Failure-Robust Multi-Robot Tasks Planning under Linear Temporal Logic Specifications

Feifei Huang, Xiang Yin and Shaoyuan Li

***Abstract*—In this work, we investigate the problem of multi-robot planning for tasks specified by linear temporal logic (LTL) formulas. The objective is to synthesize a reactive plan for the team of robots such that a global LTL task is fulfilled while minimizing the satisfaction time. We consider the scenario where some robots may fail during the execution so that they cannot contribute to the satisfaction of the task. Specifically, we assume that there are at most k robot that may fail. An effective task planning algorithm is presented that guarantees the robust satisfaction of the task under the upper-bounded number of failure robots. Our approach is based on constructing a team transition system that can capture both asynchronous motion of robots and possible robot failures. Then by utilizing value-iteration algorithm over the product of the task automata and the team transition system, a reactive and optimal strategy is computed. We illustrate the proposed algorithm by multi-robot coordination in grid-world.**

***Index Terms*—Multi-Robot Systems, Task Planning, Linear Temporal Logic, Failure Robustness**

I. Introduction

Task planning is one of the central problems in multi-robot systems, which have been widely used in nowadays cyber-physical engineering systems [2], [5], [27]. Traditional works on multi-robot planning mainly focus on low-level tasks such that collision avoidance and target point navigation [7], [14]. However, such low-level task specifications are not rich enough to support the increasing demand for complexity behaviors of the multi-robot system. Therefore, in the past years, task planning for high-level specifications has been becoming increasingly more important research topic in the multi-robot literature [13].

Formal methods is one of the most promising approaches for synthesizing high-level plans for multi-robot system to accomplish complex task. In particular, Linear Temporal Logic (LTL) [1] provides a well-structured, user-friendly and enough expressive way for specifying the complex task of the system. For example, it allows us to describe tasks such as "surveil a critical region *infinitely often*" or "keep searching for a target *until* it is found". In this work, we will focus on a specific fragment of LTL called co-safe LTL (scLTL), which is widely used to described tasks that should be satisfied within a finite horizon such as "*eventually* deliver the goods while avoiding obstacles".

This work was supported by the National Natural Science Foundation of China (62061136004, 62173226, 61833012) and the National Key Research and Development Program of China (2018AAA0101700).

F. Huang, X. Yin and S. Li are with Department of Automation and Key Laboratory of System Control and Information Processing, Shanghai Jiao Tong University, Shanghai 200240, China. E-mail: {huangfeifei,yinxiang,syli}@sjtu.edu.cn.

Due to its importance, multi-robot task planning for LTL specifications has drawn considerable attention in the literature; see, e.g., [6], [9], [10], [16], [17], [20], [22]–[24], [26]. Roughly speaking, depending on whether or not the multi-robot system is working in an uncertain environment, the coordination problem can be categorized as the *planning* problem and the *reactive synthesis* problem. In the task planning problem, it is assumed that there is no uncertainty in both the movements of the robots as well as the structure of the working space. Therefore, one needs to synthesize a plan, which is essentially an open-loop (possibly infinite) trajectory, for each robot to execute. Works along this direction can be found in, e.g., [8], [16]. In practice, due to external disturbances or adversarial attacks, the movements of robots may be uncertain. In this case, one needs to synthesize a reactive strategy that determines the action of each robot on-the-fly based on the executed trajectories [6], [12]. More recently, some works further consider the scenario where the workspace is partially-known [21] or completely unknown a priori [4], [10].

In this work, we consider the task planning problem where the workspace as well as the mobility of each robots are completely known and deterministic. Standard approach for solving the planning problem involves three steps [8], [16]. First, one needs to construct a deterministic team transition system by synchronizing the mobility model of each individual robot. Second, one needs to transform the LTL formula into a Büchi automaton and then product it with the team transition system. Finally, one needs to solve a graph search problem over the product system to order to generate an infinite trajectory satisfying the Büchi acceptance condition.

However, existing approaches for multi-robot task planning do not consider the issue of robot failures. In practice, when robots working in a severe environment, e.g., for rugged terrain search and rescue tasks, some robots in the team may either break physically or lose connections with the central station. Once a robot fails, it can no longer contribute to the satisfaction of the global task. Therefore, to guarantee the *robust satisfaction* of the overall task, one needs to plan ahead, at the offline synthesis stage, to make the generated plan have certain level of robust degree so that the satisfaction of the overall task cannot be affected when a small number of robots fail.

In this work, we propose a framework for synthesizing robust plans for multi-robot systems under possible robot failures. Specifically, we assume that there are at most k robots that could fail among the all N robots. Whenever the decision-maker detects the failure of some robot, it will update the global plan to incorporate with the failure. The

Authorized licensed use limited to: Shanghai Jiaotong University. Downloaded on July 22,2022 at 00:47:10 UTC from IEEE Xplore. Restrictions apply.

generated reactive plans ensure the satisfaction of the scLTL task even under the worst scenario where arbitrary, but at most k, robots fail. Our approach is based on building a non-deterministic transition system model that captures all possible failures of each robot. Then by taking value iterations over the product of the proposed non-deterministic system with the finite-state automaton capturing the scLTL specification, an effective reactive planning strategy is obtained that solves the problem.

We note that, the robustness issue has already been discussed in the literature in the context of formal methods in robotics. For example, [18] discussed how to synthesize plans that are robust to timing errors during the online deployment. In [21], the authors considered the robustness issue against both environmental disturbances and modeling errors. The work of [15] considered the un-modeled but bounded disturbance and ensured that the behaviours of the system remain close to the expectation. Also, [25] proposed how to synthesize feedback control strategies robust to small perturbations. However, none of the above works considers the robustness issue against individual robot failures.

The rest of this paper is organized as follows. In Section II, a brief introduction to the basic knowledge and concepts appeared in this paper is given. In Section III, we describe the details of system setup and provide a formal problem statement. In Section IV, we propose the synthesis method for this problem. In Section V, an example to illustrate the resulting strategy generated by our method is given. We conclude our contribution and discuss some future work in Section VI.

II. Preliminary

Some basic concepts and definitions are reviewed in this section, including transition systems, linear temporal logic and finite state automata. They will be used in the system modeling and the problem's solution.

A. Transition systems

Transition systems are general enough to capture the behavior of systems. It is used as a modeling formalism in generating the control strategy from a specification. We model each single robot as a weighted transition system in this work.

Definition 1 (Weighted Transition System). *A weighted Transition System, denoted by* $\mathbf{wTS}$, *is a 6-tuple* $\left(\mathcal{Q}, q^0, \rightarrow, w, \mathcal{AP}, \mathcal{L}\right)$, *where:*

- $\mathcal{Q}$ *is the set of states;*
- $q^0 \in \mathcal{Q}$ *is the initial state;*
- $\rightarrow \subseteq \mathcal{Q} \times \mathcal{Q}$ *is the transition relation. If there is a transition relation from state* q *to* q', *then* $(q, q') \in \rightarrow$, *or denoted as* $q \rightarrow q'$;
- $w : \mathcal{Q} \times \mathcal{Q} \rightarrow \mathbb{R}^+$ *is cost function;*
- $\mathcal{AP}$ *is the set of atomic propositions;*
- $\mathcal{L} : \mathcal{Q} \rightarrow 2^{\mathcal{AP}}$ *is a labelling function giving the set of atomic propositions that can be satisfied in a state.*

An infinite *trajectory* of $\mathbf{wTS}$ is an infinite sequence $\tau = q^0 q^1 \cdots$ such that $q^t \rightarrow q^{t+1}, \forall t \geq 0$. Besides, a trajectory can generate a *trace* as a sequence of atomic propositions, i.e., trace $(\tau) = \mathcal{L}\left(q^0\right) \mathcal{L}\left(q^1\right) \mathcal{L}\left(q^2\right) \cdots$, which is an infinite word over $2^{\mathcal{AP}}$. We denote trace $(\mathbf{wTS})$ as the set of all traces that can be generated from the initial state of the weighted transition system $\mathbf{wTS}$.

B. Linear temporal logic

Linear temporal logic (LTL) formula can express temporal properties in a structured, user-friendly and rigorous manner. In this paper, we use co-safe LTL formula (scLTL) to describe the global task for the multi-robot system, which is fragments of LTL.

Definition 2 (co-safe LTL Syntax). *A co-safe linear temporal logic (scLTL) formula* ϕ *over a given set of atomic propositions* $\mathcal{AP}$ *is defined recursively as following step:*

- $\top$ *is scLTL formula;*
- *atomic proposition* $\alpha \in \mathcal{AP}$ *is scLTL formula;*
- *if* ϕ *is scLTL formula,* $\neg\phi, \mathbf{X}\phi, \mathbf{F}\phi$ *are scLTL formulas;*
- *if* ϕ_1 *and* ϕ_2 *are scLTL formulas, then* $\phi_1 \wedge \phi_2, \phi_1 \vee \phi_2$ *and* $\phi_1 \mathbf{U} \phi_2$ *are all scLTL formulas.*

Specifically, $\top$ is predicate true, and $\neg$ (negation), $\wedge$ (conjunction), $\vee$ (disjunction) are standard Boolean operators. Temporal operator $\mathbf{X}\phi$ holds true at the current moment, if ϕ holds in the next moment. Formula $\phi_1 \mathbf{U} \phi_2$ holds true at the current moment, if there is some future moment for which ϕ_2 holds and ϕ_1 holds at all moments until that future moment. Formula $\mathbf{F}\phi = \top \mathbf{U} \phi$ holds true at the current moment if ϕ eventually holds true sometime in the future.

Different from full LTL formula, temporal operator $\mathbf{G}\phi = \neg\mathbf{F}\neg\phi$ can not be expressed in scLTL, because it requires ϕ *always* to be true, which can only be satisfied in infinite time.

Though scLTL formulas are interpreted over infinite words, it should be guaranteed in finite time [3]. An infinite word satisfies a scLTL formula if it contains a finite "good" prefix, which satisfies the finite horizon specification. We define $Words\left(\phi\right)$ as the set of words that satisfy the scLTL formula ϕ. Recall that the trace generated by trajectory of a transition system is an infinite word. If all traces generated from the initial state of weighted transition system $\mathbf{wTS}$ satisfy scLTL formula ϕ, i.e., trace$(\mathbf{wTS}) \subseteq Words(\phi)$, we say $\mathbf{wTS}$ satisfies ϕ, denoted as $\mathbf{wTS} \models \phi$.

C. Finite state automata

A scLTL formula ϕ can be translated into a deterministic finite state automata (FSA), which can accept all and only words with "good" prefixes of ϕ. Model checking technique for FSA is one type of standard approaches evaluating the reachability property of the system [11].

Definition 3 (Finite State Automata). *A (deterministic) finite state automata (FSA) is a tuple* $\mathcal{A} = \left(\mathcal{Q}_{\mathcal{A}}, q^0_{\mathcal{A}}, \Sigma_{\mathcal{A}}, \delta_{\mathcal{A}}, \mathcal{F}_{\mathcal{A}}\right)$, *where:*

- $\mathcal{Q}_{\mathcal{A}}$ *is a finite set of states;*

Authorized licensed use limited to: Shanghai Jiaotong University. Downloaded on July 22,2022 at 00:47:10 UTC from IEEE Xplore. Restrictions apply.

- $q_{\mathcal{A}}^0 \in \mathcal{Q}_{\mathcal{A}}$ *is the initial state;*
- $\Sigma_{\mathcal{A}}$ *is an input alphabet;*
- $\delta_{\mathcal{A}} : \mathcal{Q}_{\mathcal{A}} \times \Sigma_{\mathcal{A}} \rightarrow \mathcal{Q}_{\mathcal{A}}$ *is a deterministic transition function;*
- $\mathcal{F}_{\mathcal{A}} \subseteq \mathcal{Q}_{\mathcal{A}}$ *is a set of accepting states.*

A *run* of $\mathcal{A}$ over an input word $\pi = \pi^0\pi^1 \cdots \pi^l$, where $\pi^i \in \Sigma_{\mathcal{A}}(\ \forall i = 0, 1 \cdots l\ )$, is a sequence $q_{\mathcal{A}}^0 q_{\mathcal{A}}^1 \cdots q_{\mathcal{A}}^l q_{\mathcal{A}}^{l+1}$ such that $\delta_{\mathcal{A}}\left(q_{\mathcal{A}}^i, \pi^i\right) = q_{\mathcal{A}}^{i+1}(\ \forall i = 0, 1 \cdots l\ )$. We say FSA $\mathcal{A}$ accepts a word over $\Sigma_{\mathcal{A}}$ if and only if the corresponding run sequence ends in a state $q_{\mathcal{A}} \in \mathcal{F}_{\mathcal{A}}$.

III. Problem Statement

In this section, we will provide the details of how the multi-robot system operates and the scenario where some robots may fail. Some necessary assumptions are presented. Also, we will give a formal statement of the control synthesis problem that need to be solved.

A. Multi-robot system

Consider N mobile robots as a multi-robot system evolved in a common workspace. There are several regions of interest in the workspace. When one of the robots arrives at a region of interest, it can choose to perform specific action, thus satisfying some atomic proposition(s).

The workspace is denoted as directed paragraph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{AP}, \mathcal{L}_{\mathcal{G}})$, which includes the set of regions $\mathcal{V}$, the set of paths linking the regions $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$, a finite set of atomic propositions $\mathcal{AP}$, and a map function $\mathcal{L}_{\mathcal{G}} : \mathcal{V} \rightarrow 2^{\mathcal{AP}}$ giving the set of atomic propositions that can be satisfied at a region.

The robots' motion and execution ability are abstracted into weighted transition system as defined in Definition 1. The ith robot is represented as $\mathbf{wTS}_i = \left(\mathcal{Q}_i, q_i^0, \rightarrow_i, w_i, \mathcal{AP}_i, \mathcal{L}_i\right)$, where $\mathcal{Q}_i \subseteq \mathcal{V}$, $\rightarrow_i \subseteq \mathcal{E}$, $\mathcal{AP}_i \subseteq \mathcal{AP}$ and $\mathcal{L}_i(q) \subseteq \mathcal{L}_{\mathcal{G}}(q)$ $(\forall q \in \mathcal{Q}_i)$. Define the cost function w_i, which depends on the time cost traveling from one region in the workspace to another region.

The N-robot system needs to accomplish a high-level propositional task, specified by co-safe LTL formula, which means that the word generated by the system should satisfy the scLTL formula in finite horizon. As mentioned in Definition 2, scLTL formulas are composed of several atomic propositions, Boolean and temporal operators. In this problem, each atomic proposition is related to region(s) in the workspace and specific action. The satisfaction of atomic proposition requires robot to visit one of the corresponding regions and perform the specific action, e.g., upload data, collect data, etc.

The task is global and robots in the system collaborate for this common task. A centralized controller for this multi-robot system is needed, which apply control inputs to all the robots. Besides, in order to save time, energy or other resources, we hope that the robots' team can complete the task as soon as possible, which requires an optimal controller to be designed.

Note that we omit the low-level, dynamic motion planner in this study, and focus on designing a centralized high-level controller that drives the multi-robot system to satisfy a given task under LTL specification.

Based on the assumption that robots can always work normally according to the control input given to them, the solution would be straightforward. We can construct a product transition system that captures the joint behavior of all the robots and compute a run corresponding to the optimal path by utilizing graph-searching technique.

However, if some robot does not follow the control input, or even worse, lose the ability to move or perform any action, the joint behavior of the robots can not be captured by simply taking the parallel composition of the individual transition system. Thus we introduce the failure robustness property of the control strategy.

B. Failure robustness

We consider the scenario where some robots may fail to work during the execution, which are called "broken". Broken robots can not contribute to the satisfaction of the LTL specification task any more. The designed controller should be robust to the robot failure, i.e., drive the robots' team to accomplish the mission even when some robots lose their motion or execution ability.

To avoid the situation where all the robots are broken and lead to the inevitable failure of the high-level mission, we limit an upper bound on the number of failure robots, called *robust coefficient*. When a robot breaks down or the probability of robot failure is not restricted. For N-robot system, robust coefficient is equal to k means that there are at most k robots broken in the total N robots during the whole execution. This restriction is realistic, since the robots' failure does not happen very often. On the other hand, higher robust coefficient means more robot failures are allowed, indicating that the control policy holds stronger failure robustness.

Considering robot's failure, the transition relation of the multi-robot system will become nondeterministic. Different from the settings in [19] [9] that the uncertainty of the system is embodied in the workspace properties, direction or speed of robots' movement, here we treat the system as nondeterministic due to the scenario of robots' failure. When the centralized controller puts control input to the system, each single robot either follows the input signal and reaches the target location, or it stays where it is (which means the robot is broken).

The following assumptions are introduced:

- A robot can be accurately controlled when it is under normal working condition, but when it meets failure, it *permanently loses* the ability to move or perform any action.
- The controller has a complete knowledge of the robots' working conditions, i.e., whether the robots are under normal working condition or broken are monitored by the controller. This ensures that once a robot is broken, the controller will catch this message and update the plan.
- The broken robot will not affect other robots, e.g., the situation of blocked path is not discussed here.
- Whether a robot can work normally or not has no relation with other robots, unless the number of the broken robots

Authorized licensed use limited to: Shanghai Jiaotong University. Downloaded on July 22,2022 at 00:47:10 UTC from IEEE Xplore. Restrictions apply.

has reached the upper bound (i.e. robust coefficient k). After the number reaches robust coefficient, there will be no more broken robots, and the rest of robots can always follow the control input to visit regions in the workspace.

From the prospect of a single robot, since the relation between the control input and the transition of states is deterministic, the set of control input signal can be omitted in the robot's transition system. We emphasize that the determinism mentioned here is not contradict to the uncertainty of the system as we mentioned above. The latter must be taken into account when multiple robots perform a global task as a team.

C. Synthesis objective

To accomplish the complex logical task and the failure-robust property, our aim is to design a centralized controller that can send control input signal to each robot.

Based on the multi-robot system and the robot-failure setup, we propose three requirements for the centralized controller:

- Control the multi-robot system to satisfy the given global LTL task;
- Ensure that the system can complete the task when at most k (robust coefficient) robots are broken;
- Spend minimum time to complete the task in the worst case, which means that the number of failure robots reaches the upper bound k.

Here, the first requirement claims basic feasibility, the second requirement claims failure-robustness of the controller, and the third requirement further asks for optimality.

In brief, the controller need to plan trajectories for all the robots and react to the robot failure event, therefore, it follows a reactive strategy. We define the reactive strategy as a time-invariant map from the robots' states (including their position and working condition information) to the next states to be visited. The detailed form of the strategy will be given in Section IV.

We then address the following problem statement:

Problem 1. *In the workspace modeled as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{AP}, \mathcal{L}_{\mathcal{G}})$, given a team of N robots represented as weighted transition systems $\{\mathbf{wTS}_1 \cdots \mathbf{wTS}_N\}$, a global scLTL formula task ϕ over $\mathcal{AP}$ and robust coefficient k ($0 \leq k < N$), generate a control reactive strategy for the multi-robot system that can fulfill the task ϕ while in the worst case there are at most k robots broken, and minimize the total time cost in that case.*

IV. Solution

We can conclude from Section III that the system is nondeterministic, as whether the robots perform the planned trajectory or which robot(s) fail to work after the controller sends the input signal is not deterministic. To solve the problem, we first modify the weighted transition system to capture both the asynchronous motion of robots and possible failures of the multi-robot system, which is called *Nondeterministic Transition System*. Then construct a product automata based on the *Nondeterministic Transition System* and the finite state automata translated from the scLTL formula. Therefore, we can solve the control synthesis problem by using value-iteration algorithm on the product automata. We design a centralized controller that have a complete knowledge of the workspace and the working condition of robots.

A. Nondeterministic characteristic of system

To characterize the setup that robot can not move any more after robot failure happened, we modify the weighted transition system of each robot into a Nondeterministic transition system. The augmented transition system can capture robot failure. Inspired by [**?**], we construct team transition system and plan the team of robots in an asynchronization way.

Definition 4 (Nondeterministic Transition System)**.** *A Nondeterministic transition system ($\mathbf{NTS}$) for robot r_i, based on weighted transition system $\mathbf{wTS}_i = \big(\mathcal{Q}_i, q_i^0, \rightarrow_i, w_i, \mathcal{AP}_i, \mathcal{L}_i\big)$, is denoted by the tuple $\mathbf{NTS}_i = \big(\mathcal{Q}'_i, q_i^0, \rightarrow'_i, w'_i, \mathcal{AP}_i, \mathcal{L}'_i\big)$, where:*

- $\mathcal{Q}'_i = \mathcal{Q}_i \cup \{b\}$ is the new set of states, here state b represents the broken state of robot r_i;
- q_i^0 is the initial state. Suppose that robot is not broken at first, so the definition of q_i^0 is the same as in $\mathbf{wTS}$;
- $\rightarrow'_i \subseteq \mathcal{Q}'_i \times \mathcal{Q}'_i$ is the transition relation. $(q_i, \hat{q}_i) \in \rightarrow'_i$ if and only if one of the following conditions is true: 1) $(q_i, \hat{q}_i) \in \rightarrow_i$; 2) $(q_i, \hat{q}_i) = (b, b)$; 3) $q_i \in \mathcal{Q}_i$ and $\hat{q}_i = b$. Specially, $(b, q_i) \notin \rightarrow'_i$ for any $q_i \in \mathcal{Q}_i$;
- $w'_i : \mathcal{Q}'_i \times \mathcal{Q}'_i \rightarrow \mathbb{R}$ is the cost function, and

$$w'_i(q_i, \hat{q}_i) = \begin{cases} w_i(q_i, \hat{q}_i) & \text{if } (q_i, \hat{q}_i) \in \rightarrow_i \\ \infty & \text{if } (q_i, \hat{q}_i) = (b, b) \\ 0 & \text{if } q_i \in \mathcal{Q}_i \text{ and } \hat{q}_i = b \end{cases} \tag{1}$$

- $\mathcal{AP}_i$ are basically similar to the definition of $\mathcal{AP}_i$ in $\mathbf{wTS}$;
- $\mathcal{L}'_i : \mathcal{Q}'_i \rightarrow 2^{\mathcal{AP}_i}$ is the labeling function,

$$\mathcal{L}'_i(q_i) = \begin{cases} \mathcal{L}_i(q_i) & \text{if } q_i \in \mathcal{Q}_i \\ \emptyset & \text{if } q_i = b \end{cases} \tag{2}$$

A vital difference between $\mathbf{wTS}$ and $\mathbf{NTS}$ is that, in $\mathbf{wTS}$ the controller can explicitly drive the robot system to a desired state (which should satisfy the transition relation). However, each control input in $\mathbf{NTS}$ may have nondeterministic outcome, i.e., the desired state or the broken state b.

For simplicity of expression, we use elements with no prime to represent Nondeterministic transition system instead of weighted Transition System in the following contents when there is no ambiguity.

B. Asynchronous motion of robots

To synthesize the multi-robot system, a common way is to make a team transition system that captures states of each robot. Basically, the synthesizing approach can be classified as synchronous way and asynchronous way. The synchronous motion of the robots' team means that all of the robots start a step simultaneously. Because some robots have to travel longer

Authorized licensed use limited to: Shanghai Jiaotong University. Downloaded on July 22,2022 at 00:47:10 UTC from IEEE Xplore. Restrictions apply.

distance, the others must wait. This synthesizing approach will cause the waste of robot resource.

Inspired by [19], we introduced *traveling state* into the team transition system states, to capture the system states at the instant that some robots are at regions of interest while others are traveling between the regions. Adding *traveling state* into the team transition system will benefit to synthesize the multi-robot system in an asynchronous way and save the time that robots wait for synchronization, thus proving the efficiency of robots' motion.

Definition 5 (Traveling State)**.** *Given Nondeterministic transition system* $\mathbf{NTS}_i = \left(\mathcal{Q}_i, q_i^0, \rightarrow_i, w_i, \mathcal{AP}_i, \mathcal{L}_i\right)$*, which models the motion and execution ability of robot* r_i*.* $(q_i, s_i, \hat{q_i}) \in \mathcal{Q}_i \times \mathbb{N} \times \mathcal{Q}_i$ *is called a* traveling state *if* $(q_i, \hat{q_i}) \in \rightarrow_i$*,* $0 < s_i < w_i(q_i, \hat{q_i})$*.*

Intuitively, $(q_i, s_i, \hat{q_i})$ represents the state of robot r_i at instant when r_i is traveling from q_i to $\hat{q_i}$ and remains s_i time to reach $\hat{q_i}$. As mentioned in Section III-A, the cost function in this paper represents the total time it takes from one state to another, so the remained time s_i is constrained by $0 < s_i < w_i(q_i, \hat{q_i})$.

Compared with *traveling state*, *regular state* denotes the instant that robot has reached the region and is ready to perform specific atomic proposition. Unlike [19], here we consider a unified form for these two kind of states. *Regular state* is denoted as $(q_i, s_i, \hat{q_i})$, where $s_i = 0$, indicating robot r_i is at state $\hat{q_i}$.

Define $\#_b$ as a function mapping the amount of broken robots in the system. Let $q = ((q_1, s_1, \hat{q_1}), (q_2, s_2, \hat{q_2}) \cdots (q_N, s_N, \hat{q_N}))$, where $(q_i, s_i, \hat{q_i})$ is the state of robot r_i (*traveling* or *regular states*), $\#_b(q)$ is equal to the number of $q_i = b (i = 1, 2 \cdots N)$ in q.

Now we can construct the team transition system, which captures the asynchronous motion of robots and possible robot failures.

Definition 6 (Team Transition System)**.** *Based on Nondeterministic transition system* $\mathbf{NTS}_1, \mathbf{NTS}_2 \cdots \mathbf{NTS}_N$*, team transition system (*$\mathbf{TTS}$*) is a tuple* $(\tilde{\mathcal{Q}}, q^0, \rightarrow, w, \mathcal{AP}, \mathcal{L}, k)$*, where:*

- k *denotes robust coefficient;*
- $\tilde{\mathcal{Q}} \subseteq (\mathcal{Q}_1 \times \mathbb{N} \times \mathcal{Q}_1) \times (\mathcal{Q}_2 \times \mathbb{N} \times \mathcal{Q}_2) \times (\mathcal{Q}_3 \times \mathbb{N} \times \mathcal{Q}_3) \cdots (\mathcal{Q}_N \times \mathbb{N} \times \mathcal{Q}_N)$ *is the set of states, where* $(\mathcal{Q}_i \times \mathbb{N} \times \mathcal{Q}_i)$ *is the state of robot* r_i*, either traveling state (defined in Definition 5) or regular state.* $\#_b(q) \leq k, \forall q \in \tilde{\mathcal{Q}}$*;*
- $q^0 = \left((q_1^0, 0, q_1^0), (q_2^0, 0, q_2^0) \cdots (q_N^0, 0, q_N^0)\right) \in \tilde{\mathcal{Q}}$ *is the initial state;*
- $\rightarrow \subseteq \tilde{\mathcal{Q}} \times \tilde{\mathcal{Q}}$ *is the transition relation. We show which relation belongs to the set below:*

  Let $q = ((q_1, s_1, \hat{q_1}), (q_2, s_2, \hat{q_2}) \cdots (q_N, s_N, \hat{q_N})), q' = ((q_1', s_1', \hat{q_1}'), (q_2', s_2', \hat{s_2}') \cdots (q_N', s_N', \hat{q_N}'))$.

  $(q, q') \in \rightarrow$ *if and only if for any* $i = 1, 2 \cdots N$*,*

$$(q_i, \hat{q_i}) \in \rightarrow_i \ \textit{and} \begin{cases} \hat{q_i} = q_i' & \textit{if } s_i = 0 \\ q_i = q_i', \hat{q_i} = \hat{q_i}' & \textit{if } s_i \neq 0 \end{cases} \quad (3)$$

$$\textit{Let } t_i = \begin{cases} s_i & \textit{if } s_i \neq 0 \\ w(q_i', \hat{q_i'}) & \textit{if } s_i = 0 \end{cases}, \ T = \min_{i=1,2\cdots N, t_i \neq 0} t_i, \quad (4)$$

  then $s_i' = \max\{t_i - T, 0\}$.
  It is easy to see that there exists at least one robot r_i *that is in a* regular state *in* q' *, i.e.,* $s_i' = 0$.
- $w : \tilde{\mathcal{Q}} \times \tilde{\mathcal{Q}} \rightarrow \mathbb{R}^+$ *is the cost function. Let* $w(q, q')$ *equal to* T *in (4);*
- $\mathcal{AP} = \cup_{i=1}^n \mathcal{AP}_i$ *is the set of atomic propositions;*
- $\mathcal{L}$ *is the labeling function. Let* $\mathcal{L}(q) = \cup_{s_i=0} \mathcal{L}_i(\hat{q_i})$*, where* $q = ((q_1, s_1, \hat{q_1}), (q_2, s_2, \hat{q_2}) \cdots (q_N, s_N, \hat{q_N}))$.

Let the centralized controller check each robot's working status before the state of team transition system is going to be changed, i.e., some robots are about to reach *regular state*. The reasons why this scheme does not affect system operation are as follows:

The traveling process of a robot from one region to the next can be considered open-loop. As stated in Section III-B, robots under normal operating condition can precisely move as given speed and direction. Therefore, the controller only need to check whether robot meets failure when the system state is going to change, i.e. some robots are going to reach their *regular states* and perform atomic propositions. Though this scheme may cut down system efficiency (because the controller does not re-plan the trajectory at first time when the robot breaks), it will signally reduce model complexity, as robots are not allowed to back out on the way. Also, the controller does not need to monitor the working status of the robots all the time. We make this scheme as a trade off between complexity and efficiency.

We now verify that the transition relation of the broken states is satisfied in the constructed team transition system.

Suppose $q, q' \in \tilde{\mathcal{Q}}$ and $(q, q') \in \rightarrow$, where $q = ((q_1, s_1, \hat{q_1}), (q_2, s_2, \hat{q_2}) \cdots (q_N, s_N, \hat{q_N}))$ and $q' = ((q_1', s_1', \hat{q_1}'), (q_2', s_2', \hat{q_2}') \cdots (q_N', s_N', \hat{q_N}'))$. Consider two situations that robot r_i is broken:

1)$s_i = 0$. This happens when robot r_i is broken during the traveling to state $\hat{q_i}$. According to the control scheme, the controller does not identify the robot failure until some robot reach *regular state*. Here, it is the broken robot that reached *regular state* first . It doesn't matter what the origin $\hat{q_i}$ was since it will change to b when $s_i = 0$. According to (3), $\hat{q_i} = q_i' = b$; and Definition 4, $\hat{q_i'} = b$. Thus, $t_i = w(q_i', \hat{q_i'}) = w(b, b) = \infty$, it won't influence T and $r_i' = \infty$. The robot r_i will never leave state b, which conforms to the fact that it's permanently broken.

2)$s_i \neq 0$. This means that robot r_i is broken when other robots reach *regular state*. The controller will update the state of team transition system before the atomic proposition is performed (now r_i is in state b), and the change of state $\hat{q_i}$

Authorized licensed use limited to: Shanghai Jiaotong University. Downloaded on July 22,2022 at 00:47:10 UTC from IEEE Xplore. Restrictions apply.

won't affect the labeling atomic propositions. At next step, $q_i = q_i', \hat{q}_i = \hat{q_i'} = b, w(q_i, q_i') = 0, s_i = 0$, then it is the situation 1).

C. Construction of product automata

It is a common way to construct a product automata that can both be generated by the robots' team and satisfy the temporal logic task to solve the task planning problem. Before we give the definition of product automata, we introduce the *uncontrollable state set* to define states that the system *may reach* under the same control input due to the system uncertainty.

Definition 7 (Uncontrollable State Set). *Given $q, q' \in \tilde{Q}, q' = ((q_1', s_1', \hat{q_1}'), (q_2', s_2', \hat{q_2}') \cdots (q_N', s_N', \hat{q_N}'))$, $(q, q') \in \to$, the uncontrollable state set for transition (q, q') is denoted by $\mathbb{U}(q, q') = \{\bar{q} \mid (q, \bar{q}) \in \to$, $\bar{q} = ((\bar{q_1}, \bar{s_1}, \hat{\bar{q}}_1), (\bar{q_2}, \bar{s_2}, \hat{\bar{q}}_2) \cdots (\bar{q_N}, \bar{s_N}, \hat{\bar{q}}_N))$, $\forall i = 1, 2 \cdots N$, $\bar{q_i} = b$ or $\bar{q_i} = q_i'\}$.*

Note that $\#_b(q') \leq \#_b(\bar{q}) \leq k, \forall \bar{q} \in \mathbb{U}(q, q')$.

Intuitively speaking, when the system is trying to move from q to q', it may nondeterministically fall into the state in $\mathbb{U}(q, q')$ as some robots may fail to move.

Definition 8 (Product Automata). *The product automata $\mathcal{P}$ between the team transition system $\mathbf{TTS} = (\tilde{Q}, q^0, \to, w, \mathcal{AP}, \mathcal{L}, k)$ and the finite state automata $\mathcal{A}_\phi = (\mathcal{Q}_\mathcal{A}, q^0_\mathcal{A}, \Sigma_\mathcal{A}, \delta_\mathcal{A}, \mathcal{F}_\mathcal{A})$, is defined as the tuple $\mathcal{P} = (S_\mathcal{P}, S_{\mathcal{P},0}, \delta_\mathcal{P}, \mathcal{F}_\mathcal{P}, w_\mathcal{P}, \mathcal{AP})$, consisting of*

- *a finite set of states $S_\mathcal{P} = \tilde{Q} \times \mathcal{Q}_\mathcal{A}$;*
- *a set of initial states $S^0_\mathcal{P} = q^0 \times q^0_\mathcal{A}$;*
- *a transition relation $\delta_\mathcal{P} \subseteq S_\mathcal{P} \times S_\mathcal{P}$, where $((q, q_\mathcal{A}), (\hat{q}, \hat{q_\mathcal{A}})) \in \delta_\mathcal{P}$ if and only if $(q, \hat{q}) \in \to$, $(q_\mathcal{A}, \mathcal{L}(q), \hat{q_\mathcal{A}}) \in \delta_\mathcal{A}$;*
- *a set of accepting states $\mathcal{F}_\mathcal{P} = \tilde{Q} \times \mathcal{F}_\mathcal{A}$;*
- *cost function $w_\mathcal{P} : S_\mathcal{P} \times S_\mathcal{P} \to \mathbb{R}^+$, $w_\mathcal{P}((q, q_\mathcal{A}), (q', q'_\mathcal{A})) = w(q, q')$;*
- *a set of atomic proposition $\mathcal{AP}$, the same as in $\mathbf{TTS}$.*

Similarly, we define the uncontrollable set $\mathbb{U}$ of the state of product automata as $\mathbb{U}((q, q_\mathcal{A}), (\hat{q}, \hat{q_\mathcal{A}})) = \{(\bar{q}, \bar{q_\mathcal{A}}) \mid \bar{q} \in \mathbb{U}(q, q'),\ ((q, q_\mathcal{A}), (\bar{q}, \bar{q_\mathcal{A}})) \in \delta_\mathcal{P}\}$.

In order to maintain the unity of symbol, those unreachable states need to be removed from the automata when constructing the product automata. This prune process can be realized by a simple graph search program. Note that the product automata mentioned in the following paragraph is assumed to have already completed this pruning process.

The product of the finite state automata and the team transition system accepts all the words that can both satisfy the scLTL formula task and be generated by the multi-robot system. Therefore, Problem 1 can be transformed to the problem of computing an optimal control strategy for $\mathcal{P}$ which minimizes the time cost of reaching a state in $\mathcal{F}_\mathcal{P}$ from $S^0_\mathcal{P}$ in the worst case that there are k broken robots.

D. Task planning algorithm

Because scLTL formula can be verified in finite horizon, at least one of accepting states of the product automata should be visited in finite horizon. Thus, the optimal strategy for the product automata can be solved by value-iteration method. The main idea of this method is to iteratively compute the cost of each available successor state at each state. By always choosing the optimal successor, it will finally converge to the optimal strategy.

Let $J^i(s_\mathcal{P}, \hat{s_\mathcal{P}})$ denote the cost of controlling the system to state $\hat{s_\mathcal{P}} \in S_\mathcal{P}$ from state $s_\mathcal{P} \in S_\mathcal{P}$, $(s_\mathcal{P}, \hat{s_\mathcal{P}}) \in \delta_\mathcal{P}$, and μ^i_P denote the respective strategy at the ith iteration. We also define an arbitrarily small threshold $\theta \geq 0$.

The value-iteration process starts from the accepting states. First, set $J^0(s_\mathcal{P}, \hat{s_\mathcal{P}}) = 0$ for all $s_\mathcal{P} \in \mathcal{F}_\mathcal{P}$ and $J^0(s_\mathcal{P}, \hat{s_\mathcal{P}}) = \infty$ for all $s_\mathcal{P} \in \mathcal{S}_\mathcal{P} \backslash \mathcal{F}_\mathcal{P}$. The initial strategy is achieved by arbitrarily selecting available successor states for each state in $S_\mathcal{P}$. At the ith iteration, the cost of controlling the system to state $\hat{s_\mathcal{P}}$ at each state $s_\mathcal{P} \in S_\mathcal{P}$ is updated, where $J^i(s_\mathcal{P}, \hat{s_\mathcal{P}}) =$

$$\begin{cases} 0 & \\ & \text{for } s_\mathcal{P} \in \mathcal{F}_\mathcal{P} \\ \max\limits_{\bar{s_\mathcal{P}} \in \mathbb{U}(s_p, \hat{s_p})} \{w_\mathcal{P}(s_\mathcal{P}, \bar{s_\mathcal{P}}) + J^{i-1}(\bar{s_\mathcal{P}}, {\mu_\mathcal{P}}^{i-1}(\bar{s_\mathcal{P}}))\} & \\ & \text{for } s_\mathcal{P} \notin \mathcal{F}_\mathcal{P} \end{cases} \tag{5}$$

Then, we update the policy for each state such that

$$\mu^i_\mathcal{P}(s_\mathcal{P}) = \arg\min_{\hat{s_\mathcal{P}}} J^i(s_\mathcal{P}, \hat{s_\mathcal{P}}) \tag{6}$$

This value iteration process can be terminated when the following inequality is satisfied

$$\max_{s_\mathcal{P} \in S_\mathcal{P}} \left| J^i\left(s_\mathcal{P}, \mu^i_\mathcal{P}(s_\mathcal{P})\right) - J^{i-1}\left(s_\mathcal{P}, \mu^{i-1}_\mathcal{P}(s_\mathcal{P})\right) \right| \leq \theta \tag{7}$$

which means that $J\left(s_\mathcal{P}, \mu_\mathcal{P}(s_\mathcal{P})\right)$ has converged to the fixed point, denoted as $J^*(s_\mathcal{P}, \hat{s_\mathcal{P}})$. The strategy for this product automata can be computed as $\mu^*_\mathcal{P}(s_\mathcal{P}) = \arg\min_{\hat{s_\mathcal{P}}} J^*(s_\mathcal{P}, \hat{s_\mathcal{P}})$, which maps the current state to the next state to be visited. By projecting the optimal reactive strategy to the weighted transition system of robot $r_1, r_2 \cdots r_N$, a solution to Problem 1 can be obtained.

V. Case study

In this section, we provide some cases of multi-robot coordination in grid-world to illustrate the optimal strategy produced by the task planning algorithm proposed in Section IV.

Suppose there is a factory at a risk of water leakage that needs to be repaired by robots. The factory environment can be abstracted into a 10×10 grid-style workspace as Fig. 1 shows. The gray grids represent the region of fixed equipment that cannot be accessed by robots. Some regions of interest are colored, e.g., water valves (blue grids), detection stations (yellow grids) and upload station (pink grid). Robots can perform special action when on the colored grid as

Authorized licensed use limited to: Shanghai Jiaotong University. Downloaded on July 22,2022 at 00:47:10 UTC from IEEE Xplore. Restrictions apply.

marked in Fig. 1. The set of atomic proposition is denoted as $\mathcal{AP} = \{v_1, v_2, d_1, d_2, d_3, d_4, u\}$.

There are four robots $\{r_1, r_2, r_3, r_4\}$ with the same ability to move and perform action, represented by a circle in the figure. When it is green, it means that the robot is in normal working condition, while when it is red, it is broken and can no longer move or perform any action. The green circles in Fig. 1 represent the initial positions of the 4 robots, and they are all in normal working conditions initially.

The global cooperative task that the robots' team needs to accomplish is as follows: They should first go to the two water valves and close them, then collect data at any one of the detection stations, and finally go to the upload station to upload the collected data. It can be expressed by scLTL as

$$\begin{aligned} \phi = & (\neg(d_1 \vee d_2 \vee d_3 \vee d_4)\, \mathcal{U}\, v_1) \\ & \wedge (\neg(d_1 \vee d_2 \vee d_3 \vee d_4)\, \mathcal{U}\, v_2) \\ & \wedge (\neg\, u\, \mathcal{U}\, (d_1 \vee d_2 \vee d_3 \vee d_4)) \\ & \wedge \mathbf{F}\, u \end{aligned}$$

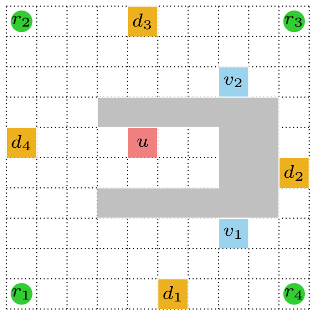

Fig. 1. A 10×10 grid-style workspace of a factory

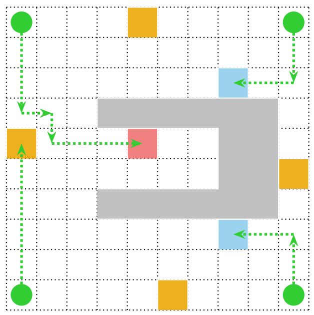
Fig. 2. The initial planned trajectory for the multi-robot system

Resulted from the water leakage risk, robots in the workspace may face the problem of short-circuit fault and fail to work properly. We assume that there are at most 2 robots broken during the whole execution, i.e., robust coefficient $k = 2$. We need to generate the optimal reactive strategy for robots to complete the global scLTL task ϕ even in worst case there are 2 robots broken.

The initial planned trajectory is shown in Fig. 2. Each robot needs to complete only one atomic proposition. Notice that the speeds of all robots are supposed to be the same, so the traveling time is proportional to the length of the green track in the figure. The trace generated by the trajectory in Fig. 2 is v_1, v_2, d_4, u. It can be verified that it satisfies scLTL formula ϕ. The total time cost is 8 time units.

We then show the re-planned paths after some robots are broken in the process of execution. The solid lines in green represent the trajectory that the robots have already passed. The dotted lines in green represent the trajectory planned for these robots, based on the assumption that there are no more broken robots. The red lines represent the trajectory which can not be completed any longer by broken robots.

Case 1 : Robot r_3 and r_4 are broken meanwhile after 2 time units.

Fig. 3(a) shows the instant that two robots are broken. The re-planned trajectory is shown in Fig. 3(b). Because there are only two working robots now, they need to change the trajectory and perform more atomic propositions, i.e., robot r_1 needs to visit v_1, d_2, and robot r_2 needs to visit v_2, r. The trace generated by the trajectory in Fig. 3(b) is v_1, v_2, d_2, u. The total time cost is 18 time units.

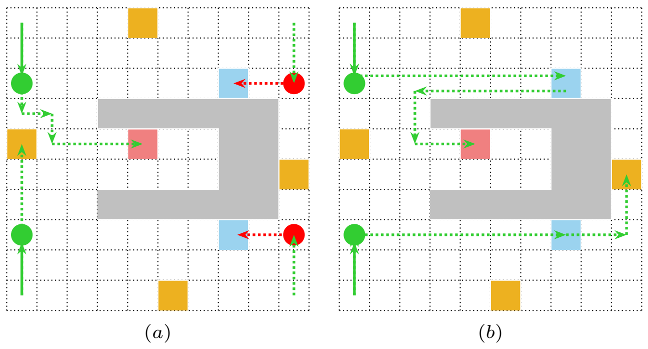

Fig. 3. The instant that robot r_3 and r_4 are broken and the re-planned trajectory

Case 2 : Robot r_1 and r_2 are broken meanwhile after 2 time units.

Fig. 4 shows this case, where r_3, r_4 can not stop after they perform v_2 and v_1. The trace generated by trajectory in Fig. 4(b) is v_1, v_2, d_1, u. The total time cost is 13 time units.

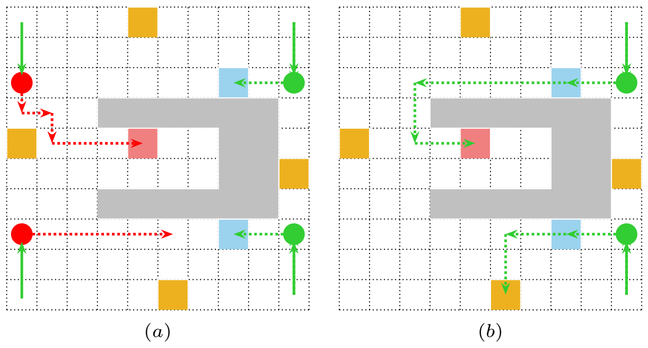

Fig. 4. The instant that robot r_1 and r_2 are broken and the re-planned trajectory

Case 3 : Robot r_3 is broken after 2 time units and r_2 is broken after 5 time units.

Fig. 5 shows the instant that robot r_3 is broken and the re-planned trajectory after one robot failure. Robot r_2 is supposed to perform v_2 (to make up for r_3's work). The trace generated by trajectory in Fig. 5(b) is v_2, v_1, d_1, u. The total time cost is 9 time units.

r_2 is also broken after 5 time units as Fig. 6(a). Fig. 6(b) shows the trajectory after second update. Robot r_1 and r_4 change their next sub-task again. The trace generated by trajectory in Fig. 6(b) is v_2, v_1, d_4, u. The total time cost is 20 time units.

Authorized licensed use limited to: Shanghai Jiaotong University. Downloaded on July 22,2022 at 00:47:10 UTC from IEEE Xplore. Restrictions apply.

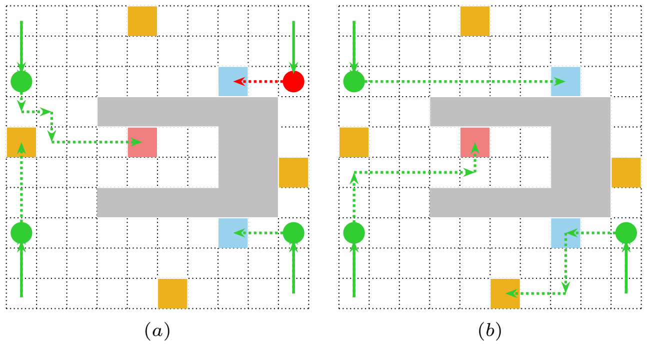
Fig. 5. The instant that robot r_3 is broken and the re-planned trajectory

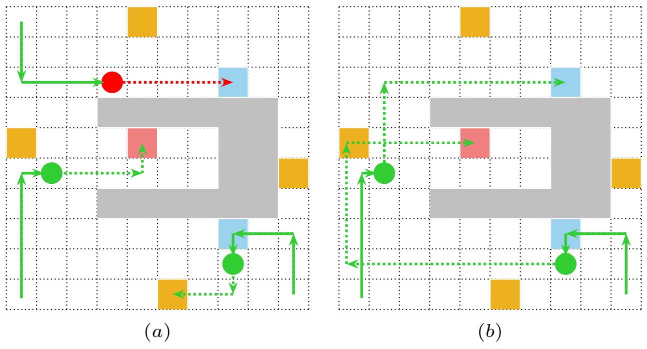
Fig. 6. The instant that robot r_4 is broken and the re-planned trajectory

VI. Conclusion

In this work, we investigated the problem of robust LTL task planning for multi-robot system under possible individual robot failures. We provided an effective approach for synthesizing reactive planning strategies that guarantees the satisfaction of the desired scLTL task even k out of N robots fail such that better system reliability. Illustrative examples were provided to demonstrate the effectiveness of the proposed algorithm. In the future, we plan to extend the proposed algorithm to handle tasks described by general LTL formulae. Furthermore, we would like to improve the proposed algorithm to mitigate the computational complexity of the proposed algorithm.

References

[1] C. Baier and J.-P. Katoen. *Principles of Model Checking*. MIT press, 2008.

[2] H. Bank, S. D'souza, and A. Rasam. Temporal logic (tl)-based autonomy for smart manufacturing systems. *Procedia Manufacturing*, 26:1221–1229, 2018.

[3] C. Belta, B. Yordanov, and E.A. Gol. *Formal methods for discrete-time dynamical systems*, volume 15. Springer, 2017.

[4] A. Bozkurt, Y. Wang, M. Zavlanos, and M. Pajic. Control synthesis from linear temporal logic specifications using model-free reinforcement learning. In *2020 IEEE International Conference on Robotics and Automation*, pages 10349–10355. IEEE, 2020.

[6] X. Ding, S. Smith, C. Belta, and D. Rus. Optimal control of markov decision processes with linear temporal logic constraints. *IEEE Transactions on Automatic Control*, 59(5):1244–1257, 2014.

[5] S. Coogan, M. Arcak, and C. Belta. Formal methods for control of traffic flow: Automated control synthesis from finite-state transition models. *IEEE Control Systems Magazine*, 37(2):109–128, 2017.

[7] M.P. Fanti, A. Mangini, G. Pedroncelli, and W. Ukovich. A decentralized control strategy for the coordination of agv systems. *Control Engineering Practice*, 70:86–97, 2018.

[8] M. Guo and D. Dimarogonas. Multi-agent plan reconfiguration under local ltl specifications. *The International Journal of Robotics Research*, 34(2):218–235, 2015.

[9] M. Guo and M. Zavlanos. Probabilistic motion planning under temporal tasks and soft constraints. *IEEE Transactions on Automatic Control*, 63(12):4051–4066, 2018.

[10] Y. Kantaros and G. Pappas. Optimal temporal logic planning for multi-robot systems in uncertain semantic maps. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4127–4132. IEEE, 2019.

[11] D. Kasenberg and M. Scheutz. Interpretable apprenticeship learning with temporal logic specifications. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 4914–4921. IEEE, 2017.

[12] H. Kress-Gazit, G. Fainekos, and G. Pappas. Temporal-logic-based reactive mission and motion planning. *IEEE Transactions on Robotics*, 25(6):1370–1381, 2009.

[13] H. Kress-Gazit, M. Lahijanian, and V. Raman. Synthesis for robots: Guarantees and feedback for robot behavior. *Annual Review of Control, Robotics, and Autonomous Systems*, 1:211–236, 2018.

[14] S. LaValle. *Planning Algorithms*. Cambridge university press, 2006.

[15] R. Majumdar, E. Render, and P. Tabuada. Robust discrete synthesis against unspecified disturbances. In *Proceedings of the 14th international conference on Hybrid systems: computation and control*, pages 211–220, 2011.

[16] S. Smith, J. Tůmová, C. Belta, and D. Rus. Optimal path planning for surveillance with temporal-logic constraints. *The International Journal of Robotics Research*, 30(14):1695–1708, 2011.

[17] A. Ulusoy and C. Belta. Receding horizon temporal logic control in dynamic environments. *The International Journal of Robotics Research*, 33(12):1593–1607, 2014.

[18] A. Ulusoy, S Smith, X Ding, and C Belta. Robust multi-robot optimal path planning with temporal logic constraints. In *2012 IEEE International Conference on Robotics and Automation*, pages 4693–4698. IEEE, 2012.

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

[20] A. Ulusoy, T. Wongpiromsarn, and C. Belta. Incremental controller synthesis in probabilistic environments with temporal logic constraints. *The International Journal of Robotics Research*, 33(8):1130–1144, 2014.

[21] E. Wolff, U. Topcu, and R. Murray. Robust control of uncertain markov decision processes with temporal logic specifications. In *2012 IEEE 51st IEEE Conference on Decision and Control*, pages 3372–3379. IEEE, 2012.

[22] Y. Xie, X. Yin, S. Li, and M. Zamani. Secure-by-construction controller synthesis for stochastic systems under linear temporal logic specifications. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 7015–7021. IEEE, 2021.

[23] S. Yang, X. Yin, S. Li, and M. Zamani. Secure-by-construction optimal path planning for linear temporal logic tasks. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 4460–4466. IEEE, 2020.

[24] Y. Yang, X. Yin, and S. Li. A distributed framework for multi-robot task planning with temporal logic specifications. In *2020 IEEE 16th International Conference on Control & Automation (ICCA)*, pages 570–575. IEEE, 2020.

[25] B. Yordanov, J. Tumova, I. Cerna, J. Barnat, and C. Belta. Temporal logic control of discrete-time piecewise affine systems. *IEEE Transactions on Automatic Control*, 57(6):1491–1504, 2011.

[26] X. Yu, X. Yin, S. Li, and Z. Li. Security-preserving multi-agent coordination for complex temporal logic tasks. *Control Engineering Practice*, 123:105130, 2022.

[27] J. Zhao, X. Yu, X. Li, and H. Wang. Bearing-only formation tracking control of multi-agent systems with local reference frames and constant-velocity leaders. *IEEE Control Systems Letters*, 5(1):1–6, 2020.

Authorized licensed use limited to: Shanghai Jiaotong University. Downloaded on July 22,2022 at 00:47:10 UTC from IEEE Xplore. Restrictions apply.