Robustness-Based Synthesis for Time Window Temporal Logic Specifications via Mixed-Integer Linear Programming

Philip Smith[1,*], Ahmad Ahmad[1,*], Kevin Leahy[1]

***Abstract*— Time Window Temporal Logic (TWTL) is a rich specification language for cyber-physical systems that can compactly express sequential tasks with explicit timing constraints. In this paper, we consider the problem of synthesizing control inputs for discrete-time linear systems subject to TWTL task specifications. Building on the quantitative semantics (robustness) recently introduced for TWTL in [1], we encode the robust satisfaction of a TWTL formula as a set of Mixed-Integer Linear constraints and pose synthesis as a Mixed Integer Linear Program (MILP) that maximizes the robustness degree. We prove that any feasible solution with positive objective value guarantees Boolean satisfaction of the specification. We address two synthesis settings: an *open-loop* formulation that optimizes the full control sequence from the initial state, and a *closed-loop* receding-horizon Model Predictive Controller (MPC) formulation that re-solves the MILP at each step using the current measured state. A key feature of our MPC formulation is a *task-adaptive horizon* that exploits the TWTL Deterministic Finite Automaton (DFA) to determine the active sub-task at each step, limiting the prediction horizon to the remaining window of the current task rather than the full formula horizon, this makes each re-solve significantly cheaper than the initial open-loop solve.**

I. Introduction

Robot tasks often use spatio-temporal constraints: a surveillance drone which must dwell for a duration in multiple regions, or a multi-robot team which must complete sub-tasks in sequence while respecting timing windows. Temporal logics (TLs) [2], [3] address this by providing a language for stating what a system must accomplish over time. As such, they have become a standard tool for encoding high-level task specifications for cyber-physical systems. Signal Temporal Logic (STL) [4] and Metric Temporal Logic (MTL) [5] are widely used concrete-time logics. Time Window Temporal Logic (TWTL) [6] was introduced more recently as an alternative that (i) expresses sequential tasks compactly through a dedicated concatenation operator, and (ii) admits automata translations whose complexity is *independent* of the formula time bounds, making it particularly attractive for automata-based planning and synthesis [7].

Satisfying a TL specification requires synthesizing a control policy [8]. Optimization-based synthesis from TL specifications is enabled by a robustness measure quantifies Boolean satisfaction as a real value. Such a measure allows synthesis to optimize satisfaction margin, such as clearance of obstacles, alongside other optimization concerns like path length or battery life, and lends itself naturally to an MPC formulation. Optimizing robustness in this way allows plans to survive in a real, noisy world. For STL, the classical robustness [9] has been encoded as Mixed-Integer Linear constraints by Raman et al. [10], who also propose a receding-horizon MPC framework where the MILP is re-solved at each step with a *fixed* horizon equal to the formula time bound, with the executed history pinned as equality constraints. Until recently, TWTL lacked a robustness measure. In [1], we introduced two sound quantitative semantics for TWTL— a $\min$-$\max$-based robustness (ρ) and an Arithmetic Geometric Mean (AGM) robustness [11]— together with runtime monitors for partial runs.

A robustness measure only scores a given run; it does not produce one. What has been missing for TWTL is a synthesis procedure to optimize for ρ. In this paper, we leverage ρ from [1] to develop the first MILP synthesis procedure for TWTL.

Contributions. 1) We encode robustness as a recursive big-M MILP encoding and prove correctness (Sec. IV, Theorem 4.1); 2) We formulate an open-loop controller (Problem 3.1) and a closed-loop MPC controller (Problem 3.2) with a *task-adaptive horizon* that exploits the TWTL DFA to bound the prediction horizon rather than use the full formula horizon T (Sec. V); 3) We propose a warm-start strategy for the MPC re-solves that reduces the per-step online cost to linear in the number of TWTL sub-tasks for state updates and $O(1)$ for horizon shrinkage, plus the MILP re-solve on a progressively smaller problem (Sec. V-C); and 4) We demonstrate the incresed efficiency of direct TWTL enconding by comparing against TWTL translated to STL (Sec. VI-C)

Related work. While other works have previously used TWTL for monitoring [12], [13], or as a basis for sampling-based planning [6], [7], [14], we perform synthesis directly using the specification and a model of system dynamics (Sec. II-A). Unlike sampling-based planners, which offer probabilistic completeness but no hard guarantees on robustness degree, our MILP formulation deterministically maximizes robustness and enables closed-loop MPC with disturbance rejection. The only prior work to use MILP with TWTL is [15], which encodes automaton transitions (not robustness) as the MILP objective; ours is the first robustness-maximizing MILP for TWTL. Our encoding is directly analogous to [10] for STL, with TWTL's hold and within operators playing the roles of STL's always and eventually. This approach was also used successfully with another temporal logic (wSTL) in [16]. Our MPC horizon strategy differs from Raman et al.'s [10] fixed-H approach by exploiting TWTL's automata

* Equal contribution, [1] Department of Robotics Engineering, Worcester Polytechnic Institute, {psmith3,aahmad4,kleahy}@wpi.edu

structure to obtain a task-adaptive horizon, a mechanism that is not available for STL without an automaton.

II. Preliminaries

A. Dynamical System

Consider a discrete-time linear time-invariant (LTI) system:

$$\begin{aligned} x_{t+1} &= Ax_t + Bu_t, \\ o_t &= Cx_t, \end{aligned} \tag{1}$$

where $x_t \in \mathcal{X} \subseteq \mathbb{R}^n$ is the state, $u_t \in \mathcal{U} \subseteq \mathbb{R}^m$ is the control input, and $o_t \in \mathcal{O} \subseteq \mathbb{R}^p$ is the observable output; $\mathcal{X}$ and $\mathcal{U}$ are convex polytopes. $\mathbf{o}_{t_1,t_2} := o_{t_1} o_{t_1+1} \cdots o_{t_2}$ denotes the output word over $[t_1, t_2]$. We fix a finite set $\Pi = \{\pi_1, \ldots, \pi_k\}$ of atomic propositions. Each $\pi_A \in \Pi$ is identified with a satisfaction region $A \subseteq \mathcal{O}$. An observation o_t satisfies π_A iff $o_t \in A$. In [1] a general predicate function h is used to represent these atomic propositions, which we spoecialize to an affine form via Assumtion 2.1

Assumption 2.1: All atomic propositions $\pi_A \in \Pi$ have affine predicate functions $h(o) = \mathbf{c}_A^T o - b_A$, where $\mathbf{c}_A \in \mathbb{R}^p$ and $b_A, \sigma \in \mathbb{R}$, so that $A = \{o \mid h(o) > \sigma\}$. Without loss of generality we absorb σ into b_A, so that $h(o_t) = \mathbf{c}_A^T C x_t - b_A$ throughout.

B. Time Window Temporal Logic

TWTL syntax is defined, inductively, as follows [6].

$$\phi \ ::= \ H^d s \mid H^d \neg s \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \neg\phi \mid \phi_1 \cdot \phi_2 \mid [\phi]^{[a,b]} \tag{2}$$

where s is an atomic proposition with affine predicate function by Assumption 2.1, H^d is the *hold* operator, $\cdot$ is the *concatenation* operator, and $[\phi]^{[a,b]}$ is the *within* operator, with $d, a, b \in \mathbb{Z}_{\geq 0}$ and $b \geq a$. From a given TWTL formula, the time horizion $T = ||\phi|| \in \mathbb{N}$ is computed, recursively, as follows [1].

$$||\phi|| := \begin{cases} \max(||\phi_1||, ||\phi_2||); & \text{if } \phi \in \{\phi_1 \wedge \phi_2, \phi_1 \vee \phi_2\} \\ ||\phi_1||; & \text{if } \phi = \neg\phi_1 \\ ||\phi_1|| + ||\phi_2|| + \Delta t; & \text{if } \phi = \phi_1 \cdot \phi_2 \\ d\Delta t; & \text{if } \phi = H^d \pi_A \\ b; & \text{if } \phi = [\phi_1]^{[a,b]} \end{cases} \tag{3}$$

The hold operator $H_d s$ specifies that $s \in AP$ should be repeated for d time units. The semantics of $H_d \neg s$ is defined similarly, but for d time units only symbols from $AP \setminus \{s\}$ should appear. The word $\mathbf{o}_{t_1,t_2}$ satisfies $\phi_1 \wedge \phi_2$, $\phi_1 \vee \phi_2$, or $\neg\phi$ if $\mathbf{o}_{t_1,t_2}$ satisfies both formulae, at least one formula, or does not satisfy the formula, respectively. The within operator $[\phi]_{[a,b]}$ bounds the satisfaction of ϕ to the time window $[a, b]$. The concatenation operator $\phi_1 \cdot \phi_2$ specifies that first ϕ_1 must be satisfied, and then immediately ϕ_2 must be satisfied [6].

C. TWTL DFA

Every TWTL formula ϕ translates to a DFA [6] $\mathcal{A}_\phi = (Q, \Sigma, \delta, q_0, F)$, where Q is the state set, $\Sigma = 2^\Pi$, $\delta : Q \times \Sigma \to Q$, q_0 is the initial state, and $F \subseteq Q$ is the accepting set. A critical property of TWTL is that $|Q|$ is *independent* of the magnitude of the time bounds a, b, d [6]. The DFA state $q_{t+1} = \delta(q_t, l(o_t))$ is updated at each step via a single table lookup ($O(1)$), where $l : \mathbb{R}^p \to 2^\Pi$.

D. TWTL Robustness

Definition 2.1 (TWTL Robustness [1]): Given ϕ and $\mathbf{o}_{t_1,t_2}$, the robustness $\rho(\mathbf{o}_{t_1,t_2}, \phi)$ is defined recursively as:

$$\begin{aligned} \rho(\mathbf{o}_{t_1,t_2}, H^d \pi_A) &:= \begin{cases} \min\limits_{t \in [t_1, t_1+d]} h(o_t); & (t_2 - t_1 \geq d) \\ \rho_\perp; & \text{otherwise} \end{cases} \\ \rho(\mathbf{o}_{t_1,t_2}, \phi_1 \wedge \phi_2) &:= \min\{\rho(\mathbf{o}_{t_1,t_2}, \phi_1), \rho(\mathbf{o}_{t_1,t_2}, \phi_2)\} \\ \rho(\mathbf{o}_{t_1,t_2}, \phi_1 \vee \phi_2) &:= \max\{\rho(\mathbf{o}_{t_1,t_2}, \phi_1), \rho(\mathbf{o}_{t_1,t_2}, \phi_2)\} \\ \rho(\mathbf{o}_{t_1,t_2}, \neg\phi) &:= -\rho(\mathbf{o}_{t_1,t_2}, \phi) \\ \rho(\mathbf{o}_{t_1,t_2}, \phi_1 \cdot \phi_2) &:= \\ &\max_{t \in [t_1,t_2)} \big\{\min\big\{\rho(\mathbf{o}_{t_1,t}, \phi_1), \rho(\mathbf{o}_{t+1,t_2}, \phi_2)\big\}\big\} \\ \rho(\mathbf{o}_{t_1,t_2}, [\phi]^{[a,b]}) &:= \begin{cases} \max\limits_{t \geq t_1 + a} \{\rho(\mathbf{o}_{t,t_1+b}, \phi)\}; & (t_2 - t_1 \geq b) \\ \rho_\perp; & \text{otherwise} \end{cases} \end{aligned} \tag{4}$$

where $\rho_\perp$ is a large negative constant.

Lemma 2.1 (Soundness [1]): $\rho(\mathbf{o}_{t_1,t_2}, \phi) > 0 \implies \mathbf{o}_{t_1,t_2} \models \phi$ and $\rho(\mathbf{o}_{t_1,t_2}, \phi) < 0 \implies \mathbf{o}_{t_1,t_2} \not\models \phi$. In words, a positive robustness means the word satisfies the specification.

III. Problem Formulation

We introduce two synthesis problems: an open-loop problem and an MPC closed-loop problem.

Problem 3.1 (Open-Loop Synthesis): Given system (1) with initial state x_0, formula ϕ under Assumption 2.1, and sets $\mathcal{X}$, $\mathcal{U}$, find the control sequence $\mathbf{u}^\star = u_0^\star \cdots u_{T-1}^\star$ solving:

$$\begin{aligned} \mathbf{u}^\star = \arg\max_{\mathbf{u}} \ & \rho(\mathbf{o}_{0,T}, \phi) \\ \text{s.t.} \quad & \text{dynamics constrants from (1)} \end{aligned} \tag{5}$$

Remark 3.1 (Open-loop nature): Problem 3.1 computes $\mathbf{u}^\star$ once at $t = 0$ from the fixed initial state x_0. By Lemma 2.1, any solution with $\rho(\mathbf{o}_{0,T}, \phi) > 0$ guarantees $\mathbf{o}_{0,T} \models \phi$. However, there is no mechanism to react to disturbances during execution. Therefore, we introduce problem 3.2 to address this limitation.

Problem 3.2 (Receding-Horizon MPC Synthesis): Given system (1), formula ϕ, and sets $\mathcal{X}$, $\mathcal{U}$: at each time step $t \in [0, T-1]$, given the measured state x_t, the DFA state q_t, the active task index $i(q_t)$, and its task-adaptive horizon H_t (Definition 5.2), find a closed-loop receding horizon control that exploits the residual formula at the current DFA state.

IV. MILP Encoding of TWTL Robustness

We show that Problems 3.1 and 3.2 reduce to MILPs. Following [10], for each sub-formula ψ of ϕ evaluated over $\mathbf{o}_{t_1,t_2}$, we introduce a continuous variable $r(\psi, t_1, t_2)$ representing $\rho(\mathbf{o}_{t_1,t_2}, \psi)$. The top-level variable $r(\phi, 0, T)$ (or $r(\phi_t, t, t{+}H_t)$ in MPC) is the MILP objective. A strong indicator of MILP performance is binary variable count, so we will mention the number of binary variables required for each operator (Remark 4.1).

A. Big-M Constant

All encodings use $M > 0$ satisfying $M \geq \rho_\top - \rho_\bot$, where $\rho_\top$, $\rho_\bot$ bound the achievable robustness over $\mathcal{X}$. Since $h(o_t) = \mathbf{c}_A^T C x_t - b_A$ is linear and $\mathcal{X}$ is a polytope, tight values are obtained by solving simple LPs over $\mathcal{X}$ *once offline*.

B. Operator Encodings

1) Predicate: For $\psi = \pi_A$, introduce $r(\pi_A, t) \in \mathbb{R}$ for each t:

$$r(\pi_A, t) = h(o_t) = \mathbf{c}_A^T C x_t - b_A. \tag{6}$$

One linear equality per time step; *no binary variables*. The sign of $r(\pi_A, t)$ encodes satisfaction.

2) Hold (Temporal Conjunction): Analogue of STL's always $\square_{[0,d]}\pi_A$. For $\psi = H^d\pi_A$ with $t_2 - t_1 \geq d$:

$$r(H^d\pi_A, t_1, t_2) \leq r(\pi_A,\, t_1 + k), \qquad k = 0, 1, \ldots, d. \tag{7}$$

Tight at $\min_k r(\pi_A, t_1{+}k)$ under maximization. *No binary variables.* If $t_2 - t_1 < d$, set $r(H^d\pi_A, t_1, t_2) = \rho_\bot$.

3) Negation:

$$r(\neg\varphi, t_1, t_2) = -\,r(\varphi, t_1, t_2). \quad \textit{No binary variables.} \tag{8}$$

4) Conjunction:

$$r(\phi_1 \wedge \phi_2, t_1, t_2) \leq r(\phi_i, t_1, t_2), \quad i = 1, 2. \tag{9}$$

Tight at minimum under maximization. *No binary variables.*

5) Disjunction: Using binary $z^\vee \in \{0, 1\}$:

$$\begin{aligned} &r(\phi_1 \vee \phi_2, t_1, t_2) \geq r(\phi_i, t_1, t_2), \quad i = 1, 2, \\ &r(\phi_1 \vee \phi_2, t_1, t_2) \leq r(\phi_1, t_1, t_2) + M(1 - z^\vee), \\ &r(\phi_1 \vee \phi_2, t_1, t_2) \leq r(\phi_2, t_1, t_2) + M\, z^\vee. \end{aligned} \tag{10}$$

One binary variable per disjunction.

6) Within (Temporal Disjunction): Analogue of STL's eventually $\Diamond_{[a,b]}\varphi$. Using binary $z_t^W \in \{0, 1\}$ for each $t \in [t_1{+}a,\, t_1{+}b]$:

$$\begin{aligned} &\sum_{t=t_1+a}^{t_1+b} z_t^W = 1, \\ &r([\varphi]^{[a,b]}, t_1, t_2) \geq r(\varphi, t, t_1 + b), \quad \forall t \in [t_1{+}a,\, t_1{+}b], \\ &r([\varphi]^{[a,b]}, t_1, t_2) \leq r(\varphi, t, t_1 + b) + M(1 - z_t^W), \\ &\qquad\qquad \forall t \in [t_1{+}a,\, t_1{+}b]. \end{aligned} \tag{11}$$

$b{-}a{+}1$ *binary variables.* If $t_2 - t_1 < b$, set $r([\varphi]^{[a,b]}, t_1, t_2) = \rho_\bot$.

7) Concatenation: Using binary $z_t^C, z_t^S \in \{0, 1\}$ for each split $t \in [t_1, t_2)$ and auxiliary inner-minimum variables r_t^C:

$$\begin{aligned} &z_t^C \geq z_{t-1}^C + z_t^S, \quad \forall t \in [t_1, t_2), \\ &z_t^C \leq z_{t-1}^C + M(z_t^S), \quad \forall t \in [t_1, t_2), \\ &z_0^C = 0, \qquad \sum_{t=t_1}^{t_2-1} z_t^S = 1 \\ &r_t^C \leq r(\phi_1, t_1, t) + M(z_t^C), \quad \forall t \in [t_1, t_2), \\ &r_t^C \geq r(\phi_1, t_1, t) - M(z_t^C), \quad \forall t \in [t_1, t_2), \\ &r_t^C \leq r(\phi_2, t, t_2) + M(1 - z_t^C), \quad \forall t \in [t_1, t_2), \\ &r_t^C \geq r(\phi_2, t, t_2) - M(1 - z_t^C), \quad \forall t \in [t_1, t_2), \\ &r(\phi_1 \cdot \phi_2, t_1, t_2) \leq r_t^C, \quad \forall t \in [t_1, t_2), \end{aligned} \tag{12}$$

$2 * (t_2 - t_1)$ *binary variables* per concatenation.

C. Full MILP (Open-Loop)

$$\begin{aligned} \max \quad & r(\phi, 0, T) \\ \text{s.t.} \quad & \text{dynamics constraints from (1)} \\ & \text{constraints (6)–(12) } \forall \text{ sub-formulae of } \phi. \end{aligned} \tag{13}$$

Theorem 4.1 (MILP Correctness): Let $(\mathbf{u}^\star, \mathbf{x}^\star, \mathbf{o}^\star)$ be an optimal solution to (13) with $r(\phi, 0, T) > 0$. Then $\rho(\mathbf{o}^\star_{0,T}, \phi) > 0$, and hence $\mathbf{o}^\star_{0,T} \models \phi$.

Proof: Structural induction on ϕ.

Predicate: $r(\pi_A, t) = h(o_t) = \rho(\mathbf{o}_t, \pi_A)$ by (6). *Hold*: constraints (7) give $r(H^d\pi_A, \cdot) \leq r(\pi_A, t_1{+}k)$ for all k; tight at $\min_k h(o_{t_1+k}) = \rho(\mathbf{o}_{t_1,t_2}, H^d\pi_A)$. *Negation*: direct from (8). *Conjunction*: upper bounds in (9) tighten to the minimum. *Disjunction*: lower bounds force $r \geq \max\{r(\phi_1), r(\phi_2)\}$; big-$M$ upper bound with $z^\vee$ enforces equality.

Within: sum constraint selects $t^\star$; lower bound and big-M upper bound enforce $r = r(\varphi, t^\star, t_1{+}b)$*Concatenation*: upper bounds on r_t^C recover the inner minimum; sum and big-M constraints enforce $r = r_{t^\star}^C$ at the maximizing split. Soundness follows from Lemma 2.1. ∎

Remark 4.1 (Binary variable count): For $\phi = [\phi_1]^{[a_1,b_1]} \cdot \cdots \cdot [\phi_k]^{[a_k,b_k]}$, the open-loop MILP has $O(kT)$ binary variables, dominated by the concatenation operators. Incorporating the DFA automaton structure to replace split-point binaries with automaton transition constraints reduces this to $O(k \log T)$ following [17]; we leave this as future work.

V. Receding-Horizon MPC

We now develop the closed-loop synthesis of Problem 3.2, introducing the task-adaptive horizon, the residual formula, and the warm-start re-solve strategy that together make the MPC tractable (Algorithm 1).

A. Residual Formula

Definition 5.1 (Residual Formula): Given formula ϕ with DFA $\mathcal{A}_\phi$ and current DFA state $q_t \in Q$, the *residual formula* ϕ_t is the sub-formula of ϕ that remains to be satisfied from q_t. It is determined by the accepting paths of $\mathcal{A}_\phi$ from q_t:

$$\mathbf{o}_{t,T} \models \phi_t \iff \delta^*(q_t, \mathbf{o}_{t,T}) \in F, \tag{14}$$

where δ^* is the DFA transition function.

For the typical TWTL formula $\phi = [\phi_1]^{[a_1,b_1]} \cdot [\phi_2]^{[a_2,b_2]} \cdots [\phi_k]^{[a_k,b_k]}$, the DFA state q_t directly identifies the active task index $i(q_t) \in \{1, \ldots, k\}$, so:

$$\phi_t = [\phi_i]^{[a'_i,b'_i]} \cdot [\phi_{i+1}]^{[a_{i+1},b_{i+1}]} \cdots [\phi_k]^{[a_k,b_k]}, \qquad (15)$$

where $[a'_i, b'_i] = [a_i - (t - t_i^s),\ b_i - (t - t_i^s)]$ are the time-shifted bounds for task i, and t_i^s is the time task i started.

Remark 5.1 (Cost of residual formula extraction): Given q_t, extracting ϕ_t costs $O(1)$ (a DFA table lookup to identify $i(q_t)$) plus $O(k)$ to construct the remaining task chain and adjust the time bounds $[a'_i, b'_i]$ for task i. This is negligible relative to the MILP re-solve.

Remark 5.2 (Alternate residual formula): An alternate approach for extracting the residual formula is to adapt the strategy developed in [12] with periodic rewriting of the formula as part of the monitoring efforts. Whereas [12] employs rewriting as an online monitoring effort, our proposed adaptation utilizes rewriting as a simplification method in order to reduce the computational load of a replan. It would do this by removing parts of the formula that have already been satisfied, reducing the number of unnecessary binary variables. This would help with both local re-plans within a DFA state, as it reduces the number of time points still in consideration, as well as a global re-plan due to the removal of already completed tasks. We leave implemenation and benchmarking the effectiveness of this approach as future work.

B. Task-Adaptive Horizon

The full formula horizon $T = \|\phi\|$ is typically large (the sum of all task windows). Solving the MILP over $[t, T]$ at every step is expensive and unnecessary, since the structure of ϕ decomposes into sequential tasks

Definition 5.2 (Task-Adaptive Horizon): Given active task $i = i(q_t)$ started at time t_i^s, and global safety constraints with horizon T_{safe}, the *task-adaptive horizon* at time t is: $H_t := \min\big(b_i - (t - t_i^s),\ T - t\big)$, the smaller of the remaining window of task i and the total remaining horizon.

Remark 5.3 (Comparison with Raman et al. [10]): Raman et al. use a *fixed* horizon $H = T$ at every step, with the executed history $x_0, \ldots, x_{t-1}$ pinned as equality constraints, and re-encode the *full* formula ϕ at every re-solve. Problem 3.2 differs in two ways. First, it uses the *task-adaptive* horizon $H_t \le \max_i b_i$ (Definition 5.2), which is in general much smaller than T and resets at each task transition rather than shrinking monotonically. Second, it encodes only the *residual* formula ϕ_t (Definition 5.1), which is structurally simpler than ϕ once sub-tasks are completed. Both differences are enabled by TWTL's automata structure and are not available for general STL without an automaton.

Remark 5.4 (Why $H_t \le \max_i b_i$, not T): The key observation is that task i must be completed within $b_i - (t - t_i^s)$ steps, regardless of what follows. Planning beyond this window does not help task i and unnecessarily inflates the MILP. Once task i completes and the DFA advances, the horizon *resets* to b_{i+1} for task $i+1$ rather than continuing to shrink from T. The maximum prediction horizon at any step is therefore $\max_i b_i$, *not* T.

TABLE I

HORIZON STRATEGY COMPARISON FOR THE RUNNING EXAMPLE ($T = 50$, $\max_i b_i = 11$, $k = 3$ TASKS). BINARY VARIABLE COUNT EXCLUDES THE SAFETY CONSTRAINT (ZERO COST).

Strategy	**Horizon at step t**	**Binary vars (approx.)**
Raman et al. (fixed $H = T$)	Always 50	~ 150, constant
Shrinking	$T - t$, starts at 50	Decreasing from 150
Task-adaptive (ours)	≤ 11, resets per task	≤ 30, always small

Remark 5.5 (Three horizon strategies): Table I compares the three strategies for the example formula $\phi = ([H^4\pi_A]^{[0,8]} \cdot [H^4\pi_B]^{[0,10]} \cdot [H^3\pi_C]^{[0,11]}) \wedge H^{50}\neg\pi_O$ ($T \approx 50$, $\max_i b_i = 11$).

C. Warm-Start Strategy

At each step, the MILP must be rebuilt with the updated initial state x_t and the shrunken time window $[a'_i, b'_i]$. Rebuilding from scratch is expensive; we instead use a warm-start strategy that makes the per-step overhead negligible.

Offline precomputation. The MILP constraint *matrices* depend on ϕ_t and H_t, not on x_t. Because the DFA has finitely many states ($|Q|$ independent of time bounds), we precompute one parametric MILP per DFA state $q \in Q$ using the full task window b_i as the horizon. This produces a set of matrices $\{(A_q, G_q, h_q)\}_{q \in Q}$ encoding the equality, inequality, and integrality constraints for each residual formula, computed *once offline before execution*.

Online update. At each step t, the online cost has three components:

1) *State update ($O(n)$).* Update the initial-condition equality: $x_{t+1} = Ax_t + Bu_t$ is the only RHS entry that changes. This is a vector update of size n.
2) *Window shrinkage ($O(1)$).* As the task window shrinks by one step, the earliest valid start time for the within operator increases by one. Fix the corresponding binary variable $z^W_{t_1+a'} = 0$ (the newly expired start time) as a simple bound update, rather than removing the constraint. Cost: update one variable bound.
3) *Task transition ($O(n)$).* When the DFA advances from q_t to a new state q_{t+1} (task i completes), swap the precomputed MILP matrices from $(A_{q_t}, G_{q_t}, h_{q_t})$ to $(A_{q_{t+1}}, G_{q_{t+1}}, h_{q_{t+1}})$ and update the initial state. This is an $O(1)$ pointer swap plus an $O(n)$ RHS update.

After the RHS and bound updates, the MILP is warm-started from the previous step's optimal solution, which is feasible for the updated problem up to minor perturbations introduced by x_t. Modern solvers (Gurobi [18], MOSEK [19]) exploit warm starts aggressively, typically resolving in a fraction of the cold-start time when the problem changes only slightly.

Remark 5.6 (Total online cost per step): The online overhead per MPC step — excluding the MILP re-solve

Algorithm 1: Task-Adaptive Receding-Horizon MPC for TWTL Synthesis

Input: System (eq. 1), x_0, formula ϕ, DFA $\mathcal{A}_\phi$, precomputed MILP matrices $\{(A_q, G_q, h_q)\}_{q \in Q}$
Output: Applied control sequence $u_0, u_1, \ldots$

Offline: Precompute MILP matrices $\{(A_q, G_q, h_q)\}_{q \in Q}$; compute $\rho_\top$, $\rho_\bot$ via LP
Initialize DFA: $q_0 \leftarrow q_{\text{init}}$; $t_i^s \leftarrow 0$; warm-start solution $\mathbf{u}^{\text{ws}} \leftarrow \mathbf{0}$
for $t = 0, 1, \ldots, T-1$ **do**
  $i \leftarrow i(q_t)$ `// Active task index: O(1)`
  $H_t \leftarrow \min(b_i - (t - t_i^s),\ T - t)$ Load MILP for q_t: matrices $(A_{q_t}, G_{q_t}, h_{q_t})$
  Update RHS with x_t `// O(n)`
  Fix $z_{t'}^W = 0$ for expired start times $t' < t + a_i'$ Warm-start solver from $\mathbf{u}^{\text{ws}}$
  Solve MILP over $[t,\ t{+}H_t]$
  if *infeasible* ***or*** $r(\phi_t, t, t{+}H_t) \leq 0$ **then**
    Report infeasibility at step t; **break**
  Apply $u_t \leftarrow u_t^\star$; observe x_{t+1}, $o_t = Cx_t$
  $\mathbf{u}^{\text{ws}} \leftarrow$ shift optimal sequence by 1
  $q_{t+1} \leftarrow \delta(q_t, l(o_t))$ `// DFA update: O(1)`
  if $q_{t+1} \in F$ **then**
    Report ϕ satisfied; **break**
  if $i(q_{t+1}) \neq i(q_t)$ **then**
    $t_{i+1}^s \leftarrow t + 1$

— is $O(n)$. This is identical to the parametric MPC observation in [10] but applied to a *progressively smaller* MILP; binary variable count decreases as tasks complete, therefore, re-solve times tend to shorten over the execution horizon.

D. MPC Algorithm

Theorem 5.1 (MPC Satisfaction Guarantee): If Algorithm 1 does not report infeasibility and $r(\phi_t, t, t{+}H_t) > 0$ at every step t, then $\mathbf{o}_{0,T} \models \phi$.

Proof: At each step t, by Theorem 4.1, $r(\phi_t, t, t{+}H_t) > 0$ implies $\rho(\mathbf{o}_{t,t+H_t}, \phi_t) > 0$, hence $\mathbf{o}_{t,t+H_t} \models \phi_t$ by Lemma 2.1. By Definition 5.1, satisfaction of ϕ_t from q_t means $\delta^*(q_t, \mathbf{o}_{t,t+H_t}) \in F$. The DFA correctly tracks execution via $q_{t+1} = \delta(q_t, l(o_t))$, so the full trajectory $\mathbf{o}_{0,T}$ reaches an accepting state, i.e., $\mathbf{o}_{0,T} \models \phi$. ■

Remark 5.7 (Connection to runtime monitors [1]): The runtime robustness monitors of [1] provide an interval $[\rho](\mathbf{o}_{0,t}, \phi)$ at each step. When the lower bound $\underline{\rho}$ of this interval approaches zero, it signals the current trajectory is at risk. This provides a principled, monitor-triggered replanning condition: re-solve only when $\underline{\rho} \leq \epsilon$ for a threshold $\epsilon > 0$, reducing unnecessary MILP solves in disturbance-free segments.

VI. Numerical Example

A. Experimental Setup

For ease of comparison, we will use the same TWTL formula for all of the following examples:

$$\phi = ([H^4 \pi_A]^{[0,8]} \cdot [H^4 \pi_B]^{[0,10]} \cdot [H^3 \pi_C]^{[0,11]}) \wedge H^{50} \neg \pi_O \tag{16}$$

All of the following examples take place in a 20×20 continuous $\mathbb{R}^2$ workspace, where the agent moves under double integrator dynamics discretized at $\Delta t = 1$. The propositions π_A, π_B, and π_C correspond to three square regions of interest the agent must visit and hold in, with a single square obstacle placed in the workspace. The agent starts at the origin $(0, 0)$ for every run.

Each specification is encoded as a MILP and solved with Gurobi [18]. All timing results were collected on an Apple M1 Pro MacBook Pro with 16 GB of RAM, and reported solve times are wall clock times.

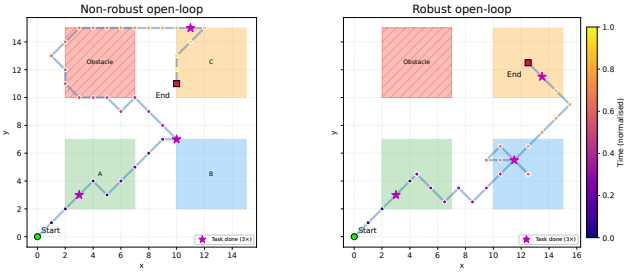

Fig. 1. Open loop solving, with non-robustness based path (L) and robustness based path (R) using the specification from eq. (16)

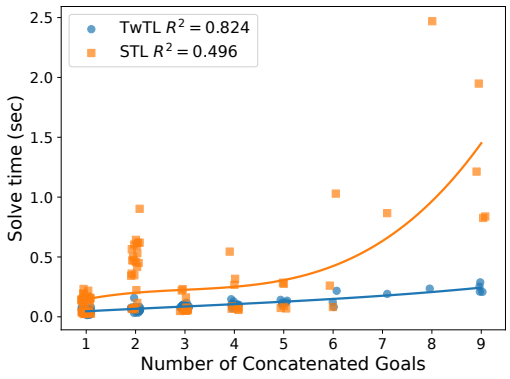

Fig. 2. Solve time relative to number of concatenated goals. A 3rd order regression was applied for the plotted line, and then a 0.1 random jitter was applied to the x-axis for readability

B. Robustness Relevance

To show the importance of having a robustness measure, we compare planned paths: one where the solver is not optimizing for robustness, and one where it is. While both satisfy the specification, the robustness based path is qualitatively and quantitatively more robust to failures (Fig. 1).

C. Open Loop Solve Time Comparison with STL

To compare solving times between TWTL and STL, we translate TWTL specifications into STL specifications. In STL, (16) translates to: $\phi = \Diamond_{[0,4]}(\Box_{[0,4]}(\pi_A) \wedge \Diamond_{[0,6]}(\Box_{[0,4]}(\pi_B) \wedge \Diamond_{[0,8]}(\Box_{[0,3]}(\pi_C)))) \wedge \Box_{[0,50]}(\neg \pi_O)$

We benchmark TWTL and STL solve times for randomized number goals and task durations with semantically equivalent specifications (Fig. 2). For smaller numbers of tasks, they are about equal with STL performing slightly better on average, but with four or more subtasks, TWTL performs better due to the reduction in binary variables.

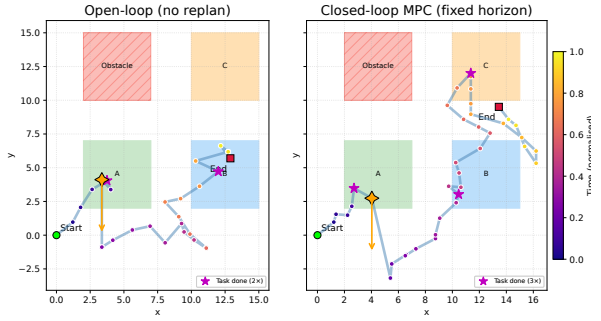

Fig. 3. Comparing recovery of Open loop (L) vs Closed loop (R) MPC after a disturbance. Disturbance happened at time step 6, notated with an orange star, and was unknown prior to planning. This again follows the specification in eq. 16. The open loop trajectory follows its original plan which just happens to wander into region B

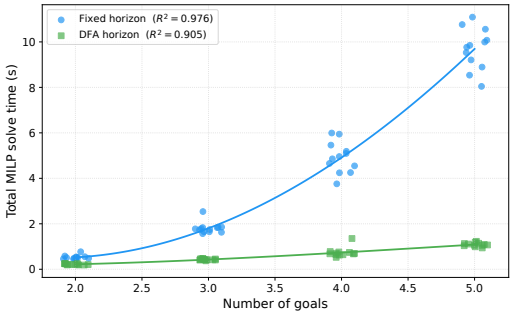

Fig. 4. Total solve time over the course of a run comparing a fixed horizion approach (analogous to [10]) with our proposed DFA approach over an increasing number of sequential goals.

D. Open Loop vs Closed Loop

To demonstrate the importance of closed-loop control, we introduce a disturbance to the path, displacing the agent down as it attempts to follow the open loop control inputs causing it to fail, while MPC is able to adapt and replan, completing all three tasks (Fig. 3).

E. Fixed Horizion vs DFA approach

An advantage of TWTL is the ability to utilize a DFA for closed-loop planning and control (Sec. II-C). DFA replanning outperforms the fixed horizon approach, with increased benefit as the number of tasks increases (Fig. 4).

VII. Conclusion and Future Work

We presented a robustness-maximizing MILP synthesis framework for TWTL specifications, comprising an open-loop trajectory optimizer and a closed-loop receding-horizon MPC controller. The MILP encoding extends the approach of [10] for STL to TWTL's hold, within, and concatenation operators, with a provable correctness guarantee via soundness of ρ. We also showed that for problems which are best expressed as multiple sequential subtasks, MILP based synthesis of TWTL is more efficient than MILP based synthesis of STL. The MPC formulation introduces a task-adaptive prediction horizon that exploits the TWTL DFA to bound the per-step MILP size by $\max_i b_i$ rather than T—a feature not available for STL—and a warm-start strategy that reduces per-step online overhead to $O(n)$.

In future work, we plan to 1) Exploit the DFA structure to further reduce binary variables via logarithmic encoding [17] (Remark 4.1); 2) Extend to the AGM robustness η of [1] via a mixed-integer second-order cone program (MISOCP) formulation; 3) Utilize rewriting on both current state and whole specifications to reduce binary variables following. [12]; and 4) Apply the framework to multi-agent planning with TWTL task allocation.

References

[1] A. Ahmad, C.-I. Vasile, R. Tron, and C. Belta, "Robustness measures and monitors for time window temporal logic," in *2023 62nd IEEE CDC*. IEEE, 2023, pp. 6841–6846.

[2] C. Baier and J.-P. Katoen, *Principles of model checking*. MIT press, 2008.

[3] S. Konur, "A survey on temporal logics for specifying and verifying real-time systems," *Frontiers of Computer Science*, vol. 7, no. 3, pp. 370–403, June 2013.

[4] O. Maler and D. Nickovic, "Monitoring temporal properties of continuous signals," in *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*. Springer, 2004, pp. 152–166.

[5] R. Koymans, "Specifying real-time properties with metric temporal logic," *Real-time systems*, vol. 2, no. 4, pp. 255–299, 1990.

[6] C.-I. Vasile, D. Aksaray, and C. Belta, "Time window temporal logic," *Theoretical Computer Science*, vol. 691, pp. 27–54, 2017.

[7] F. Penedo, C.-I. Vasile, and C. Belta, "Language-guided sampling-based planning using temporal relaxation," in *Algorithmic Foundations of Robotics XII*. Springer, 2020, pp. 128–143.

[8] C. Belta and S. Sadraddini, "Formal methods for control synthesis: An optimization perspective," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 2, pp. 115–140, 2019.

[9] A. Donzé and O. Maler, "Robust satisfaction of temporal logic over real-valued signals," in *FORMATS 2010, Klosterneuburg, Austria, September 8-10, 2010. Proceedings 8*. Springer, 2010, pp. 92–106.

[10] V. Raman, A. Donzé, M. Maasoumy, R. M. Murray, A. Sangiovanni-Vincentelli, and S. A. Seshia, "Model predictive control with signal temporal logic specifications," in *Proc. IEEE CDC*, 2014, pp. 81–87.

[11] N. Mehdipour, C.-I. Vasile, and C. Belta, "Arithmetic-geometric mean robustness for control from signal temporal logic specifications," in *2019 American Control Conference (ACC)*. IEEE, 2019, pp. 1690–1695.

[12] E. Bonnah and K. A. Hoque, "Runtime monitoring of time window temporal logic," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 5888–5895, 2022.

[13] E. Bonnah and K. Hoque, "Qtwtl: Quality aware time window temporal logic for performance monitoring," 08 2023.

[14] A. Ahmad, S. Liu, R. Tron, and C. Belta, "Rrt $^{\eta}$: Sampling-based motion planning and control from stl specifications using arithmetic-geometric mean robustness," *arXiv preprint arXiv:2602.16825*, 2026.

[15] D. Kamale and C.-I. Vasile, "Optimal Control Synthesis with Relaxed Global Temporal Logic Specifications for Homogeneous Multi-robot Teams," in *2024 IEEE ICRA*. IEEE, 2024, pp. 250–256.

[16] G. A. Cardona, D. Kamale, and C.-I. Vasile, "Mixed Integer Linear Programming Approach for Control Synthesis with Weighted Signal Temporal Logic," ser. HSCC '23, New York, NY, USA, May 2023, pp. 1–12.

[17] V. Kurtz and H. Lin, "Mixed-Integer Programming for Signal Temporal Logic with Fewer Binary Variables," May 2022, arXiv:2204.06367 [eess.SY].

[18] Gurobi Optimization, LLC, "Gurobi optimizer reference manual," 2026.

[19] M. ApS, *Mosek Documentation*, 2019. [Online]. Available: https://www.mosek.com/documentation/