Robust Temporal Logic Task Planning for Multirobot Systems Under Permanent Robot Failures

Bohan Cui, *Student Member, IEEE*, Feifei Huang, *Student Member, IEEE*, Shaoyuan Li, *Senior Member, IEEE*, and Xiang Yin, *Member, IEEE*

***Abstract*— We investigate the multirobot task planning problem for intricate tasks specified by linear temporal logic (LTL) formulae. While most studies on this topic assume flawless robot performance, it is crucial to recognize that *failures* can always occur in the real world due to errors or disturbances. Therefore, to enhance the robustness of task planning for multirobot systems (MRSs), one must take the unexpected robot failures into account. In this article, we formulate and solve a new type of failure-aware multirobot task planning problem. Specifically, we aim to find a *failure-robust* plan that ensures the LTL task can always be accomplished, even if a maximum number of robots fail at any instant during the execution, where a failed robot can no longer contribute to the satisfaction of the LTL task. To achieve this, we extend the mixed-integer linear programming (MILP) approach to the failure-robust setting. To overcome the computational complexity, we identify a fragment of LTL formulae called the free-union-closed LTL, which allows for more scalable synthesis without considering the global combinatorial issue. We provide a systematic method to check this property, as well as several commonly used patterns as instances. We demonstrate the effectiveness of our approach through simulation and real-world experiments, showcasing our failure-robust plans and the efficiency of our simplified algorithm. Our approach offers an optimal and efficient way to achieve robustness in multirobot path planning under unforeseen failure events.**

***Index Terms*— Failure-awareness, linear temporal logic (LTL), multirobot systems (MRSs), task planning.**

I. Introduction

MULTIROBOT systems (MRSs) have been widely used in many different engineering cyber-physical systems such as autonomous warehouses [1], [2], [3], intelligent manufacturing systems [4], persistent surveillance [5], [6], and environment explorations [7], [8]. In the study of MRSs, one of the most fundamental problems is the *task planning* problem that seeks to generate executable paths for a team of heterogeneous or homogeneous robots that work collaboratively to achieve some specified global tasks. Many existing works have focused on finding paths for MRSs to enable simple reach-avoid navigations [9], [10], [11]. However, due to increasing demands for complex tasks, there has been a growing interest in the MRS literature for path planning for high-level specifications. This involves generating executable paths for each robot to accomplish tasks that require more advanced logic; see, e.g., [12], [13], [14], [15], [16], [17], [18], [19], [20].

Formal method provides a promising approach for automatically synthesizing high-level plans for MRS with correctness guarantees [21], [22]. In particular, linear temporal logic (LTL) provides a well-structured and user-friendly language for specifying the temporal behaviors of multirobot [23], [24]. In this article, we consider a fragment of LTL called syntactically co-safe LTL (scLTL) [25], [26], in which each satisfaction trace has a "good prefix." Therefore, scLTL can describe desired behaviors in finite horizons such as "*eventually* reach some critical region" or "keep searching for a target *until* it is found."

In the past years, MRS task planning for LTL specifications has drawn considerable attention and has been successfully applied in many different applications. For example, Wolff et al. [27] presented a linear-programming-based method for optimal control of discrete-time dynamical systems. In [28], the automata-theoretical approach was used to find global plans for a team of robots in dynamic environments. In [29] and [30], sampling-based algorithms were developed to mitigate the computational complexity when the number of agents increases. In addition, distributed approaches have also been developed to coordinate each individual robot without communication such that the global task can be still achieved [31], [32]. More recently, different approaches have been proposed to solve the MRS temporal logic planning problem when the environment is only partially known [33], [34], [35], [36].

Most of the existing works on temporal logic path planning assume an ideal scenario where each robot operates flawlessly. In practical applications, however, robot *failures* are nonnegligible in severe working conditions. For instance, a robot may suffer from permanent damage to its mechanical components or lack of power, rendering it useless in accomplishing the task. In such cases, the LTL task depends on the collaboration of all robots, and a critical robot's failure could potentially jeopardize the entire mission. Therefore, it is crucial to account for the impact of potential failures during the design phase to

Received 14 July 2024; accepted 15 October 2024. Date of publication 28 November 2024; date of current version 25 February 2025. This work was supported by the National Natural Science Foundation of China under Grant 62173226, Grant 62061136004, and Grant 92367203. Recommended by Associate Editor S. Sacone. *(Corresponding author: Xiang Yin.)*

The authors are with the Department of Automation, Key Laboratory of System Control and Information Processing, Ministry of Education of China, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: bohan_cui@sjtu.edu.cn; huangfeifei@sjtu.edu.cn; syli@sjtu.edu.cn; yinxiang@sjtu.edu.cn).

Digital Object Identifier 10.1109/TCST.2024.3494392

ensure the robustness of the synthesized plans. On the other hand, there may exist multiple paths that satisfy the given specification and the "*failure-robust*" requirement. In this case, one would like to choose an optimal path, which satisfies the task under abnormal working conditions while optimizes the cost under normal working conditions.

In this article, we explore the optimal scLTL path planning problem for MRSs under robot failures. Specifically, we investigate the scenario, where no online communication between robots is available during their executions. Hence, one needs to synthesize the plans for all robots in a centralized manner, while at the execution stage, each robot will execute its own plan in a distributed manner. However, in contrast to existing works, here we consider a scenario where some robots may fail and can no longer contribute to the satisfaction of the overall task. Therefore, our objective becomes to synthesize a "failure-robust" plan for the team of robots, which means that even if a bounded number of robots fail, the remaining functioning robots can still successfully accomplish the LTL task by following predesigned plans. Furthermore, in addition to the failure-robust requirement for the worst case, we further want to select a plan with the lowest accumulated cost for the nominal case.

Our basic approach for solving the failure-robust LTL planning problem still relies on globally solving the mixed-integer linear programming (MILP) where we encode the path of each robot and the satisfaction of the joint LTL task. To address the issue of robot failures, we introduce the notion of *enable sequence* to capture the working status of each robot. To achieve the optimality of the plan, we consider a cost function that evaluates the cumulative cost of all robots over the entire time horizon under normal working conditions. However, this basic approach can quickly become computationally intractable, as it requires enumerating all possible combinations of failed robots and failure instants. To mitigate the computational complexity, we further identify a new fragment of LTL formula called the free-union-closed LTL. Specifically, we show that for LTL tasks within the fragment, the corresponding robust planning problem can be solved more efficiently by restricting to a simple case, in which the number of failure robots is fixed from the initial time and no more robots will fail during execution. This fragment requires significantly lower computational complexity, enabling us to find solutions more efficiently.

There are several related works in the literature where the issue of robot failures in path planning has been discussed. For instance, [37], [38] address the target tracking issue in MRSs, considering sensor failures due to attacks, while our article explores actuator failure. Concerning LTL task planning, [39] presents counting LTL (cLTL) to coordinate tasks among multiple robots, which requires that one task be satisfied by multiple robots. The cLTL approach can indeed address the issue of failure robustness to some extent. To achieve this, one needs to explicitly decompose the original LTL formula into a set of subformulae that can be handled by a single robot, and then manually assign an additional number of robots for each subformula. However, there is no systematic decomposition and assignment procedure provided in [39] for the purpose of failure robustness. Furthermore, by restricting the structure of cLTL formulae a priori, the solution may be unnecessarily restrictive compared with our approach, which directly solves the robust planning problem. In the work [40], a solution to the failure-robust reactive synthesis problem for LTL tasks is proposed, allowing each robot to adapt its plan dynamically based on others' failure status. Nonetheless, this method typically relies on a global controller or extensive interrobot communication, which might not always be possible. Our focus is on developing a failure-robust plan for a robot team that can be executed in a fully distributed "open-loop" manner, which is more suitable for more challenging environments where global control or robot-to-robot communication is infeasible.

The rest of this article is organized as follows. In Section II, we present the system model and the temporal logic tasks. The failure-robust LTL planning problem is formally formulated in Section III. In Section IV, a basic solution to this problem based on MILP is provided. A more scalable approach for a particular fragment of formulae called free-union-closed LTL is developed in Section V, and related properties for this fragment and formula instances are discussed in Section VI. Both the numerical and hardware experiments are provided in Section VII. Finally, we conclude this article in Section VIII. Preliminary and partial results in this article were previously discussed in [41]. In comparison to [41], this article differs in the following aspects. First, while [41] focuses solely on ensuring the robust satisfaction of the logic task, this article also incorporates numerical considerations. Second, this article presents a systematic method for determining whether an LTL formula belongs to the free-union-closed fragment. Finally, a comprehensive series of numerical experiments and hardware implementations are included in this article to demonstrate the efficacy of the proposed methodology.

II. Preliminaries

A. System Model

We consider a group of n homogeneous robots that operate within the same workspace. We denote by $\mathcal{I} = \{1, \ldots, n\}$ the index set for robots. The workspace consists of a finite collection of distinct regions or states, denoted as S. The connectivity between these states is defined by a symmetric neighborhood relationship, denoted as $\mathcal{N} \subseteq S \times S$. Specifically, if $(s, s') \in \mathcal{N}$, we say that states s and s' are neighborhoods. This neighborhood relationship implies that a robot can only move to a neighboring state at any given time instant. The cost of moving between neighboring states is denoted by $\mathcal{C} : \mathcal{N} \to \mathbb{N}$. For instance, $\mathcal{C}(s, s') = c$ indicates that the cost of a robot moving from s to s' (or from s' to s) is equal to c.

We also assume that each state in the workspace may have certain properties of interest to the user. These properties could be any relevant characteristics, such as resource availability, environmental conditions, or task requirements. Each state has a unique combination of these properties, making them potentially more or less desirable for the robots to visit or occupy. More specifically, let $\mathcal{AP}$ be a finite set of *atomic propositions*. The properties of each state are represented by a labeling function $L : S \to 2^{\mathcal{AP}}$, i.e., for each state $s \in S$,

$L(s)$ denotes the set of atomic propositions hold at state s. For a set of states $S' \subseteq S$, we also denote $L(S') = \cup_{s\in S'} L(s)$.

B. Paths and Traces

For each robot $i \in \mathcal{I}$, a *path* is an infinite sequence over the state space satisfying the transition relationship and S^ω denotes the set of all paths over S. Formally, we say $\mathbf{p}_i = \mathbf{p}_i^0\mathbf{p}_i^1\mathbf{p}_i^2\cdots \in S^\omega$ is a path for the ith robot if

$$\mathbf{p}_i^0 = s_i^{\text{int}} \quad \text{and} \quad \left(\mathbf{p}_i^t, \mathbf{p}_i^{t+1}\right) \in \mathcal{N} \quad \forall t \in \mathbb{N} \tag{1}$$

where s_i^{int} denotes the initial state of the ith robot. The set of all possible paths of the ith robot is denoted as $\mathbf{P}_i$.

We assume that the movement of each robot is fully synchronized, e.g., with a global time clock. Then the collective behavior of the team of n robots is a *joint path*, $\mathbf{p} = (\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n)$, where each $\mathbf{p}_i \in \mathbf{P}_i$ is a path for the ith robot. The set of all possible joint paths of the n-robot system satisfying the above conditions is denoted by $\mathbf{P}$.

Given that the robots are collaborating to achieve a common task, we define the set of atomic propositions that hold at each instant as the union of all atomic propositions satisfied by each robot. In other words, we consider the collection of atomic propositions that are true for any of the robots in the workspace. Then given a joint path $\mathbf{p}$, its *trace* is a sequence of atomic propositions (i.e., *word*) defined by $L(\mathbf{p}) = L(\cup_{i\in\mathcal{I}}\mathbf{p}_i^0)L(\cup_{i\in\mathcal{I}}\mathbf{p}_i^1)\cdots \in (2^{\mathcal{AP}})^\omega$.

C. LTL Tasks

The desired cooperative task for the team of robots is described by an LTL formula. Formally, the syntax of LTL is defined as follows:

$$\varphi ::= \text{ true } \mid a \in \mathcal{AP} \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \bigcirc \varphi \mid \varphi_1 \ \mathbf{U} \ \varphi_2$$

where $\neg$ and $\wedge$ are the Boolean operators "negation" and "conjunction," respectively, and $\bigcirc$ and $\mathbf{U}$ are the temporal operators "next" and "until," respectively. For the brevity in expression, define temporal operators "finally" $\mathbf{F}\phi = \top \ \mathbf{U} \ \phi$.

Particularly, an LTL formula belongs to the class of co-safe LTL (scLTL) formulae if it exclusively uses the temporal operators $\bigcirc$, $\mathbf{U}$, and $\mathbf{F}$, and the negation operator $\neg$ occurs only in front of an atomic proposition [21]. For example, "finally reach critical region a and do not reach region b until reach region c" can be expressed by scLTL formula $\mathbf{F}a \wedge (\neg b \ \mathbf{U} \ c)$. One of the most important properties of scLTL formulae is that their satisfaction can be determined in finite horizons. Specifically, for any infinite word $\sigma = \sigma_0\sigma_1\sigma_2\cdots \in (2^{\mathcal{AP}})^\omega$, it satisfies an scLTL formula φ iff it contains a finite good "prefix" that satisfies φ. We denote by $(\sigma, t) \models \varphi$ if σ satisfies LTL formula φ at time t. When $t = 0$, we omit instant t and write it as $\sigma \models \varphi$. For further details on the semantics of LTL, we refer the reader to the book [42].

The standard LTL path planning problem for MRSs (without considering failures) is formulated as follows.

Problem 1 (LTL Planning Problem): For an MRS with n robots in a workspace with states set S and initial states $s_1^{\text{int}}, s_2^{\text{int}}, \ldots, s_n^{\text{int}} \in S$, neighborhood relationship $\mathcal{N}$, and labeling function L, where $\mathcal{AP}$ is the set of atomic propositions, given an scLTL formula φ, find a joint path $\mathbf{p} \in \mathbf{P}$ such that $L(\mathbf{p}) \models \varphi$.

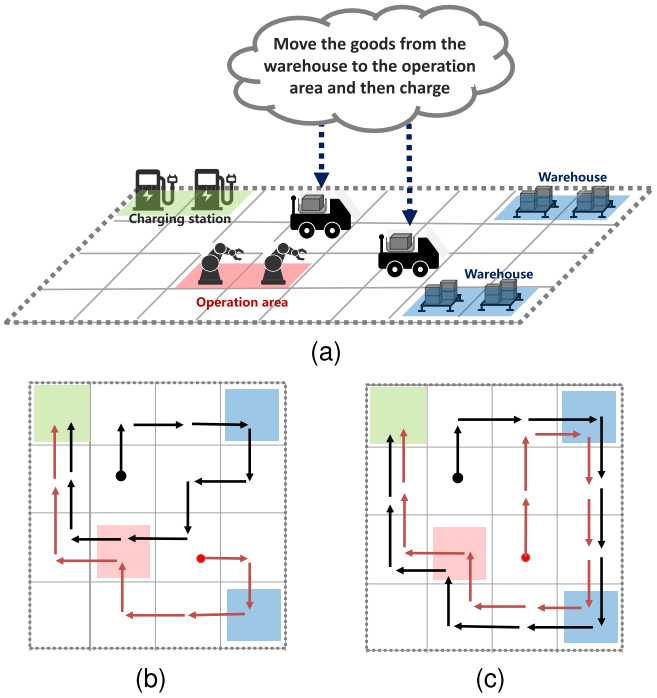


Fig. 1. Two solutions for the task planning of autonomous vehicles. (a) Workspace of an unmanned factory with two autonomous robots. (b) Optimal but nonrobust path. (c) Failure-robust path.

Note that while a joint path can be infinite in general, in our case, the workspace is finite and we restrict the task to scLTL formulae. As a result, the robots can always complete the task within a finite horizon. In this study, we specifically focus on finite joint paths denoted as $\mathbf{p} = \mathbf{p}^0\mathbf{p}^1\cdots\mathbf{p}^{h-1}$ and the *planning horizon* h can be an arbitrary integer. To solve the programming problem, we fix h and refer to it as the *planning horizon*. We denote planning time set $\{0, 1, \ldots, h-1\}$ as $\mathcal{H}$.

III. Failure-Robust LTL Planning

Problem 1 assumes an ideal scenario where all robots work correctly during their operations. However, in practice, robot failures may occur, and the failed robots cannot contribute to the overall task accomplishment. To address this issue, we present our model for the failure-robust planning problem in this section. To better motivate our study, we first consider the following example.

Consider an unmanned factory with two autonomous robots shown in Fig. 1(a). The objective of the robots is to first deliver goods from the warehouse to the operation area and then to go to the charging station. We assume that each robot will execute its preplanned path without communication. For this task-planning problem, we have found two possible solutions.

- *Optimal but Nonrobust Solution:* Fig. 1(b) shows a feasible solution to achieve the above task in a cost-optimal manner (in terms of the total distance of paths). Specifically, each robot first goes to a warehouse to pick up goods and then goes to an operation area and the charging station in order. However, this joint path is not robust

against robot failures. Particularly, if one robot fails, then the corresponding goods in a workspace cannot be delivered.

- *Failure-Robust Solution:* Fig. 1(c) provides an alternative solution. Although it involves a longer path for each robot, it is robust under robot failures. Specifically, no matter which robot fails, the other robot will still be able to visit two warehouses and correctly deliver goods. We refer to this solution as a "failure-robust" plan, and our objective is to develop an algorithm to synthesize such plans automatically.

Specifically, our problem setting is developed based on the following assumptions.

A1 Each robot may experience a failure at any point during its execution.
A2 The failure is considered permanent, meaning that the robot cannot recover from the failure.
A3 A failed robot is unable to contribute to the satisfaction of atomic propositions for the global task.
A4 There are at most $k < n$ (n is the number of robots) robot failures during the entire execution.
A5 The robots cannot exchange information with each other during the execution process.

To formalize the above setting, for each robot $i \in \mathcal{I}$, we introduce a binary sequence of the following form:

$$\mathbf{e}_i = e_i^0 e_i^1 e_i^2 \cdots \in \{0, 1\}^\omega$$

which is referred to as the *enable sequence*, to capture the working status of each robot. Specifically, for each time instant $t \in \mathbb{N}$, $e_i^t = 1$ means that the ith robot is working normally at instant t and $e_i^t = 0$ means that the ith robot has failed at instant t. Furthermore, due to the permanent failure assumption A2, we require that

$$\forall t \in \mathbb{N} \quad \forall \Delta \in \mathbb{N} : \mathbf{e}_i^t = 0 \Rightarrow \mathbf{e}_i^{t+\Delta} = 0$$

i.e., whenever a robot fails at some time instant, it will stay in failure status from then on.

According to Assumption A3, once a robot fails, it can no longer contribute to the overall task. To this end, we use the symbol b to represent that a robot is at a "failure state," and we extend the labeling function to domain $S \cup \{b\}$ by adding $L(b) = \emptyset$. Therefore, for any path $\mathbf{p}_i$ of robot i, an enable sequence $\mathbf{e}_i$ induces a new *failure-enabled path* over $S \cup \{b\}$ denoted by $\mathbf{p}_i \otimes \mathbf{e}_i$, where

$$(\mathbf{p}_i \otimes \mathbf{e}_i)^t = \begin{cases} \mathbf{p}_i^t, & \text{if } \mathbf{e}_i^t = 1 \\ b, & \text{if } \mathbf{e}_i^t = 0 \end{cases} \quad \forall t \in \mathbb{N}. \tag{2}$$

Correspondingly, the trace of $\mathbf{p}_i \otimes \mathbf{e}_i$ is $L(\mathbf{p}_i \otimes \mathbf{e}_i)$, where

$$L(\mathbf{p}_i \otimes \mathbf{e}_i)^t = \begin{cases} L(\mathbf{p}_i^t), & \text{if } \mathbf{e}_i^t = 1 \\ \emptyset, & \text{if } \mathbf{e}_i^t = 0 \end{cases} \quad \forall t \in \mathbb{N}. \tag{3}$$

For the entire team of n robots, a *joint enable sequence* is an n-tuple of form

$$\mathbf{e} = (\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_n)$$

where each $\mathbf{e}_i$ is an enable sequence. Furthermore, by the assumption of the bounded number of failure robots in A4, we further require that

$$\sum_{i \in \mathcal{I}} \mathbf{e}_i^t \geq n - k \quad \forall t \in \mathbb{N}.$$

We denote by $\mathbf{E}_k$ the set of all possible joint enable sequences, where k is the maximum number of potentially broken robots. Then given a joint path $\mathbf{p} \in \mathbf{P}$ and a joint enable sequence $\mathbf{e} \in \mathbf{E}_k$, the joint failure-enabled path is $\mathbf{p} \otimes \mathbf{e} = (\mathbf{p}_1 \otimes \mathbf{e}_1, \ldots, \mathbf{p}_n \otimes \mathbf{e}_n)$ with trace $L(\mathbf{p} \otimes \mathbf{e})$.

Now, we formulate the failure-robust LTL task planning problem as follows.

Problem 2 (Failure-Robust LTL Planning Problem): For an MRS system with n robots in a workspace with S and initial states $s_1^{\text{int}}, s_2^{\text{int}}, \ldots, s_n^{\text{int}} \in S$, neighborhood relationship $\mathcal{N}$, and labeling function L, given an scLTL formula φ, a maximum number of potential broken robots k, find a joint path $\mathbf{p} \in \mathbf{P}$ such that $\forall \mathbf{e} \in \mathbf{E}_k : L(\mathbf{p} \otimes \mathbf{e}) \models \varphi$.

We make the following remarks regarding our setting.

Remark 1: The variable k represents the maximum number of failures that the team can handle, which is determined in advance as a design parameter to account for worst case scenarios. Note that Assumption 4 does not necessarily mean that more than k robot failures will not occur. Instead, it represents the *level of robustness* for which the plans can *guarantee* the satisfaction of the LTL task. In other words, by assuming that there might be up to k robot failures during the entire execution, we establish a benchmark for evaluating the robustness of the planning approach. It is important to note that k is typically much smaller than the total number n of robots on the team. If k is set too high, then one may first seek to improve the reliability of the robots on the hardware side rather than using redundancy algorithms to improve the robustness of the system.

Remark 2: Note that in this article, the problem requires a global solving process. However, the execution of joint plans can be inherently decentralized. This means that each robot can follow its path in an open-loop manner without the need for any information exchange. In our previous work [40], we proposed a reactive strategy using a centralized method, which can essentially handle the situation where more than k failures occur. Nonetheless, in this setting, robots must request global information from all others to anticipate potential failures online, which is not the main focus of this article. Therefore, the approach presented in this article is better suited for extreme environments with limited or costly online communication options.

In addition to the failure-robust requirement for the worst case, in realistic task planning, we also care about the efficiency of robots completing the cooperative task for the nominal case, such as with the lowest accumulated cost. For example, if the system has a sufficient number of robots, then an intuitive way to ensure failure robustness is to let many robots execute the same path, and failure-robust requirements can be naturally fulfilled. However, this overly conservative strategy may potentially lead to excessive redundancy and resource waste, as we may be able to achieve a more efficient

division of labor while ensuring failure robustness. Thus, we further introduce the optimal problem of failure-robust LTL planning.

Throughout the entire task completion process, we evaluate the efficiency of the plan based on the total movement cost of all robots in the team. The cost function is defined as follows:

$$J(\mathbf{p}) = \sum_{i\in\mathcal{I}} \sum_{j,j+1\in\mathcal{H}} \mathcal{C}\left(\mathbf{p}_i^j, \mathbf{p}_i^{j+1}\right) \tag{4}$$

which allows us to measure the overall effectiveness of a joint path $\mathbf{p}$, while taking into account the contributions and cooperation of each individual robot.

Overall, our objective is to find a joint path that minimizes the cost function among all the failure-robust joint paths. Formally, the optimal synthesis problem is as follows.

Problem 3 (Optimal Failure-Robust LTL Planning Problem): For an MRS with n robots in a workspace with S and initial states $s_1^{\text{int}}, s_2^{\text{int}}, \ldots, s_n^{\text{int}} \in S$, neighborhood relationship $\mathcal{N}$, labeling function L, and cost mapping function $\mathcal{C}$, given an scLTL formula φ, a maximum number of potential broken robots k, find an optimal joint path $\mathbf{p} \in \tilde{\mathbf{P}}$ such that $\forall \mathbf{p}' \in \tilde{\mathbf{P}}$, $J(\mathbf{p}) \le J(\mathbf{p}')$, where $\tilde{\mathbf{P}} = \{\mathbf{p} | \forall \mathbf{e} \in \mathbf{E}_k : L(\mathbf{p} \otimes \mathbf{e}) \models \varphi\}$.

IV. MILP-Based Failure-Robust Solution

In the field of LTL path planning, an efficient approach is to use MILP to find a feasible plan [43]. The main idea behind this approach is to introduce a set of variables that encode the states of the robots and the satisfaction of the task. In this section, we extend the MILP-based approach to the failure-robust LTL planning problem.

A. MILP-Based Standard LTL Planning

Encoding Paths: As mentioned earlier, we restrict our attention to paths in finite form

$$\mathbf{p} = \mathbf{p}^0 \mathbf{p}^1 \cdots \mathbf{p}^{h-1}$$

where h is a predetermined fixed planning horizon. Then we introduce the following variables.

- *State Variables:* $\mathbf{p}^0, \mathbf{p}^1, \ldots, \mathbf{p}^{h-1} \in S^n$ representing the joint state of all robots within the planning horizon.

For these state variables, they should satisfy the **dynamic constraint** in (1).

Encoding LTL Tasks: To encode the LTL task φ, let Φ be the set of all subformulae in φ including the atomic propositions. Then for each proposition $\phi \in \Phi$, we introduce a set of

- *LTL Variables:* $z_\phi^0, z_\phi^1, \ldots, z_\phi^{h-1} \in \{0, 1\}$ representing the satisfaction of each subformula at each instant.

These LTL variables should satisfy the **LTL constraints** defined as follows: for each $t \in \mathcal{H}$, we have

- *Atomic Propositions:* if $\phi = a \in \mathcal{AP}$, then

$$z_a^t = 1 \quad \text{iff } a \in L\left(\mathbf{p}^t\right). \tag{5}$$

- *Negation:* if $\phi = \neg\phi'$, then

$$z_\phi^t = 1 \quad \text{iff } z_{\phi'}^t = 0. \tag{6}$$

- *Conjunction:* if $\phi = \bigwedge_{i=1}^m \phi_i$, then

$$z_\phi^t = 1 \quad \text{iff } \forall i \in \{1, 2, \ldots, m\} : z_{\phi_i}^t = 1. \tag{7}$$

- *Next:* if $\phi = \bigcirc\phi'$, then

$$z_\phi^t = 1 \quad \text{iff } z_{\phi'}^{t+1} = 1 \ (t \le h-2). \tag{8}$$

- *Until:* if $\phi = \phi_1 \ \mathbf{U} \ \phi_2$, then

$$\begin{aligned} z_\phi^t = 1 \quad & \text{iff } z_{\phi_2}^t = 1 (t \le h-1) \\ & \text{or } z_{\phi_1}^t = 1, \quad z_\phi^{t+1} = 1 (t \le h-2). \end{aligned} \tag{9}$$

The readers are referred to [43] for more details on how to express the aforementioned constraints in MILP formulation. It should be noted that due to the finite planning horizon in our study, certain temporal operators that require future LTL variables, such as $\bigcirc$ and $\mathbf{U}$, may not have meaningful interpretations. However, we emphasize that as long as the value of h is set to a sufficiently large value, the individual variable losses will not affect the solution of the problem.

B. Robust Constraints

The existing approach mentioned above is only applicable to the ideal scenario where robot failures are not considered (Problem 1). To synthesize a failure-robust plan, we need to take into account the possible failure events, in other words, to encode the joint enable sequence. Since in MILP formulation we restrict our attention to a finite horizon of length h, the encoding of joint enable sequences is modified accordingly.

Specifically, for each robot $i \in \mathcal{I}$, we introduce a set of

- *Failure-Enable Variables:* $e_i^0, e_i^1, \ldots, e_i^{h-1} \in \{0, 1\}$ representing the failure status of robot i.

Clearly, the failure-enable variables should satisfy the following **permanent failure constraint**:

$$\text{if } \mathbf{e}_i^t = 0 \quad \text{then } \mathbf{e}_i^j = 0 \quad \forall j \ge t \quad \forall t \in \mathcal{H} \tag{10}$$

and the **maximum failure constraint**

$$\sum_{i\in\mathcal{I}} e_i^t \ge n - k \quad \forall t \in \mathcal{H}. \tag{11}$$

These constraints on finite joint enable sequences are related to the planning horizon h and the maximum number of potentially broken robots k. Therefore, we define $\mathbf{E}_{k,h}$ as the set of all finite joint enable sequences satisfying the aforementioned constraints.

Finally, by considering the failure-enable variables, the atomic proposition variables should further be restricted by considering both the labeling function and the enable variable. Therefore, we need to replace the constraint on atomic propositions (5) with the following **label-enable constraint**:

$$z_a^t = 1 \quad \text{iff } a \in L\left(\mathbf{p}^t \otimes \mathbf{e}^t\right) \tag{12}$$

and all other LTL constraints remain the same.

Given an instance of Problem 3, the following robust-feasibility problem can be formed.

MILP for Problem 3:

$$\begin{aligned}
&\arg\min_{\mathbf{p}\in\mathbf{P}} J(\mathbf{p})\\
&\text{s.t. dynamic constraint} && (1)\\
&\quad \text{LTL constraints} && (6)\text{–}(9)\\
&\quad \text{label-enable constraint} && (12)\\
&\quad \text{satisfaction constraint} && z_\varphi^0 = 1,\ \text{for } \forall \mathbf{e} \in \mathbf{E}_{k,h}.
\end{aligned}$$

Note that in this MILP problem, only the state variables $\mathbf{p}$ are the decision variables. LTL variables evolve with the changes in the state variables, while the failure-enable variables are parameters that must be iterated and validated across all failure-robust constraints. Therefore, the solution to the above MILP problem provides a basic solution to Problem 3.

Remark 3: Note that the feasibility of the MILP generally depends on the selection of horizon h. Particularly, a small horizon h may lead to the infeasibility of the problem. To ensure the completeness of the MILP, the upper bound of h is the product of the size of the joint workspace and the size of the automaton accepting the scLTL formula. On the other hand, a higher h will bring a heavier computational burden, and there exists a tradeoff between feasibility and efficiency. Therefore, in practice, one usually starts by selecting a relatively small h (for example, the largest Manhattan distance in the workspace) and then gradually increases the value of h to the upper bound until a solution is obtained.

Remark 4: In this work, we implicitly assume that multiple robots can stay in the same state without collision. In some scenarios, each state can only accommodate at most one robot, and we need to explicitly consider the issue of collision avoidance. In this case, one can further add a collision avoidance constraint to the MILP such that, for each state variable, two robots cannot occupy the same state simultaneously.

V. Scalable Synthesis for Initial Robustness

In practice, the MILP problem presented in Section IV-B is complex to solve due to the robust constraint that considers all possible robot failures at any instant within the planning horizon. In this section, we introduce a scalable case called the *initial-failure* scenario, where all potential robot failures are assumed to occur only at the initial instant. We identify a fragment of LTL formulae called *free-union-closed LTL formulae*, and we demonstrate that if an LTL formula is free-union-closed, then to ensure the failure-robust constraints for the entire planning horizon, it suffices to consider failure-robust constraints only for the initial-failure scenario. This approach can significantly reduce the complexity of the MILP problem.

A. Initial Robustness

In the failure-robust planning problem, we require the planned path $\mathbf{p} \in \mathbf{P}$ to satisfy

$$\forall \mathbf{e} \in \mathbf{E}_k : L(\mathbf{p} \otimes \mathbf{e}) \models \varphi.$$

Hereafter, we refer to the above condition as *global robustness* since the enable sequence set $\mathbf{E}_k$ considers possible failures globally.

To mitigate the complexity, we further restrict our attention to a special case where robots are only supposed to break initially. Furthermore, for the worst case analysis, we assume that there are exactly k broken robots initially. To this end, we define the set of all possible initial-failure enable sequences by the following equation:

$$\mathbf{E}_k^{\text{int}} = \left\{ \mathbf{e} \in \mathbf{E}_k \,\middle|\, \sum \mathbf{e}_i^0 = n - k \right\}.$$

Then we say that a planned path $\mathbf{p} \in \mathbf{P}$ is *initially robust* if the LTL task is fulfilled whenever there are exactly k robots that fail initially, and all the remaining robots operate correctly for the entire planning horizon. This means that we only need to consider all possible initial-failure scenarios to ensure robustness.

Definition 1 (Initial Robustness): Given an scLTL formula φ, a maximum number of potential failures k, a joint path $\mathbf{p}$ is said to be *initially-robust* if $\forall \mathbf{e} \in \mathbf{E}_k^{\text{int}} : L(\mathbf{p} \otimes \mathbf{e}) \models \varphi$.

Based on the above definition, we define a new failure-robust planning problem as follows.

Problem 4: For an MRS with n robots in a workspace with states S, initial states $s_1^{\text{int}}, s_2^{\text{int}}, \ldots, s_n^{\text{int}} \in S$, neighborhood relationship $\mathcal{N}$, labeling function L, and cost mapping function $\mathcal{C}$, given an scLTL formula φ, the maximum number of potential broken robots k, find an optimal joint path $\mathbf{p} \in \tilde{\mathbf{P}}$ such that $\forall \mathbf{p}' \in \tilde{\mathbf{P}}$, $J(\mathbf{p}) \leq J(\mathbf{p}')$, where $\tilde{\mathbf{P}} = \{\mathbf{p} \in \mathbf{P} | \forall \mathbf{e} \in \mathbf{E}_k^{\text{int}} : L(\mathbf{p} \otimes \mathbf{e}) \models \varphi\}$.

Analogously, we can solve an MILP problem to get an initially robust joint path.

MILP for Problem 4:

$$\begin{aligned}
&\arg\min_{\mathbf{p}\in\mathbf{P}} J(\mathbf{p})\\
&\text{s.t. dynamic constraint} && (1)\\
&\quad \text{LTL constraints} && (6)\text{–}(9)\\
&\quad \text{label-enable constraint} && (12)\\
&\quad \text{satisfaction constraint} && z_\varphi^0 = 1,\ \text{for } \forall \mathbf{e} \in \mathbf{E}_{k,h}^{\text{int}}.
\end{aligned}$$

Here, $\mathbf{E}_{k,h}^{\text{int}} \subseteq \mathbf{E}_{k,h}$ is the set of finite initial-failure enable sequences. The MILP for initial robustness improves the original MILP for global robustness in two ways. First, it considers exact k broken robots, rather than any subset of k robots. Second, and more importantly, it only considers initial robot failures, which avoids the exponential growth of variables that would occur if robot failures were considered over an extended planning horizon h.

Remark 5: In principle, the failure-robust planning problem is inherently NP-hard in both the total number of robots and the number of possible failures. Compared with the MILP for global robustness, the MILP for initial robustness can significantly reduce the complexity of the problem by limiting the consideration of robot failures. Specifically, the original MILP for global robustness involves $\mathcal{O}(\sum_{i=0}^{k} \mathbf{C}_n^i h^i)$ failure-robust constraints, where $|\cdot|$ represents the number of elements in a set. However, the case of MILP for initial robustness only involves $\mathcal{O}(\mathbf{C}_n^k)$ failure-robust constraints, which can lead to faster computation times and more efficient solutions.

B. Comparisons of Global Robustness and Initial Robustness

As discussed above, the main advantage of solving Problem 4 is its computational efficiency since it involves fewer variables and constraints compared with Problem 3. Due to computational efficiency, our general idea is to tackle Problem 4 for an initially robust solution rather than addressing Problem 3. Nonetheless, our goal remains to ensure that the synthesized plan is globally robust. According to the definitions, it is clear that a globally robust plan is also initially robust. However, the reverse is not always the case, as illustrated by the following counterexamples.

Example 1: Suppose we have a team of two robots and consider the case of $k = 1$. The global LTL task for the team of two robots is given by the following equation:

$$\varphi = \mathbf{F}(a \wedge \neg b) \vee \mathbf{F}(\neg a \wedge b)$$

where a and b represent two different types of regions. The overall task requires the robots to eventually visit either region a or region b, but these two regions cannot be both visited. If we only consider initial robustness, then we can design a plan where two robots visit a and b simultaneously. This plan is initially robust since one robot is assumed to fail initially. However, this plan is not globally robust since if neither robot fails, regions a and b may be visited simultaneously and we cannot find an instant where exactly one region is visited.

Example 2: Suppose we have a team of two robots and consider the case of $k = 1$. The global LTL task for the team of two robots is given by the following equation:

$$(\mathbf{F}a \rightarrow \mathbf{F}b) \wedge \mathbf{F}c.$$

A possible initially robust plan is to let one robot straightly visit c and the other visit a and b and then c. It can be verified that no matter which robot is broken initially or no robot is broken, they can always fulfill the task. However, if the second robot is broken after it visits a, the first robot will not visit b according to its plan, thus the task cannot be fulfilled.

The above two examples illustrate why initial robustness does not necessarily imply global robustness. In the first example, two individual plans that satisfy the LTL task may conflict with each other, and in the second example, noninitial failures during the execution may cause additional problems. To identify fragments of LTL formulae where initial robustness and global robustness are equivalent, our approach is to exclude those scenarios where initial robustness may lead to plans that do not remain robust in the face of potential failures over an extended planning horizon.

C. Free-Union-Closed LTL Formulae

Although initial robustness and global robustness are not equivalent in general, as discussed earlier, there are certain fragments of LTL formulae where they are equivalent. In our work, we identify such fragments of LTL formulae called the *free-union-closed LTL*.

Definition 2 (Free-Unions): Let $\mathbf{w} = w_0 w_1 w_2, \ldots, \mathbf{v} = v_0 v_1 v_2 \cdots \in (2^{\mathcal{AP}})^\omega$ be two infinite words. We say word $\mathbf{w}' = w'_0 w'_1 w'_2 \cdots \in (2^{\mathcal{AP}})^\omega$ is a *free-union* of $\mathbf{w}$ and $\mathbf{v}$ if for some $j \in \mathbb{N}$, we have

$$w'_i = \begin{cases} w_i \cup v_i, & \text{if } i < j \\ w_i, & \text{if } i \geq j. \end{cases} \tag{13}$$

We denote by $\mathbf{w} \uplus \mathbf{v}$ the set of all free-unions of $\mathbf{w}$ and $\mathbf{v}$.

Intuitively, we can consider $j \in \mathbb{N}$ as the time instant from which word $\mathbf{v}$ becomes ineffective. Therefore, $\mathbf{v}$ can only contribute to the atomic propositions union of $\mathbf{w}$ up to instant $j-1$. It is important to note that the free-union of $\mathbf{w}$ and $\mathbf{v}$ is not unique, as we are free to choose instant j to be any value from 0 to infinity.

Based on the free-union operator of two words, we introduce the fragment of *free-union-closed LTL*. This fragment ensures that for any two words satisfying the LTL formula, any of their free-unions will still satisfy the same LTL formula.

Definition 3 (Free-Union-Closed LTL): Given an LTL formula φ and let $\mathsf{word}(\varphi) = \{\sigma \in (2^{\mathcal{AP}})^\omega \mid \sigma \models \varphi\}$ be the set of all words satisfying φ. Formula φ is said to be *free-union-closed* if for any $\mathbf{w}, \mathbf{v} \in \mathsf{word}(\varphi)$, we have $\mathbf{w} \uplus \mathbf{v} \subseteq \mathsf{word}(\varphi)$.

The following theorem is the main result of this section, that for any free-union-closed LTL formula, initial robustness and global robustness are equivalent. Therefore, to solve Problem 3, we can solve Problem 4 instead.

Theorem 1: For free-union-closed LTL formula φ, initial robustness of joint path $\mathbf{p}$ is equivalent to global robustness, i.e., $\forall \mathbf{e}' \in \mathbf{E}_k^{\text{int}}, L(\mathbf{p} \otimes \mathbf{e}') \models \varphi \Leftrightarrow \forall \mathbf{e} \in \mathbf{E}_k, L(\mathbf{p} \otimes \mathbf{e}) \models \varphi$.

Proof: Clearly, global robustness implies initial robustness. Therefore, we prove the other direction by contradiction.

Given a free-union-closed LTL task φ, assume that there exists a joint path $\mathbf{p}$ that is initially robust, but not globally robust, i.e., $\exists \mathbf{p} \in \mathbf{P}$, s.t.$\forall \mathbf{e}' \in \mathbf{E}_k^{\text{int}}, L(\mathbf{p} \otimes \mathbf{e}') \models \varphi$ and $\exists \mathbf{e} \in \mathbf{E}_k \setminus \mathbf{E}_k^{\text{int}}, L(\mathbf{p} \otimes \mathbf{e}) \not\models \varphi$. We construct an initial-failure enable sequence $\bar{\mathbf{e}} \in \mathbf{E}_k^{\text{int}}$ based on $\mathbf{e}$ such that for $\forall i \in \mathcal{I}, \forall t \in \mathbb{N}$

$$\text{if } \mathbf{e}_i^t = 0 \quad \text{then } \bar{\mathbf{e}}_i^t = 0.$$

The relationship between $\mathbf{e}$ and $\bar{\mathbf{e}}$ can be explained by considering two scenarios: no more than k failures within the whole planning horizon, and all the robot failures occur at the initial instant. If the total number of failures is less than k in $\mathbf{e}$, $\bar{\mathbf{e}}$ randomly supplements the other failures. Therefore, these two enable sequences to capture the cases of global robustness and initial robustness, respectively.

Let $\mathcal{B}$ be the index set of robots that do *not* initially fail, but *will fail* in the future execution time, i.e., $\mathcal{B} = \{i | \mathbf{e}_i^0 \neq 0, \bar{\mathbf{e}}_i^0 = 0\}$. We construct an additional auxiliary enable sequence $\bar{\mathbf{e}}^i \in \mathbf{E}_k^{\text{int}}$ for each $i \in \mathcal{B}$ based on $\bar{\mathbf{e}}$. The sequence $\bar{\mathbf{e}}^i$ represents the scenario where one effective robot in $\bar{\mathbf{e}}$ failed initially, and robot i is effective instead. It is important to note that $\bar{\mathbf{e}}^i$ still contains k failures, thus belonging to $\mathbf{E}_k^{\text{int}}$. Since $\mathbf{p}$ is assumed to be initially robust, we have $L(\mathbf{p} \otimes \bar{\mathbf{e}}) \models \varphi$ and $\forall i \in \mathcal{B}$, $L(\mathbf{p} \otimes \bar{\mathbf{e}}^i) \models \varphi$. Therefore, we know that $L(\mathbf{p} \otimes \bar{\mathbf{e}}) \in \mathsf{word}(\varphi)$ and $L(\mathbf{p} \otimes \bar{\mathbf{e}}^i) \in \mathsf{word}(\varphi)$ for all $i \in \mathcal{B}$. To simplify notation, we denote the union set of $L(\mathbf{p} \otimes \bar{\mathbf{e}})$ and the traces of $\mathbf{p}$ with auxiliary enable sequences as $\uplus(L(\mathbf{p} \otimes \bar{\mathbf{e}}), \sum_{i \in \mathcal{B}}^{+} L(\mathbf{p} \otimes \bar{\mathbf{e}}^i))$. As φ is free-union-closed, we can conclude that $\uplus(L(\mathbf{p} \otimes \bar{\mathbf{e}}), \sum_{i \in \mathcal{B}}^{+} L(\mathbf{p} \otimes \bar{\mathbf{e}}^i)) \subseteq \mathsf{word}(\varphi)$. By construction of the auxiliary initial-failure enable sequence,

we can easily see that $L(\mathbf{p}\otimes\mathbf{e}) \in \biguplus(L(\mathbf{p}\otimes\bar{\mathbf{e}}), \sum^{+}_{i\in\mathcal{B}} L(\mathbf{p}\otimes\bar{\mathbf{e}}^i))$. This implies that $L(\mathbf{p}\otimes\mathbf{e})$ satisfies φ, which contradicts our assumption. □

VI. Property of Free-Union-Closed LTL

In Section V-C, we have identified a tractable fragment of LTL, for which robust plans can be synthesized more efficiently. In this section, we discuss in more detail template formulae for this fragment and how to check whether or not an LTL formula belongs to this fragment.

A. Instances of Free-Union-Closed LTL

First, we show that the free-union-closed property is closed under conjunction.

Theorem 2: If LTL formulae φ_1 and φ_2 are both free-union-closed LTL formulae, then $\varphi_1 \wedge \varphi_2$ is also free-union-closed.

Proof: We prove by contraposition, i.e., for any LTL formula $\varphi = \varphi_1 \wedge \varphi_2$, if φ is not free-union-closed, then either φ_1 or φ_2 is not free-union-closed.

As φ is not free-union-closed, we can find two words σ_1 and σ_2, s.t., $\sigma_1 \models \varphi, \sigma_2 \models \varphi, \exists\sigma \in \sigma_1 \biguplus \sigma_2$ and $\sigma \not\models \varphi$. We can get that either $\sigma \not\models \varphi_1$ or $\sigma \not\models \varphi_2$. In addition, $\sigma_1 \models \varphi_1, \sigma_1 \models \varphi_2, \sigma_2 \models \varphi_1, \sigma_2 \models \varphi_2$, then we can get that either φ_1 or φ_2 is not free-union-closed. □

On the other hand, the free-union-closed property is not closed under disjunction. To see this, we provide a simple counterexample. Let $\mathcal{AP} = \{a_0, a_1, a_2, a_3\}, \varphi_1 = \neg a_1 \mathbf{U} a_2, \varphi_2 = \mathbf{F} a_3$. It is easy to show that φ_1 and φ_2 are free-union-closed. Let $\sigma_1 = a_0a_2a_1, \ldots, \sigma_2 = a_1a_2a_3\cdots$, we can choose $\sigma = \{a_0a_1\}a_2a_1\cdots \in \sigma_1 \biguplus \sigma_2$ and verify that $\sigma \not\models \varphi_1 \vee \varphi_2$. Thus, $\varphi_1 \vee \varphi_2$ is not free-union-closed.

Next, we provide some useful patterns that are free-union-closed. To the end, we start with the Boolean expressions and introduce the following definitions.

Definition 4: Let ϕ be a Boolean expression over $\mathcal{AP}$, i.e., ϕ is an LTL formula without temporal operators. We say that formula ϕ is

- *Monotone* if for any $X_1, X_2 \in 2^{\mathcal{AP}}$, we have $X_1 \models \phi \wedge X_1 \subseteq X_2 \Rightarrow X_2 \models \phi$.
- *Closed* if for any $X_1, X_2 \in 2^{\mathcal{AP}}$, we have $X_1 \models \phi \wedge X_2 \models \phi \Rightarrow X_1 \cup X_2 \models \phi$.

The following are some immediate observations of the monotonicity and closeness of Boolean expressions.

- If ϕ is monotone, then it is also closed. However, the converse direction is not true in general. For example, for $\phi = \neg a$, it is closed but is not monotone.
- If ϕ_1 and ϕ_2 are monotone, their conjunction $\phi_1 \wedge \phi_2$ and disjunction $\phi_1 \vee \phi_2$ are also monotone. However, the negation $\neg\phi_1$ may not be monotone.

Based on the monotonicity and closeness of Boolean expressions, we show that two commonly used patterns of scLTL formulae are free-union-closed, and therefore, the corresponding plan can be efficiently synthesized based on initial robustness.

Proposition 1: Let ϕ, ϕ_1, ϕ_2 be the Boolean expressions over $\mathcal{AP}$. Then we have the following results.

- If ϕ is monotone, then formula $\mathbf{F}\phi$ is free-union-closed.
- If ϕ_1 is closed and ϕ_2 is monotone, then formula $\phi_1\mathbf{U}\phi_2$ is free-union-closed.

Proof: The first pattern is proven as follows. Let $\mathbf{w} = w_0w_1w_2\cdots$ and $\mathbf{v} = v_0v_1v_2\cdots$ be two words that satisfy the formula $\mathbf{F}\phi$. Randomly select $\mathbf{w}' = w'_0w'_1w'_2\cdots$ from the free-union of $\mathbf{w}$ and $\mathbf{v}$. According to the definition of $\mathbf{F}$, we can get that $\exists i \in \mathbb{N}$ such that w_i satisfies ϕ. Since ϕ is monotone, it follows that $w_i \cup v_i$ also satisfies ϕ. Therefore, regardless of whether w'_i is equal to w_i or $w_i \cup v_i$, w'_i satisfies ϕ. Consequently, $\mathbf{w}'$ satisfies $\mathbf{F}\phi$.

For the second pattern, let $\mathbf{w} = w_0w_1w_2\cdots$ and $\mathbf{v} = v_0v_1v_2\cdots$ be two words that satisfy $\phi_1\mathbf{U}\phi_2$. We randomly select $\mathbf{w}' = w'_0w'_1w'_2\cdots$ from the free-union of $\mathbf{w}$ and $\mathbf{v}$. By definition, there exists a position in the world where ϕ_2 is satisfied, and before that moment, ϕ_1 holds consistently. Let $(w, t_1) \models \phi_2$ and $(v, t_2) \models \phi_2$. It can be inferred that $\forall t < t_1, (w, t) \models \phi_1$ and $\forall t < t_2, (v, t) \models \phi_1$.

When $t_1 < t_2$, it holds that $\forall t < t_1, (v, t) \models \phi_1$. Consequently, we have $(w \wedge v, t) \models \phi_1$ (since ϕ_1 is closed) and $(w \wedge v, t_1) \models \phi_2$ (since ϕ_2 is monotone). The situation is similar for $t_1 \geq t_2$. Thus, we can conclude that $\mathbf{w}'$ satisfies $\phi_1\mathbf{U}\phi_2$. □

B. Verification of Free-Union-Closed LTL

When an LTL formula does not belong to the above-identified patterns, we cannot conclude that it is free-union-closed or not directly. Here, we provide a general automata-based verification algorithm to verify whether or not an arbitrary LTL formula is free-union-closed.

Definition 5 (Finite-State Automata): A deterministic finite-state automaton (DFA) is a tuple $\mathcal{A} = (Q, q_0, \Sigma, \delta, F)$, where

- Q is the finite set of states;
- $q_0 \in Q$ is the initial state;
- $\Sigma = 2^{\mathcal{AP}}$ is the finite alphabet;
- $\delta : Q \times \Sigma \to Q$ is the state transition function;
- $F \subseteq Q$ is the set of accepting states.

A DFA $\mathcal{A}$ defines a language $\mathsf{word}(\mathcal{A})$ over words $\sigma = s_0s_1\cdots s_n \in \Sigma^\omega$. The word induces a run $\rho = \rho_0\rho_1\cdots\rho_n \in Q^\omega$ with $\rho_0 = q_0$ and for every $i \in \mathbb{N}$, we have $\rho_{i+1} = \delta(\rho_i, s_i)$. We say that ρ is accepted if and only if $\rho_n \in F$, and we say the word σ is accepted if its induced run is accepted.

It is well-known that for any scLTL formula φ over $\mathcal{AP}$, there exists a DFA $\mathcal{A}$ with alphabet $\Sigma = 2^{\mathcal{AP}}$ that accepts all and only good prefixes of φ. Now, we can check the free-union-closed property by constructing an auxiliary automaton based on DFA.

Definition 6 (Auxiliary Automata): Based on DFA $\mathcal{A} = (Q, q_0, \Sigma, \delta, F)$, the auxiliary automaton is defined as a new tuple $\hat{\mathcal{A}} = (\hat{Q}, \hat{Q}_0, \hat{\Sigma}, \hat{\delta}, \hat{F})$, where

- $\hat{Q} \subseteq Q \times Q \times Q \times \mathcal{M}$, $\mathcal{M} = \{0, 1\}$ is the set of mark states;
- $\hat{Q}_0 = \{(q_0, q_0, q_0, m) \subseteq \hat{Q} \mid m \in \mathcal{M}\}$;
- $\hat{\Sigma} \subseteq \Sigma \times \Sigma$;
- $\hat{\delta} : \hat{Q} \times \hat{\Sigma} \to 2^{\hat{Q}}$, and $\hat{\delta}(\langle q, q^1, q^2, m\rangle, \langle t, o\rangle) = \langle q', q^{1\prime}, q^{2\prime}, m'\rangle$ iff all the following conditions are true:

i) $m' \leq m$
ii) $q^{1\prime} = \delta(q^1, t)$, $q^{2\prime} = \delta(q^2, o)$
iii) $q' = \delta(q, s)$, where

$$s = \begin{cases} t \cup o, & \text{if } m = 1 \\ t, & \text{if } m = 0; \end{cases}$$

- $\hat{F} \subseteq \bar{F} \times F \times F$, where $\bar{F} = Q \setminus F$.

The auxiliary automaton $\hat{\mathcal{A}}$ defines $\mathsf{word}(\hat{\mathcal{A}})$. Here, the input words are pairs of sequences, denoted as $\langle \sigma^1, \sigma^2 \rangle$, where $\sigma^1 = t_0 t_1 t_2 \cdots t_n$ and $\sigma^2 = o_0 o_1 o_2 \cdots o_n$. A run for $\langle \sigma^1, \sigma^2 \rangle$ denotes a sequence $\rho = \rho_0 \rho_1 \cdots \rho_n$ with $\rho_0 \in \hat{Q}_0$ and for every $i \in \{0, 1, \ldots, n-1\}$, $\rho_{i+1} \in \hat{\delta}(\rho_i, \langle t_i, o_i \rangle)$. The run ρ is accepted by the auxiliary automaton iff $\rho_n \in \hat{F}$.

Based on the auxiliary automaton, the following result shows how to verify whether an scLTL formula is free-union-closed or not.

Proposition 2: For a scLTL formula φ, let $\mathcal{A}$ be the finite state automaton accepting φ and $\hat{\mathcal{A}}$ be the corresponding auxiliary automaton. Then an LTL formula φ is free-union-closed iff $\mathsf{word}(\hat{\mathcal{A}}) = \emptyset$.

Proof: Suppose we have an LTL formula denoted as φ, with the corresponding DFA represented by $\mathcal{A}$, we can construct an auxiliary automaton $\hat{\mathcal{A}}$ based on $\mathcal{A}$. Assuming that φ is free-union closed, while $\mathsf{word}(\hat{\mathcal{A}}) \neq \emptyset$.

Given this assumption, we can select a run $\rho = \rho_0 \rho_1 \cdots \rho_n$ randomly from the word set $\mathsf{word}(\hat{\mathcal{A}})$, accompanied by its associated input $\langle \sigma_1, \sigma_2 \rangle$. Here, each element of the run is represented as $\rho_i = \langle q_i, q_i^1, q_i^2 \rangle$.

Based on the acceptance condition of the auxiliary automaton, it follows that the state ρ_n is reachable and resides in the set $\hat{F}$. Consequently, we observe that $q_n \notin F$, $q_n^1 \in F$, and $q_n^2 \in F$. This implies that while the individual inputs σ_1 and σ_2 can be accepted by the original DFA $\mathcal{A}$, their free-union fails to satisfy the acceptance condition. As a result, we establish a contradiction to the assumption that the formula φ is free-union-closed. □

VII. Experimental Results

In this section, we present a set of experimental results illustrating our proposed failure-robust path planning algorithm. All the simulations are carried out by a laptop computer with Intel Core i7 CPU, 2.80 GHz and 16 GB of RAM. First, we provide a case study in a simulation environment to better illustrate the scenario of robot failures. Then we conduct a set of numerical experiments to show the scalability of our approach for the case of initial robustness compared with the case of global robustness. Finally, we provide a real-world implementation of three robots to further illustrate our approach.

Our code implementations, videos for simulations, and videos for hardware experiments can be found at https://github.com/PiRA502/Failure-Robust_Task_Planning.

A. Simulation Case Study

Consider an example of delivery robots working in a library workspace shown in Fig. 2. The simulation is performed in the

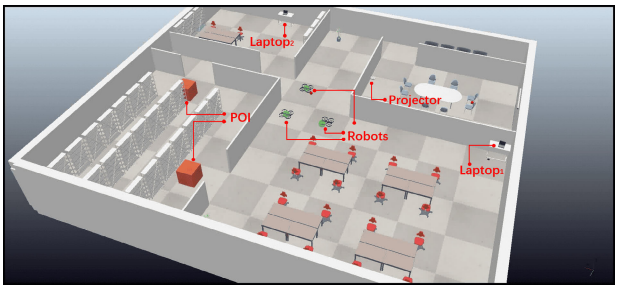


Fig. 2. Model diagram of library workspace.

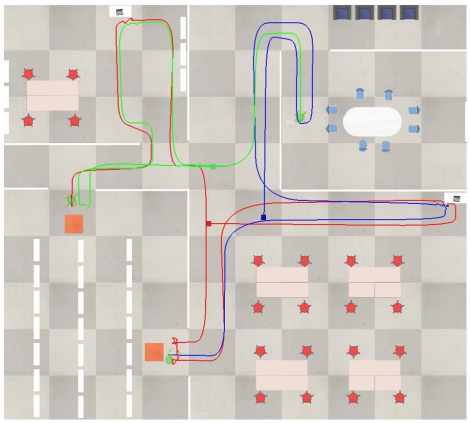


Fig. 3. Failure-robust plan for a three-robot system to inspect all the laptops and then reach book pickup points and inspect the projector.

V-REP robot simulator [44]. The workspace has two reading rooms equipped with laptops l_1 and l_2 for book retrieval, bookshelves with book pickup point *bp*, and a conference room with projection system *proj*.

There are three delivery robots in the workspace and we assume that at most one robot may fail during the execution, i.e., $k = 1$. The overall task is as follows. Each laptop needs to be visited to inquire about book status. Then a robot must proceed to any book pickup point to confirm the availability of the books. Furthermore, the robots must also visit the conference room to check whether the projection system is in the off state. This task can be represented using scLTL as follows:

$$\mathbf{F}(l_1 \wedge \mathbf{F}bp) \wedge \mathbf{F}(l_2 \wedge \mathbf{F}bp) \wedge \mathbf{F}proj.$$

A feasible failure-robust solution is depicted in Fig. 3, where green, red, and blue lines represent the planned paths for the three robots, respectively. It is evident that regardless of which robot experiences a failure and at what instant in the time horizon, the global task can always be completed.

B. Numerical Experiments

Next, we provide detailed numerical experiments to illustrate the scalability of our approach when system parameters change, including the size of the workspace, the number of robots, and the upper bound of potentially broken robots. For each set of experiments, we provide the MILP computing time for both the initial robustness solution and global robustness solution, which intuitively reflects the efficiency of the two algorithms.

1) Experimental Setting: We evaluate our algorithm for four types of overall tasks. The scLTL formulae are as follows:

$$\varphi_1 = \mathbf{F}r$$
$$\varphi_2 = \mathbf{F}r_1 \wedge \mathbf{F}r_2$$
$$\varphi_3 = (\neg r_2 \ \mathbf{U} \ r_1) \wedge \mathbf{F}r_2$$
$$\varphi_4 = \mathbf{F}(r_1 \wedge \mathbf{F}r_2)$$

where r, r_1 and r_2 are atomic propositions. Specifically, task φ_1 represents a reachability task and task φ_2 represents the combination of two reachability subtasks. Tasks φ_3 and φ_4 represent priority tasks of different strictness levels. Due to the nesting of temporal operators, task φ_4 is the most complex task. Generally speaking, the complexity of these four tasks gradually increases and this will be reflected in the subsequent experimental data. By applying templates or constructing auxiliary automata, it can be verified that these four formulae all belong to free-union-closed LTL. This theoretically ensures the equivalence of task planning between the initially robust algorithm and the globally robust algorithm.

We create the workspace for robots using grid-style maps. The placements of atomic propositions on the workspace are randomly generated, along with the initial states of the robots. In the subsequent content, we will present various configurations of experiments. For each configuration, we randomly generate 30 workspaces, encompassing different placements of atomic propositions and initial states of robots.

2) Scalability Results: In the numerical experiments, we change different system parameters, including the size of the workspace, the total number of robots, and the maximum number of potentially broken robots. For each system parameter and each overall task, we randomly generate 30 workspaces and apply two MILP algorithms to solve the same task planning problem. We then compare the efficiency of two algorithms by their MILP-running time.

In the first set of experiments, we fix the number of robots to be $n = 4$ and the maximum number of potentially broken robots to be $k = 1$. Then we consider workspaces of sizes ranging from 5×5 to 9×9. The statistic result for the averaged running time for each case is shown in Table I. For the task φ_4 with a map size of 9×9, in the context of globally robust task planning, the complexity of the MILP problem has grown so high that we cannot obtain a feasible solution within a reasonable time (60 min). However, for initially robust task planning, it can be solved within 35 s. Across different map sizes and tasks, the initially robust planning algorithm, compared with the globally robust planning algorithm, can reduce MILP-running time by over 94%.

In the second set of experiments, we fix the workspace size to be 5×5 and the maximum number of potentially broken robots to be $k = 1$. We then vary the number of robots from 2 to 8. The results of this experiment set are presented in Table II. As the number of robots increases, the disparities become more pronounced. For task φ_4 with eight robots, the globally robust planning algorithm fails to find a solution within the desired time, whereas the initially robust planning algorithm can solve this problem in 33 s.

TABLE I
MILP-RUNNING TIMES OF TWO ROBUST PLANNING ALGORITHMS ON DIFFERENT TASKS AND GRID-WORLD SIZES

	5 × 5		7 × 7		9 × 9	
	global	initial	global	initial	global	initial
φ_1	23.07	1.37	49.64	1.58	189.24	2.7
φ_2	66.29	2.30	95.33	2.29	425.58	4.94
φ_3	96.01	3.16	143.30	3.50	537.83	7.40
φ_4	378.01	10.85	584.47	13.53	-	34.37

TABLE II
MILP-RUNNING TIMES OF TWO ROBUST PLANNING ALGORITHMS ON DIFFERENT TASKS AND NUMBER OF ROBOTS

	N = 2		N = 4		N = 8	
	global	initial	global	initial	global	initial
φ_1	4.73	0.80	23.07	1.37	220.70	3.81
φ_2	12.22	0.92	66.29	2.30	470.64	7.06
φ_3	15.59	1.44	96.01	3.16	691.41	10.06
φ_4	90.85	3.87	378.01	10.85	-	32.67

TABLE III
MILP-RUNNING TIMES OF TWO ROBUST PLANNING ALGORITHMS ON DIFFERENT TASKS AND NUMBER OF FAILURE ROBOTS

	k = 1		k = 2		k = 3	
	global	initial	global	initial	global	initial
φ_1	220.70	3.81	-	30.54	-	99.53
φ_2	470.64	7.06	-	66.66	-	189.19
φ_3	691.41	10.06	-	94.57	-	315.61
φ_4	-	32.67	-	563.14	-	1592.75

In the last set of experiments, we fix the size of the workspace to 5×5 and the number of robots to eight. Then we consider the maximum number of potential broken [robots ranging from one to three. The results of this set of experiments are shown in Table III. When the maximum number of potentially broken robots exceeds one, regardless of the task, the globally robust planning algorithm cannot obtain results within a reasonable time. On the other hand, for initially robust task planning, even in the most complex configuration (task φ_4, $k = 3$), it takes less than 30 min, which is still within the acceptable range for offline planning. Overall, the maximum number of potentially broken robots has a greater impact on the MILP-running time. When k is set to a higher value, the globally robust planning algorithm is highly likely to be unable to find a solution within a reasonable time. This further confirms the necessity of the simplified algorithm by considering initial robustness, particularly when considering larger scale MRSs with a higher potential for robots' failures.

C. Hardware Experiments

Finally, to validate the effectiveness of our approach, we implement the proposed algorithm in Turtlebot3-Burger mobile robots in our laboratory. Specifically, we consider robots working in a 5×5 grid-style environment with four specified regions (R_1, R_2, D_1, and D_2) and two obstacle regions. The overall task is to assign robots to visit all four specific regions. This task can be represented using the

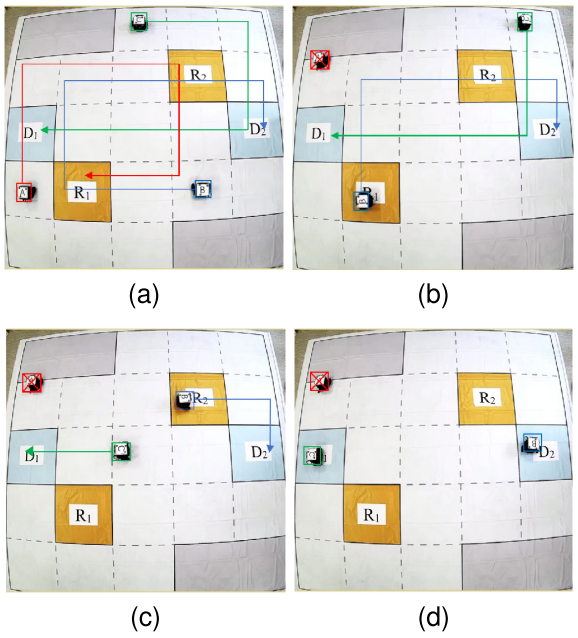


Fig. 4. Snapshots of a three-robot system, where robot **A** fails. (a) First snapshot. (b) Second snapshot. (c) Third snapshot. (d) Fourth snapshot.

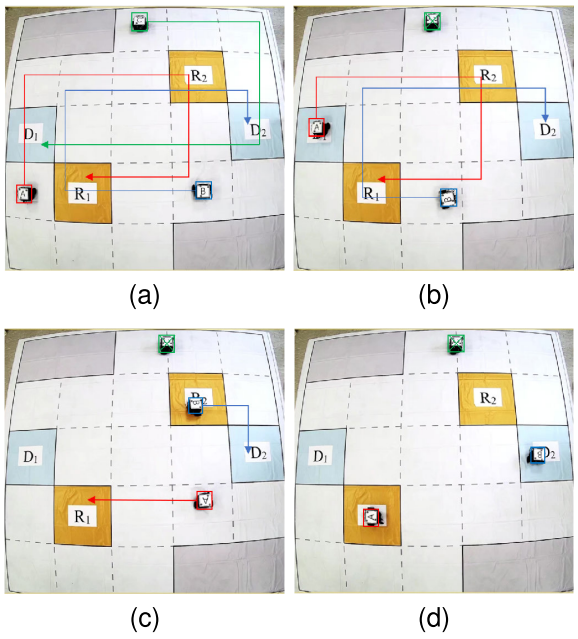


Fig. 5. Snapshots of a three-robot system, where robot **C** fails. (a) First snapshot. (b) Second snapshot. (c) Third snapshot. (d) Fourth snapshot.

following scLTL formula:

$$\mathbf{F}R_1 \wedge \mathbf{F}R_2 \wedge \mathbf{F}D_1 \wedge \mathbf{F}D_2.$$

First, we consider the situation where there are three robots **A**, **B**, and **C**, and at most one robot may fail. The snapshots of the experimental results are shown in Figs. 4 and 5. Specifically, Fig. 4 shows the case, where robot **A** failed midway. As shown in Fig. 4(a), robot **A** failed after visiting region D_1, resulting in the inability to visit regions R_1 and R_2. However, as seen in Fig. 4(b) and (c), robot **B** visited these two regions according to the plan, ensuring the successful completion of the overall task. On the other hand, Fig. 5 shows

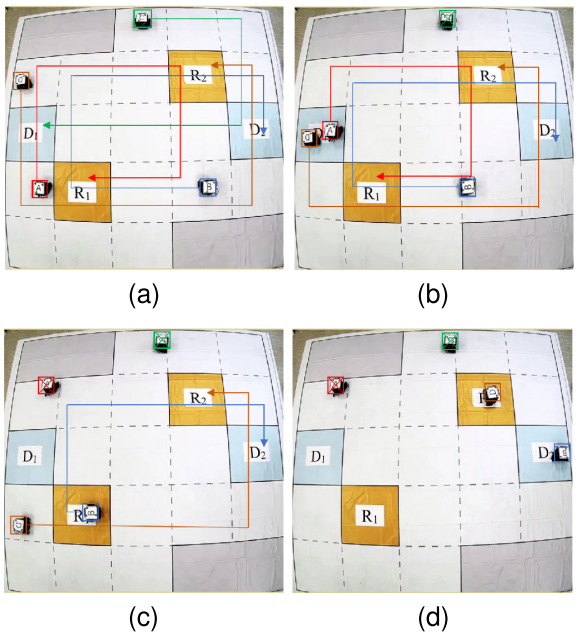


Fig. 6. Snapshots of a four-robot system, where two robots fail. (a) First snapshot. (b) Second snapshot. (c) Third snapshot. (d) Fourth snapshot.

a different case, where robot **C** failed initially. Similarly, the inability to visit regions D_1 and D_2 by robot **C**'s failure was compensated by robot **A** and **B**.

In the second example, we consider the situation where there are four robots, **A**, **B**, **C**, and **D**, and at most two robots may fail. The snapshots of the experimental results are shown in Fig. 6. Fig. 6(a) shows the failure-robust task planning for each robot, where every region is visited by at least three different robots. This requires the redundancy for robot executions. However, when some robots happen to fail as shown in Fig. 6(b) and (c), as long as the remaining robots in normal working conditions continue to execute tasks according to the original plan, the overall task can be ensured to fulfill.

VIII. Conclusion

In this article, we proposed a failure-robust LTL task planning approach to enhance the robustness of MRSs. The proposed task planning approach relies on MILP and takes the potential robot failures into account. In addition, we identified a scalable fragment of LTL formulae, called free-union-closed, to improve the computation efficiency of the synthesis procedure. We provided typical patterns and a systematic method for verifying whether an arbitrary LTL formula is free-union-closed or not. Numerical experiments were provided to illustrate the scalability of the proposed approach. Simulation case studies and real-world experiments were provided to illustrate the effectiveness of our approach. The limitation of our approach is still the scalability issue when the number of total robots or possible failures increases. In the future, we would like to investigate how to leverage recently developed scalable formal synthesis techniques, such as sampling-based approaches [30], [45] or Petri-net-based approaches [13], [46], to further mitigate the computational complexity.

REFERENCES

[1] Y. Tatsumoto, M. Shiraishi, K. Cai, and Z. Lin, "Application of online supervisory control of discrete-event systems to multi-robot warehouse automation," *Control Eng. Pract.*, vol. 81, pp. 97–104, Dec. 2018.

[2] M. P. Fanti, A. M. Mangini, G. Pedroncelli, and W. Ukovich, "A decentralized control strategy for the coordination of AGV systems," *Control Eng. Pract.*, vol. 70, pp. 86–97, Jan. 2018.

[3] F. Basile, P. Chiacchio, and E. Di Marino, "An auction-based approach to control automated warehouses using smart vehicles," *Control Eng. Pract.*, vol. 90, pp. 285–300, Sep. 2019.

[4] I. Kovalenko, D. Tilbury, and K. Barton, "The model-based product agent: A control oriented architecture for intelligent products in multi-agent manufacturing systems," *Control Eng. Pract.*, vol. 86, pp. 105–117, May 2019.

[5] T. Wang, P. Huang, and G. Dong, "Cooperative persistent surveillance on a road network by multi-UGVs with detection ability," *IEEE Trans. Ind. Electron.*, vol. 69, no. 11, pp. 11468–11478, Nov. 2022.

[6] P. Lv, G. Luo, Z. Ma, S. Li, and X. Yin, "Optimal multi-robot path planning for cyclic tasks using Petri nets," *Control Eng. Pract.*, vol. 138, Sep. 2023, Art. no. 105600.

[7] H. R. Karimi and Y. Lu, "Guidance and control methodologies for marine vehicles: A survey," *Control Eng. Pract.*, vol. 111, Jun. 2021, Art. no. 104785.

[8] Y. Tian, Y. Chang, F. Herrera Arias, C. Nieto-Granda, J. P. How, and L. Carlone, "Kimera-multi: Robust, distributed, dense metric-semantic SLAM for multi-robot systems," *IEEE Trans. Robot.*, vol. 38, no. 4, pp. 2022–2038, Aug. 2022.

[9] S. M. LaValle, *Planning Algorithms*, vol. 2. Cambridge, U.K.: Cambridge Univ. Press, 2006, pp. 3671–3678.

[10] R. Majumdar, K. Mallik, M. Salamati, S. Soudjani, and M. Zareian, "Symbolic reach-avoid control of multi-agent systems," in *Proc. ACM/IEEE 12th Int. Conf. Cyber-Phys. Syst.*, May 2021, pp. 209–220.

[11] Z. Huang, W. Lan, and X. Yu, "A formal control framework of autonomous vehicle for signal temporal logic tasks and obstacle avoidance," *IEEE Trans. Intell. Vehicles*, vol. 9, no. 1, pp. 1930–1940, Jan. 2024.

[12] M. Kloetzer and C. Mahulea, "LTL-based planning in environments with probabilistic observations," *IEEE Trans. Autom. Sci. Eng.*, vol. 12, no. 4, pp. 1407–1420, Oct. 2015.

[13] M. Kloetzer and C. Mahulea, "Path planning for robotic teams based on LTL specifications and Petri net models," *Discrete Event Dyn. Syst.*, vol. 30, pp. 79–88, Mar. 2020.

[14] Z. He, Y. Dong, G. Ren, C. Gu, and Z. Li, "Path planning for automated guided vehicle systems with time constraints using timed Petri nets," *Meas. Control*, vol. 53, nos. 9–10, pp. 2030–2040, 2020.

[15] M. Cai, S. Xiao, Z. Li, and Z. Kan, "Optimal probabilistic motion planning with potential infeasible LTL constraints," *IEEE Trans. Autom. Control*, vol. 68, no. 1, pp. 301–316, Jan. 2023.

[16] W. Shi, Z. He, W. Tang, W. Liu, and Z. Ma, "Path planning of multi-robot systems with Boolean specifications based on simulated annealing," *IEEE Robot. Autom. Lett.*, vol. 7, no. 3, pp. 6091–6098, Jul. 2022.

[17] S. Liu, A. Trivedi, X. Yin, and M. Zamani, "Secure-by-construction synthesis of cyber-physical systems," *Annu. Rev. Control*, vol. 53, pp. 30–50, Jan. 2022.

[18] X. Yu, X. Yin, S. Li, and Z. Li, "Security-preserving multi-agent coordination for complex temporal logic tasks," *Control Eng. Pract.*, vol. 123, Jun. 2022, Art. no. 105130.

[19] Z. Liu, M. Guo, and Z. Li, "Time minimization and online synchronization for multi-agent systems under collaborative temporal logic tasks," *Automatica*, vol. 159, Jan. 2024, Art. no. 111377.

[20] T. Yang, Y. Zou, S. Li, X. Yin, and T. Jia, "Signal temporal logic synthesis under model predictive control: A low complexity approach," *Control Eng. Pract.*, vol. 143, Feb. 2024, Art. no. 105782.

[21] C. Belta, B. Yordanov, and E. A. Gol, *Formal Methods for Discrete-time Dynamical Systems*, vol. 15. Berlin, Germany: Springer, 2017.

[22] C. Belta and S. Sadraddini, "Formal methods for control synthesis: An optimization perspective," *Annu. Rev. Control Robot. Auton. Syst.*, vol. 2, pp. 115–140, May 2019.

[23] A. Pacheck and H. Kress-Gazit, "Physically feasible repair of reactive, linear temporal logic-based, high-level tasks," *IEEE Trans. Robot.*, vol. 39, no. 6, pp. 4653–4670, Dec. 2023.

[24] W. Dong, X. Yin, and S. Li, "A uniform framework for diagnosis of discrete-event systems with unreliable sensors using linear temporal logic," *IEEE Trans. Autom. Control*, vol. 69, no. 1, pp. 145–160, Jan. 2024.

[25] K. Cho, J. Suh, C. J. Tomlin, and S. Oh, "Cost-aware path planning under co-safe temporal logic specifications," *IEEE Robot. Autom. Lett.*, vol. 2, no. 4, pp. 2308–2315, Oct. 2017.

[26] A. Sakakibara and T. Ushio, "On-line permissive supervisory control of discrete event systems for scLTL specifications," *IEEE Control Syst. Lett.*, vol. 4, no. 3, pp. 530–535, Jul. 2020.

[27] E. M. Wolff, U. Topcu, and R. M. Murray, "Optimization-based trajectory generation with linear temporal logic specifications," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2014, pp. 5319–5325.

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

[29] Y. Kantaros and M. M. Zavlanos, "Sampling-based optimal control synthesis for multirobot systems under global temporal tasks," *IEEE Trans. Autom. Control*, vol. 64, no. 5, pp. 1916–1931, May 2019.

[30] Y. Kantaros and M. M. Zavlanos, "STyLuS: A temporal logic optimal control synthesis algorithm for large-scale multi-robot systems," *Int. J. Robot. Res.*, vol. 39, no. 7, pp. 812–836, Jun. 2020.

[31] P. Schillinger, M. Bürger, and D. V. Dimarogonas, "Simultaneous task allocation and planning for temporal logic goals in heterogeneous multi-robot systems," *Int. J. Robot. Res.*, vol. 37, no. 7, pp. 818–838, Jun. 2018.

[32] P. Yu and D. V. Dimarogonas, "Distributed motion coordination for multirobot systems under LTL specifications," *IEEE Trans. Robot.*, vol. 38, no. 2, pp. 1047–1062, Apr. 2022.

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

[34] Y. Kantaros, S. Kalluraya, Q. Jin, and G. J. Pappas, "Perception-based temporal logic planning in uncertain semantic maps," *IEEE Trans. Robot.*, vol. 38, no. 4, pp. 2536–2556, Aug. 2022.

[35] D. Tian et al., "Two-phase motion planning under signal temporal logic specifications in partially unknown environments," *IEEE Trans. Ind. Electron.*, vol. 70, no. 7, pp. 7113–7121, Jul. 2023.

[36] J. Zhao, K. Zhu, S. Li, and X. Yin, "To explore or not to explore: Regret-based LTL planning in partially-known environments," *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 11337–11343, 2023.

[37] L. Zhou, V. Tzoumas, G. J. Pappas, and P. Tokekar, "Resilient active target tracking with multiple robots," *IEEE Robot. Autom. Lett.*, vol. 4, no. 1, pp. 129–136, Jan. 2019.

[38] L. Zhou, V. Tzoumas, G. J. Pappas, and P. Tokekar, "Distributed attack-robust submodular maximization for multirobot planning," *IEEE Trans. Robot.*, vol. 38, no. 5, pp. 3097–3112, Oct. 2022.

[39] Y. E. Sahin, P. Nilsson, and N. Ozay, "Multirobot coordination with counting temporal logics," *IEEE Trans. Robot.*, vol. 36, no. 4, pp. 1189–1206, Aug. 2020.

[40] F. Huang, X. Yin, and S. Li, "Failure-robust multi-robot tasks planning under linear temporal logic specifications," in *Proc. 13th Asian Control Conf. (ASCC)*, May 2022, pp. 1052–1059.

[41] F. Huang, S. Li, and X. Yin, "Synthesis of failure-robust plans for multi-robot systems under temporal logic specifications," in *Proc. IEEE 19th Int. Conf. Autom. Sci. Eng. (CASE)*, Aug. 2023, pp. 1–6.

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

[43] Y. E. Sahin, P. Nilsson, and N. Ozay, "Provably-correct coordination of large collections of agents with counting temporal logic constraints," in *Proc. ACM/IEEE 8th Int. Conf. Cyber-Physical Syst. (ICCPS)*, Apr. 2017, pp. 249–258.

[44] E. Rohmer, S. P. N. Singh, and M. Freese, "V-REP: A versatile and scalable robot simulation framework," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Nov. 2013, pp. 1321–1326.

[45] C. I. Vasile and C. Belta, "Sampling-based temporal logic path planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Nov. 2013, pp. 4817–4822.

[46] S. Hustiu, C. Mahulea, M. Kloetzer, and J.-J. Lesage, "On multi-robot path planning based on Petri net models and LTL specifications," *IEEE Trans. Autom. Control*, vol. 69, no. 9, pp. 6373–6380, Sep. 2024.

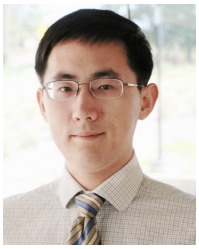
Bohan Cui (Student Member, IEEE) was born in Shandong, China, in 1999. He received the B.S. degree in automation from Harbin Institute of Technology, Harbin, China, in 2021. He is currently pursuing the Ph.D. degree in control science and engineering with the Department of Automation, Shanghai Jiao Tong University, Shanghai, China.

His current research interests include temporal logic task planning, discrete event systems, and game theory.

Feifei Huang (Student Member, IEEE) was born in Shanghai, China, in 1998. She received the B.S. degree in automation and the M.S. degree in control science and engineering from Shanghai Jiao Tong University, Shanghai, in 2021 and 2024, respectively.

Shaoyuan Li (Senior Member, IEEE) was born in Hebei, China, in 1965. He received the B.S. and M.S. degrees in automation from Hebei University of Technology, Tianjin, China, in 1987 and 1992, respectively, and the Ph.D. degree in automatic control theory and application from Nankai University, Tianjin, in 1997.

Since 1997, he has been with the Department of Automation, Shanghai Jiao Tong University, Shanghai, China, where he is currently a Professor. His current research interests include model predictive control, dynamic system optimization, and cyber-physical systems.



Xiang Yin (Member, IEEE) was born in Anhui, China, in 1991. He received the B.Eng. degree in electrical engineering from Zhejiang University, Zhejiang, China, in 2012, and the M.S. and Ph.D. degrees in electrical engineering from the University of Michigan, Ann Arbor, MI, USA, in 2013 and 2017, respectively.

Since 2017, he has been with the Department of Automation, Shanghai Jiao Tong University, Shanghai, China, where he is an Associate Professor. His research interests include formal methods, discrete-event systems, and cyber-physical systems.

Dr. Yin is a member of IEEE CSS Conference Editorial Board. He was a recipient of IEEE Conference on Decision and Control Best Student Paper Award Finalist in 2016. The Chair of the IEEE CSS Technical Committee on Discrete Event Systems and an Associate Editor for the *Journal of Discrete Event Dynamic Systems: Theory and Applications*.