2015 23rd Mediterranean Conference on Control and Automation (MED)
June 16-19, 2015. Torremolinos, Spain

Motion Planning in Dynamic Environments with Bounded Time Temporal Logic Specifications

Dipankar Maity and John S. Baras

Abstract— **In this paper, we consider the problem of robotic motion planning that satisfies some bounded time high level specifications. Although temporal logic can efficiently express high level specifications such as coverage, obstacle avoidance, temporal ordering of tasks etc., it fails to address problems with explicit timing constraints. The inherent limitations of Linear Temporal Logic (LTL) to address problems with explicit timing constraints have been overcome by translating the planning problem from the workspace of the robot to a higher dimensional space called *spacetime* where the existing LTL semantics and grammar are sufficient to mathematically formulate the bounded time high level specifications. A discrete path will be generated, that will meet the specifications with all timing constraints and, at the same time, it will optimize some cost function. A continuous trajectory satisfying the continuous dynamics of the robot will be generated from the discrete path using proper control laws.**

Index Terms— **Temporal logic path planning, Robotics, Time constrained motion planning, Planning in dynamic environments.**

I. INTRODUCTION

Motion planning [1],[2] of a robot is mainly maneuvering the robot from its initial position and configuration to a final position and configuration while maintaining all physical and environmental constraints. This started with the focus on finding optimal trajectories to reach the goal while avoiding obstacles [3] and then evolved into generating plans or paths for complex planning objectives in dynamic or even more complex environments [4],[5]. Planning problems in dynamic or uncertain environments were approached mainly by velocity tuning method [6]. Studies have been also done on the complexities of planning problems in dynamic environments [7], [8]. Though these techniques along with the theories of traditional optimal control with artificial potential functions [3], [9] or cell decomposition or sampling based methods [2] served as promising approaches for robotic path planning, they failed to address problems with multiple goals or a particular sequence of goals.

Researchers have come up with novel formulations and efficient computational approaches to mathematically formulate specifications such as motion sequencing, synchronization etc. Temporal logics such as linear temporal logic (LTL), computational tree logic (CTL), developed for model checking, have been widely accepted by the robotics community for the purpose of motion planning [10], [11]. Development of sophisticated model checking tools such as SPIN [12] and NuSMV [13] made it easier to generate discrete robot paths satisfying the objectives or to produce counter examples proving that the objectives are not achievable. Though temporal logic can efficiently overcome the previous issues with motion sequencing, planning with multiple goals, however, they cannot mathematically formulate a planning problem with time bounded objectives.

In robotics and related fields, there is a class of problems where timing constraints are common, such as simple as "go to position 1 by time t_1 and eventually go to position 2 ". Even in a simple navigation problem, if the environment is dynamic, we have to face timing constraints which cannot be handled by LTL. Planning in dynamic environments has been done in heuristic ways which do not necessarily give the optimal solution [6], [8]. Planning with time bounded objectives are hard because the discrete path has to be generated in such a way that one can find an equivalent continuous path respecting all the constraints in the dynamics of the robot and simultaneously satisfying the specifications. [14] considered planning in dynamic environments for time bounded temporal logic specifications, however, the dynamics of the robot were modeled using probabilistic Markov models. Mixed Integer Linear Programming (MILP) approaches have been also considered to solve planning problems by formulating the problems as mixed integer linear optimization problems [15], [16], [17].

Metric Temporal Logic (MTL) deals with model checking under timing constraints. The complexity of MTL model checking is undecided and MITL (Metric Interval Temporal logic), a subset of MTL, has the complexity of EXPSPACE-complete [18], whereas LTL is PSPACE-complete [18]. Signal Temporal Logic (STL), similar to MTL also performs model checking with timing constraints [19], [20]. Temporal logic has been widely used by the robotics community [10], [11], [14] to solve problems with complex tasks and that motivates us to use LTL for the planning problems with time constraints. Based on the facts that LTL is computationally less expensive and that there is good availability of tools to check LTL specifications, the goal in this work will be to translate a bounded time high level specification to a purely LTL specification.

So far the temporal logic planning problems, which generally do not include any explicit timing specification, have been solved in the workspace/configurationspace of the robot

Research partially supported by US Air Force Office of Scientific Research MURI grant FA9550-09-1-0538, by DARPA through ARO grant W911NF1410384, by National Science Foundation (NSF) grant CNS-1035655, and by National Institute of Standards and Technology (NIST) grant 70NANB11H148.

The authors are with the Institute of Systems Research and the Department of Electrical and Computer Engineering, University of Maryland College Park, MD, USA. {dmaity, baras}@umd.edu

[10], [11], [14]. We will introduce time explicitly along with the workspace and plan in a higher dimensional space. Hereafter this higher dimensional space will be called *spacetime*. Any discrete trajectory generated in spacetime will be forced to meet the explicit timing constraints.

In this paper, we will consider completing a task with multiple subtasks and some subtasks have to be finished within certain time bound while avoiding the moving and static obstacles in a dynamic environment. We are also interested in minimizing some cost function along the way of task completion. For this work, we consider the cost function to be the total time to complete the work. One can consider any cost function with the same framework that we are going to propose (as pointed out in section V-A after remark 5.7). To solve the problem, we will borrow some bounded time temporal operators from MTL and then translate them into usual LTL operators on spacetime. To accomplish the goals, we propose a framework to extend the LTL to incorporate both finite and infinite (e.g. periodic surveillance etc.) duration tasks. A method to synthesize the suitable control laws is proposed to steer the robot while obeying all constraints. An automata theoretic approach [21], [22] is adopted to check whether the problem specifications can be met. Finally, we are also interested in finding control laws that will generate a continuous trajectory which is equivalent to the discrete path generated by the automaton.

II. PROBLEM FORMULATION

We consider a robot whose dynamics are given by (1).

$$\dot{x} = f(t, x, u), \;\; x(0) \in \mathcal{X}_0, \;\; x(t) \in \mathcal{X}, \;\; u(t) \in \mathcal{U} \qquad (1)$$

$x(t)$ is the position of the robot at time $t \geq t_0$, $\mathcal{X}_0 \subseteq \mathcal{X}$ is a compact set that represents the set of possible initial positions of the robot. The goal of this work is to find a control law $u(t) \in \mathcal{U}$ so that the trajectory generated by (1) follows some time specific high level requirements. We consider the presence of static as well as time varying objects within the environment where the robot stays for any time $t(\geq t_0)$. The environment is modeled as time varying and the dynamic properties of the environment can be used, for example, to describe the presence of moving obstacles or to describe the state of a door being open or closed at time t.

Let $\Pi = \{\pi_1, \pi_2, \cdots \pi_n\}$ be the set of atomic propositions which labels $\mathcal{X}$ as a collection of rooms, doors, free space, obstacles etc. As the environment is dynamic, there could be a moving body in some part of the free space making that part to be treated as obstacle for that time. The occupied place becomes free space as soon as the body moves to a new place. Thus, labeling the environment is time dependent. We define a map, F, to label the time varying environment.

$$F : \mathcal{X} \times \mathcal{I} \rightarrow 2^{\Pi} \qquad (2)$$

where $\mathcal{I} = \{[a, b) | \;\; b \geq a \geq 0\}$ and throughout the paper 2^{Ω} denotes the power set of a set Ω. We also define $\Pi_{\mathcal{I}} = \{\pi_{k,\mathbf{I}} | \;\; \pi_k \in \Pi, \mathbf{I} \in \mathcal{I}\}$ and a mapping $F_{\mathcal{I}} : \mathcal{X} \rightarrow 2^{\Pi_{\mathcal{I}}}$ s.t. $F_{\mathcal{I}}(x) = \pi_{k,\mathbf{I}}$ iff $F(x, \mathbf{I}) = \pi_k$.

The general problem we are interested in solving is:

Problem 2.1: *Given a workspace, a bounded time high level task (ϕ) and a cost function, find suitable control law ($u(t)$) such that the robot with dynamics (1) complete the given task while avoiding collision with all moving and static obstacles in the environment and minimizes the cost function.*

Given any environment, one can approximate and decompose it in polygonal cells [1], [2], [3] and obtain a cellularly decomposed environment similar to one in Fig. 1. Cellular decomposition of the workspace is a well studied problem and the interested readers are directed to [1], [2, Section 6.3] and the references therein. For the rest of the paper we will consider our planning problem in the workspace shown in Fig. 2 where the spacetime is represented for the planning problem. The blue segments in Fig. 2 represent walls, the red segments represent doors and the black continuous curve is the trajectory of the moving obstacle. The doors may be closed for certain time period or open otherwise as shown in Fig. 2 with the black surface. The 2D projection of this spacetime on the workspace is given in Fig. 1. The initial position of the robot is I (the yellow cell). Only to illustrate the problem clearly, 2D workspace is considered and time is represented as the third dimension; otherwise, the same framework works for 3D workspace as well. We consider the following example problem to illustrate our method.

Example 2.2: *Starting from I, visit R_3 within the time interval $\mathbf{I}_1$, visit R_4 within time interval $\mathbf{I}_2$; before visiting R_3 or R_4, robot must visit R_2. Eventually visit R_1 and R_5, and complete the whole task in the least time.*

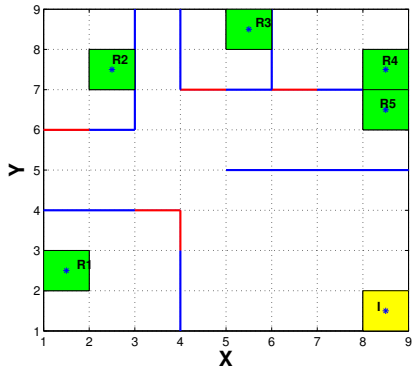

Fig. 1. Rectangular decomposition on the workspace of the robot (numbers on the X and Y axes are only to uniquely identify a cell)

Like Problem 2.1, Example 2.2 has two aspects: first, it requires some jobs to be done within specific time interval and second, it has an optimization aspect of completing the whole task in the least time. The task specification (without the optimization part) itself cannot be expressed using LTL logic due to the explicit timing specifications. The existing LTL grammar and semantics have to be extended to capture these types of explicit timing specifications, and for this purpose, we will define some notations and operators for LTL.

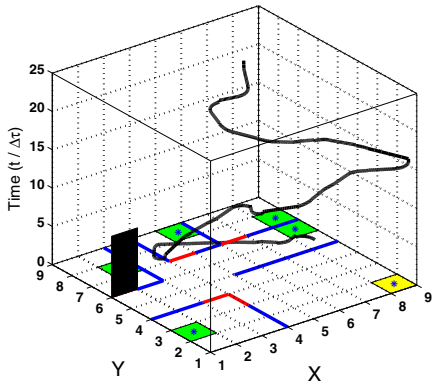

Fig. 2. Discretized spacetime with obstacles (The black surface represents the fact that the door to that region is closed for that time duration. The black curve is the continuous trajectory of a moving obstacle.)

III. PRELIMINARIES

This section provides some concepts and notations on LTL, *Finite Transition System* (FTS) and *Büchi Automata* (BA) which will be used throughout the paper. Let us denote the trajectory of the system (1) starting at t_0 as $x[t_0] = \{x(s) \mid s \geq t_0 \;\; \dot{x} = f(t,x,u),\; x(t_0) = x_0\}$.

Definition 3.1: *The syntax of LTL formulas are defined according to the following grammar rules:*

$$\phi ::= \top \mid \pi \mid \neg\phi \mid \phi \vee \phi \mid \phi \mathbf{U} \phi \mid \phi \mathbf{R} \phi$$

where $\pi \in \Pi$, $\top$ and $\bot(= \neg\top)$ are the Boolean constants `true` and `false` respectively. $\vee$ denotes the conjunction operator and $\neg$ denotes the negation operator. $\mathbf{U}$ and $\mathbf{R}$ symbolize the $Until$ and $Release$ operators respectively. Other temporal logic operators such as eventually ($\Diamond$), always ($\Box$) etc. can be represented using the grammar in definition 3.1.

Definition 3.2: *The semantics of any formula ϕ is recursively defined as:*
$x[t_0] \models \pi$ *iff* $\pi \in F(x(t_0), t_0)$
$x[t_0] \models \neg\pi$ *iff* $\pi \notin F(x(t_0), t_0)$
$x[t_0] \models \phi_1 \vee \phi_2$ *iff* $x[t_0] \models \phi_1$ *or* $x[t_0] \models \phi_2$
$x[t_0] \models \phi_1 \wedge \phi_2$ *iff* $x[t_0] \models \phi_1$ *and* $x[t_0] \models \phi_2$
$x[t_0] \models \phi_1 \mathbf{U} \phi_2$ *iff* $\exists s \geq t_0$ *s.t.* $x[s] \models \phi_2$ *and* $\forall\; t_0 \leq s' < s,\; x[s'] \models \phi_1$.
$x[t_0] \models \phi_1 \mathbf{R} \phi_2$ *iff* $\forall s \geq t_0\; x[s] \models \phi_2$ *or* $\exists s'$ *s.t.* $t_0 \leq s' < s,\; x[s'] \models \phi_1$.

More details on LTL grammar and semantics can be found in [23].

Definition 3.3: *A Finite Transition System (FTS) is a tuple $\mathcal{E} = \{Q, Q_0, \Pi_{I_D}, \mathcal{A}, TS, \rightarrow_{\mathcal{E}}, h_{\mathcal{E}}\}$, where:*
Q is a set of states.
$Q_0 \subseteq Q$ is the set of possible initial states.
Π_{I_D} is the set of atomic propositions.
$\mathcal{A}$ is the set of all actions or policies.
$TS : Q \times \mathcal{A} \rightarrow Q$ is a mapping that dictates the transition from one state to another by applying some action.
$\rightarrow_{\mathcal{E}} \subseteq Q \times Q$ captures the transitional relationship between the states. $(Q_i, Q_j) \in \rightarrow_{\mathcal{E}}$ iff $\exists \alpha \in \mathcal{A}$ s.t. $Q_j = TS(Q_i, \alpha)$.
$h_{\mathcal{E}} : Q \rightarrow 2^{\Pi_{I_D}}$ is the map which assigns atomic propositions to the states where those propositions are satisfied.

Definition 3.4: *A Büchi automaton (BA) is a tuple $\mathcal{B} = \{S_B, S_{0B}, \Sigma_B, \delta_B, F_B\}$ where:*
S_B is a set of states and S_{0B} is the initial state.
Σ_B is the set of input alphabets.
$\delta_B : S_B \times \Sigma_B \rightarrow 2^{S_B}$ is a transition relationship.
$F_B \subseteq S_B$ is a set of accepting states.

For each LTL formula ϕ, a corresponding Büchi automaton can be generated [21] that accepts the words which satisfy the specification ϕ. Generating a Büchi automaton from a given LTL formula is a well studied problem and the interested readers may confer [21].

IV. EXTENDED LINEAR TEMPORAL LOGIC

Two new operators $\mathbf{U_I}$ and $\mathbf{R_I}$ are introduced as follows:

Definition 4.1: *The extension of the LTL grammar is given by:* $\phi ::= \phi \mathbf{U_I} \phi \mid \phi \mathbf{R_I} \phi$
where $\mathbf{I} \in \mathcal{I}$. The semantics $\phi_1 \mathbf{U_I} \phi_2$ means $\exists s \in \mathbf{I}$ s.t. $x[s] \models \phi_2$ and $\forall\; t_0 \leq s' < s,\; x[s'] \models \phi_1$. It should be noted that the expression $\phi_1 \mathbf{U}_{[t_0,\infty)} \phi_2$ is equivalent to $\phi_1 \mathbf{U} \phi_2$. Once we have $\mathbf{U_I}$ and $\mathbf{R_I}$, we can always define other temporal operators such as $\Diamond_{\mathbf{I}}$, $\Box_{\mathbf{I}}$ etc. Thus, this grammar can model both finite and infinite duration tasks.
Like the LTL grammar and semantics in [23], similar grammar and semantics can be defined over the atomic propositions set $\Pi_{\mathcal{I}}$ as follows:

Definition 4.2: *The syntax of LTL formulas over $\Pi_{\mathcal{I}}$ are defined according to the following grammar rules:*
$\phi_I ::= \top_I \mid \pi_I \mid \neg\phi_I \mid \phi_I \vee \phi_I \mid \phi_I \mathbf{U} \phi_I \mid \phi_I \mathbf{R} \phi_I \mid (\phi_I)_I$
where $\top_I = \pi_I \vee \neg\pi_I$ and $\pi_I \in \Pi_{\mathcal{I}}$.

Definition 4.3: *The semantics of any formula ϕ_I is recursively defined as:*
$x[t_0] \models \pi_I$ *iff* $t_0 \in I$ *and* $\pi \in F(x(t_0), t_0)$
$x[t_0] \models \neg\pi_I$ *iff either* $t_0 \notin I$ *or* $\pi \notin F(x(t_0), t_0)$
$x[t_0] \models \phi_I$ *iff* $t_0 \in I$ *and* $x[t_0] \models \phi$
$x[t_0] \models \phi_{1,I_1} \vee \phi_{2,I_2}$ *iff* $x[t_0] \models \phi_{1,I_1}$ *or* $x[t_0] \models \phi_{2,I_2}$
$x[t_0] \models \phi_{1,I_1} \wedge \phi_{2,I_2}$ *iff* $x[t_0] \models \phi_{1,I_1}$ *and* $x[t_0] \models \phi_{2,I_2}$
$x[t_0] \models \phi_{1,I_1} \mathbf{U} \phi_{2,I_2}$ *iff* $\exists s \geq t_0$ *s.t.* $x[s] \models \phi_{2,I_2}$ *and* $\forall\; t_0 \leq s' < s,\; x[s'] \models \phi_{1,I_1}$.
$x[t_0] \models \phi_{1,I_1} \mathbf{R} \phi_{2,I_2}$ *iff* $\forall s \geq t_0\; x[s] \models \phi_{2,I_2}$ *or* $\exists s'$ *s.t.* $t_0 \leq s' < s,\; x[s'] \models \phi_{1,I_1}$.
$x[t_0] \models (\phi_{I_1})_{I_2}$ *iff* $x[t_0] \models \phi_{I_1 \cap I_2}$
It is easy to notice that setting I, I_1, I_2 as $[t_0, \infty)$ the usual LTL semantics is obtained. Now we will try to express the operator defined in definition (4.1) in terms of the grammar in definition (4.3). Let us denote the lower and upper bound of an interval, I, as $\check{I}$ and $\hat{I}$ respectively, i.e. $I = [\check{I}, \hat{I})$.

Proposition 4.4: *LTL formula $\phi_1 \mathbf{U}_I \phi_2$ is equivalent to $\phi_{1,[t_0,\hat{I})} \mathbf{U} \phi_{2,I}$*

Proof: Let $x[t_0] \models \phi_{1,[t_0,\hat{I})} \mathbf{U} \phi_{2,I}$ then $\exists s \geq t_0$ s.t. $x[s] \models \phi_{2,I}$ and $\forall\; t_0 \leq s' < s,\; x[s'] \models \phi_{1,[t_0,\hat{I})}$. Using definition 4.3, $x[s] \models \phi_{2,I}$ is rewritten as $s \in I$ and $x[s] \models \phi_2$. Since $s \in I$ and $s' \in [t_0, s)$, then always $s' \in [t_0, \hat{I})$. Therefore, $x[t_0] \models \phi_{1,[t_0,\hat{I})} \mathbf{U} \phi_{2,I}$ if $\exists s (\geq t_0) \in I$ s.t. $x[s] \models \phi_2$ and $\forall\; t_0 \leq s' < s,\; x[s'] \models \phi_1$ which is equivalent to say $x[t_0] \models \phi_1 \mathbf{U}_I \phi_2$. Similarly it can be proved that $x[t_0] \models \phi_1 \mathbf{U}_I \phi_2$ implies $x[t_0] \models \phi_{1,[t_0,\hat{I})} \mathbf{U} \phi_{2,I}$. ■

Remark 4.5: *LTL formula* $\phi_1 \mathbf{R}_I \phi_2$ *is equivalent to* $\phi_{1,I}\mathbf{R}\phi_{2,[t_0,\infty)}$.

Any bounded time high level specification can be represented by the usual and extended LTL grammars described in definitions (3.1) and (4.1) and can be converted to an equivalent formula of definition (4.2) which only contains the usual LTL operators. The advantage of having usual LTL operators is that we can easily translate the formula to a Büchi automaton which is an important and necessary step for model checking.

The explicit time dependent part of example 2.2 can be represented by the LTL formula $\phi = (\neg\mathcal{O}\mathbf{U}_{\mathbf{I}_1} R_3) \wedge (\neg\mathcal{O}\mathbf{U}_{\mathbf{I}_2} R_4)$; where $\mathcal{O}$ represents the set of obstacles (the walls, moving obstacles or the closed doors).

V. ROBOT MOTION PLANNING

In this section a discrete path that will satisfy the requirements of example 2.2 for the robot will be generated.

A. Generating Discrete Path

To proceed with the formal verification, the workspace is discretized. There exists several techniques for cell decomposition in polygonal environments [2], [3]. We will divide our workspace, $\mathcal{X}$, in rectangles. Let $Q = \{q_1, q_2, ..., q_m\}$ be a partition in the workspace. Let us define a map $T : \mathcal{X} \to Q$ to partition the continuous workspace $\mathcal{X}$. $\forall x, y \in \mathcal{X}$ and $I \in \mathcal{I}$, the map T has the following properties: a) if $T(x) = T(y)$, then $F(x, I) = F(y, I)$, and b) if $T(x) = T(y)$ and $x' \in q_j$ is reachable from $x \in q_i$, then $\exists y' \in q_j$ such that y' is reachable from y. The set Q_0 denotes the set of states where the robot can stay initially i.e. $\cup_{q_0 \in Q_0} T^{-1}(q_0) \subseteq \mathcal{X}_0$. Discretization on the set $\mathcal{I}$ is defined as $\mathcal{I}_D = \{[t_0 + n_1\Delta\tau, t_0 + n_2\Delta\tau) |\ n_2 > n_1 = 0, 1, \cdots\}$. Like the choice of cell size in cell decomposition, the choice of $\Delta\tau$ determines how accurately the dynamicity of the environment is captured.

Let us also define $\mathcal{Q} = \{(q, I_k) |\ q \in Q, I_k = (t_0 + k\Delta\tau, t_0 + (k+1)\Delta\tau] \in \mathcal{I}_D\}$ as the discretization of the continuous spacetime. The mapping $L : \mathcal{Q} \to 2^{\Pi_{\mathcal{I}_D}}$ is defined as $L(q, I_k) = \pi_I$ *iff* $F(x, I) = \pi$ s.t. $T(x) = q$ and $I_k \subseteq I$, $\forall I$, $I_k \in \mathcal{I}_D$. Let $\mathcal{A}$ denotes the set of actions the robot can take in this discretized spacetime. The set of actions available at any particular state $q \in \mathcal{Q}$ will be denoted by $\mathcal{A}_q\,(\subseteq \mathcal{A})$.

The discretized spacetime for example 2.2, given in Fig. 2, contains sixty four blocks (8×8) in each time layer. Fig. 3 illustrates the numbering scheme and transitional relations that have been adopted. $x,\ x+1,\ x+8$ and $x-64$ in Fig. 3 denote the numbers of the blocks. The arrows show the possible transition from one block to another. The letters (f, b, l, r, u) represent the action to be taken in order to perform that transition. The discretized spacetime can be represented by an FTS($\mathcal{E}$) similar to that given in definition (3.3).

Definition 5.1: *The equivalent FTS($\mathcal{E}$) of the discretized spacetime is given by a tuple* $\{\mathcal{Q}, \mathcal{Q}_0, \Pi_{I_D}, \mathcal{A}, TS, \to_{\mathcal{E}}, h_{\mathcal{E}}\}$ *where* $\mathcal{Q}$ *is the set of states.*

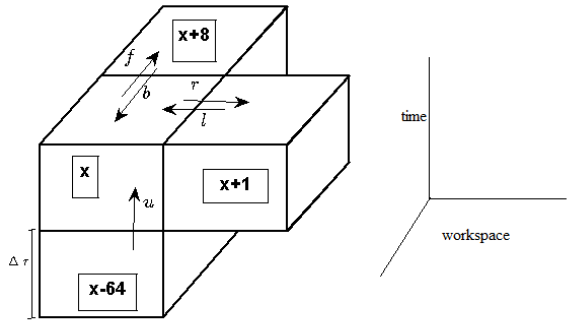

Fig. 3. Transitional relationship among the blocks in discretized spacetime.

$\mathcal{Q}_0 = \{(q_0, I_0) \mid q_0 \in Q_0\}$.

$\mathcal{A} = \{f, b, l, r, u\}$ *where* f, b, l, r, u *are the abbreviation of* $forward$, $backward$, $left$, $right$ *and* up *respectively.*

TS *denotes the possible transitions upon applying action* $\alpha \in \mathcal{A}$ *on a robot residing at block* q. *As the names suggest, applying* $\mathcal{A}_q = r$ *at a block* q *makes the robot to move to the block right to* q, *provided there is a block right to* q. *If there is no block right to* q, $r \notin \mathcal{A}_q$.

$\to_{\mathcal{E}} \subseteq \mathcal{Q} \times \mathcal{Q}$ *captures the topological relationship between the blocks.* $\big((q_i, I_{k_i}),\ (q_j, I_{k_j})\big) \in \to_{\mathcal{E}}$ *iff* $\big(k_i = k_j$ *and* q_i, q_j *share a common edge, i.e.* $(q_j, I_{k_j}) \in TS\big((q_i, I_{k_i}), \mathcal{A}_q\big)$ *with* $\mathcal{A}_q \in \{f, b, l, r\}\big)$ *or* $(q_i = q_j,\ k_i + 1 = k_j$ *i.e.* $\mathcal{A}_q = u)$.

$h_{\mathcal{E}} : \mathcal{Q} \to 2^{\Pi_{I_D}}$ *is the map same as* L.

The up action corresponds to movements in the time direction and since time is unidirectional, there is no $down$ action present in the FTS.

Let $p : \mathbb{N} \to \mathcal{Q}$ be a path for the robot in the discretized spacetime ($\mathcal{Q}$) with $p(0) \in \mathcal{Q}_0$ and $(p(i), p(i+1)) \in \to_{\mathcal{E}}$. The sequence of actions taken is denoted by $\alpha = \alpha_0\alpha_1 \cdots \alpha_n$ which satisfies the fact that $\forall i = 0, \cdots, n,\ p(i+1) = TS(p(i), \alpha_i)$.

Analogous to the continuous LTL formulas given in definitions (4.2) and (4.3), we can also have discrete LTL formulas and grammars, on the set of atomic propositions $\big(\Pi_{\mathcal{I}_D}\big)$, as given in definitions (5.2) and (5.3).

Definition 5.2: *The syntax of bounded time discrete LTL formulas are defined in the following grammar rules:*

$$\phi_{I_D} ::= \top_{I_D} \mid \pi_{I_D} \mid \neg\phi_{I_D} \mid \phi_{I_D} \vee \phi_{I_D} \mid \phi_{I_D}\mathbf{U}\phi_{I_D} \mid \phi_{I_D}\mathbf{R}\phi_{I_D} \mid \bigcirc \phi_{I_D} \mid (\phi_{I_D})_{I_D}$$

where $\pi_{I_D} \in \Pi_{\mathcal{I}_D}$, $\top_{I_D} = \pi_{I_D} \vee \neg\pi_{I_D}$ and $\bigcirc$ is the next operator.

We define two projection functions $pr_1 : p(i) \to Q$ and $pr_2 : p(i) \to \mathcal{I}_D$ on a path $p = p(0)p(1)\cdots p(l)$ in the discretized spacetime. Let $p(i) = (q_i, I_{k_i}) \in \mathcal{Q}$ then $pr_1(p(i)) = q_i$ and $pr_2(p(i)) = I_{k_i}$. We will use the short hand notation $p[i_0]$ to denote the portion of the path p that starts from $p(i_0)$ i.e. $p[i_0] = p(i_0)p(i_0+1)\cdots p(l)$ (with this notation, $p \equiv p[0]$).

Definition 5.3: *The semantics of any formula* ϕ_{I_D} *on a discrete path* p *is recursively defined as:*

$p \models \pi_{I_D}$ *iff* $pr_2(p(0)) \subseteq I_D$ *and* $\pi_{I_D} \in L(p(0))$

$p \models \neg\pi_{I_D}$ *iff either* $pr_2(p(0)) \nsubseteq I_D$ *or* $\pi_{I_D} \notin L(p(0))$

$p \models \phi_{I_D}$ *iff* $pr_2(p(0)) \subseteq I_D$ *and* $pr_1(p) \models \phi$.

$p \models \phi_{1,I_{D_1}} \vee \phi_{2,I_{D_2}}$ *iff* $p \models \phi_{1,I_{D_1}}$ *or* $p \models \phi_{2,I_{D_2}}$

$p \models \phi_{1,I_{D_1}} \wedge \phi_{2,I_{D_2}}$ *iff* $p \models \phi_{1,I_{D_1}}$ *and* $p \models \phi_{2,I_{D_2}}$

$p \models \bigcirc\phi_{I_D}$ *iff* $p[1] \models \phi_{I_D}$
$p \models \phi_{1,I_{D1}} \mathbf{U} \phi_{2,I_{D2}}$ *iff* $\exists i \geq 0$ *s.t.* $p[i] \models \phi_{2,I_{D2}}$ *and* $\forall 0 \leq j < i,\ p[j] \models \phi_{1,I_{D1}}$
$p \models \phi_{1,I_{D1}} \mathbf{R} \phi_{2,I_{D2}}$ *iff* $\forall i \geq 0\ p[i] \models \phi_{2,I_{D2}}$ *or* $\exists j$ *s.t.* $0 \leq j < i,\ p[j] \models \phi_{1,I_{D1}}$
$p \models (\phi_{I_{D1}})_{I_{D2}}$ *iff* $p \models \phi_{I_{D1} \cap I_{D2}}$.
The FTS($\mathcal{E}$) of the environment can be translated into an equivalent Büchi automaton as given in definition 5.4.

Definition 5.4: *The Büchi automaton ($\mathcal{E}'$) corresponding to the FTS($\mathcal{E}$) in definition (5.1) is a tuple* $\mathcal{E}' = \{\mathcal{Q}', q_e, 2^{\Pi_{I_D}}, \delta_{\mathcal{E}'}, F_{\mathcal{E}'}\}$ *where:*
$\mathcal{Q}' = \mathcal{Q} \cup q_e$ *for* $q_e \notin \mathcal{Q}$.
$\delta_{\mathcal{E}'} : \mathcal{Q}' \times 2^{\Pi_{I_D}} \rightarrow 2^{\mathcal{Q}'}$ *s.t.* $(q_i, I_{k_i}) \in \delta_{\mathcal{E}'}\big((q_j, I_{k_j}), \pi\big)$ *iff* $\big((q_i, I_{k_i}), (q_j, I_{k_j})\big) \in \rightarrow_{\mathcal{E}}$ *and* $\pi \in h_{\mathcal{E}}(q_j, I_{k_j})$, *and* $(q_0, I_{k_0}) \in \delta_{\mathcal{E}'}\big(q_e, \pi_0\big)$ *iff* $(q_0, I_{K_0}) \in \mathcal{Q}_0$ *and* $\pi_0 \in h_{\mathcal{E}}(q_0, I_{k_0})$
$F_{\mathcal{E}'}$ *is the set of accepting states.*

The aim is to find a path on $\mathcal{E}'$ that satisfies a given LTL formula ϕ_{I_D} (representing the time bounded task specification). That is to find the language, $\mathcal{L}$, which is accepted by both automata $\mathcal{B}_{\phi_{I_D}}$ and $\mathcal{E}'$. This can be done by constructing a product automaton $\mathcal{B}_{\phi_{I_D}} \times \mathcal{E}'$ whose language will be $\mathcal{L}(\mathcal{B}_{\phi_{I_D}}) \cap \mathcal{L}(\mathcal{E}')$.

Definition 5.5: *The product automaton* $\mathcal{P} = \{S_{\mathcal{P}}, S_{0\mathcal{P}}, 2^{\Pi_{I_D}}, \delta_{\mathcal{P}}, F_{\mathcal{P}}\}$ *where:*
$S_{\mathcal{P}} = \mathcal{Q}' \times S_{\mathcal{B}_\phi}$ *and* $S_{0\mathcal{P}} = \{(q_e, S_{0B_\phi})\}$.
$\delta_{\mathcal{P}} : S_{\mathcal{P}} \times 2^{\Pi_{I_D}} \rightarrow 2^{S_{\mathcal{P}}}$ *s.t.* $\big((q_i, I_{k_i}), S_n\big) \in \delta_{\mathcal{P}}\big(((q_j, I_{k_j}), S_m), \pi\big)$ *iff* $(q_i, I_{k_i}) \in \delta_{\mathcal{E}'}((q_j, I_{k_j}), \pi)$ *and* $S_n \in \delta_{B_\phi}(S_m, \pi)$
$F_{\mathcal{P}} = F_{\mathcal{E}'} \times F_{\mathcal{B}_\phi}$

By construction, the language of this product automaton is the common language of both automata $\mathcal{E}'$ and $\mathcal{B}_\phi$, i.e. $\mathcal{L}(\mathcal{P}) = \mathcal{L}(\mathcal{E}') \cap \mathcal{L}(\mathcal{B}_\phi)$. A run, $r : \mathbb{N} \rightarrow S_{\mathcal{P}}$ of $\mathcal{P}$, is a sequence of states which is obtained by applying an input trace ω; i.e. $r(0) \in S_{0\mathcal{P}}$ and $\forall i \geq 0,\ r(i+1) \in \delta_{\mathcal{P}}(r(i), \omega(i))$. An accepting run of an automaton contains at least one final state. More precisely, a run r of $\mathcal{P}$ over an infinite trace ω is accepting if and only if $iof(r) \cap F_{\mathcal{P}} \neq \emptyset$, where $iof(r)$ is the function that returns the set of states that are encountered infinitely often in the run r. If $\mathcal{L}(\mathcal{P}) \neq \emptyset$, there exists accepting run(s) for $\mathcal{P}$. An infinite length accepting run consists of two parts a prefix and a periodic suffix.

Remark 5.6: *Let* $pr : S_{\mathcal{P}} \rightarrow \mathcal{Q}'$ *be a projection function such that* $pr(q, s) = q$; *where* $q \in \mathcal{Q}'$ *and* $s \in S_{\mathcal{B}_\phi}$. *If* r *is an accepting run of* $\mathcal{P}$, p *is a path on* $\mathcal{E}$; *where* $p(i) = pr(r(i))$.

Let us define $\mathcal{R}_{\mathcal{P}} = \{r \mid r$ is an accepting run of $\mathcal{P}\}$. Note that the cardinality of $\mathcal{R}_{\mathcal{P}}$ can be more than one.

Remark 5.7: *Problem 2.2 is feasible iff* $\mathcal{R}_{\mathcal{P}} \neq \emptyset$.

We will convert this product automaton $\mathcal{P}$ to a directed weighted graph $\mathcal{G}(V, E, W)$. V is the set of nodes and E, the set of edges, is a binary relation on V, and W is the set of weights associated with the edges. Through this conversion process, the states of the automaton become the nodes of the graph and the transitional relation ($\delta_{\mathcal{P}}$) defines the set of the edges (E). Thus, $S_{\mathcal{P}}$ and V are basically the same set and hence can be related using a bijective mapping Z(say). $(V_i, V_j) \triangleq E_{ij} \in E$ *iff* $\exists \pi_{I_D}$ s.t. $Z^{-1}(V_i) \in \delta_{\mathcal{P}}(Z^{-1}(V_j), \pi_{I_D})$. Depending on the cost function to be minimized, the weights on the edges can be constructed accordingly. Since each edge represents a transition from one node (position and time) to another with the proper application of an action, the position, time and action information are available at each edge. Any cost that is a function of position, time and action can be calculated easily for the edges and can be put as a weight on the edge. For example 2.2 the weight w_{ij} associated with the edge E_{ij} is defined to be $w_{ij} = \big(pr_2 \circ pr \circ Z^{-1}(V_j) - pr_2 \circ pr \circ Z^{-1}(V_i)\big)/\Delta\tau$ [where $f \circ g(h) = f(g(h))$]. From the topological and transitional relation given in definition 5.1, one can easily check that $w_{ij} \in \{0, 1\}$, and $w_{ij} = 1$ only when $\mathcal{A}_q = u$. Let us denote $V_0 = \{v \mid Z^{-1}(v) \in S_{0\mathcal{P}}\}$ and $V_{\mathcal{P}} = \{v \mid Z^{-1}(v) \in F_{\mathcal{P}}\}$. Let $P_{\mathcal{G}}$ be the set of all paths on the graph $\mathcal{G}$ that start on a node $v \in V_0$ and end on a node $v \in V_{\mathcal{P}}$. $p_g : \mathbb{N} \rightarrow V$ be a path on $\mathcal{G}$ such that $p_g(0) \in V_0$, $(p_g(i), p_g(i+1)) \in E$ and $p_g(l) \in V_{\mathcal{P}}$ (l is the length of the path). Cost associated with the link $p_g(i) \rightarrow p_g(i+1)$ is $C_{i,i+1} = \big(pr_2 \circ pr \circ Z^{-1}(p_g(i+1)) - pr_2 \circ pr \circ Z^{-1}(p_g(i))\big)/\Delta\tau$. Therefore, a path with the least cumulative link cost will complete the given task in the least time. In other words, the solution of example 2.2 is the solution of the optimization problem 5.8 which can be solved efficiently by using a suitable graph search algorithm.

Problem 5.8:

$$\begin{aligned} \min_{p_g} \quad & \Sigma_{i=0}^{l-1} C_{i,i+1} \\ \text{subject to} \quad & p_g \in P_{\mathcal{G}} \end{aligned}$$

Due to the equivalence between $\mathcal{P}$ and $\mathcal{G}$, an equivalent run r on $\mathcal{P}$ can be obtained for the path $p_g \in P_{\mathcal{G}}$. Let p be the projection of r on $\mathcal{Q}$ i.e. $p(i) = pr(r(i))$. The time and space sequence of p_g are $pr_2 \circ pr(r(i))$ and $pr_1 \circ pr(r(i))$ respectively. It is possible to find a path $p_g \in P_{\mathcal{G}}$ such that $\forall i = 0, 1, \cdots l$, $pr_2 \circ pr(r(i)) = I_0$. Such a path requires the whole task to be done in $\Delta\tau$ amount of time and hence the cost for that path will be zero. This discrepancy arises since the dynamics or the physical constraints of the robot have not been considered in the formulation of problem 5.8. Practically, it may not be plausible to complete the whole task in that small time. The reachability property of the robot has to be incorporated in this planning problem. We will consider the reachable set from a cell $q_i \in Q$ to be the set of cells that can be reached from any point in q_i within $\Delta\tau$ amount of time. For this paper, we consider $\Delta\tau$ to be the smallest time s.t. for every $x \in B_0$ in Fig. (4), $\forall i \in \{1, 2, \cdots, 12\}$, $\exists y \in B_i$ which is reachable from x within $\Delta\tau$ time. This particular reachability property enforces the requirement that between two successive applications of action u, there can be at most two actions from the set $\{f, b, l, r\}$. Let us define a new atomic proposition ξ_k such that $p[i] \models \xi_k$ *iff* $pr_2(p(i)) = I_k$. $\zeta_k = \Box\big((\xi_k \wedge \bigcirc\xi_k \wedge \bigcirc\bigcirc\xi_k) \Rightarrow \bigcirc\bigcirc\bigcirc(\neg\xi_k)\big)$ ensures no more than two transitions within the $k-th$ time layer. Thus, the equivalent LTL specification of the reachability constraint is given by $\phi_{reach} = \underset{\forall k=0,1,\cdots}{\wedge} \zeta_k$. If ϕ_1 is the LTL representation of example 2.2, $\phi = \phi_1 \wedge \phi_{reach}$ ensures

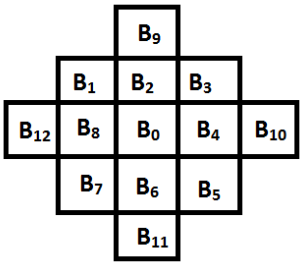

Fig. 4. Set of reachable cells within time $\Delta\tau$ from cell B_0.

that any path satisfying ϕ will solve example 2.2 with the reachability constraint.

B. From Discrete Path to Continuous Trajectory

The solution of problem 5.8 is a discrete path, on the discretized spacetime, that satisfies the temporal specification and minimizes the given cost function. The goal is to find some input $u(t) \in \mathcal{U}$ for all $t \geq t_0$ so that the robot can follow the discrete trajectory. In this paper, the unicycle robot dynamics given in (3) are considered.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos(\theta) & 0 \\ \sin(\theta) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{3}$$

x and y are the position coordinates and θ is the orientation or heading angle. v and ω are the control inputs. The discrete path is essentially a sequence of blocks that has to be visited. While transitioning from one block to another, the robot must be confined inside those blocks. This imposes some extra conditions that have to be taken care of while generating the continuous trajectory. A state feedback technique is used to achieve this. Let the robot is currently at block q_1 with positions and orientation given by $[x_1, y_1, \theta_1]$ and it has to go to block q_2. For this transition we consider the final position, $[x_2, y_2]$, to be the center of the block q_2 and θ_2 is determined by the following rule:

$$\theta_2 = \begin{cases} 0^o & \text{if } |\Delta x| \geq |\Delta y| \text{ and } \Delta x \geq 0 \\ 180^o & \text{if } |\Delta x| \geq |\Delta y| \text{ and } \Delta x < 0 \\ 90^0 & \text{if } |\Delta x| < |\Delta y| \text{ and } \Delta y \geq 0 \\ -90^0 & \text{if } |\Delta x| < |\Delta y| \text{ and } \Delta y < 0 \end{cases}$$

where $\Delta x = x_2 - x_1$ and $\Delta y = y_2 - y_1$. With these initial and final conditions, the robot starts moving and let at time t, it is at $[x(t), y(t), \theta(t)]$. At this time t, angular velocity, $\omega(t)$, is along $\theta_2 - \theta(t)$ and $|\omega(t)| \leq \omega_{\max}$. Linear velocity, $v(t)$, is chosen to be $min(dist(x(t), \delta X), dist(y(t), \delta Y), v_{\max})$. δX is the set of boundary points of the blocks q_1 and q_2 along the x-axis; similar definition for δY along y-axis as well. $dist(a, A) = inf\{\|x - a\|_2 \mid x \in A\}$ is a function that gives the minimum distance from a to the set A. These choices of inputs for the dynamics (3) ensure that the continuous trajectory never leaves the blocks q_1 and q_2. The continuous trajectory being confined within blocks q_1 and q_2, and the properties of the map T ensure that the continuous trajectory satisfies the temporal specification.

VI. SIMULATIONS

We consider example 2.2 in a dynamic environment to test our approach. Noting that the choice of t_0 does not matter under this setting, we consider $t_0 = 0$. The dynamic environment contains a moving obstacle and the door to region R_2 is closed for the time interval $[0, 8\Delta\tau]$. The dynamic behaviors are incorporated in formulating the FTS ($\mathcal{E}$) and the Büchi automata ($\mathcal{E}'$). The reachability of the robot is considered to be the same as what given in Fig. 4. I_1 and I_2 in example 2.2 are considered to be $[14\Delta\tau, 17\Delta\tau]$ and $[16\Delta\tau, 21\Delta\tau]$ respectively. The discrete path is obtained by solving problem 5.8 using Dijkstra's algorithm [24]. The continuous path is generated by the controller described in section V-B with $\omega_{\max} = \pi/4\ rad/s$, $v_{\max} = 0.4\ m/s$ and the cellsize (Fig. 5) is $1 \times 1\ m^2$. The initial configuration of the robot is assumed to be $[x_0, y_0, \theta_0] = [8.5, 1.5, 135^o]$. The projected (in workspace) trajectories, both discrete and continuous, are shown in Fig. 5. The continuous trajectory in spacetime is shown in Fig. 6. For this example we considered total planning time upto $25\Delta\tau$ and hence the FTS $\mathcal{E}$ has 1600 states and the automata for the LTL specification consists of 128 states.

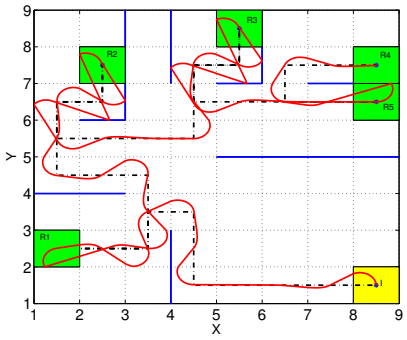

Fig. 5. Projected continuous and discrete trajectories. (Dashed black line is the discrete path and the red curve is the corresponding continuous trajectory.)

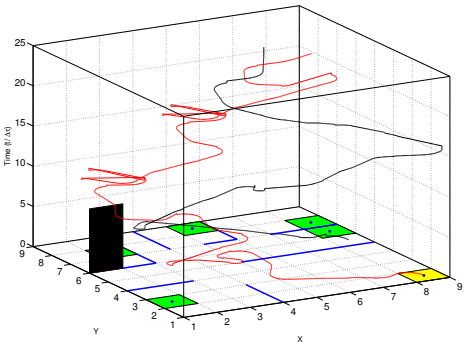

Fig. 6. Continuous trajectory in spacetime. (Red curve is the trajectory generated for the robot and the black curve is the obstacle trajectory.)

VII. CONCLUSIONS

In this paper, we presented a method to generate continuous trajectories of a robot in a dynamic environment subject to some bounded time temporal logic specifications and optimization objective. We extended the rules of LTL

to incorporate timing constraints and we also projected the planning problem into a higher dimensional space to formally generate a path. minimize the task completion time but one can consider other objectives under the same framework. One possible way to find control inputs, for the continuous system (3), is proposed to make the robot follow the discrete optimal path obtained by solving problem 5.8. In this work we considered task of finite duration, however, task with infinite duration can also be formulated using the proposed framework. The generated continuous trajectory has high curvature at some points, solely because no constraint on the curvature of the trajectory was imposed while generating the continuous trajectory.

As a future direction, one might consider this framework for multi-robot planning probelms or planning in a dynamic environment to incrementally improve the solution while satisfying the timing constraints.

REFERENCES

[1] J.-C. Latombe, *Robot Motion Planning*. Boston, MA: Kluwer Academic Publishers, 1991.

[2] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006. [Available at http://msl.cs.uiuc.edu/planning/]

[3] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, 2005.

[4] R. Sharma, "Locally efficient path planning in an uncertain, dynamic environment using a probabilistic model," *IEEE Transactions on Robotics and Automation,* vol. 8, no. 1, pp. 105-110, 1992.

[5] S. LaValle and R. Sharma, "On Motion planning in changing, partially-predictable environments," *International Journal of robotics Research*, vol. 16, no. 6, pp. 775-805, 1997.

[6] K. Kant, and S. W. Zucker, "Toward efficient trajectory planning: The path-velocity decomposition, "*The International Journal of Robotics Research* vol. 5, no. 3, pp. 72-89, 1986.

[7] J. Reif, and M. Sharir, "Motion planning in the presence of moving obstacles, "*Journal of the ACM (JACM)*, vol. 41, no. 4, 764-790, 1994.

[8] M. Erdmann, and T. Lozano-Perez, "On multiple moving objects,"*Algorithmica*, vol. 2, no. 1-4, pp. 477-521, 1987.

[9] W. Xi, X. Tan and J. S. Baras, "A Hybrid Scheme for Distributed Control of Autonomous Swarms, " In Proceeding of *2005 American Control Conference*, Portland, OR, USA, June 8-10, 2005.

[10] 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.

[11] A Ulusoy, S. L. Smith, X. C. Ding and C. Belta, "Robust multi-robot optimal path planning with temporal logic constraints," In Procceding of *IEEE International Conference on Robotics and Automation*, RiverCentre, Saint Paul, Minnesota, USA, May 14-18, 2012.

[12] G. Holzmann, "The model checker SPIN," *IEEE Transaction on Software Engineering*, vol. 25, no. 5, pp. 279-295, 1997.

[13] A. Cimatti, E. M. Clarke, E. Giunchiglia, F Guinchiglia, M. Pistore, M. roveri, R Sebastiani and A Tacchella, "NuSMV 2: An openSource Tool for Symbolic Model Checking,"In Proceeding of *International conference on Computer-Aided Verification*, Copenhagen, Denmark, July 27-31, 2002.

[14] A. I. Medina Ayala, S. B. Anderson and C. Belta, "Probabilistic control from time-bounded temporal logic specifications in dynamic environments," In the Proceeding of *IEEE International Conference on Robotics and Automation*, RiverCentre, Saint Paul, Minnesota, USA May 14-18, 2012.

[15] A. Richards, and J. P. How, "Aircraft trajectory planning with collision avoidance using mixed integer linear programming, "*In Proceedings of 2002 American Control Conference*, vol. 3, pp. 1936-1941, 2002.

[16] E. M. Wolff, U. Topcu, and R. M. Murray, "Automaton-guided controller synthesis for nonlinear systems with temporal logic, " *In Intelligent Robots and Systems (IROS), IEEE/RSJ International Conference on*, pp. 4332-4339, 2013.

[17] Karaman, Sertac and Frazzoli, Emilio, "Vehicle routing problem with metric temporal logic specifications, "*Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, pp. 3953-3958, 2008

[18] J. Ouaknine and J. Worrell, "Some recent results in metric temporal logic," In *Formal Modeling and Analysis of Timed Systems*, pp. 1-13, Springer Berlin Heilderberg, 2008.

[19] A. Donz, O. Maler, E. Bartocci, D. Nickovic, R. Grosu, and S. Smolka, "On temporal logic and signal processing, " *In Automated Technology for Verification and Analysis*, pp. 92-106, Springer Berlin Heidelberg.

[20] A. Donz, and O. Maler, "Robust satisfaction of temporal logic over real-valued signals, ", Springer Berlin Heidelberg, 2010.

[21] E. M. Clarke, O. Grumberg and D. Peled, *Model checking*. MIT Press, 1999.

[22] M. Y. Vardi and P Wolper, "An automata-theoretic approach to automatic program verification," *Logic in Computer Science*, pp. 322-331, 1986.

[23] C. Baier and J. P. Katoen, *Principles of Model Checking*, vol. 26202649, Cambridge: MIT Press, 2008

[24] S. Skiena, "Dijkstra's Algorithm," *Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica, Reading, MA: Addison-Wesley*, pp. 225-227, 1990.