Bottom-up Motion and Task Coordination for Loosely-coupled Multi-agent Systems with Dependent Local Tasks

Meng Guo and Dimos V. Dimarogonas

***Abstract*— We propose a bottom-up motion and task coordination scheme for loosely-coupled multi-agent systems under dependent local tasks. Instead of defining a global task for the whole team, each agent is assigned locally a task as syntactically co-safe linear temporal logic formulas that specify both motion and action requirements. Inter-agent dependency is introduced by collaborative actions of which the execution requires multiple agents' collaboration. The proposed solution contains an off-line initial plan synthesis, an on-line request-reply messages exchange and a real-time plan adaptation algorithm. It is distributed in that any decision is made locally based on local computation and local communication within neighboring agents. It is scalable and resilient to agent failures as the dependency is formed and removed dynamically based on the plan execution status and agent capabilities, instead of pre-assigned agent identities. The overall scheme is demonstrated by a simulated scenario.**

I. Introduction

Temporal-logic-based motion and task planning has gained significant attention in recent years, as it provides an automated controller synthesis approach for autonomous robots. Temporal logics such as Linear Temporal Logic (LTL) and Computation Tree Logic (CTL) provide formal high level languages that can describe planning objectives more complex than the well-studied point-to-point navigation problem [16]. The task specification is given as a temporal logic formula with respect to a discretized abstraction of the robot motion [1], [3]. Then 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 [5], [6], [15]. Similar methodology has also been applied for multi-agent systems [4], [12], [22]. 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 [4] or partially-synchronized [13] 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.

In contrast, we assume that there is no pre-specified global task and individual task specifications are assigned locally to each agent as LTL formulas [10], [11], [20], which favors a *bottom-up* formulation. These local tasks can be independent [11] or dependent where one agent may need others' collaboration to fulfill its own task. In the latter case, coordination is thus crucial for the accomplishment of all local tasks. This way of formulation instead is particularly useful for multi-agent systems where the number of agents are large, the agents are heterogeneous and each agent has a clear task assignment.

The greatest challenge of task coordination for multi-agent systems under dependent *local* tasks is the computational complexity. A centralized solution requires the direct composition of all agents' model to represent all possible behaviors. This problem is addressed in [10] by grouping the agents into dependency clusters such that composition is only needed for each cluster, while [20] proposes a receding horizon approach that decomposes the synthesis problem into shorter horizon planning problems that are solved iteratively. In general, the derived plan needs to be executed in a synchronized fashion by all agents, which limits the flexibility and robustness of the overall system.

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 completely the composition of different agents' models or tasks, which is replaced by an on-line 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 are formed and removed dynamically. We show that the proposed scheme guarantees the satisfaction of all local tasks under the assumption that they are loosely-coupled. It can also potentially detect and recover from agent failures.

The rest of the paper is organized as follows: Sec. II introduces some preliminaries. The problem is stated formally in Sec. III. Sec. IV presents the initial plan synthesis strategy. The on-line coordination scheme is described in Sec. V. The overall structure is discussed in Sec. VI. A case study is presented in Sec. VII and we conclude in Sec. VIII.

II. Preliminaries

A. Syntactically Co-safe 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*), U

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. mengg, dimos@kth.se. This work was supported by the Swedish Research Council (VR) and EU STREP RECONFIG: FP7-ICT-2011-9-600825.

(*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 Chapter 5 of [2]. One particular class of LTL we consider in this paper is the syntactically co-safe LTL (sc-LTL) [14]. They only contain the $\bigcirc$, U and $\Diamond$ operators and are written in positive normal form. In contrast, the satisfaction of an sc-LTL formula can be achieved in a 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 alternatively be captured through a Nondeterministic Büchi automaton (NBA) $\mathcal{A}_\varphi$ [2], which is 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; $Q_0, \mathcal{F} \subseteq Q$ are the initial and accepting states. There are fast translation tools [8] to obtain $\mathcal{A}_\varphi$ given φ.

III. Problem Formulation

We consider K autonomous agents with heterogeneous capabilities within a fully-known workspace. Each agent with the identity $k \in \mathcal{K} = \{1, 2, \cdots, K\}$ is capable of navigating within the workspace and performing various actions.

A. Motion Abstraction

The workspace consists of N partitions as the regions of interest, denoted by $\Pi = \{\pi_1, \pi_2, \cdots, \pi_N\}$. 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, see [3], [4], [17]. Besides, there is a set of atomic propositions describing the properties of the workspace, denoted by $\Psi^k_{\mathcal{M}}$. Similar to [9], agent k's motion within the workspace is modeled as a finite transition system (FTS):

$$\mathcal{M}^k \triangleq (\Pi, \longrightarrow^k_{\mathcal{M}}, \Pi^k_{\mathcal{M},0}, \Psi^k_{\mathcal{M}}, L^k_{\mathcal{M}}, T^k_{\mathcal{M}}), \qquad (1)$$

where $\longrightarrow^k_{\mathcal{M}} \subseteq \Pi \times \Pi$ is the transition relation; $\Pi^k_{\mathcal{M},0} \in \Pi$ is the initial region agent k starts from; $L^k_{\mathcal{M}} : \Pi \to 2^{\Psi^k_{\mathcal{M}}}$ is the labeling function, indicating the properties held by each region; $T^k_{\mathcal{M}} : \longrightarrow^k_{\mathcal{M}} \to \mathbb{R}^+$ estimates the time each transition takes. A path of $\mathcal{M}^k$ is a sequence of regions $\pi_0\pi_1\cdots\pi_N$, where $(\pi_n, \pi_{n+1}) \in \longrightarrow^k_{\mathcal{M}}$, $\forall n = 0, 1, \cdots, N-1$. Note that $\mathcal{M}^k$ might be different between the agents due to heterogeneity. The workspace model from Sec. VII is shown in Fig. 1. Moreover, each agent k has a set of neighboring agents, denoted by $\mathcal{K}^k \subseteq \mathcal{K}$. Agent k can exchange messages directly with any agent $g \in \mathcal{K}^k$.

B. Action Model

Besides the motion ability, agent k is capable of performing a set of actions denoted by $\Sigma^k \triangleq \Sigma^k_l \cup \Sigma^k_c \cup \Sigma^k_h$, where

- Σ^k_l is a set of *local* actions, which can be done by agent k itself;
- Σ^k_c is a set of *collaborative* actions, which can be done by agent k but requires collaborations from other agents;
- Σ^k_h is a set of *assisting* actions, which agent k offers to other agents to accomplish their collaborative actions.

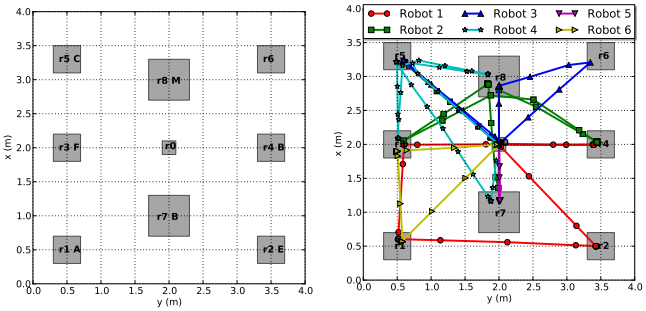

Fig. 1: Left: the regions are in gray and labeled by objects of interest inside; Right: the final trajectories of all agents after accomplishing their local tasks in Sec. VII-C.1.

In other words, Σ^k_l and Σ^k_c contain actions that can be *initiated* by agent k, denoted by $\Sigma^k_a = \Sigma^k_l \cup \Sigma^k_c$, while Σ^k_h contains assisting actions only to *assist* other agents. By default, $\sigma_0 = \texttt{None} \in \Sigma^k_l$ means that none of the actions is performed. Moreover, denote by $\Sigma^{\sim k}_h$ the set of *external* assisting actions agent k depends on, which can only be provided by some of its neighbors in $\mathcal{K}^k$, i.e., $\Sigma^{\sim k}_h \subseteq \cup_{i\in\mathcal{K}^k}\Sigma^i_h$. Table I shows the action sets of the agents that will be simulated in Sec. VII. The action model of agent k is modeled by a six-tuple:

$$\mathcal{A}^k \triangleq (\Sigma^k, \Psi^k_\Sigma, L^k_\Sigma, \texttt{Cond}^k, \texttt{Dura}^k, \texttt{Depd}^k), \qquad (2)$$

where Σ^k is the set of actions defined earlier; Ψ^k_Σ is a set of atomic propositions related to the agent's *active* actions; $L^k_\Sigma : \Sigma^k \to 2^{\Psi^k_\Sigma}$ is the labeling function. $L^k_\Sigma(\sigma_h) = \emptyset$, $\forall\sigma_h \in \Sigma^k_h$ and $L^k_\Sigma(\sigma_a) \subseteq \Psi^k_\Sigma$, $\forall\sigma_a \in \Sigma^k_a$; $\texttt{Cond}^k : \Sigma^k \times 2^{\Psi^k_{\mathcal{M}}} \to \top/\bot$ indicates the set of region properties that have to be fulfilled in order to perform an action; $\texttt{Dura}^k : \Sigma^k \to \mathbb{R}^+$ is the estimated time duration of each action. $\texttt{Dura}^k(\sigma_0) = T_0 > 0$ is a design parameter; $\texttt{Depd}^k : \Sigma^k \to 2^{\Sigma^{\sim k}_h}$ is the *dependence* function. $\texttt{Depd}^k(\sigma_s) = \emptyset$, $\forall\sigma_s \in \Sigma^k_l \cup \Sigma^k_h$ and $\texttt{Depd}^k(\sigma_c) \subseteq \Sigma^{\sim k}_h$, $\forall\sigma_c \in \Sigma^k_c$. Namely each collaborative action depends on a set of assisting actions from its neighbors. This is useful for defining complex collaborations involving multiple agents.

Remark 1: Compared with defining dependency 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.

Definition 1: A local or assisting action $\sigma_s \in \Sigma^k_l \cup \Sigma^k_h$ is said to be *done* at region $\pi_i \in \Pi$ if two conditions hold: (i) $\texttt{Cond}^k(\sigma_s, L^k_{\mathcal{M}}(\pi_i)) = \top$; (ii) σ_s is activated for period $\texttt{Dura}^k(\sigma_s)$. For a collaborative action $\sigma_c \in \Sigma^k_c$, another condition is needed: (iii) all assisting actions in $\texttt{Depd}^k(\sigma_c)$ are done by other agents at the same region π_i. ■

Remark 2: Different from [9], the action model by (2) can model both local and collaborative actions.

C. Complete Agent Model

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

Agent	Σ_l^k	Σ_c^k	Σ_h^k	$\Sigma_h^{\sim k}$
$\mathfrak{R}_1$	l_A, u_A	l_B, u_B	$\emptyset$	h_B
$\mathfrak{R}_2$	s	$\emptyset$	h_B, h_{C_1}, h_F	$\emptyset$
$\mathfrak{R}_3$	$\emptyset$	o_M	h_{C_2}, h_F	h_M
$\mathfrak{R}_4$	s	a_C	h_M, h_F	h_{C_1}, h_{C_2}
$\mathfrak{R}_5$	m_D	$\emptyset$	h_B, h_{C_1}, h_{C_2}	$\emptyset$
$\mathfrak{R}_6$	o_E	c_F	h_B, h_M	h_F

TABLE I: Action sets (besides σ_0) described in Sec. VII.

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

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

D. Task Specification

The local task of agent k, denoted by φ^k, is given as an sc-safe LTL formula over the set of atomic propositions $\Psi_{\mathcal{G}}^k$ from Def. 2. Thus φ^k 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^k of $\mathcal{G}^k$, then τ^k fulfils φ^k if $\texttt{trace}(\tau^k) \models \varphi^k$ where the satisfaction relation is defined in Sec. II-A. One special case is that when $\varphi^k \triangleq \top$, agent k does not have a local task and serves as an assisting agent. In summary, we consider the following problem:

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

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 Sect. V.

A. Plan as Motion and Action Sequence

We intend to find a finite path of $\mathcal{G}^k$ whose trace satisfies the co-safe formula φ^k as described in Sec. III-D. We rely on the automaton-based model-checking approach (see Alg. 11 in [2]) by checking the emptiness of the product automaton. Let $\mathcal{A}_{\varphi^k}$ be the NBA associated with φ^k from Sec. II-A. The product automaton $\mathcal{A}_p^k$ is defined as follows:

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

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

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

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

Denote by $R_{p,\text{init}}^k$ the solution. Algorithm 1 in [9] solves the above problem, which is omitted here due to limited space. It utilizes Dijkstra's algorithm [16] for computing the shortest path from any initial state in $Q_{p,0}^k$ to every reachable accepting state in $\mathcal{F}_p^k$ and checks if there is cycle back to $q_{p,N}$. The worst-case complexity is $\mathcal{O}(|\delta_p^k| \cdot \log |Q_p^k| \cdot |Q_{p,0}^k|)$. By projecting $R_{p,\text{init}}^k$ onto $\Pi_{\mathcal{G}}^k$, it gives the initial motion and action plan $\tau_{\mathcal{G},\text{init}}^k = R_{p,\text{init}}^k|_{\Pi_{\mathcal{G}}^k}$ that fulfils φ^k.

Remark 3: The initial plans are synthesized locally instead of by a central unit [4] or within a cluster [10].

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

V. Distributed Collaborative Task Coordination

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

A. Planned Motion and Actions in Horizon

Denote by $\pi_{\mathcal{G},t}^k \in \Pi_{\mathcal{G}}^k$ the state of agent k at time t. After the system starts, assume $\pi_{\mathcal{G},t}^k$ is the i_{th} element in

Algorithm 1: Plan in horizon and request, `Request()`

Input: $\tau^k_{\mathcal{G},\text{init}}$, $\pi^k_{\mathcal{G},t}$, H^k
Output: $\tau^k_{\mathcal{G},H}$, $\textbf{Request}^k$

1 $\tau^k_{\mathcal{G},\text{init}}[i] = \pi^k_{\mathcal{G},t}$, $s = 0$, $T_m = 0$, $\textbf{Request}^k = \emptyset$
2 **while** $T < H^k$ *and* $i + s \leq |\tau^k_{\mathcal{G},init}|$ **do**
3 $s = s + 1$
4 $T_m = T_m + T^k_{\mathcal{G},\text{init}}\left(\tau^k_{\mathcal{G},\text{init}}[i], \tau^k_{\mathcal{G},\text{init}}[i+s]\right)$
5 $\langle \pi_i, \sigma_m \rangle = \tau^k_{\mathcal{G},\text{init}}[i+s]$
6 **if** $\textbf{Request}^k = \emptyset$ *and* $\sigma_m \in \Sigma^k_c$ **then**
7 **forall the** $\sigma_d \in \texttt{Depd}^k(\sigma_m)$ **do**
8 add (σ_d, π_i, T_m) to $\textbf{Request}^k$
9 $j = i + s$, $\tau^k_{\mathcal{G},H} = \tau^k_{\mathcal{G},\text{init}}[i:j]$
10 **return** $\tau^k_{\mathcal{G},H}$, $\textbf{Request}^k$

Algorithm 2: Reply to request by agent g, `Reply()`

Input: $\textbf{Request}^k_g$, $\widehat{R}^g_{p,-}$, $\mathcal{A}^g_p$, $\overline{T}^g$
Output: $\textbf{Reply}^g_k$, $\widehat{P}$

1 **forall the** $(\sigma_d, \pi_i, T_m) \in \textbf{Request}^k_g$ **do**
2 **if** $\overline{T}^g$ *is* 0 **then**
3 $\left(\widehat{R}^g_{p,+}, b^g_d, t^g_d\right) =$ $\texttt{EvalReq}(\widehat{R}^g_{p,-}, (\pi_i, \sigma_d, T_m), \mathcal{A}^g_p)$
4 **if** b^g_d *is* $\top$ **then**
5 $\widehat{P}(\sigma_d) = \widehat{R}^g_{p,+}$
6 add (σ_d, b^g_d, t^g_d) to $\textbf{Reply}^g_k$
7 add $(\sigma_d, \perp, 0)$ to $\textbf{Reply}^g_k$
8 **Return** $\textbf{Reply}^g_k$, $\widehat{P}$

$\tau^k_{\mathcal{G},\text{init}}$, namely, $\pi^k_{\mathcal{G},t} = \tau^k_{\mathcal{G},\text{init}}[i]$. Each agent $k \in \mathcal{K}$ is given a bounded planning horizon $0 < H^k < \infty$, which is the time ahead agent k checks its plan. Similar approach can be found in [18] for a single dynamic system. Then the sequence of states agent k is expected to reach within the time H^k, denoted by $\tau^k_{\mathcal{G},H}$, is the segment $\tau^k_{\mathcal{G},H} = \tau^k_{\mathcal{G},\text{init}}[i:j]$, where the index $j \geq i$ is the solution to this optimization problem: $\min\ j$, subject to $\sum_{s=i}^{j} T^k_{\mathcal{G}}\left(\tau^k_{\mathcal{G},\text{init}}[s], \tau^k_{\mathcal{G},\text{init}}[s+1]\right) \geq H^k$. It can be solved by iterating through the sequence of $\tau^k_{\mathcal{G},\text{init}}$ and computing the accumulated cost, which is then compared with H^k. If it does not have a solution, it means H^k is larger than the total cost of the rest of the plan $\tau^k_{\mathcal{G},\text{init}}[i:]$, then $j = |\tau^k_{\mathcal{G},\text{init}}|$, see Lines 1-5, 9-10 of Alg. 1. The time horizon avoids coordinating on collaborative actions that will be done within a long time from now.

B. Request to Neighbours

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

$$\textbf{Request}^k = \{(\sigma_d, \pi_i, T_m), \forall \sigma_d \in \texttt{Depd}^k(\sigma_m)\}, \quad (4)$$

where $\texttt{Depd}^k(\sigma_m)$ is the set of external assisting actions that σ_m depends on by (2); $\pi_i \in \Pi$ is the region where σ_m will be performed; $T_m \geq 0$ is the estimated time when σ_m will be performed from now. Assume that $\langle \pi_i, \sigma_m \rangle$ is the l_{th} element of $\tau^k_{\mathcal{G},H}$. Then $T_m = \sum_{s=1}^{l} T^k_{\mathcal{G}}\left(\tau^k_{\mathcal{G},H}[s], \tau^k_{\mathcal{G},H}[s+1]\right)$, see Lines 4-9 of Alg. 1. Each element $(\sigma_d, \pi_i, T_m) \in \textbf{Request}^k$ contains the message that "agent k is requesting the assisting action σ_d at region π_i in the estimated time T_m from now". The request message from agent k to each agent $g \in \mathcal{K}^k$, denoted by $\textbf{Request}^k_g$, is the same as $\textbf{Request}^k$, i.e., $\textbf{Request}^k_g = \textbf{Request}^k$, $\forall g \in \mathcal{K}^k$.

Remark 4: Note that the request message is sent only for the first collaborative action in $\tau^k_{\mathcal{G},H}$ within the time horizon H^k (see Line 6 of Alg. 1), as the outcome of this request would greatly affect the second collaborative action in $\tau^k_{\mathcal{G},H}$.

C. Request Evaluation and Reply

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

$$\textbf{Reply}^g_k = \{(\sigma_d, b^g_d, t^g_d), \forall (\sigma_d, \pi_i, T_m) \in \textbf{Request}^k_g\}, \quad (5)$$

where σ_d is the requested assisting action by agent k; b^g_d is a boolean variable indicating the feasibility of agent g offering action σ_d at region π_i; $t^g_d \geq 0$ is estimated time when that can happen. We describe how to determine b^g_d and t^g_d below.

Denote by $\overline{T}^g \geq 0$ the estimated finishing time of the current collaboration agent k is engaged in. It is initialized as 0 and updated in Sec. V-E. As shown in Alg. 2, (I) if $\overline{T}^g > 0$, it means agent g is engaged in a collaboration. Then $\textbf{Reply}^g_k = \{(\sigma_d, \perp, 0), \forall (\sigma_d, \pi_i, T_m) \in \textbf{Request}^k_g\}$, meaning that agent g would *reject* any request before its current collaboration is finished. (II) If $\overline{T}^g = 0$, it means agent g is available to offer assisting actions. Then for each request $(\sigma_d, \pi_i, T_m) \in \textbf{Request}^k_g$, agent k needs to evaluate it in terms of feasibility and cost to determine b^g_d and t^g_d.

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

Algorithm 3: Evaluate the request, `EvalReq()`

Input: $\widehat{R}^g_{p,-}$, (π_i, σ_d, T_m), $q^g_{p,t}$, $\mathcal{A}^g_p$
Output: $\widehat{R}^g_{p,+}$, b^g_d, t^g_d

1 $q^g_{p,t} = \widehat{R}^g_{p,-}[1]$, $q^g_{p,f} = \widehat{R}^g_{p,-}[\text{-}1]$, $\pi_j = q^g_{p,t}|_{\Pi^g_{\mathcal{M}}}$
2 Compute S_d, S_c
3 $\bar{c} = \texttt{Cost}(\widehat{R}^g_{p,-}, \mathcal{A}^g_p)$
4 (P_1, C_1) = $\texttt{DijksTA}\big(\mathcal{A}^g_p, q^g_{p,t}, S_d, S_c\big)$
5 (P_2, C_2) = $\texttt{DijksTA}\big(\texttt{Reverse}(\mathcal{A}^g_p), q^g_{p,f}, S_d, \emptyset\big)$
6 **forall the** $q^g_{p,r} \in S_d$ **do**
7 **if** $P1(q^g_{p,r})$ *and* $P2(q_{p,r})$ *exist* **then**
8 $C_3(q^g_{p,r}) = |C_1(q^g_{p,r}) - T_m| + \alpha^g\big(C_1(q^g_{p,r}) + C_2(q^g_{p,r}) - \bar{c}\big)$
9 Find the $q^{g,\star}_{p,r} \in S_d$ that minimizes $C_3(q^g_{p,r})$
10 **if** $q^{g,\star}_{p,r} \neq \emptyset$ **then**
11 $P = P_1(q^{g,\star}_{p,r}) + \texttt{Reverse}\big(P_2(q^{g,\star}_{p,r})\big)$
12 **Return** $\widehat{R}^g_{p,+} = P$, $b^g_d = \top$, $t^g_d = C_1(q^{g,\star}_{p,r})$
13 **Return** $\widehat{R}^g_{p,+} = \emptyset$, $b^g_d = \bot$, $t^g_d = 0$

Firstly, the set of product states in $\mathcal{A}^g_p$ corresponding to $\langle \pi_i, \sigma_d \rangle$ is given by $S_d = \{q_p \in Q^g_p \,|\, q_p|_{\Pi^g_{\mathcal{G}}} = \langle \pi_i, \sigma_d \rangle\}$. Consider $\widehat{R}^g_{p,+}$ with the following structure:

$$\widehat{R}^g_{p,+} = q^g_{p,t} \cdots q^g_{p,r} \cdots q^g_{p,f}, \tag{6}$$

where $q^g_{p,r} \in S_d$, meaning that it passes through at least one state within S_d. Thus the corresponding plan would contain $\langle \pi_i, \sigma_d \rangle$, which fulfils the condition (i) above. Regarding conditions (ii) and (iii), we define the *balanced* cost of $\widehat{R}^g_{p,+}$:

$$\begin{aligned} & \texttt{BalCost}(\widehat{R}^g_{p,+}, T_m, \mathcal{A}^k_p) \\ = \; & |\sum_{s=1}^{r-t} W^g_p\big(\widehat{R}^g_{p,+}[s], \widehat{R}^g_{p,+}[s+1]\big) - T_m| \\ & + \alpha^g \left(\texttt{Cost}(\widehat{R}^g_{p,+}, \mathcal{A}^k_p) - \texttt{Cost}(\widehat{R}^g_{p,-}, \mathcal{A}^k_p)\right), \end{aligned} \tag{7}$$

where the first part stands for the estimated time gap between the requested time T_m by agent k and the actual time based on $\widehat{R}^g_{p,+}$ (for condition (ii)); the second term is the additional cost of $\widehat{R}^g_{p,+}$, compared to $\widehat{R}^g_{p,-}$ (for condition (iii)); $\alpha^g > 0$ is a design parameter as the relative weighting.

Problem 3: Given $\widehat{R}^g_{p,-}$, S_d and $\mathcal{A}^g_p$, find the path segement $\widehat{R}^g_{p,+}$ that minimizes (7).

Alg. 3 solves the above problem by the *bidirectional* Dijkstra algorithm [19]. 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 [16].

In Line 4, $\texttt{DijksTA}(\mathcal{A}^g_p, q^g_{p,t}, S_d, S_c)$ determines the shortest path (saved in P_1) from $q^g_{p,t}$ 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: $S_c = \{q_p \in Q^g_p \,|\, q_p|_{\Pi^g_{\mathcal{G}}} = \langle \pi^g_{\mathcal{G}}, \sigma_n \rangle, \sigma_n \in \Sigma^g_c \cup \Sigma^g_h\}$.

Request	$\mathfrak{R}_1, (h_B, r_4, 11)$	$\mathfrak{R}_4, (h_{C_1}, r_5, 14)$	$\mathfrak{R}_4, (h_{C_2}, r_5, 14)$
$\mathfrak{R}_1$	$--$	$(h_{C_1}, \bot, 0)$	$(h_{C_2}, \bot, 0)$
$\mathfrak{R}_2$	$(h_B, \top, 13.1)$	$(h_{C_1}, \top, 14.6)$	$(h_{C_2}, \bot, 0)$
$\mathfrak{R}_3$	$(h_B, \bot, 0)$	$(h_{C_1}, \bot, 0)$	$(h_{C_2}, \top, 16.2)$
$\mathfrak{R}_4$	$(h_B, \bot, 0)$	$--$	$--$
$\mathfrak{R}_5$	$(h_B, \top, 15.7)$	$(h_{C_1}, \top, 15.4)$	$(h_{C_2}, \top, 15.4)$
$\mathfrak{R}_6$	$(h_B, \top, 18.2)$	$(h_{C_1}, \bot, 0)$	$(h_{C_2}, \bot, 0)$

TABLE II: Request and reply messages exchanged for collaborative actions o_M and a_C in Sec. VII.

In Line 5, $\texttt{DijksTA}(\texttt{Reverse}(\mathcal{A}^g_p), q^g_{p,f}, S_d, \emptyset)$ is called to determined the shortest path (saved in P_2) from $q^g_{p,f}$ to every state in S_d within the reversed $\mathcal{A}^g_p$ and the associated distances (saved in C_2); $\texttt{Reverse}(\mathcal{A}^g_p)$ is the directed graph obtained by inverting the direction of all edges in $G(\mathcal{A}^g_p)$ while keeping the weights unchanged, where $G(\mathcal{A}^g_p)$ is the directed graph associated with $\mathcal{A}^g_p$ [2]. In Lines 7-8, for each state $q^g_{p,r} \in S_d$, the balanced cost of the corresponding $\widehat{R}^g_{p,+}$ by (7) is computed. The one that yields the minimal cost is denoted by $q^{g,\star}_{p,r}$. At last, $\widehat{R}^g_{p,+}$ is formed by concatenating the shortest path from $q^g_{p,t}$ to $q^{g,\star}_{p,r}$ and the reversed shortest path from $q^g_{p,f}$ to $q^{g,\star}_{p,r}$. Last but not least, if $q^{g,\star}_{p,r}$ returns empty, it means that agent k could not offer the requested collaboration thus $b^g_d = \bot$ and $t^g_d = 0$ as in Line 13. The complexity of function $\texttt{DijksTA}(\cdot)$ over $\mathcal{A}^g_p$ is $\mathcal{O}(|\delta^g_p| \cdot \log|Q^g_p|)$. Reversing $\mathcal{A}^g_p$ has the complexity linear to $\mathcal{O}(|\delta^g_p|)$.

Remark 5: Note that $\widehat{R}^g_{p,+}$ is the *potentially-revised* run, i.e., agent g does not change its current plan but saves $\widehat{R}^g_{p,+}$ in $\widehat{P}$ (see Line 5 of Alg. 2) and waits for the confirmation from agent k, which will be discussed in Sec. V-E.

It is worth mentioning that in case agent k 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. Table II shows the request and reply messages regarding two collaborative actions in Sec. VII.

Lemma 1: If $b^g_d = \top$ from Alg. 3, action σ_d can be done at the estimated time t^g_d by agent g following $\widehat{R}^g_{p,+}$.

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

D. Confirmation

Based on the replies from $g \in \mathcal{K}^k$, agent k needs to acknowledge them by sending back confirmation messages:

$$\textbf{Confirm}^k_g = \{(\sigma_d, c^g_d, f_m), \forall \sigma_d \in \texttt{Depd}^k(\sigma_m)\}, \tag{8}$$

where σ_d is the requested assisting action; c^g_d is a boolean variable, indicating whether agent g is confirmed to provide σ_d; f_m is the estimated time to finish action σ_m.

The choices of $\{c^g_d, g \in \mathcal{K}^k\}$ should satisfy two constraints: (i) *exactly* one agent in $\mathcal{K}^k_t$ can be the confirmed collaborator for each action $\sigma_d \in \texttt{Depd}^k(\sigma_m)$; (ii) each agent in $\mathcal{K}^k$ can be confirmed for *at most* one action in

Algorithm 4: Delay collaboration, `DelayCol()`

Input: $\widehat{R}^k_{p,-}$, $q^k_{p,t}$, $\langle \pi_i, \sigma_d \rangle$, $\mathcal{A}^k_p$, $\widehat{\Sigma}^k_c$
Output: $\widehat{R}^k_{p,+}$

compute S_d given $\langle \pi_i, \sigma_d \rangle$, S_c
$q^k_{p,t} = \widehat{R}^k_{p,-}[1]$, $q^k_{p,f} = \widehat{R}^k_{p,-}[\text{-}1]$
(P_1, C_1) = $\texttt{DijksTA}\big(\mathcal{A}^k_p, q^k_{p,t}, S_d, S_c\big)$
(P_2, C_2) = $\texttt{DijksTA}\big(\texttt{Reverse}(\mathcal{A}^k_p), q^k_{p,f}, S_d, \emptyset\big)$
forall the $q^k_{p,d} \in S_d$ **do**
  if $C_1(q^k_{p,d}) > T_m + D^k$ *and* $P_2(q^k_{p,d})$ *exists* **then**
    $\widehat{R}^k_{p,+} = P_1(q^k_{p,d}) + \texttt{Reverse}\big(P_2(q^k_{p,d})\big)$
    Return $\widehat{R}^k_{p,+}$

$\texttt{Depd}^k(\sigma_m)$. Meanwhile, the estimated finishing time f_m should be as early as possible.

Let $|\mathcal{K}^k| = N_1$ and $|\texttt{Depd}^k(\sigma_m)| = N_2$. Without loss of generality, denote by $\mathcal{K}^k = \{1, \cdots, N_1\}$ and $\texttt{Depd}^k(\sigma_m) = \{\sigma_1, \cdots, \sigma_{N_2}\}$. The problem of finding $\{c^g_d\}$ and f_m can be readily formulated as an integer programming problem [23]:

$$\begin{aligned} \textbf{min} \quad & f_m \\ \textbf{s.t.} \quad & f_m = \textbf{max}_d\{c^g_d \cdot t^g_d, T_m\} \\ & \sum_{d=1}^{N_2} b^g_d \cdot c^g_d \le 1, \quad \forall g \in \{1, \cdots, N_1\}, \\ & \sum_{g=1}^{N_1} b^g_d \cdot c^g_d = 1, \quad \forall d \in \{1, \cdots, N_2\}, \end{aligned} \tag{9}$$

where $(\sigma_d, b^g_d, t^g_d) \in \textbf{Reply}^g_k$ from (5). Any stand-alone integer programming solver can be used to obtain $\{c^g_d\}$ and f_m once (9) is formulated, e.g., "`Gurobi`" and "`CVXOPT`".

Then $\forall g \in \mathcal{K}^k$ and $\forall \sigma_d \in \texttt{Depd}^k(\sigma_m)$, consider two cases: (I) if (9) has a solution, both $\{c^g_d\}$ and f_m exist. If c^g_d is $\top$, add $(\sigma_d, \top, f_m)$ to $\textbf{Confirm}^k_g$; otherwise, add $(\sigma_d, \bot, 0)$ to $\textbf{Confirm}^k_g$; (II) if (9) has no solutions, add $(\sigma_d, \bot, 0)$ to $\textbf{Confirm}^k_g$. It means that σ_m can not be fulfilled according to the current replies. Then how agent k needs to delay σ_m and revise its plan will be given in Sec. V-E.

Remark 6: Note the optimization problem (9) is solved locally by agent k regarding the requested collaborative task σ_m, with $|\mathcal{K}^k| \cdot |\texttt{Depd}^k(\sigma_m)|$ Boolean variables.

E. Plan Adaptation

After sending out the confirmation messages, agent k checks the following: (I) if (9) has a solution, it means that σ_m can be fulfilled and $R^k_{p,t}$ remains unchanged. $\overline{T}^k$ is set to f_m to indicate that agent k is engaged in the collaboration until the estimated time f_m; (II) otherwise, it means that according to the current replies σ_m can not be done as planned in $R^k_{p,t}$. Thus agent k needs to revise its plan by delaying this collaborative action σ_m. Alg. 4 revises $R^k_{p,-}$ and delays σ_m by time D^k, where $D^k > 0$ is a design parameter. Function $\texttt{DijksTA}(\cdot)$ from Alg. 3 is used to find a path from $q^k_{p,t}$ to one state in S_d whose cost is larger than $T_m + D^k$ and at the same time reachable to the accepting state $q^k_{p,f}$. 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 $\textbf{Confirm}^k_g$, each agent $g \in \mathcal{K}^k$ checks the following: (I) if $b^g_d = \top$, it means agent g 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 Alg. 3. In particular, the plan segment $\widehat{R}^g_p$ is set to $\widehat{R}^g_{p,+}$ and $\overline{T}^g$ is set to f_m; (II) if agent $b^g_d = \bot, \forall \sigma_d \in \texttt{Depd}^k(\sigma_m)$, it means g is not confirmed as a collaborator. Then R^g_p remains unchanged and $\overline{T}^g$ is set to 0.

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

Theorem 2: If (9) has a solution, the estimated time of accomplishing action σ_m is f_m.

Proof: Since each assisting action $\sigma_d \in \texttt{Depd}^k(\sigma_m)$ has been assigned exactly to one agent $g \in \mathcal{K}^k$, then σ_d can be accomplished by agent g at time t^g_d by Lemma 1. Thus σ_m can be accomplished by agent k at the estimated time f_m, which is the latest time for all actions by (9). ■

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

F. Failure Detection and Recovery

Due to the unavoidable characteristics and constraints of physical robots, any agent may fail (stop being functional) at anytime. Detection of this type of failure is particularly important for collaborations involving several agents.

We propose an inquiry and acknowledgement mechanism by communication. In particular, agent k that has confirmed on its collaborative action σ_m would continuously monitor the status of its confirmed collaborators for each assisting action $\sigma_d \in \texttt{Depd}^k(\sigma_m)$ until σ_m is done. If, for instance, agent $h \in \mathcal{K}^k$ who is confirmed to offer action σ_d fails to acknowledge agent k's inquiry for a limited time, agent k may assume that agent h has failed. As a result, agent k needs to re-assign the assisting action σ_d to another agent in $\mathcal{K}^k$. This can be done as follows: (i) send a new request $\{(\pi_i, \sigma_d, T_m)\}$ to all neighbors within $\mathcal{K}^k$ and wait for the reply; (ii) based on the replies $\{(\sigma_d, b^g_d, t^g_d)\}, \forall g \in \mathcal{K}^k\}$, choose the new collaborator $\hat{h}$ for action σ_d as follows: $\hat{h} = \text{argmin}_{g \in \mathcal{K}^k}\{|f_m - t^g_d \cdot b^g_d|\}$, where f_m is the previously confirmed time from (9). Namely, it picks agent $\hat{h}$ that can offer action σ_d at the time closest to f_m; (iii) send the confirmation $\{(\sigma_d, \top, f_m)\}$ to agent $\hat{h}$; (iv) agent $\hat{h}$ would adapt its plan and be engaged in the collaboration on σ_m as described in Sec. V-E.

G. Loosely-coupled System

As mentioned in Sec. I, we aim to apply this distributed coordination scheme to loosely-coupled multi-agent systems, where collaborations among the agents are (i) local in the

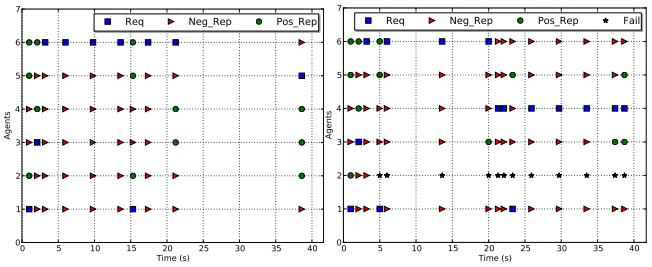

Fig. 2: Messages exchanged for both scenarios of Sec. VII. Blue square stands for request messages by (4) while red triangles and green dots stand for reply messages by (5) with b_d^g being $\perp$ and $\top$. Black stars indicate the agent failure.

sense that only neighboring agents are needed; (ii) sparse in the sense that they are needed infrequently compared with the total number of activities of all agents required by their local tasks. In particular, we impose the following assumption:

Assumption 1: There exists a finite time $\mathbf{T} > 0$ such that for each agent $k \in \mathcal{K}$ and any collaborative action σ_m requested by agent k initially at time $t_m > 0$, problem (9) for σ_m will have a solution within time $t_m + \mathbf{T}$.

Namely, the above assumption indicates that any collaborative action required by each agent $k \in \mathcal{K}$ in order to satisfy the local task φ^k should always eventually be provided in finite time by agent k and its neighbors.

VI. 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 Alg. 2, waits for the confirmation and adjusts its plan accordingly. Otherwise, it sends out requests by Alg. 1, waits for reply, sends confirmation back by (9) and at last adapts its plan by Alg. 4. The correctness of the proposed scheme is guaranteed by Theorem 3 below:

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

Proof: Starting from the initial plan $\tau^k_{\mathcal{G},\text{init}}$ for agent $k \in \mathcal{K}$, motion and local actions in $\tau^k_{\mathcal{G},\text{init}}$ can be accomplished locally by an agent itself. If $\tau^k_{\mathcal{G},\text{init}}$ remains unchanged for agent k, we only need to show that collaborative actions in $\tau^k_{\mathcal{G},\text{init}}$ can be accomplished. The fact that $\tau^k_{\mathcal{G},\text{init}}$ remains unchanged indicates that agent k's requests for each collaborative action σ_m are fulfilled, i.e., (9) for σ_m has a solution. By Theorem 2, action σ_m can be accomplished in finite time f_m. Since $\tau^k_{\mathcal{G},\text{init}}$ is a finite sequence, φ^k can be satisfied in finite time as every motion and action inside can be done in finite time. On the other hand, if $\tau^k_{\mathcal{G},\text{init}}$ has to be adapted in real-time, the reasons are: (i) agent k is confirmed to assist its neighbor g on one collaboration; (ii) agent k has made a request for a collaborative action σ_m and it is delayed by Alg. 4 as (9) has no solution. For case (i), Alg. 3 guarantees that after the assistance its updated run $\hat{R}^k_{p,+}$ still satisfies φ^k in finite time. For case (ii), by Assumption 1, (9) 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. ■

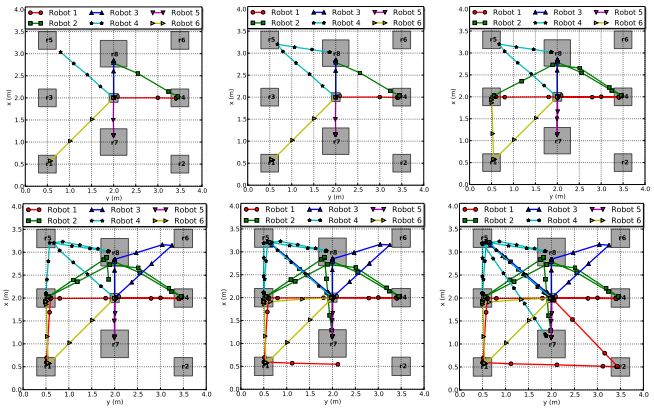

Fig. 3: Snapshots of agents collaborating on actions l_B, o_M, u_B, c_F and a_C. Detail descriptions are in Sec. VII-C.1.

VII. Case Study

In the case study, we present a simulated example of six autonomous robots with heterogeneous capacities. The proposed algorithms are implemented in Python 2.7.

A. System Description

The workspace of size $4m \times 4m$ is given in Fig. 1, within which there are nine rectangular regions of interest r_0, r_1, $r_2, \cdots, r_8$ (in gray). The regions are labelled by the objects of interest, like A, B, C, M, E, F.

Denote by the six agents $\mathfrak{R}_1, \mathfrak{R}_2, \cdots, \mathfrak{R}_6$. They all satisfy the single-integrator dynamics, i.e., $\dot{x} = u$, where $x, u \in \mathbb{R}^2$ are the 2-D position and velocity. The agents have velocities between $0.6m/s$ and $1m/s$. Besides, the action sets of each agent are shown in Table I. Agent $\mathfrak{R}_1$ can load and unload (l_A, u_A) a light object A; load and unload (l_B, u_B) a heavy object B with others' assisting action h_B. Agent $\mathfrak{R}_2$ can take pictures (s); help load and unload (h_B) B; help assemble (h_{C1}) C. Agent $\mathfrak{R}_3$ can operate (o_M) machine M with assisting action h_M; help assemble (h_{C2}) C. Agent $\mathfrak{R}_4$ can assemble (a_C) C with assisting actions h_{C1}, h_{C2}; help operate (h_M) M. Agent $\mathfrak{R}_5$ can maintain D, help assemble (h_{C1}, h_{C2}) C, and help load and unload (h_B) B. Agent $\mathfrak{R}_6$ can operate object (o_E) E, charge object (c_F) F with assisting action h_F, help operate (h_M) M and help load and unload (h_B) B. We assume all actions have duration $10s$ and the time horizon H^k is set to $20s$ uniformly. Any two agents can exchange messages directly. "`Gurobi`" for Python is used as the mix integer programming solver.

B. Task Description

Each agent is assigned a local task: agent $\mathfrak{R}_1$ has to deliver A to r_2 and B to r_3. Then $\varphi_1 = \Diamond\big(l_A \wedge \Diamond(r_2 \wedge \bigcirc u_A)\big) \wedge \Diamond\big(l_B \wedge \Diamond(r_3 \wedge \bigcirc u_B)\big)$; Agent $\mathfrak{R}_2$ has to surveil regions r_7, r_8 and take pictures there. $\varphi_2 = \Diamond(r_7 \wedge \bigcirc s) \wedge \Diamond(r_8 \wedge \bigcirc s)$; Agent $\mathfrak{R}_3$ has to operate M and visit r_6. $\varphi_3 = \Diamond(o_M \wedge \Diamond r_6)$; Agent $\mathfrak{R}_4$ has to take a photo in r_7, assemble C and visit r_6. $\varphi_4 = \Diamond(r_7 \wedge \bigcirc s) \wedge \Diamond(a_C)$; Agent $\mathfrak{R}_5$ needs to maintain D and back to region r_0. $\varphi_5 = \Diamond(m_D \wedge \Diamond r_0)$; Agent $\mathfrak{R}_6$ needs to operate E first and then charge F. $\varphi_6 = \bigcirc(o_E \wedge \Diamond c_F)$.

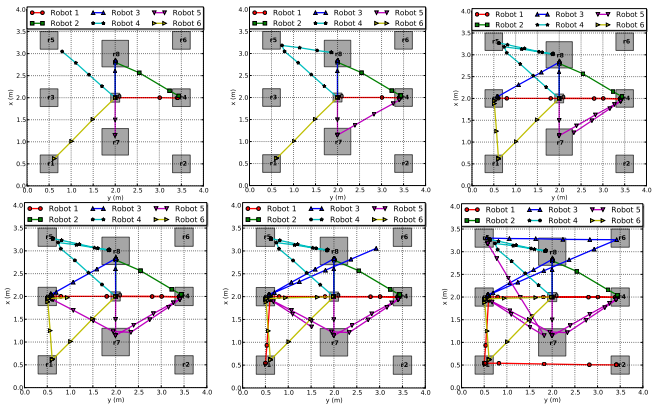

Fig. 4: Snapshots of agents collaborating on actions l_B, o_M, u_B, c_F and a_C for Sec. VII-C.2. Note that $\mathfrak{R}_2$ failed at $t = 5s$, instead $\mathfrak{R}_5$ is re-assigned to assist $\mathfrak{R}_1$ on l_B, u_B.

C. Simulation Results

All agents start form the center $(2m, 2m)$. The synthesized initial plan of each agent is as follows: $r_0 r_4 \, l_B \, r_3 \, u_B \, r_1 \, l_A \, r_2 \, u_A$ for $\mathfrak{R}_1$; $r_0 \, r_8 \, s \, r_7 \, s$ for $\mathfrak{R}_2$; $r_0 \, r_8 \, o_M \, r_6$ for $\mathfrak{R}_3$; $r_0 \, r_5 \, a_C \, r_7 \, s$ for $\mathfrak{R}_4$; $r_0 \, r_7 \, m_D$ for $\mathfrak{R}_5$; $r_0 \, r_1 \, o_E \, r_3 c_F$ for $\mathfrak{R}_6$. We simulate first one nominal scenario and another with $\mathfrak{R}_2$'s failure since $t = 5s$. The messages exchanged during both scenarios are shown in Fig. 2 and the simulation videos are available online [21].

1) Nominal Scenario: As shown in Fig. 3, agent $\mathfrak{R}_1$ firstly sends the request on action h_B and the reply messages are shown in Table II. $\mathfrak{R}_2$ is confirmed as the collaborator and changes its plan to $r_0 \, r_4 \, h_B \, r_8 \, s \, r_7 \, s$. $\mathfrak{R}_1$ finishes action l_B at $t = 14.3s$ At the same time, $\mathfrak{R}_4$ is chosen to help $\mathfrak{R}_3$ on action o_M, which is done at $t = 21.1s$. Afterwards, $\mathfrak{R}_1$ finishes u_B at $t = 28.4s$ with $\mathfrak{R}_2$ offering h_B. Agent $\mathfrak{R}_6$'s request for action c_F keeps getting delayed until $t = 21.1s$, as $\mathfrak{R}_2$, $\mathfrak{R}_3$, $\mathfrak{R}_4$ are engaged in other collaborations. Afterwards agent $\mathfrak{R}_4$ is confirmed to offer action h_F and c_F is done at $t = 38.0s$. After that, agent $\mathfrak{R}_4$ sends the request for a_C regarding h_{C_1} and h_{C_2}. The reply messages are shown in Table II. After solving (9), $\mathfrak{R}_2$ and $\mathfrak{R}_3$ offer actions h_{C_1} and h_{C_2}, respectively, which are done at $t = 54.1s$. By $t = 70.3s$, all agents accomplish their local tasks.

2) Failure Recovery: To illustrate the effectiveness of our approach for handling agent failures, we stop simulating agent $\mathfrak{R}_2$ since $t = 5s$ whereas the other agents remain the same. As shown in Fig. 4, initially agent $\mathfrak{R}_2$ is confirmed to offer h_B to $\mathfrak{R}_1$. However since $\mathfrak{R}_2$ fails at $t = 5s$, $\mathfrak{R}_1$ detects that by the inquiry and acknowledgment mechanism described in Sec. V-F and resends a request regarding h_B, for which $\mathfrak{R}_5$ is confirmed as the new collaborator. Then l_B is done at $t = 22.3s$. Afterwards, again with the help of $\mathfrak{R}_5$, $\mathfrak{R}_1$ finishes u_B at $t = 37.8s$. $\mathfrak{R}_3$ finishes o_M with the help of $\mathfrak{R}_4$ at $t = 29.7s$. Before this time, agent $\mathfrak{R}_6$ has to delay its action c_F as both $\mathfrak{R}_3$ and $\mathfrak{R}_4$ are engaged in o_M and $\mathfrak{R}_2$ has failed. Then $\mathfrak{R}_3$ offers h_F to $\mathfrak{R}_6$ at $t = 36s$. After that, $\mathfrak{R}_4$ finishes a_C with the help of $\mathfrak{R}_3$ and $\mathfrak{R}_5$ at $t = 68.9s$. At last, each agent accomplishes its local task by $t = 76.5s$.

VIII. Summary and Future Work

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

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. *IEEE International Conference on Robotics and Automation(ICRA)*, 2689–2696, 2010.

[2] C. Baier and J.-P Katoen. Principles of model checking. *The MIT Press*, 2008.

[3] C. Belta, A. Bicchi, M. Egerstedt, E. Frazzoli, E. Klavins and G. J. Pappas. Symbolic planning and control of robot motion. *IEEE Robotics and Automation Magazine*, 14(1): 61–70, 2007.

[4] X. Ding, M. Kloetzer, Y. Chen and C. Belta. Automatic deployment of robotic teams. *IEEE Robotics Automation Magazine*, 18: 75–86, 2011.

[5] G. E. Fainekos, A. Girard, H. Kress-Gazit and G. J. Pappas. Temporal Logic Motion Planning for Dynamic Mobile Robots. *Automatica*, 45(2): 343–352, 2009.

[6] G. E. Fainekos. Revising temporal logic specifications for motion planning. *IEEE Conference on Robotics and Automation*, 2011.

[7] I. F. Filippidis, D. V. Dimarogonas and K. J. Kyriakopoulos. Decentralized Multi-Agent Control from Local LTL Specifications. *IEEE Conference on Decision and Control(CDC)*, 6235–6240, 2012.

[8] P. Gastin and D. Oddoux. Fast LTL to Büchi automaton translation. *International Conference on Computer Aided Verification*, 2001.

[9] M. Guo, K. H. Johansson and D. V. Dimarogonas. Motion and Action Planning under LTL Specification using Navigation Functions and Action Description Language. *IEEE/RSJ International Conference on Intelligent Robots and Systems(IROS)*, 240–245, 2013.

[10] M. Guo and D. V. Dimarogonas. Reconfiguration in Motion Planning of Single- and Multi-agent Systems under Infeasible Local LTL Specifications. *IEEE Conference on Decision and Control(CDC)*, 2013.

[11] M. Guo and D. V. Dimarogonas. Multi-agent Plan Reconfiguration under Local LTL Specifications. *International Journal of Robotics Research*, 34(2): 218-235, 2015.

[12] S. Karaman and E. Frazzoli. Vehicle routing with linear temporal logic specifications: Applications to Multi-UAV Mission Planning. *Navigation, and Control Conference in AIAA Guidance*, 2008.

[13] M. Kloetzer, X. C. Ding and C. Belta. Multi-robot deployment from LTL specifications with reduced communication, *IEEE Conference on Decision and Control(CDC)*, 4867–4872, 2011

[14] O. Kupferman and M. Y. Vardi. Model checking of safety properties. *Formal Methods in System Design*, 19:291–314, 2001.

[15] H. Kress-Gazit, T. Wongpiromsarn and U. Topcu. Correct, reactive robot control from abstraction and temporal logic specifications. *IEEE Robotics and Automation Magazine*, 18(3): 65-74, 2011.

[16] S. M. LaValle. Planning algorithms. *Cambridge University Press*, 2006.

[17] S. G. Loizou and K. J. Kyriakopoulos. Automatic synthesis of multi-agent motion tasks based on LTL specifications. *IEEE Conference on Decision and Control(CDC)*, 78–83, 2005.

[18] T. Wongpiromsarn, U. Topcu, and R. M. Murray. Receding horizon control for temporal logic specifications. *13th ACM International Conference on Hybrid Systems: Computation and Control*, 101-110, 2010.

[19] I. Pohl. Bi-directional search. *IBM TJ Watson Research Center*, 1970.

[20] J. Tumova and D. V. Dimarogonas. A receding horizon approach to multi-agent planning from local LTL specifications. *American Control Conference (ACC)*, 1775–1780, 2014.

[21] Simulation videos. `https://drive.google.com/folderview?id=0B3YLWjQD8hZhWjI2cUlKSlUxT0E&usp=sharing`

[22] A. Ulusoy, S. L. Smith, X. C. Ding, C. Belta, D. Rus. Optimality and robustness in multi-robot path planning with temporal logic constraints. *The International Journal of Robotics Research*, 32(8): 889–911, 2013.

[23] L. A. Wolsey. *Integer Programming*. New York: Wiley, 1998.