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 complex robot tasks and practical implementations.

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

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 approach. In [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 approach. In [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 approach.

### 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]: \( \phi ::= T \mid p \mid \phi_1 \land \phi_2 \mid \neg \phi \mid \diamond \phi \mid \phi_1 \cup \phi_2 \), where \( T \equiv \text{true} \), \( p \in \text{AP} \) and \( \diamond \) (next), and \( U \) (until). For brevity, we omit the derivations of other useful operators like \( \square \) (always), \( \lozenge \) (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 \( \phi \) consists of a satisfying prefix that can be followed by an arbitrary suffix. A language of words that satisfy an LTL formula \( \phi \) over \( \text{AP} \) can be captured through a nondeterministic Büchi automaton (NBA) \( \mathcal{A}_\phi \), defined as \( \mathcal{A}_\phi = (Q, \mathcal{AP}, \delta, Q_0, F) \), where \( Q \) is a set of states, \( \mathcal{AP} \) is the set of all alphabets, \( \delta \subseteq Q \times \mathcal{AP} \times Q \) is a transition relation, and \( Q_0 \) and \( F \subseteq Q \) 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 \( v = s_1 s_2 \ldots s_N \), where \( s_i \in S \), \( i = 1, \ldots, N \). Denote by \( v[j] \), the \( j \)th state along \( v \), i.e., \( v[j] = s_j \); \( v[i:] \) the segment from the \( i \)th to the \( j \)th element for \( j \geq i \) (including both); \( v[i:] \) the segment from the \( i \)th 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 \( N = \{a_i, i = 1, 2, \ldots, N\} \) the agent index. Then, we assume \( K \) groups within the team, denoted by \( K = \{g_k, k = 1, 2, \ldots, K\} \). 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 \( g_k \subseteq N \), \( \cup_{g_k \in \mathcal{K}} g_k = N \), 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 \( \text{a priori} \) 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 \( \Psi^{a_i}_{M} \). Similar to [10], agent \( a_i \)'s motion within the workspace is modeled as a finite transition system (FTS)

\[
\mathcal{M}^{a_i} \triangleq (\Pi, \rightarrow_{M}, \Pi_{M,0}^{a_i}, \Psi^{a_i}_{M}, L^{a_i}_{M}, T^{a_i}_{M})
\]
where $a_t^{ai} \subseteq \Sigma_1 \times \Pi$ is the transition relation, $\Pi_{M,0}^{ai} \in \Pi$ is the initial region agent $a_i$ starts from, $L_{M}^{ai} : \Pi \rightarrow 2^{\Sigma_1 M}$ is the labeling function, indicating the properties held by each region, and $T_{M}^{ai} : a_t^{ai} \rightarrow \mathbb{R}^+$ gives the time each transition takes. A path of $M^{ai}$ is a sequence of regions $\sigma_0 \pi_1 \ldots \pi_p$, where $(\pi_p, \pi_{p+1}) \in a_t^{ai}$, $\forall p = 0, 1, \ldots, P - 1$. Note that $M^{ai}$ 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 $N_i^{ai} \subseteq \mathcal{N}$. Agent $a_i$ can exchange messages directly with any agent $a_j \in N_i^{ai}$. 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 $a_k \in g_k, \forall g_k \in \mathcal{K}$ that $\mathcal{E}_1(t) = \{(a_{i_1}, a_{i_2}, a_{i_3}), \forall g_k \in \mathcal{K}\}$.
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 $a_k$ 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 $N_i^{ai} = \{a_j \in \mathcal{N} | (a_i, a_j) \in \mathcal{E}(t)\}$. 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 $\Sigma_1^{ai} \triangleq \Sigma_1^a \cup \Sigma_1^c \cup \Sigma_1^h$, where $\Sigma_1^a$ is a set of local actions, which can be done by agent $a_i$ itself; $\Sigma_1^c$ is a set of collaborative actions, which can be done by agent $a_i$ but requires collaborations from other agents; and $\Sigma_1^h$ is a set of assisting actions, in which agent $a_i$ offers to other agents to accomplish their collaborative actions.

In other words, $\Sigma_1^a$ and $\Sigma_1^c$ contain actions that can be initiated by agent $a_i$, denoted by $\Sigma_1^a = \Sigma_1^a \cup \Sigma_1^c$, while $\Sigma_1^h$ contains assisting actions only to assist other agents. By default, $\sigma_0 = \emptyset \in \Sigma_1^a$ means that none of the actions is performed. Moreover, denote by $\Sigma_1^{-ai}$ the set of external assisting actions agent $a_i$ depends on, which can be provided by other agents in $\mathcal{N}$, i.e., $\Sigma_1^{-ai} \subseteq \bigcup_{a_j \in \mathcal{N}} \Sigma_1^{aj}$. The action model of agent $a_i$ is modeled by a six-tuple:

$$A^{ai} \triangleq (\Sigma_1^{ai}, \Psi_2^{ai}, L_2^{ai}, \text{Cond}^{ai}, \text{Dura}^{ai}, \text{Depd}^{ai}) \quad (2)$$

where $\Sigma_1^{ai}$ is the set of actions defined earlier, $\Psi_2^{ai}$ is a set of atomic propositions related to the agent’s active actions, and $L_2^{ai} : \Sigma_1^{ai} \rightarrow \mathbb{R}^+$ is the labeling function. $L_2^{ai}(\sigma_0) = \emptyset$, $\forall \sigma_0 \in \Sigma_1^{ai}$ and $L_2^{ai}(\sigma_0) \subseteq \Psi_2^{ai}, \forall \sigma_0 \in \Sigma_1^{ai}$; $\text{Cond}^{ai} : \Sigma_1^{ai} \times 2^{\Sigma_1 M} \rightarrow \mathbb{T}$ gives the time each execution is performed in order to perform an action; $\text{Dura}^{ai} : \Sigma_1^{ai} \rightarrow \mathbb{R}^+$ is the estimated time duration of each action. $\text{Dura}^{ai}(\sigma_0) = T_0 > 0$ is a design parameter and $\text{Depd}^{ai} : \Sigma_1^{ai} \rightarrow 2^{\Sigma_1 M}$ is the dependence function. $\text{Depd}^{ai}(\sigma_i) = \emptyset$, $\forall \sigma_i \in \Sigma_1^{ai} \cup \Sigma_1^h$ and $\text{Depd}^{ai}(\sigma_i) \subseteq \Sigma_1^{ai}$, $\forall \sigma_i \in \Sigma_1^{ai}$. 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 $\sigma_j \in \Sigma_1^{ai} \cup \Sigma_1^{aj}$ is said to be done at region $\pi_k \in \Pi$ if two conditions hold the following.

1. $\text{Cond}^{ai}(\sigma_j, L_2^{ai}(\sigma_j)) = \top$.
2. $\sigma_j$ is activated for period $\text{Dura}^{ai}(\sigma_j)$. For a collaborative action $\sigma_c \in \Sigma_1^{ci}$, another condition is needed.
3. All assisting actions in $\text{Depd}^{ai}(\sigma_i)$ are done by other agents at the same region $\pi_k$.

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 $G^{ai}$, refers to the FTS that models both its motion and actions.

**Definition 3**: Given $M^{ai}$ and $A^{ai}$, agent $a_i$’s complete model can be constructed as follows: $G^{ai} = (\Pi_G^{ai}, \rho^{ai}_G, \pi_G^{ai}, L_G^{ai}, T_G^{ai})$, where $\Pi_G^{ai} = \Pi \times \Sigma_1^{ai}, \pi_G^{ai} = \langle \pi_j, \sigma_0 \rangle \in \Pi_G^{ai}, \forall \pi_j, \forall \sigma_0 \in \Sigma_1^{ai}; \rho^{ai}_G(\pi_j, \sigma_0) = \emptyset$ for $\langle \pi_j, \sigma_0 \rangle \in \Pi_G^{ai}$. \(L_G^{ai}(\pi_j, \sigma_0)\) is the set of actions, $\forall \sigma_0$, $\forall \sigma \neq \sigma_0$, $\pi_i = \pi_j$, $\text{Cond}^{ai}(\sigma_i, L_G^{ai}(\pi_i)) = \top$; or 3) $\sigma_i \in \Sigma_1^{ai}, \sigma_0 = \sigma_0, \pi_i = \pi_j$; $\Pi_G^{ai}_0 = \Pi_G^{ai}, \sigma_0 = \emptyset$; $\rho^{ai}_G(\pi_j, \sigma_0) = \emptyset$. $T_G^{ai}(\pi_j, \sigma_0) = T_0$.

Note that when defining $\rho^{ai}_G$, the condition of performing an action is verified over the properties of each region. Thus, $G^{ai}$ is a standard FTS [2]. Its infinite path is denoted by $\tau^{ai} = \pi_G^{ai}(\pi_1, \ldots, \pi_{P+1}), \forall \pi_i \in \Pi_G^{ai}, \pi_G^{ai} = \Pi_G^{ai}_0$, and $(\pi_{G,i}, \pi_{G,i+1}) \in a_t^{ai}$, $\forall \pi_i = 0, \ldots, P - 1$. Its trace is $\text{trace}(\tau^{ai}) = L_G^{ai}(\pi_0, \pi_1, \ldots, \pi_{P+1})$. Where $\pi_i \in \Pi_G^{ai}_0$, and $(\pi_{G,i+1}, \pi_{G,i+1+1})$.
E. Task Specification

The local task of agent \( a_i \), denoted by \( \varphi^{a_i} \), is given as an sc-safe LTL formula over the set of atomic propositions \( \Psi_Q^G \) from Definition 3. Thus, \( \varphi^{a_i} \) 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 \( \tau_i \) of \( G^a_i \), then \( \tau_i \) fulfills \( \varphi^{a_i} \) if \( \text{trace}(\tau_i) \models \varphi^{a_i} \) where the satisfaction relation is defined in Section II-A. One special case is that when \( \varphi^{a_i} \models \top \), 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 \( G^a_i \) and the locally assigned task \( \varphi^{a_i} \), design a distributed control and coordination scheme such that \( \varphi^{a_i} \) is fulfilled for all \( a_i \in N \).

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 \( G^a_i \) whose trace satisfies the cosafe formula \( \varphi^{a_i} \), 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 \( \varphi^{a_i} \), i.e.,

\[
\mathcal{A}^{\varphi^{a_i}} = (Q^{\varphi^{a_i}}, 2^Q^{\varphi^{a_i}}, \delta^{\varphi^{a_i}}, Q_0^{\varphi^{a_i}}, F^{\varphi^{a_i}})
\]

of states \( \mathcal{Q}^{\varphi^{a_i}} \) served as the starting point for the real-time coordination and adaptation scheme in Section V.

\[
\text{A}^{\varphi^{a_i}} = \mathcal{A}^{\varphi^{a_i}} \otimes \mathcal{A}^{a_i} = (Q_{p,0}^{\varphi^{a_i}}, Q_{p,0}^{\varphi^{a_i}} \otimes Q_{p,0}^{a_i}, F_{p,0}^{\varphi^{a_i}} \otimes F_{p,0}^{a_i}, W_{p,0}^{\varphi^{a_i}} \otimes W_{p,0}^{a_i})
\]

where \( Q_{p,0}^{\varphi^{a_i}} = (\varphi^{a_i}, q_0) \in Q_{p,0}^{\varphi^{a_i}}, \forall q_0 \in Q_{p,0}^{\varphi^{a_i}}, \forall \varrho_{\varphi^{a_i}} \in \varrho_{\varphi^{a_i}}, \forall q_0 \in Q_{p,0}^{\varphi^{a_i}} \lhd (\varphi^{a_i}, q_0, (\varphi^{a_i}, q_0)) \in \delta_{p,0}^{\varphi^{a_i}} \) if \( \varphi^{a_i} \rightarrow_{p,0} q_{p,0} \) is the set of initial states; \( F_{p,0}^{\varphi^{a_i}} = \varrho_{\varphi^{a_i}} \times F_{p,0}^{\varphi^{a_i}} \) is the set of accepting states; \( W_{p,0}^{\varphi^{a_i}} = \delta_{p,0}^{\varphi^{a_i}} \times W_{p,0}^{\varphi^{a_i}} \rightarrow \mathbb{R}^{+} \). \( W_{p,0}^{\varphi^{a_i}} ((\varphi^{a_i}, q_m), (\varphi^{a_i}, q_n)) = T_{p,0}^{\varphi^{a_i}} (\varphi^{a_i}, q_m) \), where \( (\varphi^{a_i}, q_m) \in \delta_{p,0}^{\varphi^{a_i}} ((\varphi^{a_i}, q_m)) \).

There exists a finite path of \( \mathcal{A}^{\varphi^{a_i}} \) satisfying \( \varphi^{a_i} \) if and only if \( \mathcal{A}^{\varphi^{a_i}} \) has a finite path from an initial state to an accepting state. Then, this path could be projected back to \( G^a_i \) as a finite path, the trace of which should satisfy \( \varphi^{a_i} \) automatically [2].

\[
\text{cost}(R_{p,0}^{\varphi^{a_i}}, A_{p,0}^{a_i}) = \sum_{i=0}^{P-1} W_{p,0}^{a_i} (q_{p,i}, q_{p,i+1})
\]

is the summed weights along \( R_{p,0}^{\varphi^{a_i}} \). The 5th element is defined by \( R_{p,0}^{\varphi^{a_i}}[s] = q_{p,s}^{a_i} \) and the segment from the \( s \)th to the \( l \)th element is \( R_{p,0}^{\varphi^{a_i}}[s:l] = q_{p,s}^{a_i} \ldots q_{p,l}^{a_i} \), where \( s \leq l \leq P \).

Problem 2: Find a finite path \( R_{p,0}^{\varphi^{a_i}} \) of \( \mathcal{A}^{\varphi^{a_i}} \) with the above structure that minimizes its total cost.

Denote by \( R_{p,0}^{\varphi^{a_i}} \) 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 \( F_{p,0}^{\varphi^{a_i}} \) and checks if there is cycle back to \( q_{p,0} \).

The worst case complexity is \( O(|Q_{p,0}^{\varphi^{a_i}}| \cdot |Q_{p,0}^{a_i}|) \). By projecting the worst case onto \( Q_{p,0}^{\varphi^{a_i}} \), it gives the initial motion and action plan \( R_{p,0}^{\varphi^{a_i}} \) that fulfills \( \varphi^{a_i} \).

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

The plan \( R_{p,0}^{\varphi^{a_i}} \) can be executed by activating the motion or actions in sequence. However, since \( R_{p,0}^{\varphi^{a_i}} \) may contain several collaborative actions from \( \tau_i \), to satisfy \( \varphi^{a_i} \), the successful execution of \( R_{p,0}^{\varphi^{a_i}} \) depends on other agents’ collaboration, which, however, is not guaranteed, since \( R_{p,0}^{\varphi^{a_i}} \) 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 \( R_{p,0}^{\varphi^{a_i}} \) 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 \( \pi^{a_i,t} \in \Pi^a_i \) the state of agent \( a_i \) at time \( t \). After the system starts, assume \( \pi^{a_i,0} \) is the 4th element in \( t_{p,0}^{a_i} \), namely, \( \pi^{a_i,t} = t_{p,0}^{a_i}[4] \). Each agent \( a_i \in N \) 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 \( F_{p,t}^{a_i,H} = \tau_{p,t}^{a_i,H}[1:H] \), where the index \( f \geq 1 \) is the solution to this optimization problem: \( \min f, \text{subject to } \sum_{i=1}^{f-1} \tau_{p,t}^{a_i,H}[i] \leq H^{a_i} \). It can be solved by iterating through the sequence of \( \tau_{p,t}^{a_i,H} \) 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 \( \tau_{p,t}^{a_i,H}[1:] \), then \( f = \tau_{p,t}^{a_i,H}[1:] \) (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 \( \tau_{p,t}^{a_i,H} \) as the motion and actions in horizon, agent \( a_i \) needs to check whether it needs others’ collaboration within \( F_{p,t}^{a_i,H} \). This is done by verifying whether a collaborative action needs to be performed to reach the states in \( F_{p,t}^{a_i,H} \). More specifically, for the first state \( (\pi_{h},\pi_{m}) \in F_{p,t}^{a_i,H} \) satisfying \( \pi_{m} \in \pi_{c}^{a_i} \), agent \( a_i \) needs to broadcast a request to all agents within its communication network \( \Pi^{a_i} \) regarding this action.

This request message has the following format:

\[
\text{Request}^{a_i} = (\sigma_d, \pi_{1}, T_m), \quad \forall \sigma_d \in \text{Dep}\pi^{a_i}(\pi_{m})
\]
can always communicate with agent ai and updated in Section V-F. Algorithm 2 shows the following.

\begin{algorithm}
\caption{Reply to Request by Agent ai, Reply()}
\begin{algorithmic}
\State {Input: $\text{Request}_{ai}^{\sigma}$, $R_{p_1}^{\sigma}$, $A_p^{\sigma}$, $T_m^{\sigma}$}
\State {Output: $\hat{R}_{p_1}^{\sigma}$, $\hat{P}$}
\State {1 for all $\langle \sigma_d, \pi_h, T_m \rangle \in \text{Request}_{ai}^{\sigma}$ do}
\State {2 \indent if $T_m$ is available to offer $\hat{R}_{p_1}^{\sigma}$ then}
\State {3 \indent \indent $\hat{R}_{p_1}^{\sigma} = \{\langle \sigma_d, \pi_h, T_m \rangle \}$}
\State {4 \indent \indent return $\hat{R}_{p_1}^{\sigma}$}
\State {5 \indent \indent \indent $\hat{P}(\sigma_d, \pi_h, T_m)$}
\State {6 \indent \indent \indent add $(\sigma_d, b^a_{d}, t^a_{d})$ to Reply$_k^{\sigma}$}
\State {7 \indent \indent return $\hat{R}_{p_1}^{\sigma}$, $\hat{P}$}
\State {8 \indent \indent $\hat{R}_{p_1}^{\sigma}$, $\hat{P}$}
\State {9 \indent \indent $\hat{R}_{p_1}^{\sigma}$, $\hat{P}$}
\State {10 return $\hat{R}_{p_1}^{\sigma}$, $\hat{P}$}
\end{algorithmic}
\end{algorithm}

Then, for each request $(\sigma_d, \pi_h, T_m) \in \text{Request}_{ai}^{\sigma}$, agent ai needs to evaluate it in terms of feasibility and cost to determine $b^a_{d}$ and $t^a_{d}$.

Clearly, agent ai needs to potentially revise its current plan to incorporate the request, i.e., to offer the assisting action $\sigma_d$ at region $\pi_h$ by estimated time $T_m$. Denote by $\hat{R}_{p_1}^{\sigma}$ the plan of agent $g$ before the potential revision, of which the corresponding accepting run is $R_{p_1}^{\sigma}$. Assume that agent ai’s current state $q_{p_1,t}$ is the $t$th element of $R_{p_1}^{\sigma}$ and the accepting state $q_{p_1,t}$ is the last and the $t$th element. Then, the segment from $q_{p_1,t}$ to $q_{p_1,f}$ is given by $R_{p_1}^{\sigma} = R_{p_1}^{\sigma}[l:f]$. We intend to find another segment $R_{p_1}^{\sigma}$ within $A_p^{\sigma}$ from $q_{p_1,t}$ to $q_{p_1,f}$, such that by following $R_{p_1}^{\sigma}$: 1) agent ai should reach state $(\pi_h, \sigma_d)$; 2) the estimated time to reach $(\pi_h, \sigma_d)$ should be close to $T_m$; and 3) the additional cost of $R_{p_1}^{\sigma}$ compared with $R_{p_1}^{\sigma}$ should be small. We enforce those conditions below.

First, the set of product states in $A_p^{\sigma}$ corresponding to $(\pi_h, \sigma_d)$ is given by $S_d = \{q_{p_1} \in Q_p^{\sigma} | q_{p_1}^{[t]} = (\pi_h, \sigma_d)\}$. Consider $R_{p_1}^{\sigma}$ with the following structure: $R_{p_1}^{\sigma} = q_{p_1,t}^{0}, q_{p_1,t}^{1}, \ldots, q_{p_1,f}^{t}$, where $q_{p_1,r} \in S_d$, meaning that it passes through at least one state within $S_d$. Thus, the corresponding plan would contain $(\pi_h, \sigma_d)$, which fulfills the condition 1) above. Regarding conditions 2) and 3), we define the balanced cost of $R_{p_1}^{\sigma}$.

\begin{equation}
\text{BalCost}(R_{p_1}^{\sigma}, T_m, A_p^{\sigma}) = \\
\sum_{s=1}^{t-1} W_p^{\sigma}(\hat{R}_{p_1}^{\sigma}[s], \hat{R}_{p_1}^{\sigma}[s + 1] - T_m) \nonumber + a^{\sigma}(\text{Cost}(\hat{R}_{p_1}^{\sigma}, A_p^{\sigma}) - \text{Cost}(\hat{R}_{p_1}^{\sigma}, A_p^{\sigma}))
\end{equation}

where the first part stands for the estimated time gap between the requested time $T_m$ by agent ai and the actual time based on $\hat{R}_{p_1}^{\sigma}$ (for condition 2), the second term is the additional cost of $R_{p_1}^{\sigma}$, compared with $R_{p_1}^{\sigma}$ (for condition 3), and $a^{\sigma} > 0$ is a design parameter as a relative weighting.

**Problem 3:** Given $\hat{R}_{p_1}^{\sigma}$, $S_d$, and $A_p^{\sigma}$, find the path segment $\hat{R}_{p_1}^{\sigma}$ that minimizes (6). ▲
Algorithm 3 solves the above problem by the bidirectional Dijkstra algorithm [22]. It utilizes the function Dijkstra(A) 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, Dijkstra(A_pj−, q^{aj}_{pj}, S_d, S_c) determines the shortest path (saved in P_1) from q^{aj}_{pj} 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 = {p ∈ Q_p | q_p|_{\Pi_q} = (\pi_0, \sigma_n), \sigma_n ∈ Σ^0 \cup (\Sigma^0\{\sigma_d\})}. In Line 5, Dijkstra(Reverse(A_pj), q^{aj}_{pj}, S_d, \emptyset) is called to determine the shortest path (saved in P_2) from q^{aj}_{pj} to every state in S_d within the reversed A_pj and the associated distances (saved in C_2); Reverse(A_pj) is the directed graph obtained by inverting the direction of all edges in G(A_pj) while keeping the weights unchanged, where G(A_pj) is the directed graph associated with A_pj [2]. In Lines 7 and 8, for each state q^{aj}_{pr} ∈ S_d, the balanced cost of R^{aj}_{pj+} by (6) is computed. The one that yields the minimal cost is denoted by q^{aj*}_{pr}. At last, R^{aj*}_{pj+} is formed by concatenating the shortest path from q^{aj}_{pi} to q^{aj*}_{pr} and the reversed shortest path from q^{aj}_{pr} to q^{aj*}_{pj}. If q^{aj*}_{pr} returns empty, then agent a_i could not offer the requested collaboration thus b^{aj}_{ai} = ∅ and t^{aj}_{ai} = ∞, as in Line 13. The complexity of function Dijkstra(A) is O(|q_p|) · log{|Q_p|}). Reversing A_pj has the complexity linear to O(|q_p|). Note that when C_1(q^{aj*}_{pr}) < 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 A_pj that have the same cost by (6) as the one derived by Algorithm 3, i.e., the nonshortest path from q^{aj}_{pj} to S_d without the waiting action in the end.

Remark 4: Note that R^{aj*}_{pj+} from Algorithm 3 is the potentially revised run, i.e., agent a_j does not change its current plan but saves R^{aj}_{pj+} in 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^{aj}_{ai} = ∅ from Algorithm 3, action σ_d can be done at the estimated time t^{aj}_{ai} by agent a_j following R^{aj*}_{pj+}.

Proof: Since the first segment of R^{aj*}_{pj+} from q^{aj}_{pj} to q^{aj*}_{pr} is derived by Dijkstra(A) 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^{aj}_{ai}. □

D. Confirmation

Based on the replies from a_j ∈ N^{a_i}_1, agent a_i needs to acknowledge them by sending back confirmation messages

\[ \text{Confirm}^{a_i}_{aj} = \{(σ_d, c^{aj}_{ai}, f_m), \forall σ_d ∈ \text{Dep}^{a_i}(σ_m)\} \]  

where σ_d is the requested assisting action, c^{aj}_{ai} 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.

The choices of \{c^{aj*}_{ai}, a_j ∈ N^{a_i}_1\} should satisfy two constraints: 1) exactly one agent in N^{a_i}_1 can be the confirmed collaborator for each action σ_d ∈ Dep^{a_i}(σ_m) and 2) each agent in N^{a_i}_1 can be confirmed for at most one action in Dep^{a_i}(σ_m). Meanwhile, the estimated finishing time f_m should be as small as possible.

Let |N^{a_i}_1| = N_1 and |Dep^{a_i}(σ_m)| = N_2. Without the loss of generality, denote by N^{a_i}_1 = \{1, ..., N_1\} and Dep^{a_i}(σ_m) = \{σ_1, ..., σ_{N_2}\}. The problem of finding \{c^{aj*}_{ai}\} and f_m can be readily formulated as an integer programming problem [27]

\[
\min f_m \\
\text{s.t.} \quad f_m = \max_d \{c^{aj}_{ai} \cdot i^d_{ai}, T_m\} \\
\sum_{d=1}^{N_2} b^{aj}_{ai} \cdot c^{aj}_{ai} \leq 1 \quad \forall a_j \in \{1, ..., N_1\} \\
\sum_{g=1}^{N_1} b^{aj}_{ai} \cdot c^{aj}_{ai} = 1 \quad \forall d \in \{1, ..., N_2\} 
\]

where (σ_d, b^{aj}_{ai}, i^{aj}_{ai}) ∈ Reply^{a_i}_{ai} from (5). Any stand-alone integer programming solver can be used to obtain \{c^{aj*}_{ai}\} and f_m once (8) is formulated, e.g., “Gurobi” [14]. Then, \forall a_j ∈ N^{a_i}_1 and \forall σ_d ∈ Dep^{a_i}(σ_m), consider the two cases that are as follows.

1) If (8) has a solution, both \{c^{aj*}_{ai}\} and f_m exist. If c^{aj*}_{ai} is 1, add (σ_d, T, f_m) to \text{Confirm}^{a_i}_{aj}; otherwise, add (σ_d, ∞, ∞) to \text{Confirm}^{a_i}_{aj}.

2) If (8) has no solutions, add (σ_d, ∞, ∞) to \text{Confirm}^{a_i}_{aj}.

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 $\sigma_m$, with $|N^{\text{gi}}_a| \cdot |\text{Depd}^{\text{gi}}(\sigma_m)|$ Boolean variables.

E. Request to Coordinators

As mentioned in Section III-B, the neighboring set $N^{\text{gi}}_a$ 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 $a_{c\text{k}}$:

$$\text{CoReq}^{a_{c\text{k}}}_{ai} = \{(\sigma_d, \pi_h, T_m), \forall \sigma_d \in \text{Depd}^{\text{ai}}(\sigma_m)\}$$  \hspace{1cm} (9)

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

$$\text{CoReq}^{a_{c\prime k}}_{ai} = \{(\sigma_d, \pi_h, T_m), \forall \sigma_d \in \text{Depd}^{\text{ai}}(\sigma_m)\}$$  \hspace{1cm} (10)

where $a_{c\prime k}$ is the coordinator of group $g_k$. Upon receiving this coordination request, each coordinator $a_{c\prime k}$ then re-relays this request to each group member $a_j \in g_k$ and can wait for the coordination reply as follows:

$$\text{CoRep}^{a_{c\prime k}}_{ai} = \{\{\sigma_d, b_{d}^{a_{c\prime k}}, t_{d}^{a_{c\prime k}}\}, \forall (\sigma_d, \pi_h, T_m) \in \text{CoReq}^{a_{c\prime k}}_{ai}\}$$  \hspace{1cm} (11)

where $(\sigma_d, b_{d}^{a_{c\prime k}}, t_{d}^{a_{c\prime k}})$ is determined by Algorithm 3 as described in Section V-C. Based on these replies that for all $(\sigma_d, \pi_h, T_m) \in \text{CoReq}^{a_{c\prime k}}_{ai}$, the coordinator $a_{c\prime k}$ finds the group member $a_{jd}$ in $g_k$ satisfying that

$$a_{jd} = \arg\min_{a_{jd} \in g_k, b_{d}^{a_{c\prime k}} = \top} \{ T_m - t_{d}^{a_{c\prime k}} \}$$  \hspace{1cm} (12)

where agent $a_{jd}$ minimizes the time difference to the expected finish time $T_m$ of the assisting action $\sigma_d \in \text{Depd}^{\text{ai}}(\sigma_m)$. Then, the coordination reply from $a_{c\prime k}$ to $a_{c\text{k}}$ is given by

$$\text{CoRep}^{a_{c\text{k}}}_{ai} = \{\{\sigma_d, b_{d}^{a_{c\text{k}}}, t_{d}^{a_{c\text{k}}}\}, \forall (\sigma_d, \pi_h, T_m) \in \text{CoReq}^{a_{c\text{k}}}_{ai}\}$$

where note that if for an assisting action $\sigma_d$, it holds that $b_{d}^{a_{c\prime k}} = \bot$ holds, $\forall a_{jd} \in g_k$, we set $b_{d} = \bot$ and $t_{d} = \infty$.

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

$$\min \ f_m \hspace{1cm} \text{s.t.} \hspace{1cm} f_m = \max_{d} \{c_{d}^{a_{c\text{k}}}, t_{d}^{a_{c\text{k}}}, T_m\}$$

$$\sum_{d=1}^{N_2} b_{d}^{a_{c\prime k}} \cdot c_{d}^{a_{c\prime k}} \leq 1 \ \forall a_{jd} \in g_k \in \mathcal{K}$$

$$\sum_{g=1}^{K} b_{d}^{a_{c\prime k}} \cdot c_{d}^{a_{c\prime k}} = 1 \ \forall \sigma_d \in \text{Depd}^{\text{ai}}(\sigma_m)$$  \hspace{1cm} (13)

where $N_2 = |\text{Depd}^{\text{ai}}(\sigma_m)|$ and $\text{CoRep}^{a_{c\text{k}}}_{ai} = \{\{\sigma_d, b_{d}^{a_{c\text{k}}}, t_{d}^{a_{c\text{k}}}, \forall (\sigma_d, \pi_h, T_m) \in \text{CoReq}^{a_{c\text{k}}}_{ai}\}$. 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 $\sigma_m$, with $|\mathcal{K}| \cdot |\text{Depd}^{\text{ai}}(\sigma_m)|$ 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 $\sigma_m$ can be fulfilled by the local neighbors. Else, if (8) has no solution but (13) has a solution, it means that $\sigma_m$ can be fulfilled by agents outside its neighbor network. Then, $R^{\text{ai}}_{p\text{.j}}$ remains unchanged. $\overline{T^{\text{f}}}^{\text{ai}}$ 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 $\sigma_m$ cannot be done as planned in $R^{\text{ai}}_{p\text{.j}}$. Thus, agent $a_i$ needs to revise its plan by delaying this collaborative action $\sigma_m$.

Algorithm 4 revises $R^{\text{ai}}_{p\text{.j}}$ and delays $\sigma_m$ by time $D^{\text{ai}}$, where $D^{\text{ai}} > 0$ is a design parameter. Function $\overline{\text{DimjksTAT}}(\cdot)$ from Algorithm 3 is used to find a path from $q_i^{\text{ai}}_{p\text{.j}}$ to one state in $S_d$ whose cost is larger than $T_m + D^{\text{ai}}$ and the accepting state $q_i^{\text{ai}}_{p\text{.j}}$ is reachable from this state. Such a path can always be found as the action $a_0$ that takes time $T_0$ can be repeated as many times as needed.

On the other hand, upon receiving $\text{Confirm}^{a_{ij}}_{ai}$, each agent $a_j \in N^{\text{gi}}_a$ checks the following.

1) If $c_{d}^{a_{ij}} = \top$ for some $\sigma_d \in \text{Depd}^{\text{ai}}(\sigma_m)$, then, agent $a_j$ is confirmed to offer the assisting action $\sigma_d$. As a result, it modifies its plan based on the potential set of plans $\overline{P}$ from Algorithm 3. In particular, the plan segment $\overline{R}^{\text{ai}}_{p\text{.j}}$ is set to $\overline{R}^{\text{ai}}_{p\text{.j}} + \overline{T}^{\text{ai}}_{p\text{.j}}$ is set to $f_m$.

2) If $c_{d}^{a_{ij}} = \bot, \forall \sigma_d \in \text{Depd}^{\text{ai}}(\sigma_m)$, it means $g$ is not confirmed as a collaborator. Then, $R^{\text{ai}}_{p\text{.j}}$ remains unchanged and $\overline{T}^{\text{ai}}_{p\text{.j}}$ is set to $0$.

Afterward, agent $a_i$ and all confirmed collaborators in $N^{\text{gi}}_a$ 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 $\sigma_m$ is done.

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

Proof: Since each assisting action $\sigma_d \in \text{Depd}^{\text{ai}}(\sigma_m)$ has been assigned exactly to one agent $a_j \in N^{\text{gi}}_a$, then, $\sigma_d$ can be accomplished by agent $a_j$ at time $t_{d}^{a_{ij}}$ by Lemma 1. Thus, $\sigma_m$ 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: $R^m_{p_1}, q_{p_1}, \langle \pi_h, \sigma_d \rangle, A^m_p, \Sigma_c$

Output: $\tilde{R}^i_{p_1}$

1. compute $S_d$ given $\langle \pi_h, \sigma_d \rangle$, $S_c$
2. $q_{p_1} = R^m_{p_1}[1]$, $q_{p_1} = R^m_{p_1}[-1]$
3. $(P_1, C_1) = \text{DiJk}eTA(A^m_p, q_{p_1}, S_d, S_c)$
4. $(P_2, C_2) = \text{DiJk}eTA(\text{Reverse}(A^m_p), q_{p_1}, S_d, 0)$
5. forall the $q_{p,d}^i \in S_d$ do
6. if $C_1(q_{p,d}^i) > T_m + D^m$ and $P_2(q_{p,d}^i)$ exists then
7. $\tilde{R}^i_{p_1} = P_1(q_{p,d}^i) + \text{Reverse}(P_2(q_{p,d}^i))$
8. Return $\tilde{R}^i_{p_1}$

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 $q_{a_i}^m \equiv T$. 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 $T > 0$ such that for each agent $a_i \in N$ and any collaborative action $\sigma_m$ requested by agent $a_i$, initially at time $t_m > 0$, problems (8) or (13) for $\sigma_m$ will have a solution within time $t_m + T$.

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 $\sigma_1$ 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 $\sigma_2$ 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 $g_k \in K$ and their initial plans are given by $\tau_{a_i}^{\text{init}}$ and $\tau_{a_j}^{\text{init}}$ from Section IV.

A. Formulation

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

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

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

$$
\tau_{G,i}^{a_i} = \tau_{G,i}^{a_i}[-m:(l_1 - 1)]
\oplus \tau_{G,j}^{a_j,-}[g_1:g_2] \oplus \tau_{G,j}^{a_i,-}[(l_2 + 1):]
$$

and the new future plan of agent $a_j$ is given by

$$
\tau_{G,j}^{a_j} = \tau_{G,j}^{a_j}[-n:(g_1 - 1)]
\oplus \tau_{G,j}^{a_i,-}[l_1:l_2] \oplus \tau_{G,j}^{a_i,-}[(g_2 + 1):]
$$

where $\oplus$ is the concatenation operation of two sequences. This swapping is allowed only if two conditions hold: $(\tau_{G,j}^{a_i,-}[l_1 - 1], \tau_{G,j}^{a_j}[-g_1])$, $(\tau_{G,j}^{a_j}[-g_2], \tau_{G,j}^{a_i}[-l_1 + 1])$ $\rightarrow a_i$ and $(\tau_{G,j}^{a_i}[-g_1 - 1], \tau_{G,j}^{a_j}[-l_1], \tau_{G,j}^{a_i}[-g_2 - 1], \tau_{G,j}^{a_j}[-l_2 + 1])$ $\rightarrow a_j$.

Namely, only the chosen segments of $\tau_{G,i}^{a_i}$ and $\tau_{G,j}^{a_j}$ 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

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

1) Agent $a_j$ can start executing $r_{G_i}^{a_j^+}[l_1:l_2]$ only after agent $ai$ has finished execution $r_{G_i}^{a_j^+}[m;l_1 - 1]$.

2) Agent $a_j$ can start executing $r_{G_i}^{a_j^+}[g_2 + 1:]$ only after agent $ai$ has finished execution $r_{G_i}^{a_j^+}[g_1;g_2]$.

3) Another two analogous conditions for agent $ai$.

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

Problem 4: Find the segment $r_{G_i}^{a_i^+}[l_1;l_2]$ of $r_{G_i}^{a_i}$ and the segment $r_{G_i}^{a_j^+}[g_1;g_2]$ of $r_{G_i}^{a_j}$, such that after the swapping, $\Delta_{ai,aj} = \Theta_{ai,aj} + \Theta_{aj,ai}$ becomes maximal among all possible choices, while $\Delta_{ai,aj} > 0$ and $\Theta_{ai,aj} < \Theta_{ai,aj}$ hold, where $\Delta$ and $\Theta$ are design parameters.

Note that $\Delta$ in the above formulation serves as the minimal reduction of the total cost to facilitate a swapping, while $\Theta$ is the maximal total waiting time that is allowed. Since agents $ai$ and $aj$ are homogeneous, the cost of agent $ai$ executing the segment $r_{G_i}^{a_j}$ and $r_{G_i}^{a_j^+}[g_1;g_2]$ is the same as agent $aj$. The same arguments hold for agent $aj$ executing $r_{G_i}^{a_i^+}[l_1;l_2]$. 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 $ai$ before and after the swapping as:

$\Delta_{ai} = \Delta_{ai}^1 + \text{cost}(r_{G_i}^{a_i^+}[g_1;g_2]) - \text{cost}(r_{G_i}^{a_i^+}[l_1;l_2]) + \Delta_{ai}^2$, where $\Delta_{ai}^1 = T_{G_i}^a(r_{G_i}^{a_i^+}[g_1;g_2]) - T_{G_i}^a(r_{G_i}^{a_i^+}[l_1;l_2])$ and $\Delta_{ai}^2 = T_{G_i}^a(r_{G_i}^{a_j}[g_1;g_2]), r_{G_i}^{a_i^+}[l_1;l_2]) - T_{G_i}^a(r_{G_i}^{a_j^+}[g_1;g_2]), r_{G_i}^{a_j^+}[l_1;l_2]))$. Analogously for agent $aj$.

B. Plan Swapping and Synchronization

After the segments $r_{G_i}^{a_i^+}[l_1;l_2]$ and $r_{G_i}^{a_j^+}[g_1;g_2]$ are derived as the solution to Problem 4, agents $ai$ and $aj$ should exchange the segments and generate the updated plan, as shown in (14) and (15). While executing the updated plan, it is crucial that agents $ai$ and $aj$ synchronize with each other when they start and finish executing the swapped segments. As discussed earlier, when agent $ai$ starts executing $r_{G_i}^{a_i^+}[l_1;l_2]$ only after it synchronizes with agent $aj$ that agent $aj$ has finished executing $r_{G_i}^{a_j^+}[l_1;l_2]$. Analogous arguments hold for agent $aj$.

Note that $\Theta$ and $\Delta$ 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 $\phi_{ai}^a$ can be accomplished in finite time, $\forall a_i \in \mathcal{N}$. 

\[ \Delta_{ai}^1 + \Delta_{aj}^2, \]
Proof: Starting from the initial plan $\tau_{G,\text{init}}^a$ for agent $a_k \in N$, motion and local actions in $\tau_{G,\text{init}}^a$ can be accomplished locally by an agent itself. If $\tau_{G,\text{init}}^a$ remains unchanged for agent $a_k$, we only need to show that the collaborative actions in $\tau_{G,\text{init}}^a$ can be accomplished. The fact that $\tau_{G,\text{init}}^a$ remains unchanged indicates that agent $a_k$’s requests for each collaborative action $\sigma_m$ are fulfilled, i.e., (8) or (13) for $\sigma_m$ has a solution. By Theorem 2, action $\sigma_m$ can be accomplished in finite time $f_m$. Since $\tau_{G,\text{init}}^a$ is a finite sequence, $\varphi^{a_i}$ can be satisfied in finite time as every motion and action inside can be done in finite time. On the other hand, if $\tau_{G,\text{init}}^a$ has to be adapted in real time, the reasons are: 1) agent $a_k$ is confirmed to assist its neighbor $g$ on one collaboration and 2) agent $a_k$ has made a request for a collaborative action $\sigma_m$ 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 $\hat{R}_{p,i}^{a_i}$, still satisfies $\varphi^{a_i}$ in finite time. For case 2), by Assumption 3, (8) or (13) will have a feasible solution within at most time $T$, meaning that $\sigma_m$ 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 discretized 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 $R_1$, $R_2$, ..., $R_6$. 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 hearCircle and hearCircle2 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 $o_k^l$, $\forall l = 1,2,3$ and $\forall k = 2,3,4$. Each of them is capable of providing the actions pick2 and drop2 for each object $l = 1, 2, 3$, e.g., agent $a_{2,1}$ can pickup object $o_2^1$ with action pick2 and drop it with action drop2. The dependence between the actions is clarified as follows:

1) To pickup and drop $o_1^1$ can be done by agent $a_{k,i} \in g_k$ itself, $\forall a_{k,i} \in g_k$ and $g_k = 2,3,4$. 2) To pickup and drop $o_2^2$ by action pick2 and drop2, agent $a_{k,i}$ needs assisting actions hpick3 and hdrop3, respectively, from any other UAV $a_j \in g_l$ or UGV $a_j \in g_k$ within the same group, $g_k = 2,3,4$. 3) To pickup $o_3^3$ by action pick3, agent $a_{k,i}$ needs assisting action hpick3 from any agent $a_j \in g_l$ and action hdrop3 from another agent $a_j \in g_l$. Similarly, to drop $o_4^4$ by action drop3, agent $a_{k,i}$ needs assisting action hdrop3 from any agent $a_j \in g_l$ and action hdrop3 from another agent $a_j \in g_l$, 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, $\phi_{1,1} = (\neg(R_1 \land \text{record}_1)) \land (\neg(R_2 \land \text{record}_1)) \land (\neg(R_3 \land \text{circle}_1))$. 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, $\phi_{2,1} = (\neg(pick2_1 \land \neg(R_2 \land \text{drop}_1))) \land (\neg(pick2_1 \land \neg(R_4 \land \text{drop}_1))) \land (\neg(pick3_1 \land \neg(R_6 \land \text{drop}_1))) \land (\neg(pick4_1 \land \neg(R_6 \land \text{drop}_1)))$ and $\phi_{2,2} = (\neg(pick3_2 \land \neg(R_3 \land \text{drop}_2))) \land (\neg(pick3_2 \land \neg(R_3 \land \text{drop}_2))) \land (\neg(pick4_2 \land \neg(R_3 \land \text{drop}_2))) \land (\neg(pick4_2 \land \neg(R_3 \land \text{drop}_2)))$. 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 $\phi_{2,1}$ consists of 8 states and 27 edges.
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 "$r_{1}^a_1 = R_1$record$_1 R_2$record$_1 R_3$circle$_1$"; while for UGV $a_{2,1}$, its initial plan is given by "$r_{2}^u_2 = S_1$pick$_2 R_2$drop$_2 S_3$pick$_3 R_6$drop$_3 S_2$pick$_4 R_4$drop$_4$." 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 actions performed during the simulation. 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 $1600^{20} \times 8^5 \times 46^{15}$ states (approximately $3.5 \times 10^{93}$). 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 109 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


