2023 62nd IEEE Conference on Decision and Control (CDC)
December 13-15, 2023. Marina Bay Sands, Singapore

Resilient Temporal Logic Planning in the Presence of Robot Failures

Samarth Kalluraya[1], George J. Pappas[2], Yiannis Kantaros[1]

Abstract— **Several task and motion planning algorithms have been proposed recently to design paths for mobile robot teams with collaborative high-level missions specified using formal languages, such as Linear Temporal Logic (LTL). However, the designed paths often lack reactivity to failures of robot capabilities (e.g., sensing, mobility, or manipulation) that can occur due to unanticipated events (e.g., human intervention or system malfunctioning) which in turn may compromise mission performance. To address this novel challenge, in this paper, we propose a new resilient mission planning algorithm for teams of heterogeneous robots with collaborative LTL missions. The robots are heterogeneous with respect to their capabilities while the mission requires applications of these skills at certain areas in the environment in a temporal/logical order. The proposed method designs paths that can adapt to unexpected failures of robot capabilities. This is accomplished by re-allocating sub-tasks to the robots based on their currently functioning skills while minimally disrupting the existing team motion plans. We provide experiments and theoretical guarantees demonstrating the efficiency and resiliency of the proposed algorithm.**

I. Introduction

Linear Temporal Logic (LTL) has been widely used in robot motion planning to define diverse missions beyond simple reach-avoid requirements [1], such as surveillance, cooperative manipulation, and delivery [2]–[4]. Planning algorithms with temporal logic missions have been proposed in [5]–[10], focusing on robot teams with known dynamics in known environments. Recent extensions of these works have addressed unknown and dynamic environments while assuming known system dynamics [11]–[15]. The latter assumption has been relaxed in [16]–[18] using machine learning methods. These works lack resiliency to robot failures since they assume that the robot's capabilities remain uncompromised during deployment. However, this assumption may be violated in practice due to various sources of uncertainty robots face, including inclement weather, human interventions, and component malfunctions.

In order to improve resiliency in mission planning, we propose a new resilient mission planning algorithm for teams of heterogeneous robots with collaborative missions expressed as LTL formulas. The robots are heterogeneous with respect to their capabilities which may include e.g., mobility, sensing, or manipulation while the LTL formula requires them to apply their capabilities at certain areas and/or objects. Given an LTL mission, the proposed algorithm designs resilient plans in the sense that they adapt to robot failures that include loss of capabilities (e.g., grasping) or complete removal of the robot (e.g., due to battery draining). The plans are defined as sequences of robot locations and actions. This is accomplished by re-allocating sub-tasks to the robots based on their currently functioning (if any) skills. Once the tasks are re-assigned, the previously designed paths are minimally revised to adapt to the capability failures. The proposed method aims to minimally disrupt the multi-robot behavior when failures occur, by minimizing the number of re-assignments and by avoiding re-planning for the whole team (if possible). The latter is particularly important as global task re-allocation and re-planning from scratch would be computationally prohibitive and impractical to perform at runtime, especially for large robot teams. We provide extensive experiments with teams of heterogeneous ground and aerial robots as well as theoretical analysis demonstrating the efficiency and resiliency of the proposed algorithm against multiple unexpected failures of robot capabilities.

Related works: Several task allocation methods have been proposed recently that assign either local LTL tasks [19], [20] or sub-tasks (i.e., atomic propositions) of a global collaborative LTL mission [21] to robots. In these works, task assignment is performed offline, i.e., before robot deployment, while robot failures during the mission are not considered. When robot failures occur, these works can be used online to globally re-allocate tasks to the robots. However, as discussed earlier, global task re-assignment at runtime is impractical. The closest works to ours are the ones proposed in [22]–[24]. Particularly, [22] builds a product automaton modeling the multi-robot state space, the specification space, as well as possible robot failures. Using this product system, control strategies that are reactive to failures can be extracted. Nevertheless, that work considers homogeneous robots while product-based methods lack scalability with respect to the number of robots, the size of the environment, task complexity, and number of failures. Conceptually similar approaches are proposed in [23], [24]. For instance, [23] considers robots with heterogeneous abilities where robots locally react to environmental and robot state changes. However, unlike our work, in case of failures of robot skills, [23] requires global task re-allocation and re-planning over a team automaton.

Contributions: *First*, we propose a resilient temporal logic mission planning algorithm for heterogeneous robot teams against robot failures. *Second*, the proposed algorithm aims to minimally disrupt the multi-robot behavior in case of failures as it avoids global re-assignment/re-planning. *Third*, we provide correctness, completeness, and optimality guarantees of the proposed method. *Fourth*, we provide experiments that demonstrate the efficiency of our algorithm.

[1]Authors are with the Department of Electrical and Systems Engineering, Washington University at St. Louis, St. Louis, MO, 63130, USA. k.samarth,ioannisk@wustl.edu, [2]Author is with the Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, PA, 19104, USA. pappasg@seas.upenn.edu. This work was supported by the ARL grant DCIST CRA W911NF-17-2-0181.

2023 62nd IEEE Conference on Decision and Control (CDC) | 979-8-3503-0124-3/23/$31.00 ©2023 IEEE | DOI: 10.1109/CDC49753.2023.10383968

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 14:29:33 UTC from IEEE Xplore. Restrictions apply.

II. Problem Definition

A. Modeling of Robots and Environment

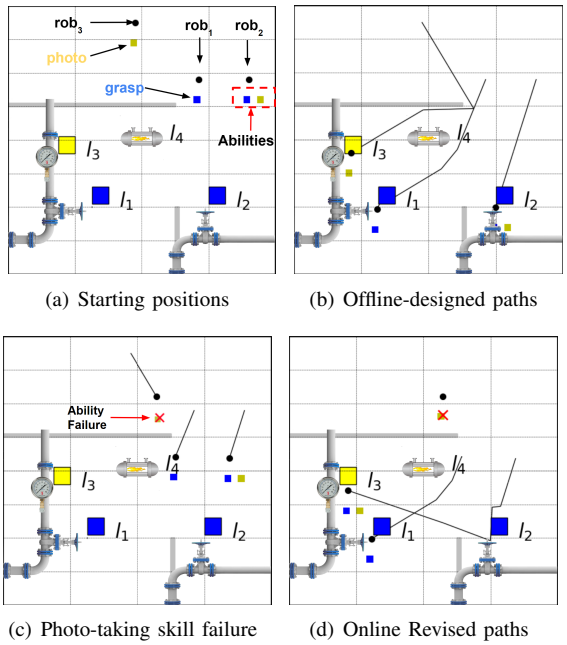
Consider a team of $N > 0$ mobile robots governed by the following dynamics: $\mathbf{p}_j(t+1) = \mathbf{f}_j(\mathbf{p}_j(t), \mathbf{u}_j(t))$, for all $j \in \mathcal{R} = \{1, \ldots, N\}$, where $\mathbf{p}_j(t) \in \mathbb{R}^n$ stands for the state (e.g., position and orientation) of robot j at discrete time t, and $\mathbf{u}_j(t) \in \mathbb{R}^b$ stands for control input. Hereafter, we compactly denote the dynamics of all robots as: $\mathbf{p}(t+1) = \mathbf{f}(\mathbf{p}(t), \mathbf{u}(t))$, where $\mathbf{p}(t) \in \mathbb{R}^{nN}$, $\forall t \geq 0$, and $\mathbf{u}(t) \in \mathbb{R}^{bN}$. We assume that the robot state $\mathbf{p}(t)$ is known for all time instants $t \geq 0$. The robots reside in a known environment $\Omega \subseteq \mathbb{R}^d$, $d \in \{2, 3\}$ with obstacle-free space denoted by $\Omega_{\text{free}} \subseteq \Omega$. We assume that Ω_{free} is populated with $M > 0$ regions/objects of interests, denoted by ℓ_e, with known locations $\mathbf{x}_e \in \Omega_{\text{free}}$, $e \in \{1, \ldots, M\}$.

B. Heterogeneous Robot Abilities and Robot Failures

We consider robots that are heterogeneous with respect to their skills. The robots have collectively $C > 0$ number of abilities amongst themselves. Each ability is represented by $c \in \{1, \ldots, C\}$. For instance, c can represent mobility, manipulation, fire extinguishing, or various sensing skills. We define the set $\mathcal{C} = [1, \ldots, c, \ldots, C]$ collecting all robot capabilities. The skills of robot j are represented by a $\mathbf{Z}_j(t) = [\zeta_1^j(t), \ldots, \zeta_c^j(t), \ldots, \zeta_C^j(t)]$, where $\zeta_c^j(t)$ is either equal to 1 if robot j has the ability c at time t and 0 otherwise. Observe that the set of skills for the robots depends on time. A failure of capability c for robot j occurs at time t if $\zeta_c^j(t-1) = 1$ and $\zeta_c^j(t) = 0$. We model removals of robots by setting $\mathbf{Z}_j(t)$ to be a zero vector. The failures can occur at unknown time instants t but we assume that vectors $\mathbf{Z}_j(t)$ are known at each time $t \geq 0$, for all robots j. In other words, we assume that the robots are equipped with a health monitoring system that allows them to keep track of their active/inactive abilities. Also, we assume all-to-all communication. This ensures that the team is aware of any capability failure. For simplicity, we assume that robots cannot apply more than one skill at a time. Based on the individual robot abilities, we partition the multi-robot system into C teams. We define a robot team $\mathcal{T}_c(t)$ at time t as a set collecting the robots with $\zeta_c^j(t) = 1$ i.e., $\mathcal{T}_c(t) = \{j \in \mathcal{R} \mid \zeta_c^j(t) = 1\}$; a robot may belong to more than one team at a time.

C. Mission Specification

The goal of the robots is to accomplish a high-level long-horizon collaborative mission captured by a global Linear Temporal Logic (LTL) specification ϕ, that requires them to apply their capabilities at specific regions/objects ℓ_e in a temporal and logical fashion. LTL comprises a set of atomic propositions (i.e., Boolean variables), denoted by $\mathcal{AP}$, Boolean operators, (i.e., conjunction $\wedge$, and negation $\neg$), and two temporal operators, next $\bigcirc$ and until $\mathcal{U}$. LTL formulas over a set $\mathcal{AP}$ can be constructed based on the following grammar: $\phi ::= \text{true} \mid \pi \mid \phi_1 \wedge \phi_2 \mid \neg\phi \mid \bigcirc \phi \mid \phi_1 \,\mathcal{U}\, \phi_2$, where $\pi \in \mathcal{AP}$. For brevity, we abstain from presenting the derivations of other Boolean and temporal operators, e.g., *always* $\Box$, *eventually* $\Diamond$, *implication* $\Rightarrow$, which can be found in [1]. We consider LTL tasks constructed based on the following team-based atomic predicate:



(a) Starting positions (b) Offline-designed paths

(c) Photo-taking skill failure (d) Online Revised paths

Fig. 1. Example 2.1: The small squares below each robot indicate the abilities each robot possesses; the ability to operate valves and take photos are denoted by the blue and the yellow squares, respectively. The locations are represented by the larger squares, whose color indicates which ability needs to be used at that location as per ϕ.

$$\pi_{\mathcal{T}_c}(j, c, \ell_e) = \begin{cases} \text{true}, & \text{if } j \in \mathcal{T}_c \text{ applies } c \text{ at } \ell_e \\ \text{false}, & \text{otherwise} \end{cases} \tag{1}$$

The predicate in (1) is true when *any* robot in the team $\mathcal{T}_c$ applies the skill c (e.g., 'grasp') at the region/object ℓ_e. The robot that has been assigned with this sub-task/predicate is denoted by j in (1). We assume an initial assignment of robots j to predicates (1) is given and that the resulting mission ϕ is feasible; this is a common assumption in the majority of related works. Based on the definition of this predicate, we also have that $\neg\pi_{\mathcal{T}_c}(j, c, \ell_e)$ is satisfied if *none* of the robots in $\mathcal{T}_c$ applies the skill c at ℓ_e. In other words, if a predicate appears with a negation in front of it, then it concerns the whole team $\mathcal{T}_c$ and not a specific robot j; i.e., the definition of robot j in (1) is redundant and, therefore, replaced by the symbol $\varnothing$. We also assume that the robots have perfect skills, i.e., once robot j applies the skill c at ℓ_e, then (1) is satisfied. An example of a simple LTL mission is given in Examples 2.1-2.2. Given an LTL formula, a multi-robot path τ can be constructed using existing methods such as [5]. The path τ is defined as an infinite sequence of states i.e., $\tau = \tau(0), \ldots, \tau(t) \ldots$. In τ, each state $\tau(t)$ is defined as $\tau(t) = [\mathbf{p}(t), \mathbf{s}(t)]$, where $\mathbf{p}(t)$ is the multi-robot system state and $\mathbf{s}(t) = [s_1(t), \ldots, s_N(t)]$, $s_j(t) \in \mathcal{C}$. In other words, $s_j(t)$ determines which skill robot j should apply at time t; if robot j does not need to apply any skill at time t, then, with slight abuse of notation, we denote this by $s_j(t) = \varnothing$.

D. Problem Statement: Resilient Temporal Logic Planning

Consider a robot team tasked with completing a mission ϕ. As the robots execute a designed plan τ, certain robot skills may fail unexpectedly; see Ex. 2.1-Ex. 2.2. In this case, τ may no longer be feasible compromising mission performance. Our goal is to address the following problem:

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 14:29:33 UTC from IEEE Xplore. Restrictions apply.

Problem 1: Consider a mission ϕ defined over predicates (1), an initial assignment of these predicates to robots, and an offline generated plan τ satisfying ϕ. When failures of robot skills occur (possibly more than one at a time), design (i) an online task re-allocation method that re-assigns sub-tasks (i.e., atomic predicates) associated with the failed skills to the robots based on their current skills; (ii) revise τ to satisfy the LTL formula arising after task re-allocation.

Example 2.1 (Scenario I): Consider a group of $N = 3$ robots with simple dynamics of the form $\mathbf{p}(t+1) = \mathbf{p}(t) + \mathbf{u}(t)$. The abilities among the robots are denoted by c_1, c_2, and c_3 referring to mobility, grasping valves and turning them on, and photo-taking skills using a camera, respectively. The skill-based teams are defined as $\mathcal{T}_{c_1}(0) = \{1, 2, 3\}$, $\mathcal{T}_{c_2}(0) = \{1, 2\}$, and $\mathcal{T}_{c_3}(0) = \{2, 3\}$. The robots are responsible for accomplishing a pipeline inspection task modeled by the following LTL formula: $\phi = \Diamond(\pi_1 \wedge \Diamond(\pi_2)) \wedge \Diamond\pi_3 \wedge \Box\neg\pi_4$ where $\pi_1 = \pi_{\mathcal{T}_2}(1, c_2, \ell_1)$, $\pi_2 = \pi_{\mathcal{T}_3}(3, c_3, \ell_3)$, $\pi_3 = \pi_{\mathcal{T}_2}(2, c_2, \ell_2)$ (1), and $\pi_4 = \pi_{\mathcal{R}}(\varnothing, c_1, \ell_4)$. In words, it requires robot 1, to turn on the valve at location ℓ_1 of a pipeline (modeled by π_1), and, subsequently, robot 3 to take a photo of the flow gauge there (modeled by π_2). Eventually, robot 2 must turn on a valve at another location ℓ_2 (modeled by π_3). At all times all robots must avoid moving close to ℓ_4 (c_1) as there is a boiler present there, with high temperatures unfavorable for the robots (modeled by π_4). The initial locations and skills of the robots are shown in Fig. 1(a). A feasible plan τ is shown in Figure 1(b). As the robots execute these plans, robot capabilities may fail. Specifically, Fig. 1(c) shows a case where the camera of robot 3 failed at $t = 3$, i.e., $\zeta^3_{c_3}(2) = 1$, $\zeta^3_{c_3}(3) = 0$, and $\mathcal{T}_{c_3}(3) = \{2\}$. Due to this failure, π_2, which was originally assigned to robot 3, cannot be satisfied. As a result, it has to be re-assigned to another robot in team $\mathcal{T}_{c_3}(3)$. The proposed algorithm assigns π_2 to robot 2. Paths satisfying the mission after this re-assignment are shown in Fig. 1(d). This example is re-visited in Sec. IV.

Example 2.2 (Scenario II): Consider a team of $N = 5$ robots with skills c_1, c_2, c_3, c_4, and c_5. The skills $c_1 - c_3$ are defined as above. The remaining ones refer to picking up objects and temperature sensors. The skill-based teams are $\mathcal{T}_{c_1}(0) = \{1, 2, 3, 4, 5\}$, $\mathcal{T}_{c_2}(0) = \{1, 4\}$, $\mathcal{T}_{c_3}(0) = \{1, 2, 4, 5\}$, $\mathcal{T}_{c_4}(0) = \{3\}$, and $\mathcal{T}_{c_5}(0) = \{4, 5\}$. Consider the LTL mission: $\phi = \Diamond(\pi_1 \wedge \Diamond((\pi_2 || \pi_3) \wedge \pi_5 \wedge \Diamond(\pi_1))) \wedge \Box\neg\pi_4$, where $\pi_1 = \pi_{\mathcal{T}_2}(1, c_2, \ell_1)$, $\pi_2 = \pi_{\mathcal{T}_3}(2, c_3, \ell_2)$, $\pi_3 = \pi_{\mathcal{T}_4}(3, c_4, \ell_3)$, $\pi_4 = \pi_{\mathcal{T}_2}(\varnothing, c_1, \ell_2)$, and $\pi_5 = \pi_{\mathcal{T}_5}(5, c_5, \ell_5)$. If skill c_3 of robot 2 and c_4 of robot 3 fails at $t = 1$, then a robot $i \in \mathcal{T}_{c_3}(1) = \{1, 4, 5\}$ needs to take over π_2. Observe that although robot 4 is not assigned any predicate, it cannot be assigned π_2 as satisfaction of π_2 by 4 would result in satisfaction of π_4 which would violate ϕ. Robot 1 also has the same constraint. As a result, only 5 can take over π_2. However, note that robot 5 needs to complete π_5 at the same time π_2 needs to be satisfied. Also observe that $\mathcal{T}_{c_4}(1) = \varnothing$, and thus π_3 cannot be satisfied. Our proposed algorithm is designed to handle such challenging scenarios; see Sec. III. This example is re-visited in Sec. III-IV.

Remark 2.3 (Independence of Sub-tasks): Assume that some skills of robot j fail. We assume that all atomic predicates associated with j and its failed skills that appear in ϕ (irrespective of the locations ℓ_e) can be re-allocated independently from each other, i.e., they do need to be assigned to the same robot $i \neq j$. For example, in Ex. 2.2 observe that the atomic proposition π_1 appears twice in ϕ. If c_2 fails for robot 1, then the 'leftmost' and the 'rightmost' proposition π_1 do not have to be satisfied by the same robot.

III. Resilient Temporal Logic Planning Against Robot Skill Failures

In this section we present an algorithm to solve Problem 1. Our solution consists of three parts. In the first part, we generate offline a path τ satisfying ϕ using our recently proposed sampling-based planner [5]; see Sections III-A-III-B. In the second part, we design an online local task re-allocation algorithm re-assigning sub-tasks to robots. The proposed algorithm is local in the sense that it aims to minimize the total number of sub-task re-allocations that have to happen due to the failed skill; see Sections III-C-III-E. Given the revised LTL formula, we propose an online re-planning method that locally revises the current robot paths to design new feasible ones.

A. From LTL Missions to Automata

Given an LTL mission ϕ, we translate it, offline, into a Nondeterministic Büchi Automaton (NBA) [1].

Definition 3.1 (NBA): A Nondeterministic Büchi Automaton (NBA) B over $\Sigma = 2^{\mathcal{AP}}$ is defined as a tuple $B = \left(\mathcal{Q}_B, \mathcal{Q}^0_B, \Sigma, \delta_B, \mathcal{Q}^F_B\right)$, where $\mathcal{Q}_B$ is the set of states, $\mathcal{Q}^0_B \subseteq \mathcal{Q}_B$ is a set of initial states, Σ is an alphabet, $\delta_B : \mathcal{Q}_B \times \Sigma \to 2^{\mathcal{Q}_B}$ is a non-deterministic transition relation, and $\mathcal{Q}^F_B \in \mathcal{Q}_B$ is a set of accepting/final states.

Next, we discuss the accepting condition of the NBA that is used to find plans τ that satisfy ϕ. We define a labeling function $L : \mathbb{R}^N \times \mathcal{C}^N \to 2^{\mathcal{AP}}$ determining which atomic propositions are true given the multi-robot state $\mathbf{p}(t)$ and the applied skills $\mathbf{s}(t)$. An infinite run $\rho_B = q^1_B \dots q^k_B \dots$ of B over an infinite word $w = \sigma_0\sigma_1\sigma_2 \dots \in \Sigma^\omega$, where $\sigma_k \in \Sigma$, $\forall k \in \mathbb{N}$, is an infinite sequence of NBA states q^k_B, $\forall k \in \mathbb{N}$, such that $q^{k+1}_B \in \delta_B(q^k_B, \sigma_k)$ and $q^0_B \in \mathcal{Q}^0_B$. An infinite run ρ_B is called *accepting* if $\text{Inf}(\rho_B) \bigcap \mathcal{Q}^F_B \neq \emptyset$, where $\text{Inf}(\rho_B)$ represents the set of states that appear in ρ_B infinitely often. A plan $\tau = [\mathbf{p}(0), \mathbf{s}(0)], [\mathbf{p}(1), \mathbf{s}(1)], \dots, [\mathbf{p}(t), \mathbf{s}(t)], \dots$ is feasible if the word $w = \sigma_0\sigma_1 \dots \sigma_t \dots$ where $\sigma_t = L([\mathbf{p}(t), \mathbf{s}(t)])$, results in at least one accepting run ρ_B. Since we assume that robots cannot apply more than one skill at a time, in what follows, we assume that the NBA is pruned as in [5] by removing transitions that violate this assumption.

B. Sampling-based planner

Given an LTL formula ϕ, we design feasible plans τ using the sampling-based planner developed in [5] due to its abstraction-free and scalability benefits. Specifically, [5] proposes a probabilistically complete and asymptotically optimal planner that incrementally builds trees that explore both the robot motion space and the automaton state-space. Nevertheless, any other temporal logic planner can be used.

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 14:29:33 UTC from IEEE Xplore. Restrictions apply.

C. Setting Up the Task Reallocation Process

Assume that a feasible plan τ has been generated as in Section III-B. Consider the case where, as the robots execute the plan τ, a sub-set of robot capabilities fail at an a priori unknown time t resulting in new vectors $\mathbf{Z}_j(t)$ for some robots $j \in \mathcal{R}$. Let $\mathcal{AP}_F \subseteq \mathcal{AP}$ be a set collecting all atomic predicates $\pi_{\mathcal{T}_c}(j, c, \ell_e)$ in ϕ that can never be true due to the failed skills, given the current assignment of predicates to robots.[1] With slight abuse of terminology, we refer to $\pi_{\mathcal{T}_c}(j, c, \ell_e) \in \mathcal{AP}_F$ as a 'failed' atomic predicate. The following process is repeated for all failed atomic predicates sequentially and in any order (line 1, Alg. 1).

Our goal is to re-assign $\pi_{\mathcal{T}_c}(j, c, \ell_e) \in \mathcal{AP}_F$ to a new robot $i \in \mathcal{T}_c$; again, with slight abuse of terminology, we also refer to this process as 'fixing/repairing' the failed predicate. Informally, the key challenge here is that possibly all robots $i \in \mathcal{T}_c$ may be busy with other sub-tasks; see Ex. 2.2. In this case, repairing the failed predicate may trigger a sequence of task re-assignments. The reason is that fixing the failed predicate requires another robot $i \in \mathcal{T}_c$ to take over it while, at the same time, another robot may have to take over the sub-task that was originally assigned to robot i and so on until there are no unassigned sub-tasks. Our goal is to fix the failed predicate by making the minimum possible number of task re-assignments to minimally disrupt mission planning.

Assume that failures occur at time t. Let $q_B^{\text{cur}} = q_B(t)$ be the NBA state that the robots have reached after executing the first t steps of their plan τ. Given a failed predicate $\pi_{\mathcal{T}_c}(j, c, \ell_e)$, we compute all NBA states that can be reached from q_B^{cur} through a multi-hop path. This step can be implemented by treating the NBA as a directed graph and checking which states $q'_B \in \mathcal{Q}_B$ can be reached from q_B^{cur}. We collect these states (including q_B^{cur}) in a set $\hat{\mathcal{Q}}_B^{\text{cur}} \subseteq \mathcal{Q}_B$. Then, among all NBA transitions between states $q'_B, q''_B \in \hat{\mathcal{Q}}_B^{\text{cur}}$, we collect in a set $\mathcal{E}$ the ones for which $\pi_{\mathcal{T}_c}(j, c, \ell_e)$ appears in the corresponding Boolean formula $b_{q'_B, q''_B}$ (line 2, Alg. 1). With slight abuse of notation, let $e = (q'_B, q''_B)$ denote an edge in $\mathcal{E}$ (i.e., an NBA transition from q'_B to q''_B). Given the edge $e \in \mathcal{E}$, we define the Boolean formula $b_{q'_B, q''_B}$ defined over $\mathcal{AP}$ for which it holds that if $q''_B \in \delta_B(q'_B, \sigma)$, for some $\sigma \in \Sigma$, then $\sigma \models b_{q'_B, q''_B}$ and vice versa. Such Boolean formulas can be constructed automatically using existing tools such as [25].

D. Constraints for Task Reallocation

The key idea in our approach is to inspect all edges $e \in \mathcal{E}$ and re-assign the predicates appearing in the respective formulas $b_{q'_B, q''_B}$ so that there is no unassigned predicate and that the resulting LTL formula remains feasible (i.e., the robots can still generate paths that yield accepting NBA runs). A necessary condition to preserve the feasibility of the LTL formula after task re-allocation is that $b_{q'_B, q''_B}$ should be feasible (i.e., it can become 'true') after task reallocation, $\forall e = (q'_B, q''_B) \in \mathcal{E}$; see also Rem. 3.8. In other words, there should exist a symbol $\sigma = L([\mathbf{p}, \mathbf{s}]) \in \Sigma$ generated by the robots, that satisfies the revised formulas $b_{q'_B, q''_B}$ arising after task re-assignment.

Next, we will translate this condition into constraints that should be respected when a failed predicate is repaired in an edge $e = (q'_B, q''_B)$ (where q'_B may coincide with q''_B). To this end, we re-write the Boolean $b_{q'_B, q''_B}$ in a disjunctive normal form (DNF), i.e., $b_{q'_B, q''_B} = \bigvee_{d=1}^{D} b^d_{q'_B, q''_B}$, for some $D > 0$.[2] For each Boolean formula $b^d_{q'_B, q''_B}$, we define the set $\mathcal{R}^d_{q'_B, q''_B} \subseteq \mathcal{R}$ that collects robots that appear in $b^d_{q'_B, q''_B}$; see also Ex. 3.4. Given $b^d_{q'_B, q''_B}$, we also define the set $\mathcal{AP}_i$ that collects all predicates of the form (1) that appear in $b^d_{q'_B, q''_B}$ assuming that the ones that are associated with skills c for which $\zeta^i_c(t) = 1$ are all assigned to robot i. Given $\mathcal{AP}_i$, we can construct the alphabet $\Sigma_i = 2^{\mathcal{AP}_i}$. For instance, if $b^d_{q'_B, q''_B} = \pi_{\mathcal{T}_c}(j, c, \ell_e) \wedge \pi_{\mathcal{T}_m}(i, m, \ell_f)$, then $\mathcal{AP}_i = \{\pi_{\mathcal{T}_c}(i, c, \ell_e), \pi_{\mathcal{T}_m}(i, m, \ell_f)\}$ if $i \in \mathcal{T}_m \cap \mathcal{T}_c$ and $\Sigma_i = \{\pi_{\mathcal{T}_c}(i, c, \ell_e), \pi_{\mathcal{T}_m}(i, m, \ell_f), \pi_{\mathcal{T}_c}(i, c, \ell_e)\pi_{\mathcal{T}_m}(i, m, \ell_f), \epsilon\}$, where ϵ stands for the empty symbol. Then, we define following function modeling task re-allocation constraints.

Definition 3.2 (Function $V^d_{q'_B, q''_B}$): The set-valued function $V^d_{q'_B, q''_B} : \mathcal{R} \to \Sigma_i$, given as input a robot index $i \in \mathcal{R}$, returns a set collecting all symbols $\sigma_i \in \Sigma_i$ that if robot $i \in \mathcal{R}$ generates, then $b^d_{q'_B, q''_B}$ will be 'false' regardless of the values of the other predicates. We define $V^d_{q'_B, q''_B}(i) = \emptyset$ for all robots $i \in \mathcal{R} \setminus \mathcal{R}^d_{q'_B, q''_B}$.

The sets $V^d_{q'_B, q''_B}(i)$ capture *constraints* on what tasks/predicates a robot i can take over. Specifically, when re-assigning predicates appearing in an edge $e = (q'_B, q''_B)$ (where q'_B may coincide with q''_B) robot $i \in \mathcal{R}^d_{q'_B, q''_B}$ cannot take over predicates that appear in $V^d_{q'_B, q''_B;}(i)$ as this would falsify $b^d_{q'_B, q''_B}$. Next, we define the function $g^d_{q'_B, q''_B}$ that will be used later to determine which robots are currently busy with other sub-tasks.

Definition 3.3 (Function $g^d_{q'_B, q''_B}$): The function $g^d_{q'_B, q''_B} : \mathcal{R} \to \mathcal{AP}$, given as an input a robot index $i \in \mathcal{R}$, returns a set collecting the atomic predicates that are assigned to robot i in $b^d_{q'_B, q''_B}$ excluding the negated ones and the failed predicate.[3] We define $g(i) = \varnothing$, for all robots $i \notin \mathcal{R}^d_{q'_B, q''_B}$ and for all robots $i \in \mathcal{R}^d_{q'_B, q''_B}$ appearing in negated predicates or in the failed predicate.

Example 3.4 (Function $g^d_{q'_B, q''_B}$, $V^d_{q'_B, q''_B}$ and sets $\mathcal{R}^d_{q'_B, q''_B}$): Consider the LTL formula given in Example 2.2. We focus on an NBA transition with: $b^d_{q'_B, q''_B} = \pi_2 \wedge \pi_5 \wedge \neg\pi_4$, where recall $\pi_2 = \pi_{\mathcal{T}_3}(2, c_3, \ell_2)$, $\pi_5 = \pi_{\mathcal{T}_5}(5, c_5, \ell_5)$, and $\pi_4 = \pi_{\mathcal{T}_2}(\varnothing, c_1, \ell_2)$. Then, $\mathcal{R}^d_{q'_B, q''_B} = \{2, 5\} \cup \mathcal{T}_2$. Also, $g^d_{q'_B, q''_B}(i) = \varnothing$, for all robots $i \in \mathcal{R}^d_{q'_B, q''_B} \setminus \{2, 5\}$ and $g(2) = \pi_2$, $g(5) = \pi_5$. We also have $V^d_{q'_B, q''_B}(i) = \emptyset$ for all $i \notin \mathcal{T}_2$ and $V^d_{q'_B, q''_B}(i) = \{\pi_{\mathcal{T}_2}(i, c_1, \ell_2), \pi_{\mathcal{T}_3}(i, c_3, \ell_2), \pi_{\mathcal{T}_2}(i, c_1, \ell_2)\pi_{\mathcal{T}_3}(i, c_3, \ell_2)\}$

[1] Negated predicates of the form $\neg\pi_{\mathcal{T}_c}(\varnothing, c, \ell_e)$ are not included in $\mathcal{AP}_F$ as such predicates refer to the whole team $\mathcal{T}_c$ and not a specific robot. As a result, it is not meaningful to re-assign them.

[2] The failed predicate may appear in more than one sub-formula $b^d_{q'_B, q''_B}$ but it will be repaired separately/independently for each sub-formula; see Remark 2.3.

[3] There is at most one predicate assigned to a robot $i \in \mathcal{R}^d_{q'_B, q''_B}$ as all NBA transitions requiring a robot to satisfy more than one predicate at a time are pruned; see Section III-A.

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 14:29:33 UTC from IEEE Xplore. Restrictions apply.

Algorithm 1: Local Task Re-allocation

Input: (i) NBA $\mathcal{B}$, (ii) Current NBA state q_B^{cur}; (iii) Set of failed predicates $\mathcal{AP}_F$
Output: Revised NBA

1 **for** *every* $\pi \in \mathcal{AP}_F$ **do**
2 Define the ordered set of edges $\mathcal{E}$;
3 **for** *every* $e = (q'_B, q''_B) \in \mathcal{E}$ **do**
4 Rewrite: $b_{q'_B,q''_B} = \bigvee_{d=1}^{D} b^d_{q'_B,q''_B}$;
5 **for** $d = 1, \ldots, D$ **do**
6 Define $\mathcal{G}$ and functions $V^d_{q'_B,q''_B}, g^d_{q'_B,q''_B}$;
7 Apply Alg. 2 to compute a sequence of re-assignments $p = p(0), \ldots, p(P)$;
8 **if** $\exists p$ **then**
9 Re-assign atomic predicates as per p;
10 **else**
11 Assign $\pi =$ False;
12 Revise $b^d_{q'_B,q''_B}$;

for all $i \in \mathcal{T}_2 \cap \mathcal{T}_3$. Notice that $\pi_{\mathcal{T}_2}(i, c_1, \ell_2)$ and $\pi_{\mathcal{T}_2}(i, c_1, \ell_2)\pi_{\mathcal{T}_3}(i, c_3, \ell_2)$ are included because of $\neg\pi_{\mathcal{T}_2}(\varnothing, c_1, \ell_2)$ in $b^d_{q'_B,q''_B}$. Also, $\pi_{\mathcal{T}_3}(i, c_3, \ell_2)$ is included because if robot 1 or 4 satisfies it, then they will be close to location ℓ_2; therefore, $\pi_{\mathcal{T}_2}(i, c_1, \ell_2)$ will also be satisfied resulting in violation of $b^d_{q'_B,q''_B}$.

E. Local Task Reallocation Algorithm

In this section, we present the proposed task re-allocation algorithm. Once the set $\mathcal{E}$ is constructed, we repeat the following steps for each edge $e \in \mathcal{E}$ in parallel (line 3, Alg. 1). Given e, we express its corresponding Boolean formula in a DNF form $b_{q'_B,q''_B} = \bigvee_{d=1}^{D} b^d_{q'_B,q''_B}$ (whether $q'_B = q''_B$ or not) (line 4, Alg. 1). Then we repeat the following steps in parallel for each sub-formula $b^d_{q'_B,q''_B}$ (line 5, Alg. 1).

First, we define the following directed graph capturing all possible reassignments in $b^d_{q'_B,q''_B}$ (even ones that violate constraints discussed in Section III-D). Then, we will search over it to find re-assignments of the sub-tasks/predicates appearing in $b^d_{q'_B,q''_B}$ so that there are no unassigned predicates and all assignments respect the constraints captured by $V^d_{q'_B,q'_B}$. We denote this graph by $\mathcal{G} = \{\mathcal{V}_\mathcal{G}, \mathcal{E}_\mathcal{G}, w_\mathcal{G}\}$, where $\mathcal{V}_\mathcal{G}, \mathcal{E}_\mathcal{G}, w_\mathcal{G}$ denote the set of nodes, edges, and a cost function $w_\mathcal{G} : \mathcal{E}_\mathcal{G} \to \mathbb{R}$. The set of nodes is defined as $\mathcal{V}_\mathcal{G} = \mathcal{R}$. An edge from a node a to $a' \neq a$ exists if $a' \in \mathcal{T}_c$, where c is the skill required to satisfy the predicate $g^d_{q'_B,q''_B}(a)$. If $g^d_{q'_B,q''_B}(a) = \varnothing$, then there no outgoing edges from a. The physical meaning of an edge is that robot a' can take over the predicate (if any) assigned to a in $b^d_{q'_B,q''_B}$. The cost function $w_\mathcal{G}$ is defined so that each edge has a cost equal to 1. We emphasize that we do not explicitly construct this graph; instead, the task re-allocation algorithm only requires knowledge of the nodes $a' \in \mathcal{T}_c$ that can be reached in one hop from any node a (line 6, Alg. 1).

Then, we apply a constrained Breadth First Search (BFS) algorithm over $\mathcal{G}$ to re-assign sub-tasks (i.e., predicates appearing in $b^d_{q'_B,q''_B}$) to robots (line 7, Alg. 1). Our goal is to find a path in $\mathcal{G}$ from the robot associated with failed predicate, denoted by a_{root}, to any node a' satisfying $g^d_{q'_B,q''_B}(a') = \varnothing$ (i.e., a' is not assigned any task in $b^d_{q'_B,q''_B}$) while respecting the constraints discussed in Section III-D.

Algorithm 2: Breadth First Search

Input: (i) Failed predicate $\pi_{\mathcal{T}_c}(j, c, \ell_e)$, (ii) $V^d_{q'_B,q''_B}$, (iii) $g^d_{q'_B,q''_B}$, (iv) Teams $\mathcal{T}_c(t),\ \forall c \in \mathcal{C}$
Output: Path p

1 $a_{\text{root}} = j$;
2 $\mathcal{Q} = [a_{\text{root}}]$;
3 $\texttt{Flag}_{\text{root}}$ = True;
4 **while** $\sim$*empty*($\mathcal{Q}$) **do**
5 $a \leftarrow$ POP($\mathcal{Q}$);
6 **if** $a \in \mathcal{A}$ & $\sim \texttt{Flag}_{root}$ **then**
7 Using `Parent` function return path p ;
8 $\texttt{Flag}_{\text{root}}$=False;
9 **for** a' *adjacent to* a *in* $\mathcal{G}$ **do**
10 **if** $g^d_{q'_B,q''_B}(a) \notin V^d_{q'_B,q''_B}(a')$ & a' *not explored* **then**
11 Label a' as explored;
12 $\texttt{Parent}(a') = a$;
13 Append a' to $\mathcal{Q}$;
14 return ($\emptyset$)

We define the set $\mathcal{A} = \{a \in \mathcal{R} \mid g^d_{q'_B,q''_B}(a) = \varnothing\}$. Let $p = p(0), p(1), \ldots, p(P)$ denote such path over $\mathcal{G}$, where $p(0) = a_{\text{root}}$, $p(P) \in \mathcal{A}$ and $p(k) \notin \mathcal{A}$, for all $k \in \{2, \ldots, P-1\}$. Such a path will dictate the re-assignment of tasks required to fix the failed predicate. Specifically, the robot $p(k+1)$ takes over the sub-task of robot $p(k)$ in $b^d_{q'_B,q''_B}$ (i.e., the atomic predicate $g^d_{q'_B,q''_B}(p(k))$). Note that this means that the robot $p(k+1)$ gives up on its current sub-tasks (which will be taken over by the robot $p(k+2)$). Also, since this path should respect the constraints discussed in Section III-D, it must also hold that $g^d_{q'_B,q''_B}(p(k)) \notin V^d_{q'_B,q''_B}(p(k+1))$ for all $k \in \{2, \ldots, P-1\}$.

We apply a BFS algorithm to find the shortest path p summarized in Alg. 2; see Fig. 2. Notice that there are two key differences with the standard BFS algorithm. The first one concerns when nodes are added to the queue data structure $\mathcal{Q}$ of the BFS algorithm. Specifically, when a node a is popped/removed from $\mathcal{Q}$, then each adjacent node a' is added to $\mathcal{Q}$ if (1) it has not been explored yet (as in standard BFS); and (2) $g^d_{q'_B,q''_B}(a) \notin V^d_{q'_B,q''_B}(a')$ (line 10, Alg. 2); The first constraint prevents cases where a single robot will have to replace two robots as this may result in Boolean formulas $b^d_{q'_B,q''_B}$ that is satisfied if a robot applies more than one skill simultaneously which is unfeasible (see Section II). The second constraint ensures satisfaction of the constraints discussed in Section III-D. The second difference from standard BFS is that the root node is not initially labeled as 'explored'. This allows the robot with the failed skill (root) to take over other sub-tasks using its remaining active skills (if any). Finally, we note that the graph-search process is terminated when the first feasible path from a_{root} to any node in $\mathcal{A}$ is found (line 6-7, Alg. 2). The reason is that this path corresponds to the minimum possible re-assignments that can occur to fix the failed predicate in $b^d_{q'_B,q''_B}$; see Section III-G.

Once all failed predicates are fixed, the associated formulas $b_{q'_B,q''_B}$ are accordingly revised, yielding a new NBA (line 9, Alg. 1). If a failed predicate cannot be fixed (i.e., Alg. 2 cannot find a feasible path), then this failed predicate is replaced by the logical 'false' in all related formulas $b_{q'_B,q''_B}$ as there is no robot that can take over it (line 11, Alg. 1); note

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 14:29:33 UTC from IEEE Xplore. Restrictions apply.

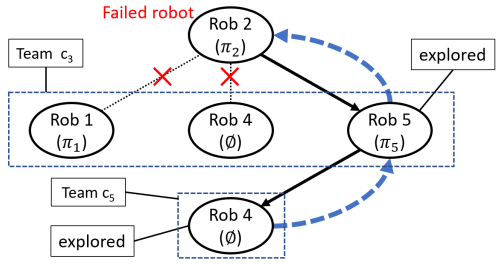


Fig. 2. Consider in Example 2.2 the case where skill c_3 of robot 2 fails, i.e., the failed predicate is π_2. We present the BFS tree (Alg. 2) built to fix π_2 for the NBA transition enabled by $b^d_{q'_B,q''_B} = \pi_1 \wedge \pi_2 \wedge \pi_5 \wedge \neg\pi_4$. The set $\mathcal{A}$ is defined as $\mathcal{A} = \{4\}$ and the root of the tree is robot 2. Robots 1, 4, 5 are adjacent to robot 2 in $\mathcal{G}$. Robots 1 and 4 are not connected to robot 2 because they do not satisfy $g^d_{q'_B,q''_B}(2) \notin V^d_{q'_B,q''_B}(a') = \{\pi_2, \pi_4, \pi_2\pi_4\}$, for $a' \in \{1, 4\}$. Robot 5 is connected to robot 2 and subsequently, robot 4 is connected to robot 5. The blue dashed arrows show the re-assignment process along the computed path p, i.e., robot 4 will take over π_5 and robot 5 will take over the failed predicate.

that this does not necessarily imply infeasibility of the LTL formula. This revised NBA is an input to an online planner that designs new paths; see Sec. III-F.

F. Online replanning

Assume certain robot capabilities fail at time t and that Alg. 1 has re-assigned tasks to robots. A straightforward solution to revise the robot paths is to apply the sampling-based planner [5] (see Section III-B) *globally* to build new plans for *all* robots from scratch. Nevertheless, global re-planning may be unnecessary given the 'local' task re-allocations made by Alg. 1 while it may be impractical for large robot teams. Inspired by [11], we propose an alternative re-planning approach that aims to *locally* revise only the robot paths affected by the failures using [5]. A detailed description can be found in [26].

G. Algorithm Analysis

In this section, we discuss correctness, completeness and optimality of the proposed algorithm. The proofs are omitted due to space limitations and can be found in [26].

Proposition 3.5 (Completeness/Optimality of Alg. 2): Consider a failed predicate $\pi \in \mathcal{AP}_F$ and Boolean formula $b^d_{q'_B,q''_B}$ that contains π. Alg. 2 is complete, i.e., if there exists a re-allocation of all predicates (i.e., path p) that appear in $b^d_{q'_B,q''_B}$ satisfying the constraints discussed in Section III-D, then Alg. 2 will find it. Algorithm 2 is also optimal, i.e., if there exist multiple feasible re-allocations it will find the re-allocation with the minimum number of re-assignments.

Proposition 3.6 (Completeness/Optimality of Alg. 1): Consider a failed predicate $\pi \in \mathcal{AP}_F$ and a set of NBA edges $\mathcal{E}$ depending on the π. If there exist feasible re-allocations (i.e., paths p) of all predicates for all Boolean formulas $b^d_{q'_B,q''_B}$ for a given edge $e \in \mathcal{E}$ and for all $e \in \mathcal{E}$, then Alg. 1 will find them. It will also find the solution with the minimum number of re-assignments across all sub-formulas and edges.

Proposition 3.7 (Soundness/Completeness of Replanning): Consider a re-allocation of predicates to robots by Alg. 1. If the re-planning algorithm (Section III-F) returns a new prefix-suffix plan, then this plan satisfies ϕ. Also, if there exists a feasible prefix-suffix plan satisfying the revised formula, then the re-planning algorithm will find it.

Remark 3.8 (Limitations): The constraints defined in Section III-D cannot ensure feasibility of the revised LTL formula. In fact, satisfying these constraints is necessary, but not sufficient, to ensure the existence of robot paths that can result in accepting NBA runs. Additional constraints can be introduced in the task re-allocation process, modeling necessary conditions to ensure that a transition is physically realizable; see e.g., Def. 3.4 in [12].

IV. Experimental Validation

In this section, we provide experiments to demonstrate the performance of the proposed algorithm in the presence of unexpected multiple failures. We conducted our experiments in Python 3 on a computer with Intel Core i7-8565U 1.80GHz and 16Gb RAM. Videos and additional case studies can be found in [26], [27].

A. Pipeline Inspection Task - Single Failure

We revisit the pipeline inspection task discussed in Ex. 2.1. The considered LTL mission corresponds to an NBA with 8 states. Due to failure of the camera for robot 3, π_2, that was originally assigned to robot 3, cannot be satisfied. At this point, Algorithm 1 is called to fix all NBA edges associated with π_2. The total number of affected edges is 6. In all edges, π_2 was assigned to robot 2. The time needed for this reassignment is 0.0009 secs and the new paths are generated in 0.26 secs. Observe in Fig. 1(d) robot 2 completed its original task of operating the valve at ℓ_2 and then took over the role of robot 3 by visiting ℓ_3 to take a photo. The results for the case study discussed in Ex. 2.2 can be found in [27].

B. Aerial sensing task - Multiple failures

We consider a mission involving $N = 7$ drones with dynamics as in [28]. The abilities of the drones are defined as c_1, c_2, c_3, and c_4 pertaining to mobility, data transmission capability, photo-taking skills using a camera, and infrared imaging, respectively. Drones 1, 2 and 3 have abilities c_1 and c_2. Drones 4 and 5 have abilities c_1, c_2, and c_3, and drones 6 and 7 have abilities c_1, c_3, c_4. The drones are responsible for accomplishing an aerial sensing task while transmitting the collected data to a base station.

Specifically, each of drones 1, 2, and 3 should eventually always be present at locations ℓ_1, ℓ_2, and ℓ_3, respectively, to be able to transmit data collected from other drones to a base station (being close to ℓ_1) in a multi-hop fashion. Each of the remaining drones needs to complete two tasks one after the other. Drones 4 and 5 have to take photos at two locations each, while drones 5 and 6 need to take infrared locations at two locations each. This mission is captured by the following formula: $\phi = \Diamond(\Box\xi_1 \wedge \pi_1 \wedge \Diamond(\pi_2)) \wedge \Diamond(\Box\xi_1 \wedge \pi_3 \wedge \Diamond(\pi_4)) \wedge \Diamond(\Box\xi_1 \wedge \pi_5 \wedge \Diamond(\pi_6)) \wedge \Diamond(\Box\xi_1 \wedge \pi_7 \wedge \Diamond(\pi_8))$ where ξ_1 is a Boolean formula modeling the transmission objective defined as $\xi_1 = \pi_{\mathcal{T}_2}(1, c_2, \ell_1) \wedge \pi_{\mathcal{T}_2}(2, c_2, \ell_2) \wedge \pi_{\mathcal{T}_2}(3, c_2, \ell_3)$, and $\pi_1 = \pi_{\mathcal{T}_3}(4, c_3, \ell_4)$, $\pi_2 = \pi_{\mathcal{T}_3}(4, c_3, \ell_5)$, $\pi_3 = \pi_{\mathcal{T}_3}(5, c_3, \ell_6)$, $\pi_4 = \pi_{\mathcal{T}_3}(5, c_3, \ell_7)$, $\pi_5 = \pi_{\mathcal{T}_4}(6, c_4, \ell_8)$, $\pi_6 = \pi_{\mathcal{T}_4}(6, c_4, \ell_9)$, $\pi_7 = \pi_{\mathcal{T}_4}(7, c_4, \ell_{10})$, $\pi_8 = \pi_{\mathcal{T}_4}(7, c_4, \ell_{11})$. This mission corresponds to an NBA with 132 states.

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 14:29:33 UTC from IEEE Xplore. Restrictions apply.

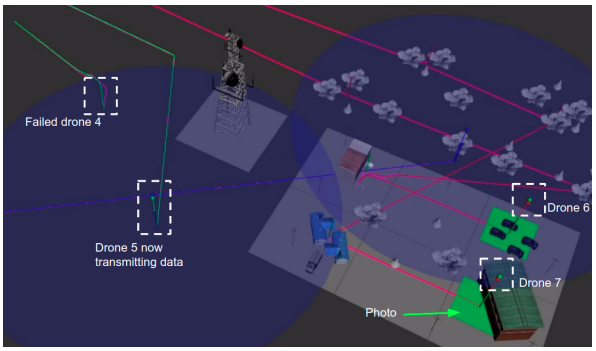


Fig. 3. Drone 5 performs a transmission task (blue circles denote transmission range) originally assigned to the failed drone 2. Drones 6 & 7 take photos completing tasks initially assigned to the failed drones 4 & 5.

We simulate the failure of all skills of two drones 2 and 4 at $t = 9$ (when the robots are currently in the initial NBA state), requiring reassignment of their associated atomic predicates ϕ. Due to this, $|\mathcal{E}| = 892$ NBA edges need to be repaired by Algorithm 1. Algorithm 1 reassigns drone 2's task of transmission to drone 5 and drone 4 and 5's original photographing tasks to drones 6 and 7, respectively. This reassignment process took 2.28 secs and planning the new paths took 1.1 secs. Toward the end of the mission, at $t = 58$, robot 7 failed completely as well. In this case, Algorithm 1 had to fix one failed predicate π_2 (which was originally assigned to robot 4 and later taken over by robot 7) and one edge. Alg. 1 reassigns this predicate to robot 6 in 0.0013 secs. We note here that since the total number of NBA states is large (132), a failure in the initial part of the mission (when the robots are in the initial NBA state) leads to a large set $\hat{\mathcal{Q}}_B^{\text{cur}}$ (see Section III-C) and, consequently to large set $\mathcal{E}$ of NBA edges that need to be repaired. As a result, the runtime required to fix the failed predicates in that case requires more time compared to failures toward the end of the mission where only a few edges needed to be fixed. A screenshot of the simulation is shown in Fig. 3.

V. Conclusion

This paper proposed a temporal logic mission planning algorithm for heterogeneous robot teams that is resilient against failures of robot capabilities. The proposed algorithm aims was supported theoretically and experimentally.

References

[1] C. Baier and J.-P. Katoen, *Principles of model checking*. MIT press Cambridge, 2008, vol. 26202649.

[2] D. Aksaray, K. Leahy, and C. Belta, "Distributed multi-agent persistent surveillance under temporal logic constraints," *IFAC-PapersOnLine*, vol. 48, no. 22, pp. 174–179, 2015.

[3] C. K. Verginis and D. V. Dimarogonas, "Control of cooperative manipulator-endowed systems under high-level tasks and uncertain dynamics," *Annual Reviews in Control*, 2022.

[4] A. S. Asarkaya, D. Aksaray, and Y. Yazıcıoğlu, "Temporal-logic-constrained hybrid reinforcement learning to perform optimal aerial monitoring with delivery drones," in *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2021, pp. 285–294.

[5] X. Luo, Y. Kantaros, and M. M. Zavlanos, "An abstraction-free method for multirobot temporal logic optimal control synthesis," *IEEE Transactions on Robotics*, 2021.

[6] 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.

[7] T. Wongpiromsarn, U. Topcu, and R. M. Murray, "Receding horizon temporal logic planning," *IEEE Transactions on Automatic Control*, vol. 57, no. 11, pp. 2817–2830, 2012.

[8] C. I. Vasile and C. Belta, "Sampling-based temporal logic path planning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Tokyo, Japan, November 2013, pp. 4817–4822.

[9] M. Kloetzer and C. Belta, "A fully automated framework for control of linear systems from temporal logic specifications," *IEEE Transactions on Automatic Control*, vol. 53, no. 1, pp. 287–297, 2008.

[10] D. Gujarathi and I. Saha, "Mt*: Multi-robot path planning for temporal logic specifications," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 13 692–13 699.

[11] 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 (ICRA)*, Karlsruhe, Germany, 2013, pp. 5025–5032.

[12] Y. Kantaros, M. Malencia, V. Kumar, and G. J. Pappas, "Reactive temporal logic planning for multiple robots in unknown environments," in *IEEE International Conference on Robotics and Automation (ICRA)*, Paris, France, June 2020, pp. 11 479–11 485.

[13] S. Kalluraya, G. J. Pappas, and Y. Kantaros, "Multi-robot mission planning in dynamic semantic environments," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

[14] M. Lahijanian, M. R. Maly, D. Fried, L. E. Kavraki, H. Kress-Gazit, and M. Y. Vardi, "Iterative temporal planning in uncertain environments with partial satisfaction guarantees," *IEEE Transactions on Robotics*, vol. 32, no. 3, pp. 583–599, 2016.

[15] Z. Li, M. Cai, S. Xiao, and Z. Kan, "Online motion planning with soft metric interval temporal logic in unknown dynamic environment," *IEEE Control Systems Letters*, vol. 6, pp. 2293–2298, 2022.

[16] M. Hasanbeig, Y. Kantaros, A. Abate, D. Kroening, G. J. Pappas, and I. Lee, "Reinforcement learning for temporal logic control synthesis with probabilistic satisfaction guarantees," in *IEEE Conference on Decision and Control (CDC)*, Nice, France, December 2019.

[17] M. Cai, M. Mann, Z. Serlin, K. Leahy, and C.-I. Vasile, "Learning minimally-violating continuous control for infeasible linear temporal logic specifications," in *American Control Conference*, 2023.

[18] X. Sun and Y. Shoukry, "Neurosymbolic motion and task planning for linear temporal logic tasks," *arXiv preprint arXiv:2210.05180*, 2022.

[19] P. Schillinger, M. Bürger, and D. Dimarogonas, "Decomposition of finite ltl specifications for efficient multi-agent planning," in *13th International Symposium on Distributed Autonomous Robotic Systems*, London, UK, November, 2016.

[20] 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 (ICRA)*, 2020, pp. 7712–7718.

[21] X. Luo and M. M. Zavlanos, "Temporal logic task allocation in heterogeneous multirobot systems," *IEEE Transactions on Robotics,*, vol. 38, no. 6, pp. 3602–3621, 2022.

[22] F. Huang, X. Yin, and S. Li, "Failure-robust multi-robot tasks planning under linear temporal logic specifications," in *13th Asian Control Conference (ASCC 2022)*. IEEE, 2022.

[23] Z. Zhou, D. J. Lee, Y. Yoshinaga, S. Balakirsky, D. Guo, and Y. Zhao, "Reactive task allocation and planning for quadrupedal and wheeled robot teaming," in *2022 IEEE 18th International Conference on Automation Science and Engineering (CASE)*, 2022, pp. 2110–2117.

[24] F. Faruq, D. Parker, B. Laccrda, and N. Hawes, "Simultaneous task allocation and planning under uncertainty," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 3559–3564.

[25] P. Gastin and D. Oddoux, "Fast ltl to büchi automata translation," in *International Conference on Computer Aided Verification*. Springer, 2001, pp. 53–65.

[26] S. Kalluraya, G. J. Pappas, and Y. Kantaros, "Resilient temporal logic planning in the presence of robot failures." https://arxiv.org/pdf/2305.05485.pdf.

[27] "Simulation videos," https://vimeo.com/813432282.

[28] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, *Robot Operating System (ROS): The Complete Reference (Volume 1)*. Cham: Springer International Publishing, 2016, ch. RotorS—A Modular Gazebo MAV Simulator Framework, pp. 595–625. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-26054-9_23

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 14:29:33 UTC from IEEE Xplore. Restrictions apply.