Task and Motion Coordination for Heterogeneous Multiagent Systems With Loosely Coupled Local Tasks

Meng Guo and Dimos V. Dimarogonas

***Abstract*— We consider a multiagent system that consists of heterogeneous groups of homogeneous agents. Instead of defining a global task for the whole team, each agent is assigned a local task as syntactically cosafe linear temporal logic formulas that specify both motion and action requirements. Interagent dependence is introduced by collaborative actions, of which the execution requires multiple agents' collaboration. To ensure the satisfaction of all local tasks without central coordination, we propose a bottom–up motion and task coordination strategy that contains an off-line initial plan synthesis and an online coordination scheme based on real-time exchange of request and reply messages. It facilitates not only the collaboration among heterogeneous agents but also the task swapping between homogeneous agents to reduce the total execution cost. It is distributed as any decision is made locally by each agent based on local computation and communication within neighboring agents. It is scalable and resilient to agent failures as the dependence is formed and removed dynamically based on agent capabilities and their plan execution status, instead of preassigned agent identities. The overall scheme is demonstrated by a simulated scenario of 20 agents with loosely coupled local tasks.**

***Note to Practitioners*—This paper was motivated by the problem of coordinating a large number of autonomous robots in collaborative manufacturing and delivery processes, where each robot has an individual task but collaborations among the robots are essential for the accomplishment of all tasks. Existing approaches to address such problems rely on the direct composition of all robots' models and tasks, yielding a high complexity of computation and difficulty of coordination, where all robot behaviors are synchronized at all time and coordinated centrally. In this paper, through a local request and reply communication protocol, we remove these constraints and impose only local coordination rules among the robots. Moreover, compared with the common solutions, where interrobot collaborations are bound to preassigned robot identities, we allow the robots to negotiate and choose their collaborators in a flexible way. It is guaranteed that the individual task of each robot is satisfied under the collaboration with other robots, but without the need of a central coordinator. We have shown that it is particularly useful for loosely coupled multirobot applications, where the interrobot collaborations are feasible and local. Simulation results suggest that this approach is applicable to large-scale multirobot systems and can be easily integrated with suitable low-level motion control strategies, but it has not yet been tested in experimental environments. In the future research, we will address more complex robot tasks and practical implementations.**

***Index Terms*— Formal methods, linear temporal logics (LTLs), motion and task coordination, multiagent system.**

Manuscript received December 18, 2015; revised July 5, 2016; accepted November 8, 2016. Date of publication December 9, 2016; date of current version April 5, 2017. This paper was recommended for publication by Editor J. Wen upon evaluation of the reviewers' comments. This work was supported in part by the Swedish Research Council (VR), in part by the H2020 ERC Starting Grant BUCOPHSYS, and in part by the EU STREP RECONFIGFP7-ICT-2011-9-600825. This paper was presented at the IEEE International Conference on Automation Science and Engineering (CASE), Gothenburg, Sweden, Aug. 2015.

The authors are with the KTH Centre for Autonomous Systems and ACCESS Linnaeus Center, EES, KTH Royal Institute of Technology, SE-100 44 Stockholm, Sweden (e-mail: mengg@kth.se; dimos@kth.se).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TASE.2016.2628389

I. Introduction

TEMPORAL logics, such as linear temporal logic (LTL) and computation tree logic, have gained significant attention in recent years, due to their usage as formal high-level languages to describe more complex planning objectives for autonomous robots, compared with the well-studied point-to-point navigation problem [18]. Particularly, the high-level task specification is given as a temporal logic formula with respect to a discretized abstraction of the robot motion within the workspace [1], [4]. A high-level discrete plan is found by off-the-shelf model-checking algorithms given the abstraction and task specification [2]. This discrete plan is then implemented through the corresponding low-level continuous controller [7], [8]. Thus, this provides an automated motion and task planning framework for autonomous robots under formal high-level tasks. Similar methodology has also been applied to multiagent systems [6], [15], [26]. Most of the existing work focuses on decomposing a global specification to bisimilar local ones in a *top–down* approach, which can be then assigned and implemented by individual agents in a synchronized [6] or partially synchronized [16] manner. This way of problem formulation naturally favors a tightly coupled structure, meaning that the role of each agent is fixed and their behaviors should be globally coordinated. Normally, a central monitoring unit is essential for both the plan synthesis and plan execution under this formulation. Supervisory control under dynamic control specifications is discussed in [23].

In contrast, we assume that there is no prespecified global task, and individual tasks are assigned locally to each agent as LTL formulas, which favor a *bottom–up* formulation [11], [12], [24]. It is particularly useful for multiagent systems, where the number of agents is large and the agents have clear task assignments given their heterogeneous capabilities. The case, where these local tasks are fully independent,

is considered in [12] for a partially known workspace. Additional relative-motion constraints among neighboring agents, such as relative-distance and connectivity maintenance constraints, are addressed in [3] and [20].

On the other hand, these local tasks can be dependent, when one agent needs other agents' collaboration to perform certain actions, rendering interagent coordination thus crucial for the accomplishment of all local tasks. The greatest challenge of task coordination for multiagent systems under dependent *local* tasks is the computational complexity. A centralized solution requires the direct composition of all agents' models to represent all possible behaviors, which is subject to combinatorial blowup. This issue is addressed in [11] by grouping the agents into dependence clusters such that the composition is only needed for each cluster, while [24] proposes a receding horizon approach that decomposes the synthesis problem into shorter horizon planning problems that are solved iteratively. In general, the local plans derived through composition need to be executed in a synchronized fashion by the agents to ensure the temporal constraints from all local tasks, which greatly limits the flexibility and robustness of the overall system. Nevertheless, the completeness of any solution can only be verified surely through direct composition of all agent models under the intersection of all local tasks. However, for loosely coupled systems, where the required collaborations among the agents are *local* and *sparse*, given the large total number of agents and their assigned tasks, we aim here at avoiding the composition of different agents' models or tasks, which is replaced by an online request, and reply scheme and a real-time plan adaptation algorithm. In addition, we aim for a distributed coordination scheme, where motion and actions are coordinated only when needed, and collaborative relations among the agents are formed and removed dynamically. We show that the proposed scheme guarantees the satisfaction of all local tasks and potential agent failures can be recovered. Some potential applications can be found in [5].

This paper builds on preliminary results from [13]. In contrast to [13], we introduce here a novel formulation of heterogeneous groups of homogeneous agents, which leads to more efficient coordination schemes. A two-layer communication network that allows for both static and dynamic communications is proposed in contrast to the purely static network considered in [13]. Moreover, a global coordination scheme is designed to be activated in case local coordination provides no solution. We also enrich the analysis by proposing a task swapping scheme between homogeneous agents to reduce the total execution cost. At last, a new case study with a larger number of agents and more complex task assignments is considered, depicting the scalability of our approach.

The rest of this paper is organized as follows. Section II introduces some preliminaries. The problem is stated formally in Section III. Section IV presents the initial plan synthesis strategy. The online coordination scheme is described in Section V, followed by the task swapping algorithm introduced in Section VI. The overall structure is discussed in Section VII. A case study is presented in Section VIII and we conclude in Section IX.

II. Preliminaries

A. Syntactically Cosafe LTL and Büchi Automaton

Atomic propositions are Boolean variables that can be either true or false. The ingredients of an LTL formula are a set of atomic propositions (AP) and several Boolean and temporal operators, which are specified according to the following syntax [2]: $\varphi ::= \top \mid p \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid \bigcirc\varphi \mid \varphi_1 \mathsf{U} \varphi_2$, where $\top \triangleq \texttt{True}$, $p \in AP$ and $\bigcirc$ (*next*), and U (*until*). $\bot \triangleq \neg\top$. For brevity, we omit the derivations of other useful operators like $\Box$ (*always*), $\Diamond$ (*eventually*), $\Rightarrow$ (*implication*), and the semantics of LTL. We refer the readers to [2, Ch. 5]. One particular class of LTL we consider in this paper is the syntactically cosafe LTL (sc-LTL) [17]. It only contains the $\bigcirc$, U, and $\Diamond$ operators and is written in positive normal form. The satisfaction of an sc-LTL formula can be achieved in finite time, i.e., each word satisfying an sc-LTL formula φ consists of a *satisfying prefix* that can be followed by an arbitrary suffix. A language of words that satisfy an LTL formula φ over AP can be captured through a nondeterministic Büchi automaton (NBA) $\mathcal{A}_\varphi$ [2], defined as $\mathcal{A}_\varphi = (Q, 2^{AP}, \delta, Q_0, \mathcal{F})$, where Q is a set of states, 2^{AP} is the set of all alphabets, $\delta \subseteq Q \times 2^{AP} \times Q$ is a transition relation, and Q_0 and $\mathcal{F} \subseteq Q$ are the initial and accepting states. There are fast translation tools [9].

B. Notations

Given a set of states S, a finite sequence of elements in S is denoted by $\mathbf{v} = s_1 s_2 \ldots s_N$, where $s_i \in S$, $i = 1, \ldots, N$. Denote by $\mathbf{v}[j]$, the jth state along $\mathbf{v}$, i.e., $\mathbf{v}[j] = s_j$; $v[i:j]$ the segment from the ith to the jth element for $j \geq i$ (including both); $\mathbf{v}[i:]$ the segment from the ith to the last element.

III. Problem Formulation

We consider $N \geq 1$ autonomous agents with heterogeneous capabilities within a fully known workspace. Each agent has the capabilities of navigating within the workspace and performing various actions. Denote by $\mathcal{N} = \{a_i,\ i = 1, 2, \ldots, N\}$ the agent index. Then, we assume K groups within the team, denoted by $\mathcal{K} = \{g_k,\ k = 1, 2, \ldots, K\}$. Agents within the same group are homogeneous with the same motion and action capabilities, while agents belonging to different groups are heterogeneous with different motion or action capabilities. Note that $g_k \subseteq \mathcal{N}$, $\cup_{g_k \in \mathcal{K}} g_k = \mathcal{N}$, and $g_k \cap g_l = \emptyset$, $\forall g_k,\ g_l \in \mathcal{K}$. Below, we define the model of agent motion and actions in detail.

A. Motion Abstraction

The workspace consists of M partitions as the regions of interest, denoted by $\Pi = \{\pi_1, \pi_2, \ldots, \pi_M\}$. We assume that these symbols are assigned *a priori* and known by all agents. There are different cell decomposition techniques available, depending on the agent dynamics and the associated control approaches [4], [6], [19]. Besides, there is a set of atomic propositions describing the properties of the workspace, denoted by $\Psi^{a_i}_{\mathcal{M}}$. Similar to [10], agent a_i's motion within the workspace is modeled as a finite transition system (FTS)

$$\mathcal{M}^{a_i} \triangleq \left(\Pi,\ \longrightarrow^{a_i}_{\mathcal{M}},\ \Pi^{a_i}_{\mathcal{M},0},\ \Psi^{a_i}_{\mathcal{M}},\ L^{a_i}_{\mathcal{M}},\ T^{a_i}_{\mathcal{M}}\right) \tag{1}$$

where $\longrightarrow_{\mathcal{M}}^{a_i} \subseteq \Pi \times \Pi$ is the transition relation, $\Pi_{\mathcal{M},0}^{a_i} \in \Pi$ is the initial region agent a_i starts from, $L_{\mathcal{M}}^{a_i} : \Pi \to 2^{\Psi_{\mathcal{M}}^{a_i}}$ is the labeling function, indicating the properties held by each region, and $T_{\mathcal{M}}^{a_i} : \longrightarrow_{\mathcal{M}}^{a_i} \to \mathbb{R}^+$ gives the time each transition takes. A path of $\mathcal{M}^{a_i}$ is a sequence of regions $\pi_0\pi_1 \ldots \pi_P$, where $(\pi_p, \pi_{p+1}) \in \longrightarrow_{\mathcal{M}}^{a_i}$, $\forall p = 0, 1, \ldots, P-1$. Note that $\mathcal{M}^{a_i}$ are different among heterogeneous agents, while for homogeneous agents the only difference lies in the initial region.

B. Communication Model

Moreover, each agent a_i has a set of neighboring agents at time $t > 0$, denoted by $\mathcal{N}_t^{a_i} \subseteq \mathcal{N}$. Agent a_i can exchange messages directly with any agent $a_j \in \mathcal{N}_t^{a_i}$. Inspired by the idea of backbone network [5], we define the following *two-layer* communication network.

Definition 1: The communication network $\mathcal{C}(t) = (\mathcal{N}, \mathcal{E}(t))$ at time $t > 0$, where $\mathcal{N}$ is the set of nodes and $\mathcal{E}(t) \subseteq \mathcal{N} \times \mathcal{N}$ is the set of edges, which has two layers: $\mathcal{E}(t) = \mathcal{E}_1(t) \cup \mathcal{E}_2(t)$.

1) The first layer is static. For all $t > 0$, there exists $a_{c_k} \in g_k, \forall g_k \in \mathcal{K}$ that $\mathcal{E}_1(t) = \{(a_{c_{k_1}}, a_{c_{k_2}}), \forall g_{k_1}, g_{k_2} \in \mathcal{K}\} \cup \{(a_{c_k}, a_{j,k}), \forall a_{j,k} \in g_k, \forall g_k \in \mathcal{K}\}$.
2) The second layer is dynamic. At time $t > 0$, for any pair of agents $a_i, a_j \in \mathcal{N}$, $(a_i, a_j) \in \mathcal{E}_2(t)$ if agents a_i and a_j satisfy their underlying communication model. ▲

Regarding the first layer above, we call the agent a_{c_k} within each group $g_k \in \mathcal{K}$ the "*coordinator*" of that group. The second layer depends on the underlying communication model of the actual system, e.g., one commonly seen example is the proximity model that two agents can communicate if their relative distance is less than the communication radius. Based on $\mathcal{C}(t)$, each agent's neighboring set is given by $\mathcal{N}_t^{a_i} = \{a_j \in \mathcal{N} \mid (a_i, a_j) \in \mathcal{E}(t)\}$. The network $\mathcal{C}(t)$ is always connected due to the existence of the first layer. In other words, any message can be communicated either directly or indirectly between any two agents within the system.

C. Action Model

Besides the motion ability, agent a_i is capable of performing a set of actions denoted by $\Sigma^{a_i} \triangleq \Sigma_l^{a_i} \cup \Sigma_c^{a_i} \cup \Sigma_h^{a_i}$, where $\Sigma_l^{a_i}$ is a set of *local* actions, which can be done by agent a_i itself; $\Sigma_c^{a_i}$ is a set of *collaborative* actions, which can be done by agent a_i but requires collaborations from other agents; and $\Sigma_h^{a_i}$ is a set of *assisting* actions, in which agent a_i offers to other agents to accomplish their collaborative actions.

In other words, $\Sigma_l^{a_i}$ and $\Sigma_c^{a_i}$ contain actions that can be *initiated* by agent a_i, denoted by $\Sigma_a^{a_i} = \Sigma_l^{a_i} \cup \Sigma_c^{a_i}$, while $\Sigma_h^{a_i}$ contains assisting actions only to *assist* other agents. By default, $\sigma_0 = \texttt{None} \in \Sigma_l^{a_i}$ means that none of the actions is performed. Moreover, denote by $\Sigma_h^{\sim a_i}$ the set of *external* assisting actions agent a_i depends on, which can be provided by other agents in $\mathcal{N}$, i.e., $\Sigma_h^{\sim a_i} \subseteq \cup_{a_j \in \mathcal{N}} \Sigma_h^{a_j}$. The action model of agent a_i is modeled by a six-tuple

$$\mathcal{A}^{a_i} \triangleq \left(\Sigma^{a_i}, \Psi_\Sigma^{a_i}, L_\Sigma^{a_i}, \texttt{Cond}^{a_i}, \texttt{Dura}^{a_i}, \texttt{Depd}^{a_i}\right) \quad (2)$$

where Σ^{a_i} is the set of actions defined earlier, $\Psi_\Sigma^{a_i}$ is a set of atomic propositions related to the agent's *active* actions, and $L_\Sigma^{a_i} : \Sigma^{a_i} \to 2^{\Psi_\Sigma^{a_i}}$ is the labeling function. $L_\Sigma^{a_i}(\sigma_h) = \emptyset$, $\forall \sigma_h \in \Sigma_h^{a_i}$ and $L_\Sigma^{a_i}(\sigma_a) \subseteq \Psi_\Sigma^{a_i}$, $\forall \sigma_a \in \Sigma_a^{a_i}$; $\texttt{Cond}^{a_i} : \Sigma^{a_i} \times 2^{\Psi_{\mathcal{M}}^{a_i}} \to \top/\bot$ indicates the set of region properties that have to be fulfilled in order to perform an action; $\texttt{Dura}^{a_i} : \Sigma^{a_i} \to \mathbb{R}^+$ is the estimated time duration of each action. $\texttt{Dura}^{a_i}(\sigma_0) = T_0 > 0$ is a design parameter and $\texttt{Depd}^{a_i} : \Sigma^{a_i} \to 2^{\Sigma_h^{\sim a_i}}$ is the *dependence* function. $\texttt{Depd}^{a_i}(\sigma_s) = \emptyset$, $\forall \sigma_s \in \Sigma_l^{a_i} \cup \Sigma_h^{a_i}$ and $\texttt{Depd}^{a_i}(\sigma_c) \subseteq \Sigma_h^{\sim a_i}$, $\forall \sigma_c \in \Sigma_c^{a_i}$. Namely, each collaborative action depends on a set of assisting actions from other agents. Moreover, different collaborative actions can depend on the same assisting actions. This is useful for defining complex collaborations involving multiple agents.

Definition 2: A local or assisting action $\sigma_s \in \Sigma_l^{a_i} \cup \Sigma_h^{a_i}$ is said to be done at region $\pi_h \in \Pi$ if two conditions hold the following.

1) $\texttt{Cond}^{a_i}(\sigma_s, L_{\mathcal{M}}^{a_i}(\pi_h)) = \top$.
2) σ_s is activated for period $\texttt{Dura}^{a_i}(\sigma_s)$. For a collaborative action $\sigma_c \in \Sigma_c^{a_i}$, another condition is needed.
3) All assisting actions in $\texttt{Depd}^{a_i}(\sigma_c)$ are done by other agents at the same region π_h. ▲

By the definition above, the execution of assisting actions can overlap and a collaborative action is accomplished only after the last assisting action is done at the same region. Compared with defining dependence directly on agent identities, our action model allows more system flexibility, since the agent identities need not be known *a priori*, and new or existing agents can be added or removed during run time.

Remark 1: Different from [10], the action model by (2) can model both local and collaborative actions. ▲

D. Complete Agent Model

A complete agent model, denoted by $\mathcal{G}^{a_i}$, refers to the FTS that models both its motion and actions.

Definition 3: Given $\mathcal{M}^{a_i}$ and $\mathcal{A}^{a_i}$, agent a_i's complete model can be constructed as follows: $\mathcal{G}^{a_i} = (\Pi_{\mathcal{G}}^{a_i}, \longrightarrow_{\mathcal{G}}^{a_i}, \Pi_{\mathcal{G},0}^{a_i}, \Psi_{\mathcal{G}}^{a_i}, L_{\mathcal{G}}^{a_i}, T_{\mathcal{G}}^{a_i})$, where $\Pi_{\mathcal{G}}^{a_i} = \Pi \times \Sigma^{a_i}$. $\pi_{\mathcal{G},i} = \langle \pi_j, \sigma_n \rangle \in \Pi_{\mathcal{G}}^{a_i}$, $\forall \pi_j \in \Pi$, $\forall \sigma_n \in \Sigma^{a_i}$; $\longrightarrow_{\mathcal{G}}^{a_i} \subseteq \Pi_{\mathcal{G}}^{a_i} \times \Pi_{\mathcal{G}}^{a_i}$. $(\langle \pi_h, \sigma_m \rangle, \langle \pi_j, \sigma_n \rangle) \in \longrightarrow_{\mathcal{G}}^{a_i}$ if: 1) $\sigma_n = \sigma_m = \sigma_0$, $\pi_h \longrightarrow_{\mathcal{M}}^{a_i} \pi_j$; 2) $\sigma_m = \sigma_0$, $\sigma_n \neq \sigma_0$ and $\pi_h = \pi_j$, $\texttt{Cond}^{a_i}(\sigma_n, L_{\mathcal{M}}^{a_i}(\pi_h)) = \top$; *or* 3) $\sigma_m \in \Sigma^{a_i}$, $\sigma_n = \sigma_0$, and $\pi_h = \pi_j$; $\Pi_{\mathcal{G},0}^{a_i} = \langle \Pi_{\mathcal{M},0}^{a_i}, \sigma_0 \rangle$ is the initial state; $\Psi_{\mathcal{G}}^{a_i} = \Psi_{\mathcal{M}}^{a_i} \cup \Psi_\Sigma^{a_i}$; $L_{\mathcal{G}}^{a_i} : \Pi_{\mathcal{G}}^{a_i} \to 2^{\Psi^k}$. $L_{\mathcal{G}}^{a_i}(\langle \pi_h, \sigma_m \rangle) = L_{\mathcal{M}}^{a_i}(\pi_h) \cup L_\Sigma^{a_i}(\sigma_m)$; $T_{\mathcal{G}}^{a_i} : \longrightarrow_{\mathcal{G}}^{a_i} \to \mathbb{R}^+$. For case 1) above, $T_{\mathcal{G}}^{a_i}(\langle \pi_h, \sigma_m \rangle, \langle \pi_j, \sigma_n \rangle) = T_{\mathcal{M}}^{a_i}(\pi_h, \pi_j)$; for case 2), $T_{\mathcal{G}}^{a_i}(\langle \pi_h, \sigma_m \rangle, \langle \pi_j, \sigma_n \rangle) = \texttt{Dura}^{a_i}(\sigma_n)$; and for case 3), $T_{\mathcal{G}}^{a_i}(\langle \pi_h, \sigma_m \rangle, \langle \pi_j, \sigma_n \rangle) = T_0$. ▲

Note that when defining $\longrightarrow_{\mathcal{G}}^{a_i}$ above, the condition of performing an action is verified over the properties of each region. Thus, $\mathcal{G}^{a_i}$ is a standard FTS [2]. Its finite path is denoted by $\tau^{a_i} = \pi_{\mathcal{G},0}\pi_{\mathcal{G},1} \ldots \pi_{\mathcal{G},P}$, where $\pi_{\mathcal{G},i} \in \Pi_{\mathcal{G}}^{a_i}$, $\pi_{\mathcal{G},0} = \Pi_{\mathcal{G},0}^{a_i}$, and $(\pi_{\mathcal{G},s}, \pi_{\mathcal{G},s+1}) \in \longrightarrow_{\mathcal{G}}^{a_i}$, $\forall s = 0, \ldots, P-1$. Its trace is $\texttt{trace}(\tau^{a_i}) = L_{\mathcal{G}}^{a_i}(\pi_{\mathcal{G},0}) L_{\mathcal{G}}^{a_i}(\pi_{\mathcal{G},1}) \ldots L_{\mathcal{G}}^{a_i}(\pi_{\mathcal{G},P})$.

E. Task Specification

The local task of agent a_i, denoted by φ^{a_i}, is given as an sc-safe LTL formula over the set of atomic propositions $\Psi_{\mathcal{G}}^{a_i}$ from Definition 3. Thus, φ^{a_i} can contain requirements on agent's motion, local and collaborative actions. As mentioned earlier, an sc-safe LTL formula can be fulfilled by a finite prefix. In particular, given a finite path τ^{a_i} of $\mathcal{G}^{a_i}$, then τ^{a_i} fulfills φ^{a_i} if $\texttt{trace}(\tau^{a_i}) \models \varphi^{a_i}$ where the satisfaction relation is defined in Section II-A. One special case is that when $\varphi^{a_i} \triangleq \top$, agent a_i does not have a local task and serves as an assisting agent. In summary, we consider the following problem.

Problem 1: Given $\mathcal{G}^{a_i}$ and the locally assigned task φ^{a_i}, design a distributed control and coordination scheme such that φ^{a_i} is fulfilled for all $a_i \in \mathcal{N}$. ▲

IV. Off-Line Initial Plan Synthesis

In this section, we describe how to synthesize an *initial* motion and action plan for each agent, which happens off-line and serves as a starting point for the real-time coordination and adaptation scheme in Section V.

A. Plan as Motion and Action Sequence

We intend to find a finite path of $\mathcal{G}^{a_i}$, whose trace satisfies the cosafe formula φ^{a_i}, as described in Section III-E. We rely on the automaton-based model-checking approach (see [2, Algorithm 11]) by checking the emptiness of the product automaton. Let $\mathcal{A}_{\varphi^{a_i}}$ be the NBA associated with φ^{a_i}, i.e., $\mathcal{A}_{\varphi} = (Q^{a_i}, 2^{\Psi_{\mathcal{G}}^{a_i}}, \delta^{a_i}, Q_0^{a_i}, \mathcal{F}^{a_i})$, of which the notations are defined as in Section II-A. The product automaton $\mathcal{A}_p^{a_i}$ is defined as follows:

$$\mathcal{A}_p^{a_i} = \mathcal{G}^{a_i} \otimes \mathcal{A}_{\varphi^{a_i}} = \left(Q_p^{a_i},\, \delta_p^{a_i},\, Q_{p,0}^{a_i},\, \mathcal{F}_p^{a_i},\, W_p^{a_i}\right) \tag{3}$$

where $Q_p^{a_i} = \Pi_{\mathcal{G}}^{a_i} \times Q^{a_i}$, $q_p = \langle \pi_{\mathcal{G}}, q_n \rangle \in Q_p^{a_i}$, $\forall \pi_{\mathcal{G}} \in \Pi_{\mathcal{G}}^{a_i}$, $\forall q_n \in Q^{a_i}$; $(\langle \pi_{\mathcal{G},s}, q_m \rangle, \langle \pi_{\mathcal{G},l}, q_n \rangle) \in \delta_p^{a_i}$ if $\pi_{\mathcal{G},s} \longrightarrow_{\mathcal{G}}^{a_i} \pi_{\mathcal{G},l}$ and $(q_m, L_{\mathcal{G}}^{a_i}(\pi_{\mathcal{G},s}), q_n) \in \delta^{a_i}$; $Q_{p,0}^{a_i} = \{\Pi_{\mathcal{G},0}^{a_i}\} \times Q_0^{a_i}$ is the set of initial states; $\mathcal{F}_p^{a_i} = \Pi_{\mathcal{G}}^{a_i} \times \mathcal{F}^{a_i}$ is the set of accepting states; $W_p^{a_i} : \delta_p^{a_i} \times Q_p^{a_i} \to \mathbb{R}^+$. $W_p^{a_i}(\langle \pi_{\mathcal{G},s}, q_m \rangle, \langle \pi_{\mathcal{G},l}, q_n \rangle) = T_{\mathcal{G}}^{a_i}(\pi_{\mathcal{G},s}, \pi_{\mathcal{G},l})$, where $\langle \pi_{\mathcal{G},l}, q_n \rangle \in \delta_p^{a_i}(\langle \pi_{\mathcal{G},s}, q_m \rangle)$.

There exists a finite path of $\mathcal{G}^{a_i}$ satisfying φ^{a_i} if and only if $\mathcal{A}_p^{a_i}$ has a finite path from an initial state to an accepting state. Then, this path could be projected back to $\mathcal{G}^{a_i}$ as a finite path, the trace of which should satisfy φ^{a_i} automatically [2]. Let $R_p^{a_i} = q_{p,0}^{a_i} q_{p,1}^{a_i} \dots q_{p,P}^{a_i}$ be a finite path of $\mathcal{A}_p^{a_i}$, where $q_{p,0}^{a_i} \in Q_{p,0}^{a_i}$, $q_{p,P}^{a_i} \in \mathcal{F}_p^{a_i}$, and $q_{p,s}^{a_i} \in Q_p^{a_i}$ and $(q_{p,s}^{a_i}, q_{p,s+1}^{a_i}) \in \delta_p^{a_i}$, $\forall s = 0, \dots, P-1$. The *cost* of $R_p^{a_i}$ is defined by $\texttt{Cost}(R_p^{a_i}, \mathcal{A}_p^{a_i}) = \sum_{s=0}^{P-1} W_p^{a_i}(q_{p,s}^{a_i}, q_{p,s+1}^{a_i})$, which is the summed weights along $R_p^{a_i}$. The sth element is given by $R_p^{a_i}[s] = q_{p,s}^{a_i}$, and the segment from the sth to the lth element is $R_p^{a_i}[s:l] = q_{p,s}^{a_i} q_{p,s+1}^{a_i} \dots q_{p,l}^{a_i}$, where $s \le j \le P$.

Problem 2: Find a finite path $R_p^{a_i}$ of $\mathcal{A}_p^{a_i}$ with the above structure that minimizes its total cost.

Denote by $R_{p,\text{init}}^{a_i}$ the solution. [10, Algorithm 1] solves the above problem, which is omitted here due to limited space. It utilizes Dijkstra's algorithm [18] for computing the shortest path from any initial state in $Q_{p,0}^{a_i}$ to every reachable accepting state in $\mathcal{F}_p^{a_i}$ and checks if there is cycle back to $q_{p,P}$. The worst case complexity is $\mathcal{O}(|\delta_p^{a_i}| \cdot \log|Q_p^{a_i}| \cdot |Q_{p,0}^{a_i}|)$. By projecting $R_{p,\text{init}}^{a_i}$ onto $\Pi_{\mathcal{G}}^{a_i}$, it gives the initial motion and action plan $\tau_{\mathcal{G},\text{init}}^{a_i} = R_{p,\text{init}}^{a_i}|_{\Pi_{\mathcal{G}}^{a_i}}$ that fulfills φ^{a_i}.

Remark 2: The initial plans are synthesized locally instead of by a central unit [6] or within a cluster [11]. ▲

The plan $\tau_{\mathcal{G},\text{init}}^{a_i}$ can be executed by activating the motion or actions in sequence. However, since $\tau_{\mathcal{G},\text{init}}^{a_i}$ may contain several collaborative actions from $\Sigma_c^{a_i}$ to satisfy φ^{a_i}, the successful execution of $\tau_{\mathcal{G},\text{init}}^{a_i}$ depends on other agents' collaboration, which, however, is not guaranteed, since $\tau_{\mathcal{G},\text{init}}^{a_i}$ is synthesized off-line and locally. We resolve this problem by a real-time coordination and adaptation scheme in Section V.

V. Distributed Collaborative Task Coordination

As mentioned earlier, there is no guarantee that the initial plan $\tau_{\mathcal{G},\text{init}}^{a_i}$ can be executed successfully, if it contains collaborative actions. In this section, we propose a distributed and online coordination scheme, which involves four major parts: 1) a request and reply exchange protocol driven by collaborative actions in a finite horizon; 2) an optimization and confirmation mechanism, by solving a mixed integer program based on the replies; 3) a real-time plan adaptation algorithm given the confirmation; and 4) an agent failure detection and recovery scheme along with the plan execution.

A. Planned Motion and Actions in Horizon

Denote by $\pi_{\mathcal{G},t}^{a_i} \in \Pi_{\mathcal{G}}^{a_i}$ the state of agent a_i at time t. After the system starts, assume $\pi_{\mathcal{G},t}^{a_i}$ is the lth element in $\tau_{\mathcal{G},\text{init}}^{a_i}$, namely, $\pi_{\mathcal{G},t}^{a_i} = \tau_{\mathcal{G},\text{init}}^{a_i}[l]$. Each agent $a_i \in \mathcal{N}$ is given a bounded planning horizon $0 < H^{a_i} < \infty$, which is the time ahead agent a_i checks its plan. Similar approach can be found in [21] for a single dynamic system. Then, the sequence of states agent a_i is expected to reach within the time H^{a_i}, denoted by $\tau_{\mathcal{G},H}^{a_i}$ is the segment $\tau_{\mathcal{G},H}^{a_i} = \tau_{\mathcal{G},\text{init}}^{a_i}[l:f]$, where the index $f \ge l$ is the solution to this optimization problem: min f, subject to $\sum_{s=l}^{f} T_{\mathcal{G}}^{a_i}(\tau_{\mathcal{G},\text{init}}^{a_i}[s], \tau_{\mathcal{G},\text{init}}^{a_i}[s+1]) \ge H^{a_i}$. It can be solved by iterating through the sequence of $\tau_{\mathcal{G},\text{init}}^{a_i}$ and computing the accumulated cost, which is then compared with H^{a_i}. If it does not have a solution, it means H^{a_i} is larger than the total cost of the rest of the plan $\tau_{\mathcal{G},\text{init}}^{a_i}[l:]$, then $f = |\tau_{\mathcal{G},\text{init}}^{a_i}|$ (see Lines 1–5, 9, and 10 of Algorithm 1). The time horizon avoids the agents coordinating on collaborative actions that need to be done within a long time from now.

B. Request to Neighbors

Given $\tau_{\mathcal{G},H}^{a_i}$ as the motion and actions in horizon, agent a_i needs to check whether it needs others' collaboration within $\tau_{\mathcal{G},H}^{a_i}$. This is done by verifying whether a collaborative action needs to be performed to reach the states in $\tau_{\mathcal{G},H}^{a_i}$. More specifically, for the *first* state $\langle \pi_h, \sigma_m \rangle \in \tau_{\mathcal{G},H}^{a_i}$ satisfying $\sigma_m \in \Sigma_c^{a_i}$, agent a_i needs to *broadcast* a request to all agents within its communication network $\mathcal{N}_t^{a_i}$ regarding this action. This request message has the following format:

$$\textbf{Request}^{a_i} = \{(\sigma_d,\, \pi_h,\, T_m), \quad \forall \sigma_d \in \texttt{Dep}^{a_i}(\sigma_m)\} \tag{4}$$

Algorithm 1 Plan in Horizon and Request, `Request()`

Input: $\tau^{a_i}_{\mathcal{G},\text{init}}$, $\pi^{a_i}_{\mathcal{G},t}$, H^{a_i}
Output: $\tau^{a_i}_{\mathcal{G},H}$, $\mathbf{Request}^{a_i}$

1 $\tau^{a_i}_{\mathcal{G},\text{init}}[l] = \pi^{a_i}_{\mathcal{G},t}$, $s = 0$, $T_m = 0$, $\mathbf{Request}^{a_i} = \emptyset$
2 **while** $T < H^{a_i}$ *and* $l + s \leq |\tau^{a_i}_{\mathcal{G},init}|$ **do**
3 $s = s + 1$
4 $T_m = T_m + T^{a_i}_{\mathcal{G}}\left(\tau^{a_i}_{\mathcal{G},\text{init}}[l+s-1], \tau^{a_i}_{\mathcal{G},\text{init}}[l+s]\right)$
5 $\langle \pi_h, \sigma_m \rangle = \tau^{a_i}_{\mathcal{G},\text{init}}[l+s]$
6 **if** $\mathbf{Request}^{a_i} = \emptyset$ *and* $\sigma_m \in \Sigma^{a_i}_c$ **then**
7 **forall the** $\sigma_d \in \texttt{Depd}^{a_i}(\sigma_m)$ **do**
8 add (σ_d, π_h, T_m) to $\mathbf{Request}^{a_i}$
9 $f = l + s$, $\tau^{a_i}_{\mathcal{G},H} = \tau^{a_i}_{\mathcal{G},\text{init}}[l:f]$
10 **return** $\tau^{a_i}_{\mathcal{G},H}$, $\mathbf{Request}^{a_i}$

Algorithm 2 Reply to Request by Agent a_j, `Reply()`

Input: $\mathbf{Request}^k_g$, $\widehat{R}^{a_j}_{p,-}$, $\mathcal{A}^{a_j}_p$, $\overline{T}^{a_j}$
Output: $\mathbf{Reply}^{a_j}_k$, $\widehat{P}$

1 **forall the** $(\sigma_d, \pi_h, T_m) \in \mathbf{Request}^k_g$ **do**
2 **if** $\overline{T}^{a_j}$ *is* 0 **then**
3 $\left(\widehat{R}^{a_j}_{p,+}, b^{a_j}_d, t^{a_j}_d\right) =$ `EvalReq`$(\widehat{R}^{a_j}_{p,-}, (\sigma_d, \pi_h, T_m), \mathcal{A}^{a_j}_p)$
4 **if** $b^{a_j}_d$ *is* $\top$ **then**
5 $\widehat{P}(\sigma_d) = \widehat{R}^{a_j}_{p,+}$
6 add $(\sigma_d, b^{a_j}_d, t^{a_j}_d)$ to $\mathbf{Reply}^{a_j}_k$
7 add $(\sigma_d, \perp, \infty)$ to $\mathbf{Reply}^{a_j}_k$
8 **Return** $\mathbf{Reply}^{a_j}_k$, $\widehat{P}$

where $\texttt{Depd}^{a_i}(\sigma_m)$ is the set of external assisting actions that σ_m depends on by (2), $\pi_h \in \Pi$ is the region, where σ_m will be performed, and $T_m \geq 0$ is the estimated time when σ_m will be performed from now. Assume that $\langle \pi_h, \sigma_m \rangle$ is the fth element of $\tau^{a_i}_{\mathcal{G},H}$. Then, $T_m = \sum_{s=l}^{f-1} T^{a_i}_{\mathcal{G}}(\tau^{a_i}_{\mathcal{G},H}[s], \tau^{a_i}_{\mathcal{G},H}[s+1])$ (see Lines 4-9 of Algorithm 1). Each element $(\sigma_d, \pi_h, T_m) \in \mathbf{Request}^{a_i}$ contains the message that "agent a_i is requesting the assisting action σ_d at region π_h in the estimated time T_m from now." The request message from agent a_i to each agent $a_j \in \mathcal{N}^{a_i}_t$, denoted by $\mathbf{Request}^{a_i}_{a_j}$, is the same as $\mathbf{Request}^{a_i}$, i.e., $\mathbf{Request}^{a_i}_{a_j} = \mathbf{Request}^{a_i}$, $\forall g \in \mathcal{N}^{a_i}_t$.

Remark 3: Note that the request message is sent only for the first collaborative action in $\tau^{a_i}_{\mathcal{G},H}$ within the time horizon H^{a_i} (see Line 6 of Algorithm 1), as the outcome of this request affects the second collaborative action in $\tau^{a_i}_{\mathcal{G},H}$. ▲

C. Request Evaluation and Reply

Upon receiving the request, agent $a_j \in \mathcal{N}^{a_i}_t$ needs to evaluate this request in terms of *feasibility* and *cost*, in order to reply to agent a_i. Specifically, the reply message from agent a_j to agent a_i has the following format:

$$\mathbf{Reply}^{a_j}_{a_i} = \left\{\left(\sigma_d, b^{a_j}_d, t^{a_j}_d\right), \quad \forall\left(\sigma_d, \pi_h, T_m\right) \in \mathbf{Request}^{a_i}_{a_j}\right\} \tag{5}$$

where σ_d is the requested assisting action by agent a_i, $b^{a_j}_d$ is a Boolean variable indicating the feasibility of agent a_j offering action σ_d at region π_h, and $t^{a_j}_d \geq 0$ is the time when that can happen. Note that due to the communication network, agent a_j can always communicate with a_i either directly or indirectly. We describe how to determine $b^{a_j}_d$ and $t^{a_j}_d$ below.

Denote by $\overline{T}^{a_j} \geq 0$ the estimated finishing time of the current collaboration agent a_i is engaged in. It is initialized as 0 and updated in Section V-F. Algorithm 2 shows the following.

1) If $\overline{T}^{a_j} > 0$, it means agent a_j is engaged in a collaboration. Then, $\mathbf{Reply}^{a_j}_{a_i} = \{(\sigma_d, \perp, \infty), \forall(\sigma_d, \pi_h, T_m) \in \mathbf{Request}^{a_i}_{a_j}\}$, meaning that agent a_j would *reject* any request before its current collaboration is finished.
2) If $\overline{T}^{a_j} = 0$, it means agent a_j is available to offer assisting actions.

Then, for each request $(\sigma_d, \pi_h, T_m) \in \mathbf{Request}^k_g$, agent a_i needs to evaluate it in terms of feasibility and cost to determine $b^{a_j}_d$ and $t^{a_j}_d$.

Clearly, agent a_j needs to potentially revise its current plan to incorporate the request, i.e., to offer the assisting action σ_d at region π_h by estimated time T_m. Denote by $\tau^{a_j}_{\mathcal{G},t^-}$ the plan of agent g before the potential revision, of which the corresponding accepting run is $R^{a_j}_{p,t^-}$. Assume that agent a_j's current state $q^{a_j}_{p,t}$ is the lth element of $R^{a_j}_{p,t^-}$ and the accepting state $q^{a_j}_{p,f}$ is the last and the fth element. Then, the segment from $q^{a_j}_{p,t}$ to $q^{a_j}_{p,f}$ is given by $\widehat{R}^{a_j}_{p,-} = R^{a_j}_{p,t^-}[l:f]$. We intend to find another segment $\widehat{R}^{a_j}_{p,+}$ within $\mathcal{A}^{a_j}_p$ from $q^{a_j}_{p,t}$ to $q^{a_j}_{p,f}$, such that by following $\widehat{R}^{a_j}_{p,+}$: 1) agent a_j should *reach* state $\langle \pi_h, \sigma_d \rangle$; 2) the estimated time to reach $\langle \pi_h, \sigma_d \rangle$ should be *close* to T_m; and 3) the additional cost of $\widehat{R}^{a_j}_{p,+}$ compared with $\widehat{R}^{a_j}_{p,-}$ should be *small*. We enforce those conditions below.

First, the set of product states in $\mathcal{A}^{a_j}_p$ corresponding to $\langle \pi_h, \sigma_d \rangle$ is given by $S_d = \{q_p \in Q^{a_j}_p \,|\, q_p|_{\Pi^{a_j}_{\mathcal{G}}} = \langle \pi_h, \sigma_d \rangle\}$. Consider $\widehat{R}^{a_j}_{p,+}$ with the following structure: $\widehat{R}^{a_j}_{p,+} = q^{a_j}_{p,t} \ldots q^{a_j}_{p,r} \ldots q^{a_j}_{p,f}$, where $q^{a_j}_{p,r} \in S_d$, meaning that it passes through at least one state within S_d. Thus, the corresponding plan would contain $\langle \pi_h, \sigma_d \rangle$, which fulfills the condition 1) above. Regarding conditions 2) and 3), we define the *balanced* cost of $\widehat{R}^{a_j}_{p,+}$

$$\begin{aligned}\texttt{BalCost}&\left(\widehat{R}^{a_j}_{p,+}, T_m, \mathcal{A}^{a_i}_p\right) \\ &= \left|\sum_{s=1}^{r-t} W^{a_j}_p\left(\widehat{R}^{a_j}_{p,+}[s], \widehat{R}^{a_j}_{p,+}[s+1]\right) - T_m\right| \\ &\quad + \alpha^{a_j}\left(\texttt{Cost}\left(\widehat{R}^{a_j}_{p,+}, \mathcal{A}^{a_j}_p\right) - \texttt{Cost}\left(\widehat{R}^{a_j}_{p,-}, \mathcal{A}^{a_j}_p\right)\right)\end{aligned} \tag{6}$$

where the first part stands for the estimated time gap between the requested time T_m by agent a_i and the actual time based on $\widehat{R}^{a_j}_{p,+}$ [for condition 2)], the second term is the additional cost of $\widehat{R}^{a_j}_{p,+}$, compared with $\widehat{R}^{a_j}_{p,-}$ [for condition 3)], and $\alpha^{a_j} > 0$ is a design parameter as a relative weighting.

Problem 3: Given $\widehat{R}^{a_j}_{p,-}$, S_d and $\mathcal{A}^{a_j}_p$, find the path segement $\widehat{R}^{a_j}_{p,+}$ that minimizes (6). ▲

Algorithm 3 solves the above problem by the *bidirectional* Dijkstra algorithm [22]. It utilizes the function $\texttt{DijksTA}(\cdot)$ that computes shortest paths in a weighted graph from the single source state to every state in the set of target states, while at the same time *avoiding* a set of states. It is a simple extension of the classic Dijkstra shortest path algorithm [18].

In Line 4, $\texttt{DijksTA}(\mathcal{A}_p^{a_j}, q_{p,t}^{a_j}, S_d, S_c)$ determines the shortest path (saved in P_1) from $q_{p,t}^{a_j}$ to every state in S_d while avoiding any state belonging to S_c and the associated costs (saved in C_1), where S_c is the set of all product states associated with a collaborative or an assisting action: $S_c = \{q_p \in Q_p^{a_j} \,|\, q_p|_{\Pi_{\mathcal{G}}^{a_j}} = \langle \pi_{\mathcal{G}}^{a_j}, \sigma_n \rangle, \sigma_n \in \Sigma_c^{a_j} \cup \{\Sigma_h^{a_j} \backslash \{\sigma_d\}\}\}$. In Line 5, $\texttt{DijksTA}(\texttt{Reverse}(\mathcal{A}_p^{a_j}), q_{p,f}^{a_j}, S_d, \emptyset)$ is called to determine the shortest path (saved in P_2) from $q_{p,f}^{a_j}$ to every state in S_d within the reversed $\mathcal{A}_p^{a_j}$ and the associated distances (saved in C_2); $\texttt{Reverse}(\mathcal{A}_p^{a_j})$ is the directed graph obtained by inverting the direction of all edges in $G(\mathcal{A}_p^{a_j})$ while keeping the weights unchanged, where $G(\mathcal{A}_p^{a_j})$ is the directed graph associated with $\mathcal{A}_p^{a_j}$ [2]. In Lines 7 and 8, for each state $q_{p,r}^{a_j} \in S_d$, the balanced cost of $\widehat{R}_{p,+}^{a_j}$ by (6) is computed. The one that yields the minimal cost is denoted by $q_{p,r}^{a_j,\star}$. At last, $\widehat{R}_{p,+}^{a_j}$ is formed by concatenating the shortest path from $q_{p,t}^{a_j}$ to $q_{p,r}^{a_j,\star}$ and the reversed shortest path from $q_{p,f}^{a_j}$ to $q_{p,r}^{a_j,\star}$. If $q_{p,r}^{a_j,\star}$ returns empty, then, agent a_i could not offer the requested collaboration thus $b_d^{a_j} = \perp$ and $t_d^{a_j} = \infty$, as in Line 13. The complexity of function $\texttt{DijksTA}(\cdot)$ is $\mathcal{O}(|\delta_p^{a_j}| \cdot \log |Q_p^{a_j}|)$. Reversing $\mathcal{A}_p^{a_j}$ has the complexity linear to $\mathcal{O}(|\delta_p^{a_j}|)$. Note that when $C_1(q_{p,r}^{a_j,\star}) < T_m$, then, agent a_j arrives at the service region before agent a_i and it waits there by performing action σ_0. Moreover, there might be other paths in $\mathcal{A}_p^{a_j}$ that have the same cost by (6) as the one derived by Algorithm 3, i.e., the nonshortest path from $q_{p,t}^{a_j}$ to S_d without the waiting action in the end.

Remark 4: Note that $\widehat{R}_{p,+}^{a_j}$ from Algorithm 3 is the *potentially revised* run, i.e., agent a_j does not change its current plan but saves $\widehat{R}_{p,+}^{a_j}$ in $\widehat{P}$ (see Line 5 of Algorithm 2). ▲

It is worth mentioning that in case agent a_i receives requests from multiple agents, it needs to reply to one agent first and wait for the confirmation before it replies to the next agent.

Lemma 1: If $b_d^{a_j} = \top$ from Algorithm 3, action σ_d can be done at the estimated time $t_d^{a_j}$ by agent a_j following $\widehat{R}_{p,+}^{a_j}$.

Proof: Since the first segment of $\widehat{R}_{p,+}^{a_j}$ from $q_{p,t}$ to $q_{p,r}^{a_j,\star}$ is derived by $\texttt{DijksTA}(\cdot)$ in Line 5 of Algorithm 3, it does not contain any collaborative or assisting actions except σ_d. Thus, it can be accomplished by agent a_j itself with only motions and local actions, of which the estimated time is $t_d^{a_j}$. □

D. Confirmation

Based on the replies from $a_j \in \mathcal{N}_t^{a_i}$, agent a_i needs to acknowledge them by sending back confirmation messages

$$\textbf{Confirm}_{a_j}^{a_i} = \left\{\left(\sigma_d,\, c_d^{a_j},\, f_m\right), \quad \forall \sigma_d \in \texttt{Depd}^{a_i}(\sigma_m)\right\} \tag{7}$$

where σ_d is the requested assisting action, $c_d^{a_j}$ is a Boolean variable, indicating whether agent a_j is confirmed to provide σ_d, and f_m is the estimated time to finish action σ_m.

Algorithm 3 Evaluate the Request, $\texttt{EvalReq}()$

Input: $\widehat{R}_{p,-}^{a_j}$, (σ_d, π_h, T_m), $q_{p,t}^{a_j}$, $\mathcal{A}_p^{a_j}$
Output: $\widehat{R}_{p,+}^{a_j}$, $b_d^{a_j}$, $t_d^{a_j}$

1 $q_{p,t}^{a_j} = \widehat{R}_{p,-}^{a_j}[1]$, $q_{p,f}^{a_j} = \widehat{R}_{p,-}^{a_j}[\text{-}1]$, $\pi_s = q_{p,t}^{a_j}|_{\Pi_{\mathcal{M}}^{a_j}}$
2 Compute S_d, S_c
3 $\bar{c} = \texttt{Cost}(\widehat{R}_{p,-}^{a_j}, \mathcal{A}_p^{a_j})$
4 $(P_1, C_1) = \texttt{DijksTA}\left(\mathcal{A}_p^{a_j}, q_{p,t}^{a_j}, S_d, S_c\right)$
5 $(P_2, C_2) = \texttt{DijksTA}\left(\texttt{Reverse}(\mathcal{A}_p^{a_j}), q_{p,f}^{a_j}, S_d, \emptyset\right)$
6 **forall the** $q_{p,r}^{a_j} \in S_d$ **do**
7 **if** $P_1(q_{p,r}^{a_j})$ *and* $P_2(q_{p,r})$ *exist* **then**
8 $C_3(q_{p,r}^{a_j}) = |C_1(q_{p,r}^{a_j}) - T_m| + \alpha^{a_j}\left(C_1(q_{p,r}^{a_j}) + C_2(q_{p,r}^{a_j}) - \bar{c}\right)$
9 Find the $q_{p,r}^{a_j,\star} \in S_d$ that minimizes $C_3(q_{p,r}^{a_j})$
10 **if** $q_{p,r}^{a_j,\star} \neq \emptyset$ **then**
11 $P = P_1(q_{p,r}^{a_j,\star}) + \texttt{Reverse}\left(P_2(q_{p,r}^{a_j,\star})\right)$
12 **Return** $\widehat{R}_{p,+}^{a_j} = P$, $b_d^{a_j} = \top$, $t_d^{a_j} = C_1(q_{p,r}^{a_j,\star})$
13 **Return** $\widehat{R}_{p,+}^{a_j} = \emptyset$, $b_d^{a_j} = \perp$, $t_d^{a_j} = \infty$

The choices of $\{c_d^{a_j}, a_j \in \mathcal{N}_t^{a_i}\}$ should satisfy two constraints: 1) *exactly* one agent in $\mathcal{N}_t^{a_i}$ can be the confirmed collaborator for each action $\sigma_d \in \texttt{Depd}^{a_i}(\sigma_m)$ and 2) each agent in $\mathcal{N}_t^{a_i}$ can be confirmed for *at most* one action in $\texttt{Depd}^{a_i}(\sigma_m)$. Meanwhile, the estimated finishing time f_m should be as small as possible.

Let $|\mathcal{N}_t^{a_i}| = N_1$ and $|\texttt{Depd}^{a_i}(\sigma_m)| = N_2$. Without the loss of generality, denote by $\mathcal{N}_t^{a_i} = \{1, \ldots, N_1\}$ and $\texttt{Depd}^{a_i}(\sigma_m) = \{\sigma_1, \ldots, \sigma_{N_2}\}$. The problem of finding $\{c_d^{a_j}\}$ and f_m can be readily formulated as an integer programming problem [27]

$$\begin{aligned}
&\textbf{min } f_m \\
&\textbf{s.t. } f_m = \textbf{max}_d\left\{c_d^{a_j} \cdot t_d^{a_j},\, T_m\right\} \\
&\quad \sum_{d=1}^{N_2} b_d^{a_j} \cdot c_d^{a_j} \leq 1 \quad \forall a_j \in \{1, \ldots, N_1\} \\
&\quad \sum_{g=1}^{N_1} b_d^{a_j} \cdot c_d^{a_j} = 1 \quad \forall d \in \{1, \ldots, N_2\}
\end{aligned} \tag{8}$$

where $(\sigma_d, b_d^{a_j}, t_d^{a_j}) \in \textbf{Reply}_{a_i}^{a_j}$ from (5). Any stand-alone integer programming solver can be used to obtain $\{c_d^{a_j}\}$ and f_m once (8) is formulated, e.g., "Gurobi" [14].

Then, $\forall a_j \in \mathcal{N}_t^{a_i}$ and $\forall \sigma_d \in \texttt{Depd}^{a_i}(\sigma_m)$, consider the two cases that are as follows.

1) If (8) has a solution, both $\{c_d^{a_j}\}$ and f_m exist. If $c_d^{a_j}$ is $\top$, add $(\sigma_d, \top, f_m)$ to $\textbf{Confirm}_{a_j}^{a_i}$; otherwise, add $(\sigma_d, \perp, \infty)$ to $\textbf{Confirm}_{a_j}^{a_i}$.
2) If (8) has no solutions, add $(\sigma_d, \perp, \infty)$ to $\textbf{Confirm}_{a_j}^{a_i}$.

It means that σ_m cannot be fulfilled according to the current replies. Then, how agent a_i needs to delay σ_m and revise its plan will be given in Section V-F.

Remark 5: Note that the optimization problem (8) is solved locally by agent a_i regarding its collaborative action σ_m, with $|\mathcal{N}_t^{a_i}| \cdot |\texttt{Depd}^{a_i}(\sigma_m)|$ Boolean variables. ▲

E. Request to Coordinators

As mentioned in Section III-B, the neighboring set $\mathcal{N}_t^{a_i}$ includes only the local neighbors through local communication based on the agent's communication model. Consequently, the requests are limited to the local neighbors and thus the optimization problem (8) might not have a solution at time t. According to the two-layer communication network proposed in Definition 1, the first layer allows agent a_i to communicate with its own and other group coordinators.

In particular, if (8) returns no solution, meaning that agent a_i's collaboration request cannot be fulfilled by its local neighbors. Then, it sends the following *coordination request* to its coordinator $a_{c_k} \in g_k$:

$$\mathbf{CoReq}_{a_{c_k}}^{a_i} = \{(\sigma_d,\, \pi_h,\, T_m), \forall \sigma_d \in \texttt{Depd}^{a_i}(\sigma_m)\} \tag{9}$$

which has the same content as (4), but different headers. Then, the coordinator a_{c_k} would broadcast the same message to the coordinator of every other group $g_{k'} \in \mathcal{K}$ and $g_{k'} \neq g_k$

$$\mathbf{CoReq}_{a_{c_{k'}}}^{a_{c_k}} = \{(\sigma_d,\, \pi_h,\, T_m), \quad \forall \sigma_d \in \texttt{Depd}^{a_i}(\sigma_m)\} \tag{10}$$

where $a_{c_{k'}}$ is the coordinator of group $g_{k'}$. Upon receiving this coordination request, each coordinator $a_{c_{k'}}$ then relays this request to each group members $a_j \in g_{k'}$ and can wait for the *coordination reply* as follows:

$$\mathbf{CoRep}_{a_{c_{k'}}}^{a_j} = \left\{\left(\sigma_d,\, b_d^{a_j},\, t_d^{a_j}\right), \quad \forall (\sigma_d,\, \pi_h,\, T_m) \in \mathbf{CoReq}_{a_{c_{k'}}}^{a_{c_k}}\right\} \tag{11}$$

where $(\sigma_d,\, b_d^{a_j},\, t_d^{a_j})$ is determined by Algorithm 3 as described in Section V-C. Based on these replies that for all $(\sigma_d,\, \pi_h,\, T_m) \in \mathbf{CoReq}_{a_{c_{k'}}}^{a_{c_k}}$, the coordinator $a_{c_{k'}}$ finds the group member $a_{j_d} \in g_{k'}$ satisfying that

$$a_{j_d} = \text{argmin}_{\left\{a_j \in g_{k'},\, b_d^{a_j} = \top\right\}} \left|T_m - t_d^{a_j}\right|, \tag{12}$$

where agent a_{j_d} minimizes the time difference to the expected finish time T_m of the assisting action $\sigma_d \in \texttt{Depd}^{a_i}(\sigma_m)$. Then, the coordination reply from $a_{c_{k'}}$ to a_{c_k} is given by

$$\mathbf{CoRep}_{a_{c_k}}^{a_{c_{k'}}} = \left\{\left(\sigma_d,\, b_d^{a_{j_d}},\, t_d^{a_{j_d}}\right), \quad \forall (\sigma_d,\, \pi_h,\, T_m) \in \mathbf{CoReq}_{a_{c_{k'}}}^{a_{c_k}}\right\}$$

where note that if for an assisting action σ_d, it holds that $b_d^{a_j} = \perp$ holds, $\forall a_j \in g_{k'}$, we set $b_d = \perp$ and $t_d = \infty$.

As a result, the coordinator a_{c_k} passes the coordination replies to agent a_i, which sends the original request. Based on these replies, another integer programming problem is formulated instead

$$\begin{aligned}
&\textbf{min}\ f_m \\
&\textbf{s.t.}\ f_m = \textbf{max}_d\left\{c_d^{a_{j_d}} \cdot t_d^{a_{j_d}},\, T_m\right\} \\
&\quad \sum_{d=1}^{N_2} b_d^{a_{j_d}} \cdot c_d^{a_{j_d}} \leq 1 \quad \forall a_{j_d} \in g_{k'} \in \mathcal{K} \\
&\quad \sum_{g=1}^{K} b_d^{a_{j_d}} \cdot c_d^{a_{j_d}} = 1 \quad \forall \sigma_d \in \texttt{Depd}^{a_i}(\sigma_m)
\end{aligned} \tag{13}$$

where $N_2 = |\texttt{Depd}^{a_i}(\sigma_m)|$ and $\mathbf{CoRep}_{a_{c_k}}^{a_{c_{k'}}} = \{(\sigma_d,\, b_d^{a_{j_d}},\, t_d^{a_{j_d}}), \forall (\sigma_d,\, \pi_h,\, T_m) \in \mathbf{CoReq}_{a_{c_{k'}}}^{a_{c_k}}\}$. Given the solution, the confirmation process is similar to Section V-D, which is omitted here due to limited space.

Remark 6: Compared with (8), (13) is also solved locally by agent a_i regarding its collaborative action σ_m, with $|\mathcal{K}| \cdot |\texttt{Depd}^{a_i}(\sigma_m)|$ Boolean variables. ▲

Finally, it is worth mentioning the tradeoff between the communication cost and the feasibility of the optimization problem by (13). Namely, the more replies are received by agent a_i, the more likely problem (13) would have a solution. An optimal strategy on when the request to coordinators should be sent is part of our future work.

F. Plan Adaptation

After sending out the confirmation messages, agent a_i checks the following.

1) If (8) has a solution, it means that σ_m can be fulfilled by the local neighbors. Else, if (8) has no solution but (13) has a solution, it means that σ_m can be fulfilled by agents outside its neighbor network. Then, $R_{p,t}^{a_i}$ remains unchanged. $\overline{T}^{a_i}$ is set to f_m to indicate that agent a_i is engaged in the collaboration until the estimated time f_m.
2) Otherwise, it means that according to the current replies σ_m cannot be done as planned in $R_{p,t}^{a_i}$. Thus, agent a_i needs to revise its plan by delaying this collaborative action σ_m.

Algorithm 4 revises $R_{p,-}^{a_i}$ and delays σ_m by time D^{a_i}, where $D^{a_i} > 0$ is a design parameter. Function $\texttt{DijksTA}(\cdot)$ from Algorithm 3 is used to find a path from $q_{p,t}^{a_i}$ to one state in S_d whose cost is larger than $T_m + D^{a_i}$ and the accepting state $q_{p,f}^{a_i}$ is reachable from this state. Such a path can always be found as the action σ_0 that takes time T_0 can be repeated as many times as needed.

On the other hand, upon receiving $\mathbf{Confirm}_{a_j}^{a_i}$, each agent $a_j \in \mathcal{N}_t^{a_i}$ checks the following.

1) If $c_d^{a_j} = \top$ for some $\sigma_d \in \texttt{Depd}^{a_i}(\sigma_m)$, then, agent a_j is confirmed to offer the assisting action σ_d. As a result, it modifies its plan based on the potential set of plans $\widehat{P}$ from Algorithm 3. In particular, the plan segment $\widehat{R}_p^{a_j}$ is set to $\widehat{R}_{p,+}^{a_j}$ and $\overline{T}^{a_j}$ is set to f_m.
2) If $c_d^{a_j} = \perp, \forall \sigma_d \in \texttt{Depd}^{a_i}(\sigma_m)$, it means g is not confirmed as a collaborator. Then, $R_p^{a_j}$ remains unchanged and $\overline{T}^{a_j}$ is set to 0.

Afterward, agent a_i and all confirmed collaborators in $\mathcal{N}_t^{a_i}$ would execute its plan by following the motion and action in sequence. They would reject any further request as described in Section V-C until the collaboration for action σ_m is done.

Theorem 2: If (8) or (13) has a solution, the estimated time of accomplishing action σ_m is f_m.

Proof: Since each assisting action $\sigma_d \in \texttt{Depd}^{a_i}(\sigma_m)$ has been assigned exactly to one agent $a_j \in \mathcal{N}_t^{a_i}$, then, σ_d can be accomplished by agent a_j at time $t_d^{a_j}$ by Lemma 1. Thus, σ_m can be accomplished by agent a_i at the estimated time f_m, which is the latest time for all actions by (8) or (13). □

Algorithm 4 Delay Collaboration, `DelayCol()`

Input: $\widehat{R}^{a_i}_{p,-}$, $q^{a_i}_{p,t}$, $\langle \pi_h, \sigma_d \rangle$, $\mathcal{A}^{a_i}_p$, $\widehat{\Sigma}^{a_i}_c$

Output: $\widehat{R}^{a_i}_{p,+}$

compute S_d given $\langle \pi_h, \sigma_d \rangle$, S_c

$q^{a_i}_{p,t} = \widehat{R}^{a_i}_{p,-}[1]$, $q^{a_i}_{p,f} = \widehat{R}^{a_i}_{p,-}[\text{-}1]$

(P_1, C_1) = `DijksTA`$\big(\mathcal{A}^{a_i}_p, q^{a_i}_{p,t}, S_d, S_c\big)$

(P_2, C_2) = `DijksTA`$\big($`Reverse`$(\mathcal{A}^{a_i}_p), q^{a_i}_{p,f}, S_d, \emptyset\big)$

forall the $q^{a_i}_{p,d} \in S_d$ **do**

  if $C_1(q^{a_i}_{p,d}) > T_m + D^{a_i}$ *and* $P_2(q^{a_i}_{p,d})$ *exists* **then**

    $\widehat{R}^{a_i}_{p,+} = P_1(q^{a_i}_{p,d}) +$ `Reverse`$\big(P_2(q^{a_i}_{p,d})\big)$

    Return $\widehat{R}^{a_i}_{p,+}$

Since each agent has a plan as a finite sequence of motion and actions, when one agent finishes executing its plan, it would become an assisting agent by setting $\varphi^{a_i} \triangleq \top$. Then, it would stay at one region and collaborate with others.

G. Loosely Coupled System

As mentioned in Section I, we aim at applying this distributed coordination scheme to loosely coupled multiagent systems, where collaborations among the agents are: sparse in the sense that they are needed infrequently compared with the total number of activities of all agents required by their local tasks; local in the sense that the collaborations are done among neighboring agents. In other words, whenever an agent formulates and solves the coordination problem by (8) or (13), it will always have a solution within a bounded time, when some of its neighboring agents are available and can provide the requested collaborations.

Assumption 3: There exists a finite time $\mathbf{T} > 0$ such that for each agent $a_i \in \mathcal{N}$ and any collaborative action σ_m requested by agent a_i initially at time $t_m > 0$, problem (8) or (13) for σ_m will have a solution within time $t_m + \mathbf{T}$. ▲

Note that the above assumption does not require that (8) or (13) *always* has a solution, rather we allow the collaboration action to be delayed as discussed in Section V-F if no solutions of both can be found. Since the coordination procedure will be repeated afterward, as long as there exits a finite time bound, when either one of the problems has a solution then the collaboration will be accomplished. This is not restrictive for loosely coupled local tasks as sc-LTL formulas because: 1) the large number of agents can provide the requested collaboration and 2) whenever one agent finishes executing its plan, it becomes fully available to collaborate with the others. How to verify this assumption online and dynamically is closely related to the verification of partially feasible LTL task specifications, which is part of our on-going research.

Remark 7: Assumption 3 is necessary to exclude some tightly couple multiagent systems even under sc-LTL task formulas. For instance, assume that agent a_1 has the task to perform collaborative action σ_1 at region r_1 (with the assisting action $\sigma_{d,1}$ from agent a_2) and it cannot cross region r_0 before that. On the other hand, assume that agent a_2 has the task to perform action σ_2 at region r_0 (with the assisting action $\sigma_{d,2}$ from agent a_1) and it cannot cross region r_1 before that. Clearly, both problems (8) or (13) would have no solutions. ▲

VI. Task Swapping Among Homogeneous Agents

As mentioned in Section III, homogeneous agents within the same group have the same motion and action capabilities. In other words, if a local task is feasible to one agent, it is also feasible to another agent in the same group. Thus, it is possible for two homogeneous agents to swap part of their local plans and still satisfy both of their local tasks. In this part, we focus on how this can be done in an efficient way, such that the total cost of satisfying both tasks is reduced. In particular, consider two homogeneous agents $a_i, a_j \in g_k$, where $g_k \in \mathcal{K}$ and their initial plans are given by $\tau^{a_i}_{\mathcal{G},\text{init}}$ and $\tau^{a_j}_{\mathcal{G},\text{init}}$ from Section IV.

A. Formulation

Assume that at time $t > 0$, the state of agent a_i is $\pi^{a_i}_{\mathcal{G},t}$, which is the mth element of $\tau^{a_i}_{\mathcal{G},\text{init}}$ and the state of agent a_j is $\pi^{a_j}_{\mathcal{G},t}$, which is the nth element of $\tau^{a_j}_{\mathcal{G},\text{init}}$. Then, the future plan of agent a_i is given by $\tau^{a_i}_{\mathcal{G}} = \tau^{a_i}_{\mathcal{G},\text{init}}[m{:}]$ and $\tau^{a_j}_{\mathcal{G}} = \tau^{a_j}_{\mathcal{G},\text{init}}[n{:}]$ for agent a_j. Furthermore, denote by $\tau^{a_i,-}_{\mathcal{G}}$ and $\tau^{a_i,+}_{\mathcal{G}}$ the future plan of agent a_i before and after the swapping; $\tau^{a_j,-}_{\mathcal{G}}$ and $\tau^{a_j,+}_{\mathcal{G}}$ for agent a_j. Note that $\tau^{a_i,-}_{\mathcal{G}} = \tau^{a_i}_{\mathcal{G}}$ and $\tau^{a_j,-}_{\mathcal{G}} = \tau^{a_j}_{\mathcal{G}}$.

Following the same notation as before, the segment of $\tau^{a_i,-}_{\mathcal{G}}$ from the l_1th to the l_2th element is given by $\tau^{a_i,-}_{\mathcal{G}}[l_1{:}l_2]$, where $l_2 > l_1$. And, the segment of $\tau^{a_j,-}_{\mathcal{G}}$ from the g_1th to the g_2th element is given by $\tau^{a_j,-}_{\mathcal{G}}[g_1{:}g_2]$, where $g_2 > g_1$. The cost of a plan segment is simply the accumulated weights of each transition along the segment, e.g., $\texttt{cost}(\tau^{a_j,-}_{\mathcal{G}}[g_1{:}g_2]) = \sum_{s=g_1}^{g_2} T^{a_i}_{\mathcal{G}}(\tau^{a_j,-}_{\mathcal{G}}[s], \tau^{a_j,-}_{\mathcal{G}}[s+1])$. Now, we can formally define the plan segments swapping procedure below.

Definition 4: After agents a_i and a_j swap their segments $\tau^{a_i}_{\mathcal{G}}[l_1{:}l_2]$ and $\tau^{a_j}_{\mathcal{G}}[g_1{:}g_2]$ at time t, the updated future plan of agent a_i is given by

$$\tau^{a_i,+}_{\mathcal{G}} = \tau^{a_i,-}_{\mathcal{G}}[m{:}(l_1-1)] \oplus \tau^{a_j,-}_{\mathcal{G}}[g_1{:}g_2] \oplus \tau^{a_i,-}_{\mathcal{G}}[(l_2+1){:}] \tag{14}$$

and the new future plan of agent a_j is given by

$$\tau^{a_j,+}_{\mathcal{G}} = \tau^{a_j,-}_{\mathcal{G}}[n{:}(g_1-1)] \oplus \tau^{a_i,-}_{\mathcal{G}}[l_1{:}l_2] \oplus \tau^{a_j,-}_{\mathcal{G}}[(g_2+1){:}] \tag{15}$$

where $\oplus$ is the concatenation operation of two sequences. This swapping is **allowed** only if two conditions hold: $(\tau^{a_i,-}_{\mathcal{G}}[l_1-1], \tau^{a_j,-}_{\mathcal{G}}[g_1]), (\tau^{a_j,-}_{\mathcal{G}}[g_2], \tau^{a_i,-}_{\mathcal{G}}[l_2+1]) \in \longrightarrow^{a_i}_{\mathcal{G}}$ and $(\tau^{a_j,-}_{\mathcal{G}}[g_1-1], \tau^{a_i,-}_{\mathcal{G}}[l_1]), (\tau^{a_i,-}_{\mathcal{G}}[l_2], \tau^{a_j,-}_{\mathcal{G}}[g_2+1]) \in \longrightarrow^{a_j}_{\mathcal{G}}$. ▲

Namely, only the chosen segments of $\tau^{a_i}_{\mathcal{G}}$ and $\tau^{a_j}_{\mathcal{G}}$ are swapped, while the rest remains unchanged. These two conditions in the end ensure that it is allowed for both agents to start and exit executing the swapped plan segments. Then, we can compute the difference in the total cost of both agents' plans before and after the swapping

$$\Delta_{a_i,a_j} = \Delta_{a_i} + \Delta_{a_j} \tag{16}$$

where $\Delta_{a_i} = \texttt{cost}(\tau_{\mathcal{G}}^{a_i,-}) - \texttt{cost}(\tau_{\mathcal{G}}^{a_i,+})$ is the cost difference for agent a_i, $\Delta_{a_j} = \texttt{cost}(\tau_{\mathcal{G}}^{a_j,-}) - \texttt{cost}(\tau_{\mathcal{G}}^{a_j,+})$ is the cost difference for agent a_j, and $\Delta_{a_i,a_j} \in \mathbb{R}$ is the summed difference. However, consider the original temporal property of $\tau_{\mathcal{G}}^{a_i,-}$ and $\tau_{\mathcal{G}}^{a_j,-}$: the segment $\tau_{\mathcal{G}}^{a_i,-}[l_1{:}l_2]$ to be executed after $\tau_{\mathcal{G}}^{a_i,-}[m{:}(l_1-1)]$ and before $\tau_{\mathcal{G}}^{a_i,-}[(l_2+1){:}]$. Thus, the execution of the swapped segments needs to be synchronized such that the following conditions hold.

1) Agent a_j can start executing $\tau_{\mathcal{G}}^{a_i,-}[l_1{:}l_2]$ only after agent a_i has finished execution $\tau_{\mathcal{G}}^{a_i,-}[m{:}(l_1-1)]$.
2) Agent a_j can start executing $\tau_{\mathcal{G}}^{a_j,-}[(g_2+1){:}]$ only after agent a_i has finished execution $\tau_{\mathcal{G}}^{a_j,-}[g_1{:}g_2]$.
3) Another two analogous conditions for agent a_i.

If any of the above conditions does not hold, either agent a_i or a_j has to wait for the conditions to hold. In order to measure this waiting time, we first calculate the time difference between when agent a_i starts executing $\tau_{\mathcal{G}}^{a_j,-}[g_1{:}g_2]$ and when agent a_j starts executing $\tau_{\mathcal{G}}^{a_i,-}[l_1{:}l_2]$ (denoted by $\Theta^s_{a_i,a_j}$), is given as $\Theta^s_{a_i,a_j} = |\texttt{cost}(\tau_{\mathcal{G}}^{a_i,-}[m{:}(l_1-1)]) - \texttt{cost}(\tau_{\mathcal{G}}^{a_j,-}[n{:}(g_1-1)])|$. The time difference between when agent a_i starts executing $\tau_{\mathcal{G}}^{a_j,-}[g_1{:}g_2]$ and when agent a_j starts executing $\tau_{\mathcal{G}}^{a_i,-}[l_1{:}l_2]$ (denoted by $\Theta^f_{a_i,a_j}$), is given as $\Theta^f_{a_i,a_j} = |\texttt{cost}(\tau_{\mathcal{G}}^{a_i,-}[m{:}(l_1-1)] \oplus \tau_{\mathcal{G}}^{a_j,-}[g_1{:}g_2]) - \texttt{cost}(\tau_{\mathcal{G}}^{a_j,-}[n{:}(g_1-1)] \oplus \tau_{\mathcal{G}}^{a_i,-}[l_1{:}l_2])|$. Thus, the total waiting time for agents a_i and a_j (denoted by Θ_{a_i,a_j}) is given as: $\Theta_{a_i,a_j} = \Theta^s_{a_i,a_j} + \Theta^f_{a_i,a_j}$, which is nonnegative for any choice of the swapping segments.

Problem 4: Find the segment $\tau_{\mathcal{G}}^{a_i,-}[l_1{:}l_2]$ of $\tau_{\mathcal{G}}^{a_i,-}$ and the segment $\tau_{\mathcal{G}}^{a_j,-}[g_1{:}g_2]$ of $\tau_{\mathcal{G}}^{a_j,-}$, such that after the swapping, $\Delta_{a_i,a_j} - \Theta_{a_i,a_j}$ becomes maximal among all possible choices, while $\Delta_{a_i,a_j} > \underline{\Delta}$ and $\Theta_{a_i,a_j} < \overline{\Theta}$ hold, where $\underline{\Delta}$ and $\overline{\Theta} > 0$ are design parameters. ▲

Note that $\underline{\Delta}$ in the above formulation serves as the minimal reduction of the total cost to facilitate a swapping, while $\overline{\Theta}$ is the maximal total waiting time that is allowed. Since agents a_i and a_j are homogeneous, the cost of agent a_i executing the segment $\tau_{\mathcal{G}}^{a_j,-}[g_1{:}g_2]$ is the same as agent a_j. The same arguments hold for agent a_j executing $\tau_{\mathcal{G}}^{a_i,-}[l_1{:}l_2]$. It means that the potential advantage of swapping these two segments lies in the fact that the cost of starting and exiting executing the segments are different. Based on this insight, we calculate the difference in the execution cost for agent a_i before and after the swapping as: $\Delta_{a_i} = \Delta^1_{a_i} + \texttt{cost}(\tau_{\mathcal{G}}^{a_j,-}[g_1{:}g_2]) - \texttt{cost}(\tau_{\mathcal{G}}^{a_i,-}[l_1{:}l_2]) + \Delta^2_{a_i}$, where $\Delta^1_{a_i} = T^{a_i}_{\mathcal{G}}(\tau_{\mathcal{G}}^{a_i,-}[l_1-1],\ \tau_{\mathcal{G}}^{a_j,-}[g_1]) - T^{a_i}_{\mathcal{G}}(\tau_{\mathcal{G}}^{a_i,-}[l_1-1],\ \tau_{\mathcal{G}}^{a_j,-}[l_1])$ and $\Delta^2_{a_i} = T^{a_i}_{\mathcal{G}}(\tau_{\mathcal{G}}^{a_j,-}[g_2],\ \tau_{\mathcal{G}}^{a_i,-}[l_2+1]) - T^{a_i}_{\mathcal{G}}(\tau_{\mathcal{G}}^{a_i,-}[l_2],\ \tau_{\mathcal{G}}^{a_j,-}[l_2+1])$. Analogously for agent a_j, the difference in the execution cost before and after the swapping also consists of three parts: $\Delta_{a_j} = \Delta^1_{a_j} + \texttt{cost}(\tau_{\mathcal{G}}^{a_i,-}[l_1{:}l_2]) - \texttt{cost}(\tau_{\mathcal{G}}^{a_j,-}[g_1{:}g_2]) + \Delta^2_{a_j}$, where $\Delta^1_{a_j} = T^{a_j}_{\mathcal{G}}(\tau_{\mathcal{G}}^{a_j,-}[g_1-1],\ \tau_{\mathcal{G}}^{a_i,-}[l_1]) - T^{a_j}_{\mathcal{G}}(\tau_{\mathcal{G}}^{a_j,-}[g_1-1],\ \tau_{\mathcal{G}}^{a_j,-}[g_1])$ and $\Delta^2_{a_j} = T^{a_j}_{\mathcal{G}}(\tau_{\mathcal{G}}^{a_i,-}[l_2],\ \tau_{\mathcal{G}}^{a_j,-}[g_2+1]) - T^{a_j}_{\mathcal{G}}(\tau_{\mathcal{G}}^{a_j,-}[g_2],\ \tau_{\mathcal{G}}^{a_j,-}[g_2+1])$. By combining Δ_{a_j} and Δ_{a_j} above, we have the total difference: $\Delta_{a_i,a_j} = \Delta^1_{a_i} + \Delta^2_{a_i} + \Delta^1_{a_j} + \Delta^2_{a_j}$, which implies that Δ_{a_i,a_j} consists of the cost differences of agents a_i and a_j for entering and exiting the swapped segments $\tau_{\mathcal{G}}^{a_i,-}[l_1{:}l_2]$ and $\tau_{\mathcal{G}}^{a_j,-}[g_1{:}g_2]$, respectively.

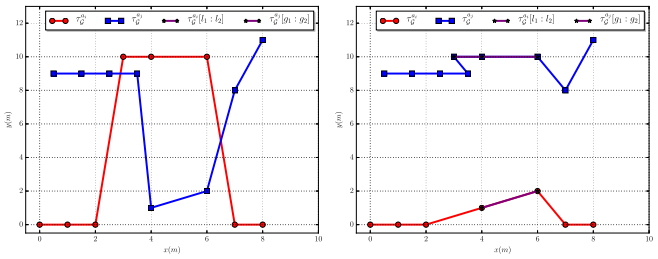

Fig. 1. Discrete plans $\tau_{\mathcal{G}}^{a_i}$ and $\tau_{\mathcal{G}}^{a_j}$ before (left) and after (right) applying the plan swapping scheme. The swapped segments are marked by purple stars. The cost of $\tau_{\mathcal{G}}^{a_i}$ and $\tau_{\mathcal{G}}^{a_j}$ and the total cost are given by (26.1, 22.5, and 48.6) and (9.7, 12.5, and 22.2) before and after swapping, where $\underline{\Delta}$ and $\overline{\Theta}$ are set to 10 and 5, respectively.

As a result, the algorithm that solves Problem 4 is as follows: enumerate all allowed combinations of l_1, l_2, g_1, g_2 that $l_2 > l_1 \geq m$ and $g_2 > g_1 \geq n$ and the two conditions hold; compute the total cost reduction Δ_{a_i,a_j} and the total waiting time Θ_{a_i,a_j}; check if they satisfy that $\Delta_{a_i,a_j} > \underline{\Delta}$ and $\Theta_{a_i,a_j} < \overline{\Theta}$; find the pairs $[l_1{:}l_2]$ and $[g_1{:}g_2]$ that yield the maximal $(\Delta_{a_i,a_j} - \Theta_{a_i,a_j})$. As a result, $\tau_{\mathcal{G}}^{a_i,-}[l_1{:}l_2]$ and $\tau_{\mathcal{G}}^{a_j,-}[g_1{:}g_2]$ are the solution to Problem 4. Note that this algorithm is run locally either by agent a_i or a_j.

B. Plan Swapping and Synchronization

After the segments $\tau_{\mathcal{G}}^{a_i,-}[l_1{:}l_2]$ and $\tau_{\mathcal{G}}^{a_j,-}[g_1{:}g_2]$ are derived as the solution to Problem 4, agents a_i and a_j should exchange the segments and generate the updated plan, as show in (14) and (15). While executing the updated plan, it is crucial that agents a_i and a_j synchronize with each other when they start and finish executing the swapped segments. As discussed earlier, when agent a_i starts executing $\tau_{\mathcal{G}}^{a_i,-}[(l_2+1){:}]$ only after it synchronizes with agent a_j that agent a_j has finished executing $\tau_{\mathcal{G}}^{a_i,-}[l_1{:}l_2]$. Analogous arguments hold for agent a_j. An example is shown in Fig. 1 that the reduction of total cost and the waiting time all fulfill the given constraints.

Note that $\overline{\Theta}$ and $\underline{\Delta}$ in Problem 4 are two design parameters closely related to how often the above task swapping is activated between any two homogeneous agents. It is part of our ongoing work to investigate an online strategy to tune them based on the agent's plan execution status.

VII. Overall Structure

During the real-time execution, each agent executes its plan and checks first if any request is received. If so, it replies to them by Algorithm 2, waits for the confirmation, and adjusts its plan accordingly. Otherwise, it sends out requests by Algorithm 1, waits for reply, sends confirmation back by (8), and at last adapts its plan by Algorithm 4. The correctness of the proposed scheme is guaranteed by Theorem 3.

Theorem 3: Under Assumption 3, the proposed coordination scheme solves Problem 1. Namely, all local tasks φ^{a_i} can be accomplished in finite time, $\forall a_i \in \mathcal{N}$.

Proof: Starting from the initial plan $\tau^{a_i}_{\mathcal{G},\text{init}}$ for agent $a_i \in \mathcal{N}$, motion and local actions in $\tau^{a_i}_{\mathcal{G},\text{init}}$ can be accomplished locally by an agent itself. If $\tau^{a_i}_{\mathcal{G},\text{init}}$ remains unchanged for agent a_i, we only need to show that the collaborative actions in $\tau^{a_i}_{\mathcal{G},\text{init}}$ can be accomplished. The fact that $\tau^{a_i}_{\mathcal{G},\text{init}}$ remains unchanged indicates that agent a_i's requests for each collaborative action σ_m are fulfilled, i.e., (8) or (13) for σ_m has a solution. By Theorem 2, action σ_m can be accomplished in finite time f_m. Since $\tau^{a_i}_{\mathcal{G},\text{init}}$ is a finite sequence, φ^{a_i} can be satisfied in finite time as every motion and action inside can be done in finite time. On the other hand, if $\tau^{a_i}_{\mathcal{G},\text{init}}$ has to be adapted in real time, the reasons are: 1) agent a_i is confirmed to assist its neighbor g on one collaboration and 2) agent a_i has made a request for a collaborative action σ_m and it is delayed by Algorithm 4 as (8) and (13) have no solution. For case 1), Algorithm 3 guarantees that after the assistance its updated run $\hat{R}^{a_i}_{p,+}$ still satisfies φ^{a_i} in finite time. For case 2), by Assumption 3, (8) or (13) will have a feasible solution within at most time $\mathbf{T}$, meaning that σ_m will be done within finite time. This completes the proof. □

VIII. Case Study

We simulate a system of 20 heterogeneous agents with four groups, i.e., each group has five agents. One group is the team of unmanned aerial vehicles (UAVs) (denoted by g_1) and three other groups are different types of unmanned ground vehicles (UGVs) (denoted by g_2, g_3, and g_4). For simplicity, denoted by $g_k = \{a_{k,i}, i = 1, 2, \ldots, 5\}$, $\forall k = 1, 2, 3, 4$ the agents and its group. The proposed algorithms are implemented in Python 2.7. All simulations are carried out on a desktop computer (3.06-GHz Duo CPU and 8-GB of RAM). The mixed integer program solver `Gurobi` for Python [14] is used here.

A. System Description

The workspace we consider is of size 60 m × 60 m representing a clustered environment, as shown in Fig. 2, within which there are four base stations, where the four groups of agents start initially. The workspace is discritized into 1.5-m-width grids to simplify the motion control of each agent. We assume that each agent can travel freely from any grid to its adjacent grid, and the trajectory from an initial grid to a goal grid is given by the shortest path between them. All agents have a constant velocity chosen between 5 and 12 m/s. There are six regions representing the residential areas, which are of major interest to the agents, denoted by $R_1, R_2, \ldots, R_6$. Obstacles are scattered within the workspace. The associated motion FTS consists of 1600 states and 6400 edges. For each UAV $a_{1,i} \in g_1$, it has two actions: `record` to record video and `circle` to circle a certain area. `record` can be done by each UAV itself but `circle` needs two assisting actions $\texttt{hcirclea}$ and $\texttt{hcircleb}$ from two other UAVs. On the other hand, for each UGV $a_{k,i} \in g_k$, there are three storage areas S_1, S_2, and S_3 with three different objects of interest o^l_k, $\forall l = 1, 2, 3$ and $\forall k = 2, 3, 4$. Each of them is capable of providing the actions $\texttt{pick}^l_k$ and $\texttt{drop}^l_k$ for each object $l = 1, 2, 3$, e.g., agent $a_{2,1} \in g_1$ can pickup object o^2_2 with action $\texttt{pick}^2_2$ and drop it with action $\texttt{drop}^2_2$. The dependence between the actions is clarified as follows.

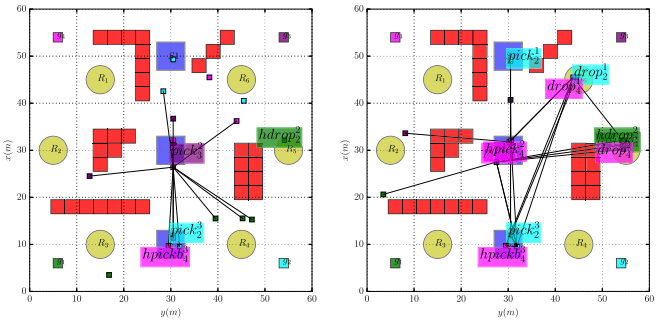

Fig. 2. Snapshots of the simulation at 55.8 and 48.6 s. Agents within the groups g_1, g_2, g_3, and g_4 are squares marked by green, cyan, purple, and magenta, respectively, initially starting from four corners. Regions of interest are labeled by symbols. Communication links between the agents are indicated by straight lines. Action names performed by the agents are highlighted by text boxes. Left: example of local communication by the second layer. Right: communication through coordinators within the first layer.

1) To pickup and drop o^1_k can be done by agent $a_{k,i} \in g_k$ itself, $\forall a_{k,i} \in g_k$ and $\forall k = 2, 3, 4$.
2) To pickup and drop o^2_k by action $\texttt{pick}^2_k$ and $\texttt{drop}^2_k$, agent $a_{k,i}$ needs assisting actions $\texttt{hpick}^2_k$ and $\texttt{hdrop}^2_k$, respectively, from any other UAV $a_j \in g_1$ or UGV $a_j \in g_k$ within the same group, $\forall k = 2, 3, 4$.
3) To pickup o^3_k by action $\texttt{pick}^3_k$, agent $a_{k,i}$ needs assisting action $\texttt{hpicka}^3_{k'}$ from any agent $a_{j'} \in g_{k'}$ and action $\texttt{hpickb}^3_{k''}$ from another agent $a_{j''} \in g_{k''}$. Similarly, to drop o^3_k by action $\texttt{drop}^3_k$, agent $a_{k,i}$ needs assisting action $\texttt{hdropa}^3_{k'}$ from any agent $a_{j'} \in g_{k'}$ and action $\texttt{hdropb}^3_{k''}$ from another agent $a_{j''} \in g_{k''}$, where $k, k', k'' \in \{1, 2, 3, 4\}$ and $k \neq k' \neq k''$, i.e., two agents from two different groups.

We assume that local action can be done in 5 s while the assisting and collaborative actions take 10 s. The communication network satisfies Definition 1 and the communication model is based on the prolixity model with radius set to 15 m. The UAV $a_{1,1}$ and the UGVs $a_{2,1}, a_{3,1}, a_{4,1}$ are chosen as the coordinators of each group.

B. Task Specification

For simplicity, we use the same notation for the atomic propositions that are associated with the regions and actions. Each UAV has the task to surveil three of the residential areas by recording videos at two areas and circling another one. For instance, $\varphi_{1,1} = (\Diamond(R_1 \wedge \texttt{record}_1)) \wedge (\Diamond(R_2 \wedge \texttt{record}_1)) \wedge (\Diamond(R_3 \wedge \texttt{circle}_1)))$. Each UGV has the task to pickup an object from the storage and deliver it to one base station, which has to be done for each object of interest to different base stations. For instance, $\varphi_{2,1} = (\Diamond(\texttt{pick}^1_2 \wedge \Diamond(R_2 \wedge \texttt{drop}^1_2))) \wedge (\Diamond(\texttt{pick}^2_2 \wedge \Diamond(R_4 \wedge \texttt{drop}^2_2))) \wedge (\Diamond(\texttt{pick}^3_2 \wedge \Diamond(R_6 \wedge \texttt{drop}^3_2))))$ and $\varphi_{4,2} = (\Diamond(\texttt{pick}^1_4 \wedge \Diamond(R_3 \wedge \texttt{drop}^1_4))) \wedge (\Diamond(\texttt{pick}^2_4 \wedge \Diamond(R_5 \wedge \texttt{drop}^2_4))) \wedge ((\Diamond\texttt{pick}^3_4 \wedge \Diamond(R_6 \wedge \texttt{drop}^3_4))))$. Note that we need not specify where the UGVs should pickup the objects or where the assisting actions should be performed. The above tasks are all sc-LTL formulas and can be finished in finite time. Due to limited space, we have omitted here the detailed task specification and formulas of other agents. The NBA associated with $\varphi_{2,1}$ consists of 8 states and 27 edges,

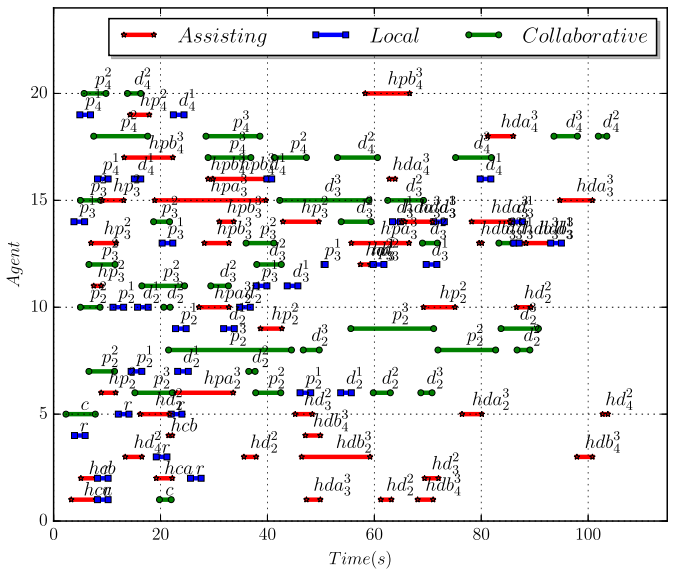

Fig. 3. Image presents the complete sequence of local, assisting, and collaborative actions performed by each agent along with time. Agent identities are converted into integers within [1, 20]. They are labeled by the action names shorten by replacing "pick" with "p," "drop" with "d," "circle" with "c," and "record" with "r."

while the NBA associated with $\varphi_{2,1}$ and $\varphi_{4,2}$ above consists of 46 states and 342 transitions, via [9].

C. Results

The system is simulated for 115 s before all agents accomplish their local tasks, of which some snapshots are shown in Fig. 2. The initial local plan is synthesized, as described in Section IV, e.g., for the UAV $a_{1,1}$ mentioned before, its initial motion and action plan is given by "$\tau_{\mathcal{G}}^{a_{1,1}} = R_1\,\texttt{record}_1\,R_2\,\texttt{record}_1\,R_3\,\texttt{circle}_1$"; while for UGV $a_{2,1}$, its initial plan is given by "$\tau_{\mathcal{G}}^{a_{2,1}} = S_1\,\texttt{pick}_2^1\,R_2\,\texttt{drop}_2^1\,S_3\,\texttt{pick}_2^3\,R_6\,\texttt{drop}_2^3\,S_2\,\texttt{pick}_2^2 R_4\,\texttt{drop}_2^2$." It is worth mentioning that the initial plan of any agent cannot be accomplished by itself as they all consist of at least one collaborative action, which requires the collaboration of other agents. Fig. 3 illustrates the local, assisting, and collaborative actions performed during the simulation. It can be seen that any collaboration for the same collaborative action is not bound to a set of agents with fixed identities, rather on the capabilities and running status of other agents. In addition, the request and reply messages are exchanged among the agents based on the proposed communication protocol. The number of messages exchanged within the team along with time is illustrated in Fig. 4, which implies that the interagent communication is sparse and only triggered by the collaborative actions. The first-layer communication is activated only when the local coordination by (8) fails. The complete simulation video can be found in [25].

Furthermore, to show the effect of the plan swapping scheme mentioned in Section VI, we simulate the system under the same setup while enabling the planning swapping scheme among homogeneous agents within the same group. Specifically, we apply the task swapping scheme after the system starts to the following pairs of agents:

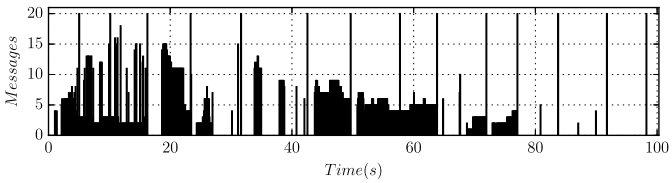

Fig. 4. Number of messages exchanged among the agents along with time. The first-layer communication is activated whenever the number reaches 20.

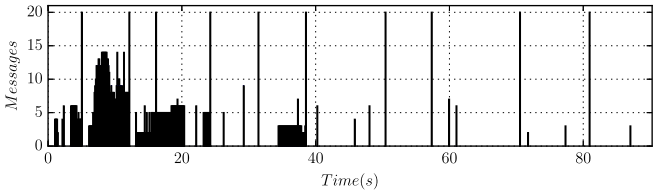

Fig. 5. Number of messages exchanged among the agents along with time after the task swapping scheme at time 0.

agent $a_{1,1}$ swaps its segment "$R_2\texttt{record}$" with the segment "$R_5\texttt{record}$" of $a_{1,2}$; agent $a_{1,3}$ swaps its segment "$R_6\texttt{record}$" with the segment "$R_5\texttt{record}$" of $a_{1,4}$; agent $a_{2,1}$ swaps its segment "$\texttt{pick}_2^2 R_3 \texttt{drop}_2^2$" with the segment "$\texttt{pick}_3^1 R_4 \texttt{drop}_3^1$" of $a_{2,3}$; agent $a_{3,5}$ swaps its segment "$\texttt{pick}_3^3 R_4 \texttt{drop}_3^3$" with the segment "$\texttt{pick}_3^2 R_2 \texttt{drop}_3^2$" of $a_{3,2}$; and agent $a_{4,3}$ swaps its segment "$\texttt{pick}_4^2 R_6 \texttt{drop}_4^2$" with the segment "$\texttt{pick}_4^3 R_2 \texttt{drop}_4^3$" of $a_{4,2}$. Note that the cost reduction $\underline{\Delta}$ and the synchronization delay $\overline{\Theta}$ in Problem 4 are set to be 15 and 10 s. It took 105 s for all agents to accomplish their local tasks. The exchanged messages among the agents is shown in Fig. 5. It can be seen that the collaborations have been shifted to the beginning of the simulation. The simulation video can be found in [25].

D. Computational Complexity

There are 20 agents in the team, each of which has an FTS with 1600 states and 6400 edges. The NBA associated with each UAV, e.g., $\varphi_{1,1}$, consists of 8 states and 27 transitions. The NBA associated with each UGV, e.g., $\varphi_{2,1}$, consists of 46 states and 342 transitions. As a result, the centralized approach requires computing the product automaton between the FTS and NBA for the whole team, which would have $1600^{20} \cdot 8^5 \cdot 46^{15}$ states (approximately 3.5×10^{93}). Thus, the centralized approach would be intractable for most practical applications. On the other hand, our proposed approach relies on only local plan synthesis, local communication, and local coordination. The local product automaton has around 10^5 states and the algorithm to synthesize the local policy in Section IV takes 2 s in average for all agents. The communication protocols by Algorithms 1 and 2 rely on exchanging simple string messages. The coordination Algorithms 3 and 4 are run locally by each agent.

IX. Summary and Future Work

We present a bottom–up scheme for distributed motion and task coordination of multiagent systems, where the agents are given dependent local tasks. It relies on the off-line initial plan synthesis, the online request and reply messages exchange protocol, and the real-time plan adaptation algorithm. A task

swapping scheme is proposed to reduce the total execution cost of the system. Future work is focused on general LTL task formulas, which are not considered here, since ensuring fairness is challenging when each agent has a local plan as an infinite sequence of motion and actions.

REFERENCES

[1] A. Bhatia, L. E. Kavraki, and M. Y. Vardi, "Sampling-based motion planning with temporal goals," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2010, pp. 2689–2696.

[2] C. Baier and J.-P. Katoen, *Principles of Model Checking*. Cambridge, MA, USA: MIT Press, 2008.

[3] K. E. Bekris, K. I. Tsianos, L. E. Kavraki, "Safe and distributed kinodynamic replanning for vehicular networks," *Mobile Netw. Appl.*, vol. 14, no. 3, pp. 292–308, Jun. 2009.

[4] C. Belta, A. Bicchi, M. Egerstedt, E. Frazzoli, E. Klavins, and G. J. Pappas, "Symbolic planning and control of robot motion," *IEEE Robot. Autom. Mag.*, vol. 14, no. 1, pp. 61–70, Mar. 2007.

[5] C.-H. Chang, S.-C. Wang, and C. C. Wang, "Exploiting moving objects: Multi-robot simultaneous localization and tracking," *IEEE Trans. Autom. Sci. Eng.*, vol. 13, no. 2, pp. 810–827, Apr. 2016.

[6] X. C. Ding, M. Kloetzer, Y. Chen, and C. Belta, "Automatic deployment of robotic teams," *IEEE Robot. Autom. Mag.*, vol. 18, no. 3, pp. 75–86, Sep. 2011.

[7] G. E. Fainekos, A. Girard, H. Kress-Gazit, and G. J. Pappas, "Temporal logic motion planning for dynamic robots," *Automatica*, vol. 45, no. 2, pp. 343–352, Feb. 2009.

[8] G. E. Fainekos, "Revising temporal logic specifications for motion planning," in *Proc. IEEE Conf. Robot. Autom.*, May 2011, pp. 40–45.

[9] P. Gastin and D. Oddoux, "Fast LTL to Büchi automaton translation," in *Proc. Int. Conf. Comput. Aided Verification*, Jul. 2001, pp. 53–65.

[10] M. Guo, K. H. Johansson, D. V. Dimarogonas, "Motion and action planning under LTL specifications using navigation functions and action description language," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Nov. 2013, pp. 240–245.

[11] M. Guo, D. V. Dimarogonas, "Reconfiguration in motion planning of single- and multi-agent systems under infeasible local LTL specifications," in *Proc. IEEE Conf. Decision Control (CDC)*, Dec. 2013, pp. 2758–2763.

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

[13] M. Guo, D. V. Dimarogonas, "Bottom-up motion and task coordination for loosely-coupled multi-agent systems with dependent local tasks," in *Proc. IEEE Int. Conf. Autom. Sci. Eng. (CASE)*, Aug. 2015, pp. 348–355.

[14] *Gurobi Optimization Tool*. Accessed 2015. [Online]. Available: http://www.gurobi.com/

[15] S. Karaman and E. Frazzoli, "Vehicle routing with linear temporal logic specifications: Applications to multi-UAV mission planning," in *Proc. Control Conf. AIAA Guid. Navigat.*, 2008.

[16] M. Kloetzer, X. C. Ding, and C. Belta, "Multi-robot deployment from LTL specifications with reduced communication," in *Proc. IEEE Conf. Decision Control (CDC)*, Dec. 2011, pp. 4867–4872.

[17] O. Kupferman and M. Y. Vardi, "Model checking of safety properties," *Formal Methods Syst. Design*, vol. 19, no. 3, pp. 291–314, Nov. 2001.

[18] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 2006.

[19] S. G. Loizou and K. J. Kyriakopoulos, "Automatic synthesis of multi-agent motion tasks based on LTL specifications," in *Proc. IEEE Conf. Decision Control (CDC)*, Dec. 2005, pp. 78–83.

[20] G. Notarstefano, M. Egerstedt, and M. Haque, "Containment in leader–follower networks with switching communication topologies," *Automatica*, vol. 47, no. 5, pp. 1035–1040, May 2011.

[21] T. Wongpiromsarn, U. Topcu, and R. M. Murray, "Receding horizon control for temporal logic specifications," in *Proc. 13th ACM Int. Conf. Hybrid Syst., Comput. Control*, Apr. 2010, pp. 101–110.

[22] I. Pohl, "Bidirectional search," *Mach. Intell.*, vol. 6, pp. 127–140, 1971.

[23] R. Sampath, H. Darabi, U. Buy, and J. Liu, "Control reconfiguration of discrete event systems with dynamic control specifications," *IEEE Trans. Autom. Sci. Eng.*, vol. 5, no. 1, pp. 84–100, Jan. 2008.

[24] J. Tumova and D. V. Dimarogonas, "A receding horizon approach to multi-agent planning from local LTL specifications," in *Proc. Amer. Control Conf. (ACC)*, Jun. 2014, 1775–1780.

[25] *Videos*, Accessed on 2015. [Online]. Available: https://vimeo.com/148774433 and https://vimeo.com/148774898

[26] A. Ulusoy, S. L. Smith, X. C. Ding, C. Belta, D. Rus, "Optimality and robustness in multi-robot path planning with temporal logic constraints," *Int. J. Robot. Res.*, vol. 32, no. 8, pp. 889–911, Jul. 2013.

[27] L. A. Wolsey, *Integer Programming*. New York, NY, USA: Wiley, 1998.

Meng Guo received the M.Sc. degree in system, control and robotics and the Ph.D. degree in electrical engineering from the KTH Royal Institute of Technology, Stockholm, Sweden, in 2011 and 2016, respectively.

He is currently a Post-Doctoral Associate with the Department of Mechanical Engineering and Materials Science, Duke University, Durham, NC, USA. His current research interests include distributed motion and task planning of multi-agent systems and formal control synthesis.

Dimos V. Dimarogonas was born in Athens, Greece, in 1978. He received the Diploma degree in electrical and computer engineering and the Ph.D. degree in mechanical engineering from the National Technical University of Athens, Zografou, Greece, in 2001 and 2007, respectively.

Between 2007 and 2009, he was a Post-Doctoral Researcher with the Automatic Control Laboratory, School of Electrical Engineering, ACCESS Linnaeus Center, KTH Royal Institute of Technology, Stockholm, Sweden. Between 2009 and 2010, he was a Post-Doctoral Associate at the Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Boston, MA, USA. He is currently an Associate Professor with the Automatic Control Laboratory, ACCESS Linnaeus Center, KTH Royal Institute of Technology. His current research interests include multiagent systems, hybrid systems and control, robot navigation and networked control. He was awarded a Docent in Automatic Control from KTH in 2012. He received an ERC Starting Grant from the European Commission for the proposal BUCOPHSYS in 2014 and was awarded a Wallenberg Academy Fellow Grant in 2015.

Prof. Dimarogonas serves on the Editorial Board of *Automatica*, the IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING, and the *IET Control Theory and Applications*. He is a Technical Chamber of Greece.