Attraction-Based Receding Horizon Path Planning with Temporal Logic Constraints

Mária Svoreňová, Jana Tůmová, Jiří Barnat and Ivana Černá

Abstract—Our goal in this paper is to plan the motion of a robot in a partitioned environment with dynamically changing, locally sensed rewards. We assume that arbitrary assumptions on the reward dynamics can be given. The robot aims to accomplish a high-level temporal logic surveillance mission and to locally optimize the collection of the rewards in the visited regions. These two objectives often conflict and only a compromise between them can be reached. We address this issue by taking into consideration a user-defined preference function that captures the trade-off between the importance of collecting high rewards and the importance of making progress towards a surveyed region. Our solution leverages ideas from the automata-based approach to model checking. We demonstrate the utilization and benefits of the suggested framework in an illustrative example.

I. INTRODUCTION

In this paper, we consider the problem of robot path planning (see, e.g., [1] for an overview) with more complex missions than "Go from $A$ to $B$ while avoiding obstacles."

Recently, different versions of temporal logics, such as Linear Temporal Logic (LTL), Computation Tree Logic (CTL), or $\mu$-calculus have been successfully employed to specify such robotic missions [2], [3], [4], [5], [6], [7], [8]. We have chosen LTL [9], [10] as the specification means for its resemblance to natural language and its ability to express interesting robot behavior, such as "Repeatedly survey regions $A$ and $B$ while avoiding dangerous regions. Make sure, that $A$ is always visited in between two successive visits to $B$ and vice versa."

We assume that the robot motion in the environment is modeled as a transition system, which is obtained by partitioning the environment into regions (for instance using well-known triangulations and rectangular partitions). Each region is modeled as a state of the transition system and the robot’s capability to move between the regions as transitions between the corresponding states. Our transition system is deterministic, i.e., a control input for the robot is the next region (state) to be visited. Moreover, the transition system is weighted, i.e., each transition is equipped with the time duration this transition takes.

The robot’s task is to collect rewards that dynamically appear, disappear and change their values in the environment regions and that can be sensed only within a certain vicinity of the robot’s current state. A traditional approach to this kind of problem, i.e., an optimization problem defined on a dynamically changing plant, is model predictive control (MPC) [11]. The method is based on iterative re-planning and optimization of a cost function over a finite horizon and hence, it is also called receding horizon control.

In this work, we focus on interconnecting the receding horizon control with the synthesis of a path that is provably correct with respect to a given temporal logic formula. This idea appeared in [12], [13], where the receding horizon approach was employed to fight the high computational complexity of reactive motion planning with a specification in GR(1) fragment of LTL. However, the authors did not consider any rewards collection to be optimized. In contrast, the authors in [14] addressed a similar problem that we do. They assumed a deterministic weighted transition system with locally sensed rewards changing according to an unknown dynamics. While they required the satisfaction of an LTL mission, they also targeted to collect maximal rewards locally, within a given horizon. These two goals often cannot be reached simultaneously. If the robot primarily collects high rewards, the mission might never be satisfied and vice versa, if the robot is planned to accomplish the mission, the collected rewards might become low. The authors utilized ideas from the automata-based approach to model checking in order to iteratively find a local path maximizing the collected rewards among the local paths that ensure that a step towards the mission satisfaction is made. This way, they managed to compromise between the two goals.

Our work can be seen as a different, generalized approach to the above problem. We allow the trade-off between the two goals to be partially driven by user-defined preferences that may dynamically change during the execution of the robot. In particular, we assume an LTL mission that includes surveillance of a set of regions and a user-defined preference function expressing the desired trade-off between the surveillance and the reward collection given the history of the robot motion. In other words the preference function determines in each moment whether moving towards a surveyed place or optimization of the collected rewards is of a higher priority. Whereas the local path planned in [14] always guarantees progress towards the satisfaction of the mission, in our case this progress may be deliberately postponed (for a finite amount of time) if the collection of the rewards is prioritized. For example, consider a garbage truck that is required to periodically visit two garbage disposal plants $A$ and $B$ and to arrive to a plant as fully loaded as possible. In [14], each local plan for the truck would send the truck closer to $A$ (or $B$, respectively) and the truck might arrive half-empty. In contrast, through the preference function, we can...
define that collecting the garbage is the primary target until the truck is full enough to drive to a plant and that once it is, driving towards $A$ (or $B$, respectively) becomes the priority. Besides that, we generalize the problem from [14] in the following sense. The authors there assumed that the reward dynamics is completely unknown. Therefore, when planning, they estimate that the rewards collected along a planned local path would be equal to the sum of the rewards that are currently seen at the states of this path and they aim to maximize it. We consider that arbitrary assumptions on the reward dynamics might be given and we estimate the rewards collected along a planned local path accordingly. We also allow for a broader class of optimization functions.

In our solution, we leverage ideas from the automata-based approach to model-checking to provably guarantee the satisfaction of the mission and we introduce several extensions that allow us to support both the preference function and the arbitrary assumptions on the reward dynamics. We build a so-called product automaton that captures all the runs of the transition system that satisfy the mission. We employ the preference function to compute the *attraction* of states in the product automaton and at each time, we choose the most attractive state to be visited next. While the value of the preference function is low, the robot is primarily driven by the sensed rewards. However, as the preference function grows, the surveillance is prioritized and the attraction forces the robot to move not only towards the surveyed regions, but also towards accepting states of the product automaton, i.e., towards the satisfaction of the global specification.

Our contribution can be summarized as follows. We develop a general framework for robot motion planning with high-level LTL mission specifications and locally optimal reward collection with respect to given reward dynamics assumptions and local rewards sensing. We introduce a novel approach that allows to prescribe whether the rewards collection or the mission progress are of a higher interest. We present several illustrative examples and simulation results to demonstrate the usability of our approach.

The rest of the paper is organized as follows. In Section II we review necessary preliminaries. In Section III the problem is described in detail and stated formally. In Section IV we present the solution, correctness and completeness proofs and discussions on the solution optimality. In Section V a case study is introduced and we conclude in Section VI.

II. NOTATION AND PRELIMINARIES

In this section we introduce notation and preliminaries used throughout the paper.

Given a set $S$, we denote by $S^+$ and $S^*$ all finite, nonempty and all infinite sequences of elements from $S$, respectively.

**Definition 1 (Weighted Deterministic Transition System):**

A weighted deterministic transition system (TS) is a tuple $T = (Q, q_0, T, \Pi, L, W)$, where

- $Q$ is a finite set of states;
- $q_0 \in Q$ is an initial state;
- $T \subseteq Q \times Q$ is a transition relation;
- $\Pi$ is a set of atomic propositions;
- $L : Q \rightarrow 2^\Pi$ is a labeling function; and
- $W : T \rightarrow \mathbb{R}_{>0}$ is a weight function.

The states of the transition system represent the regions of the environment and the transitions represent the robot’s capabilities to move between them. We assume that there is a transition from each state. The atomic propositions are properties that are either true or false in each region of the environment, for instance "This region is a pickup/delivery location.". The labeling function $L$ assigns to each state the set of atomic propositions that hold true in this state. The weight function assigns to each transition the amount of time that this transition takes. If the robot is in a state $q$ at time $t$ and follows a transition $(q, q') \in T$, then it is in the state $q'$ at time $t + W((q, q'))$. The time spent in states is 0.

A run of $T$ is an infinite sequence $\rho = q_0 q_1 \ldots$ such that $q_0$ is the initial state and $(q_i, q_{i+1}) \in T$, for all $i \geq 0$. A finite run $\rho_{\text{fin}} = q_i \ldots q_j$ of $T$ is a finite subsequence of a run $\rho = q_0 \ldots q_i \ldots q_j \ldots$ of $T$. A run prefix $\rho_{\text{prefix}}$ of $T$ is a finite run that originates at the initial state $q_0$. For simplicity, we denote by $q = \rho (q \in \rho_{\text{fin}})$ the fact that the state $q$ occurs in the run $\rho$ (the finite run $\rho_{\text{fin}}$). Associated with a run $\rho = q_0 q_1 \ldots$ (and a run prefix $\rho_{\text{prefix}} = q_0 \ldots q_n$) there is a sequence of time instances $t_0 \ldots t_n$ (and $t_0 \ldots t_n$), where $t_0 = 0$, and $t_n$ denotes the time at which the state $q_{n}$ is reached ($t_{i+1} = t_i + W((q_i, q_{i+1}))$. A run $\rho = q_0 q_1 \ldots$ generates a unique word $\omega(\rho) = L(q_0) L(q_1) \ldots$ A control strategy $C : Q^+ \rightarrow Q$ for $T$ assigns the next state to be visited to each run prefix of $T$. The run generated by $C$ is $\rho = q_0 q_1 \ldots$ such that $q_i = C(q_{i-1} \ldots q_0)$, for all $i \geq 1$.

With a slight abuse of notation, we use $W(\rho_{\text{fin}})$, where $\rho_{\text{fin}} = q_i \ldots q_j$ is a finite run of $T$, to denote the total weight of the sequence of transitions $(q_i, q_{i+1}) \ldots (q_j, q_{j+1})$, i.e., $W(\rho_{\text{fin}}) = \sum_{i=1}^{j-1} W((q_k, q_{k+1}))$. Furthermore, we define $W^*(q_i, q_j)$ as the minimum weight of a finite run from $q_i$ to $q_j$. In particular, $W^*(q_i, q_i) = 0$, and $W^*(q_i, q_j) = \infty$ if there does not exist a finite run from $q_i$ to $q_j$.

**Definition 2 (Linear Temporal Logic):** A linear temporal logic (LTL) formula $\phi$ over the set of atomic propositions $\Pi$ is defined according to the following rules:

$$\phi ::= T \mid \pi \mid \neg \phi \mid \phi \lor \phi \mid \phi \land \phi \mid X \phi \mid \phi U \phi \mid G \phi \mid F \phi,$$

where $T$ is always true, $\pi \in \Pi$ is an atomic proposition, $\neg$ (negation), $\lor$ (disjunction) and $\land$ (conjunction) are standard Boolean connectives, and $X$ (next), $U$ (until), $G$ (always) and $F$ (eventually) are temporal operators.

The semantics of LTL is defined over infinite sequences over $2^\Pi$, such as those generated by the transition system $T$ from Def. 1. Assume that $\phi$, $\phi_1$ and $\phi_2$ are LTL formulas over $\Pi$ and $\omega = \omega(0) \omega(1) \ldots \in (2^\Pi)^\omega$ is a word generated by a run $\rho$ of $T$. The word $\omega$ satisfies an atomic proposition $\pi$ if $\pi$ holds in the first position of $\omega$, i.e., if $\pi \in \omega(0)$. The formula $X \phi$ states that $\phi$ needs to hold next, i.e., for the word $\omega(1) \ldots$. The formula $\phi_1 U \phi_2$ means that $\phi_2$ is true eventually, while $\phi_1$ is true at least until $\phi_2$ becomes true. Formulas $G \phi$ and $F \phi$ state that $\phi$ holds always and eventually, respectively. More expressiveness can be achieved by combining the operators. A detailed description of LTL
can be found in [10]. As expected, a run $\rho$ of $T$ satisfies $\phi$ if and only if the word $\omega(\rho)$ generated by $\rho$ satisfies $\phi$.

Definition 3 (Büchi Automaton): A Büchi automaton (BA) is a tuple $B = (S, s_0, \Sigma, \delta, F)$, where
- $S$ is a finite set of states;
- $s_0 \in S$ is an initial state;
- $\Sigma$ is an input alphabet;
- $\delta \subseteq S \times \Sigma \times S$ is a transition relation; and
- $F \subseteq S$ is a set of accepting states.

The semantics of a Büchi automaton is defined over infinite input words. Note that if $\Sigma = 2^\mathbb{N}$, then the input words are infinite sequences of sets of atomic propositions, such as those generated by $T$. A run of $B$ over an input word $\sigma = a_0a_1 \ldots \in \Sigma^\omega$ is a sequence of states $q = s_0s_1 \ldots$ such that $s_0$ is the initial state and $(s_i, a_i, s_{i+1}) \in \delta$, for all $i \geq 0$.

A run $\rho$ is accepting if and only if a state from $F$ appears in $\rho$ infinitely many times. A word $\sigma$ is accepted by the Büchi automaton if there exists an accepting run over $\sigma$.

For any LTL formula $\phi$ over $\Pi$, there exists a Büchi automaton $B_\phi$ with input alphabet $2^\mathbb{N}$ accepting all and only the words satisfying formula $\phi$. Algorithms for translation of an LTL formula into a corresponding Büchi automaton were proposed [15], and several tools are available [16].

Definition 4 (Weighted Product Automaton): A weighted product automaton between a TS $T = (Q, q_0, T, \Pi, L, W)$ and a BA $B_\phi = (S, s_0, \Sigma^\mathbb{N}, \delta, F)$ is a tuple $\mathcal{P} = T \times B_\phi = (S_P, s_{p0}, \delta_P, F_P, W_P)$, where
- $S_P = Q \times S$ is a set of states;
- $s_{p0} = (q_0, s_0)$ is the initial state;
- $\delta_P \subseteq S_P \times S_P$ is a transition relation, where $(q, s), (q', s') \in \delta_P$ if and only if $(q, q') \in T$ and $(s, L(q), s') \in \delta$;
- $F_P = Q \times F$ is the set of accepting states; and
- $W_P : \delta_P \rightarrow \mathbb{R}_{\geq 0}$ is a weight function, where $W_P((q, s), (q', s')) = W((q, q'))$, for all $(q, s), (q', s') \in \delta_P$.

Note that the product automaton defined above is a weighted version of a standard Büchi automaton with a trivial alphabet that is thus omitted. We denote by $\alpha(\rho_P)$ the projection of a run $\rho$ of $\mathcal{P}$ onto its first components, i.e., $\alpha((q_0, s_0)(q_1, s_1) \ldots) = q_0q_1 \ldots$. An accepting run $\rho_P$ of the product automaton $\mathcal{P}$ can be projected onto a run $\alpha(\rho)$ of $T$ that satisfies the formula $\phi$, and vice versa, if $\rho = q_0q_1 \ldots$ is a run of $T$ satisfying $\phi$, then there exists an accepting run $\rho_P = (q_0, s_0)(q_1, s_1) \ldots$ of $\mathcal{P}$.

The product automaton can be also viewed as a transition system $T_{\rho} = (S_P, s_{p0}, \delta_P, L_P, W_P)$, where $L_P((q, s)) = L(q)$, for all $(q, s) \in S_P$. Hence, the objects that are defined on a transition system are defined on the product automaton in the expected way. Namely, we use $W_P(\rho_{Prun}), W_P(p, p_j)$ and $C_P(\rho_{Pfix})$ to denote the total weight of a finite run $\rho_{Prun}$, the minimum weight between states $p_i, p_j$ and the control strategy for $\mathcal{P}$, respectively.

III. PROBLEM FORMULATION

Consider a robot moving in a partitioned environment modeled as a weighted deterministic transition system. The states of the transition system correspond to individual regions of the environment and the transition between them model the robot motion capabilities. Assume, that there is a dynamically changing non-negative real-valued reward associated with each state of the transition system. The robot senses the rewards in its close proximity and collects the rewards as it visits the regions of the environment, i.e., as the states of the transition system change. Moreover, the robot is given a high-level LTL mission. The problem addressed in previous literature [14] is to design a control strategy that (1) guarantees the satisfaction of the mission and (2) locally maximizes the collected rewards.

We focus on a different version of the above problem allowing for partial regulation of the trade-off between the two objectives. In particular, first, we consider a user-defined preference function that, given a history of robot’s movement, expresses whether moving closer to a region under surveillance or collecting rewards is prioritized. Second, we consider arbitrary reward dynamics that might be unknown, known partially or even fully. We capture the concrete reward dynamics assumptions through a so-called state potential function. The problem we address is to design a control strategy that (1) guarantees the satisfaction of the mission, (2) locally optimizes the collection of rewards, and (3) takes into consideration the preference function and the reward dynamics assumptions.

We formalize the problem as follows. The robot motion in the environment is given as a TS $T = (Q, q_0, T, \Pi, L, W)$ (Def. [1]). The rewards can be sensed at time $t_k$ within the visibility range $v \in \mathbb{R}_{\geq 0}$ from the robot’s current position $q_k$. We denote by $V(q_k) = \{ q \mid W^*(q_k, q) \leq v \}$ the set of states that are within the visibility range $v$ from $q_k$ (assuming that $q \in V(q_k)$, for all $(q_k, q) \in T$) and by $R : Q \times Q^+ \rightarrow \mathbb{R}_{\geq 0}$ the reward function, where $R(q, q_0 \ldots q_k)$ is the reward sensed in the state $q$ at time $t_k$ after executing the run prefix $q_0 \ldots q_k$. Note that $R(q, q_0 \ldots q_k)$ is defined iff $q \in V(q_k)$ and it is known only at time $t_k$ (and later), not earlier.

A user-defined planning horizon and a state potential function are employed to capture user’s assumptions about the reward dynamics and her interests. For instance, the values of the rewards may increase or decrease at most by 1 during 1 time unit, they may appear according to a probabilistic distribution, or their changes might be random. The user might have full, partial or no knowledge of the reward dynamics. The rewards might disappear once they are collected by the robot, or they might not. The user might be interested in the maximal, expected, or minimal sum of rewards that can be collected from a given state during a finite run whose weight is no more than the planning horizon. The concrete definitions of the planning horizon and the state potential function are meant to be specifically tailored for different cases. Formally, the horizon is $h \in \mathbb{R}_{\geq 0}$, $h = \max(q, q') \in T W(q, q')$ and the state potential function is $pot : Q \times Q^+ \times \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$, where $pot(q, q_0 \ldots q_k, h)$ is the potential of the state $q$ at time $t_k$. More precisely, the value of $pot(q, q_0 \ldots q_k, h)$ is defined for all $q$, where $(q_k, q) \in T$ and captures the rewards that can be collected
after execution of the run prefix $q_0 \ldots q_k$ during a finite run $\rho_{\text{fin}} \in P_{\text{fin}}(q, q_k, h)$, where

$$P_{\text{fin}}(q; q_k, h) = \{ \rho_{\text{fin}} \mid \rho_{\text{fin}} \text{ is a finite run of } T, \text{ such that }
\begin{align*}
( i) & \rho_{\text{fin}} \text{ originates at } q; \\
( ii) & W(\rho_{\text{fin}}) + W((q, q_k)) \leq h; \text{ and} \\
( iii) & \text{the states that appear in } \rho_{\text{fin}} \text{ belong to } V(q_k)\}.
\end{align*}$$

Note, that the visibility range $v$ and the planning horizon $h$ are independent. Whilst $v$ determines the set of states whose rewards are visible from the current state $q_k$, $h$ gives the maximal total weight of a planned finite run $\rho_{\text{fin}}$ within $V(q_k)$, which can be even greater than $v$.

**Example 1:** The function stating that the potential of $q$ is the maximal sum of rewards that can be collected from $q$ assuming that the rewards do not change while the robot can sense them and that they disappear once collected is

$$\text{pot}(q, q_0 \ldots q_k, h) = \max_{\rho_{\text{fin}} \in P_{\text{fin}}(q; q_k, h)} \sum_{q' \in \rho_{\text{fin}}} R(q', q_0 \ldots q_k).$$

In fact, this is how authors in [14] estimate the amount of rewards collected on a local path. To define our problem, we assume that there is a set of regions labeled with a so-called surveillance proposition $\pi_{\text{sur}} \in \Pi$ and a part of the mission is to periodically fulfill the surveillance proposition by visiting one of those regions. The missions are then expressed as LTL formulas of form

$$\phi = \varphi \land G F \pi_{\text{sur}},$$

where $\varphi$ is an arbitrary LTL formula over $\Pi$. The subformula $G F \pi_{\text{sur}}$ states that the surveillance proposition $\pi_{\text{sur}}$ has to be visited always eventually, i.e., infinitely many times. Note, that formulas $\phi = \varphi \land G F T$ hold true if and only if $\varphi$ hold true and therefore the prescribed form does not restrict the full LTL expressivity.

The user can partially guide whether the robot should collect high rewards or whether it should rather make a step towards the satisfaction of the surveillance proposition $\pi_{\text{sur}}$ through a preference function. For example, the preference function can grow linearly with time since the latest visit to $\pi_{\text{sur}}$, meaning that going towards $\pi_{\text{sur}}$ gradually gains more importance. In contrast, the value of the preference function can stay low until the latest visit to $\pi_{\text{sur}}$ happened no later than 100 time units ago and after that increase rapidly, expressing that the robot is preferred to collect rewards for 100 time units and then to move towards $\pi_{\text{sur}}$ quickly.

Formally, the preference function $\text{pref} : Q^+ \to \mathbb{R}_{\geq 0}$ assigns a non-negative real value to each executed run prefix $q_0 \ldots q_k$ of $T$ (possibly) taking into account the current values of the state potential function.

**Example 2:** An example of a preference function is

$$\text{pref}(q_0 \ldots q_k) = 0.01 \cdot W_i \cdot \max_{(q_0 \ldots q_k) \in T} \text{pot}(q_0 \ldots q_k, h),$$

where $W_i = W(q_i \ldots q_k)$, such that $\pi_{\text{sur}} \in L(q_i)$, and $\pi_{\text{sur}} \notin L(q_j)$, for all $i < j \leq k$. If the surveyed state is being avoided, the total weight $W_i$ since the last visit to a surveyed state gradually grows and eventually, the value of $\text{pref}(q_0 \ldots q_k)$ overgrows the value of $\text{pot}(q_0 \ldots q_k, h)$ for all $q$.

A shortening indicator function $I$ indicates whether a transition leads the robot closer to a state subject to surveillance. $I : T \to \{0, 1\}$ is defined as follows:

$$I((q, q')) = \begin{cases} 1 & \text{if } \min_{q_\in \Pi} W^*(q', \pi) < \min_{q_\in \Pi} W^*(q, \pi), \\ 0 & \text{otherwise,} \end{cases}$$

where $(q, q') \in T$ and $Q_\in = \{ q_\in \mid \pi_{\text{sur}} \in L(q_\in) \}$.

We are now ready to formally state our problem.

**Problem Formulation 1:** Given the robot motion model $T = (Q, q_0, T, L, L, W)$; the surveillance proposition $\pi_{\text{sur}} \in \Pi$; the visibility range $v$; the reward $R(q_0, q_1 \ldots q_k)$ at time $t_k$, for all $q \in V(q_k)$; the planning horizon $h$; the state potential function $\text{pot}$; the LTL formula $\phi$ over $\Pi$ (Eq. [1]); and the preference function $\text{pref}$, find a control strategy $C$, such that

(i) the run generated by $C$ satisfies the mission $\phi$ and

(ii) assuming that $q = C(q_0 \ldots q_k)$, the cost function

$$\text{pot}(q_0 \ldots q_k, h) + I((q_k, q)) \cdot \text{pref}(q_0 \ldots q_k)$$

is maximized at each time $t_k$.

Intuitively, condition (ii) is interpreted as follows. At each time, the aim is to go to the state with the best trade-off between the amount of potentially collected rewards and the importance of fast surveillance. The higher the value of the preference function, the more likely a state closer to $\pi_{\text{sur}}$ is to be chosen. Note that, in general, the satisfaction of the condition (ii) may cause violation of the objective (i). Our goal is thus to provably guarantee accomplishment of the mission and to maximize Eq. [2] if possible.

Our approach leverages some ideas from the automata-based solution from [14]. However, several issues have to be overcome to support the user-defined trade-off as it will become clear in the following section. The solution consists of two consecutive steps. The first one is an offline preparation before the deployment of the system. It involves a construction of a BA for the given LTL mission and its product with the TS. The offline algorithm assigns two Boolean indicators to each transition of the product automaton, which indicate whether the transition induces a progress to a subgoal, i.e., a surveyed state of the transition system and both a surveyed state and an accepting state of the product automaton, respectively. In the second step, an online feedback algorithm, which determines the next state to be visited by the robot, is iteratively run. In each iteration, attractions of the states of the product automaton are computed. The repeated choices of the maximal attraction states lead to an eventual visit not only to a surveyed state, but also to an accepting state of the product automaton, assuming that the following holds:

**Assumption 1:** For each run $q_0q_1 \ldots$, with the property that $\exists m_1, \forall m > n_1 : \pi_{\text{sur}} \notin L(q_m)$, it holds that $\exists n_2, \forall m > n_2 : \text{pref}(q_0 \ldots q_m) > \text{pot}(q_0 \ldots q_m, h)$, for all $q$, where $(q_m, q) \in T$.

As we will show in Sec. [1VC], the satisfaction of the LTL mission is guaranteed provided that the above assumption
is true. From now on, we assume that Assump. \[1\] holds. Intuitively, it says that if a visit to a surveyed state is postponed for a long time, the value of the preference function overweight the value of the state potentials. Note that this is, in fact, quite natural. It only captures the fact, that the user who defines the potential and the preference function wishes to satisfy the LTL formula in long term and therefore her interest in collecting the rewards. Several examples of pot and pref functions that respect this assumption will be shown in Section [IV]

IV. Solution

In this section, we give the details of our solution to Problem [1] and prove its correctness and completeness. Discussions on the optimality of the solution are included, too.

A. Offline Indicator Assignment

Let $B_\phi = (S, S_0, \Sigma, \delta, F)$ be a Büchi automaton corresponding to the LTL formula $\phi = \forall G F \pi_{surr}$ (Eq. [1]) and $P = T \times B_\phi = (S_\phi, S_{\phi 0}, \delta_\phi, F_\phi, W_\phi)$ the product automaton constructed according to Def. [4].

Let $S_{\phi \pi} = \{(q, s) \in S_\phi \mid \pi_{surr} \in L(q)\}$ denote the subset of states of $P$ that project onto the surveyed states in $T$. Furthermore, let $F_{\phi \pi}^\infty \subseteq F_\phi$ and $S_{\phi \pi}^\infty \subseteq S_{\phi \pi}$ be the sets of states from which $S_{\phi \pi}$ and $F_\phi$ can be visited infinitely many times, respectively. Sets $F_{\phi \pi}^\infty$ and $S_{\phi \pi}^\infty$ can be computed iteratively as the maximal sets of states from which a state in $S_{\phi \pi}$ and $F_{\phi \pi}$ is reachable via a finite run of nonzero length, respectively (see Alg. [4] lines 39).

**Lemma 1:** A run $\varphi_\pi$ of $P$ is accepting iff a state from $F_{\phi \pi}^\infty$ and a state from $S_{\phi \pi}^\infty$ appear in $\varphi_\pi$ infinitely many times.

**Proof:** Let $\varphi_\pi = \varphi_\pi(0) \varphi_\pi(1) \ldots$ be an accepting run of $P$, i.e., a run with infinitely many visits to $F_\phi$. Note that there is a state in $S_{\phi \pi}$ that appears in $\varphi_\pi$ infinitely many times, because $\varphi_\pi$ satisfies $\phi$ and hence $G F \pi_{surr}$. Then there exist infinite index sets $I, J \subseteq \mathbb{N}$, where $\varphi_\pi(i) \in F_\phi$, $\varphi_\pi(j) \in S_{\phi \pi}$, for all $i \in I$ and $j \in J$. For each state $\varphi_\pi(i) \in F_\phi$, $i \in I$ there exist infinitely many states $\varphi_\pi(j) \in S_{\phi \pi}$ where $i < j$, and analogous holds for each state $\varphi_\pi(j) \in S_{\phi \pi}, j \in J$. Hence, all states $\varphi_\pi(i), \varphi_\pi(j)$ where $i \in I, j \in J$ belong to $F_{\phi \pi}^\infty \cup S_{\phi \pi}^\infty$, respectively. On the other hand, if a state from $F_{\phi \pi}^\infty$ occurs on $\varphi_\pi$ infinitely many times, then $\varphi_\pi$ is clearly accepting.

For each state $p \in S_\phi$ we define the minimum weight of a finite run from $p$ to a state from $S_{\phi \pi}^\infty$

$$W^{\phi}_{\pi}(p) = \min_{p' \in S_{\phi \pi}^\infty} W_{\phi}(p, p')$$  (3)

and the minimum weight of a finite run from $p$ to $S_{\phi \pi}^\infty$ containing a state $p' \in F_{\phi \pi}$

$$W^{\phi}_{F\pi}(p, p') = \min_{p'' \in S_{\phi \pi}^\infty} \left(W^{\phi}_{\pi}(p, p') + W^{\phi}_{\pi}(p', p'')\right).$$  (4)

Moreover, we define

$$W^{\phi}_{\pi\phi}(p) = \left(W^{\phi}_{\pi}(p, p'), W^{\phi}_{F\pi}(p, p')\right)$$  (5)

where $p' \in F_{\phi \pi}^\infty$ minimizes $W^{\phi}_{\pi}(p, p')$ among the set of states that minimize Eq. [4]. Given $W^{\phi}_{\pi}(p_1) = (u_1, v_1)$ and $W^{\phi}_{\pi}(p_2) = (u_2, v_2)$, $W^{\phi}_{\pi}(p_1) < W^{\phi}_{\pi}(p_2)$ if and only if $u_1 < u_2$ and $v_1 < v_2$.

Note that each state $p \in S_\phi$ with $W_{\phi \pi}(p) = \infty$ or $W^{\phi}_{\pi}(p) = (\infty, \infty)$ cannot occur on any accepting run of $P$.

Therefore, we assume from now on that $P$ contains only states $p \in S_\phi$ with $W_{\phi \pi}(p) < \infty$ and $W^{\phi}_{\pi}(p) < (\infty, \infty)$.

**Lemma 2:** $\forall p \in S_\phi \setminus S_{\phi \pi}^\infty, \exists p' \in S_\phi \setminus F_{\phi \pi}^\infty, \exists p'' \in S_{\phi \pi} \setminus S_{\phi \pi}^\infty$.

**Proof:** Follows directly from Eq. [3] and [4].

We are now ready to define the shortening indicator functions $I_{\phi \pi}, I_{\phi \phi}: \delta \rightarrow \{1, 0\}$, which indicate whether a transition induces progress towards the set $S_{\phi \pi}^\infty$ and towards both the set $F_{\phi \pi}^\infty$ and the set $S_{\phi \pi}^\infty$ via a state in $F_{\phi \pi}^\infty$, respectively.

$$I_{\phi \pi}(\langle p, p' \rangle) = \begin{cases} 1 \text{ if } W^{\phi}_{\pi}(p) > W^{\phi}_{\pi}(p'), \\ 0 \text{ otherwise,} \end{cases}$$  (6)

where $x \in \{\pi, \phi\}$.

**Corollary 1:** $\forall p \in S_\phi \setminus S_{\phi \pi}^\infty, \exists (p, p') \in \delta$, such that $I_{\phi \pi}(\langle p, p' \rangle) = 1$ and $\forall p \in S_\phi \setminus F_{\phi \pi}^\infty, \exists (p, p') \in \delta$, such that $I_{\phi \phi}(\langle p, p' \rangle) = 1$.

The outline of the indicator assignment procedure for the product automaton $P$ is summarized in Alg. [1].

**B. Online Planning**

The online planning algorithm is run at each $t_k$, such that $q_0 \ldots q_k$ is the executed run prefix so far (i.e., till the current time $t_k$) and it determines the next state $C(q_0 \ldots q_k)$ of $T$ to be visited. Simply put, we plan in the product automaton $P$ and then we project the planned onto $T$. Formally, $T$ starts in its initial state $q_0$ and $P$ in its initial state $(q_0, s_0)$. For each run prefix $(q_0, s_0) \ldots (q_k, s_k)$ of $P$, the algorithm computes the next state of $P$, denoted by $C_{\phi}(q_0, s_0) \ldots (q_k, s_k) = (q_{k+1}, s_{k+1})$. The next state of $T$ is $C(q_0 \ldots q_k) = q_{k+1}$.

To guarantee that the control strategy $C$ generates a run of $T$ satisfying $\phi$, it is sufficient to ensure that the control
strategy $C_p$ generates a run of $P$ that visits $F_p$ infinitely many times. In $T$, the high value of the preference function $\phi$ was used to guide the robot towards $\pi_{sur}$. Projected into the product automaton, the high value of $\phi$ can "send" the robot towards a state in $S_{P,F}^\infty$. We expand this idea and use the preference function to guide the robot not only towards $S_{P,F}^\infty$, but also towards $F_{P,F}^\infty$. This way, we ensure that $F_{P,F}^\infty$ is indeed visited infinitely many times.

In particular, we introduce two subgoals in $P$. The first one is the mission subgoal, when a visit to $F_{P,F}^\infty$ is targeted. The second one is the surveillance subgoal, when we aim to visit $S_{P,F}^\infty$. At each time, one of the subgoals is to be achieved and once it is, the subgoals are switched and the other one is to be achieved. Progress towards both subgoals is governed by the preference function to guide the robot not only towards $F_{P,F}^\infty$, but also towards $F_{P,F}^\infty$. Thus, we define a projection function $\bar{\pi}$ which we define for the product automaton in analogous way as the cost function (Eq. 2) for Problem 1.

Consider the product $P$ obtained after the execution of the offline preparation algorithm (Alg. 1). Assume, that $\phi$ is satisfiable, i.e., that $F_{P,F}^\infty$ and $S_{P,F}^\infty$ are both nonempty and $(q_0, s_0) \in S_p$. The product $P$ naturally inherits the rewards from $T$, i.e., $R_P(q, s, (q_0, s_0)\ldots(q_k, s_k)) = R(q_0 \ldots q_k)$. Thus, the value of reward function can be computed on the product automaton (or, more precisely, on its underlying TS $T_P$) using $R_P$. We use $p_{ot}\bar{\pi}(p, q_{\text{prefix}})$ to denote the value of the potential function for a state $p$ computed on $P$.

The value of the attraction $\text{attr}_P : S_p \times S_p \times \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$ is computed differently for both subgoals. Initially, the subgoal to be achieved is the surveillance one and the attraction is

$$\text{attr}_P(p, q_{\text{prefix}}, h) = p_{ot}\bar{\pi}(p, q_{\text{prefix}}, h) + I_P\bar{\pi}(p_{k}, p) \cdot \phi(q_{\text{prefix}}),$$

where $q_{\text{prefix}} = p_0 \ldots p_k$, $(p_k, p) \in \delta_P$. For any run prefix $p_0 \ldots p_k$, let $C_p(p_0 \ldots p_k)$ be the state with the highest attraction (if there are more of them, we choose one randomly). Hence, if the attraction of a state that is not closer to the subgoal is higher than the attraction of ones that are, the collection of rewards is preferred and vice versa. However, note that repeated choices of the states that maximize $\phi$ and vice versa. Therefore, if a run prefix $p_0 \ldots p_k$ is visited infinitely many times, the "non-shortening" transitions might be always chosen to follow and a visit to $F_{P,F}^\infty$ might be infinitely postponed.

Thus, we define a projection function $\bar{\alpha}$ that projects a run prefix $q_{\text{prefix}}$ of $P$ onto the corresponding run of $T$ while removing $\pi_{sur}$ from some of the states. In particular, on $\bar{\alpha}(q_{\text{prefix}})$, the proposition $\pi_{sur}$ appears at most once in between two successive visits to an accepting state in $F_{P,F}^\infty$.

**Definition 5 (Projection $\bar{\alpha}$):** Let $T = (q, q_0, T, \Pi, L, W)$ be a transition system, where $Q = Q \cup \{q^0 | q \in Q\}$; if $(q, q') \in T$, then $(q, q'), (q, q')$, $(q', q')$; $(q', q') \in T$. If $W(q, q') = W(q, q') = W(q, q') = W(q, q')$, and $L(q) = L(q)$, and $L(q) = L(q) \setminus \{s_{sur}\}$, for all $q \in Q$. Let $q_{\text{prefix}} = (q_0, s_0) \ldots (q_k, s_k)$ be a run prefix of $P$, $\bar{\rho}_{\text{prefix}}(0) = q_0$, $\bar{\rho}_{\text{prefix}}(i) = q_i$ if $s_{\text{sur}} \not\in L(q_i)$, or $\pi_{sur} \in L(q_i)$ and $\exists j < i$, such that $(q_j, s_j) \in F_{P,F}^\infty$ and $s_{\text{sur}} \not\in L(q_i)$, for all $j \leq l < i$; and $\bar{\rho}_{\text{prefix}}(i) = q_i$, otherwise.

The definition of the attraction for the mission mode is

$$\text{attr}_P(p, q_{\text{prefix}}, h) = p_{ot}(p, q_{\text{prefix}}, h) + I_P\bar{\alpha}(p_{k}, p) \cdot \phi(q_{\text{prefix}}),$$

where $q_{\text{prefix}} = p_0 \ldots p_k$, $(p_k, p) \in \delta_P$. Similarly as for the surveillance subgoal, the state $C_p(p_0 \ldots p_k)$ is the state maximizing attraction (if there are more of them, we choose one randomly). The construction of the attraction together with Assump. 1 ensure that the mission subgoal is always eventually reached. Once it is, we aim for the surveillance subgoal again. If both of the subgoals are reached simultaneously, the surveillance subgoal is set to be reached.

The outline of the solution to Problem 1 is given in Alg. 2.

**Algorithm 2 Solution to Problem 1**

**Input:** $T$, $\pi_{sur}$, $\nu$, $R$, $h$, $\phi$, $\rho$, $\phi$ **Output:** Control strategy $C$

1. compute $B_p$, $P = T \times B_p$ and run Alg. 1
2. if $F_{P,F}^\infty = \emptyset$ or $(q_0, s_0) \not\in S_p$ then
3. return "Mission cannot be accomplished".
4. end if
5. $q_{\text{prefix}} := (q_0, s_0)$, subgoal := $\pi_{sur}$, $k := 0$
6. while true do
7. for all $p$, s.t. $(p_k, p) \in \delta_P$ do
8. compute $\text{attr}(p, q_{\text{prefix}}, h)$ (Eq. 7) if subgoal = $\pi_{sur}$ and $\text{Eq. 8}$ if subgoal = $\phi$
9. end for
10. $C_p(q_{\text{prefix}}) := p$ maximizing $\text{attr}_P(p, q_{\text{prefix}}, h)$
11. $C(\alpha(q_{\text{prefix}})) := \alpha(C_p(q_{\text{prefix}}))$
12. if subgoal = $\pi_{sur}$ and $C_p(q_{\text{prefix}}) \in S_{P,F}^\infty$ then
13. subgoal := $\phi$
14. end if
15. if subgoal = $\phi$ and $C_p(q_{\text{prefix}}) \in F_{P,F}^\infty$ then
16. subgoal := $\pi_{sur}$
17. end if
18. concatenate $C_p(q_{\text{prefix}})$ to $q_{\text{prefix}}$; $k := k + 1$
19. end while

**C. Discussion**

In this section, we prove that under Assump. 1 our algorithm is correct and complete with respect to the satisfaction of the LTL formula (condition (i) of Problem 1). We discuss the sub-optimality of the solution and we introduce an assumption, under which the local plan is optimal with respect to condition (ii) of Problem 1 among the solutions that do not cause an immediate, unparsable violation of $\phi$.

**Theorem 1 (Correctness and Completeness):** Alg. 2 returns a strategy $C$ that generates a run of $T$ satisfying $\phi$ if and only if such a strategy exists.
This is necessary for correctness of the algorithm, however, Second, the preference function in the attraction function cannot be accomplished. Then the product automaton leads closer to $\pi$, an immediate violation of the formula. In contrast, if the taking into consideration only finite runs that do not cause a state of the current subgoal of the online planner is the surveillance subgoal, the following optimality statement can be made: the mission subgoal is the current one. Then, according to Assump. 1 and the definition of the attraction function, the value of the preference function in the attraction function (Eq. 7) is computed for $\alpha(p_0 \ldots p_k)$ instead for $\alpha(p_0 \ldots p_k)$, as a result, the value of $\text{pref}(\alpha(p_0 \ldots p_k))$ in the attraction function (Eq. 7) might be different than the corresponding value of $\text{pref}(q_0 \ldots q_k)$ in the cost function (Eq. 2).

In case $F_{\infty} = \{ q' \in S_P \mid q \in S_{\infty} \}$ and $(q, q') \in \delta_P$, the mission subgoal is reached always exactly one planning step after the surveillance subgoal is reached. Therefore, we can reach the optimality that was stated in the previous paragraph for the surveillance subgoal also for the mission subgoal, since all the transitions from $S_{\infty}^\pi_\text{sur}$ are always "shortening" with respect to $F_{\infty}$. In particular, this is the case if a Büchi automaton with the property that all the transitions leading to an accepting states are labeled with a set containing $\pi_{\text{sur}}$, is used in the product automaton construction. For instance, a surveillance fragment of LTL defined in [17] guarantees existence of such a BA. The fragment includes LTL formulas that require to repeatedly visit a surveillance proposition $\pi_{\text{sur}}$ (called an optimizing proposition in [17]) and to visit a given set of regions in between any two successive visits to states satisfying $\pi_{\text{sur}}$. In addition, ordering constraints, request-response properties, and safety properties are allowed.

Complexity: The size of a BA for an LTL formula $\phi$ is $2^{O(|\phi|)}$ in the worst case, where $|\phi|$ denotes the length of the formula $\phi$ [15]. However, note that the actual size of the BA is in practice often quite small. The size of the product automaton $P$ is $O(|Q| \cdot 2^{O(|\phi|)})$. A simple modification of the Floyd-Warshall algorithm is employed to find the minimum weights between each pair of states in $O(|P|^3)$. The same complexity is reached for the computation of $F_{\infty}^P \cdot S_{\infty}^P \cdot W_{\text{fin}}^P$ and $W_{\text{sur}}^P$. The shortening indicators $I_{\text{fin}}, I_{\text{sur}}$ can be computed in linear time and space with respect to the size of $P$. The overall complexity of Alg. 1 is $O(|Q| \cdot 2^{O(|\phi|)})^3$. The complexity of the online planning algorithm highly depends on the complexity of the state potential and the preference functions. The set $P_{\text{fin}}(q, q_k, h)$ can be computed in $O(d^h)$, where $d$ denotes the maximal out-degree of states of $P$. If pot and pref functions took constant time to compute, the online planning algorithm would be in $O(d \cdot d^h)$ per iteration.

V. Example

We implemented the framework with several concrete choices of the state potential and the preference function in a Java applet [18]. In this section, we report on simulation results to illustrate employment of our approach.

We consider a data gathering robot in a grid-like partitioned environment modeled as a TS depicted in Fig. 1. The robot collects data packages of various, changing sizes

| Table I: Definitions of the state potential and the preference functions used in the case study. |
|---------------------------------------------------------------|
| $\text{pot}_1(q, q_0 \ldots q_k, h) = \max_{\rho(0) \ldots \rho(n) \in P_{\text{fin}}(q, q_k, h)} \sum_{i=0}^{n} f_1(\rho(i), q_0 \ldots q_k, \rho(0) \ldots \rho(n))$, and |
| $\text{pot}_2(q, q_0 \ldots q_k, h) = \max_{\rho(0) \ldots \rho(n) \in P_{\text{fin}}(q, q_k, h)} \left( \max_{i=0 \ldots n} f_2(\rho(i), q_0 \ldots q_k, \rho(0) \ldots \rho(n)) \right)$, |
| where $f_1, f_2(\rho(i), q_0 \ldots q_k, \rho(0) \ldots \rho(n)) = R(\rho(i), q_0 \ldots q_k) - W(\rho(0) \ldots \rho(i))$ if this value $> 0$, $\rho(i) \neq q_k$ and $\rho(j) \neq \rho(i)$ for all $j < i$, and $f_1 = 15$, $f_2 = 0$ otherwise. |
| $\text{pref}_1(q_0 \ldots q_k) = 0$ if $W(q_n \ldots q_k) \leq 50$, and $\text{pref}_2(q_0 \ldots q_k, h)$ otherwise, |
| $\text{pref}_2(q_0 \ldots q_k) = \frac{\sqrt[q]{W(q_n \ldots q_k)}}{q_n \ldots q_k} \cdot \text{pot}^*(q_0 \ldots q_k)$, and $\text{pref}_2(q_0 \ldots q_k) = \frac{1}{\sqrt[q]{W(q_n \ldots q_k)}} \cdot \text{pot}^*(q_0 \ldots q_k)$, |
| where $i_\text{p}$ is maximal $0 \leq i \leq k$, such that $q_{i_\text{p}} \in \pi_{\text{sur}}$ and $\text{pot}^*(q_0 \ldots q_k)$ is the maximal $\text{pot}(q, q_0 \ldots q_k, h)$ among all $q$, where $(q_k, q) \in T$. |
Fig. 1: A transition system representing the robot (illustrated as the black dot) motion model in a partitioned environment. Individual regions are depicted as nodes (states). Transmitters are in green (labeled with propositions $a$ and $b$, respectively), unsafe locations (labeled with $u$) are in red. The set of transitions contains every pair $(q, q')$ of vertically, horizontally or diagonally neighboring states. Weights of a horizontal and a vertical transition are 2, weight of a diagonal transition is 3.

(rewards) in the visited regions. The following is known about the reward dynamics: A non-negative natural reward can appear in a state with the current reward equal to 0. The probability of the fresh reward being from $\{0, \ldots, 15\}$ is 50% as well as from $\{16, \ldots, 60\}$ (i.e., the smaller-sized data packages are more likely to occur). The reward drops by 1 every time unit as the data outdated. The visibility range $v$ is 6. For example, in Fig. 1 the visibility region $V(q_0)$ for the current state $q_0$ is depicted as the blue-shaded area.

The mission assigned to the robot is to alternately visit the two transmitters (in green, labeled with propositions $a$, and $b$, respectively), while avoiding unsafe locations (in red, labeled with $u$). The surveillance proposition $\pi_{sur}$ is true in both transmitter regions. The LTL formula for the mission is

$$\phi = G (a \Rightarrow X (\neg a U b)) \land G (b \Rightarrow X (\neg b U a)) \land G (\neg u) \land GF \pi_{sur}.$$ 

In our simulations, we consider the planning horizon $h = 9$ and several variants of the state potential function and the preference function that are summarized in Table I. The first state potential function $\phi_{pot_1}$ is the maximal sum of rewards that can be collected on a finite run while taking into account the reward behavior assumptions described above. If the run visits a state more than once or a reward of a state drops below 0, we assume the reward there is 15. The second state potential function $\phi_{pot_2}$ is defined as the maximal size of a single data package that can be collected on a finite run.

The respective ratio of the value of $\phi$ and the maximum value of $\phi$ is always non-decreasing with the time elapsed since last transmission and the value of $\phi$ overgrows the maximum value of $\phi$ when the elapsed time is 50. Intuitively, $\phi_{pot_1}$ sets zero importance on going towards a transmitter if the last transmission occurred not more than 50 time units ago. On the other hand, $\phi_{pot_2}$ rises quite slowly at the beginning and very quickly later. In contrast, the function $\phi_{pot_3}$ grows very fast in the beginning and its growth slows down.

The experiments were run on Mac OS X 10.7.3 with...
2.7 GHz Intel Core i5 and 4 GB DDR3 memory. The BA had 8 states (3 accepting) and it satisfied the condition for optimality from Sec. [IV-C]. The product automaton had 800 states. The offline part of Alg. [2] took 6 seconds and one iteration of the online planning algorithm 1-2 milliseconds.

VI. CONCLUSIONS AND FUTURE WORK
We proposed a general framework for robot motion planning in environment with dynamically changing rewards. While a high-level surveillance mission is guaranteed to be accomplished, the user-defined priorities on trade-off between the surveillance frequency and the reward collection are taken into account. The motion of the robot is modeled as a weighted transition system. Although the weights are in this paper interpreted as time durations of the transitions, they can be, in general interpreted, as any quantitative aspect, such as length or cost. In future work, we would like to extend the suggested framework for systems that are modeled as Markov decision processes and to reaching solution optimality for special subclasses of the reward dynamics. Our plan is also to extend the implementation of the framework.

VII. ACKNOWLEDGEMENT
We thank Calin Belta and Dennis Ding from Boston University for many useful discussions.

REFERENCES
[1] S. M. LaValle, Planning Algorithms. Cambridge Univ. Press, 2006.
[2] M. Antoniotti and B. Mishra, “Discrete Event Models + Temporal Logic = Supervisory Controller: Automatic Synthesis of Locomotion Controllers,” in Proceedings of IEEE ICRA, 1995, pp. 1441–1446.
[3] S. G. Loizou and K. J. Kyriakopoulos, “Automatic Synthesis of Multiagent Motion Tasks Based on LTL Specifications,” in Proceedings of IEEE CDC, 2004, pp. 153–158.
[4] M. Kloetzer and C. Belta, “A Fully Automated Framework for Control of Linear Systems from Temporal Logic Specifications,” IEEE Transactions on Automatic Control, vol. 53, no. 1, pp. 287–297, 2008.
[5] S. Karaman and E. Frazzoli, “Sampling-based Motion Planning with Deterministic μ-Calculus Specifications,” in Proceedings of IEEE CDC, 2009, pp. 2222–2229.
[6] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas, “Temporal Logic-based Reactive Mission and Motion Planning,” IEEE Transactions on Robotics, vol. 25, no. 6, pp. 1370–1381, 2009.
[7] G. E. Fainekos, A. Girard, H. Kress-Gazit, and G. J. Pappas, “Temporal Logic Motion Planning for Dynamic Robots,” Automatica, vol. 45, no. 2, pp. 343–352, 2009.
[8] M. Lahijanian, S. B. Andersson, and C. Belta, “Control of Markov Decision Processes from PCTL Specifications,” in Proceedings of ACC, 2011, pp. 311 –316.
[9] A. Pnueh, “The Temporal Logic of Programs,” in Proceedings of IEEE FOCS, 1977, pp. 46–57.
[10] C. Baier, J.-P. Katoen, and K. G. Larsen, Principles of Model Checking. MIT Press, 2008.
[11] J. B. Rawlings and D. D. Q. Mayne, Model Predictive Control Theory and Design. Nob Hill Pub., 2009.
[12] T. Wongpiromsarn, U. Topcu, and R. M. Murray, “Receding Horizon Temporal Logic Planning for Dynamical Systems,” in Proceedings of IEEE CDC/CCC, 2009, pp. 5997–6004.
[13] ——, “Receding Horizon Control for Temporal Logic Specifications,” in Proceedings of HSCC, 2010, pp. 101–110.
[14] X. C. Ding, C. Belta, and C. G. Cassandras, “Receding Horizon Surveillance with Temporal Logic Specifications,” in Proceedings of IEEE CDC, 2010, pp. 256–261.
[15] P. Gastin and D. Oddoux, “Fast LTL to Büchi Automata Translation,” in Proceedings of CAV, 2001, pp. 53–65.
[16] J. Klein. (2007) ltl2dstar – LTL to Deterministic Streett and Rabin Automata. [Online]. Available: [http://www.ltl2dstar.de/](http://www.ltl2dstar.de/)
[17] Y. Chen, J. Tumova, and C. Belta, “LTL Robot Motion Control based on Automata Learning of Environmental dynamics,” in Proceedings of IEEE ICRA, 2012, pp. 5177–5182.
[18] M. Svořenova, J. Tumova, J. Barnat, and I. Černá. (2012) Simulation of attraction-based approach to receding horizon control with LTL specification. [Online]. Available: [http://www.fi.muni.cz/$sim$x175388/simulationLTLrhc](http://www.fi.muni.cz/$sim$x175388/simulationLTLrhc)