Leveraging Quadrupedal Robots in Heterogeneous Multi-Robot Teaming with Run-Time Disturbances

Ziyi Zhou[1], and Ye Zhao[1]

***Abstract*— This work aims to establish a reactive planning framework under both single- and multi-robot scenarios given Linear Temporal Logic (LTL) specifications. The capabilities of both quadrupedal and wheeled robots are leveraged via a heterogeneous team to accomplish a variety of navigation and delivery tasks. However, when deployed in the real world, all robots especially the legged robots, can be susceptible to different types of disturbances, including but not limited to locomotion failures, human interventions, and environmental changes. To address these run-time disturbances, we propose task-level reallocation strategies to coordinate the robot teaming online while guaranteeing the completion of the original task. A locomotion-level reactive planner is designed for legged robots to achieve the task-level goal in response to unexpected terrain changes, which are not considered in the task-level.**

I. Introduction

Mobile robots have been extensively investigated and deployed in various service applications such as assembly [1], surveillance, [2] and search and rescue [3]. Distinct types of robots can form a heterogeneous team as shown in Fig. 1 to compensate for their individual disadvantages. Recent works on multi-robot systems have been focusing on mission planning problems with the assistance of formal languages such as Linear Temporal Logic (LTL) [4]. Originally proposed for model checking [5], LTL is a powerful tool used in the robotics community with a preponderance of research primarily conducted on wheeled robots [6], [7] and legged robots [8]–[10] for task and motion planning. There have also been works [6], [11]–[14] on multi-agent systems. However, objectives are explicitly assigned to individual robots rather than having one global specification. This can be challenging for a large team of robots [15], [16], where in most cases, a global task is simpler to define. Therefore, a simultaneous task allocation and planning (STAP) problem given a global LTL specification attracts more attention.

A line of research exists where STAP problems have been solved with global LTL specifications. In [2], [17] a product model is constructed with an exponential complexity, while [18] proposed a team model automaton with a linear complexity, assuming that each robot conducts its task independently. Other works also focused on advanced search algorithms [15], [19], concrete time constraints [20], and collaborative tasks [21]. However, failure recovery during real-world deployment is rarely studied, especially considering unstructured environments such as rough terrain. Single and multi-robot scenarios have been demonstrated with disturbances caused by a change in the environment [22]–[25] or a failure to perform an action [26], but all have been limited to local LTL specifications. Therefore, task reallocation capabilities during the online execution is desirable for STAP problems given global LTL specifications, which is addressed in this study.

Fig. 1: A multi-robot teaming consists of two turtlebots and a Mini Cheetah.

Meanwhile, quadrupedal robots have been popularized for their superior traversability over unstructured terrains [27]. Nevertheless, even with exceptional locomotion capabilities, legged systems are often unstable, fragile, and less suitable for performing prolonged tasks compared to wheeled robots. Although the aforementioned task reallocation can handle run-time failures that are explicitly abstracted, a lower-level planner is dispensable to allow robust locomotion when following the task-level actions. Another set of LTL specification can be designed only for a single legged robot to ensure reasonable motion primitive switch in reaction to unexpected terrain changes.

Our contributions are summarized as follows:

- In the task level, to handle potential disturbances, we propose local and global task reallocation approaches given a global LTL specification. This approach eliminates the need to reconstruct the entire team model or resynthesize a new task.
- In the locomotion level, we propose a reactive locomotion planner given a local LTL specification. A robot-centric terrain map serves as an input to the planner, which chooses an appropriate motion primitive such as walking or jumping to robustly navigate dynamic environments.
- We evaluate our pipeline in a simulated hospital environment with a heterogeneous robot team consisting of both quadrupedal and mobile robots. An open-source software package[1] is provided for the proposed reactive multi-robot task allocation and planning framework.

[1]The authors are with the Laboratory for Intelligent Decision and Autonomous Robots, Woodruff School of Mechanical Engineering, Georgia Institute of Technology. {zhouziyi, yzhao301}@gatech.edu.

[1]https://github.com/GTLIDAR/ltl_multi_agent.

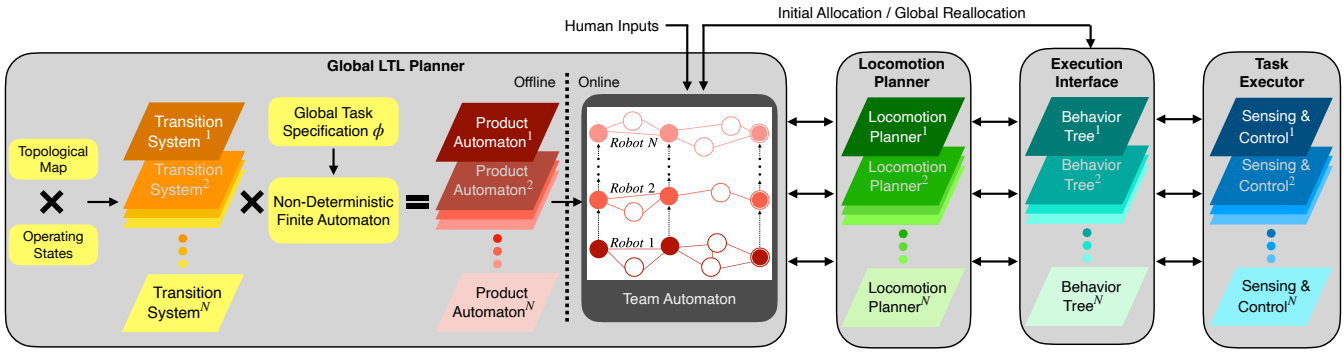

Fig. 2: An overview of the hierarchical planning framework. The global LTL planner portrays the high-level LTL task planner. Once the product automaton is created, it is sent to the team automaton, where a sequence of actions are assigned to individual robots. During execution, the locomotion planner, the behavior tree, and the hardware on the robot will react to disturbances and handle accordingly.

II. Preliminaries

A. LTL basics

Linear temporal logic (LTL) has been widely used to encode temporal task specifications and automatically synthesize the system's transition behaviors. A specification ϕ is constructed from atomic propositions $\pi \in \Pi$, which is evaluated to be True or False and follows the syntax $\phi := \pi | \neg\phi | \phi_1 \wedge \phi_2 | \bigcirc\phi | \phi_1 \mathcal{U} \phi_2 | \phi_1 \mathcal{R} \phi_2$. Boolean operators $\neg$ "not" and $\wedge$ "and" in addition to a set of temporal operators $\bigcirc$ "next", $\mathcal{U}$ "until", and $\mathcal{R}$ "release", are denoted. To be concise, we omit the derivations of other boolean operators such as $\vee$ "or", $\rightarrow$ "implies", and $\leftrightarrow$ "if and only if", as well as temporal operators $\Diamond\phi$ "eventually ϕ" and $\Box\phi$ "always ϕ".

A common usage of LTL is for constructing an automaton. A non-deterministic automaton (NFA) is defined as a tuple $\mathcal{Q} = (S_\mathcal{Q}, S_{0,\mathcal{Q}}, \Sigma, \delta_\mathcal{Q}, F)$ such that $S_\mathcal{Q}$ is a set of states ($s_\mathcal{Q} \in S_\mathcal{Q}$), $S_{0,\mathcal{Q}} \subseteq S_\mathcal{Q}$ is a set of initial state, Σ is the input alphabet, $\delta_\mathcal{Q}$ is a set of transition relations such that $\delta_\mathcal{Q} : S_\mathcal{Q} \times \Sigma \rightarrow 2^{S_\mathcal{Q}}$, and F is a set of accepting final states. In addition, LTL formulas are evaluated over a sequence $\sigma : \mathbb{N} \rightarrow 2^\Pi$ where $\sigma(t) \subset \Pi$ represents all true propositions at time t [28].

For this framework, a transition system (TS) is created by combining data from the topological map and the robots' operating states. A TS is defined as a tuple $\mathcal{T} = (S_\mathcal{T}, s_{0,\mathcal{T}}, A_\mathcal{T}, \Pi_\mathcal{T}, \mathcal{L})$ such that $S_\mathcal{T}$ is a set of system states ($s_\mathcal{T} \in \mathcal{S}_\mathcal{T}$), $s_{0,\mathcal{T}} \in \mathcal{S}_\mathcal{T}$ is the initial system state, $A_\mathcal{T}$ is a set of available system actions, $\Pi_\mathcal{T}$ is the set of system propositions, and $\mathcal{L} : S_\mathcal{T} \rightarrow 2^{\Pi_\mathcal{T}}$ is a labeling function that assigns atomic propositions to states [15]. We use $\texttt{Succ}(s_\mathcal{T}) = \{s'_\mathcal{T} \in S_\mathcal{T} | (s_\mathcal{T}, s'_\mathcal{T}) \in A_\mathcal{T}\}$ to denote the successors of $s_\mathcal{T}$ and $\texttt{Pred}(s_\mathcal{T}) = \{s^*_\mathcal{T} \in S_\mathcal{T} | (s^*_\mathcal{T}, s_\mathcal{T}) \in A_\mathcal{T}\}$ as the predecessors of $s_\mathcal{T}$. By combining a TS with an NFA, a product automaton (PA) $\mathcal{P}$ composed of system states and mission specifications is generated. It is represented by $\mathcal{P} = \mathcal{Q} \otimes \mathcal{T} = (S_\mathcal{P}, S_{0,\mathcal{P}}, A_\mathcal{P})$ such that $S_\mathcal{P} = S_\mathcal{Q} \times S_\mathcal{T}$ is the set of states ($s_\mathcal{P} \in S_\mathcal{P}$), $S_{0,\mathcal{P}} = S_{0,\mathcal{Q}} \times \{s_{0,\mathcal{T}}\}$ is the set of initial states, and $A_\mathcal{P} = \{((s_\mathcal{Q}, s_\mathcal{T}), (s'_\mathcal{Q}, s'_\mathcal{T})) \in S_\mathcal{P} \times S_\mathcal{P} : (s_\mathcal{T}, s'_\mathcal{T}) \in A_\mathcal{T} \wedge s'_\mathcal{Q} \in \delta_\mathcal{Q}(s_\mathcal{Q}, \mathcal{L}(s_\mathcal{T}))\}$.

B. Offline task allocation

We introduce the baseline task allocation method and terminologies used throughout this paper. Readers are referred to [18] for more details. Given the LTL semantics above, a global task ϕ along with its corresponding NFA $\mathcal{Q}$ can be specified for a whole team of N agents, each of which has its own TS $\mathcal{T}^{(r)}$ and a corresponding PA $\mathcal{P}^{(r)} = \mathcal{Q} \otimes \mathcal{T}^{(r)}, r = \{1, \ldots, N\}$. Next, we will introduce a criterion that identifies the decomposed parts of ϕ based on the assumption that each agent executes its sub-task independently.

Definition 1 (Finite Decomposition [18]). *Let $\mathcal{J}_r$ with $r \in \{1, \ldots, N\}$ be a set of finite LTL task specifications and σ_i is any sequence s.t. $\sigma_r \models \mathcal{J}_r$. These tasks are called decomposition of the global finite LTL specification ϕ, iff:*

$$\sigma_{j_1} \ldots \sigma_{j_r} \ldots \sigma_{j_N} \models \phi \tag{1}$$

for all permutations of $j_r \in \{1, \ldots, N\}$ and all respective sequences σ_r.

Based on this criterion, a decomposition set $\mathcal{D} \subseteq S_\mathcal{Q}$ can be derived from $\mathcal{Q}$, which includes all NFA states that allocate tasks. Then the action-state sequence allocated to each agent is computed by first constructing a team automaton.

Definition 2 (Team automaton [18]). *The team automaton $\mathcal{G}$ is a union of N local PA $\mathcal{P}^{(r)}$ with $r \in \{1, \ldots, N\}$ and defined by $\mathcal{G} := (S_\mathcal{G}, S_{0,\mathcal{G}}, F_\mathcal{G}, A_\mathcal{G})$, where:*

- $S_\mathcal{G} = \{(r, s_\mathcal{Q}, s_\mathcal{T}) : r \in \{1, \ldots, N\}, (s_\mathcal{Q}, s_\mathcal{T}) \in S^{(r)}_\mathcal{P}\}$ *is the set of states;*
- $S_{0,\mathcal{G}} = \{(r, s_\mathcal{Q}, s_\mathcal{T}) : r = 1, (s_\mathcal{Q}, s_\mathcal{T}) \in S^{(1)}_{0,\mathcal{P}}\}$ *is the set of initial states;*
- $F_\mathcal{G} = \{(r, s_\mathcal{Q}, s_\mathcal{T}) \in S_\mathcal{G} : s_\mathcal{Q} \in F\}$ *is the set of final accepting states;*
- $A_\mathcal{G} = \bigcup_r A^{(r)}_\mathcal{P} \cup \zeta$ *is the set of actions including switch transitions ζ.*

We use $\delta_\mathcal{G} : S_\mathcal{G} \rightarrow S_\mathcal{G}$ to denote the transitions corresponding to $A_\mathcal{G}$. The set $\zeta \subset S_\mathcal{G} \times S_\mathcal{G}$ denotes the *switch transitions*, each of which $\varsigma = ((i, s_\mathcal{Q}, s_\mathcal{T}), (j, s'_\mathcal{Q}, s'_\mathcal{T}))$ is defined as a transition between two states in $\mathcal{G}$ iff: 1) $j = i + 1$: connects to the next agent; 2) $s_\mathcal{Q} = s'_\mathcal{Q}$: the

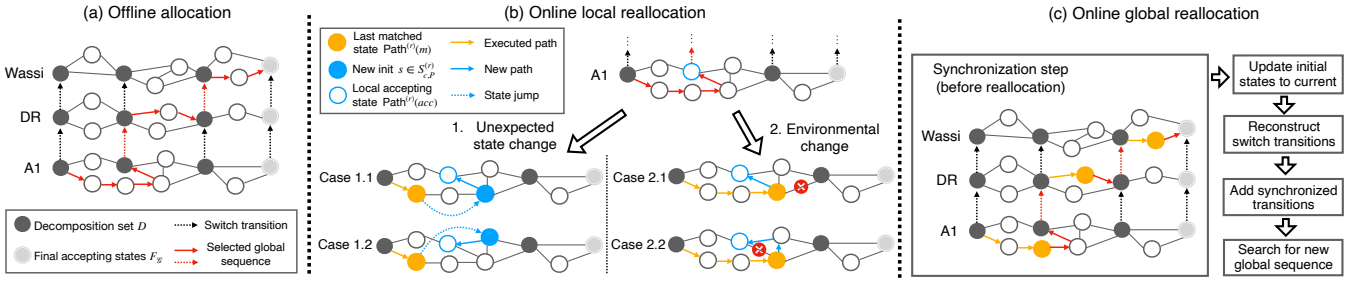

Fig. 3: Conceptual illustrations of local and global reallocations: (a) offline allocation given the team automaton. A global sequence (the red lines with arrows) is found during the offline phase as an initial task allocation. Assuming A1 undergoes a disturbance such as *unexpected state change* and *environmental change* as shown in subfigure (b). In correspondence to Algorithm 1, Case 1.1 refers to a scenario where the new initial state after the state jump is on the original offline-generated global sequence, while Case 1.2 corresponds to a completely new initial state and requires a replanning. Similarly in Algorithm 2, Case 2.1 demonstrates a scenario where the transition change doesn't affect the execution of the original global path, while a replanning is required in Case 2.2 due to a deleted edge on the original global path (represented by the cross marker). Subfigure (c) reveals a block diagram on the essential steps for a global reallocation. The synchronization step notifies the planner of each agent's current state (denoted by the yellow circle) and execution history to proceed with the remaining steps. More details are explained in Sec. IV-C.

NFA state remains unchanged; 3) $s'_{\mathcal{T}} = s^{(j)}_{0,\mathcal{T}}$: points to an initial agent state; 4) $s_{\mathcal{Q}} \in \mathcal{D}$: the NFA state is inside the decomposition set.

The team automaton is a combination of every agent's PA $\mathcal{P}^{(r)}$ [2] with additional switch transitions which can reassign an agent's remaining task to another agent. As illustrated in Fig. 3(a), if a global action sequence β on the team automaton is found, we can state that task allocation and planning have been accomplished simultaneously (namely STAP [18]). By projecting β onto the PA of each agent, tasks can be executed in parallel. This process of finding an initial set of action-state sequence is called the *offline allocation*, whereas the state space complexity scales linearly with the number of agents. The rest of this paper will focus on addressing external disturbances during real-world deployment.

III. Problem Formulation

We first abstract the possible disturbances into multiple classes. Unless specified, the following disturbances apply to both legged and wheeled robots.

- **Loss of balance** refers to a scenario where a *legged robot* falls due to an unstable gait or erratic controller output.
- **Critical failure** refers to an irrecoverable hardware or software malfunction such as a damaged motor or a software glitch.
- **Unexpected robot state change** refers to a situation where a robot detects a sudden shift in the robot's state.
- **Environmental change** refers to an environmental event preventing the robot from continuing its current task. For example, navigating from region A to region B might not be feasible any more if the passage is blocked.
- **Terrain change** represents a height variation during locomotion by querying a 2.5D grid map centered around the robot itself, which is only considered for *legged robot*.

To handle the above disturbances, we adopt a hierarchical framework that handles each category of disturbance. For the first four types of disturbances, we seek to find a reallocation strategy that is specific to each category of disturbance. Two aspects need to be investigated: 1) a formal guarantee to complete the global task; 2) a set of completed tasks by the whole team before a reallocation is triggered. To this end, we define a STAP reallocation problem as the following:

Problem Statement. *Given an initial task assignment and the current TS of every agent, one finds a set of new action-state sequences for the agents to accomplish the global task without restarting the whole mission or re-synthesizing a new mission.*

To address the terrain change which requires a finer abstraction of the environment, a reactive locomotion planner is designed separately in a single-robot level. A set of local LTL specifications that ensures a reliable transition between different locomotion primitives is the other problem to be solved in this work.

Fig. 2 shows the hierarchical architecture consisting of 1) a high-level task planner that performs offline allocation and online reallocation; 2) a mid-level locomotion planner to execute the assigned action plan for a legged robot while ensuring robust locomotion given terrain changes; 3) low-level monitoring using Behavior Tree (BT) and controllers that power the actuators on legged and wheeled robots.

IV. Multi-Robot Reallocation

To solve the STAP reallocation problem, we propose two approaches: a local and global approach, both of which are designed at the high level. Fig. 3(b) and 3(c) show the workflow and conceptual examples of both local and global reallocation. We consider two fundamental types of changes, which are general enough to incorporate specific types of abstracted disturbances.

A. Local reallocation addressing unexpected state changes

The offline task allocation process generates an action-state sequence for each agent, which assumes every action is performed successfully. During execution unexpected interventions could occur, which would undermine the orig-

[2]For simplicity, we abuse $\mathcal{P}^{(r)}$ to denote a sub-graph in $\mathcal{G}$ which contains the same states $S^{(r)}_{\mathcal{P}}$ and transitions except, the robot index r is appended.

inal plan. For instance, if a human removes a load carried by the robot before the robot reaches its destination, an unforeseen robot state change occurs. To resolve this intervention, we introduce the local task reallocation approach. Suppose $\texttt{Path}^{(r)}$ is a planned state sequence for robot r, generated from the original team automaton. Let $\texttt{Path}^{(r)}(\text{acc})$ denote the agent's local accepting state and $s_{\mathcal{T}}^{\triangle}$ be the current state after the intervention. By comparing the state sequence execution history and $\texttt{Path}^{(r)}$, the last matched state is identified as $\texttt{Path}^{(r)}(m)$, whose NFA and TS are written as $\texttt{Path}_{\mathcal{Q}}^{(r)}(m)$ and $\texttt{Path}_{\mathcal{T}}^{(r)}(m)$. Thus, $\texttt{Path}_{\mathcal{T}}^{(r)}(m+1) = s_{\mathcal{T}}^{\triangle}$. Now, the problem is reformulated to find a path on the local PA, starting from an up-to-date initial set defined as $S_{c,\mathcal{P}}^{(r)} = \{(r, s_{\mathcal{Q}}, s_{\mathcal{T}}) \in S_{\mathcal{P}}^{(r)} | s_{\mathcal{T}} = s_{\mathcal{T}}^{\triangle}, \forall s_{\mathcal{Q}} \in \delta_{\mathcal{Q}}(\texttt{Path}_{\mathcal{Q}}^{(r)}(m), \mathcal{L}(\texttt{Path}_{\mathcal{T}}^{(r)}(m)))\}$. The `find-path` method throughout this work is performed by Dijkstra's algorithm. Note that the original path can be reused if the current state happens to be on the agent's original path. This local task reallocation approach is summarized in Algorithm 1. $\texttt{Path}^{(r)}(i:)$ denotes a sub-path starting from the i^{th} element.

Algorithm 1 Local reallocation: unexpected state change

Input: $\mathcal{P}^{(r)}, \texttt{Path}^{(r)}$
Output: A new path $\texttt{Path}^{(r)+}$

```
Path^(r)+ ← empty path
Path-set^(r) ← empty set
for s in S_{c,P}^(r) do
    if s in Path^(r) then
        i ← getIndex(Path^(r)(s))
        Path-set^(r).append(Path^(r)(i:))
        continue
    else
        Path-set^(r).append(find-path(s, Path^(r)(acc)))
    end if
end for
Path^(r)+ ← find-best(Path-set^(r))
```

B. Local reallocation addressing transition system changes

In the previous section, the disturbance shifts the robot's state but does not modify the transition between different states, which will be addressed in this section. Such a disturbance will directly impact the TS. For instance, if the floor is occupied by an impassable object, the mobile robot would encounter a navigation failure and would not be able to transition to its next expected state. The critical failure can also be understood as no transitions exist at current state. In this case, the robot will receive the changes to be made on TS called $Info(t)^{(r)}$. Each update contains three types of information: 1) $(s_{\mathcal{T}}, s'_{\mathcal{T}}) \in Add(t)$ if $s_{\mathcal{T}}$ is allowed to transit to $s'_{\mathcal{T}}$; 2) $(s_{\mathcal{T}}, s'_{\mathcal{T}}) \in Delete(t)$ if $s_{\mathcal{T}}$ is not allowed to transit to $s'_{\mathcal{T}}$; 3) $(b, s_{\mathcal{T}}) \in Relabel(t)$ if the labeling function of state $s_{\mathcal{T}}$ is updated to $b \subseteq 2^{AP}$.

The TS change can be reflected by directly modifying the team automaton using the PA revision strategy in [22]. When a single agent r receives an update, the latest team automaton is revised by only updating corresponding $\mathcal{P}^{(r)}(t)$[3]. All deleted transitions, i.e., edges, are added into a set $R(t)$.

Definition 3 (Updating rules)**.** *$\mathcal{G}(t)$ (more specifically, only $\mathcal{P}^{(r)}(t)$) is updated given the $Info(t)$ from agent r following the rules:*

- *If $(s_{\mathcal{T}}, s'_{\mathcal{T}}) \in Add(t)$, $(r, s^n_{\mathcal{Q}}, s'_{\mathcal{T}})$ is in $\delta_{\mathcal{G}}((r, s^m_{\mathcal{Q}}, s_{\mathcal{T}}))$ for $\forall s^n_{\mathcal{Q}}, s^m_{\mathcal{Q}}$ satisfying $s^n_{\mathcal{Q}} \in \delta_{\mathcal{Q}}(s^m_{\mathcal{Q}}, \mathcal{L}(s_{\mathcal{T}}))$;*
- *If $(s_{\mathcal{T}}, s'_{\mathcal{T}}) \in Delete(t)$, $(r, s^n_{\mathcal{Q}}, s'_{\mathcal{T}})$ is deleted from $\delta_{\mathcal{G}}((r, s^m_{\mathcal{Q}}, s_{\mathcal{T}}))$ for $\forall s^n_{\mathcal{Q}}, s^m_{\mathcal{Q}} \in S^{(r)}_{\mathcal{Q}}$;*
- *If $(b, s_{\mathcal{T}}) \in Relabel(t)$, then $\forall s^*_{\mathcal{T}} \in \texttt{Pred}(s_{\mathcal{T}})$: $(r, s^n_{\mathcal{Q}}, s^*_{\mathcal{T}})$ is added to $\delta_{\mathcal{G}}((r, s^m_{\mathcal{Q}}, s_{\mathcal{T}}))$ for $\forall s^n_{\mathcal{Q}} \in \delta_{\mathcal{Q}}(s^m_{\mathcal{Q}}, b)$; $(r, s^n_{\mathcal{Q}}, s^*_{\mathcal{T}})$ is deleted from $\delta_{\mathcal{G}}((r, s^m_{\mathcal{Q}}, s_{\mathcal{T}}))$ for $\forall s^n_{\mathcal{Q}} \notin \delta_{\mathcal{Q}}(s^m_{\mathcal{Q}}, b)$*

If a disturbance was detected on an agent's TS, a new type of task reallocation algorithm is necessary. Note that no unexpected robot state is assumed in this case and the last matched state is equivalent to the current state, i.e. $\texttt{Path}_{\mathcal{T}}^{(r)}(m) = s_{\mathcal{T}}^{\triangle}$. Given the revised local PA $\mathcal{P}^{(r)}(t)$, we propose a different replanning approach in Algorithm 2, compared to the one in Sec. IV-A.

Algorithm 2 Local reallocation: environmental change

Input: $\mathcal{P}^{(r)}(t), \texttt{Path}^{(r)}, Info(t)$
Output: A new path $\texttt{Path}^{(r)+}$

```
Path^(r)+ ← empty path
P^(r)(t), R(t) ← UpdatePA(P^(r)(t), Info(t))
if R(t) ∩ edge(Path^(r)) ≠ ∅ then
    Path^(r)+ ← find-path(Path^(r)(m), Path^(r)(acc))
else
    Path^(r)+ ← Path^(r)(m:)
end if
```

C. Global reallocation

The two aforementioned local task reallocation approaches do not consider replanning for the whole team, which results in a sub-optimal strategy. Furthermore, if the local task reallocation fails to find a new plan for the agent, a succeeding global task reallocation over the entire team is activated. First, a synchronization step will be executed where the task planner requests for each agent's current TS state and sets it to be the latest initial TS state $s_{0,\mathcal{T}}^{(r)}(t)$. Then the initial PA set $S_{0,\mathcal{P}}^{(r)}(t)$ is updated accordingly by keeping $S_{0,\mathcal{Q}}^{(r)}$ the same. Since each $\mathcal{P}^{(r)}(t)$ has been updated during local reallocation if needed, the team automaton is only modified by updating the initial set of states and switch transitions, in addition to appending the *synchronized transitions*.

Definition 4 (Synchronized team automaton)**.** *The synchronized team model $\mathcal{R} := \mathcal{G}(t)$ is a union of N product automata $\mathcal{P}^{(r)}(t)$ given the updated product states after synchronization, where $r \in \{1, \ldots, N\}$ and $\mathcal{R} := (S_{\mathcal{R}}, S_{0,\mathcal{R}}, F_{\mathcal{R}}, A_{\mathcal{R}})$ consists of:*

- *$S_{\mathcal{R}} = \{(r, s_{\mathcal{Q}}, s_{\mathcal{T}}) : r \in \{1, \ldots, N\}, (s_{\mathcal{Q}}, s_{\mathcal{T}}) \in S_{\mathcal{P}}^{(r)}(t)\}$ is the set of states;*

[3]We use $\cdot(t)$ to denote an updated automaton at time t given the transition relation is changed.

- $S_{0,\mathcal{R}} = \{(r, s_\mathcal{Q}, s_\mathcal{T}) : r = 1, (s_\mathcal{Q}, s_\mathcal{T}) \in S^{(1)}_{0,\mathcal{P}}(t)\}$ *is the set of initial states;*
- $F_\mathcal{R} = \{(r, s_\mathcal{Q}, s_\mathcal{T}) \in S_\mathcal{R} : q \in F\}$ *is the set of final accepting states, which remains unchanged since the NFA accepting states are fixed;*
- $A_\mathcal{R} = \bigcup_r A^{(r)}_\mathcal{P}(t) \cup \zeta(t) \cup \xi(t)$ *is the set of actions that include the updated switch transitions* $\zeta(t)$ *and the newly proposed synchronized transitions* $\xi(t)$.

Suppose $\texttt{ExePath}^{(r)}$ is the executed state sequence acquired from each agent r and $\texttt{ExePath}^{(r)}_\mathcal{Q}$ is the projected NFA state sequence. The definition of a synchronized transition is as follows:

Definition 5 (Synchronized transition)**.** *The set* $\xi \subset S_\mathcal{R} \times S_\mathcal{R}$ *denotes synchronized transitions. Each element* $\varepsilon = ((i, s_\mathcal{Q}, s_\mathcal{T}), (j, s'_\mathcal{Q}, s'_\mathcal{T}))$ *satisfies:*

- $i = j$*: connects the same agent;*
- $s_\mathcal{Q} = \texttt{ExePath}^{(r)}_\mathcal{Q}(\text{init}), s'_\mathcal{Q} = \texttt{ExePath}^{(r)}_\mathcal{Q}(\text{final})$, $r \in \{1, \ldots, N\}$ *starts from the initial NFA state and points to the most recent NFA nodes upon request for synchronization of each agent;*
- $s_\mathcal{T} = s'_\mathcal{T}$*: TS state is preserved.*

This synchronized transition allows a new transition between two NFA states inside each agent's $\mathcal{P}^{(r)}(t)$. Once this is complete, each agent will be aware of the task completion status of the whole team and avoid performing redundant tasks. In the original team automaton [28], the four properties including correctness, independence, completeness, and ordered sequence are proposed to justify the rationale of finding a global path on the team automaton for a task allocation. Here we claim that our synchronized team automaton preserves these properties, so that a new global action sequence β can be found by applying the same search algorithm performed during the offline phase (as presented in Sec. II). This process leads to a global task reallocation that assigns new sub-tasks to all agents.

V. Preliminary Results

A. Experiment set-up for simulation and hardware

To evaluate the feasibility and robustness of the proposed multi-robot task allocation and planning framework, we first establish a simulation of a hospital environment in Gazebo [29] and create a topological map for defining the TS, as shown in Fig. 4. The simulation architecture is composed of a high-level LTL planning layer based on a ROS package from [30], a mid-level execution interface using BehaviorTree.CPP [31], and a low-level navigation and controller layer using ROS navigation stack and appropriate controllers for each robot model.

B. Case study

We evaluate our framework on a heterogeneous team of robots consisting of a delivery robot DR, a walk training robot Wassi, and a quadrupedal robot A1 with both capabilities. A delivery robot consists of two simple operating states (*Loaded, Standby*), where the *Standby* state is equivalent

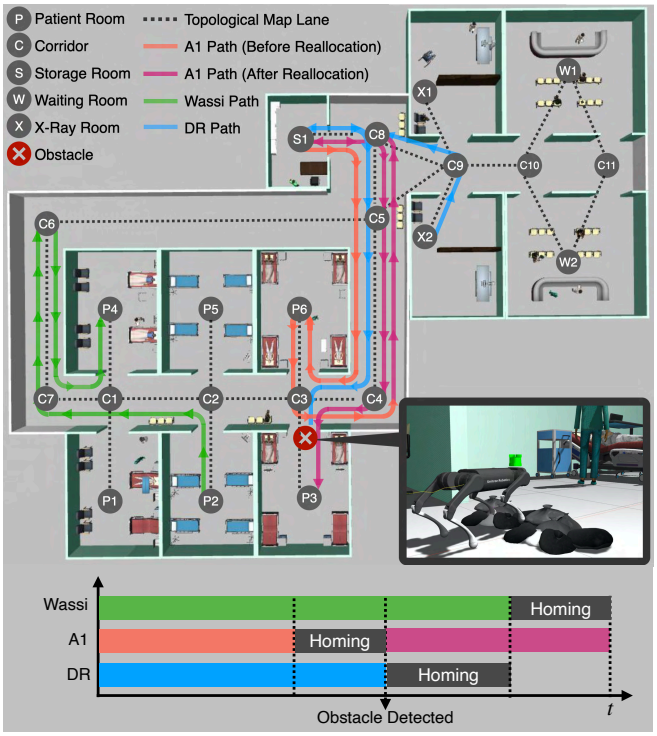

Fig. 4: Topological map of a hospital environment consisting of locations that are explicitly defined in each robot's transition system. The execution history for each robot is depicted for achieving global task ϕ under a scenario where an unexpected garbage heap is detected. The timeline for task execution and reallocation is illustrated at the bottom.

to *Unloaded State*. Likewise, a training robot consists of 4 operating states (*Standby, Camera On, User Located, Training*), where the robot visually locates and helps seniors who need walking training assistance. These operating states are encoded into TS for each type of robot.

We conduct a series of case studies in a hospital environment simulation to evaluate the reactive strategies proposed in Sec. IV. The global mission is defined as:

Scenario 1. *"Deliver medicines to locations p3 and p6; meet a patient at c1; complete a walk training along the corridor between c1 and c6, and then send the patient back to p4."*

$$\begin{aligned}\phi_1 =& \Diamond(p3 \wedge Standby) \wedge \Box\left((\neg p3 \wedge \bigcirc p3) \rightarrow Loaded\right)\\ &\wedge \Diamond(p6 \wedge Standby) \wedge \Box\left((\neg p6 \wedge \bigcirc p6) \rightarrow Loaded\right)\\ &\wedge \Diamond(p4 \wedge Standby) \wedge \Box\left((\neg p4 \wedge \bigcirc p4) \rightarrow Training\right)\\ &\wedge \Diamond(c1 \wedge Training \wedge \Diamond(c7 \wedge Training \wedge \Diamond(c6 \wedge Tr\\ &\text{-}aining \wedge \Diamond(c7 \wedge Training \wedge \Diamond(c1 \wedge Training)))))\end{aligned}$$

Three robots, A1, DR and Wassi, are placed at various locations in the hospital before the start of the simulation. Next, the task planner decomposes the specification and assigns sub-tasks to each robot offline, which takes 21 seconds in total. 92% of the computation time is taken by the generation of PA for each robot, which only needs to be performed once.

During the simulation, each robot will encounter a disturbance. Since the integration with exteroceptive systems is out of this paper's scope, all the disturbances except

the locomotion failure are triggered by manually setting the checkers in BT to be false. A diagram of this simulation is displayed in Fig. 4.

1) External force is applied to A1 and induces a loss of balance. A handcrafted whole-body recovery stand trajectory is tracked by a PD controller to assist A1 to resume its task.

2) A critical failure is induced for Wassi when performing the walk training task from $c7$ to $c6$. A global reallocation strategy assigns A1 to complete its delivery task first, and then proceeds to finish Wassi's incomplete walk training task.

3) The *Loaded* state of DR suddenly becomes a *Standby* state. This simulates a situation in which the robot unexpectedly loses its cargo. A local reallocation succeeds in instructing the robot to return to $s1$ and pick up another cargo. Then DR is instructed to complete the original delivery task.

4) A garbage heap is placed in front of $p3$ to simulate an environmental change. This obstruction can only be traversed by the legged robot A1. The routes taken by each robot and the timeline for obstacle detection and task allocation are portrayed in Fig. 4. While performing its task, DR encounters the obstruction and fails to find an alternate plan via local reallocation. While A1 is returning home after completing the delivery task to $p6$, it is assigned to take over DR's incomplete delivery task at $c4$. As a result, A1 goes to $s1$ to retrieve the object for delivery, and completes the task by traveling to $p3$.

VI. Conclusion and Future Work

In this work, we present a heterogeneous, multi-robot task allocation and planning framework equipped with a hierarchically reactive mechanism from extensive disturbances. A local and global task reallocation is performed at the high level where an LTL-based team automaton is generated to follow a formal guarantee. Future works include the implementation of the proposed reactive locomotion planner given terrain information, and the integration to the higher level task planner.

References

[1] D. Halperin, J.-C. Latombe, and R. H. Wilson, "A general framework for assembly planning: The motion space approach," *Algorithmica*, vol. 26, no. 3, pp. 577–601, 2000.

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

[3] J. S. Jennings, G. Whelan, and W. F. Evans, "Cooperative search and rescue with a team of mobile robots," in *International Conference on Advanced Robotics. Proceedings.* IEEE, 1997, pp. 193–200.

[4] Y. Shoukry, P. Nuzzo, A. Balkan, I. Saha, A. L. Sangiovanni-Vincentelli, S. A. Seshia, G. J. Pappas, and P. Tabuada, "Linear temporal logic motion planning for teams of underactuated robots using satisfiability modulo convex programming," in *2017 IEEE 56th annual conference on decision and control (CDC)*. IEEE, 2017, pp. 1132–1137.

[5] A. Pnueli, "The temporal logic of programs," in *Annual Symposium on Foundations of Computer Science*. IEEE, 1977, pp. 46–57.

[6] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas, "Temporal-logic-based reactive mission and motion planning," *IEEE transactions on robotics*, vol. 25, no. 6, pp. 1370–1381, 2009.

[7] J. A. DeCastro, J. Alonso-Mora, V. Raman, D. Rus, and H. Kress-Gazit, "Collision-free reactive mission and motion planning for multi-robot systems," in *Robotics research*. Springer, 2018, pp. 459–476.

[8] J. Warnke, A. Shamsah, Y. Li, and Y. Zhao, "Towards safe locomotion navigation in partially observable environments with uneven terrain," in *IEEE Conference on Decision and Control*, 2020, pp. 958–965.

[9] Y. Zhao, Y. Li, L. Sentis, U. Topcu, and J. Liu, "Reactive task and motion planning for robust whole-body dynamic locomotion in constrained environments," *The International Journal of Robotics Research*, p. 02783649221077714, 2022.

[10] S. Kulgod, W. Chen, J. Huang, Y. Zhao, and N. Atanasov, "Temporal logic guided locomotion planning and control in cluttered environments," in *American Control Conference*. IEEE, 2020, pp. 5425–5432.

[11] M. E. Cao, J. Warnke, Y. Han, X. Ni, Y. Zhao, and S. Coogan, "Leveraging heterogeneous capabilities in multi-agent systems for environmental conflict resolution," *arXiv preprint arXiv:2206.01833*, 2022.

[12] M. Guo and D. V. Dimarogonas, "Multi-agent plan reconfiguration under local ltl specifications," *The International Journal of Robotics Research*, vol. 34, no. 2, pp. 218–235, 2015.

[13] J. Tumova and D. V. Dimarogonas, "Multi-agent planning under local ltl specifications and event-based synchronization," *Automatica*, vol. 70, pp. 239–248, 2016.

[14] Y. Kantaros and M. M. Zavlanos, "Stylus*: A temporal logic optimal control synthesis algorithm for large-scale multi-robot systems," *The International Journal of Robotics Research*, vol. 39, no. 7, pp. 812–836, 2020.

[15] C. Banks, S. Wilson, S. Coogan, and M. Egerstedt, "Multi-agent task allocation using cross-entropy temporal logic optimization," in *IEEE International Conference on Robotics and Automation*. IEEE, 2020, pp. 7712–7718.

[16] A. Kolling, P. Walker, N. Chakraborty, K. Sycara, and M. Lewis, "Human interaction with robot swarms: A survey," *IEEE Transactions on Human-Machine Systems*, vol. 46, no. 1, pp. 9–26, 2015.

[17] Y. Chen, X. C. Ding, A. Stefanescu, and C. Belta, "Formal approach to the deployment of distributed robotic teams," *IEEE Transactions on Robotics*, vol. 28, no. 1, pp. 158–171, 2011.

[18] P. Schillinger, M. Bürger, and D. V. Dimarogonas, "Decomposition of finite ltl specifications for efficient multi-agent planning," in *Distributed Autonomous Robotic Systems*. Springer, 2018, pp. 253–267.

[19] P. Schillinger, M. Bürger, and D. Dimarogonas, "Multi-objective search for optimal multi-robot planning with finite ltl specifications and resource constraints," in *IEEE International Conference on Robotics and Automation*. IEEE, 2017, pp. 768–774.

[20] K. Leahy, A. Jones, and C.-I. Vasile, "Fast decomposition of temporal logic specifications for heterogeneous teams," *arXiv preprint arXiv:2010.00030*, 2020.

[21] X. Luo and M. M. Zavlanos, "Temporal logic task allocation in heterogeneous multi-robot systems," *arXiv preprint arXiv:2101.05694*, 2021.

[22] M. Guo, K. H. Johansson, and D. V. Dimarogonas, "Revising motion planning under linear temporal logic specifications in partially known workspaces," in *IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 5025–5032.

[23] M. Guo and D. V. Dimarogonas, "Reconfiguration in motion planning of single- and multi-agent systems under infeasible local ltl specifications," in *52nd IEEE Conference on Decision and Control*. IEEE, 2013, pp. 2758–2763.

[24] S. C. Livingston, R. M. Murray, and J. W. Burdick, "Backtracking temporal logic synthesis for uncertain environments," in *IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 5163–5170.

[25] S. Li, D. Park, Y. Sung, J. A. Shah, and N. Roy, "Reactive task and motion planning under temporal logic specifications," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 12 618–12 624.

[26] J. Tumova, A. Marzinotto, D. V. Dimarogonas, and D. Kragic, "Maximally satisfying ltl action planning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 1503–1510.

[27] P. Biswal and P. K. Mohanty, "Development of quadruped walking robots: a review," *Ain Shams Engineering Journal*, 2020.

[28] P. Schillinger, M. Bürger, and D. V. Dimarogonas, "Simultaneous task allocation and planning for temporal logic goals in heterogeneous multi-robot systems," *The international journal of robotics research*, vol. 37, no. 7, pp. 818–838, 2018.

[29] OpenRobotics. (May) Hospital. Open Robotics. [Online]. Available: https://fuel.ignitionrobotics.org/1.0/OpenRobotics/fuel/collections/Hospital

[30] R. Baran, X. Tan, P. Varnai, P. Yu, S. Ahlberg, M. Guo, W. S. Cortez, and D. V. Dimarogonas, "A ros package for human-in-the-loop planning and control under linear temporal logic tasks," in *IEEE International Conference on Automation Science and Engineering*, 2021.

[31] D. Faconti and M. Colledanchise, "BehaviorTree.CPP," 2019. [Online]. Available: https://github.com/BehaviorTree/BehaviorTree.CPP