An STL-based Approach to Resilient Control for Cyber-Physical Systems

Hongkai Chen
hongkai.chen@stonybrook.edu
Stony Brook University
Stony Brook, NY, USA

Scott A. Smolka
sas@cs.stonybrook.edu
Stony Brook University
Stony Brook, NY, USA

Nicola Paoletti
nicola.paoletti@kcl.ac.uk
King's College London
London, UK

Shan Lin
shan.x.lin@stonybrook.edu
Stony Brook University
Stony Brook, NY, USA

ABSTRACT

We present *ResilienC*, a framework for resilient control of Cyber-Physical Systems subject to STL-based requirements. ResilienC utilizes a recently developed formalism for specifying CPS resiliency in terms of sets of (rec, dur) real-valued pairs, where *rec* represents the system's capability to rapidly recover from a property violation (*recoverability*), and *dur* is reflective of its ability to avoid violations post-recovery (*durability*). We define the *resilient STL control problem* as one of *multi-objective optimization*, where the recoverability and durability of the desired STL specification are maximized. When neither objective is prioritized over the other, the solution to the problem is a set of *Pareto-optimal* system trajectories. We present a precise solution method to the resilient STL control problem using a mixed-integer linear programming encoding and an *a posteriori* ϵ-constraint approach for efficiently retrieving the complete set of optimally resilient solutions. In ResilienC, at each time-step, the optimal control action selected from the set of Pareto-optimal solutions by a *Decision Maker* strategy realizes a form of *Model Predictive Control*. We demonstrate the practical utility of the ResilienC framework on two significant case studies: autonomous vehicle lane keeping and deadline-driven, multi-region package delivery.

ACM Reference Format:
Hongkai Chen, Scott A. Smolka, Nicola Paoletti, and Shan Lin. 2023. An STL-based Approach to Resilient Control for Cyber-Physical Systems. In *Proceedings of the 26th ACM International Conference on Hybrid Systems: Computation and Control (HSCC '23), May 09–12, 2023, San Antonio, TX, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3575870.3587119

1 INTRODUCTION

Resiliency is of fundamental importance in Cyber-Physical Systems (CPS), as such systems are expected to fulfill safety- and mission-critical requirements even in the presence of external disturbances or internal faults. Although various notions of resiliency have been proposed within a control setting [5, 27], a general formal characterization has been lacking. Recently, Chen et al. [6] used Signal Temporal Logic (STL) [16] to formally reason about resiliency in CPS. Given an STL property φ expressing a CPS requirement, the notion of resiliency introduced in [6] permits violations of φ as long as: 1) the CPS quickly recovers from the violation, and then 2) satisfies φ for an extended period of time. These two requirements are called *recoverability* and *durability*, respectively.

The results of [6] naturally suggest the following problem of *resilient STL control*: find an optimal control strategy that maximizes the system's resilience in terms of recoverability and durability. These two objectives are often at odds with each other. For example, in the *lane-keeping problem* (see Figure 2), an aggressive control strategy can quickly return the vehicle to the lane after a violation (good recoverability) but might fail to keep the vehicle in the lane for an extended period of time due to overshooting (poor durability). On the other hand, with a cautious strategy, the vehicle might take longer to re-enter the lane (poor recoverability) but subsequently manage to remain in the lane longer (good durability). In other words, without prioritizing one requirement over the other, the aggressive and cautious strategies are *mutually non-dominated* and, hence, equally resilient.

In this paper, we present a control framework called *ResilienC* (the 'C' stands for control), where the resilient STL control problem is formulated as one of multi-objective optimization, designed to maximize both the recoverability and durability of the CPS. Unlike existing techniques for STL-based control [21, 24] which focus on optimizing a single objective (e.g., spatial robustness in [21] and time robustness in [24]) and thus produce a single solution, our method results in a set of *non-dominated, aka Pareto-optimal, solutions*. Such a method is also called *a posteriori* as it avoids making any *a priori* assumptions about the relative importance of the two objectives (recoverability and durability). We achieve a Model Predictive Control (MPC) scheme with our method by deploying a *Decision Maker* (DM) strategy that, at each time step, selects the next optimal control action from among the set of Pareto-optimal solutions for execution by the plant. See Figure 1 for an overview of the ResilienC framework.

We solve the resilient STL control problem in a precise manner, in that our method can retrieve the entire set of non-dominated resilient points. To do so, we focus on CPS with linear dynamics and

HSCC '23, May 09–12, 2023, San Antonio, TX, USA

ACM ISBN 979-8-4007-0033-0/23/05…$15.00
https://doi.org/10.1145/3575870.3587119

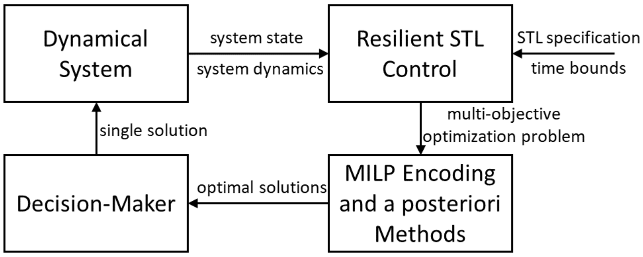


Figure 1: Overview of ResilienC architecture.

encode the problem as one of mixed-integer linear programming (MILP). In particular, the solution set of the multi-objective problem is found by solving multiple single-objective MILP instances through an ϵ-constraint approach [11].

From a theoretical standpoint, besides proving the correctness and characterizing the complexity of our algorithm, we establish an important relationship between our resilient STL control problem and the time-robust STL control problem recently introduced in [24]. Time-robust STL control seeks to optimize time robustness, i.e., the extent to which a trajectory can be shifted in time without affecting the satisfaction of the STL specification. We prove that any time-robust solution is also a resilient solution, making resilient STL control a generalization of time-robust STL control.

We evaluate ResilienC on two case studies: lane keeping and deadline-driven multi-region package delivery. Our results clearly demonstrate the effectiveness of our solution method, which provides a comprehensive view of the recoverability-durability tradeoff. Furthermore, we use case studies to assess and compare the various DM strategies.

In summary, our main contributions are the following.

- We present *ResilienC*, a resilient control framework for CPS with STL-based requirements. We define the resilient control problem as one of multi-objective optimization such that the recoverability and durability metrics associated with the STL specifications are maximized and a set of Pareto-optimal solutions is generated. We also propose various DM strategies for selecting a single optimal solution, used to generate MPC-based control actions. To the best of our knowledge, we are the first to investigate a resilient control framework that co-optimizes recoverability and durability.
- We present a precise solution method, based on an MILP encoding and an a posteriori ϵ-constraint approach, for efficiently retrieving the complete set of optimally resilient solutions.
- We prove that our resilient control framework is a generalization of time-robust STL control.
- We conducted two case studies for which we considered various control strategies that induce vastly different but equivalently resilient trajectories. We also illustrate the effects of multiple DM preferences on ResilienC-based control.

2 BACKGROUND

In this section, we provide background on the syntax and semantics of both STL and the STL-based Resiliency Specifications of [6]. Let $\xi : \mathbb{T} \rightarrow \mathbb{R}^n$ be a signal where $\mathbb{T} = \mathbb{Z}_{\geq 0}$ is the (discrete) time domain. We denote by $|\xi|$ the length of ξ. Given $t \in \mathbb{T}$ and interval I on $\mathbb{T}$, $t + I$ is used to denote the set $\{t + t' \mid t' \in I\}$.

2.1 Signal Temporal Logic

STL is a logical formalism for specifying temporal properties over real-valued signals. An STL atomic proposition $p \in AP$ is defined over ξ and is of the form $p \equiv \mu(\xi(t)) \geq c$, $c \in \mathbb{R}$, and $\mu : \mathbb{R}^n \rightarrow \mathbb{R}$. STL formulas φ are defined according to the following grammar [9]:

$$\varphi ::= p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \, \mathbf{U}_I \, \varphi_2$$

where $\mathbf{U}$ is the *until* operator and I is an interval on $\mathbb{T}$. Logical disjunction is derived from $\wedge$ and $\neg$ as usual, and operators *eventually* and *always* are derived from $\mathbf{U}$ as usual: $\mathbf{F}_I\varphi = \top\,\mathbf{U}_I\,\varphi$ and $\mathbf{G}_I\varphi = \neg\,(\mathbf{F}_I\neg\,\varphi)$. The satisfaction relation $(\xi, t) \models \varphi$, indicating whether ξ satisfies φ at time t, is defined as follows:

$$\begin{aligned}
(\xi, t) \models p \quad &\Leftrightarrow \quad \mu(\xi(t)) \geq c \\
(\xi, t) \models \neg\varphi \quad &\Leftrightarrow \quad \neg((\xi, t) \models \varphi) \\
(\xi, t) \models \varphi_1 \wedge \varphi_2 \quad &\Leftrightarrow \quad (\xi, t) \models \varphi_1 \wedge (\xi, t) \models \varphi_2 \\
(\xi, t) \models \varphi_1 \mathbf{U}_I \varphi_2 \quad &\Leftrightarrow \quad \exists\, t' \in t + I \text{ s.t. } (\xi, t') \models \varphi_2 \wedge \\
&\qquad\qquad \forall\, t'' \in [t, t'),\ (\xi, t'') \models \varphi_1
\end{aligned}$$

The *characteristic function* χ of φ relative to ξ at time t is defined such that $\chi(\varphi, \xi, t) = 1$ when $(\xi, t) \models \varphi$ and -1 otherwise [9].

STL admits a quantitative semantics called (space) robustness [9] that quantifies the extent to which ξ satisfies φ at time t. Its absolute value can be seen as the distance of ξ from the set of trajectories satisfying (positive value) or violating (negative value) φ.

STL also admits a quantitative semantics called *time robustness* [9], which is used to quantify the extent to which a trajectory can be shifted in time without affecting the satisfaction (or violation) of the STL specification. Its definition is given in terms of the real-valued function θ^+:

$$\begin{aligned}
\theta^+(p, \xi, t) \quad &= \quad \chi(p, \xi, t) \cdot \max\{d \geq 0 \; s.t. \, \forall\, t' \in [t, t + d], \\
&\qquad\qquad \chi(p, \xi, t') = \chi(p, \xi, t)\} \\
\theta^+(\neg\varphi, \xi, t) \quad &= \quad -\,\theta^+(\varphi, \xi, t) \\
\theta^+(\varphi_1 \wedge \varphi_2, \xi, t) \quad &= \quad \min(\theta^+(\varphi_1, \xi, t), \theta^+(\varphi_2, \xi, t)) \\
\theta^+(\varphi_1 \mathbf{U}_I \varphi_2, \xi, t) \quad &= \quad \max_{t' \in t+I} \min(\theta^+(\varphi_2, \xi, t'), \min_{t'' \in [t, t')} \theta^+(\varphi_1, \xi, t''))
\end{aligned}$$

As with space robustness, STL time robustness is sound in that positive or negative values of θ^+ correspond to satisfaction or violation in the usual Boolean interpretation.

2.2 STL-based Resilience

We now give an overview of the formulation of STL resilience introduced in [6]. The intuition is that given an STL formula φ, two properties characterize its resilience: recoverability and durability. Recoverability requires a signal to recover from a violation of φ within time α; durability requires a signal to maintain the satisfaction of φ for at least a duration of β. A signal is resilient if it satisfies both properties. Given an STL formula φ and $\alpha, \beta \in \mathbb{T}$, $\beta > 0$, the formula

$$R_{\alpha,\beta}(\varphi) \equiv \neg\varphi \mathbf{U}_{[0,\alpha]} \mathbf{G}_{[0,\beta)} \varphi$$

captures both temporal requirements (recoverability and durability). In [6], $R_{\alpha,\beta}(\varphi)$ expressions are the *atomic formulas* of an STL-like logic called *STL-based Resiliency Specifications* (SRS).

Akin to space and time robustness, a quantitative semantics in the form of a *Resilience Satisfaction Value* (ReSV) is proposed to measure the resilience of SRS formulas. The ReSV of $R_{\alpha,\beta}(\varphi)$ w.r.t. ξ at time t is a (rec, dur) pair given by:

$$(-t_{rec}(\varphi,\xi,t)+\alpha, t_{dur}(\varphi,\xi,t)-\beta)$$

where

$$t_{rec}(\varphi,\xi,t) = \min\left(\{d \in \mathbb{T} \mid (\xi, t+d) \models \varphi\} \cup \{|\xi| - t\}\right)$$

$$t_{dur}(\varphi,\xi,t) = \min\left(\{d \in \mathbb{T} \mid (\xi, t'+d) \models \neg\varphi\} \cup \{|\xi| - t'\}\right), \quad t' = t + t_{rec}(\varphi,\xi,t)$$

The value of $t_{rec}(\varphi,\xi,t)$ quantifies the time needed for ξ to recover from a violation of φ at time t, and $t' = t + t_{rec}(\varphi,\xi,t)$ is the (absolute) recovery time. The value of $t_{dur}(\varphi,\xi,t)$ quantifies the time period after t' during which φ remains true. Thus, *rec* tells us how early before α we recover, and *dur* how long after β φ is maintained true. Going forward, we often abbreviate $t_{rec}(\varphi,\xi,0)$ and $t_{dur}(\varphi,\xi,0)$ with t_{rec} and t_{dur}, respectively.

Akin to space and time STL robustness, the authors of [6] prove that the ReSV semantics is sound in that $(\xi,t) \models R_{\alpha,\beta}(\varphi)$ holds if $-t_{rec}(\varphi,\xi,t)+\alpha \geq 0$ and $t_{dur}(\varphi,\xi,t)-\beta \geq 0$, with at least one of the two inequalities strictly holding. Thus, the resilience of φ w.r.t. ξ at time t can be represented by a (rec, dur) pair.

The ReSV definition and the soundness result extend to composite SRS formulas. The intuition behind this extension is that the ReSV of e.g., an *always* (*eventually*) formula with bound I, represents the worst-case (best-case) resilience value attained by the subformula within I. For control purposes, however, we are only interested in SRS atoms. See also Remark 1.

For finding an optimally resilient control strategy, it is necessary to compare the resilience of φ w.r.t. two signals. In [6], an ordering relation $\succ_{re}$ is introduced specifically for this purpose.[1] The intuition is that usual Pareto-dominance $\succ$ over the reals is not consistent with resiliency satisfaction. Recall that given two real-valued tuples $x, y \in \mathbb{R}^n$, x *Pareto-dominates* y, denoted by $x \succ y$, if $x_i \geq y_i$, $1 \leq i \leq n$, and $x_i > y_i$ for at least one such i, under the usual ordering $>$. Now consider the (rec, dur) pairs $(-1, 2)$ and $(1, 1)$. By usual Pareto-dominance, $(-1, 2)$ and $(1, 1)$ are mutually non-dominated, but an ReSV of $(-1, 2)$ indicates that the system does not satisfy recoverability; namely it recovers one time unit too late. On the other hand, an ReSV of $(1, 1)$ implies satisfaction of both recoverability and durability bounds, and thus should be preferred to $(-1, 2)$. This intuition is formalized in the definition of the $\succ_{re}$ relation.

Definition 2.1 (Resiliency Binary Relations [6]). We define binary relations $\succ_{re}$, $=_{re}$, and $\prec_{re}$ in $\mathbb{Z}^2$. Let $x, y \in \mathbb{Z}^2$ with $x = (x_r, x_d)$, $y = (y_r, y_d)$, and *sign* is the signum function. We have that $x \succ_{re} y$ if one of the following holds:

(1) $sign(x_r) + sign(x_d) = sign(y_r) + sign(y_d)$, and $x \succ y$.
(2) $sign(x_r) + sign(x_d) > sign(y_r) + sign(y_d)$.

We denote by $\prec_{re}$ the dual of $\succ_{re}$. If neither $x \succ_{re} y$ nor $y \prec_{re} x$,[2] then x and y are *mutually non-dominated*, denoted $x =_{re} y$. Under this ordering, a *non-dominated set* S is such that $x =_{re} y$ for all $x, y \in S$.

Given a binary relation $\rhd$ and a non-empty set $P \subseteq \mathbb{Z}^2$, we denote with $\max_{\rhd}(P)$ the maximum set induced by the ordering $\rhd$, i.e., the largest subset $S \subseteq P$ such that $\forall x \in S, \forall y \in P, x \not\lhd y$. The minimum set is defined analogously as $\min_{\rhd}(P) = \max_{\lhd}(P)$. In the following, we will use the so-called *maximum resilience set* $\max_{\succ_{re}}(P)$, abbreviated as $\max_{re}(P)$, and the one induced by canonical Pareto-dominance $\max_{\succ}(P)$, abbreviated as $\max(P)$.

3 PROBLEM FORMULATION

Consider a discrete-time, linear dynamical system (F, G, x_0) with dynamics $x_{t+1} = Fx_t + Gu_t$, where $F \in \mathbb{R}^{n \times n}$, $G \in \mathbb{R}^{n \times m}$, $x_t \in \mathbb{R}^n$ is the system state, and $u_t \in U \subseteq \mathbb{R}^m$ is the control input at time t, with the control space U defined as a closed polytope. Any sequence of control actions $\vec{u} = [u_0, \ldots, u_{H-1}]$ induces a sequence of system states $\vec{x} = [x_0, \ldots, x_H]$ starting at x_0 and generated by the system dynamics.

We now define the *Resilient STL Control* problem as a *bi-objective optimization* problem aimed at maximizing the recoverability and durability of $\vec{x}$ with respect to φ at time 0.[3]

Problem 1 (Resilient STL Control). *Let φ be an STL formula, (F, G, x_0) the control system, H the control horizon, and $\alpha, \beta \in \mathbb{T}$, $\beta > 0$. Solve*

$$\mathcal{S}^* = \underset{\vec{u}}{max_{re}}(\alpha - t_{rec}(\varphi,\vec{x},0),\ t_{dur}(\varphi,\vec{x},0) - \beta)$$

s.t. $x_{t+1} = Fx_t + Gu_t,\ u_t \in U,\ t \in [0, \ldots, H-1].$

Remark 1. *Time $t = 0$ represents the offset of $\vec{x}$ at which φ is evaluated. The formulation of the problem is still general because we can consider trajectories starting at any state x_0. The optimization objectives of Problem 1 correspond to the ReSV semantics of $R_{\alpha,\beta}(\varphi)$ formulas, which is an atom in the SRS temporal logic of [6]. In this way, we focus on optimizing the recoverability and durability of the* first *recovery episode w.r.t. φ over an MPC-style prediction horizon H [23]. This is arguably more useful for control purposes than optimizing the ReSV of SRS formulas with temporal operators, where optimizing the ReSV of $\mathbf{G}_I R_{\alpha,\beta}(\varphi)$ ($\mathbf{F}_I R_{\alpha,\beta}(\varphi)$) corresponds to optimizing the worst-case (best-case) recovery episode w.r.t. φ within I. From a technical perspective, the ReSV of a composite formula is itself a set of non-dominated (rec, dur) pairs (as opposed to a single pair for $R_{\alpha,\beta}(\varphi)$ atoms), which would unnecessarily complicate the definition of Problem 1.*

The *optimal solution* to Problem 1 is a set $\mathcal{S}^* \subseteq \mathbb{R}^2$ of non-dominated (rec, dur) pairs, i.e., the maximum resilience set ($\max_{re}$) of all (rec, dur) pairs induced by all possible sequences of control inputs. We denote by $\mathcal{U}_1^* \subseteq U^H$ the set of optimal points in the decision (control) space, where each point induces one optimal solution in $\mathcal{S}^*$.

Example 1. *In the lane-keeping problem, a vehicle is required to stay within its lane (colored grey in Figure 2) at all times. When the*

[1] The motivation for introducing the ordering relation $\succ_{re}$ in [6] is a different one, namely for computing the semantics of composite SRS formulas.

[2] This is equivalent to saying that $sign(x_r) + sign(x_d) = sign(y_r) + sign(y_d)$ and neither $x \succ y$ nor $y \succ x$.

[3] We slightly abuse notation and use sequences instead of signals.

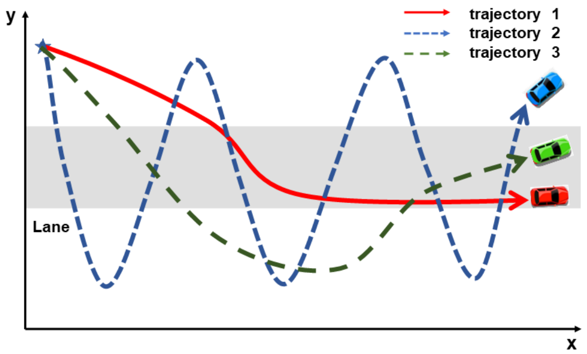


Figure 2: Vehicle trajectories corresponding to three optimal solutions provided by our ResilienC framework. (The figure is for illustrative purposes only and does not directly reflect our experimental results.)

vehicle is forced to exit the lane due to e.g. an external disturbance, the resilient STL controller predicts multiple optimal trajectories for the vehicle. Three such trajectories are shown in Figure 2. The initial location of the vehicle, marked by a star, is outside the lane, so it violates the lane-keeping specification φ. Trajectory 1 is the slowest to recover from its violation of φ, resulting in the worst recoverability among the three. However, the vehicle subsequently maintains φ until the end of the trajectory, resulting in the best durability. In trajectory 3, the vehicle aggressively steers back into the lane whenever φ is violated; so it exhibits the best recoverability. It cannot, however, maintain φ after satisfaction due to overshooting; it thus has the worst durability. The behavior of trajectory 2 lies in-between trajectories 1 and 3. In our approach, the (rec, dur) values of these three trajectories are mutually non-dominated and hence, equally resilient.

Proposition 3.1 establishes the relationship between the sets of Pareto-optimal solutions obtained under the $>_{re}$ ordering and the usual $>$ ordering. This result is useful *per se* and will be also required in Section 4 to prove the correctness of our algorithm.

Proposition 3.1. *For any $P \subseteq \mathbb{Z}^2$ with $P \neq \emptyset$,* $\max_{re}(P) \subseteq \max(P)$.

Proof. We prove that any x in $\max_{re}(P)$ is also in $\max(P)$. Let us remark that, if $x = (x_r, x_d) \in \max_{re}(P)$, then $x >_{re} y$ or $x =_{re} y$ for all $y = (y_r, y_d) \in P$. Similarly, if $x \in \max(P)$, then for all $y \in P$, $x > y$ or x and y are mutually non-dominated w.r.t. $>$. We distinguish two cases: **(i)** if $sign(x_r) + sign(x_d) = sign(y_r) + sign(y_d)$, we have $x > y$ (when $x >_{re} y$) or x and y are mutually non-dominated (when $x =_{re} y$). **(ii)** if $sign(x_r) + sign(x_d) > sign(y_r) + sign(y_d)$, we show that $x \not< y$ (i.e., $x > y$ or x and y are mutually non-dominated). If, by contradiction, y Pareto-dominates x then we have $y_r \geq x_r$ and $y_d \geq x_d$ with at least one inequality holding strictly. This contradicts the assumption $sign(x_r) + sign(x_d) > sign(y_r) + sign(y_d)$ because $sign$ function is monotonic non-decreasing. □

The related problem of *Time-Robust STL Control Synthesis* [24] seeks to maximize a single objective: the STL time robustness of $\vec{x}$ with respect to φ at time 0. For this reason, its optimal solution is a single value (if one exists).

Problem 2 (Time-Robust STL Control Synthesis). *Let φ be an STL formula, (F, G, x_0) the control system, and H the control horizon. Solve*

$$\theta^* = \max_{\vec{u}} \theta^+(\varphi, \vec{x}, 0)$$

s.t. $x_{t+1} = Fx_t + Gu_t,\ u_t \in U,\ t \in [0, \ldots, H-1]$ *and* $\theta^+(\varphi, \vec{x}, 0) \geq \theta_l > 0$.

We note that the time robustness $\theta^+(\varphi, \vec{x}, 0)$ is constrained by a positive lower bound θ_l, meaning that the above problem is not always feasible. The result θ^* is the *optimal solution* to the problem. We denote with $\mathcal{U}_2^*$ the corresponding set of *optimal points* in the decision/control space.

Remark 2 (Optimal points). *In Problem 1, there are in general multiple optimal points in $\mathcal{U}_1^*$ for two reasons: (1) $\mathcal{S}^*$ may contain multiple optimal solutions; (2) even if $\mathcal{S}^*$ contains only one solution, there might be multiple control strategies inducing the same optimal solution.*[4] *In Problem 2, there may be multiple points in $\mathcal{U}_2^*$ for a similar reason: it possibly contains multiple $\vec{u}$ yielding the same optimal time robustness θ^*.*

Proposition 3.2. *Problem 1 is solvable, i.e., an optimal point exists. Problem 2 is solvable if feasible.*

Proof. Problem 1 has a closed-polytope feasible region. Its objectives are integers and bounded, so there exists an optimal point such that the optimal values are achieved, hence solvable. Similar reasoning applies to Problem 2; thus it is solvable if feasible. □

Although the two problems seem different, we emphasize that Problem 1 generalizes Problem 2 in the sense that solving the latter is equivalent to finding a particular optimal solution to the former.[5] See the following proposition.

Proposition 3.3. *For a system (F, G, x_0), control horizon H, and STL formula φ, we have $\mathcal{U}_2^* \subseteq \mathcal{U}_1^*$.*

Proof. If Problem 2 is infeasible, it is trivial that $\mathcal{U}_2^* = \emptyset \subseteq \mathcal{U}_1^*$. Otherwise, we prove that $\vec{u} \in \mathcal{U}_1^*$ for all $\vec{u} \in \mathcal{U}_2^*$. Because of the constraint $\theta^+(\varphi, \vec{x}, 0) > 0$, the initial state of (F, G, x_0) must satisfy φ. A solution $\vec{u} \in \mathcal{U}_2^*$ maximizes $\theta^+(\varphi, \vec{x}, 0)$, and thus induces a trajectory on (F, G, x_0) where φ is maintained for as long as possible from time 0. This indicates that $\vec{u}$ maximizes $\alpha - t_{rec}$ and $t_{dur} - \beta$ to their global maximum simultaneously (t_{rec} reaches its lower bound 0 and t_{dur} is maximized to $\theta^* + 1$), which is an optimal solution in $\mathcal{S}^*$. Therefore, we have $\vec{u} \in \mathcal{U}_1^*$. □

4 SOLUTION METHOD FOR RESILIENT STL CONTROL

In this section, we introduce our solution method for solving Problem 1. We note that both the STL Boolean semantics and the resiliency objectives $\alpha - t_{rec}$ and $t_{dur} - \beta$ are discrete (hence, non-smooth), which makes gradient-based methods unsuitable. Metaheuristics similarly tend to perform poorly and do not provide optimality guarantees. For linear systems, however, prior work

[4] Even if two different controllers generate two different trajectories, these trajectories might have the same recoverability and durability.

[5] With this result at hand, we will skip any experimental comparison between time-robust and resilient controllers, as the former is a special case of the latter.

has shown that (single-objective) optimization of STL space and time robustness can be formulated and precisely solved as an MILP problem [21, 24].

Here, we take a similar approach and encode t_{rec} and t_{dur} using MILP constraints, building on the encoding of the Boolean STL semantics of Raman et al. [21]. To retrieve the full set of Pareto-optimal solutions, we define an ϵ-constraint approach [11] that solves the bi-objective problem through multiple single-objective MILP instances, where one of the objectives is optimized and the other is constrained above some given level.

Section 4.1 presents our MILP encoding of the Boolean STL semantics and the resiliency objectives. In Section 4.2, we present an ϵ-constraint approach for efficiently computing the set of non-dominated optimal solutions, and provide a proof of its correctness. Section 4.3 analyzes our algorithm's computational complexity.

4.1 MILP Encoding

The encoding method consists of the following three main steps.

(1) Boolean semantics for STL atomic propositions. Let $p = \mu(x_t) \geq c$ be an STL atomic proposition. We use binary variables $z_t^{\mu} \in \{0, 1\}$ to represent Boolean satisfaction ($z_t^{\mu} = 1$) or violation ($z_t^{\mu} = 0$) of p over the control horizon at every time step $t = 0, \dots, H$. Assuming that the STL atomic propositions are linear w.r.t. x_t (i.e., μ is a linear function), we can encode the Boolean semantics of STL atomic propositions with MILP constraints.

$$(z_t^{\mu} - 1) \cdot M \leq \mu(x_t) - c \leq z_t^{\mu} \cdot M \tag{1}$$

where M is a significantly large value.

(2) Boolean semantics for STL composite formulas. The Boolean semantics for STL composite formulas are derived from STL atomic propositions using Boolean conjunction and disjunction. For a given STL formula φ, we introduce binary variables z_t^{φ} to represent the Boolean semantics of φ at time $t = 0, \dots, H$; i.e., $z_t^{\varphi} = 1$ if φ holds at time t, 0 otherwise. The MILP encoding of z_t^{φ} using only Boolean operators can be derived inductively [21].

Negation $\varphi = \neg\varphi'$:

$$z_t^{\varphi} = 1 - z_t^{\varphi'} \tag{2}$$

Conjunction $\varphi = \bigwedge_{i=1}^{m} \varphi_i$:

$$\begin{aligned} z_t^{\varphi} &\leq z_t^{\varphi_i}, i = 1, \dots, m \\ z_t^{\varphi} &\geq 1 - m + \textstyle\sum_{i=1}^{m} z_t^{\varphi_i} \end{aligned} \tag{3}$$

Disjunction $\varphi = \bigvee_{i=1}^{m} \varphi_i$:

$$\begin{aligned} z_t^{\varphi} &\geq z_t^{\varphi_i}, i = 1, \dots, m \\ z_t^{\varphi} &\leq \textstyle\sum_{i=1}^{m} z_t^{\varphi_i} \end{aligned} \tag{4}$$

We now consider the encoding for STL formulas with temporal operators [21]. In particular, the *always* and *eventually* operators are respectively encoded as finite conjunctions and disjunctions using (3) and (4). Below, we use the notation $a_t^H = \min(a + t, H)$ and $b_t^H = \min(b + t, H)$. Note that a_t^H and b_t^H are not additional MILP variables.

Always $\varphi = \mathbf{G}_{[a,b]}\varphi'$: we encode z_t^{φ} as $\bigwedge_{i=a_t^H}^{b_t^H} z_t^{\varphi'}$.

Eventually $\varphi = \mathbf{F}_{[a,b]}\varphi'$: we encode z_t^{φ} as $\bigvee_{i=a_t^H}^{b_t^H} z_t^{\varphi'}$.

Until $\varphi = \varphi_1 \mathbf{U}_{[a,b]} \varphi_2$: the satisfaction of φ at t can be derived from those of the following formulas, including an unbounded $\mathbf{U}$, to achieve a linear encoding w.r.t. H [2]. In particular, we encode z_t^{φ} as $z_t^{\varphi'}$ given that φ is equivalent to φ', where

$$\varphi' = \mathbf{G}_{[0,a-1]}\varphi_1 \wedge \mathbf{F}_{[a,b]}\varphi_2 \wedge \mathbf{F}_{[a,a]}(\varphi_1 \mathbf{U}\, \varphi_2). \tag{5}$$

We note that $\varphi_1 \mathbf{U}_{[a,b]}\, \varphi_2$ holds if φ_1 holds before a, after which $\varphi_1 \mathbf{U}\, \varphi_2$ holds when φ_2 is satisfied before b. The first two conjuncts of (5) can be derived using the MILP encoding for the *always* and *eventually* operators. The unbounded *until* in the last conjunct is encoded as follows [21].

$$z_t^{\varphi_1 \mathbf{U}\, \varphi_2} = z_t^{\varphi_2} \vee (z_t^{\varphi_1} \wedge z_{t+1}^{\varphi_1 \mathbf{U}\, \varphi_2})$$

for all $t = 1, \dots, H - 1$, and $z_H^{\varphi_1 \mathbf{U}\, \varphi_2} = z_H^{\varphi_2}$.

(3) Resilient STL control objectives. Given an SRS expression $R_{\alpha,\beta}(\varphi)$, φ an STL formula, we introduce variables $c_t^{\varphi,rec}$ and $c_t^{\varphi,dur}$ (and associated MILP constraints) to encode $t_{rec}(\varphi, \vec{x}, t)$ and $t_{dur}(\varphi, \vec{x}, t)$, respectively. Inspired by the encoding in [24], $c_t^{\varphi,rec}$ and $c_t^{\varphi,dur}$ are defined as counters that, informally, keep track of the number of time units φ remains violated and satisfied, respectively.

$$c_t^{\varphi,rec} = (1 - z_t^{\varphi}) \cdot (c_{t+1}^{\varphi,rec} + 1), \qquad c_H^{\varphi,rec} = 0 \tag{6}$$

Variable $c_t^{\varphi,rec}$ is defined in reverse-temporal order; we first set $c_H^{\varphi,rec} = 0$. At time t, if $z_t^{\varphi} = 1$ (i.e., $(\vec{x}, t) \models \varphi$ holds), we have $c_t^{\varphi,rec} = 0$; if $z_t^{\varphi} = 0$ (i.e., $(\vec{x}, t) \models \varphi$ does not hold), we have $c_t^{\varphi,rec} = c_{t+1}^{\varphi,rec}+1$. Thus, if φ does not hold at time t, $c_t^{\varphi,rec}$ represents the time needed for φ to recover (or the time until the end of $\vec{x}$ if φ never recovers); or 0 if φ holds at t. We can see that $c_t^{\varphi,rec}$ follows exactly the definition of $t_{rec}(\varphi, \vec{x}, t)$ in Section 2.2, whereby if $(\vec{x}, t) \models \varphi$ holds, we have $t_{rec}(\varphi, \vec{x}, t) = 0$; otherwise, $t_{rec}(\varphi, \vec{x}, t)$ is the time needed for φ to recover from violation.

To define $c_t^{\varphi,dur}$, we employ additional counter variables $c_t^1, c_t^2 \in \mathbb{Z}_{\geq 0}$ for $t = 0, \dots, H$, which are similarly defined in reverse order as follows.

$$\begin{aligned} c_t^1 &= z_t^{\varphi} \cdot (c_{t+1}^1 + 1), & c_H^1 &= 0 \\ c_t^2 &= (1 - z_t^{\varphi}) \cdot (c_{t+1}^1 + c_{t+1}^2), & c_H^2 &= 0 \end{aligned} \tag{7}$$

At time t, if $z_t^{\varphi} = 1$, we have $c_t^1 = c_{t+1}^1 + 1$, meaning that c_t^1 counts how many time units φ remains true after t; if $z_t^{\varphi} = 0$, i.e., φ is false at t, we have $c_t^1 = 0$. At time t, if $z_t^{\varphi} = 1$, we have $c_t^2 = 0$; if $z_t^{\varphi} = 0$, we have $c_t^2 = c_{t+1}^1 + c_{t+1}^2$. This variable keeps track, when φ is false at t, for how long φ will remain true after the next recovery episode.

By the definition of $t_{dur}(\varphi, \vec{x}, t)$ in Section 2.2, if $(\vec{x}, t) \models \varphi$ holds, then $t_{dur}(\varphi, \vec{x}, t)$ is the time duration until a violation of φ (or the end of $\vec{x}$); if instead $(\vec{x}, t) \not\models \varphi$, $t_{dur}(\varphi, \vec{x}, t)$ refers to the duration-to-violation after the next recovery, and hence remains constant until recovery. We can see that c_t^1, c_t^2 respectively represent the behaviors of $t_{dur}(\varphi, \vec{x}, t)$ during satisfaction and violation of φ, thus

$$c_t^{\varphi,dur} = c_t^1 + c_t^2 \tag{8}$$

Remark 3. *The MILP encoding for resilience objectives involves the multiplication of binary variables and (integer) counter variables, which is nonlinear. Nonetheless, we can convert them to MILP inequality constraints using the translation of the* if-then-else *logic*

t	0	1	2	3	4	5	6	7	8	9
$\boldsymbol{\chi(\varphi, \vec{x}, t)}$	-1	1	1	-1	-1	-1	1	1	1	-1
z_t^{φ}	0	1	1	0	0	0	1	1	1	0
$c_t^1 = z_t^{\varphi} \cdot (c_{t+1}^1 + 1)$	0	2	1	0	0	0	3	2	1	0
$c_t^2 = (1 - z_t^{\varphi}) \cdot (c_{t+1}^1 + c_{t+1}^2)$	2	0	0	3	3	3	0	0	0	0
$\boldsymbol{c_t^{\varphi,rec} = (1 - z_t^{\varphi}) \cdot (c_{t+1}^{\varphi,rec} + 1)}$	1	0	0	3	2	1	0	0	0	0
$\boldsymbol{c_t^{\varphi,dur} = c_t^1 + c_t^2}$	2	2	1	3	3	3	3	2	1	0

Table 1: Example of MILP encoding $c_t^{\varphi,rec}$, $c_t^{\varphi,dur}$ of the resiliency objectives.

relation [4]. Let z be a binary variable and c an integer variable. Then the relation $y = z \cdot c$ is equivalent to the following inequalities:

$$\begin{aligned} m \cdot z \leq y \leq M \cdot z \\ c - M \cdot (1 - z) \leq y \leq c - m \cdot (1 - z) \end{aligned} \tag{9}$$

where M and m are the upper and lower bounds for c, respectively.

Table 1 provides an example of the encoding method. Consider $\vec{x}$ with $H = 9$ and STL formula φ; characteristic function $\chi(\varphi, \vec{x}, t)$ is given in the table. A step-by-step computation of $c_t^{\varphi,rec}$ and $c_t^{\varphi,dur}$ is provided; the results so obtained are the same as those for $t_{rec}(\varphi, \vec{x}, t)$ and $t_{dur}(\varphi, \vec{x}, t)$, computed using their definitions provided in Section 2.2. We define the function

$$(c_t^{\varphi,rec}, c_t^{\varphi,dur}, C_m) = \texttt{milp\_encoding}(\varphi, \vec{x})$$

that takes an STL formula φ and a sequence of system states $\vec{x}$ as input, and outputs the set of encoded MILP constraints C_m and encoded variables $c_t^{\varphi,rec}, c_t^{\varphi,dur}$ using Eqs. (1)-(8).

4.2 Multi-Objective Optimization

To address the challenge of multi-objective optimization, we propose an *a posteriori* method to reduce Problem 1 to a sequence of single-objective MILP instances (by optimizing one of the objectives and constraining the other above some given level) and to efficiently generate the exact set of non-dominated optimal solutions. The single-objectives MILP instances can be solved by standard techniques such as branch-and-bound methods [12].

A particular property of the resilient STL control problem is that both objective functions are discrete and bounded by the length of the horizon, so the number of optimal solutions is finite. Also, the optimal solutions are non-dominated with respect to the $\succ_{re}$ relation (which may not be the case with conventional Pareto dominance). We propose an algorithm for computing $\mathcal{S}^*$ taking these properties into account. First, we define the following problem.

$$P_\epsilon : \qquad \vec{u}_\epsilon^* = arg \max_{\vec{u}} \; t_{dur}(\varphi, \vec{x}, 0) - \beta$$

subject to the constraints of Problem 1 and an additional *ϵ-constraint* $\alpha - t_{rec}(\varphi, \vec{x}, 0) \geq \epsilon$. (One could equivalently maximize $\alpha - t_{rec}(\varphi, \vec{x}, 0)$ and constrain $t_{dur}(\varphi, \vec{x}, 0) - \beta \geq \epsilon$.) We respectively denote by f_ϵ^* and g_ϵ^* the values of $\alpha - t_{rec}(\varphi, \vec{x}, 0)$ and $t_{dur}(\varphi, \vec{x}, 0) - \beta$ corresponding to $\vec{u}_\epsilon^*$. Our algorithm consists of the following steps.

(1) Let $\mathcal{S}^* = \emptyset$ and $\epsilon = \alpha - (H - 1)$.
(2) If P_ϵ is feasible, go to step (3); otherwise, go to step (5).
(3) Solve P_ϵ, then $\mathcal{S}^* = \mathcal{S}^* \cup \{(f_\epsilon^*, g_\epsilon^*)\}$. Set $\epsilon = f_\epsilon^* + 1$.
(4) If $\epsilon < \alpha + 1$, go to step (2); otherwise, go to step (5).
(5) Return $\mathcal{S}^* = \max_{re} \mathcal{S}^*$.

Algorithm 1: Solution Method for Resilient STL Control

```
input  STL formula $\varphi$, control system $(F, G, x_0)$, control
       horizon $H$, time bounds $\alpha, \beta$.
output  The sets $\mathcal{S}^*$ and $\mathcal{U}_1^*$ of Problem 1.
Initialize $\mathcal{S}^* = \emptyset$ and $\epsilon = \alpha - (H - 1)$.
while $\epsilon < \alpha + 1$ do
    Let $\vec{u}$ be the decision variables.
    $(\vec{x}, C_s)$ = system_constraints($(F, G, x_0), \vec{u}$).
    $(c_0^{\varphi,rec}, c_0^{\varphi,dur}, C_m)$ = milp_encoding($\varphi, \vec{x}$).
    if $P_\epsilon$ is feasible then
        Solve $P_\epsilon$ as MILP and obtain $\vec{u}_\epsilon^*$, $f_\epsilon^*$ and $g_\epsilon^*$.
        $\mathcal{S}^* = \mathcal{S}^* \cup \{(f_\epsilon^*, g_\epsilon^*)\}$ and $\mathcal{U}_1^* = \mathcal{U}_1^* \cup \{\vec{u}_\epsilon^*\}$.
        $\epsilon = f_\epsilon^* + 1$.
    else
        $\epsilon = +\infty$.
    end if
end while
$\mathcal{S}^* = max_{re}\, \mathcal{S}^*$ and update $\mathcal{U}_1^*$ correspondingly.
Return $\mathcal{S}^*$ and $\mathcal{U}_1^*$.
```

The overall solution method is summarized in Algorithm 1. To solve problem P_ϵ, we encode it into MILP. To do so, we first generate the encoding for the control system through function `system_constraints`, which takes as input the system (F, G, x_0) and decision variables $\vec{u}$, and outputs the signal $\vec{x}$ (as a sequence of real variables) and the constraints C_s determined by the system dynamics. Second, we generate the encoding for the resiliency objectives through function `milp_encoding` as described in Section 4.1.

Proposition 4.1. *Algorithm 1 computes the exact set of optimal solutions $\mathcal{S}^*$ of Problem 1 in a finite number of steps.*

Proof. To prove the correctness of Algorithm 1, we first prove that the points in $\mathcal{S}^*$ upon entry to step (5) above include all Pareto-optimal solutions according to the traditional ordering $\succ$. Then we prove that step (5) computes the exact set of optimally resilient solutions (according to $\succ_{re}$).

To prove the first statement, it is enough to observe that at each iteration of the above while-loop, f_ϵ^* is strictly increasing and g_ϵ^* is non-increasing w.r.t. ϵ, meaning that the $(f_\epsilon^*, g_\epsilon^*)$ pair at one iteration either dominates or is mutually non-dominated by the one at the previous iteration (according to $\succ$). Hence, $\mathcal{S}^*$ include all (but not necessarily only) the Pareto-optimal solutions according to $\succ$.

For the second statement, we know that by Proposition 3.1, the maximum resilience set of the Pareto front is equivalent to that of the whole solution space. Thus, by performing $\max_{re} \mathcal{S}^*$, the output of Algorithm 1 is the set of optimal solutions (according to $\succ_{re}$). Algorithm 1 terminates in a finite number of steps both because it requires solving at most H instances of P_ϵ, and each instance terminates in a finite number of steps. □

Proposition 4.2. *In the worst case, Algorithm 1 computes H instances of problem P_ϵ.*

In the worst case, the number of increments of ϵ is H (see line 9 of Algorithm 1), resulting in H instances of solving P_ϵ at line 7.

4.3 Computation Complexity

The computation complexity of Algorithm 1 consists of two major sources: the MILP problem and the multi-objective problem. MILP problems are NP-hard and the computational complexity is highly dependent on the number of variables. In the worst case, a MILP problem solves a number of LP problems that are exponential in the number of binary and discrete variables. The complexity of LP is polynomial in the number of (real) variables.

Let φ be an STL formula with a set of atomic propositions AP. The Boolean semantics computation for STL atomic propositions introduces $O(H \cdot |AP|)$ binary variables; the Boolean semantics computation for φ introduces $O(H\cdot|\varphi|)$ binary variables [21]. Hence, the number of binary and discrete variables is $O((|AP| + |\varphi|) \cdot H)$. The MILP encoding for the resiliency objectives introduces exactly $3 \cdot H$ counter variables (i.e., $c_t^{\varphi,rec}, c_t^1, c_t^2$), hence this term is omitted. Note that the continuous variables are the sequences $\vec{x} \in \mathbb{R}^{n\cdot H}$ and $\vec{u} \in \mathbb{R}^{m\cdot H}$. Since we need to solve at most H MILP instances for P_ϵ, then the overall complexity is $O(H \cdot (2^{(|AP|+|\varphi|)\cdot H} \cdot (H \cdot (m+n))^k))$ for some $k \geq 1$. Note that computing $\max_{re} \mathcal{S}^*$ in step (5) adds a cost quadratic in H (see [6]) and hence is negligible compared to the overall complexity.

Remark 4 (Length of control horizon). *The length of the control horizon H of the resilient STL control problem should be carefully chosen. An excessively large H introduces unnecessary computational complexity without significant performance improvement: since we optimize recoverability and durability relative to the first recovery episode, what happens after the durability period does not affect these two objectives. On the other hand, insufficiently large H can make it difficult for the controller to provide an effective control action: if φ is initially violated and H is too small, it might be impossible to satisfy φ within H, and so all control strategies will have the same objective values (worst possible (rec, dur) values).*

5 CLOSED-LOOP CONTROL

In this section, we describe the remaining components of the ResilienC framework: the MPC control strategy and DMs that selects a single solution from the set of optimal solutions.

5.1 Model Predictive Control

In the MPC setting, at each time step t, we solve Problem 1 by setting x_0 to the current system state. The resilient STL controller computes a set of optimal solutions $\mathcal{S}^*$. A DM selects a solution from $\mathcal{S}^*$ and then implements only the first step of the corresponding optimal control strategy. The control system evolves following its dynamics. At time $t + 1$, x_0 is set to the evolved system state; the next implemented control action is calculated similarly using the resilient STL controller and a DM. This process is repeated at every remaining time step.

5.2 Decision-Maker Design

The optimal solutions $\mathcal{S}^*$ of Problem 1 is a set of non-dominated (rec, dur) pairs. At each step, an optimal solution is selected from $\mathcal{S}^*$ by a DM. We propose the following DM strategies representing different preferences in solution selection. ***Pro-recoverability DM***: selects the solution with maximum recoverability, representing a preference for rapid recovery. ***Pro-durability DM***: selects the solution with maximum durability, representing a preference for property maintenance post-recovery. ***Minimal-distance DM***: selects the optimal solution with minimal L_2-distance to the point $(\alpha, H - \beta)$ (the best attainable value of (rec, dur)). ***Adaptive DM***: respectively switches to pro-recoverability or pro-durability when maximum recoverability is less or greater than maximum durability. It represents a preference for the objective that is harder to achieve.

We note that a DM strategy can also represent application-specific preferences beyond recoverability and durability (e.g., the average distance to the centerline of the lane in a lane-keeping problem). We leave this extension for future work.

Remark 5 (a posteriori methods). *The current design of ResilienC uses an* a posteriori *method: the complete set of optimal solutions $\mathcal{S}^*$ is first computed, then a DM chooses one of them. However, we note that the above-defined minimal distance solution can be found without computing the Pareto front, but by solving the single-objective problem below:*

$$s^* = \min_{\vec{u}} ||(\alpha - t_{rec}, t_{dur} - \beta) - (\alpha, H - \beta)|| \qquad (10)$$

where $\alpha, H - \beta$ are upper bounds on $\alpha - t_{rec}, t_{dur} - \beta$. The solution s^* is Pareto-optimal according to $\succ$ because, if it was not, there would exist $s' \succ s^*$ with $s'_1 \geq s^*_1$ and $s'_2 \geq s^*_2$, of which at least one is a strict inequality. Hence, s' would be closer to $(\alpha, H - \beta)$ than s^*, which contradicts the fact that s^* is the optimal solution of (10). *However, s^* is not guaranteed to be an optimal solution to Problem 1, i.e., be Pareto-optimal according to $\succ_{re}$, unless we set $\alpha = 0$ and $\beta = H$. We note that the latter is a perfectly reasonable choice for the bounds, representing the strictest possible requirements for both recoverability and durability.*

6 CASE STUDIES

In this section, we demonstrate the benefits of the STL-based resilient controller via two case studies. Experiments were performed on an Intel Core i7-12700H CPU with 32GB of DDR5 RAM and a Windows 11 operating system. Our case studies have been implemented in MATLAB with YALMIP [15]; our implementation and case studies can be found in a publicly-available library. [6]

6.1 Lane Keeping

We study resilient control in a lane-keeping problem. We consider a linear, time-invariant single-track model for the vehicle with a constant nominal longitudinal speed [17]. The state-space representation of the model can be written as follows.

$$\dot{\boldsymbol{x}}_t = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & a_{c1} & 0 & a_{c2} \\ 0 & 0 & 0 & 1 \\ 0 & a_{c3} & 0 & a_{c4} \end{bmatrix} \boldsymbol{x}_t + \begin{bmatrix} 0 \\ \frac{2C_{\alpha F}}{m} \\ 0 \\ \frac{2l_F C_{\alpha F}}{I_z} \end{bmatrix} u_t, \quad x_0 = [5, 6, 0, 2]^T$$

where the state vector $\boldsymbol{x}_t = [y, y_v, \omega, \omega_v]^T$ with y being the lateral position, y_v the lateral velocity, ω the yaw angle, and ω_v the yaw

[6] See https://github.com/hongkaichensbu/resilient-stl-control

parameters	l_F	l_R	$C_{\alpha F}$	$C_{\alpha R}$	I_z	m	v
values	1.4	2.55	2200	2200	5757	2200	10
units	m	m	N/rad	N/rad	kg/m^2	kg	m/s

Table 2: Selected vehicle parameters.

velocity. Control actions u_t are steering angles of the vehicle, which are bounded by the physical limitation of the vehicle:

$$|u_t| \leq 0.72 \text{ rad}, \qquad |u_t - u_{t+1}| \leq 0.72 \text{ rad}$$

The parameters are defined as follows.

$$a_{c1} = -\frac{2C_{\alpha F} + 2C_{\alpha R}}{mv}, \qquad a_{c2} = -\frac{2l_F C_{\alpha F} - 2l_R C_{\alpha R}}{mv} - v,$$
$$a_{c3} = -\frac{2l_F C_{\alpha F} - 2l_R C_{\alpha R}}{I_z v}, \qquad a_{c4} = -\frac{2l_F^2 C_{\alpha F} + 2l_R^2 C_{\alpha R}}{I_z v}$$

where I_z is the inertial moment around the vehicle's z axis; l_F and l_R are the distances between the CoG and the front and rear axles respectively. The constants $C_{\alpha F}$ and $C_{\alpha R}$ are front and rear cornering stiffness; v is the constant nominal longitudinal speed. Our parameter selection is shown in Table 2. Letting $\Delta t = 0.1$ secs be the length of one time-step, we have $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \dot{\boldsymbol{x}}_t \cdot \Delta t$.

Lane Keeping Property: the vehicle should always remain within the lane boundaries in a time interval.

$$\varphi_{lk} = \mathbf{G}_{[0,h]}\,(p \wedge q) = \mathbf{G}_{[0,h]}\,((y_e \geq y_l) \wedge (y_e \leq y_u))$$

where y_e is the lateral difference between the vehicle and the center line of the lane. We set $y_u = 1$ m, $y_l = -1$ m, control horizon $H = 60$, $h = 2$, $\alpha = 1.8$ secs, and $\beta = 2.5$ secs. We apply ResilienC to our vehicle model on a curvy track.

We first evaluate the ResilienC solution method by using Algorithm 1 to compute the optimal solutions at the initial step. The resulting optimal solutions are $\mathcal{S}^* = \{(-0.2, 1.5), (0.1, -2.2), (0.2, -2.3)\}$; see Figure 3. In the top figure, the lane is indicated by the grey area and the starting location of the vehicle, marked by a star, is outside the lane. Each trajectory represents a predicted optimal trajectory for the vehicle and an optimal solution in $\mathcal{S}^*$. The middle figure shows the sequences of control actions for three optimal trajectories. The bottom figure shows the evolution of the lane-keeping requirement φ_{lk} over time for the three optimal trajectories.

We now compare the different behaviors of the three optimal solutions. In the top figure, trajectory 1 represents a situation where the vehicle enters the lane the latest, and yet it remains in the lane till the end of the trajectory. We can also see that from the blue solid line in the bottom figure, $z^{\varphi_{lk}}$ recovers to 1 later than the others ($t_{rec} = 2$ secs), but subsequently remains 1 for the longest duration ($t_{dur} = 4$ secs); it thus results in the optimal solution $(-0.2, 1.5)$. Trajectory 3 represents a vastly different situation where the vehicle aggressively enters the lane first and stays in the lane, but quickly exits the lane due to overshooting. The yellow dashed line in the bottom figure reflects this situation: $z^{\varphi_{lk}}$ recovers to 1 the earliest ($t_{rec} = 1.6$ secs) with the shortest subsequent duration ($t_{dur} = 0.2$ secs), resulting in the optimal solution $(0.2, -2.3)$. Trajectory 2 represents an intermediate situation: the vehicle returns to satisfy φ_{lk} with the second fastest recovery ($t_{rec} = 1.7$ secs) and the second longest subsequent duration ($t_{dur} = 0.3$ secs), yielding the optimal solution $(0.1, -2.2)$.

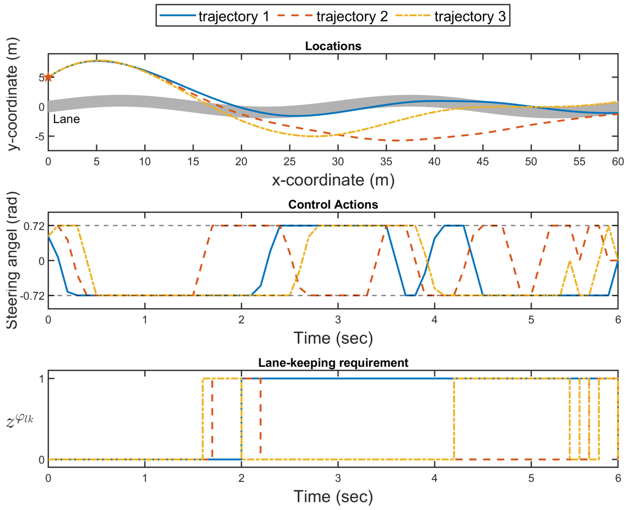


Figure 3: The optimal solutions provided by our ResilienC framework at the initial step in a lane-keeping problem.

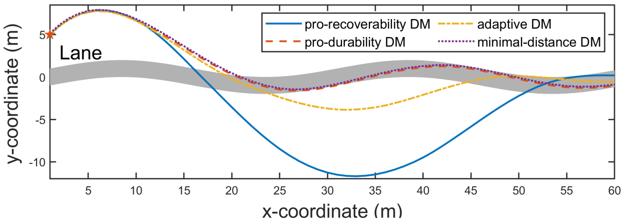


Figure 4: Simulated vehicle trajectories in ResilienC.

We then evaluate various DM strategies of our ResilienC framework in the MPC setting. Solving Problem 1 and selecting a control action took, on average, 78 msec on our machine. We roll out the MPC controller for a trajectory of 60 time steps for each DM. The (rec, dur) pair for the property φ_{lk} for the pro-recoverability DM, pro-durability DM, adaptive DM, and minimal-distance DM are respectively: $(0.3, -2.2)$, $(-0.2, 1.5)$, $(0, -2)$, and $(-0.2, 1.5)$. See the results in Figure 4. As expected, the trajectory generated by the pro-recoverability DM has better recoverability yet worse durability compared to the pro-durability DM. The trajectory generated with the adaptive DM has better recoverability compared to the pro-durability DM and better durability compared to the pro-recoverability DM, reflecting a balanced preference between recoverability and durability. The minimal-distance DM usually selects the same optimal solution as the pro-durability DM. This is because solutions with good recoverability often exhibit extremely bad durability due to overshooting, thus making them the farthest from the ideal resiliency value $(\alpha, H - \beta)$. This result evidences the usefulness of our approach in presenting the DM multiple, equally resilient, control strategies. In particular, we can see that optimizing for a fast recovery, which is roughly equivalent to maximizing STL time robustness, is not always the best strategy.

6.2 Deadline-Driven Package Delivery

We study a deadline-driven, multi-region cooperative package-delivery problem. The problem involves multiple controllable robots performing package deliveries by deadlines at multiple regions in a two-dimensional space. The robots are equipped with chargeable batteries.

We extend a robot model in [24] with a battery state component. In an N-robot system, we denote the state vector of the i-th robot as $\boldsymbol{x}^i = [l^i_x, v^i_x, y^i, l^i_y, e^i, v^i_e] \in \mathbb{R}^6$, where l^i_x, l^i_y are x, y coordinates, v^i_x, v^i_y are the x, y velocities components, e^i the battery level, and v^i_e the battery charging rate. The full state vector of the multi-robot system is $\boldsymbol{x} = [\boldsymbol{x}^1, \ldots, \boldsymbol{x}^N]^T$. Similarly, the control actions of the i-th robot are denoted by $\boldsymbol{u}^i = [u^i_1, u^i_2, u^i_3, e_{con}]$, where $u^i_1, u^i_2 \in \mathbb{R}$ associate to coordinates, $u^i_3 \in \{0, 1\}$ indicates the charging status, and $e_{con} = -1$ is the battery *consumption rate*; the control actions of the multi-robot system are $\boldsymbol{u} = [\boldsymbol{u}^1, \ldots, \boldsymbol{u}^N]^T$. The state-space representation of an N-robot system at time t is denoted as follows.

$$\boldsymbol{x}_{t+1} = F_N \cdot \boldsymbol{x}_t + G_N \cdot \boldsymbol{u}_t$$

where $\boldsymbol{x}_t$ and $\boldsymbol{u}_t$ are the system state and control actions, respectively. Matrices F_N and G_N are defined as follows.

$$F_N = I_N \otimes \begin{bmatrix} I_2 \otimes A & 0 & 0 \\ 0 & 1 & t_s \\ 0 & 0 & 0 \end{bmatrix}, \quad A = \begin{bmatrix} 1 & t_s \\ 0 & 1 \end{bmatrix}$$

$$G_N = \begin{bmatrix} B_1 & 0 & \ldots & 0 \\ 0 & B_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & B_N \end{bmatrix}, \; B_i = \begin{bmatrix} I_2 \otimes b & 0 & 0 \\ 0 & d_u & 0 \\ 0 & e^i_{ch} & 1 \end{bmatrix}, \; b = \begin{bmatrix} d_u \\ t_s \end{bmatrix}$$

where $\otimes$ is the Kronecker product and I_N is the identity matrix of size N. Parameter e^i_{ch} is the battery *charging rate* of the i-th robot, $t_s = 0.1$ is the time-step size, and $d_u = 0.005$.

Consider two robots (i.e., $N = 2$), denoted by $robot_1$ and $robot_2$, with $e^1_{ch} = 7$ and $e^2_{ch} = 2$ respectively representing an advanced and fast, and an outdated and slow, battery charging system.

Deadline-driven Package Delivery: A package must be delivered by a robot by a deadline at a delivery region. Each of the two rectangular delivery regions, R_1 and R_2, has two deadlines defined by the time intervals of the **F** operators.

$$\begin{aligned} \varphi_{s1} = \mathbf{F}_{[0,H/2]}\,(robot_1 \in R_1 \vee robot_2 \in R_1) \wedge \\ \mathbf{F}_{[H/2,H]}\,(robot_1 \in R_1 \vee robot_2 \in R_1) \\ \varphi_{s2} = \mathbf{F}_{[0,H/2]}\,(robot_1 \in R_2 \vee robot_2 \in R_2) \wedge \\ \mathbf{F}_{[H/2,H]}\,(robot_1 \in R_2 \vee robot_2 \in R_2) \end{aligned}$$

where $robot_i \in R_j$ indicates the i-th robot is inside R_j, $i = 1, 2$, $j = 1, 2$. The requirements $robot_i \in R_j$ can be expressed by a set of four linear constraints.[7]

Battery power requirement: the battery power of the robots should remain above $E_l = 10$.

$$\varphi_{c1} = (e^1 \geq E_l), \qquad \varphi_{c2} = (e^2 \geq E_l)$$

Robots' batteries can be charged in either of the two rectangular charging regions C_1 and C_2. Therefore, $u^1_3 = 1$ when $(robot_1 \in C_1) \vee (robot_1 \in C_2)$ holds, and 0 otherwise; similarly, $u^2_3 = 1$ if and only if $(robot_2 \in C_1) \vee (robot_2 \in C_2)$ holds. Constraints $robot_i \in C_j$ can be expressed as a set of four linear constraints similar to $robot_i \in R_j$. The overall requirement for the deadline-driven multi-region package delivery problem is defined as

$$\varphi_{del} = \varphi_{s1} \wedge \varphi_{s2} \wedge \varphi_{c1} \wedge \varphi_{c2}$$

We set $H = 60$, $\alpha = 25$, $\beta = 20$, $x^1_0 = [1.1, 0, 0.5, 0, 5, -1]$ and $x^2_0 = [7, 0, 2, 0, 13, -1]$. The vectors specifying the lower and upper bounds on x and those on y of R_1, R_2, C_1, and C_2 are $[0, 4, 7, 11]$, $[6, 10, 7, 11]$, $[0, 1, 0, 1]$ and $[9, 10, 0, 1]$, respectively. We restrict the control actions $||u^i_1||, ||u^i_2|| \leq 1$ for $i = 1, 2$.

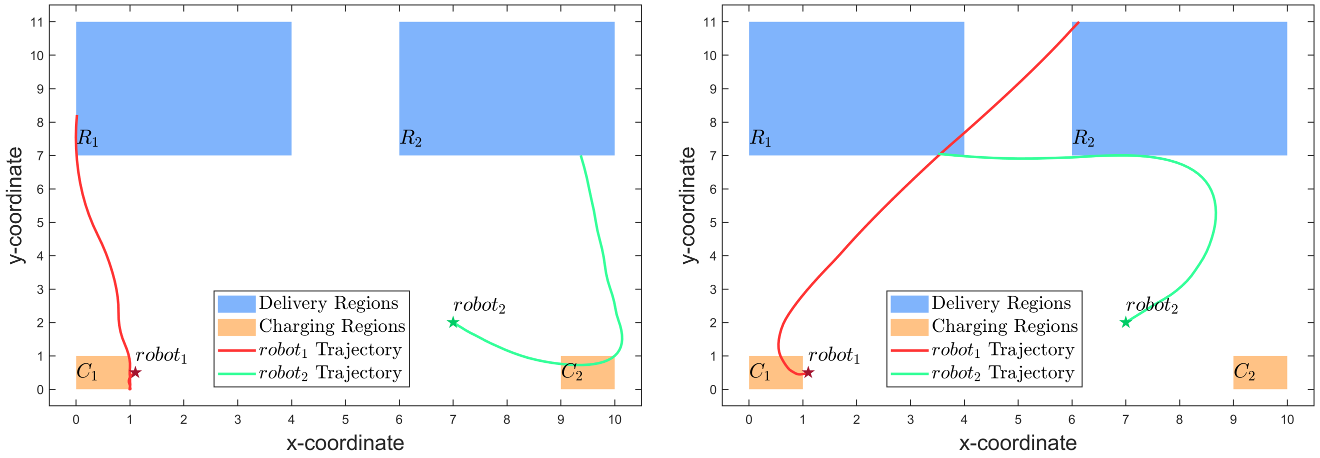
We first compute the optimal solutions at the initial time-step in our ResilienC framework using Algorithm 1. The optimal solutions are $\mathcal{S}^* = \{(-4, 4), (9, -5), (-3, 1)\}$. Figure 5 shows the first two optimal solutions. Also, Figure 6 shows the evolution of the systems: in each sub-figure, from left to right, the top row shows the charging status of $robot_1$, package-delivery status at R_1, and the battery power level of robots; the bottom row shows the charging status of $robot_2$, package-delivery status at R_2, and the overall problem requirement φ_{del}.

In Figure 5(a), an optimal situation needs $robot_2$ to go to C_2 for charging before package delivery at R_2 to ensure it has sufficient battery power for the deliveries. Meanwhile, $robot_1$ does not charge its battery in C_1 at full charging speed. This is because fast charging will not lead to quick satisfaction of φ_{del} because of the slow package delivery at R_2 by $robot_2$. In Figure 6(a), we can examine the system via the evolution of requirements: $robot_2 \in C_2$ holds between $t = 21$ and $t = 33$, after which $robot_2 \in R_2$ is true at $t = 60$; $robot_1 \in C_1$ holds irregularly, after which $robot_1 \in R_1$ is true at $t = 59$. Overall, in the bottom-right figure, φ_{del} is recovered late ($t_{rec} = 29$), but remains true for a long period of time ($t_{dur} = 24$); hence the solution $(-4, 4)$.

In contrast, Figure 5(b) depicts another optimal trajectory where $robot_2$ goes to R_2 and in turn R_1 for package deliveries without charging the battery, so to meet the deadlines. Meanwhile, $robot_1$ goes to C_1 and charges the battery at full charging speed to satisfy the battery power requirement as fast as possible; hence a quick recoverability w.r.t. φ_{del}. However, φ_{del} does not remain true as long as in the first trajectory because $robot_2$ never charges the battery and thus its battery power quickly drops below E_l. Figure 6(b) describes the system evolution: quick satisfaction of $robot_2 \in R_2$, $robot_1 \in R_1$, and $e^1 \geq E_l$ collectively create the best recoverability of φ_{del} ($t_{rec} = 16$). However, even though two package deliveries at R_1 and R_2 meet the deadlines after recovery, $e^2 \geq E_l$ cannot hold long enough because $robot_2 \in C_1$ or $robot_2 \in C_2$ never holds, causing the worst durability ($t_{dur} = 15$). Hence the solution $(9, -5)$.

We then evaluate our DM strategies. We roll out the MPC controller for 60 steps and assess the recoverability and durability of the trajectories w.r.t. φ_{del}. Solving Problem 1 and selecting a control action took, on average, 11.5 seconds on our machine. The computational complexity is due in large part to the extensive nature of the STL requirements needed for this case study. A strategy for reducing the execution time is under investigation as mentioned in the conclusion (see Section 8). The (rec, dur) pair of the trajectory with the pro-recoverability DM, the pro-durability DM, the adaptive DM, and the minimal-distance DM are respectively $(9, -5)$,

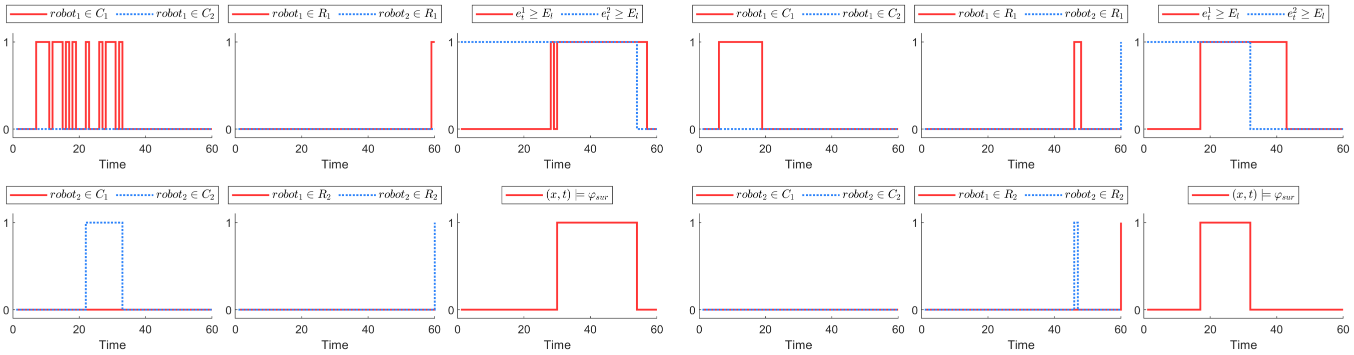
[7] For example, $robot_1 \in R_1$ requires the x^1 to be greater than R_1's lower bound on x, denoted by x_{ub}. Thus, we have $\boldsymbol{r}^1 \cdot l \geq x_{ub}$, where $l = [1, 0, 0, 0, 0, 0]^T$.



(a) Trajectory 1

(b) Trajectory 2

Figure 5: Optimal solutions provided by ResilienC at the initial step in a deadline-driven, multi-region package-delivery problem.



(a) Trajectory 1: φ_{del} has bad recoverability but good durability.

(b) Trajectory 2: : φ_{del} has good recoverability but bad durability.

Figure 6: Evolution of the system states of two trajectories.

$(-4, 4)$, $(2, -1)$, and $(-4, 4)$. As expected, the trajectories generated by the first three DM strategies respectively reflect a preference for recoverability, durability, and the recoverability-durability tradeoff. The minimal-distance DM shows a preference for good durability over good recoverability with overshooting.

7 RELATED WORK

Resilience in CPS. Logic-based formulations of resilience in CPS have been proposed. The time robustness semantics for STL is considered equivalent to resilience in [18]. However, it can only quantify the recoverability of STL violations but not the subsequent durability. Aksaray et al. [1] propose a "time shifting" STL and a resilient controller that maximizes the robustness value of the shifted formula as quickly as possible. This approach, however, does not consider the STL satisfaction durability post-recovery. Resilient control frameworks include work by Bouvier et al. [5] and Zhu et al. [27]. Their non-logic-based notions of resilience, however, do not readily lend themselves to systems subject to diverse and sophisticated temporal requirements. A survey on resilient multi-robot systems [20] discusses how resilience is defined, measured, and maintained across various robotics domains. Our work is based on the STL-based formulation of resiliency proposed by Chen et al. [6]. In this approach, the resilience of an STL formula takes into account both its recoverability and durability, which are quantified by sets of real-valued pairs.

Control under STL specifications. In [3], a controller synthesis problem is solved to ensure that the behavior of the resulting control system satisfies the desired STL specifications. The STL robustness controller [21] uses a MILP encoding of an optimization problem to maximize the space robustness [9] of the target STL specification. The controller synthesis problem for CPS subject to STL specifications has been considered in the context of reactive control [22], relaxed constrained MPC control [26], and "STL-based requirements priorities learning" via robustness slackness [8]. Extensions to the original robustness definition of STL specifications are used to tackle its disadvantages in optimization problems [10, 14].

Instead of space robustness, the time-robust control problem [24] focuses on right-time robustness, which is critical in the presence of timing uncertainty. It maximizes the right-time robustness of an STL specification of a discrete linear system. A left-right combined time robustness notion is proposed in [25] to address the weakness of left- and right-time robustness for a single-directional time shift. It also proposes a control algorithm for linear systems that maximizes the combined time robustness using MILP. An event-triggered MILP-based MPC framework [13] has been designed to maximize the overall space and time tolerances of the robustness degree of STL specifications for robot agents.

In contrast, we formally define the resilient STL control problem for CPS with STL-based requirements as one that maximizes recoverability and durability, resulting in a multi-objective optimization problem. To the best of our knowledge, we are the first to consider a resilient control framework that co-maximizes recoverability and durability.

8 CONCLUSION

We presented *ResilienC*, a resilient control framework for CPS subject to STL requirements. In ResilienC, we defined the problem of resilient control as one of multi-objective optimization that maximizes both CPS recoverability and durability w.r.t the desired STL properties. We proposed a solution method that uses a MILP encoding and an a posteriori method for computing the precise set of non-dominated optimal solutions. Each optimal solution represents an optimally resilient trajectory of the control system. We also proposed a number of DM strategies that represent various preferences for selecting a single optimal solution. We illustrated ResilienC on two case studies: lane keeping and deadline-driven multi-region package delivery. Collectively, our results showed the effectiveness of our solution methods in achieving resilient control and demonstrated the effects of DM preferences.

Future work will consider application-specific DM strategies that go beyond recoverability and durability; e.g., in the lane-keeping problem, the average distance to the centerline of the lane. We will also investigate learning a *neural controller* (NC) for our ResilienC setup. The controller presented in this paper can be run repeatedly in simulation mode to provide the training data for the NC. Approaches of this nature can be found in [7, 19]. Such an NC is expected to improve upon the execution time of the MPC controller (which has to solve a multi-objective MILP problem at every time step) for the package-delivery case study by orders of magnitude.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their valuable feedback and suggestions for improving the quality of this paper. Research supported in part by NSF CNS-1952096, CNS-1553273 (CAREER), OIA-2134840, OIA-2040599, CCF-1918225, and CPS-1446832.

REFERENCES

[1] Derya Aksaray. 2021. Resilient Satisfaction of Persistent and Safety Specifications by Autonomous Systems. In *AIAA Scitech 2021 Forum*. AIAA, Virtual Conference, 1124–1134. https://doi.org/10.2514/6.2021-1124

[2] Ezio Bartocci, Jyotirmoy Deshmukh, Alexandre Donzé, Georgios Fainekos, Oded Maler, Dejan Ničković, and Sriram Sankaranarayanan. 2018. Specification-based Monitoring of Cyber-Physical Systems: A Survey on Theory, Tools and Applications. *Lectures on Runtime Verification: Introductory and Advanced Topics* 10457 (2018), 135–175. https://doi.org/10.1007/978-3-319-75632-5_5

[3] Calin Belta and Sadra Sadraddini. 2019. Formal Methods for Control Synthesis: An Optimization Perspective. *Annual Review of Control, Robotics, and Autonomous Systems* 2 (2019), 115–140. https://doi.org/10.1146/annurev-control-053018-023717

[4] Alberto Bemporad, Fabio Danilo Torrisi, and Manfred Morari. 2001. Discrete-time Hybrid Modeling and Verification of the Batch Evaporator Process Benchmark. *European Journal of Control* 7, 4 (2001), 382–399. https://doi.org/10.3166/ejc.7.382-399

[5] Jean-Baptiste Bouvier, Kathleen Xu, and Melkior Ornik. 2021. Quantitative Resilience of Linear Driftless Systems. In *Proceedings of the Conference on Control and its Applications*. SIAM, SIAM, Virtual Conference, 32–39. https://doi.org/10.1137/1.9781611976847.5

[6] Hongkai Chen, Shan Lin, Scott A. Smolka, and Nicola Paoletti. 2022. An STL-based Formulation of Resilience in Cyber-Physical Systems. In *Proceedings of the 20th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS '22)*. Springer, Warsaw, Poland, 117–135. https://doi.org/10.1007/978-3-031-15839-1_7

[7] Hongkai Chen, Nicola Paoletti, Scott A Smolka, and Shan Lin. 2021. MPC-guided Imitation Learning of Bayesian Neural Network Policies for the Artificial Pancreas. In *2021 60th IEEE Conference on Decision and Control (CDC '21)*. IEEE, Austin, TX, USA, 2525–2532. https://doi.org/10.1109/CDC45484.2021.9683240

[8] Kyunghoon Cho and Songhwai Oh. 2018. Learning-based Model Predictive Control under Signal Temporal Logic Specifications. In *2018 IEEE International Conference on Robotics and Automation (ICRA '18)*. IEEE, Brisbane, QLD, Australia, 7322–7329. https://doi.org/10.1109/ICRA.2018.8460811

[9] Alexandre Donzé and Oded Maler. 2010. Robust Satisfaction of Temporal Logic over Real-Valued Signals. In *Proceedings of International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS '10)*. Springer, Klosterneuburg, Austria, 92–106. https://doi.org/10.1007/978-3-642-15297-9_9

[10] Iman Haghighi, Noushin Mehdipour, Ezio Bartocci, and Calin Belta. 2019. Control from Signal Temporal Logic Specifications with Smooth Cumulative Quantitative Semantics. In *2019 IEEE 58th Conference on Decision and Control (CDC '19)*. IEEE, Nice, France, 4361–4366. https://doi.org/10.1109/CDC40024.2019.9029429

[11] Marco Laumanns, Lothar Thiele, and Eckart Zitzler. 2006. An efficient, adaptive parameter variation scheme for metaheuristics based on the epsilon-constraint method. *European Journal of Operational Research* 169, 3 (2006), 932–942. https://doi.org/10.1016/j.ejor.2004.08.029

[12] Eugene L Lawler and David E Wood. 1966. Branch-and-bound methods: A survey. *Operations research* 14, 4 (1966), 699–719. https://doi.org/10.1287/opre.14.4.699

[13] Zhenyu Lin and John S Baras. 2020. Optimization-based Motion Planning and Runtime Monitoring for Robotic Agent with Space and Time Tolerances. *IFAC-PapersOnLine* 53, 2 (2020), 1874–1879. https://doi.org/10.1016/j.ifacol.2020.12.2606

[14] Lars Lindemann and Dimos V Dimarogonas. 2019. Robust control for signal temporal logic specifications using discrete average space robustness. *Automatica* 101 (2019), 377–387. https://doi.org/10.1016/j.automatica.2018.12.022

[15] Johan Löfberg. 2004. YALMIP: A Toolbox for Modeling and Optimization in MATLAB. In *Proceedings of IEEE International Symposium on Computer Aided Control Systems Design (CACSD '04)*. IEEE, Taipei, Taiwan, 284–289. https://doi.org/10.1109/CACSD.2004.1393890

[16] Oded Maler and Dejan Nickovic. 2004. Monitoring Temporal Properties of Continuous Signals. In *Proceedings of Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems (FTRTFT '04)*. Springer, Grenoble, France, 152–166. https://doi.org/10.1007/978-3-540-30206-3_12

[17] Sara Mata, Asier Zubizarreta, and Charles Pinto. 2019. Robust Tube-Based Model Predictive Control for Lateral Path Tracking. *IEEE Transactions on Intelligent Vehicles* 4, 4 (2019), 569–577. https://doi.org/10.1109/TIV.2019.2938102

[18] Noushin Mehdipour. 2021. *Resilience for Satisfaction of Temporal Logic Specifications by Dynamical Systems*. Ph. D. Dissertation. Boston University. https://open.bu.edu/handle/2144/41871

[19] Usama Mehmood, Shouvik Roy, Radu Grosu, Scott A Smolka, Scott D Stoller, and Ashish Tiwari. 2020. Neural Flocking: MPC-based Supervised Learning of Flocking Controllers. *Foundations of Software Science and Computation Structures. FoSSaCS 2020. Lecture Notes in Computer Science* 12077 (2020), 1–16. https://doi.org/10.1007/978-3-030-45231-5_1

[20] Amanda Prorok, Matthew Malencia, Luca Carlone, Gaurav S Sukhatme, Brian M Sadler, and Vijay Kumar. 2021. Beyond Robustness: A Taxonomy of Approaches towards Resilient Multi-Robot Systems. arXiv preprint. https://doi.org/10.48550/arXiv.2109.12343

[21] Vasumathi Raman, Alexandre Donzé, Mehdi Maasoumy, Richard M. Murray, Alberto Sangiovanni-Vincentelli, and Sanjit A. Seshia. 2014. Model Predictive Control with Signal Temporal Logic Specifications. In *Proceedings of the IEEE Conference on Decision and Control (CDC '14)*. IEEE, Los Angeles, USA, 81–87. https://doi.org/10.1109/CDC.2014.7039363

[22] Vasumathi Raman, Alexandre Donzé, Dorsa Sadigh, Richard M Murray, and Sanjit A Seshia. 2015. Reactive Synthesis from Signal Temporal Logic Specifications. In *Proceedings of the 18th international conference on hybrid systems: Computation and control (HSCC '15)*. ACM, Seatle, USA, 239–248. https://doi.org/10.1145/2728606.2728628

[23] James B. Rawlings. 2000. Tutorial Overview of Model Predictive Control. *IEEE Control Systems Magazine* 20, 3 (2000), 38–52. https://doi.org/10.1109/37.845037

[24] Alena Rodionova, Lars Lindemann, Manfred Morari, and George J. Pappas. 2021. Time-Robust Control for STL Specifications. In *Proceedings of IEEE Conference on Decision and Control (CDC '21)*. IEEE, Austin, USA, 572–579. https://doi.org/10.1109/CDC45484.2021.9683477

[25] Alëna Rodionova, Lars Lindemann, Manfred Morari, and George J Pappas. 2022. Combined Left and Right Temporal Robustness for Control under STL Specifications. *IEEE Control Systems Letters* 7 (2022), 619–624. https://doi.org/10.1109/LCSYS.2022.3209928

[26] Sadra Sadraddini and Calin Belta. 2015. Robust Temporal Logic Model Predictive Control. In *2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton '15)*. IEEE, Monticello, IL, USA, 772–779. https://doi.org/10.1109/ALLERTON.2015.7447084

[27] Quanyan Zhu and Tamer Başar. 2011. Robust and Resilient Control Design for Cyber-Physical Systems with an Application to Power Systems. In *Proceedings of IEEE Conference on Decision and Control and European Control Conference (CDC-ECC '11)*. IEEE, Orlando, FL, USA, 4066–4071. https://doi.org/10.1109/CDC.2011.6161031