2022 IEEE 17th International Conference on Control & Automation (ICCA)
June 27-30, 2022 (Hybrid) Naples, Italy

Temporal Logic Robot Task Planning with Active Acquisition of Information

Jiawei Zhao, Xiang Yin and Shaoyuan Li

2022 IEEE 17th International Conference on Control & Automation (ICCA) | 978-1-6654-9572-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICCA54724.2022.9831883

***Abstract*— This paper proposes a mission planning algorithm for the robot to operate in an environment with interference. The robot is equipped with proximity sensor to roughly estimate the environment and can apply active perception action to detect the environment. The goal of the robots is to accomplish complex tasks, captured by co-safe Linear Temporal Logic (scLTL) formulas. With the interference created by environment, general observation-based solution will cause a huge cost even if it can complete the task. When the robot without active perception encounters a set of states that the proximity observation can not tell, it will choose a conservative but costlier action. We aim to design an active perception plan for the robot to offset the impact which can reduce the cost. Our algorithm can trade off security and cost, to find a strategy to accomplish the complex task. We illustrate our method on motion planning case studies and show that the proposed algorithm can address complex planning tasks with smaller costs through active perception action.**

Multi-Robot Systems, Task Planning, Linear Temporal Logic, Failure Robustness

I. INTRODUCTION

Temporal logic has been widely used in task planning in recent years, as it can provide a fully automated correct-by-design controller synthesis approach for autonomous robots. Compared to simple point-to-point navigation task, Temporal logics such as linear temporal logic (LTL) provides a formal language that can accurately describe complex temporal tasks. On the other hand, the decision-making is based on information and the robot always makes task plan based on the currently known information. In the complete information environment, the robot is assumed that availability of complete and precise information. The research of robot's task planning with complete information has been relatively mature [1]–[8].

However, in the actual task execution process, the information obtained is often limited, which corresponds to the the partial observation situation. According to the modeling of environment, it can be constructed into partial observable Markov partial observable Markov decision process (POMDP) and a two-person zero-sum game. The former is applied to pure random environment [9] and the latter is applied to uncertain environment [10]. Both of them can be solved by constructing a subset [11], but their computational complexity is exponentially related to the size of the state space [12], which limits the application of the algorithm. Robot can use sensors to gain information about itself and their environment, the algorithm's complexity decreases as the system learns more about the current state. The acquisition of information is not arbitrary, we consider limited information acquisition called active perception. Since the acquisition of information is limited, active perception may not necessarily have a positive impact on the task. Therefore, it needs to consider whether information should be obtained at the current moment and what information should be obtained.

Active perception action strategies under temporal logic constraints to accomplish complex task is also considered in [13]–[17]. [13] consider a replanning algorithm for planning under partially observability with perception actions. They generate a classical planning problem that reflects information about the robot's belief state, at each state by using state sampling which leads to much smaller classical planing problems. [14] also suggests an online algorithm that repeatedly selects the next perception action to execute and plans to achieve it in a classical setting. The difference is it avoids the difficulty in representing and updating a belief by using heuristic landmarks planner.In order to alleviate computational effort in control strategies design for robot against with uncontrollable environments under incomplete information, [15] introduce active perception action to transforms a deterministic controller under complete information into a randomized controller based on observation. [16] allows for active perception in complex tasks that improve confidence in a belief when necessary to satisfy probabilistic temporal logic over reals (PRTL) specifications. The above works consider the active perception action as a free approach to get extra information that can be used for robot acting decision or the limitation of the active perception action is just the location. In this paper, we assume that the robot's active perception action has cost which is more close to the real scene and focus on the problem of synthesizing the optimal control and active sensing joint policies at the discrete level of abstraction.

The rest of this paper is organized as follows. We begin with an informal problem statement and an overview of the solution approach. In Section III, we review the theory of LTL and automata. In ection IV, we present the problem statement. In Section V, we first show how to construct the belief strategy network. Then, we propose an active perception strategy algorithm to synthesis controller. An illustrative example is provided in Section VI. Finally, we conclude the paper in Section VII.

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).

J. Zhao, 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: zjw2015@sjtu.edu.cn.

II. Motivations

We study the following problem: Given the robot task specified by temporal logic, a partially observable dynamic environment, and a set of perception actions which can be actively used to obtain additional information, construct a controller with control inputs and active perception action strategy which can control the system satisfies the specification and optimizes the progress. We illustrate the motivations of our approach through an example.

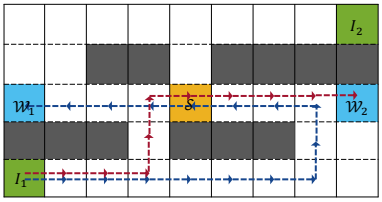


Fig. 1. This figure show the example with one goal region $\mathcal{S}$, two initial regions $\mathcal{I}$, two warehouses $\mathcal{W}$ and several obstacles. In this example, the robot will start from $\mathcal{I}_1$.

Example 1: Consider the workspace in Fig. 1. in which a robot has to collect the source in goal region $\mathcal{S}$ (yellow) and deliver it to one of the warehouses $\mathcal{W}_1$ and $\mathcal{W}_2$ (blue) while avoiding the gray obstacle area. The robot can move in three ways: go ahead (M), turn left and go ahead (L) and turn right and go ahead (R), one cell at a time. The robot will initially be placed in the green area, with $\mathcal{I}_1$ orientation set to right and the $\mathcal{I}_2$ orientation set to left. And the robot only has a partial observation of the environment, but it is equipped with sensors which can help the robot to locate. When the robot apply perception action, it can know the current location. Given the cost of perception action, so it is not preferable to apply this locate sensor all the time in the execution of task. Then, we want to design a strategy that decides, at each step, whether it is necessary to apply the perception action, and which control action to take, such that the robot can accomplish its task and best fits the user's preferences at the same time.

Suppose that in this experiment, the robot start from $\mathcal{I}_1$. In the absence of active perception, the strategy move three steps forward and turn left is not a good choice cause if the robot start from $\mathcal{I}_2$, it will run into an obstacle. To ensure the completion of the task, it has to choose a long path which is shown as blue arrows. And if the robot has the additional information which can help it located itself obtain by active perception, it can make decisions that previously seemed risky shown as the red arrows in the figure.

We can accomplish the task with active perception action in a way that users prefer. For example, if users believe that the shorter the actual distance traveled to complete the task, the better, our strategy can apply active perception action as long as the robot a little confused about its current state or physical action strategy. We now continue with some preliminary and system model for the formal problem formulation and our solution.

III. Preliminaries

In this section, we give some definitions and introduce some notations that we use in the rest of this paper. For convenience, we use action to represent the physical action and distinguish from perception action.

A. System Model

In the work, the robot motion as well as the environment uncertainty are abstracted as a labeled transition system (LTS) [18]

$$T = (X, Act, \rightarrow, X_0, \mathcal{AP}, L) \tag{1}$$

where X is a finite set of states represent the locations of the robot; Act is a finite set of control actions the robot can take; $\rightarrow \subseteq X \times Act \times X$ is the transition relation, where $(x, a, x') \in \rightarrow$ (simplified as $x \xrightarrow{a} x'$) means that by taking action a at state x, the robot may result in state x'; $X_0 \subseteq X$ is a set of possible initial states the robot may start from; $\mathcal{AP}$ is a set of atomic propositions representing some high-level properties of interest; $L : X \rightarrow 2^{\mathcal{AP}}$ is the labeling function that determines the set of atomic propositions that hold at each state. For any state $x \in X$ and control action $a \in Act$, we define $Post(x, a) = \{x' \in X : x \xrightarrow{a} x'\}$ as the set of successor states of q under a, and define $Pre(x, a) = \{x' \in X : x' \xrightarrow{a} x\}$ as the set of predecessor states of x under a. In general, an LTS is non-deterministic in the sense that $|X_0| > 1$ and $|Post(x, a)| > 1$, i.e., the initial-states and the successor states are not unique.

A *path* is an infinite sequence of states in the form of $\tau = \tau[1]\tau[2]\cdots$ such that $\tau[1] \in X_0$ and $\forall i \geq 0, \exists a \in Act : \tau[i] \xrightarrow{a} \tau[i+1]$. A finite path is defined analogously. We denote by $\texttt{Path}^{\omega}(T) \subseteq X^{\omega}$ and $\texttt{Path}^{*}(T) \subseteq X^{*}$, respectively, the set of all infinite paths and finite paths in T. The *trace* of an infinite path $\tau \in X^{\omega}$, denoted by $\texttt{Trace}(\tau) \in (2^{\mathcal{AP}})^{\omega}$, is the sequence of the set of atomic propositions that hold at each instant in the path, i.e., $\texttt{Trace}(\tau) = L(\tau[1])L(\tau[2])\cdots$. A *controller* (with full state information) is a mapping $C : X^{*} \rightarrow Act$ that assigns each finite path a control action. A path $\tau \in \omega(T)$ is feasible under controller C if $\forall i \geq 0 : \tau[i] \xrightarrow{C(\tau[1]\cdots\tau[i])} \tau[i+1]$, and we denote by $\texttt{Path}^{\omega}(C \times T)$ the set of all paths feasible under C.

B. Linear Temporal Logic

We will use co-safe linear temporal logic (scLTL) [19] to specify the desired system properties. Let $\mathcal{AP}$ be a set of atomic propositions where $\pi \in \mathcal{AP}$ is a Boolean variable. An scLTL formula over $\mathcal{AP}$ is defined as follow:

$$\varphi := \pi \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \varphi \,\mathcal{U}\, \varphi \mid \bigcirc\varphi \tag{2}$$

where $\neg$(negation),$\vee$(dis-junction), and $\wedge$(conjunction) are Boolean operators, and $\bigcirc$(next), $\mathcal{U}$(until) are temporal operators. The semantics of an LTL formula φ are defined on an infinite sequence $\sigma = \sigma_1\sigma_2\ldots$ of truth assignments to the atomic propositions, where for every $i \geqslant 0$, σ_i denotes the set of atomic propositions that are true at position i. We use

$\sigma, i \models \phi$ to indicate that an LTL formula φ satisfied by σ at position i. Intuitively, the formula $\bigcirc\varphi$ express that φ is true in the next step and $\varphi_1 \mathcal{U} \varphi_2$ express the property that φ_1 is true until φ_2 becomes true. In this paper, we consider the co-safe LTL, for a language $L \subseteq \Sigma^\omega$ of infinite words over the alphabet Σ, L is co-safety iff every $\omega \in L$ has a *good prefix* $x \in \Sigma^*$ such that for all $y \in \Sigma^\omega$, we have the concatenation $x \,.\, y$ of x and y in L. For a co-safety language L, we denoted by $pref(L)$ the set of good prefixes for L. And with this constraint, we can derive the additional temporal operator "eventually"($\Diamond\varphi = True\ \mathcal{U}\ \varphi$), where the sequence σ satisfies the formula $\Diamond\varphi$ if φ is true in some position of the sequence. Note that we also will always $\Box$ to express the obstacle avoidance requirement, which is not be defined in scLTL and different from the always in LTL.

From an scLTL formula φ, a deterministic finite automata (DFA) can be constructed. Such a DFA is given by a tuple

$$\mathcal{A}_\varphi = (Q, \Pi, \delta_\varphi, Q_0, Q_F) \tag{3}$$

where Q is a finite set of states; $\Pi = 2^{\mathcal{AP}}$ is the input alphabet, where each input symbol is a truth assignment to the propositions in $\mathcal{AP}$; $\delta_\varphi \subseteq Q \times \Pi \times Q$ is a deterministic transition function; $Q_0 \in Q$ is a set of initial states; and Q_F is a set of accepting states.

A run of ρ_φ of $\mathcal{A}_\varphi$ over a finite symbol $\sigma = \sigma_1\sigma_2 \ldots \sigma_k \in (2^{\mathcal{AP}})^*$, is a sequence $\rho_\varphi = q_0 q_1 \ldots q_k$, where $\delta_\varphi(q_k, \sigma_k) = q_{k+1}, \forall k \in \mathbb{N}$. A run ρ_φ is called accepting is $q_k \in Q_F$. For all finite path τ in $\mathtt{Path}^*(T)$, if its $\mathtt{Trace}(\tau) = L(\tau[1])L(\tau[2]) \cdots L(\tau[k])$ yields an accepting DFA run, we say that the path satisfied φ.

IV. Control under Active Information Acquisition

In this section, we introduce the observation model to the system and the systematic account of the active perception.

A. Observation Model

At each time step, the robot can get a set of sensory measurements to get an observation of itself and the environment by taking perception actions. While the measurements may be from multiple sensing units, for ease of notation, we consider their observation model by a general observation function.

We define $Sense$ be a finite set of *information acquisition actions* and define $\mathcal{O}$ a finite set of observation symbols. What the robot can observe depend on both the current state and the information acquisition action taken. Formally the observation model of the robot is specified by a mapping

$$Obs : X \times Sense \to \mathcal{O},$$

where $Obs(x, s) = o$ means that the by applying acquisition action $s \in Sense$ at state $x \in X$, the robot will observe o. Here we assume the observation mapping is deterministic. And with the observation function we define the reveal function as $Obs^{-1}(s, o) = \{x : Obs(x, s) = o\}$.

The proposed the information acquisition model is very general that captures many different observation models.

- *Static Full State Observation:* the robot always knows the current state precisely. The case can be captured by considering $Sense = \{I\}, \mathcal{O} = X$ and $\forall x \in X : Obs(x, I) = x$. Since the choice of acquisition action is unique, this information can actually be omitted, which boils down to the static full state observation setting.
- *Active Location Acquisition:* the robot may choose to deploy GPS, UAV or global camera, when need, to get its precise state information with costs. In case can be captured by considering $Sense = \{Y, N\}, \mathcal{O} = X \cup \{\text{NONE}\}$ and $\forall x \in X : Obs(x, Y) = x$ and $Obs(x, N) = \text{NONE}$.
- *Static Partial Observation:* the robot does not know the current state precisely but may know which equivalence class of the state space it is currently at. This is a very typical scenario, for example, the robot can only observe its surrounding environment. This scenario is similar to the case of static full state observation, for which we consider a single identity acquisition action $Sense = \{I\}$. The difference is that $\mathcal{O}$ is a new set of observation symbols rather than Q and Obs essentially induces an equivalence class on X, i.e., the observation model can be simplified as $Obs : X \to \mathcal{O}$.
- *Active Multiple Information Sources:* in the most general case, the robot may have both a static observation specified by $Obs_S : X \to \mathcal{O}_s$ and m different sensors it can deploy actively, which can be represented as $Obs_{a,i} : X \times \{Y, N\} \to \mathcal{O}_{a,i}, \forall i = 1, \ldots, m$. Then the entire observation model is $Obs : X \times \{Y, N\}^m \to \mathcal{O}_s \times \mathcal{O}_{a,1} \times \cdots \times \mathcal{O}_{a,m}$

In the remaining part, we assume that the robot is equipped with a passive a sensor that can observe its surrounding environment (adjacent states of the current state) and can deploy a global camera to get its precise location with cost. That is, we have

- $Sense = \{Y, N\}$, where Y means that the global camera is deployed and N means the contrary; and
- $\mathcal{O} = X \cup \{o_1, o_2, \cdots, o_n\}$

If the robot choose to use the global camera, it will get the information of the current state and if not, it will only get the partial observation of surround region. The robot will choose active perception actions to execute according to the active perception strategy which will be defined in the subsequent section.

B. Control under Active Information Acquisition

To control the robot under active acquisition of information, we need two "controllers": one controls the motion actions of the robot and the other controls the acquisition actions of the robot, which are referred to as the *control strategy* and the *sensing strategy*, respectively. Both strategies depend on the observation sequence rather than the internal state run. To be more specific, we assume that the system first issues an information acquisition to get an observation and then issues a control action based on what observed, and so forth. Therefore, the *history* available to the robot

is an alternative sequence in the form of $s_1o_1a_1s_2o_2a_2\cdots$. We denote by $\text{HIS}_A = Sense\mathcal{O}(ActSense\mathcal{O})^*$ as the set of all finite histories ending up with observations symbols; we use sub-script A as a control action should be taken then. Similarly, we denote by $\text{HIS}_S = (Sense\mathcal{O}Act)^*$ as the set of all finite histories ending up with control actions; we use sub-script S as an acquisition action should be taken then. Therefore, a control strategy is a mapping $\text{STRA}_A : \text{HIS}_A \to Act$ and a sensing strategy is a mapping $\text{STRA}_S : \text{HIS}_S \to Sense$. Then the overall strategy is a tuple $\text{STRA} = (\text{STRA}_A, \text{STRA}_S)$.

A *run* is a sequence in the form of

$$\rho = x_1(s_1o_1a_1)x_2(s_2o_2a_2)x_3\cdots \in X_0(Sense\mathcal{O}ActX)^\omega.$$

The above run is said to be feasible in T under strategy $\text{STRA} = (\text{STRA}_A, \text{STRA}_S)$ if for any $i \geq 1$, we have

- $s_i = \text{STRA}_S(s_1o_1a_1\ldots s_{i-1}o_{i-1}a_{i-1})$; and
- $o_i = Obs(x_i, s_i)$; and
- $a_i = \text{STRA}_A(s_1o_1a_1\ldots s_io_i)$; and
- $x_i \xrightarrow{a_i} x_{i+1}$

We define $\text{Run}^\omega(T, \text{STRA})$ as the set of runs feasible in T under strategy STRA. We define a finite run be a prefix of a run that ends up with a state of form $x_1(s_1o_1a_1)\cdots x_{n-1}(s_{n-1}o_{n-1}a_{n-1})x_n \in X_0(Sense\mathcal{O}ActX)^*$. We define $\text{Run}^*(T, \text{STRA})$ as the set of finite runs feasible in T under strategy STRA. The trace of a run is defined by $\text{Trace}(\rho) = L(x_1)L(x_2)\cdots \in (2^{\mathcal{AP}})^\omega$. We say that the joint strategy STRA achieves scLTL task φ if $\forall\rho \in \text{Run}^\omega(T, \text{STRA}) : \text{Trace}(\rho) \models \varphi$.

C. Optimal Synthesis Problem

We assume that both issuing control actions and issuing information acquisition actions have costs. We denote by $\text{Cost}_A : Act \to \mathbb{N}$ and $\text{Cost}_S : Sense \to \mathbb{N}$ the control cost and the information acquisition cost, respectively. Let $\rho = x_1(s_1o_1a_1)\cdots x_{n-1}(s_{n-1}o_{n-1}a_{n-1})x_n \in \text{Run}^*(T, \text{STRA})$ be a finite run feasible in T under strategy STRA. The total *joint cost* incurred is the accumulated control cost and acquisition cost, i.e.,

$$\text{Cost}(\rho) = \sum_{i=1,\ldots,n-1} \text{Cost}_A(a_i) + \text{Cost}_S(s_i)$$

The cost of an infinite run can be infinite in general. However, since we consider a scLTL φ, any infinite word satisfying φ has a good prefix. Therefore, it is of interest to only consider the accumulated cost up to the first instant satisfying φ. Then let $\rho = x_1(s_1o_1a_1)x_2(s_2o_2a_2)x_3\cdots \in \text{Run}^\omega(T, \text{STRA})$ be an infinite run. We definite $\rho_{pref,\varphi}$ be the short prefix of ρ whose traces is in $L_{pref,\varphi}$. Then the cost of an infinite run ρ is defined by

$$\text{Cost}(\rho) = \begin{cases} \text{Cost}(\rho_{pref,\varphi}) & \text{if } \text{Trace}(\rho_{pref,\varphi}) \models \varphi \\ \infty & \text{if } \text{Trace}(\rho_{pref,\varphi}) \not\models \varphi \end{cases}$$

Note that, since the transition function is non-deterministic, which means that the control strategy is reactive to the environment, $\text{Run}^\omega(T, \text{STRA})$ is not a singleton in general. Hence, the cost of the joint-strategy STRA is defined as the *worst-case*, i.e.,

$$\text{Cost}(\text{STRA}) = \max_{\rho \in \text{Run}^\omega(T, \text{STRA})} \text{Cost}(\rho)$$

Note that $\text{Cost}(\text{STRA}) = \infty$ if STRA does not achieve φ. Our goal is to minimize the joint cost while achieving the scLTL specification, which is formally formulated as the following optimal control synthesis problem.

Problem 1. *Given a LTS T, an observation model Obs and an scLTL task φ, find an optimal joint control and acquisition strategy $\text{STRA} = (\text{STRA}_A, \text{STRA}_S)$ achieving φ with minimum $\text{Cost}($STRA$)$.*

V. SOLUTION

This solution consist of two components: satisfying the scLTL specification φ and optimization of $\text{Cost}(\text{STRA})$.

A. A belief-based strategy network for making progress

First, with the labeled transition system T and the DFA $\mathcal{A}_\varphi$ represent the scLTL formula φ, we can construct a product game used for control synthesis in transition system with temporal logic constraints.

Definition 1. *(product game) Given a LTS $T = (X, Act, \to, X_0, L)$ and a DFA $\mathcal{A}_\varphi = (Q, \Pi, \delta_\varphi, Q_0, Q_F)$, a product game is a tuple $G = T \times \mathcal{A}_\varphi = (V, Act, \Delta, V_0, V_F)$, where*

- *$V = X \times Q$ is a finite set of states;*
- *Act is a finite set of control actions ;*
- *$\Delta \subseteq V \times Act \times V$ is the transition relation, defined as $\Delta((x,q),a) = (x',q')$ with $x \xrightarrow{a} x'$ and $\delta_\varphi(q, L(x')) = q'$;*
- *$V_0 \subseteq V$ is a set of ,initial states where $v_0 = (q_0, x_0) \in V_0, q_0 \in \delta_\varphi(q, L(x_0)$ with $q \in Q_0$ and $x_0 \in X_0$;*
- *$V_F = \{(x,q) | q \in Q_F\}$ is the accepting states.*

A run in game $\mathcal{G}$ is a sequence of game states $\rho = v_0v_1\ldots$ such that v_0 is the initial state and for all $i > 0$, there exists control action $a \in Act, v_i = \Delta(v_{i-1}, a)$. It is easy to build a state network using existing tools. However, the network obtained based on full information, and the system can not make a decision according to the current network state under partial observation.

With partial observation model mentioned before, the robot may not know the precisely current state but can lock certain sets of states, denoted by $\mathcal{B} \subseteq 2^V$ referred to as $belief\ state$, which is the set of states that the system thinks it can be. Initially, robot is in the initial states $b_0 = \{v_0 | v_0 \in V_0\}$. Since a controller can only make decisions based on its belief state, we need to define the bad belief state to avoid transition that may imply the mission failure. Let $V_{bad} \subseteq V$ be the set of states at which will be verified mission failure, then we can define the allow action $Allow(v) = \{a \in Act(v) | \Delta(v,a) \not\subseteq V_{bad}\}$. Therefore, given $b \in \mathcal{B}$, let $Allow(b) = \bigcap_{v \in b} Allow(v)$ be the set of safe actions with property: no matter in which state of the belief state b, by taking action in $Allow(b)$, no bad things will be happened which can also called operating within a safe area.

Then we define the accepting belief state. For an accepted belief state b, we need to ensure all the states in b has been accepted, i.e. $\{b \in \mathcal{B}_F | \ \forall v \in b, v \in V_F\}$. When all the information is available, robot can find out if the current state has been accepted or not, intuitively if the task has already been completed it does not need to make any additional movement. Security constraints will no longer apply. But when we consider the belief space, the task is not considered complete until all the states are accepted. In other words, even if only one state in belief state is not accepted, the robot still needs to move in the safe area until all the situations are accepted, that is, the mission is over. For example in Fig.4, when the current belief is b_2, because v_4 can not be accepted by the task, belief is not an accepted belief state. By apply action a, $\Delta(v_4, a) = v_6$ which can be accepted, robot can be sure that it complete the task. But if v_5 is danger area, the action a will be prohibited and robot need to find other way to an accepting belief state.

During the interaction with the environment, the robot update its $belief\ state$ depend on both the observation result and the history information include previous belief states and motion action taken. The robot alternately apply acquisition action and motion action, therefore, there are two ways to update the current belief state. We denote by $\mathcal{B}_S$ as the set of belief states need to apply acquisition action and $\mathcal{B}_A$ as the set of belief states need to apply control action. Formally, the sensing update is a mapping

$$\texttt{Update}_S : \mathcal{B}_S \times Sense \times \mathcal{O} \to \mathcal{B}_A$$

and a motion update is a mapping

$$\texttt{Update}_A : \mathcal{B}_A \times Act \to \mathcal{B}_S$$

The first condition robot updates its belief by applying information acquisition action, for example $\texttt{Update}_S(b', s, o) = b$ means that by taking acquisition action s and get the observation result o, robot update its belief state from b' to b, where $b = b' \cap Obs^{-1}(s, o)$. And the $\texttt{Update}_S(b', s)$ represent the set of beliefs that by applying acquisition action s the robot may reach. The second condition robot update its belief state after take a motion action, for example $\texttt{Update}_A(b', a) = b, b = \{(x, q) | x' \xrightarrow{a} x, \delta_\varphi(q', L(x)) = q, (x', q') \in b'\}$ means that by taking action a, robot will probabilistically reach the state in b' according to the LTS T.

The above updating rules essentially consists of two steps: take the perception action to update the state according to the observation results and take the control action to update the state according to the possible transition function. In order to separate these two updating steps, we construct a belief-based network that represents how the robot updates its beliefs based on perception action and control action. Then, we propose a synthesis method for joint control strategy using the network structure.

Example 2: Consider a transition system shown in Fig.2. Initially, the robot randomly appears in state x_1 and x_2, i.e. $X_0 = \{x_1, x_2\}$. In natural language, the desired specification

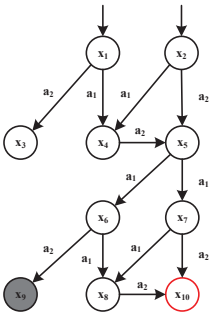


Fig. 2. The transition system in Example 2

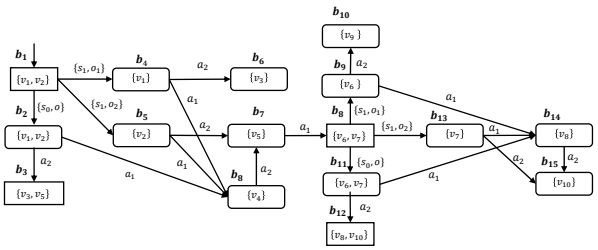


Fig. 3. Belief state network of Example 2 with active perception. Rectangular states correspond to the $\mathcal{B}_S$-state and rectangular states with rounded corners states correspond to the $\mathcal{B}_A$-state. We omit some rectangular which need to apply acquisition action but only have one element. The red rectangle is the accepted state and the shaded rectangles are unsafe states.

for the robot is as follows: *Go the collect goal in the particular area and avoid all the danger area.* This requirement is captured the formula $\varphi = \Diamond q_1 \wedge \Box \neg q_2$, where $L(x_{10}) = q_1$ and $L(x_9) = q_2$.

In Example 2, from initial state, robot's initial belief state is $b_1 = \{v_1, v_2\}$ and we assume that these two state have the same passive observation results to the system, in other words, if the robot does not apply acquisition action, it can not identify which specific state it is located. At this situation, it can choose to take useful acquisition action to judging by different outcomes such $\texttt{Update}_S(b_1, s_1, o_1) = \{v_1\}$ and $\texttt{Update}_S(b_1, s_1, o_2) = \{v_2\}$. Also, it can choose not to apply acquisition action, with no extra information, $b_2 = \texttt{Update}_S(b_1, s_0, o) = \{v_1, v_2\}$. Then we can apply control action $a \in Allow(b')$, take the robot to the next state and update the belief state by $\texttt{Update}_A(b', a)$. If the current belief state has more than one element and $Allow(b) = \emptyset$, it can choose to use acquisition action to eliminate the doubt. Then continues like this, we get the belief-based strategy network which is shown in Fig. 3.

To get the strategy for completing the task, then we need to prune all belief state from which no accepting belief state b_f can be reached. For instance, in Fig. 3 b_3, b_6, b_{10} and b_{12} need to be pruned. After that, we get a belief-based network that at any point of the network we can find the appropriate joint control strategy to reach an accepting state.

B. An active perception strategy for reducing cost

In order to find the minimum cost strategy, we will drop weights to the states in the network we obtained in the

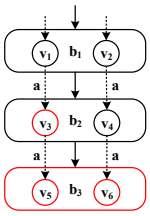


Fig. 4. The accepted belief state

previous section. The weights to the states is a mapping

$$W : \mathcal{B} \to \mathbb{Z}^+,$$

where $W(b)$ is the cost of actions that can be taken to drive the robot from the current belief state b to an accepting state in $\mathcal{B}_F$. To get the value of $W(b)$, we use a recursion algorithm.

This algorithm can be divided into two parts. The first part is to assign state values according to the network built by the method mentioned in IV-A, and the second part is to get the request run according to the state value obtained by the first part.

Initially, we set the value of the accepted belief states in $\mathcal{B}_F$ to 0, i.e., $W_0(x) = 0, \forall x \in \mathcal{B}_F$, and initialize the number of iteration to 0. Then we defined a set of the belief states which value has been assigned by $\mathcal{B}_i$ where i is equal to the number of iterations.For iteration assignment, we can still think of it as a two-player game between the robot and the environment. The goal of the robot is to minimize the total cost incurred, while the environment wants to maximize the cost. So from the current belief state by taking action a or s, when we estimate the cost of completing a task after the action, we need to consider the worst case. Similarly, in the process of iterative assignment, the worst case should be considered for the same action, and due to the robot's initiative the least cost should be considered for different actions. Then we compute the value of each belief state by value iteration as follows:

$$W_{k+1}(b) = \begin{cases} \min\limits_{s \in Sense} \texttt{Cost}_S(s) + \max W_k(b') & \text{if } b \in \mathcal{B}_S \\ \min\limits_{a \in Act(b)} \texttt{Cost}_A(a) + \max W_k(b') & \text{if } b \in \mathcal{B}_A \end{cases} \tag{4}$$

where $b' \in \mathcal{B}_k$ has been set value and $b' \in \texttt{Update}_S(b, s, o)$ and $b' \in \texttt{Update}_A(b, a)$, respectively.

The value iteration will converge to the value function by W^*. It is known that such a value for each state can be determined only by a finite number of iterations for at most $L = n$ steps, where $n = |\mathcal{B}_S \cup \mathcal{B}_A|$ is the number of states in network. Specifically, by computing value function W_L, we have

$$W^*(b) = W_L(b) \tag{5}$$

In the second part, we restart from the initial state to find the path with the minimum cost. The $W^*(b_0)$ is the optimal cost one can guarantee the robot to achieve. Then at each $\mathcal{B}_S$ state, robot choose the acquisition action from $\arg\min_{s \in Sense}(\texttt{Cost}_S(s) + max\ W^*(b', s))$ to get the observation

Algorithm 1 Get the cost value of the network tree

Input: Belief state network, control action cost function $\texttt{Cost}_A$ and acquisition action cost $\texttt{S}$.

Output: The belief state network tree with cost value

1: Let $\mathcal{B}_0 \leftarrow \{b | b \in \mathcal{B}_F\}$, $W_0(b) \leftarrow 0, \forall x \in \mathcal{B}_0$ and $i \leftarrow 0$
2: **while** the number of iteration $i < n$ where $n = |\mathcal{B}_S \cup \mathcal{B}_A|$ **do**
3: $i \leftarrow i + 1$
4: $W_i(b) \leftarrow \min_{s \in Sense} \texttt{Cost}_S(s) + \max W_k(b')$ where $b \in \mathcal{B}_S, b' \in \mathcal{B}_{i-1}$, and $\exists s \in Sense$ such that $b' \in \texttt{Update}_S(b, s, o)$
5: $W_i(b) \leftarrow \min_{a \in Act(b)} \texttt{Cost}_A(a) + \max W_k(b')$ where $b \in \mathcal{B}_A, b' \in \mathcal{B}_{i-1}$, and $\exists a \in Act$ such that $b' \in \texttt{Update}_A(b, a)$
6: $\mathcal{B}_i \leftarrow \{b | W_i(b) \neq 0\} \cup \mathcal{B}_{i-1}$
7: **end while**
8: Return the state network tree with cost value

result. And at each $\mathcal{B}_A$ state, robot can choose control action from $\arg\min_{a \in Allow(b)}(\texttt{Cost}_A(a) + max\ W^*(b', a))$ to move to the next state. Because we are considering the worst case, the results of observation and action may carry a smaller cost, but the cost of ensuring the completion of the task is still $W^*(b_0)$. By constantly choosing the acquisition action and control action in turn, we will get the optimization STRA.

VI. SIMULATION AND RESULTS

In this section, we will apply our method in the grid simulation environment. And then, by comparison with other two algorithms, we aim to show the benefits of our algorithms.

The proposed strategy is applied for a treasure hunt task on the gridworld shown in Figure 5. The treasure is buried in the region r_g which is the yellow region in the figure. The robot can move to an adjacent cell at a time denoted by $'N', 'S', 'E', 'W'$, and The robot system is equipped with a short range detector. The detector can detect whether there are obstacles which is black area in the figure or walls around the grid where the robot is located and return an array of four elements as the passive observation, i.e. if the robot's current location is $(0, 0)$, the observation result is $Obs(N, S, E, W) = (0, 1, 0, 1)$, which means there are obstacles or walls in the south and west of the current grid. In addition of the passive observation, the robot can apply active perception action to obtain extra information to locate. And we assume that when robot is closer to the target, the perception action's cost $\texttt{Cost}_S(s)$ will become more expensive, in this case, the perception cost of perception is $\texttt{Cost}_S(s) = 13 - dis(x, r_g)$ where $dis(x, r_g)$ is the distance between the current position and the goal position. And the control action cost is the distance between two adjacent regions which is shown in the figure. There is a special region in the workspace with the property of interference labeled with blue, called fringe area. When robot in the blue region, its control action will be disturbed by the environment and lead to different results, for example when robot is in

the fringe area and make a physical action $'E'$ which is the blue arrow in Fig.5, then in the next time the robot will appear in $'East - North'$ or $'East - South'$ cell that the red arrows point to. The objective of the robot is to arrive in the treasure region and get the treasure while avoiding obstacles and the danger area. Formally, the temporal logic formula is $\phi = \Diamond(r_g \wedge dig) \wedge \Box\neg r_d \wedge \Box\neg r_b$ where dig is the action to get the treasure, r_b is the obstacle and r_d is the danger area.

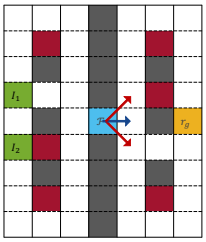
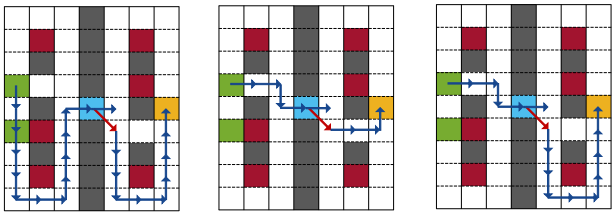


Fig. 5. The gridworld of areas with different properties. The black cells, red cells, blue cell, green cells and yellow cell are obstacles, danger areas r_d, fringe area, initial states and goal state r_g, respectively. The blue arrow is the action chosen by the robot and the red arrows are the actions disturbed by the environment in fringe area.



(a) (b) (c)

Fig. 6. Possible run with different strategys. In this example, robot start from $\mathcal{I}_1$ and the environment will cause the robot to deflect downwards in fringe area.

In what follows, we show three experiments with different planning strategy (1. Planning without perception action. 2. Using active perception action whenever it can use. 3. Using the strategy that mention in the section before). A possible execution of the different strategies is depicted in Fig. 6, where the robot start in $\mathcal{I}_1$. Fig. 6(a) shows the trajectory of the strategy without perception action. The robot is guaranteed to complete its mission, it has to choose a long way around the dangerous area to get the goal region. In Fig. 6(b), robot's strategy applies the active perception action whenever it has doubts, so it can always find the closest path. Because the perception cost increases when the robot close to the goal, the cost can sometimes be higher than the first strategy. Finally, in Fig. 6(c), the robot uses the Algorithm 1 getting the active perception strategy to accomplish the task. The robot can autonomously choose whether to adopt the active perception action, so it can complete the task at a lower cost than the previous two methods.

VII. Conclusion

In this paper, we proposed to use active perception for robot to accomplish complex task captured by co-safe linear temporal logic with a low cost in an interference environment. The robot is equipped with proximity sensor that allow it to roughly estimate the current state and the active perception action can get more and accurate information to counteract environmental influences. Compared with the observation-based strategy without active perception, the active perception strategy leads to a cost-efficient way of perception design and path planning.

References

[1] H. Kress-Gazit, G.E. Fainekos, and G.J. Pappas. Temporal-logic-based reactive mission and motion planning. *IEEE transactions on robotics*, 25(6):1370–1381, 2009.

[2] H. Kress-Gazit, T. Wongpiromsarn, and U. Topcu. Correct, reactive, high-level robot control. *IEEE Robotics & Automation Magazine*, 18(3):65–74, 2011.

[3] T. Wongpiromsarn, U. Topcu, and R.M. Murray. Receding horizon temporal logic planning. *IEEE Transactions on Automatic Control*, 57(11):2817–2830, 2012.

[4] 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.

[5] P. Lv, X. Yin, Y. Ji, and S. Li. A game-theoretical approach for optimal supervisory control of discrete event systems for cyclic tasks. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 324–330. IEEE, 2021.

[6] 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.

[7] S. Liu, A. Trivedi, X. Yin, and M. Zamani. Secure-by-construction synthesis of cyber-physical systems. *Annual Reviews in Control*, 2022.

[8] 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.

[9] L.P. Kaelbling, M.L. Littman, and A.R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

[10] A. Arnold, A. Vincent, and I. Walukiewicz. Games for synthesis of controllers with partial observation. *Theoretical Computer Science*, 303(1):7–34, 2003.

[11] K. Chatterjee, L. Doyen, T.A. Henzinger, and J.-F. Raskin. Algorithms for omega-regular games with imperfect information. In *International Workshop on Computer Science Logic*, pages 287–302. Springer, 2006.

[12] K. Chatterjee and L. Doyen. The complexity of partial-observation parity games. In *International Conference on Logic for Programming Artificial Intelligence and Reasoning*, pages 1–14. Springer, 2010.

[13] R.I. Brafman and G. Shani. Replanning in domains with partial information and sensing actions. *Journal of Artificial Intelligence Research*, 45:565–600, 2012.

[14] G. Shani, R. Brafman, S. Maliah, and E. Karpas. Heuristics for planning under partial observability with sensing actions. *Heuristics and Search for Domain-Independent Planning*, page 30, 2013.

[15] J. Fu and U. Topcu. Synthesis of joint control and active sensing strategies under temporal logic constraints. *IEEE Transactions on Automatic Control*, 61(11):3464–3476, 2016.

[16] R. R. da Silva, V. Kurtz, and H. Lin. Active perception and control from temporal logic specifications. *IEEE Control Systems Letters*, 3(4):1068–1073, 2019.

[17] X. Yin and S. Lafortune. A general approach for optimizing dynamic sensor activation for discrete event systems. *Automatica*, 105:376–383, 2019.

[18] M. Kloetzer and C. Belta. A fully automated framework for control of linear systems from temporal logic specifications. *IEEE Transactions on Automatic Control*, 53(1):287–297, 2008.

[19] O. Kupferman and M.Y. Vardi. Model checking of safety properties. *Formal Methods in System Design*, 19(3):291–314, 2001.