Online Motion Planning With Soft Metric Interval Temporal Logic in Unknown Dynamic Environment

Zhiliang Li, Mingyu Cai, *Member, IEEE*, Shaoping Xiao, and Zhen Kan, *Member, IEEE*

***Abstract*—Motion planning of an autonomous system with high-level specifications has wide applications. However, research of formal languages involving timed temporal logic is still under investigation. Challenges arise when the operating environment is dynamic and unknown since the environment can be found prohibitive, leading to potentially conflicting tasks where pre-specified missions cannot be fully satisfied. Such issues become even more challenging when considering time-bound requirements. This letter proposes a control framework to address these challenges, considering hard constraints to enforce safety requirements and soft constraints to enable task relaxation. The metric interval temporal logic (MITL) specifications are employed to deal with time constraints. By constructing a relaxed timed product automaton, an online motion planning strategy is synthesized with a receding horizon controller to generate policies, achieving multiple objectives in decreasing order of priority 1) formally guarantee the satisfaction of hard safety constraints; 2) mostly fulfill soft timed tasks; 3) collect time-varying rewards as much as possible. Simulation results are provided to validate the proposed approach.**

***Index Terms*—Receding horizon control, formal methods in motion planning, timed temporal logic.**

I. Introduction

COMPLEX rules in modern tasks often specify desired system behaviors and timed temporal constraints that require mission completion within a given period. Performing such tasks can be challenging, especially when the operating environment is dynamic and unknown. For instance, user-specified missions or temporal constraints can be found infeasible during motion planning. Therefore, this letter is motivated for online motion planning subject to timed high-level specifications. Linear temporal logic (LTL) has been widely used for task and motion planning due to its rich expressivity and resemblance to natural language [1]. When considering timed formal language, metric interval temporal logic (MITL) [2]–[4] and signal temporal logic(STL) [5] are often employed. However, most existing results are built on the assumption that user-specified tasks are feasible. New challenges arise when the operating environment is dynamic and unknown since the environment can become prohibitive, leading to mission failure.

To address these challenges, temporal logic specifications are often relaxed to be fulfilled as much as possible. A least-violating control strategy is developed in [6]–[9] to enforce the revised motion planning close to the original LTL specifications. In [10]–[12], hard and soft constraints are considered so that the satisfaction of hard constraints is guaranteed while soft constraints are minimally violated. Receding horizon control (RHC) is integrated with temporal logic specifications to deal with motion planning in dynamic environments [13]–[15]. The results mentioned above do not consider time constraints in motion planning. Although the works [11], [12] take into account time constraints, only purely static environments are considered. It is not yet understood how timed temporal tasks can be successfully managed in a dynamic and unknown environment, where predefined tasks may be infeasible. This letter considers online motion planning of an autonomous system with timed temporal specifications. Unlike STL defined over predicates, MITL provides more general time constraints and can express tasks over infinite horizons. Furthermore, MITL can be translated into timed automata that allow us to exploit graph-theoretical approaches for analysis and design. Therefore, MITL is used in this letter.

The contributions are multi-fold. First, the operating environment is not fully known a priori and dynamic in the sense of containing mobile obstacles and time-varying areas of interest that can only be observed locally. The dynamic and unknown environment can lead to potentially conflicting tasks (i.e., the prespecified MITL missions or time constraints cannot be fully satisfied). Inspired by our previous work [15], the motivation

Manuscript received October 24, 2021; revised December 20, 2021; accepted January 6, 2022. Date of publication January 21, 2022; date of current version February 1, 2022. This work was supported in part by the National Natural Science Foundation of China under Grant U2013601 and Grant 62173314. Recommended by Senior Editor C. Seatzu. *(Corresponding author: Mingyu Cai.)*

Zhiliang Li and Zhen Kan are with the Department of Automation, University of Science and Technology of China, Hefei 230026, China.

Mingyu Cai is with the Department of Mechanical Engineering and Mechanics, Lehigh University, Bethlehem, PA 18018 USA (e-mail: mingyucai0915@gmail.com).

Shaoping Xiao is with the Department of Mechanical Engineering, University of Iowa, Iowa City, IA 52242 USA.

Digital Object Identifier 10.1109/LCSYS.2022.3145058

behind the design of hard and soft constraints is that safety is crucial in real-world applications; therefore, we formulate safety requirements (e.g., avoid obstacles) as hard constraints that cannot be violated in all cases. In contrast, soft constraints can be relaxed if the environment does not permit such specifications so that the agent can accomplish the tasks as much as possible. Second, to deal with time constraints, we apply MITL specifications to model timed temporal tasks and further classify soft constraints by how they can be violated. For instance, the mission can fail because the agent cannot reach the destination on time or visit risky regions. Therefore, the innovation considers violations of both time constraints and task specifications caused by dynamic obstacles, which can be formulated as continuous and discrete types, respectively.

Our framework is to generate controllers achieving multiple objectives in decreasing order of priority: 1) formally guarantee the satisfaction of hard constraints; 2) mostly satisfy soft constraints (i.e., minimizing the violation cost); and 3) collect time-varying rewards as much as possible (e.g., visiting areas of higher interest more often). Different from [13] that assumes the LTL specifications can be exactly achieved, we relax the assumption and consider tasks with time constraints described by MITL formulas. Unlike [11], [12], we consider a dynamic and unknown environment where the agent needs to detect and update in real-time. In particular, a multi-objective RHC is synthesized online to adapt to the dynamic environment, which guarantees the safety constraint and minimum violation of the soft specification. Furthermore, it's worth noting that the RHC only considers local dynamic information online while global satisfaction is formally guaranteed, which is efficient for large-scale environments. Finally, we demonstrate the effectiveness of our algorithm by a complex infinite task in simulation.

II. Preliminaries

Definition 1 [1]: A dynamical system with finite states evolving in an environment can be modeled by a weighted transition system (WTS), which is a tuple $\mathcal{T} = (Q, q_0, \delta, \mathcal{AP}, L, \omega)$, where Q is a finite set of states; $q_0 \in Q$ is the initial state; $\delta \in Q \times Q$ is the state transitions; $\mathcal{AP}$ is the finite set of atomic propositions; $L : Q \to 2^{\mathcal{AP}}$ is a labeling function, and $\omega : \delta \to \mathbb{R}^+$ is a weight function.

A timed run of a WTS $\mathcal{T}$ is an infinite sequence $\boldsymbol{r} = (q_0, \tau_0)(q_1, \tau_1)\ldots$, where $\boldsymbol{q} = q_0q_1\ldots$ is a trajectory with $q_i \in Q$, and $\boldsymbol{\tau} = \tau_0\tau_1\ldots$ is a time sequence with $\tau_0 = 0$ and $\tau_{i+1} = \tau_i + \omega(q_i, q_{i+1}), \forall i \geq 0$. The timed run $\boldsymbol{r}$ generates a timed word $\boldsymbol{w} = (\sigma_0, \tau_0)(\sigma_1, \tau_1)\ldots$ where $\boldsymbol{\sigma} = \sigma_0\sigma_1\ldots$ is an infinite word with $\sigma_i = L(q_i)$ for $i \geq 0$. Let $R_k(q)$ denote the time-varying reward associated with a state q at time k. The reward reflects the time-varying objective in the environment. Given a predicted trajectory $\boldsymbol{q}_k = q_0q_1\ldots q_N$ at time k with a finite horizon N, the accumulated reward along the trajectory $\boldsymbol{q}_k$ can be computed as $\boldsymbol{R}_k(\boldsymbol{q}_k) = \sum_{i=1}^{N} R_k(q_i)$.

This letter mainly studies high-level planning and decision-making problems. We assume low-level controllers can achieve go-to-goal navigation, which can be abstracted by WTS. We further assume that the workspace boundaries are known, which is common in literature [3], [4], [11]–[13].

A. Metric Interval Temporal Logic

Metric interval temporal logic is a specific temporal logic that includes timed temporal specification [4]. The syntax of MITL formulas are defined as $\phi := p|\neg\phi|\phi_1 \wedge \phi_2|\Diamond_I\phi|\Box_I\phi|\phi_1\mathcal{U}_\mathcal{I}\phi_2$, where $p \in \mathcal{AP}$, $\wedge$(conjunction) and $\neg$(negation) are Boolean operators, and $\Diamond_I$(eventually), $\Box_I$(always), and $\mathcal{U}_\mathcal{I}$(until) are temporal operators bounded by the non-empty time interval $I = [a, b]$ with $a, b \in \mathbb{R}_{\geq 0}, b > a$. In this letter, we consider the class of MITL formulas with no nested temporal operators. Then the operators can be classified as temporally bounded operators that satisfy $b \neq \infty$, and non-temporally bounded operators otherwise. A formula ϕ containing a temporally bounded operator is called a temporally bounded formula. The same definition holds for non-temporally bounded formulas.

Given a timed run $\boldsymbol{r}$ of $\mathcal{T}$ and an MITL formula ϕ, let $(\boldsymbol{r}, i)$ denote the indexed element (q_i, τ_i). Then the satisfaction relationship $\models$ of MITL can be defined as:

$$\begin{aligned}
&(\boldsymbol{r}, i) \models p \Longleftrightarrow p \in L(q_i)\\
&(\boldsymbol{r}, i) \models \neg\phi \Longleftrightarrow (\boldsymbol{r}, i) \not\models \phi\\
&(\boldsymbol{r}, i) \models \phi_1 \wedge \phi_2 \Longleftrightarrow (\boldsymbol{r}, i) \models \phi_1 \text{ and} (\boldsymbol{r}, i) \models \phi_2\\
&(\boldsymbol{r}, i) \models \Diamond_I\phi \Longleftrightarrow \exists j, i \leq j, s.t. (\boldsymbol{r}, j) \models \phi, \tau_j - \tau_i \in I\\
&(\boldsymbol{r}, i) \models \Box_I\phi \Longleftrightarrow \forall j, i \leq j, \tau_j - \tau_i \in I \Rightarrow (\boldsymbol{r}, j) \models \phi\\
&(\boldsymbol{r}, i) \models \phi_1\mathcal{U}_I\phi_2 \Longleftrightarrow \exists j, i \leq j, s.t. (\boldsymbol{r}, j) \models \phi_2, \tau_j - \tau_i\\
&\qquad\qquad \in \text{I and } (\boldsymbol{r}, k) \models \phi_1 \text{ for every } i \leq k \leq j.
\end{aligned}$$

B. Timed Büchi Automaton

Let $X = \{x_1, x_2, \ldots, x_M\}$ be a finite set of clocks. The set of clock constraints $\Phi(X)$ is defined by the grammar $\varphi := \top|\neg\varphi|\varphi_1 \wedge \varphi_2|x \bowtie c$, where $x \in X$ is a clock, $c \in \mathbb{R}^+$ is a clock constant and $\bowtie \in \{<, >, \geq, \leq, =\}$. A clock valuation $\nu : X \to \mathbb{R}^+$ assigns a real value. We denote by $\nu \models \varphi$ if the valuation ν satisfies the clock constraint φ, where $\nu = (\nu_1, \ldots, \nu_M)$ with ν_i being the valuation of x_i, $\forall i \in 1, \ldots, M$. An MITL formula can be converted into a Timed Büchi Automaton (TBA) [16].

Definition 2: A TBA is a tuple $\mathcal{A} = (S, S_0, \mathcal{AP}, \mathcal{L}, X, I_X, E, F)$, where S is a finite set of states; $S_0 \subseteq S$ is the set of initial states; $2^{\mathcal{AP}}$ is the alphabet where $\mathcal{AP}$ is a finite set of atomic propositions; $\mathcal{L} : S \to 2^{\mathcal{AP}}$ is a labeling function; X is a finite set of clocks; $I_X : S \to \Phi(X)$ is a map from states to clock constraints; $E \subseteq S \times \Phi(X) \times 2^{\mathcal{AP}} \times S$ represents the set of edges of form $e = (s, g, a, s')$ where s, s' are the source and target states, g is the guard of edge via an assigned clock constraint, and $a \in 2^{\mathcal{AP}}$ is an input symbol; $F \subseteq S$ is a set of accepting states.

Definition 3: An automata timed run $\boldsymbol{r}_\mathcal{A} = (s_0, \tau_0)\ldots(s_n, \tau_n)$ of a TBA $\mathcal{A}$, corresponding to the timed run $\boldsymbol{r} = (q_0, \tau_0)\ldots(q_n, \tau_n)$ of a WTS $\mathcal{T}$, is a sequence where $s_0 \in S_0$, $s_j \in S$, and $(s_j, g_j, a_j, s_{j+1}) \in E$ $\forall j \geq 0$ such that i) $\tau_j \models g_j, j \geq 0$, and ii) $L(q_j) \subseteq \mathcal{L}(s_j), \forall j$.

III. Problem Formulation

To consider potentially infeasible specifications, we define the total violation cost of an MITL formula as follows.

Definition 4: Given a time run $\boldsymbol{r} = (q_0, \tau_0) \dots (q_n, \tau_n)$ of a WTS $\mathcal{T}$, the total violation cost of an MITL formula ϕ is defined as

$$\mathcal{W}(\boldsymbol{r}, \phi) = \sum_{k=0}^{n-1} \omega(q_k, q_{k+1}) \omega_v(q_k, q_{k+1}, \phi), \tag{1}$$

where $\omega(q_k, q_{k+1}) = \tau_{k+1} - \tau_k$ is the time required for the transition (q_k, q_{k+1}) and $\omega_v(q_k, q_{k+1}, \phi)$ is defined as the violation cost of the transition with respect to ϕ. Then, the formal statement of the problem is expressed as follows.

Problem 1: Given a weighted transition system $\mathcal{T}$, and an MITL formula $\phi = \phi_h \wedge \phi_s$, the control objective is to design a multi-goal online planning strategy, in decreasing order of priority, in which 1) ϕ_h is fully satisfied; 2) ϕ_s is fulfilled as much as possible if ϕ_s is not feasible, i.e., minimize the total violation cost $\mathcal{W}(\boldsymbol{r}, \phi_s)$; and 3) the agent collects time-varying rewards as much as possible over an infinite horizon during the task execution.

IV. Relaxed Automaton

Section IV-A presents the procedure of constructing the relaxed TBA to allow motion revision. Section IV-B presents the design of energy function that guides the satisfaction of MITL specifications. Section IV-C gives the online update of environment knowledge for motion planning.

A. Relaxed Timed Büchi Automaton

To address the violation of MITL tasks, the relaxed TBA is defined to contain two extra components (i.e., a continuous violation cost and a discrete violation cost) compared with the original TBA. This section presents the procedure of how to construct a relaxed TBA for an MITL formula $\phi = \phi_h \wedge \phi_s$.

Given the hard constraints ϕ_h, which have to be fully satisfied and cannot be violated at any time, we add a sink state $\hat{s}_{sink}$ in the relaxed TBA to indicate the violation of hard constraints.

Before developing soft constraints ϕ_s, a more detailed classification of temporal operators for MITL formulas is introduced. An MITL specification ϕ can be written as $\phi = \bigwedge_{i \in 1,2,\dots,n} \phi_i$ s.t. $\phi_i \neq \phi_j, \forall i \neq j$. For each sub-formula ϕ_i, if it is temporally bounded, ϕ_i can be either satisfied, violated, or uncertain [11]. Except until operator, each non-temporally bounded sub-formula can be classified as either satisfied/uncertain or violated/uncertain. It can be further divided into Type I and Type II, respectively. Specifically, it is of Type I (i.e., satisfied/uncertain) if ϕ_i cannot be concluded to be violated at any time during a run since there remains a possibility for it to be satisfied in the future. In contrast, it is of Type II (i.e., violated/uncertain) if ϕ_i cannot be concluded to be satisfied during a run since it remains possible to be violated in the future. For instance, when $b = \infty$, $\Diamond_{[a,b]}$ is of Type I and $\Box_{[a,b]}$ is of Type II. The until operator $\mathcal{U}_{[a,b]}$ is a special type since it consists of two parts, such that each part can be classified as different types (Type I and Type II), simultaneously.

The possible satisfaction of each sub-formula ϕ_i with respect to an ongoing timed run $\boldsymbol{r}$ can be defined as $\phi_i^{state} \in \{\phi_i^{vio}, \phi_i^{sat}, \phi_i^{unc}\}$, where ϕ_i^{vio} means that ϕ_i has been violated, ϕ_i^{sat} means that ϕ_i has been satisfied, and ϕ_i^{unc} means ϕ_i can't be concluded to be satisfied or violated currently, which depends on future information of the run.

Based on above statement, for the soft constraints $\phi_s = \bigwedge_{i \in 1,\dots,n} \phi_i$, an evaluation set φ_i of a sub-formula ϕ_i which represent possible satisfaction for a sub-formula is defined as

$$\varphi_i = \begin{cases} \{\phi_i^{vio}, \phi_i^{sat}, \phi_i^{unc}\}, & \text{if } \phi_i \text{ is temporally bounded,} \\ \{\phi_i^{sat}, \phi_i^{unc}\}, & \text{if } \phi_i \text{ is non-temporally} \\ & \text{bounded of Type I,} \\ \{\phi_i^{vio}, \phi_i^{unc}\}, & \text{if } \phi_i \text{ is non-temporally} \\ & \text{bounded of Type II,} \end{cases} \tag{2}$$

Based on (2), a sub-formula evaluation ψ_s^j of ϕ_s is defined as

$$\psi_s^j = \bigwedge_i \phi_i^{state}, \phi_i^{state} \in \varphi_i. \tag{3}$$

In (3), ψ_s^j represents one possible outcome of the formula, which can be obtained by taking an element from the evaluation set φ_i for each ϕ_i and then operating the conjunction of all these elements. Each different combination corresponds to a sub-formula evaluation ψ_s^j. Let Ψ_s denote the set of all sub-formula evaluations ψ_s^j of ϕ_s, the number of $\psi_s^j \in \Psi_s$ is equal to the product of the number of elements in the evaluation set φ_i, which can be defined as $\Psi_s = \{\psi_s^j, j = 0, 1 \dots, n-1\}$ with $n = \prod_i |\varphi_i|$ where $|\varphi_i|$ represents the number of elements in set φ_i. The set Ψ_s represents all possible outcomes of ϕ_s at any time. Every possible $\psi_s^j \in \Psi_s$ is associated with a state $\hat{s}$. The initial state $\hat{s}_0$ is the state whose corresponding sub-formulas are uncertain, which indicates no progress has been made. The accepting state $\hat{s}_F$ is the state whose corresponding temporally bounded sub-formulas and non-temporally bounded sub-formulas of Type I are satisfied, while all non-temporally bounded sub-formulas of Type II are uncertain (see Algorithm 1).

The construction of the set of atomic propositions $\mathcal{AP}$, labeling function $\mathcal{L}$, clocks X, and the map from states to clock constraints I_X in the relaxed TBA is the same as in the TBA. Here consider two different types of violation cost, i.e., a state $\hat{s} \neq \hat{s}_{sink}$ can violate soft constraints ϕ_s by either continuous violation (e.g., violating time constraints) or discrete violation (e.g., visiting risky regions). To measure their violation degrees, the outputs of continuous violation cost $v_c(\hat{s})$ and discrete violation cost $v_d(\hat{s})$ for each state $\hat{s} \neq \hat{s}_{sink}$ are defined, respectively, as

$$v_c(\hat{s}) = \begin{cases} k, & \text{if } \exists \phi_1^{\mathrm{vio}}, \phi_2^{\mathrm{vio}}, \dots \phi_{\mathrm{k}}^{\mathrm{vio}} \in \psi_{\mathrm{s}}^{\mathrm{j}} \text{ that is temporally} \\ & \text{bounded,} \\ 0, & \text{otherwise,} \end{cases} \tag{4}$$

$$v_d(\hat{s}) = \begin{cases} 1, & \text{if } \exists \phi_{\mathrm{i}}^{\mathrm{vio}} \in \psi_{\mathrm{s}}^{\mathrm{j}} \text{ that is non-temporally bounded,} \\ 0, & \text{otherwise,} \end{cases} \tag{5}$$

At the sink state $\hat{s}_{sink}$, the continuous and discrete violation costs are defined as $v_c(\hat{s}_{sink}) = v_d(\hat{s}_{sink}) = \infty$. The next step is to define violations-based edges to connect states.

Definition 5: Given soft constraints ϕ_s, the distance set between ψ_s and ψ_s' is defined as $|\psi_s - \psi_s'| = \{\phi_i : \phi_i^{state'} \neq$

Algorithm 1 Construct Set of States $\hat{S}$, Initial States $\hat{S}_0$ and Accepting States $\hat{F}$ of a Relaxed TBA

```
1: procedure INPUT: (MITL specification $\phi = \phi_h \wedge \phi_s$ )
       Output: $\hat{S}, \hat{S}_0, \hat{F}$
2:     construct state relevant to $\phi_h$:
3:     add a state $\hat{s}_{sink}$
4:     construct states relevant to $\phi_s$:
5:     $\Phi_s = \{\phi_i : \phi_s = \bigwedge_i \phi_i\}$
6:     for $\phi_i \in \Phi_s$ do
7:         if $\phi_i$ is temporally bounded then
8:             $\varphi_i = \{\phi_i^{sat}, \phi_i^{vio}, \phi_i^{unc}\}$;
9:         else if $\phi_i$ is non-temporally bounded of Type I then
10:            $\varphi_i = \{\phi_i^{sat}, \phi_i^{unc}\}$;
11:        else $\varphi_i = \{\phi_i^{vio}, \phi_i^{unc}\}$;
12:        end if
13:    end for
14:    $\psi_s^j = \bigwedge_i \phi_i^{state}, \phi_i^{state} \in \varphi_i$;
15:    $\Psi_s = \left\{\psi_s^j : j = 0, 1 \dots, n-1\right\}$ with $n = \prod_i |\varphi_i|$;
16:    $\hat{S} = \{\hat{s}_k : k = 0, 1 \dots, n\}$;
17:    $\hat{S}_0 = \hat{s}_0$, where $\hat{s}_0$ corresponds to $\psi_s^0 = \bigwedge_i \phi_i^{unc}$;
18:    $\hat{F} = \hat{s}_F$, where $\hat{s}_F$ corresponds to $\psi_s^F = \bigwedge_{i_1 \in I_1} \phi_{i_1}^{sat} \cap \bigwedge_{i_2 \in I_2} \phi_{i_2}^{unc}$, where $i_1 \in I_1$ are the indexes of sub-formulas of $\phi_s$ that are either temporally bounded or of Type I, and $i_2 \in I_2$ are the indexes of sub-formulas that are of Type II;
19: end procedure
```

$\phi_i^{state}\}$. That is, it consists of all sub-formulas ϕ_i that are under different evaluations.

We use $(\psi_s, g, a) \to \psi_s'$ to denote that all sub-formulas $\phi_i \in |\psi_s - \psi_s'|$ are (i) evaluated as uncertain in ψ_s (i.e., $\phi_i^{unc} \in \psi_s$) and (ii) re-evaluated to be either satisfied or violated in ψ_s' (i.e., $\phi_i^{state'} \in \psi_s'$, where $state' \in \{vio, sat\}$) if symbol a, which is read at time t, satisfies guard g. The edge construction can be summarized into four steps:

- Construct all edges corresponding to progress regarding the specifications (i.e., the edges that a TBA would have).
- Construct edges $\hat{E}$ of non-temporally bounded soft constraints that are no longer violated, such that $(\hat{s}, g, a, \hat{s}') \in \hat{E}$ satisfying all of the following conditions: (i) $\forall \phi_i \in |\psi_s - \psi_s'|$, $\phi_i^{vio} \in \psi_s$ where $\hat{s}$ corresponds to ψ_s, $\hat{s}'$ corresponds to ψ_s' and ϕ_i is non-temporally bounded, and (ii) $(\hat{s}'', g, a, \hat{s}') \in \hat{E}$ for some $\hat{s}''$ where $|\psi_s - \psi_s'| = |\psi_s - \psi_s''|$ or $(\hat{s}', g, a', \hat{s}) \in \hat{E}$ where $a' = 2^{\mathcal{AP}} \backslash a$.
- Construct edges $\hat{E}$ of temporally bounded soft constraints that are no longer violated, such that $(\hat{s}, g, a, \hat{s}') \in \hat{E}$ satisfies all the following conditions: (i) $\exists \phi_i \in |\psi_s - \psi_s'|$, $\phi_i^{vio} \in \psi_s$, $\phi_i^{sat} \in \psi_s'$, $\phi_i^{unc} \in \psi_s''$ where $\hat{s}$ corresponds to ψ_s, $\hat{s}'$ corresponds to ψ_s', $\hat{s}''$ corresponds to ψ_s'' and ϕ_i is temporally bounded; (ii) $(\hat{s}'', g', a, \hat{s}') \in \hat{E}$, $(\hat{s}'', g, a, \hat{s}) \in \hat{E}$ and $g = g' \backslash \Phi(X_i)$, where X_i is the set of clocks associated with ϕ_i, s.t. $\phi_i^{unc} \in \psi_s''$ and $\phi_i^{vio} \in \psi_s$.
- Construct self-loops such that $(\hat{s}, g, a, \hat{s}) \in \hat{E}$ if $\exists$ (g, a) s.t. $g \subseteq g'$, $a \subseteq a'$ where $(\hat{s}', g', a', \hat{s}) \in \hat{E}$ for some $\hat{s}'$ and $(\hat{s}, g', a', \hat{s}'') \notin \hat{E}$ for any $\hat{s}''$.

In the first step, the edges of the original TBA are constructed except self-loops. Then, we construct edges from states where $v_d = 1$, i.e., states corresponding to discrete violation (step 2). These edges can be considered as alternative routes to the ones in step 1, where some non-temporally bounded sub-formula/formulas are violated at some points. Similarly, we construct edges from states with $v_c > 0$, i.e., states corresponding to continuous violations (step 3), which ensures that the accepting states can be reached when the time-bound action finally occurs, even after the deadline is exceeded. Finally, we add self-loops to ensure no deadlocks in the automaton except the sink state $\hat{s}_{sink}$.

Definition 6: An automata timed run $\boldsymbol{r}_{\hat{\mathcal{A}}} = (\hat{s}_0, \tau_0) \dots (\hat{s}_n, \tau_n)$ of a relaxed TBA $\hat{\mathcal{A}}$, corresponding to the timed run $\boldsymbol{r} = (q_0, \tau_0) \dots (q_n, \tau_n)$ is a sequence where $\hat{s}_0 \in \hat{S}_0$, $\hat{s}_j \in \hat{S}$, and $(\hat{s}_j, g_j, a_j, \hat{s}_{j+1}) \in \hat{E}$ $\forall j \geq 0$ such that i) $\tau_j \models g_j, j \geq 0$, and ii) $L(q_j) \subseteq \mathcal{L}(\hat{s}_j), \forall j$. The continuous violation cost for the automata timed run is $\sum_{k=0}^{n-1} v_c(\hat{s}_{k+1})(\tau_{k+1} - \tau_k)$ and similarly the discrete violation cost is $\sum_{k=0}^{n-1} v_d(\hat{s}_{k+1})(\tau_{k+1} - \tau_k)$.

An example of relaxed TBA can be found in [17]. Then, a relaxed product automaton is introduced for infeasible cases.

Definition 7: Given a WTS $\mathcal{T} = (\mathrm{Q}, \mathrm{q}_0, \delta, \mathcal{AP}, \mathrm{L}, \omega)$ and a relaxed TBA $\hat{\mathcal{A}} = (\hat{S}, \hat{S}_0, \mathcal{AP}, \mathcal{L}, X, I_X, v_c, v_d, \hat{E}, \hat{F})$, the relaxed product automaton (RPA) $\hat{\mathcal{P}} = \mathcal{T} \times \hat{\mathcal{A}}$ is defined as a tuple $\hat{\mathcal{P}} = \{\hat{P}, \hat{P}_0, \mathcal{AP}, L_{\hat{\mathcal{P}}}, \delta_{\hat{\mathcal{P}}}, I_X^{\hat{\mathcal{P}}}, v_c^{\hat{\mathcal{P}}}, v_d^{\hat{\mathcal{P}}}, \mathcal{F}_{\hat{\mathcal{P}}}, \omega_{\hat{\mathcal{P}}}\}$, $\hat{P} \subseteq \{(q, \hat{s}) \in Q \times \hat{S} : L(q) \subseteq \mathcal{L}(\hat{s})\}$ is the set of states; $\hat{P}_0 = \{q_0\} \times \hat{S}_0$ is the set of initial states; $L_{\hat{\mathcal{P}}} = \hat{P} \to 2^{\mathcal{AP}}$ is a labeling function, i.e., $L_{\hat{\mathcal{P}}}(\hat{p}) = L(q)$; $\delta_{\hat{\mathcal{P}}} \subseteq \hat{P} \times \hat{P}$ is the set of transitions defined such that $((q, \hat{s}), (q', \hat{s}')) \in \delta_{\hat{\mathcal{P}}}$ if and only if $(q, q') \in \delta$ and $\exists g, a,$ s.t. $(\hat{s}, g, a, \hat{s}') \in \hat{E}$; $I_X^{\hat{\mathcal{P}}}(\hat{p}) = I_X(\hat{s})$ is a map of clock constraints; $v_c^{\hat{\mathcal{P}}}(\hat{p}) = v_c(\hat{s})$ is the continuous violation cost; $v_d^{\hat{\mathcal{P}}}(\hat{p}) = v_d(\hat{s})$ is the discrete violation cost; $\mathcal{F}_{\hat{\mathcal{P}}} = Q \times \hat{F}$ are accepting states; $\omega_{\hat{\mathcal{P}}}: \delta_{\hat{\mathcal{P}}} \to \mathbb{R}^+$ is the positive weight function, i.e., $\omega_{\hat{\mathcal{P}}}(\hat{p}, \hat{p}') = \omega(q, q')$.

By accounting both continuous and discrete violation, the violation cost with respect to ϕ_s is defined as

$$\omega_v^{\hat{\mathcal{P}}}(\hat{p}_k, \hat{p}_{k+1}, \phi_s) = (1-\alpha) v_c^{\hat{\mathcal{P}}}(\hat{p}_{k+1}) + \alpha v_d^{\hat{\mathcal{P}}}(\hat{p}_{k+1}), \quad (6)$$

where $\alpha \in [0, 1]$ measures the trade-off between continuous and discrete violations. Then based on $\mathcal{W}(\boldsymbol{r}, \phi)$ defined in (1), the total weight of a path $\hat{\boldsymbol{p}} = (q_0, \hat{s}_0) \dots (q_n, \hat{s}_n)$ for $\hat{\mathcal{P}}$ is

$$\mathcal{W}(\hat{\boldsymbol{p}}) = \sum_{k=0}^{n-1} \omega_{\hat{\mathcal{P}}}(\hat{p}_k, \hat{p}_{k+1}) \omega_v^{\hat{\mathcal{P}}}(\hat{p}_k, \hat{p}_{k+1}, \phi_s), \quad (7)$$

where $\mathcal{W}(\hat{\boldsymbol{p}})$ measures the total violations with respect to ϕ_s in the WTS. Hence, by minimizing the violation of ϕ_s, a run $\hat{\boldsymbol{p}}$ of $\hat{\mathcal{P}}$ can fulfill ϕ_s as much as possible.

B. Energy Function

Inspired by [15], we design a hybrid Lyapunov-like energy function over RPA consisting of different violation costs. Such a design can measure the minimum distance to the accepting sets from the current state and enforce the acceptance condition by decreasing the energy as the system evolves. For $\hat{p} \in \hat{P}$, we design the energy function as

$$J(\hat{p}) = \begin{cases} \min_{\hat{p}' \in \mathcal{F}^*} d(\hat{p}, \hat{p}'), & \text{if } \hat{p} \notin \mathcal{F}^*, \\ 0, & \text{if } \hat{p} \in \mathcal{F}^*, \end{cases} \quad (8)$$

where $d(\hat{p}_i, \hat{p}_j) = \min_{\hat{\boldsymbol{p}} \in \hat{\mathcal{D}}(\hat{p}_i, \hat{p}_j)} \mathcal{W}(\hat{\boldsymbol{p}})$, $\hat{\mathcal{D}}(\hat{p}_i, \hat{p}_j)$ is the set of all possible paths, and $\mathcal{F}^*$ is the largest self-reachable subset of the accepting set $\mathcal{F}_{\hat{\mathcal{P}}}$. Since $\omega_{\hat{\mathcal{P}}}$ is positive by definition, $d(\hat{p}, \hat{p}') > 0$ for all $\hat{p}, \hat{p}' \in \hat{P}$, which implies that $J(\hat{p}) \geq 0$. Particularly, $J(\hat{p}) = 0$ if $\hat{p} \in \mathcal{F}^*$. If a state in $\mathcal{F}^*$ is reachable from $\hat{p}$, then $J(\hat{p}) \neq \infty$, otherwise $J(\hat{p}) = \infty$. Therefore, $J(\hat{p})$ indicates the minimum distance from $\hat{p}$ to $\mathcal{F}^*$. As long as the energy function keeps decreasing, the generated path will eventually satisfy the acceptance condition of $\hat{\mathcal{P}}$.

C. Automaton Update

The system model needs to be updated according to the sensed information during the runtime to facilitate motion planning. Let $\text{Info}(\hat{p}) = \{L_{\hat{\mathcal{P}}}(\hat{p}') | \hat{p}' \in \text{Sense}(\hat{p})\}$ denote the newly observed labels of $\hat{p}'$ that are different from the current knowledge, where $\text{Sense}(\hat{p})$ represents neighbor states that the agent at current state $\hat{p}$ can detect and observe. Denote the sensing range by N_s. If the sensed labels $L_{\hat{\mathcal{P}}}(\hat{p}')$ are consistent with the current knowledge of $\hat{p}'$, $\text{Info}(\hat{p}) = \emptyset$; otherwise, the properties of $\hat{p}'$ have to be updated. Let $\boldsymbol{J} \in \mathbb{R}^{|\hat{P}|}$ denote the stacked J for all $\hat{p} \in \hat{P}$. The terms $\boldsymbol{J}$ are initialized from the initial knowledge of the environment. At each step, if $\text{Info}(\hat{p}) \neq \emptyset$, the weight $\omega_{\hat{\mathcal{P}}}(\hat{p}', \hat{p}'')$ and $\omega_{\hat{\mathcal{P}}}(\hat{p}'', \hat{p}')$ for states that satisfy $\hat{p}' \in \text{Sense}(\hat{p})$ and $(\hat{p}', \hat{p}'') \in \delta_{\hat{\mathcal{P}}}$ are updated. Then the energy function $\boldsymbol{J}$ is updated.

Lemma 1: The largest self-reachable set $\mathcal{F}^*$ remains the same during the automaton update.

Proof: Given $\hat{\mathcal{P}}_{(\hat{p}, \delta_{\hat{\mathcal{P}}})}$, the graph induced from $\hat{\mathcal{P}}_{(\hat{p}, \delta_{\hat{\mathcal{P}}})}$ by neglecting the weight of each transition is denoted by $\mathcal{G}(\hat{p}, \delta_{\hat{\mathcal{P}}})$. The update process only updates the cost of each transition s.t. the topological structure of $\mathcal{G}(\hat{p}, \delta_{\hat{\mathcal{P}}})$ remains the same. ∎

The construction of $\mathcal{F}^*$ involves the computation of $d(\hat{p}, \hat{p}')$ for all $\hat{p}' \in \mathcal{F}_{\hat{\mathcal{P}}}$ and the check of terminal conditions [13]. Lemma 1 indicates that $\mathcal{F}^*$ doesn't need to be updated whenever newly sensed information caused by unknown obstacles is obtained. Therefore, it reduces the complexity, and $\mathcal{F}^*$ is computed offline.

V. Control Synthesis of MITL Motion Planning

The control synthesis of the MITL motion planning strategy is based on receding horizon control (RHC). Specifically, for the current state $\hat{p}_k$, let $\hat{\boldsymbol{p}}_k = \hat{p}_{1|k}\hat{p}_{2|k} \ldots \hat{p}_{N|k}$ denote a predicted path of horizon N at time k starting from $\hat{p}_k$, where $\hat{p}_{i|k} \in \hat{P}$ satisfies $(\hat{p}_{i|k}, \hat{p}_{i+1|k}) \in \delta_{\hat{\mathcal{P}}}$ for all $i = 1, \ldots, N-1$, and $(\hat{p}_k, \hat{p}_{1|k}) \in \delta_{\hat{\mathcal{P}}}$. Let $\text{Path}(\hat{p}_k, N)$ be the set of paths of horizon N generated from $\hat{p}_k$. Note that a predicted path $\hat{\boldsymbol{p}}_k \in \text{Path}(\hat{p}_k, N)$ can uniquely project to a trajectory $\gamma_{\mathcal{T}}(\hat{\boldsymbol{p}}_k) = \boldsymbol{q} = q_1 \cdots q_N$ on $\mathcal{T}$, where $\gamma_{\mathcal{T}}(\hat{p}_{i|k}) = q_i$, $\forall i = 1, \ldots, N$. The choice of the finite horizon N depends on the local sensing range N_s of the agent. The total reward along the predicted path $\hat{\boldsymbol{p}}_k$ is $\boldsymbol{R}(\gamma_{\mathcal{T}}(\hat{\boldsymbol{p}}_k)) = \overset{[}{i} = 1]N\sum R_k(\gamma_{\mathcal{T}}(\hat{p}_{i|k}))$. Based on (7), for every predicted path $\hat{\boldsymbol{p}}_k$ the total violation cost is $\mathcal{W}(\hat{\boldsymbol{p}}_k)$. Then the utility function of RHC is designed as

$$\mathbf{U}(\hat{\boldsymbol{p}}_k) = \boldsymbol{R}(\gamma_{\mathcal{T}}(\hat{\boldsymbol{p}}_k)) - \beta \mathcal{W}(\hat{\boldsymbol{p}}_k), \tag{9}$$

where β is the relative penalty. By applying large β, maximizing the utility $\mathbf{U}(\hat{\boldsymbol{p}}_k)$ tends to bias the selection of paths towards the objectives, in the decreasing order, of 1) hard constraints ϕ_h satisfaction, 2) fulfilling soft constraints ϕ_s as much as possible, and 3) collecting time-varying rewards as much as possible. Note that continuous and discrete violations are optimized simultaneously based on the preference weight α in $\mathcal{W}(\hat{\boldsymbol{p}}_k)$. To satisfy the acceptance condition of $\hat{\mathcal{P}}$, we consider the energy function-based constraints simultaneously.

Theorem 1: For each time-step k = 0, 1, 2, ..., the optimal path $\hat{\boldsymbol{p}}^* = \hat{p}_0^* \hat{p}_1^* \ldots$ is guaranteed to satisfy the acceptance condition by applying the following RHC:

$$\hat{\boldsymbol{p}}_{k,opt} = \underset{\hat{\boldsymbol{p}}_k \in \text{Path}(\hat{p}_{k-1}^*, N)}{\operatorname{argmax}} \mathbf{U}(\hat{\boldsymbol{p}}_k), \tag{10}$$

subject to the following constraints:

1) $J(\hat{p}_0) < \infty$ if $k = 0$;
2) $J(\hat{p}_{N|k}) < J(\hat{p}_{N|k-1,opt})$ if $J(\hat{p}_{k-1}^*) > 0$ and $J(\hat{p}_{i|k-1,opt}) \neq 0$ for all $i = 1, \ldots, N$;
3) $J(\hat{p}_{i_0(\hat{p}_{k-1,opt})-1|k}) = 0$ if $J(\hat{p}_{k-1}^*) > 0$ and $J(\hat{p}_{i|k-1,opt}) = 0$ for some $i = 1, \ldots, N$, where $i_0(\hat{p}_{k-1,opt})$ is the index of the first occurrence that satisfies $J(\hat{p}_{i_0|k-1,opt}) = 0$ in $\hat{\boldsymbol{p}}_{k-1,opt}$;
4) $J(\hat{p}_{N|k}) < \infty$ if $J(\hat{p}_{k-1}^*) = 0$.

Since the environment is dynamic and unknown, the agent will update the environment according to the detected information at each time step. In addition, by selecting the predictive horizon N to be less than or equal to the sensor range N_s, we can ensure the existence of the solutions, since the local environment can be regarded as static. The detailed proof can be found in Theorem 1 and 2 in [17].

Due to the unrelaxed hard constraint, we can restrict the agent to avoid collisions at each time step based on the sensor information. We assume the local information of WTS can be accurately updated such that the hard constraint is guaranteed. The system will return no solution in cases where no feasible trajectories satisfy the hard constraint, e.g., obstacles surrounding the agent. Note that the optimality in this letter refers to local optimum since RHC controllers only optimize the objective within finite predictive steps.

The control synthesis of the MITL online motion planning strategy is presented in the form of Algorithm 2. Lines 2-3 are responsible for the offline initialization to obtain an initial $\boldsymbol{J}$. The rest of Algorithm 2 (lines 4-16) is the online receding horizon control part executed at each time step. In Lines 4-6 the receding horizon control is applied to determine $\hat{p}_0^*$ at time $k = 0$. Since the environment is dynamic and unknown, Algorithm 2 is applied at each time $k > 0$ to update $\boldsymbol{J}$ based on local sensing in Lines 7-9. The RHC is then employed based on the previously determined $\hat{p}_{k-1}^*$ to generate $\hat{\boldsymbol{p}}_{k,opt}$, where the next state is determined as $\hat{p}_k^* = \hat{p}_{1|k,opt}$ in Lines 10-12. The transition from $\hat{p}_{k-1}^*$ to $\hat{p}_k^*$ applied on $\hat{\mathcal{P}}$ corresponds to the movement of the agent at time k from $\gamma_{\mathcal{T}}(\hat{p}_{k-1}^*)$ to $\gamma_{\mathcal{T}}(\hat{p}_k^*)$ on $\mathcal{T}$ in Line 11. By repeating the process in lines 7-13, an optimal path $\hat{\boldsymbol{p}}^* = \hat{p}_0^* \hat{p}_1^* \ldots$ can be obtained to satisfy the acceptance condition of $\hat{\mathcal{P}}$.

Algorithm 2 MITL Based Online Motion Planning

```
1: procedure INPUT: (The WTS $\mathcal{T} = (Q, q_0, \delta, \mathcal{AP}, L, \omega)$ and the relax TBA $\hat{\mathcal{A}} = (\hat{S}, \hat{S}_0, \mathcal{AP}, \mathcal{L}, X, I_X, v_c, v_d, \hat{E}, \hat{F})$ corresponding to the MITL formula $\phi = \phi_h \wedge \phi_s$)
       Output: the path $\hat{\boldsymbol{p}}^* = \hat{p}_0^* \hat{p}_1^* \dots$
       Off-line Execution:
2:     Construct the RPA $\hat{\mathcal{P}} = \mathcal{T} \times \hat{\mathcal{A}}$
3:     Construct $\mathcal{F}^*$ and initialize $\boldsymbol{J}$
       Online Execution:
4:     if $\exists \hat{p_0} \in \hat{P}_0, J(\hat{p_0}) < \infty$ then
5:         Solve for $\hat{\boldsymbol{p}}_{0,opt}$
6:         $\hat{p}_0^* = \hat{p}_{1|0,opt}$ and $k \leftarrow 1$
7:         while $k > 0$ do
8:             Automaton update at $\hat{p}_{k-1}^*$ based on local sensing
9:             Locally observe rewards $\boldsymbol{R}(\gamma_{\mathcal{T}}(\hat{p}_{k-1}^*))$
10:            Solve for $\hat{\boldsymbol{p}}_{k,opt}$
11:            Implement corresponding transitions on $\hat{\mathcal{P}}$ and $\mathcal{T}$
12:            $\hat{p}_k^* = \hat{p}_{1|k,opt}$ and $k++$
13:        end while
14:    else There does not exist an accepting run from initial states
15:    end if
16: end procedure
```

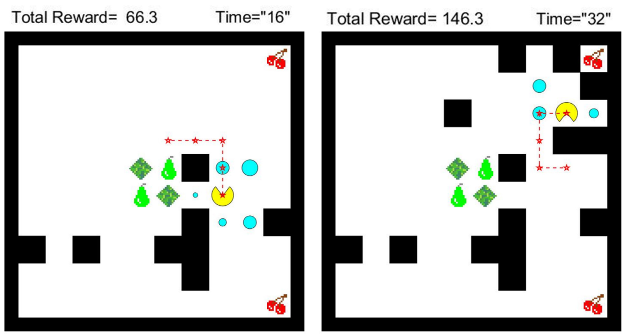

Fig. 1. Snapshots of the motion planning. Cyan dots represent the rewards with size proportional to their value. The red dotted arrow line represents the predicted trajectory. The left picture indicates Pac-Man chooses to violate the temporally bounded operator when the soft task is infeasible. The right one shows Pac-Man revises its motion plan to go to the cherry at the right bottom corner since the right top one is not accessible.

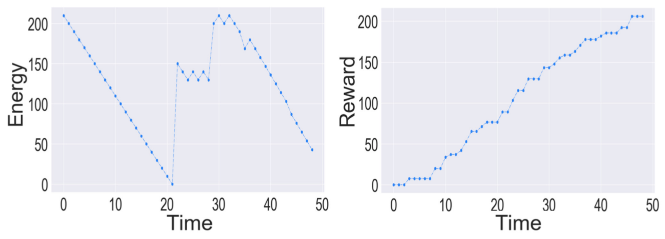

Fig. 2. Energy function and accumulative collected time-varying rewards.

VI. Case Studies

Consider an MITL specification $\phi = \phi_h \wedge \phi_s$, where $\phi_h = \square\neg\texttt{obstacle}$ and $\phi_s = \square(\neg\texttt{grass}) \wedge \square\lozenge_{t<10}\texttt{cherry} \wedge \square(\texttt{cherry} \rightarrow \lozenge_{t<20}\texttt{pear})$ in a Pac-Man game. In English, ϕ_h means the agent has to always avoid obstacles, and ϕ_s indicates the agent needs to repeatedly and sequentially eat pears and cherries within the specified time intervals while avoiding the grass. Fig. 1 shows the snapshots during mission operation. Fig. 2 shows the evolution of the energy function and the collected rewards during mission operation. The simulation video is provided.[1] More detailed results about the environment settings, analysis of scalability and computational complexity of the algorithm can be found in [17].

VII. Conclusion

In this letter, we propose a relaxed timed product automaton to handle hard and soft constraints given as MITL specifications. An online motion planning strategy is synthesized with a receding horizon controller to achieve multi-objective tasks in a dynamic and unknown environment. Future research will consider building the deterministic system online based on the real-time sensing information and develop online robust planning methods for stochastic systems.

References

[1] C. Baier and J.-P. Katoen, *Principles of Model Checking*. Cambridge, MA, USA: MIT Press, 2008.
[2] R. Alur, T. Feder, and T. A. Henzinger, "The benefits of relaxing punctuality," *J. ACM*, vol. 43, no. 1, pp. 116–146, 1996.
[3] A. Nikou, J. Tumova, and D. V. Dimarogonas, "Cooperative task planning of multi-agent systems under timed temporal specifications," in *Proc. IEEE Amer. Control Conf.*, Boston, MA, USA, 2016, pp. 7104–7109.
[4] A. Nikou, D. Boskos, J. Tumova, and D. V. Dimarogonas, "On the timed temporal logic planning of coupled multi-agent systems," *Automatica*, vol. 97, pp. 339–345, Nov. 2018.
[5] O. Maler and D. Nickovic, "Monitoring temporal properties of continuous signals," in *Proc. Formal Techn. Model. Anal. Timed Fault Tolerant Syst.*, 2004, pp. 152–166.
[6] J. Tumova, S. Karaman, C. Belta, and D. Rus, "Least-violating planning in road networks from temporal logic specifications," in *Proc. Int. Conf. Cyber-Phys. Syst.*, Vienna, Austria, 2016, pp. 1–9.
[7] C.-I. Vasile, J. Tumova, S. Karaman, C. Belta, and D. Rus, "Minimum-violation scLTL motion planning for mobility-on-demand," in *Proc. Int. Conf. Robot. Autom.*, Singapore, 2017, pp. 1481–1488.
[8] M. Cai, S. Xiao, Z. Li, and Z. Kan, "Optimal probabilistic motion planning with potential infeasible LTL constraints," *IEEE Trans. Autom. Control*, early access, Dec. 28, 2021, doi: 10.1109/TAC.2021.3138704.
[9] M. Cai, H. Peng, Z. Li, and Z. Kan, "Learning-based probabilistic LTL motion planning with environment and motion uncertainties," *IEEE Trans. Autom. Control*, vol. 66, no. 5, pp. 2386–2392, May 2021.
[10] M. Guo and D. V. Dimarogonas, "Multi-agent plan reconfiguration under local LTL specifications," *Int. J. Robot. Res.*, vol. 34, no. 2, pp. 218–235, 2015.
[11] S. Andersson and D. V. Dimarogonas, "Human in the loop least violating robot control synthesis under metric interval temporal logic specifications," in *Proc. Eur. Control Conf.*, Limassol, Cyprus, 2018, pp. 453–458.
[12] S. Ahlberg and D. V. Dimarogonas, "Human-in-the-loop control synthesis for multi-agent systems under hard and soft metric interval temporal logic specifications," in *Proc. Int. Conf. Autom. Sci. Eng.*, Vancouver, BC, Canada, 2019, pp. 788–793.
[13] X. Ding, M. Lazar, and C. Belta, "LTL receding horizon control for finite deterministic systems," *Automatica*, vol. 50, no. 2, pp. 399–408, 2014.
[14] Q. Lu and Q.-L. Han, "Mobile robot networks for environmental monitoring: A cooperative receding horizon temporal logic control approach," *IEEE Trans. Cybern.*, vol. 49, no. 2, pp. 698–711, Feb. 2019.
[15] M. Cai, H. Peng, Z. Li, H. Gao, and Z. Kan, "Receding horizon control-based motion planning with partially infeasible LTL constraints," *IEEE Contr. Syst. Lett.*, vol. 5, pp. 1279–1284, 2021.
[16] R. Alur and D. L. Dill, "A theory of timed automata," *Theor. Comput. Sci.*, vol. 126, no. 2, pp. 183–235, 1994.
[17] Z. Li, M. Cai, S. Xiao, and Z. Kan, "Online motion planning with soft metric interval temporal logic in unknown dynamic environment," 2021, *arXiv:2110.09007*.

[1]https://youtu.be/S_jfavmFIMo