Receding Horizon Control-Based Motion Planning With Partially Infeasible LTL Constraints

Mingyu Cai, *Graduate Student Member, IEEE*, Hao Peng, Zhijun Li, *Senior Member, IEEE*, Hongbo Gao, and Zhen Kan, *Member, IEEE*

***Abstract*—This letter considers online optimal motion planning of an autonomous agent subject to linear temporal logic (LTL) constraints. Since user-specified tasks may not be fully realized (i.e., partially infeasible) in a complex environment, this letter considers hard and soft LTL constraints, where hard constraints enforce safety requirements (e.g., avoid obstacles) while soft constraints represent tasks that can be relaxed to not strictly follow user specifications. The motion planning of the agent is to generate trajectories, in decreasing order of priority, to 1) guarantee the satisfaction of safety constraints; 2) mostly fulfill soft constraints (i.e., minimize the violation cost if desired tasks are partially infeasible); 3) locally optimize rewards collection over a finite horizon. To achieve these objectives, receding horizon control is synthesized with an LTL formula to maximize the accumulated utilities over a finite horizon, while ensuring that safety constraints are fully satisfied and soft constraints are mostly satisfied. Simulation and experiment results are provided to demonstrate the effectiveness of the developed motion strategy.**

***Index Terms*—Linear temporal logic, receding horizon control, motion planning.**

I. Introduction

Autonomous agents are desired to perform various user-specific missions. Since the operating environment is often complex, e.g., dynamic and not fully known *a priori*, the user-specified missions can be partially infeasible. Therefore, this letter considers online motion planning of an autonomous agent that can handle complex missions and environments.

Motion planning with LTL specifications has generated substantial interest (see [1]–[3] to name a few). When addressing dynamic environments, receding horizon control has been integrated with LTL specifications in various applications. For instance, the motion planning of a vehicle in an urban-like environment was considered in [4] and LTL-based receding horizon motion planning was developed in [5]. Other representative results include [6]–[10]. However, the aforementioned results rely on a key assumption that the task is feasible in the operating environment. In practice, this assumption can be restrictive. For instance, the agent can be tasked to visit a sequence of areas of interest, where some of them may not be reachable (e.g., surrounded by water that the ground robot cannot traverse). LTL constraints that cannot be fully satisfied are often relaxed to allow the tasks to be fulfilled as much as possible. In [11], a least-violating control strategy for finite LTL was developed to allow potentially infeasible tasks within a partially known workspace. In [12], sampling-based algorithm for minimum violation motion planning was developed. In [13] and [14], partial satisfaction of Co-safe LTL specifications was considered to deal with uncertain environment. These strategies were further extended in [1] for motion planning of service robots. However, only finite horizon motion planning was considered in the works of [1], [11]–[14]. When considering infinite horizon motion planning, the minimal revision problem was considered in [15] and [16] with the goal of making the revised motion planning close to the original LTL. In [17], cooperative control synthesis of multi-agent systems was considered in a partially known environment to maximize the satisfaction of the specified LTL constraints. However, [15] and [16] focus on the single objective of minimal revision to the original LTL and [17] optimizes the static cost (shortest path) via graph-based methods, without considering motion planning with respect to time-varying optimization objectives (e.g., reward collection). It is not yet understood how user-specific missions can be successfully managed to solve optimization problems with time-varying parameters under a dynamic environment, where desired tasks can be partially infeasible.

This letter considers control synthesis of an agent operating in a complex environment with dynamic properties and time-varying areas of interest that can only be observed locally, wherein use-specified tasks might not be fully feasible. To

Manuscript received July 27, 2020; revised September 26, 2020; accepted October 16, 2020. Date of publication October 21, 2020; date of current version November 3, 2020. This work was supported in part by the National Natural Science Foundation of China under Grant 61625303, Grant 61751310, and Grant U1913601, and in part by the Anhui Science and Technology Major Program under Grant 17030901029. Recommended by Senior Editor C. Seatzu. *(Corresponding author: Zhen Kan.)*

Mingyu Cai is with the Department of Mechanical Engineering, University of Iowa, Iowa City, IA 52246 USA.

Hao Peng is with Apex.AI Inc., Palo Alto, CA 94303 USA.

Zhijun Li, Hongbo Gao, and Zhen Kan are with the Department of Automation, University of Science and Technology of China, Hefei 230026, China (e-mail: zkan@ustc.edu.cn).

Digital Object Identifier 10.1109/LCSYS.2020.3032845

address this challenge, we consider hard and soft LTL constraints, where hard constraints enforce safety requirements (e.g., avoid obstacles) while soft constraints represents tasks that can be relaxed to not strictly follow user-specifications if the environment does not permit. A relaxed product automaton is constructed, which allows the agent to not strictly follow the desired LTL constraints. A utility function composed of the violation cost and the accumulated rewards is developed, where the violation cost is designed to quantify the differences between the revised and the desired motion plan. RHC is synthesized with an LTL formula to ensure, in decreasing orders, that 1) hard constraints are fully satisfied, 2) soft constraints are mostly satisfied, and 3) the collection of time-varying rewards is locally optimized in finite horizon. This letter is closely related to [5]. However, we extend the approach in [5] by considering partially infeasible tasks where the energy function is redesigned to take into account the violation cost of the revised path to the desired path. In addition, rigorous analysis is provided, showing the correctness of the produced infinite trajectory and the recursive feasibility of RHC-based motion planning. It's also shown the computational complexity in automaton update is reduced. Simulation and experiment results are provided to demonstrate its effectiveness.

II. Preliminaries

An LTL formula can be translated to a nondeterministic Büchi automaton (NBA).

Definition 1: An NBA is a tuple $\mathcal{B} = (S, S_0, \Delta, \Sigma, \mathcal{F})$, where S is a finite set of states; $S_0 \subseteq S$ is the set of initial states; $\Sigma \subseteq 2^{\Pi}$ is the input alphabet; $\Delta : S \times \Sigma \to 2^S$ is the transition function; and $\mathcal{F} \subseteq S$ is the set of accepting states.

Given a sequence of input $\boldsymbol{\sigma} = \sigma_0\sigma_1\sigma_2\ldots$ over Σ, a run of $\mathcal{B}$ generated by $\boldsymbol{\sigma}$ is an infinite sequence $\boldsymbol{s} = s_0s_1s_2\cdots$ where $s_0 \in S_0$, and $s_{i+1} \in \Delta(s_i, \sigma_i)$ for each $i > 0$. If the input $\boldsymbol{\sigma}$ can generate at least one run $\boldsymbol{s}$ that intersects the accepting states $\mathcal{F}$ infinitely many times, $\mathcal{B}$ is said to accept $\boldsymbol{\sigma}$. Let $\mathcal{B}_\phi$ denote the NBA generated from the LTL formula ϕ [18].

A dynamical system with finite states evolving deterministically under control inputs can be modeled by a weighted finite deterministic transition system (DTS) [19].

Definition 2: A weighted finite (DTS) is a tuple $\mathcal{T} = (Q, q_0, \delta, \Pi, L, \omega)$, where Q is a finite set of states; $q_0 \in Q$ is the initial state; $\delta \subseteq Q \times Q$ is the state transitions; Π is the finite set of atomic propositions; $L : Q \to 2^{\Pi}$ is the labeling function; and $\omega\colon \delta \to \mathbb{R}^+$ is the weight function.

A path of $\mathcal{T}$ is an infinite sequence $\boldsymbol{q} = q_0q_1\ldots$ where $q_i \in Q$ and $(q_i, q_{i+1}) \in \delta$ for $i \geq 0$. A path $\boldsymbol{q}$ over $\mathcal{T}$ generates an output sequence $\boldsymbol{\sigma} = \sigma_0\sigma_1\ldots$ where $\sigma_i = L(q_i)$ for $i \geq 0$. The strategy that generates the path $\boldsymbol{q}$ is referred to as the controller in this letter. Let $R_k(q)$ denote the time-varying reward associated with a state q at time k. The reward represents the event of dynamic interest in the environment.[1]

[1]Local sensing rewards are considered in this letter. Other types of rewards, such as in trajectory optimization [20], information gathering [21], and local tasks [6], are also applicable.

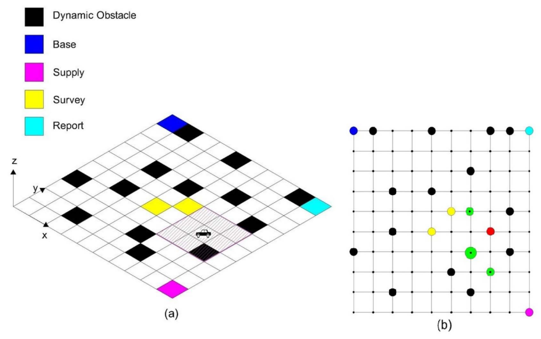

Fig. 1. (a) Example of a partitioned operating environment, where the shaded area around the vehicle indicates its local sensing. (b) The corresponding abstracted grid-like graph of (a), where the size of green dots is proportional to their reward values and the red dot represents the vehicle.

Given a trajectory $\boldsymbol{q}_k = q_0q_1\ldots q_n$, the accumulated reward is $\mathbf{R}_k(\boldsymbol{q}) = \sum_{n}^{i=1} R_k(q_i)$.

Given the defined NBA and DTS, a standard weighted product automaton can be constructed as $\tilde{\mathcal{P}} = \mathcal{T} \times \mathcal{B} = \{P_{\tilde{\mathcal{P}}}, P_{\tilde{\mathcal{P}}0}, L_{\tilde{\mathcal{P}}}, \Delta_{\tilde{\mathcal{P}}}, \mathcal{F}_{\tilde{\mathcal{P}}}, \omega_{\tilde{\mathcal{P}}}\}$, where $P_{\tilde{\mathcal{P}}}$ is the set of states; $P_{\tilde{\mathcal{P}}0}$ is the set of initial states; $L_{\tilde{\mathcal{P}}}$ is a labeling function; $\Delta_{\tilde{\mathcal{P}}}$ is the set of transitions; $\mathcal{F}_{\tilde{\mathcal{P}}}$ is the set of accepting states; $\omega_{\tilde{\mathcal{P}}} : \Delta_{\tilde{\mathcal{P}}} \to \mathbb{R}^+$ is the weight function. A detailed treatment can be found in [19] and [22].

III. Problem Formulation

To illustrate the proposed control synthesis, the following example will be used as a running example throughout this letter.

Example 1: Consider a robot operating in an environment abstracted to a labeled grid-like graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \Pi)$, where the node set $\mathcal{V}$ represents the partitioned areas, the edge set $\mathcal{E}$ indicates possible transitions, and the atomic propositions $\Pi = \{\texttt{Base}, \texttt{Supply}, \texttt{Report}, \texttt{Obstacle}, \texttt{Survey}\}$ indicate the labeled properties of the areas, as shown in Fig. 1. The robot motion in the environment is then represented by the finite DTS $\mathcal{T}$ in Def. 2 evolving over $\mathcal{G}$, where Q represents the node set $\mathcal{V}$, and the possible transitions δ are captured by the edge set $\mathcal{E}$. The environment is assumed to be only partially known to the robot, i.e., the robot may know the static destinations to visit but not the obstacles it may encounter during mission operation. The environment is dynamic in the sense of containing dynamic obstacles and time-varying rewards. The time-varying reward $R_k(q) \in \mathbb{R}^+$ is given at each step. It is further assumed that the robot can only detect obstacles, observe rewards, and sense node labels within a local area around itself. As an example application, the robot is required to complete the given LTL task, while maximizing the collected rewards. In this case, the LTL task is specified as $\phi = \phi_h \wedge \phi_s$, where the hard constraint ϕ_h requires collision avoidance and the soft constraint ϕ_s requires to visit a set of stations respectively. More details about the problem scenario can be found in [22].

Given that the time-varying reward $R_k(q_i)$, $\forall i = 1, \ldots, N$ associated with each state in DTS is unknown *a priori* and can only be locally observed, the motion planning problem in this letter is presented as follows.

Problem 1: Given a deterministic transition system $\mathcal{T}$, and a user-specified LTL formula $\phi = \phi_h \wedge \phi_s$, the control objective is to design an online planning strategy, in decreasing order of priority, that 1) ϕ_h is fully satisfied; 2) ϕ_s is fulfilled as much as possible if the task is not feasible; and 3) rewards collection at each time-step is maximized over a finite horizon during mission operation.

In Problem 1, by saying to fulfill ϕ_s as much as possible, we mean to minimize the violation of ϕ_s, which will be formally defined in Section IV-A.

IV. Relaxed Automaton and Energy Function

Section IV-A discusses how ϕ_s can be relaxed to allow motion revision and how the violation of ϕ_s can be quantified. Section IV-B describes the construction of an energy function that enforces the satisfaction of accepting conditions. Section IV-C presents how local sensing can be used to update the robot's knowledge about the environment to facilitate motion revision.

A. Relaxed LTL Specifications

An LTL specification ϕ is feasible on $\tilde{\mathcal{P}}$ if and only if there exists an accepting run for $\tilde{\mathcal{P}}$. Standard model checking methods on $\tilde{\mathcal{P}}$ are based on the assumption that the given LTL task is feasible, which may not be always true in practice. This section presents the construction of relaxed product automaton to enable online motion planning, if partial tasks are infeasible. Let $\mathcal{B}_h = (S_h, S_{h0}, \Delta_h, \Sigma_h, \mathcal{F}_h)$ and $\mathcal{B}_s = (S_s, S_{s0}, \Delta_s, \Sigma_s, \mathcal{F}_s)$ denote the NBA corresponding to ϕ_h and ϕ_s, respectively. The relaxed product automaton for $\phi = \phi_h \wedge \phi_s$ is constructed as follows.

Definition 3 (Relaxed Product Automaton): Given a weighted DTS $\mathcal{T}$ and the NBA $\mathcal{B}_h$ and $\mathcal{B}_s$, the relaxed product automaton $\mathcal{P} = \mathcal{T} \times \mathcal{B}_h \times \mathcal{B}_s$ is defined as a tuple $\mathcal{P} = \{S_\mathcal{P}, S_{\mathcal{P}0}, L_\mathcal{P}, \Delta_\mathcal{P}, \mathcal{F}_\mathcal{P}, h_\mathcal{P}\colon, \mathrm{v}_\mathcal{P}, \omega_\mathcal{P}\}$, where $S_\mathcal{P} = Q \times S_h \times S_s$ is the set of states, e.g., $s_\mathcal{P} = (q, s_h, s_s)$ and $s'_\mathcal{P} = (q', s'_h, s'_s)$ where $s_\mathcal{P}, s'_\mathcal{P} \in S_\mathcal{P}$; $S_{\mathcal{P}0} = \{q_0\} \times S_{h0} \times S_{s0}$ is the set of initial states; $L_\mathcal{P} : S_\mathcal{P} \to 2^\Pi$ is a labeling function, i.e., $L_\mathcal{P}(s_\mathcal{P}) = L(q)$; $\Delta_\mathcal{P} \subseteq S_\mathcal{P} \times S_\mathcal{P}$ is the set of transitions., defined by $((q, s_h, s_s), (q', s'_h, s'_s)) \in \Delta_\mathcal{P}$ if and only if $(q, q') \in \delta$, $\exists l_h \in 2^{\Pi_h}$, and $\exists l_s \in 2^{\Pi_s}$ such that $s'_h \in \Delta(s_h, l_h)$ and $s'_s \in \Delta(s_s, l_s)$; $\omega_\mathcal{P} : \Delta_\mathcal{P} \to \mathbb{R}^+$ is the weight function; $h_\mathcal{P} : \Delta_\mathcal{P} \to \{0, \infty\}$ is the violation measurement of the hard constraint $\mathcal{B}_h$ such that, for each transition $((q, s_h, s_s), (q, s'_h, s'_s)) \in \Delta_\mathcal{P}$, $h_\mathcal{P}((q, s_h, s_s), (q, s'_h, s'_s)) = 0$ if $s'_h \in \Delta(s_h, L(q))$, and ∞ otherwise; $\mathrm{v}_\mathcal{P} : \Delta_\mathcal{P} \to \mathbb{R}^+$ is the violation function; $\mathcal{F}_\mathcal{P} = Q \times \mathcal{F}_h \times \mathcal{F}_s$ is the set of accepting states.

The major difference between $\tilde{\mathcal{P}}$ and $\mathcal{P}$ is that, for any two states $s_\mathcal{P} = (q, s_h, s_s)$ and $s'_\mathcal{P} = (q', s'_h, s'_s)$, the constraints $s'_h \in \Delta(s_h, L(q))$ and $s'_s \in \Delta(s_s, L(q))$ in $\tilde{\mathcal{P}}$ are relaxed in $\mathcal{P}$ as defined above. Consequently, $\mathcal{P}$ is more connected than $\tilde{\mathcal{P}}$ in terms of possible transitions, which will reduce the computational complexity during automaton update (see Section IV-C). Any transition $((q, s_h, s_s), (q, s'_h, s'_s)) \in \Delta_\mathcal{P}$ that violates the hard constraint will have a infinite violation $h_\mathcal{P}(((q, s_h, s_s), (q, s'_h, s'_s)))$. To identify trajectories that violate the original ϕ_s the least when the environment is infeasible, $\mathrm{v}_\mathcal{P}$ is designed to quantify the violation cost. Suppose that $\Pi = \{\alpha_1, \alpha_2 \ldots \alpha_M\}$ and consider an evaluation function $\mathrm{Eval}\colon 2^\Pi \to \{0, 1\}^M$, where $\mathrm{Eval}(l) = (v_i)^M$ with $v_i = 1$ if $\alpha_i \in l$ and $v_i = 0$ if $\alpha_i \notin l$, where $i = 1, 2, \ldots, M$ and $l \in 2^\Pi$. To quantify the difference between two elements in 2^Π, consider $\rho(l, l') = \|v - v'\|_1 = \sum_{i=1}^M |v_i - v'_i|$, where $v = \mathrm{Eval}(l)$, $v' = \mathrm{Eval}(l')$, $l, l' \in 2^\Pi$, and $\|\cdot\|_1$ is the l_1 norm. The distance from $l \in 2^\Pi$ to a set $\mathcal{X} \subseteq 2^\Pi$ is then defined as $\mathrm{Dist}(l, \mathcal{X}) = \min_{l' \in \mathcal{X}} \rho(l, l')$ if $l \notin \mathcal{X}$, and $\mathrm{Dist}(l, \mathcal{X}) = 0$ otherwise. Now the violation cost of the transition from $s_\mathcal{P} = (q, s_h, s_s)$ to $s'_\mathcal{P} = (q', s'_h, s'_s)$ can be defined as $\mathrm{v}_\mathcal{P}(s_\mathcal{P}, s'_\mathcal{P}) = \mathrm{Dist}(L(q), \mathcal{X}(s_s, s'_s))$, where $\mathcal{X}(s_s, s'_s) = \{l \in 2^\Pi | s'_s \in \Delta(s_s, l).\}$ is the set of input alphabets that enables the transition from s_s to s'_s. Hence, the violation cost $\mathrm{v}_\mathcal{P}(s_\mathcal{P}, s'_\mathcal{P})$ quantifies how much the transition from $s_\mathcal{P}$ to $s'_\mathcal{P}$ in $\mathcal{P}$ violates the constraints imposed by ϕ_s.

Based on the defined $\mathrm{v}_\mathcal{P}(s_\mathcal{P}, s'_\mathcal{P})$, we design the weight function $\omega_\mathcal{P}(s_\mathcal{P}, s'_\mathcal{P}) = h_\mathcal{P}(s_\mathcal{P}, s'_\mathcal{P}) + \omega(q, q') + \beta \cdot \mathrm{v}_\mathcal{P}(s_\mathcal{P}, s'_\mathcal{P})$, where $\beta \in \mathbb{R}^+$ indicates the relative penalty. A larger β tends to bias the selection of trajectories with less violation cost. The weight function $\omega(q, q')$ is defined on the Euclidean distance between q and q' on $\mathcal{T}$, which measures the implementation cost of the transition from q to q'. Since each transition $(s_k^\mathcal{P}, s_{k+1}^\mathcal{P}) \in \Delta_\mathcal{P}$ is associated with a weight in Def. 3, the total weight of a trajectory $\boldsymbol{s}_\mathcal{P}$ is

$$\mathcal{W}(\boldsymbol{s}_\mathcal{P}) = \sum_{k=1}^{n-1} \Big(h_\mathcal{P}\Big(s_k^\mathcal{P}, s_{k+1}^\mathcal{P}\Big) + \omega(q_k, q_{k+1}) + \beta \cdot \mathrm{v}_\mathcal{P}\Big(s_k^\mathcal{P}, s_{k+1}^\mathcal{P}\Big)\Big). \tag{1}$$

Theorem 1: Given an accepting run $\boldsymbol{s}_\mathcal{P} = (q_0, s_{h0}, s_{s0})(q_1, s_{h1}, s_{s1}) \ldots$ of $\mathcal{P}$ for $\phi = \phi_h \wedge \phi_s$, the hard constraints ϕ_h will always be satisfied if $\mathcal{W}(\boldsymbol{s}_\mathcal{P}) \neq \infty$.

Due to space limitation, see [22] for the proof of Theorem 1. Theorem 1 indicates that any accepting run of $\mathcal{P}$ can guarantee that the hard constraints ϕ_h are satisfied by selecting the run with finite total cost in (1). An accepting run $\boldsymbol{s}_\mathcal{P}$ is valid if and only if it satisfies ϕ_h. In (1), the term $\sum_{k=1}^{n-1} \beta \cdot \mathrm{v}_\mathcal{P}(s_\mathcal{P}, s'_\mathcal{P})$ measures the violation of ϕ_s. Hence, a valid accepting run $\boldsymbol{s}_\mathcal{P}$ fulfills ϕ_s as much as possible, if the violation of ϕ_s can be minimized.

B. Energy Function

Analogous to Lyapunov theory, where the convergence of the system states to equilibrium points is indicated by a decreasing Lyapunov function, a Lyapunov-like energy function is designed in this section to enforce the acceptance condition of an automaton by requiring the distance to the accepting states to decrease as the system evolves. Based on (1), $d(s_i^\mathcal{P}, s_j^\mathcal{P}) = \min_{\boldsymbol{s}_\mathcal{P} \in \mathcal{D}(s_i^\mathcal{P}, s_j^\mathcal{P})} \mathcal{W}(\boldsymbol{s}_\mathcal{P})$ is the shortest path from $s_i^\mathcal{P}$ to $s_j^\mathcal{P}$, where $\mathcal{D}(s_i^\mathcal{P}, s_j^\mathcal{P})$ is the set of all trajectories.

Given $\mathcal{P}_{(S_\mathcal{P}, \Delta_\mathcal{P})}$, the graph induced from $\mathcal{P}_{(S_\mathcal{P}, \Delta_\mathcal{P})}$ by neglecting the weight of each transition is denoted by $\mathcal{G}_{(S_\mathcal{P}, \Delta_\mathcal{P})}$. The largest self-reachable subset of the accepting set $\mathcal{F}_\mathcal{P}$ is defined as $\mathcal{F}^*$ such that there exists a path in $\mathcal{P}$ connecting any two pairs of states in $\mathcal{F}^*$. The set $\mathcal{F}^*$

can be constructed by following similar procedures in [5] by neglecting the cost of $h_{\mathcal{P}}$.

Definition 4 (Energy Function): For $s_{\mathcal{P}} \in S_{\mathcal{P}}$, the energy function $J(s_{\mathcal{P}})$ is designed as

$$J(s_{\mathcal{P}}) = \begin{cases} \min\limits_{s'_{\mathcal{P}} \in \mathcal{F}^*} d\big(s_{\mathcal{P}}, s'_{\mathcal{P}}\big) & \text{if } s_{\mathcal{P}} \notin \mathcal{F}^*, \\ 0 & \text{if } s_{\mathcal{P}} \in \mathcal{F}^*. \end{cases} \tag{2}$$

The design of $J(s_{\mathcal{P}})$ in (4) is inspired by [5]. Different from [5], we adapt it to the relaxed product automaton by taking into account the distance from the states to the largest self-reachable subset $\mathcal{F}^*$ of the relaxed product automaton. Since $\omega_{\mathcal{P}}$ is positive by definition, $d(s_{\mathcal{P}}, s'_{\mathcal{P}}) > 0$ for all $s_{\mathcal{P}}, s'_{\mathcal{P}} \in S_{\mathcal{P}}$, which implies that $J(s_{\mathcal{P}}) \geq 0$. Particularly, $J(s_{\mathcal{P}}) = 0$ if $s_{\mathcal{P}} \in \mathcal{F}^*$. If a state in $\mathcal{F}^*$ is reachable from $s_{\mathcal{P}}$, then $J(s_{\mathcal{P}}) \neq \infty$, otherwise $J(s_{\mathcal{P}}) = \infty$. Hence, $J(s_{\mathcal{P}})$ indicates the minimum distance from $s_{\mathcal{P}}$ to $\mathcal{F}^*$.

Theorem 2: For the energy function designed in (2), if a trajectory $\boldsymbol{s}_{\mathcal{P}} = s_1^{\mathcal{P}} s_2^{\mathcal{P}} \ldots s_n^{\mathcal{P}}$ is accepting, there is no state $s_i^{\mathcal{P}}$, $\forall i = 1, \ldots, n$, with $J(s_i^{\mathcal{P}}) = \infty$, and all accepting states in $\boldsymbol{s}_{\mathcal{P}}$ are in the set $\mathcal{F}^*$ with energy 0. In addition, for any state $s_{\mathcal{P}} \in S_{\mathcal{P}}$ with $s_{\mathcal{P}} \notin \mathcal{F}^*$ and $J(s_{\mathcal{P}}) \neq \infty$, there exists at least one state $s'_{\mathcal{P}}$ with $(s_{\mathcal{P}}, s'_{\mathcal{P}}) \in \Delta_{\mathcal{P}}$ such that $J(s'_{\mathcal{P}}) < J(s_{\mathcal{P}})$.

See [22] for the proof. Theorem 2 indicates that, as long as the energy function keeps decreasing, the generated trajectory will eventually satisfy the accepting conditions in Def. 3. As a result, the designed energy function can be used to enforce the accepting condition of $\mathcal{P}$.

C. Automaton Update

Since the environment is only partially known, this section describes how the real-time information sensed by the robot during the runtime can be used to update the system model to facilitate motion planning. The robot starts with an initial, possibly imprecise, knowledge about the environment. A potential cause of infeasible task specifications is the imprecise state labels. Due to limited local sensing capability, let Q_N denote the set of sensible neighboring states and let $[\![s_{\mathcal{P}}]\!] = \{s_{\mathcal{P}} = (q, s_h, s_s) | q \in Q_N\}$ denote a class of $s_{\mathcal{P}}$ sharing the same neighboring states. Specifically, let $\text{Info}(s_{\mathcal{P}}) = \{L_{\mathcal{P}}(s'_{\mathcal{P}}) | s'_{\mathcal{P}} \in \text{Sense}(s_{\mathcal{P}})\}$ denote the newly observed labels of $s'_{\mathcal{P}}$ that are different from the current knowledge, where $\text{Sense}(s_{\mathcal{P}})$ represents a local set of states that can be sensed by the robot at $s_{\mathcal{P}}$. If the sensed labels $L_{\mathcal{P}}(s'_{\mathcal{P}})$ are consistent with the current knowledge of $s'_{\mathcal{P}}$, $\text{Info}(s_{\mathcal{P}}) = \emptyset$. Otherwise, the properties of $s'_{\mathcal{P}}$ need to be updated.

Let $\boldsymbol{J}([\![\boldsymbol{s}_{\mathcal{P}}]\!]) \in \mathbb{R}^{|[\![s_{\mathcal{P}}]\!]|}$ denote the stacked J for all $s_{\mathcal{P}} \in [\![s_{\mathcal{P}}]\!]$. For $i, j = 1, \ldots |S_{\mathcal{P}}|$, let $\mathbf{H}_{\mathcal{P}} \in \mathbb{R}^{|S_{\mathcal{P}}| \times |S_{\mathcal{P}}|}$ denote a matrix where the (i,j)th entry of $\mathbf{H}_{\mathcal{P}}$ represents $\mathrm{h}_{\mathcal{P}}(s_i^{\mathcal{P}}, s_j^{\mathcal{P}})$ and let $\mathbf{V}_{\mathcal{P}} \in \mathbb{R}^{|S_{\mathcal{P}}| \times |S_{\mathcal{P}}|}$ denote a matrix where the (i,j)th entry of $\mathbf{V}_{\mathcal{P}}$ represents the violation cost $\mathrm{v}_{\mathcal{P}}(s_i^{\mathcal{P}}, s_j^{\mathcal{P}})$. The terms $\boldsymbol{J}$, $\mathbf{H}_{\mathcal{P}}$ and $\mathbf{V}_{\mathcal{P}}$ are initialized from the initial knowledge of the environment. Algorithm 1 outlines how $\mathbf{H}_{\mathcal{P}}$, $\mathbf{V}_{\mathcal{P}}$, and $\boldsymbol{J}$ are updated based on the locally sensed information to facilitate motion planning in line 2-10. At each step, if $\text{Info}(s_{\mathcal{P}}) \neq \emptyset$, the energy function $\boldsymbol{J}$ for each states of $[\![s_{\mathcal{P}}]\!]$ is updated by Algorithm 1.

Algorithm 1 Automaton Update

```
 1: procedure INPUT: ( the current state $s_{\mathcal{P}} = (q, s_h, s_s)$, the current $J([\![s_{\mathcal{P}}]\!])$, $\mathcal{F}^*$, and Info$(s_{\mathcal{P}})$ )
        Output: the updated $\boldsymbol{J}'$
 2:     if Info$(s_{\mathcal{P}}) \neq \emptyset$ then
 3:         for all $s'_{\mathcal{P}} = (q', s'_h, s'_s) \in$ Sense$(s_{\mathcal{P}})$ such that $L_{\mathcal{P}}(s'_{\mathcal{P}}) \in$ Info$(s_{\mathcal{P}})$ do
 4:             for all $\hat{s}'_{\mathcal{P}}$ such that $(s'_{\mathcal{P}}, \hat{s}'_{\mathcal{P}}) \in \Delta_{\mathcal{P}}$ do
 5:                 Update the labels of $L_{\mathcal{P}}(s'_{\mathcal{P}})$ according to $L(q')$;
 6:                 Update hard constraint measurement $h_{\mathcal{P}}(s_{\mathcal{P}}, s'_{\mathcal{P}})$ to obtain $\mathbf{H}'_{\mathcal{P}}$;
 7:                 Update the violation cost $\mathrm{v}_{\mathcal{P}}(s_{\mathcal{P}}, s'_{\mathcal{P}})$ to obtain $\mathbf{V}'_{\mathcal{P}}$;
 8:             end for
 9:         end for
10:         Update $J([\![s_{\mathcal{P}}]\!])$ based on (2);
11:     end if
12: end procedure
```

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

Proof: By neglecting the cost of transitions in Section IV-B, the relaxed product automaton $\mathcal{P}_{(S_{\mathcal{P}}, \Delta_{\mathcal{P}})}$ can be treated as a directed graph $\mathcal{G}_{(S_{\mathcal{P}}, \Delta_{\mathcal{P}})}$. By Def.3, Alg. 1 only updates the cost of each transition. As a result, the topological structure of $\mathcal{G}_{(S_{\mathcal{P}}, \Delta_{\mathcal{P}})}$ and its corresponding $\mathcal{F}^*$ remain the same.

The construction of $\mathcal{F}^*$ in [5] involves the computation of $d(s_{\mathcal{P}}, s'_{\mathcal{P}})$ for all $s'_{\mathcal{P}} \in \mathcal{F}_{\mathcal{P}}$ and the check of terminal conditions, leading to the computational complexity of $O(|\mathcal{F}_{\mathcal{P}}|^3 + |S_{\mathcal{P}}|^2 \times |\mathcal{F}_{\mathcal{P}}|^2 + |\mathcal{F}_{\mathcal{P}}|)$. In contrast, Lemma 1 indicates that $\mathcal{F}^*$ in this letter only needs to be updated whenever newly sensed information different from its knowledge is obtained, which reduces the complexity. In the worst case, the complexity is $|Q_N|$. Instead of computing the whole relaxed product automaton, Algorithm 1 only updates partial information of the systems. ■

V. Control Synthesis of LTL Motion Planning

A. Receding Horizon Control

The general idea of RHC is to generate a predicted optimal trajectory at each time step by solving an online optimization problem to maximize a utility function over a finite horizon N. With only the first predicted step applied, the optimization problem is repeatedly solved to predict optimal trajectories. Specifically, based on the current state $s_k^{\mathcal{P}}$, let $\bar{\boldsymbol{s}}_k^{\mathcal{P}} = s_{1|k}^{\mathcal{P}} s_{2|k}^{\mathcal{P}} \ldots s_{N|k}^{\mathcal{P}}$ denote a predicted trajectory of horizon N at time k from $s_k^{\mathcal{P}}$, where the ith predicted state $s_{i|k}^{\mathcal{P}} \in S_{\mathcal{P}}$ satisfies $(s_{i|k}^{\mathcal{P}}, s_{i+1|k}^{\mathcal{P}}) \in \Delta_{\mathcal{P}}$ for all $i = 1, \ldots, N-1$, and $(s_k^{\mathcal{P}}, s_{1|k}^{\mathcal{P}}) \in \Delta_{\mathcal{P}}$. Let $\text{Path}(s_k^{\mathcal{P}}, N)$ be the set of trajectories of horizon N generated from $s_k^{\mathcal{P}}$. Note that a predicted trajectory $\bar{\boldsymbol{s}}_k^{\mathcal{P}} \in \text{Path}(s_k^{\mathcal{P}}, N)$ can uniquely project to a path $\gamma_{\mathcal{T}}(\bar{\boldsymbol{s}}_k^{\mathcal{P}}) = \boldsymbol{q} = q_1 \cdots q_N$ on $\mathcal{T}$, where $\gamma_{\mathcal{T}}(s_{i|k}^{\mathcal{P}}) = q_i$, $\forall i = 1, \ldots, N$. The finite horizon N is selected based on the robot's local sensing such that the labels $L_{\mathcal{P}}(q_i)$ and the reward $R_k(q_i)$, $\forall i = 1, \ldots, N$, are all observable by the robot at time k. The accumulated reward along the predicted trajectory $\bar{\boldsymbol{s}}_k^{\mathcal{P}}$ is $\mathbf{R}(\gamma_{\mathcal{T}}(\bar{\boldsymbol{s}}_k^{\mathcal{P}})) = \sum_{i=1}^{N} R_k(\gamma_{\mathcal{T}}(s_{i|k}^{\mathcal{P}}))$.

Once a predicted step k of RHC is implemented, the hard and soft violation cost induced from the current state $s_k^{\mathcal{P}}$ to the next predicted step $s_{1|k}^{\mathcal{P}}$ are considered, i.e., $h_{\mathcal{P}}(s_k^{\mathcal{P}}, s_{1|k}^{\mathcal{P}})$ and $\mathbf{V}(s_k^{\mathcal{P}}) = \beta \cdot \mathrm{v}_{\mathcal{P}}(s_k^{\mathcal{P}}, s_{1|k}^{\mathcal{P}})$. The utility function of RHC is then designed as

$$\mathbf{U}\left(\bar{\mathbf{s}}_k^{\mathcal{P}}\right) = -h_{\mathcal{P}}\left(s_k^{\mathcal{P}}, s_{1|k}^{\mathcal{P}}\right) + \mathbf{R}\left(\gamma_{\mathcal{T}}\left(\bar{\mathbf{s}}_k^{\mathcal{P}}\right)\right) \min\left\{e^{-\kappa \mathbf{V}\left(s_k^{\mathcal{P}}\right)}, 1\right\} \tag{3}$$

where $\kappa \in \mathbb{R}^+$ is a tuning parameter indicating how aggressively a predicted path is penalized by violating the soft task constraints and the non-zero violation $\mathbf{V}(s_k^{\mathcal{P}})$ in (3) would enforce the decrease of $\mathbf{U}(\bar{\mathbf{s}}_k^{\mathcal{P}})$. If $h_{\mathcal{P}}(s_k^{\mathcal{P}}, s_{1|k}^{\mathcal{P}}) = \infty$, it indicates that $\mathbf{U}(\bar{\mathbf{s}}_k^{\mathcal{P}})$ is negative infinite. By applying a larger κ optimizing $\mathbf{U}(\bar{\mathbf{s}}_k^{\mathcal{P}})$ tends to bias the selection of paths towards the objectives, in the decreasing order, of 1) hard task ϕ_h satisfaction, 2) fulfilling soft task ϕ_h as much as possible, and 3) time-varying rewards locally optimization.

Since maximizing $\mathbf{U}(\bar{\mathbf{s}}_k^{\mathcal{P}})$ alone cannot guarantee the satisfaction of the acceptance condition of $\mathcal{P}$, energy function the energy function based constraints are incorporated. We first select initial states from $S_{\mathcal{P}0}$ that can reach the set $\mathcal{F}^*$. The RHC executing on $S_{\mathcal{P}0}$ is designed as

$$\bar{\mathbf{s}}_{0,\text{opt}}^{\mathcal{P}} = \underset{\bar{s}_0^{\mathcal{P}} \in \text{Path}\left(s_0^{\mathcal{P}}, N\right)}{\arg\max} \quad \mathbf{U}\left(\bar{s}_0^{\mathcal{P}}\right)$$
$$\text{subject to :} \quad J\left(s_0^{\mathcal{P}}\right) < \infty. \tag{4}$$

The constraint $J(s_0^{\mathcal{P}}) < \infty$ in (4) is critical, since a bounded energy $J(s_0^{\mathcal{P}})$ guarantees the existence of a satisfying trajectory from $s_0^{\mathcal{P}}$ over $\mathcal{P}$. According to the working principle of RHC, the first element of the optimal trajectory $\bar{\mathbf{s}}^{\mathcal{P}*}$ can be determined as $s_0^{\mathcal{P}*} = s_{1|0,\text{opt}}^{\mathcal{P}}$, where $s_{1|0,\text{opt}}^{\mathcal{P}}$ is the first element of $\bar{\mathbf{s}}_{0,\text{opt}}^{\mathcal{P}}$ obtained from (4).

After determining the initial state $s_0^{\mathcal{P}*}$, RHC will be employed repeatedly to determine the optimal states $s_k^{\mathcal{P}*}$ for $k = 1, 2, \ldots$. At each time instant k, a predicted optimal trajectory $\bar{\mathbf{s}}_{k,\text{opt}}^{\mathcal{P}} = s_{1|k,\text{opt}}^{\mathcal{P}} s_{2|k,\text{opt}}^{\mathcal{P}} \ldots s_{N|k,\text{opt}}^{\mathcal{P}}$ will be constructed based on $s_{k-1}^{\mathcal{P}*}$ and $\bar{\mathbf{s}}_{k-1,\text{opt}}^{\mathcal{P}}$ obtained at the previous time $k-1$. Note that only $s_{1|k,\text{opt}}^{\mathcal{P}}$ will be applied at time k, i.e., $s_k^{\mathcal{P}*} = s_{1|k,\text{opt}}^{\mathcal{P}}$, which will then be used with $\bar{\mathbf{s}}_{k,\text{opt}}^{\mathcal{P}}$ to generate $\bar{\mathbf{s}}_{k+1,\text{opt}}^{\mathcal{P}}$.

Theorem 3: For each time $k = 1, 2 \ldots$, provided $s_{k-1}^{\mathcal{P}*}$ and $\bar{\mathbf{s}}_{k-1,\text{opt}}^{\mathcal{P}}$ from previous time step, consider a receding horizon control (RHC)

$$\bar{\mathbf{s}}_{k,\text{opt}}^{\mathcal{P}} = \underset{\bar{s}_k^{\mathcal{P}} \in \text{Path}\left(s_{k-1}^{\mathcal{P}*}, N\right)}{\arg\max} \quad \mathbf{U}\left(\bar{s}_k^{\mathcal{P}}\right) \tag{5}$$

subject to the following constraints:

1) $J(s_{N|k}^{\mathcal{P}}) < J(s_{N|k-1,\text{opt}}^{\mathcal{P}})$ if $J(s_{k-1}^{\mathcal{P}*}) > 0$ and $J(s_{i|k-1,\text{opt}}^{\mathcal{P}}) \neq 0$ for all $i = 1, \ldots, N$;
2) $J(s_{i_0(s_{k-1,\text{opt}}^{\mathcal{P}})-1|k}^{\mathcal{P}}) = 0$ if $J(s_{k-1}^{\mathcal{P}*}) > 0$ and $J(s_{i|k-1,\text{opt}}^{\mathcal{P}}) = 0$ for some $i = 1, \ldots, N$;
3) $J(s_{N|k}^{\mathcal{P}}) < \infty$ if $J(s_{k-1}^{\mathcal{P}*}) = 0$.

Applying $s_k^{\mathcal{P}*} = s_{1|k,\text{opt}}^{\mathcal{P}}$ at each time k, the optimal trajectory $\bar{\mathbf{s}}^{\mathcal{P}*} = s_0^{\mathcal{P}*} s_1^{\mathcal{P}*} \ldots$ is guaranteed to satisfy the acceptance condition of $\mathcal{P}$.

See more details in [22] for the proof. Analogous to the analysis in [5], the energy function based constraints (5) in Theorem 3 ensure that $\bar{\mathbf{s}}^{\mathcal{P}*} = s_0^{\mathcal{P}*} s_1^{\mathcal{P}*} \ldots$ intersects the accepting states $\mathcal{F}_{\mathcal{P}}$ infinitely, resulting in the satisfaction of the acceptance condition of $\mathcal{P}$. If the RHC yields an negative infinite utility, the hard constraint is violated and the robot fails to accomplish the task. In this letter, we assume ϕ_h is always feasible.

Algorithm 2 Control Synthesis of LTL Online Motion Planning

```
1:  procedure INPUT:(The DTS $\mathcal{T} = \{Q, q_0, \delta, \Pi, L, \omega\}$ and the NBA $\mathcal{B}_h, \mathcal{B}_s$
    corresponding to the user-specified LTL formula $\phi = \phi_h \wedge \phi_s$ )
        Output:  The trajectory $\bar{s}^{\mathcal{P}*} = s_0^{\mathcal{P}*} s_1^{\mathcal{P}*} \ldots$
    Off-line Execution:
2:      Construct the relaxed product automaton $\mathcal{P} = \mathcal{T} \times \mathcal{B}_h \times \mathcal{B}_s$
3:      Construct $\mathcal{F}^*$, and initialize $\mathbf{H}_{\mathcal{P}}$, $\mathbf{V}_{\mathcal{P}}$ and $\boldsymbol{J}$
    online Execution:
4:      if $\exists s_0^{\mathcal{P}} \in S_{\mathcal{P}0}$ , $J\left(s_0^{\mathcal{P}}\right) < \infty$ then
5:          Solve (4) for $\bar{s}_{0,\text{opt}}^{\mathcal{P}}$
6:          $s_0^{\mathcal{P}*} = s_{1|0,\text{opt}}^{\mathcal{P}}$ and $k \leftarrow 1$
7:          while $k > 0$ do
8:              Apply automaton update at $s_{k-1}^{\mathcal{P}*}$ in Algorithm 1 based on local sensing
9:              Locally observe rewards $R_k\left(\gamma_{\mathcal{T}}\left(s_{k-1}^{\mathcal{P}*}\right)\right)$
10:             Solve (5) for $\bar{s}_{k,\text{opt}}^{\mathcal{P}}$
11:             Implement corresponding transitions on $\mathcal{P}$ and $\mathcal{T}$
12:             $s_k^{\mathcal{P}*} = s_{1|k,\text{opt}}^{\mathcal{P}}$ and $k \leftarrow k+1$
13:         end while
14:     else
15:         There does not exist an accepting run from initial states;
16:     end if
17: end procedure
```

B. Control Synthesis

The control synthesis of the LTL online motion planning strategy is outlined in Algorithm 2. In Lines 1-3, an off-line computation is first performed over $\mathcal{P}$ to obtain an initial $\boldsymbol{J}$ and an initial violation cost $\mathbf{V}_{\mathcal{P}}$. At time $k = 0$, the receding horizon control (4) is applied to determine $s_0^{\mathcal{P}*}$ in Lines 4-7. Due to the dynamic and uncertain nature of the environment, Algorithm 1 is applied at each time $k > 0$ to update $\boldsymbol{J}([\![s_{\mathcal{P}}]\!])$ and $\mathbf{V}_{\mathcal{P}}$ based on local sensing in Lines 9-10. The RHC (5) is then employed based on the previously determined $s_{k-1}^{\mathcal{P}*}$ to generate $\bar{\mathbf{s}}_{k,\text{opt}}^{\mathcal{P}}$, where the next state is determined as $s_k^{\mathcal{P}*} = s_{1|k,\text{opt}}^{\mathcal{P}}$ in Lines 11-12. The transition from $s_{k-1}^{\mathcal{P}*}$ to $s_k^{\mathcal{P}*}$ is then immediately applied on $\mathcal{P}$, which corresponds to the movement of the robot at time k from $\gamma_{\mathcal{T}}(s_{k-1}^{\mathcal{P}*})$ to $\gamma_{\mathcal{T}}(s_k^{\mathcal{P}*})$ on $\mathcal{T}$ in Line 13. Repeating the process can generate a trajectory $\bar{\mathbf{s}}^{\mathcal{P}*} = s_0^{\mathcal{P}*} s_1^{\mathcal{P}*} \ldots$ that optimizes the utilities while satisfying the acceptance condition of $\mathcal{P}$. If $J(s_0^{\mathcal{P}}) = \infty$, there exists no trajectory that satisfies ϕ_h in Line 17.

Theorem 4 (Correctness of Algorithm 2): Given a weighted DTS $\mathcal{T} = \{Q, q_0, \delta, \Pi, L, \omega\}$ and $\mathcal{B}_h$ and $\mathcal{B}_s$ corresponding to ϕ_h and ϕ_s, respectively, if there exists an initial state $s_0^{\mathcal{P}} \in S_{\mathcal{P}0}$ with $J(s_0^{\mathcal{P}}) < \infty$, the trajectory generated by Algorithm 2 is guaranteed to satisfy the acceptance condition of $\mathcal{P}$.

If $J(s_k^{\mathcal{P}}) < \infty$, it indicates there exists a run satisfying ϕ_h and the violation cost $-h_{\mathcal{P}}(s_k^{\mathcal{P}}, s_{1|k}^{\mathcal{P}})$ in RHC ensures the satisfaction of ϕ_h since only the first predicted step is applied.

Corollary 1: Given a weighted DTS $\mathcal{T} = \{Q, q_0, \delta, \Pi, L, \omega\}$, $\mathcal{B}_h$ and $\mathcal{B}_s$, if φ_s is feasible, the solution of Algorithm 2 fully satisfies the task $\varphi = \varphi_h \wedge \varphi_s$ exactly with κ in (3) selected sufficiently large.

See [22] for the proof of Theorem 4 and Corollary 1.

Since the off-line execution involves the computation of $\mathcal{P}$, $\mathcal{F}^*$, the initial $\boldsymbol{J}$, and the initial $\mathbf{V}_{\mathcal{P}}$, its complexity is

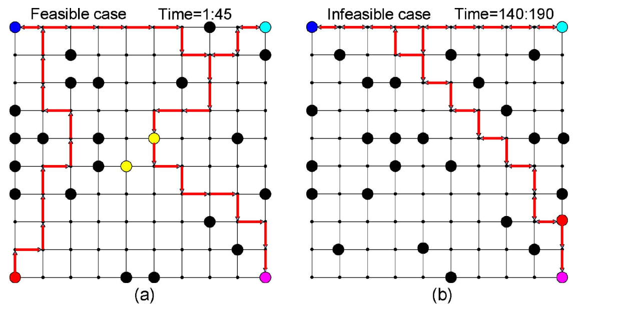

Fig. 2. The robot trajectories in feasible and infeasible cases of ϕ_S, respectively. In (a), the environment is fully feasible from t = 1s to t = 45s, and the robot successfully completes the desired task. In (b), the environment is infeasible from t = 140s to t = 190s, where yellow circles do not exist. The robot revises its motion to only sequentially visit `Base`, `Supply`, and `Report` stations. In both (a) and (b), the planned path maximizes the reward collection in a receding horizon manner.

$O(|\mathcal{F}_\mathcal{P}|^3 + |S_\mathcal{P}|^2 \times |\mathcal{F}_\mathcal{P}|^2 + |\mathcal{F}_\mathcal{P}|)$. For online execution, since $\mathcal{F}^*$ remains the same from Lemma 1, the worst case of Algorithm 1 requires $|[\![s_\mathcal{P}]\!]|$ runs of Dijkstra's algorithm. Therefore, the complexity of Algorithm 1 is at most $O(|N_1| \times |S_\mathcal{P}| + |S_\mathcal{P}|)$. In Algorithm 2, the complexity of recursive computation at each time step is bounded by $|\Delta_\delta|^N$, where N is the selected horizon in RHC. Overall, the maximum complexity of the online portion of RHC is $O(|N_1| \times |S| + |S_\mathcal{P}| + |\Delta_\delta|^N)$.

VI. Case Studies

Due to space limitation, detailed simulation and experiment results can be found in [22]. The considered LTL task is $\phi = \phi_h \wedge \phi_s$, where $\phi_h = \square\neg\texttt{Obstacle}$ is translated to a Büchi Automaton $\mathcal{B}_h$ via LTL2BA [23] with $|S_h| = 1$. In simulation, the soft task of the robot is designed as $\phi_s = \square\Diamond\texttt{Base} \wedge \square(\texttt{Base} \rightarrow \bigcirc(\neg\texttt{Base} \cup \texttt{Survey})) \wedge \square(\texttt{Survey} \rightarrow \bigcirc(\neg\texttt{Survey} \cup \texttt{Report})) \wedge \square(\texttt{Report} \rightarrow \bigcirc(\neg\texttt{Report} \cup \texttt{Supply}))$. In English, ϕ means the robot needs to always avoid `Obstacle` while repeatedly and sequentially visiting `Base`, `Survey`, `Report`, and `Supply`. Figs. 2 (a) and (b) show the trajectories of the robot in the feasible and infeasible ϕ_s, respectively. The scalability of the proposed motion planning strategy is analyzed [22]. The simulation video is provided.[2] Besides simulation, experiments were also performed on a mobile robot, Khepera IV, to verify the developed control strategy. The experiment video is provided.[3]

VII. Conclusion

An RHC-based online motion planning strategy is developed to maximize reward collection while considering hard and soft LTL constraints. Since different costs and rewards are mixed in the utility function, resulting in challenges to tune parameters, additional work will also consider reformulating the utility function into a multi-objective optimization problem and leverage advanced learning methods to address it.

[2]https://www.youtube.com/watch?v=RyRnKXDDH5U&t=30s
[3]https://youtu.be/16j6TmVUrTk

References

[1] B. Lacerda, F. Faruq, D. Parker, and N. Hawes, "Probabilistic planning with formal performance guarantees for mobile service robots," *Int. J. Robot. Res.*, vol. 38, no. 9, pp. 1098–1123, 2019.

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

[3] M. Cai, H. Peng, Z. Li, and Z. Kan, "Learning-based probabilistic LTL motion planning with environment and motion uncertainties," *IEEE Trans. Autom. Control*, early access, Jul. 3, 2020, doi: 10.1109/TAC.2020.3006967.

[4] T. Wongpiromsarn, U. Topcu, and R. M. Murray, "Receding horizon temporal logic planning," *IEEE Trans. Autom. Control*, vol. 57, no. 11, pp. 2817–2830, Nov. 2012.

[5] X. Ding, M. Lazar, and C. Belta, "LTL receding horizon control for finite deterministic systems," *Automatica*, vol. 50, no. 2, pp. 399–408, 2014.

[6] A. Ulusoy and C. Belta, "Receding horizon temporal logic control in dynamic environments," *Int. J. Robot. Res.*, vol. 33, no. 12, pp. 1593–1607, 2014.

[7] V. Raman, A. Donzé, M. Maasoumy, R. M. Murray, A. Sangiovanni-Vincentelli, and S. A. Seshia, "Model predictive control with signal temporal logic specifications," in *Proc. IEEE Conf. Decis. Control*, Los Angeles, CA, USA, 2014, pp. 81–87.

[8] J. Tumova and D. V. Dimarogonas, "A receding horizon approach to multi-agent planning from local LTL specifications," in *Proc. IEEE Amer. Control Conf.*, Portland, OR, USA, 2014, pp. 1775–1780.

[9] S. S. Farahani, R. Majumdar, V. S. Prabhu, and S. E. Z. Soudjani, "Shrinking horizon model predictive control with chance-constrained signal temporal logic specifications," in *Proc. IEEE Amer. Control Conf.*, Seattle, WA, USA, 2017, pp. 1740–1746.

[10] 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.

[11] J. Tumova, G. C. Hall, S. Karaman, E. Frazzoli, and D. Rus, "Least-violating control strategy synthesis with safety rules," in *Proc. 16th Int. Conf. Hybrid Syst. Comput. Control*, 2013, pp. 1–10.

[12] L. I. R. Castro, P. Chaudhari, J. Tmová, S. Karaman, E. Frazzoli, and D. Rus, "Incremental sampling-based algorithm for minimum-violation motion planning," in *Proc. 52nd IEEE Conf. Decis. Control*, Florence, Italy, 2013, pp. 3217–3224.

[13] M. Lahijanian, S. Almagor, D. Fried, L. E. Kavraki, and M. Y. Vardi, "This time the robot settles for a cost: A quantitative approach to temporal logic planning with partial satisfaction," in *Proc. 29th AAAI Conf. Artif. Intell.*, 2015, pp. 3664–3671.

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

[15] K. Kim and G. E. Fainekos, "Approximate solutions for the minimal revision problem of specification automata," in *Proc. IEEE Int. Conf. Intell. Robot. Syst.*, Vilamoura, Portugal, 2012, pp. 265–271.

[16] K. Kim, G. E. Fainekos, and S. Sankaranarayanan, "On the revision problem of specification automata," in *Proc. IEEE Int. Conf. Robot. Autom.*, Saint Paul, MN, USA, 2012, pp. 5171–5176.

[17] 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.

[18] P. Gastin and D. Oddoux, "Fast LTL to Büchi automata translation," in *Proc. Int. Conf. Comput. Aided Verif.*, 2001, pp. 53–65.

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

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

[21] K. Leahy, A. Jones, M. Schwager, and C. Belta, "Distributed information gathering policies under temporal logic constraints," in *Proc. 54th IEEE Conf. Decis. Control.*, Osaka, Japan, 2015, pp. 6803–6808.

[22] M. Cai, H. Peng, Z. Li, H. Gao, and Z. Kan, "Receding horizon control based online motion planning with partially infeasible LTL specifications," 2020. [Online]. Available: arXiv:2007.12123.

[23] T. Babiak, M. Křetínský, V. Řehák, and J. Strejček, "LTL to Büchi automata translation: Fast and more deterministic," in *Proc. Int. Conf. Tools Algorithm Construct. Anal. Syst.*, 2012, pp. 95–109.