2 should stay in T_2 during I_2 and it requires to meet R_1 at T_4 sometimes during I_4*

C. Problem Statement

Having the multi-agent system with limited sensing capability modeled as WTS, and a mission specified in the defined fragment of MITL, the problem can be formalized as follows:

Problem III-C.1. *Consider N heavy-duty robots and an agile robot modeled as weighted transition systems $\mathcal{T}_i$, $i \in [1, N]$ and T_0, respectively. Given MITL formulas φ_i in the form of Eq.(2) for each WTS $\mathcal{T}_i$, $i \in [1, N]$, synthesize a joint timed run ρ_{joint}, such that $\rho_{joint} \models \varphi_1 \wedge \ldots \wedge \varphi_N$ holds.*

Remark III-C.1. *A straightforward approach to this problem is to construct a product of all WTS $\mathcal{T} = \bigotimes_{i \in [1,N]} \mathcal{T}$, and use a conjunction of the individual MITL specifications as the specification for the overall multi-agent system. This approach requires the communication constraints are relaxed, it is centralized, sound and complete. We use it as a baseline to compare our approach to wrt computational complexity (in Sec.IV) and runtime in validation (in Sec.V).*

D. Approach

A schematic diagram of the proposed approach is given in Fig 2. Our approach contains the following steps:

Mission decomposition:

- **Step 1: Request and promise extraction** For each heavy-duty robot R_i, $i \in [1, N]$ operating in region $\mathcal{W}_i$ with mission specification φ_i of form (2), requests from tasks defined in φ_i^{collab} and φ_i^{meet} dependent on other robots R_j are extracted in the form of MITL formulas over S_j that they need to satisfy. See Sec IV-.1.
- **Step 2: Independent request and independent task** The mission specification φ_i is transformed into an independent task ϕ_i over S_i and several independent requests $\phi_{i,j}$ over S_j, $j \in [1, N], j \neq i$. If ϕ_i and $\phi_{i,j}$ are satisfied then φ_i is satisfied as well. See Sec IV-.2.

Execution:

- **Step 3: Request exchange** The agile robot R_0 deploys a pursuit-evasion strategy [21] that satisfies a certain transition weight relation to pursue the heavy-duty robots R_i, $i \in [1, N]$ in their work regions $\mathcal{W}_i$, collects and delivers $\bigwedge_{j \neq i} \phi_{i,j}$. After executing two rounds of pursuit-evasion strategy, all requests are exchanged within an upper-bounded time. See Sec IV-.3.
- **Step 4: Plan execution and update** Upon reception of requests, the heavy-duty robot R_j verifies whether the received request $\bigwedge_{j \neq i} \phi_{i,j}$ can be added to their specification and updates the timed run accordingly via plan synthesis. See Sec IV-.4.

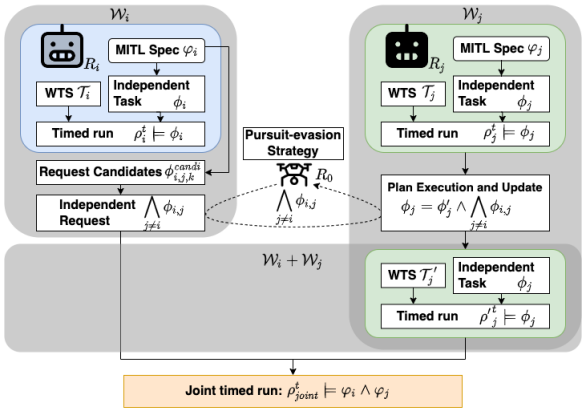


Fig. 2: A schematic diagram of the approach with two heavy-duty robots R_i, R_j and one agile robot R_0.

IV. Solution

1) Request and promise extraction: As the first step, robot $R_i, i \in [0, N]$ produces two types of *requests* to other robots: *collaboration requests* and *meet requests*. At the same time, it commits to satisfying a *promise*. The requests and the promises are given in MITL. While the satisfaction of the overall specification formula depends on the behavior of the whole team, the satisfaction of a request or a promise is dependent only on the behavior of a single agent. Specifically, given a mission specification formula φ_i (Eq. (2)), we define:

Definition IV-.1 (Collaboration request)**.** *The* collaboration request *from robot R_i to robot R_j is $\phi_{i,j}^{collab} = \varphi_j^{collab}$.*

Definition IV-.2 (Meet request candidate and meet promise candidate)**.** *Suppose $\varphi_i^{meet} = \varphi_{i,1}^{meet} \wedge \ldots \wedge \varphi_{i,\ell}^{meet}$, where $\varphi_{i,k}^{meet} = \star_{I_k}(\psi_{i,1,k}^{meet} \wedge \ldots \wedge \psi_{i,N,k}^{meet})$, $\star \in \{\Box, \Diamond\}$, $k \in \{1, \ldots, \ell\}$. The k-th* meet request candidate *from robot R_i to R_j, $i \neq j$, is $\phi_{i,j,k}^{candi} = \star_{I_k}\psi_{i,j,k}^{meet}$ and the corresponding* meet promise candidate *of robot R_i is $\phi_{i,i,k}^{candi} = \Box_{I_k}\psi_{i,i,k}^{meet}$.*

Example IV-.1. *Continuing with Example III-B.1, the collaboration request, the meet request candidate, and the meet promise candidate extracted from φ_1 are $\phi_{1,2}^{collab} = \Diamond_{I_3}(R_2@T_3^{\mathcal{W}_1})$, $\phi_{1,2,1}^{candi} = \Diamond_{I_4}(R_2@T_4^{\mathcal{W}_1})$, and $\phi_{1,1,1}^{candi} = \Box_{I_4}(R_1@T_4^{\mathcal{W}_1})$, respectively.*

We now formulate that the satisfaction of requests and promises are sufficient to satisfy the overall specification.

Lemma IV-.1. *Assume that the meet request candidate $\phi_{i,j,k}^{candi}$ and the meet promise candidate $\phi_{i,i,k}^{candi}$, $\forall j \in [1, N]$, $j \neq i$, are satisfied for all $k \in \{1, \ldots, \ell\}$. Then, the formula $\varphi_i^{meet} = \varphi_{i,1}^{meet} \wedge \ldots \wedge \varphi_{i,\ell}^{meet}$ is satisfied.*

Proof: Follows from the observation that if $\phi_{i,j,k}^{candi} = \star_{I_k}\psi_{i,j,k}^{meet}$ holds and $\phi_{i,i,k}^{candi} = \Box_{I_k}\psi_{i,i,k}^{meet}$ holds, then $\star_{I_k}(\psi_{i,i,k}^{meet} \wedge \psi_{i,j,k}^{meet})$ holds as well. Namely if $\star = \Diamond$ then there exists $t \in I_k$, such that $\psi_{i,j,k}^{meet}$ holds and at the same time $\phi_{i,i,k}^{candi}$ as well since it holds at all $t \in I_k$. Analogously, if $\star = \Box$ then for all $t \in I_k$, both $\phi_{i,j,k}^{candi}$ and $\phi_{i,i,k}^{candi}$ hold. ■

We further observe that satisfying the meet request candidates is a necessary condition to guarantee φ_i^{meet} and that satisfying the meet promise candidate $\phi_{i,i,k}^{candi}$ is a necessary condition to guarantee the subformula $\phi_{i,k}^{meet}$ of type $\Box_{I_k}(\psi_{i,1,k}^{meet} \wedge \ldots \wedge \psi_{i,N,k}^{meet})$. In contrast, it is not necessary to guarantee $\varphi_{i,k}^{meet} = \Diamond_{I_k}(\psi_{i,1,k}^{meet} \wedge \ldots \wedge \psi_{i,N,k}^{meet})$, since it presents a possibly stronger commitment for R_i than needed. R_i may not be able to satisfy the promise candidate, however, R_i may be able to satisfy a weaker version of it at the cost of sending a stronger request to other robots. To obtain the actual meet request and meet promise, we iteratively refine the time interval I_k. Once there exists a satisfying timed run of R_i, the promise candidate becomes the actually meet promise and the corresponding request candidate becomes the actual request. Formally, we define the *independent task candidate* as:

$$\varphi_i^{candi} = \varphi_i^{own} \wedge \Big(\bigwedge_{\forall k \in \{1,\ldots,l\}} \phi_{i,i,k}^{candi} \Big). \tag{4}$$

and verify whether there exists a timed run of R_i that satisfies the independent task candidate. This verification problem can be solved using off-the-shelf tools, such as UPPAAL [1].

To obtain the final meet requests and meet promises, we need to refine the meet request candidates and the meet promise candidates. This procedure is presented in Alg.1. If the meet formula is of type $\Box$ ($Always$), then the promise and request are directly the promise candidate and request candidate (line 1-2). Otherwise, we iteratively shorten interval I_k by moving its left boundary (line 3-15). In each iteration, we verify whether there exists a timed run satisfying the independent task candidate φ_i^{candi} with the shortened interval I_k' (line 8). If yes, then the current meet promise and request candidates become the actual meet promise and meet request (line 9), otherwise the candidates are updated (line 11).

Remark IV-.1. *Interval I_k' might not be the longest interval that R_i, $i \in [1, N]$ can promise to satisfy $\psi_{i,i,k}^{meet}$ of type $\Box$. However, the problem of computing a union of intervals such that $\rho_i \models \varphi_i^{own} \wedge \phi'^{candi}_{i,i,k}$, $\phi'^{candi}_{i,i,k} = \Box_{I_k''}\psi_{i,i,k}^{meet}$, $I_k'' = I_{k,1} \cup \ldots \cup I_{k,n}$, $n \in \mathbb{W}$, $I_{k,n} \subseteq I_k$, $I_{k,n} \cap I_{k,m} = \emptyset$ is NP-hard. Instead, we chose a computationally efficient suboptimal solution, where we compute the longest time interval with one side of the boundary fixed.*

2) Independent request and independent task: Having the requests and promises extracted, the next step in our solution is to integrate them in the robots' specifications. Each robot R_i builds an independent task that can be accomplished without help of other robots and sends requests to other robots that can be independently executed by them.

Formally, given a specification φ_i of the form of Eq.(2), where $\varphi_i^{meet} = \varphi_{i,1}^{meet} \wedge \ldots \wedge \varphi_{i,\ell}^{meet}$, $\varphi_{i,k}^{meet} = \star_{I_k}(\psi_{i,1,k}^{meet} \wedge \ldots \wedge \psi_{i,N,k}^{meet})$, $\star \in \{\Box, \Diamond\}$, the *independent task* specification formula for R_i is

$$\phi_i = \varphi_i^{own} \wedge \Big(\bigwedge_{\forall k \in \{1,\ldots,l\}} \phi_{i,i,k}^{meet} \Big). \tag{5}$$

Algorithm 1: Meet promise and request candidates refinement

Input : $\mathcal{T}_i$, $\phi_{i,i,k}^{candi} = \square_{[a_k,b_k]}\psi_{i,i,k}^{meet}$, $\phi_{i,j,k}^{candi} = \star_{[a_k,b_k]}\psi_{i,j,k}^{meet}$
Output: Meet promise $\phi_{i,i,k}^{meet}$, Meet request $\phi_{i,j,k}^{meet}$

```
if ⋆ = □ then
    return φ^meet_{i,i,k} = φ^candi_{i,i,k}, φ^meet_{i,j,k} = φ^candi_{i,j,k}
else
    a'_k = a_k
    while a'_k ≤ b_k do
        φ^candi_{i,i,k} = □_{I'_k} ψ^meet_{i,i,k}, I'_k = [a'_k, b_k]
        φ^candi_i = φ^own_i ∧ φ^candi_{i,i,k}
        if ∃ρ_i ⊨ φ^candi_i (by UPPAAL) then
            return φ^meet_{i,i,k} = φ^candi_{i,i,k}, φ^meet_{i,j,k} = ◇_{I'_k} ψ^meet_{i,j,k}
        else
            a'_k = a'_k + 1
        end
    end
    return "Mission impossible"
end
```

The independent request from R_i to R_j is:

$$\phi_{i,j} = \phi_{i,j}^{collab} \wedge \Big(\bigwedge_{\forall k \in \{1,\dots,\ell\}} \phi_{i,j,k}^{meet} \Big). \tag{6}$$

The following lemma states that satisfaction of all independent tasks and requests guarantees satisfaction of the overall mission.

Lemma IV-.2. *Assume that there exists a timed run of R_i such that the independent task ϕ_i is satisfied and a timed run of R_j such that the independent requests $\phi_{i,j}$ are satisfied $\forall j \in [1, N]$, $j \neq i$. Then, the formula φ_i (see Eq.2) is satisfied by the joined timed run.*

Proof: Analogous to the proof of Lemma IV-.1, if ϕ_i, $\phi_{i,j}$ hold, then $\phi_i \wedge \phi_{i,j}$, $\forall j \in [1, N]$, $j \neq i$ holds. Namely all of φ_i^{own}, $(\bigwedge_{j \neq i} \varphi_j^{collab})$, φ_i^{meet} hold, which φ_i holds. ■

Given the independent task specification ϕ_i of R_i, $i \in [1, N]$, a satisfying timed run ρ_i of the WTS $\mathcal{T}_i$ is synthesized using UPPAAL. As the independent task specifications concern only states in $\mathcal{W}_i$, we remove from $\mathcal{T}_i$ transitions leading to states outside of $\mathcal{W}_i$. The heavy-duty robots then start executing the synthesized timed run and we proceed with step 3 of our approach (Sec. III-D).

3) Request exchange: While each robot R_i, $i \in [1, N]$ keeps executing its timed runs independently, the generated independent requests $\phi_{i,j}$, $j \in [1, N], j \neq i$ are exchanged through agile robot R_0, which replaces the missing communication network. R_0 deploys a pursuit-evasion strategy, which is used to search large unknown areas without any information about the position of heavy-duty robots [22]. This strategy follows a serpentine pattern as shown in Figure 3. The *hatch distance* h_a in Figure 3 (a) is the distance between the two successive scan paths along the x-axis in the serpentine pattern [23], and r_0 is the sensing radius of R_0. The following lemma gives a relationship between the moving speed of the agile robot and the time it takes to cover an area, which follows directly from Lemma III.2 in [21].

Lemma IV-.3. *Given a rectangular, partitioned work region $\mathcal{W}_i \in \mathcal{W}$ with $W_i = \mathcal{N}_{x_i} \times \mathcal{N}_{y_i}$ cells, the upper time bound for R_0 to completely scan and cover $\mathcal{W}_i$ is:*

$$t_i^{pe} = (\lceil (\mathcal{N}_{x_i} + h_a) \times t_0^{max} \rceil \times \left\lceil \frac{\mathcal{N}_{y_i} - 2r_0}{h_a} \right\rceil \tag{7}$$

where r_0 is the sensing radius of R_0, $t_0^{max} \in \mathbb{Q}_{\geq 0}$ is the maximum weight of transitions in WTS $\mathcal{T}_0$.

Choosing an appropriate hatch distance $0 < h_a \leq 2r_0$ guarantees the coverage of the whole work region. The shorter h_a, the faster R_0 completes a back-and-forth trip in the serpentine pattern and the more back-and-forth trips are needed to cover $\mathcal{W}_i$. In [21], the authors assume that a pursued agent does not move until it has been located by the pursuer, unlike our heavy-duty robots that keep moving. Therefore, given a fixed hatch distance h_a, the agile robot R_0 needs to travel at a certain minimum speed to guarantee heavy-duty robot R_i is pursued within t_i^{pe} time steps, captured by the following lemma. Analogously, given a fixed maximum speed of the agile robot, one can derive a minimum hatch distance from the following lemma. If the hatch distance is negative, the agile robot cannot guarantee to pursue the heavy-duty robot using a single pass of a pursuit-evasion strategy. The following lemma gives a relationship of the moving speed between the agile robot and heavy-duty robots.

Lemma IV-.4. *Given a rectangular work region $\mathcal{W}_i \in \mathcal{W}$ with $W_i = \mathcal{N}_{x_i} \times \mathcal{N}_{y_i}$ cells, the upper bound on the weight of transitions in agile robot WTS $\mathcal{T}_0$ is*

$$t_0^{max} \leq \frac{(2r_0 - h_a) \times t_i^{max}}{2\mathcal{N}_{x_i} + h_a} \tag{8}$$

where $t_i^{max} \in \mathbb{Q}_{\geq 0}$ is the maximum weight of transitions between cells in WTS $\mathcal{T}_i$, $i \in [1, N]$.

Proof: The pursuit-evasion strategy divides the work region $\mathcal{W}_i$ into swept cells and unswept cells, and eventually, all cells in $\mathcal{W}_i$ have to be swept. The only scenario that R_i is not located by R_0 after $\mathcal{W}_i$ is fully swept is when R_i moved to the swept cells without being located by R_0.The maximum time from an unswept cell $w_j^{\mathcal{W}_i}$ which is adjacent to any swept cells to be swept is $(2\mathcal{N}_{x_i} + h_a) \times t_0^{max}$. If R_i cannot move from the unswept cell $w_j^{\mathcal{W}_i}$ to swept cells without being sensed by R_0, R_i cannot move from any unswept cells to swept cells without being located by R_0, which the distance R_0 traveled is no more than $(2r_0 - h_a)$. Therefore, R_i is guaranteed to be pursued by R_0 if Eq.(8) is satisfied. ■

In Fig.3 shading indicates the swept cells and the black dotted line with arrows represents the trajectory of R_0 the trajectory of R_0 before time t under the pursuit-evasion strategy. At time step t, R_i is located in the cell adjacent to

the swept cells that takes the longest time to transit from an unswept cell to a swept cell. In Fig.3 (b), the blue dotted line with arrows represents the trajectory of R_0 between time t and t' under the pursuit-evasion strategy. Similarly, the green dotted line with arrows represents the trajectory of R_i in the same time interval. To ensure the pursuit and communication, when R_i moves $(2r_0 - h_a)$ to swept cells, R_0 needs to travel at least $(2\mathcal{N}_{x_i} + h_a)$.

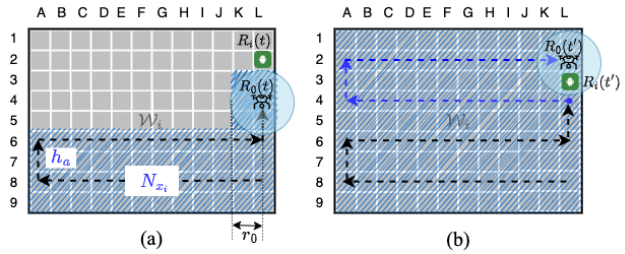


Fig. 3: Schematic of the pursuit-evasion strategy of agile robot R_0.

Once heavy-duty robot $R_i, i \in [1, N]$ is within the sensing radius of R_0 in $\mathcal{W}_i$, R_0 collects requests generated by R_i and then moves to the next closest unvisited work region $\mathcal{W}_j$ to pursue R_j. For simplicity, we assume R_0 has unlimited memory and that there is no communication delay. Pursuit of all heavy-duty robots $R_i, i \in [1, N]$ in their respective default work regions guarantees that R_0 collects all existing requests. After that, R_0 pursues all heavy-duty robots again via same strategy, in order to deliver all the requests. Hence, request delivery can be guaranteed via two rounds of pursuit-evasion strategy, and the upper time bound for R_0 to exchange all requests is formalized in the following Lemma. The following lemma defines an upper time bound for the agile robot to exchange all requests in a workspace.

Lemma IV-.5. *Assuming that the maximum pursuit-evasion time within a work region is t_{pe}^{max}, the maximum traveling time from a work region $\mathcal{W}_i$ to any adjacent $\mathcal{W}_j$ is t_w^{max}, and the maximum traveling time from $\mathcal{W}_N$ to $\mathcal{W}_1$ is t_N^{max}, the upper time bound for R_0 to exchange all requests among $R_i, i \in [1, N]$ is $t_{total} = 2Nt_{pe}^{max} + 2(N-1)t_w^{max} + t_N^{max}$.*

Proof: In the request collection/ delivery phase, the maximum traveling time of R_0 from $\mathcal{W}_i$ to the next adjacent unvisited work region is $t_w^{max} = (\mathcal{N}_{x_i} + \mathcal{N}_{y_i}) \times t_0^{max}$. From the request collection phase to the delivery phase, ie. $\mathcal{W}_N$ to $\mathcal{W}_1$, maximum $t_N^{max} = t_w^{max}$ is required if they are adjacent, otherwise, $t_N^{max} = (\mathcal{N}_x - \mathcal{N}_{x_1} + \mathcal{N}_y - \mathcal{N}_{y_1}) \times t_0^{max}$. ■

If multiple agile robots are available, each of them can follow the pursuit-evasion strategy, leading to the above maximum pursuit-evasion time being a conservative estimate. Coordination of multiple agile robots to systematically decompose the request exchange is left for future work.

4) Plan execution and update: The last step in our solution is to integrate the received requests into the robots' mission specifications, which enables coordination through independent executions. While the heavy-duty robots execute their timed runs, they will eventually (at latest at time t_{total})

Algorithm 2: Complete coordination Framework

Input : $\mathcal{T}_0, \ldots \mathcal{T}_N$, $\varphi_1 \ldots \varphi_N$
Output: $\rho_1, \ldots, \rho_N$
1 **forall** R_i, $i \in [1, N]$ *in parallel* **do**
2 Obtain ϕ_i as Eq.(5), $\bigwedge_{j \neq i} \phi_{i,j}$ as Eq.(6) (Alg.1)
3 **if** $\nexists\, \rho_i \models \phi_i$ **then**
4 **return** *"Solution does not exist"*
5 **else**
6 R_i starts executing ρ_i
7 **end**
8 **end**
9 R_0 sweeps to collect all requests
10 R_0 sweep to deliver all requests
11 **forall** R_i, $i \in [1, N]$ *in parallel* **do**
12 R_i recomputes ρ_i
13 **if** $\exists \rho_i \models \phi_i = \phi_i' \wedge \bigwedge_{j \neq i} \phi_{j,i}$ **then**
14 R_i executes the updated $\rho_i \models \phi_i$
15 **else**
16 R_i executes $\rho_i \models \phi_i$ from Eq.(5)
17 **return** *"Solution not found"*
18 **end**
19 **end**
20 **return** $\rho_i \models \phi_i = \phi_i' \wedge \bigwedge_{j \neq i} \phi_{j,i}$, $\forall i \in [1, N]$

receive independent requests from others. When request $\phi_{i,j}$ is delivered to robot R_j, R_j recomputes the timed run it is executing. If there exists a timed run for R_j satisfying the local independent task ϕ_j and the received request $\phi_{i,j}$, then the request is *accepted* by R_j. To that end, it updates its specification formula, taking into account that parts of it may have been completed already and that new parts of specifications have arrived. Conversely, if the request is *not accepted* by R_j, R_j will continue to execute the original timed run. Formally, ϕ_j becomes $\phi_j' \wedge \bigwedge_{j \neq i} \phi_{i,j}$, where ϕ_j' is ϕ_j in which each appearance of interval $[a, b], a, b \in \mathbb{Q}_{\geq 0}$ is replaced with $[max(0, a - \tau), max(0, b - \tau)]$, where τ denotes the current time. A new timed run is computed from the new specification in WTS T_j with an initial state being set to the current state.

5) Overall Algorithm: Alg.2 summarizes the overall procedure of the approach. In the *mission decomposition* step (line 1-2), all robots $R_i, i \in [1, N]$ extract requests from the assigned mission specification φ_i, resulting in an independent task ϕ_i and independent requests $\bigwedge_{j \neq i} \phi_{i,j}$ (line 2). During the *execution* step, R_0 implements two rounds of pursuit-evasion strategy in the workspace to exchange all requests while R_i, $i \in [1, N]$ executes their timed runs in parallel independently (line 9-10). Robots $R_i, i \in [1, N]$ then update their tasks and proceed with execution (line 11-19) If there exists N timed runs $\rho_i \models \phi_i' \wedge \bigwedge_{j \neq i} \phi_{j,i}$, $\forall i \in [1, N]$, the joint run satisfying all mission specifications φ_i, $\forall i \in [1, N]$ is found (line 20).

6) Soundness: Directly from Lemma IV-.2 we have:

Theorem IV-.1. *If Alg.2 returns timed runs $\rho_1, \ldots, \rho_N$,*

the joint timed run ρ_{joint} satisfies all mission specifications $\varphi_1, \ldots, \varphi_N$.

7) Completeness: There are three main factors that contribute to the incompleteness of the proposed approach. Firstly, as stated in Remark IV-.1, to simplify the computation of request extraction, a meet promise that is sufficient but more constrained than necessary is chosen in Alg. 1. Secondly, the agile robot is guaranteed to find a heavy-duty robot only under certain assumptions on the speed (i.e. transition weight of WTS) of the agile and the heavy-duty robot, defined through Lemmas IV-.3 and IV-.4. Thirdly, the request exchange by the agile robot is not instant as the robot needs to travel for request exchange. This means that some of the requests may be delivered too late to be completed by the receiving robot unless t_{total} (see Lemma IV-.5) satisfies $t_{total} < a$, for each interval $[a, b]$ that appears in the requests.

8) Complexity: We use $|\varphi_i|$ and $|\varphi_i^{meet}|$ to denote the number of cells that need to be reached given in specification formula φ_i (as Eq.2) and φ_i^{meet} (as Eq.3), respectively. The size of the mission specification increases from $\mathcal{O}(\sum_{i=1}^{N}|\varphi_i|)$ to $\mathcal{O}(\sum_{i=1}^{N}|\varphi_i| + \sum_{i=1}^{N}\sum_{k=1}^{\ell}|\varphi_{i,k}^{meet}|)$, where ℓ is the number of the meet request (candidate) in each φ_i^{meet}. This increase is due to the promise and request extraction. In the proposed approach, each heavy-duty robot R_i, $i \in [1, N]$ needs to solve at most $a'_k - a_k + 1$ path syntheses during the refinement of each existing meet task $\varphi_{i,k}^{meet}$ (as Alg.1) and two path syntheses of the independent task (Eq. (5)) and the updated specification formula (Sec. IV-.4). Therefore, we need to perform a maximum of $\mathcal{O}(2N + \sum_{i=1}^{N}\sum_{k=1}^{\ell}(a'_k - a_k + 1))$ path syntheses. Each synthesis is based on the size of single WTS $|S_i|$ and single specification formula $|\varphi_i|$. At most two path syntheses are required for centralized approaches, which are based on the size of the product of all WTS $\prod_{i=1}^{N}|S_i|$ and the conjunction of all specifications $|\bigwedge_{i\in[1,N]}\varphi_i|$

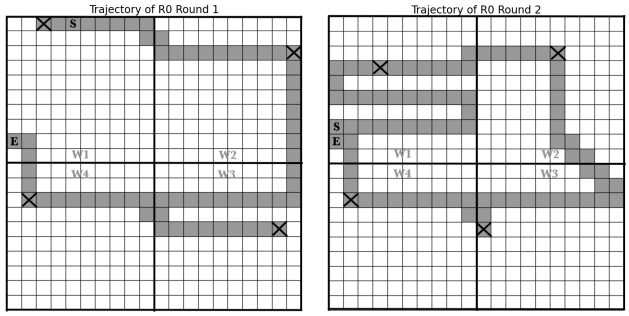
V. Simulation Results

The simulations of our approach are performed in Python3 using UPPAAL [1] as a tool to synthesis the timed runs for MITL specifications. Computation has been performed on an Intel i7-vPro processor and 16GB RAM. The implementation, more case studies, as well as installation instructions are available on Github at `https://github.com/WeiWANG0608/MAS_MITL_Communication`

1) Case study: We consider a scenario inspired by search-and-rescue in an environment partitioned into 20×20 cells. 4 heavy-duty robots are distributed in 4 adjacent work regions of size 10×10. The sensing radius of the agile robot and heavy-duty robots are $r_0 = 2$, and $r_i = 1$, $i \in [1, N]$. In each state, agents can choose actions from $\{Idle, North, South, West, East\}$. WTS $\mathcal{T}_i$ does not have transitions to cells in $\mathcal{W}_{obs}$, which enables the obstacle avoidance. The hatch distance of R_0 is $h_a = 2$. The upper bound transition weight for $\mathcal{T}_0$ is $t_0^{max} \leq \frac{1}{11}t_i$ (see Eq.(8)), where t_i is the transition weight in $\mathcal{T}_i$ of R_i, $i \in [1, N]$.

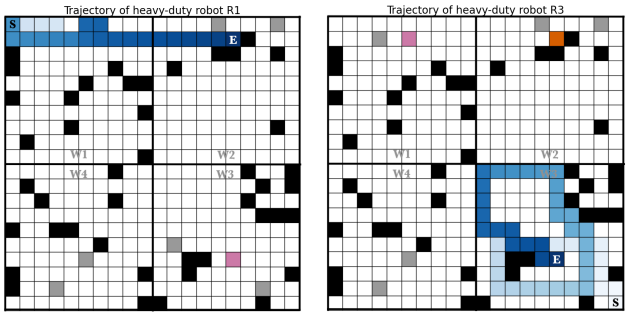
The mission specification assigned to R_i, $i \in [1, 4]$, the independent requests and the independent tasks of R_i obtained through Alg.1 are given in Table I. The orange and the pink text represents the tasks given in φ_i^{collab} and φ_i^{meet}, respectively, which matches the cells of interest in Fig. 5

R_0 collects $\phi_{1,4}$ when $t = 2$ and delivers it to R_4 at $t = 15$. $\phi_{2,1}$ is collected at $t = 5$ and is delivered to R_1 at $t = 10$. Similarly, $\phi_{3,2}$ is collected at $t = 7$ by R_0 and received by R_2 at $t = 11$. All requests are accepted by the corresponding recipient heavy-duty robots. The updated mission specification that generates the final timed run of R_i, $i \in [1, 4]$ is $\phi_i = \phi'_i \wedge \phi_{j,i}$, $i, j \in [1, 4]$, where ϕ'_i is the independent task after time shift (Sec. IV-.4).The two-round trajectory of R_0 under the pursuit-evasion strategy are given in Fig. 4(a) and Fig. 4(b). The trajectory of heavy-duty robot R_1, R_3 satisfying the updated specification is shown as Fig. 5(a)-5(b), respectively. A video of the simulation can be found at `https://youtu.be/36uCwKoGXys`.



(a) Request collection round **(b)** Request delivery round

Fig. 4: Trajectories of R_0. **S / E**: the starting cell / ending cell of the trajectory of each round. The crosses are the positions that R_0 pursues each heavy-duty robot.



(a) The trajectory of robots R_1. **(b)** The trajectory of robots R_3.

Fig. 5: Trajectories of R_i under specification ϕ_i, $i \in \{1, 3\}$. **S / E**: the starting cell / ending cell of the trajectory. The color of R_i's trajectory (blue) deepens over time and the movement of robots.

2) Quantitative Evaluation: We compared our approach to the centralized approach and show the results in Fig.6. For the centralized approach, the average time to synthesize a path and obtain a solution increases rapidly due to exponentially growing number of states of the product WTS and the increasing complexity of MITL formulas. Conversely, as the size of the work region grows and the number of heavy-duty robots increase, the average time to synthesize a path of agents in parallel and obtain a solution for the proposed method increases, but not significantly.

TABLE I: MITL Specification Formulas

Robot	Mission Specification	Independent Request	Independent Task
R_1	$\varphi_1 = \square_{[7,10]}(R_1 @ w_{13}^{\mathcal{W}_1}) \wedge \lozenge_{[50,55]}(R_1 @ w_{15}^{\mathcal{W}_1} \wedge R_4 @ w_{15}^{\mathcal{W}_1})$	$\phi_{2,1} = \square_{[67,70]}(R_1 @ w_{15}^{\mathcal{W}_2})$	$\phi_1 = \square_{[7,10]}(R_1 @ w_{13}^{\mathcal{W}_1}) \wedge \square_{[50,55]}(R_1 @ w_{15}^{\mathcal{W}_1})$
R_2	$\varphi_2 = \square_{[12,15]}(R_2 @ w_4^{\mathcal{W}_2}) \wedge \square_{[30,35]}(R_2 @ w_8^{\mathcal{W}_2} \wedge \square_{[67,70]}(R_1 @ w_{15}^{\mathcal{W}_2})$		$\phi_2 = \square_{[12,15]}(R_2 @ w_4^{\mathcal{W}_2}) \wedge \square_{[30,35]}(R_2 @ w_8^{\mathcal{W}_2})$
R_3	$\varphi_3 = \lozenge_{[8,12]}(R_3 @ w_{43}^{\mathcal{W}_3}) \wedge \square_{[25,28]}(R_3 @ w_{75}^{\mathcal{W}_3}) \wedge \square_{[60,65]}(R_3 @ w_{55}^{\mathcal{W}_3} \wedge R_2 @ w_{55}^{\mathcal{W}_3})$	$\phi_{3,2} = \square_{[60,65]}(R_2 @ w_{55}^{\mathcal{W}_3})$	$\phi_3 = \lozenge_{[8,12]}(R_3 @ w_{43}^{\mathcal{W}_3}) \wedge \square_{[25,28]}(R_3 @ w_{75}^{\mathcal{W}_3}) \wedge \square_{[60,65]}(R_3 @ w_{55}^{\mathcal{W}_3})$
R_4	$\varphi_4 = \lozenge_{[14,30]}(R_4 @ w_{57}^{\mathcal{W}_4}) \wedge \lozenge_{[25,30]}(R_4 @ w_{72}^{\mathcal{W}_4})$	$\phi_{1,4} = \lozenge_{[50,55]}(R_4 @ w_{15}^{\mathcal{W}_1})$	$\phi_4 = \lozenge_{[14,30]}(R_4 @ w_{57}^{\mathcal{W}_4}) \wedge \lozenge_{[25,30]}(R_4 @ w_{72}^{\mathcal{W}_4})$

We observed that on every instance, where the centralized (i.e. complete) algorithm was able to find a solution, our algorithm was able to find a solution, too, indicating that despite being conservative, the approach will often be able to find a solution if one exists.

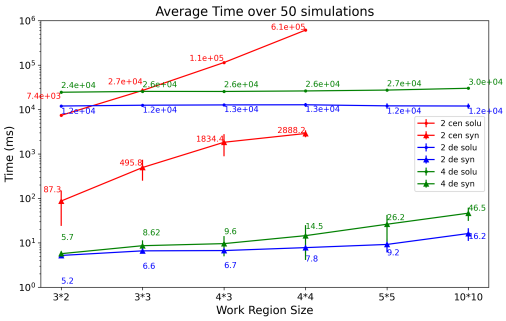


Fig. 6: Two different approaches are used for systems with 2 or 4 heavy-duty robots: a centralized approach (*cen*) and a decentralized (proposed) approach (*de*). The curves represent the average time to obtain a solution (*solu*) and path synthesis using UPPAAL (*syn*).

VI. Conclusion and Future Work

We have proposed a decentralized approach for heterogeneous multi-agent coordination motivated by search-and-rescue scenarios under MITL task specifications and limited sensing capability. The solution involves a series of conditional algorithmic instance solving such that not only individual task specifications but also request specifications are satisfied under formulated assumptions. Future research directions include the consideration of the impacts of network delay and communication instability, more efficient request exchange strategies, and completeness, as well as coordination of multiple agile vehicles.

References

[1] G. Behrmann, A. David, and K. G. Larsen, "A tutorial on UPPAAL," in *Formal methods for the design of real-time systems*, pp. 200–236, Springer, 2004.

[2] S. Bhattacharya, M. Likhachev, and V. Kumar, "Multi-agent path planning with multiple tasks and distance constraints," in *International Conference on Robotics and Automation*, pp. 953–959, IEEE, 2010.

[3] A. I. Bhatti *et al.*, "Multi-agent planning for thermalling gliders using multi level graph-search," *arXiv preprint arXiv:2007.01334*, 2020.

[4] G. Sartoretti, J. Kerr, Y. Shi, G. Wagner, T. S. Kumar, S. Koenig, and H. Choset, "Primal: Pathfinding via reinforcement and imitation multi-agent learning," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2378–2385, 2019.

[5] A. Ma, M. Ouimet, and J. Cortés, "Hierarchical reinforcement learning via dynamic subspace search for multi-agent planning," *Autonomous Robots*, vol. 44, no. 3, pp. 485–503, 2020.

[6] A. Ulusoy, S. L. Smith, X. C. Ding, C. Belta, and D. Rus, "Optimality and robustness in multi-robot path planning with temporal logic constraints," *The International Journal of Robotics Research*, vol. 32, no. 8, pp. 889–911, 2013.

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

[8] L. Hammond, A. Abate, J. Gutierrez, and M. Wooldridge, "Multi-agent reinforcement learning with temporal logic specifications," *arXiv preprint arXiv:2102.00582*, 2021.

[9] P. Yu and D. V. Dimarogonas, "Distributed motion coordination for multi-robot systems under ltl specifications," *arXiv preprint arXiv:2103.09111*, 2021.

[10] P. Yu and D. V. Dimarogonas, "A fully distributed motion coordination strategy for multi-robot systems with local information," in *2020 American Control Conference (ACC)*, pp. 1423–1428, IEEE, 2020.

[11] G. F. Schuppe and J. Tumova, "Decentralized multi-agent strategy synthesis under ltl f specifications via exchange of least-limiting advisers," in *2021 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, pp. 119–127, IEEE, 2021.

[12] J. Fang, Z. Zhang, and R. V. Cowlagi, "Decentralized route-planning to satisfy global linear temporal logic specifications on multiple aircraft," in *AIAA guidance, navigation, and control conference*, p. 1862, 2018.

[13] R. Alur and D. L. Dill, "A theory of timed automata," *Theoretical computer science*, vol. 126, no. 2, pp. 183–235, 1994.

[14] A. Nikou, D. Boskos, J. Tumova, and D. V. Dimarogonas, "On the timed temporal logic planning of coupled multi-agent systems," *Automatica*, vol. 97, pp. 339–345, 2018.

[15] R. Bhat, Y. Yazıcıoğlu, and D. Aksaray, "Distributed path planning for executing cooperative tasks with time windows," *IFAC-PapersOnLine*, vol. 52, no. 20, pp. 187–192, 2019.

[16] U. A. Fiaz and J. S. Baras, "Fast, composable rescue mission planning for uavs using metric temporal logic," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 15404–15411, 2020.

[17] Y. Kantaros and M. M. Zavlanos, "Distributed intermittent connectivity control of mobile robot networks," *IEEE Transactions on Automatic Control*, vol. 62, no. 7, pp. 3109–3121, 2016.

[18] Y. Kantaros, M. Guo, and M. M. Zavlanos, "Temporal logic task planning and intermittent connectivity control of mobile robot networks," *IEEE Transactions on Automatic Control*, vol. 64, no. 10, pp. 4105–4120, 2019.

[19] A. Nikou, S. Heshmati-Alamdari, and D. V. Dimarogonas, "Scalable time-constrained planning of multi-robot systems," *Autonomous Robots*, vol. 44, no. 8, pp. 1451–1467, 2020.

[20] A. Bhatia, L. E. Kavraki, and M. Y. Vardi, "Sampling-based motion planning with temporal goals," in *2010 IEEE International Conference on Robotics and Automation*, pp. 2689–2696, IEEE, 2010.

[21] S. D. Bopardikar, F. Bullo, and J. P. Hespanha, "On discrete-time pursuit-evasion games with sensing limitations," *IEEE Transactions on Robotics*, vol. 24, no. 6, pp. 1429–1439, 2008.

[22] T. M. Cabreira, L. B. Brisolara, and P. R. Ferreira Jr, "Survey on coverage path planning with unmanned aerial vehicles," *Drones*, vol. 3, no. 1, p. 4, 2019.

[23] H. Shipley, D. McDonnell, M. Culleton, R. Coull, R. Lupoi, G. O'Donnell, and D. Trimble, "Optimisation of process parameters to address fundamental challenges during selective laser melting of ti-6al-4v: A review," *International Journal of Machine Tools and Manufacture*, vol. 128, pp. 1–20, 2018.