Metascience in Aerospace

https://www.aimspress.com/journal/mina

Metascience in Aerospace, 2(3): 68–88.
DOI:10.3934/mina.2025004
Received: 20 February 2025
Revised: 02 July 2025
Accepted: 07 August 2025
Published: 22 August 2025

Research article

Multi-task learning for fast online adaptation under signal temporal logic specifications: An air traffic management application

Andres Arias*and Chuangchuang Sun

Department of Mechanical Engineering, Villanova University, Villanova, PA, USA

* **Correspondence:** Email: aariaslo@villanova.edu.

Abstract: Multi-task learning (MTL) seeks to improve the generalized performance during new testing tasks by exploiting useful information incorporated in related tasks. This paper developed an MTL-based control framework that considers signal temporal logic (STL) specifications applied to air traffic management. MTL settings involve both training and testing stages. In the training stage, an ensemble of tasks is generated by perturbing STL specifications considering the spatiotemporal features, ensuring a diverse yet structured task distribution. Task compliance is measured using the robustness degree, which is computed through STL semantics to quantify how well a state variables trajectory satisfies a set of specifications. Furthermore, robustness degree expressions are inherently non-convex due to log-sum exponential (LSE) terms used to approximate min and max operators, making direct optimization challenging. To address this, we employed sequential convex programming (SCP) with dynamically updated trust regions, which iteratively refine solutions to ensure convergence and stability in the optimization process. In the testing stage, new unseen tasks were introduced, requiring adaptation based on prior knowledge. The optimal solution obtained from the training phase served as a warm start, enabling rapid/few-shot adaptation to new, more perturbed tasks. Experimental validation was performed on two different dynamical systems. The first system corresponds to the spring-mass-damper system. The second system corresponds to a more comprehensive application within the air traffic control problem, scaled down to the context of quad-rotor dynamics. The results demonstrate that the proposed framework successfully satisfies new tasks' specifications within only a few SCP iterations, even under significant perturbations, highlighting the efficiency and adaptability of our approach.

Keywords: air traffic control; multi-task learning; optimal control; sequential convex programming, signal temporal logic; task generator; UAVs traffic management

Nomenclature

Sets

$\mathcal{X}$	Set of allowable values for the state variables.
$\mathcal{U}$	Set of allowable values for the control inputs.
$\underline{\mathcal{T}}$	Set of lower bounds for temporal parameters in the task formulas.
$\overline{\mathcal{T}}$	Set of upper bounds for temporal parameters in the task formulas.
$\mathcal{C}$	Set of spatial parameters that describe the formulas in a task.
$\mathcal{O}$	Set of temporal operators used in the formulas of the task.
$\mathcal{H}$	Set of predicates used in the formulas of the task.
$\mathcal{B}$	Set of Boolean operators used in the formulas of the task.

Parameters and Constants

N_T	Number of time steps along the time horizon.
x_0	Initial value of the state variables x.
x_{init}	Initial trajectory of the state variables.
u_{init}	Initial sequence of values for the control inputs.
N_h	Maximum number of sequential convex programming iterations.
M_L	Number of tasks in the training stage.
M_T	Number of tasks in the testing stage.
α	Linear quadratic regulator weight factor for the optimal control problem.
Q	Positive semidefinite weight matrix for the state variables in the linear quadratic regulator.
R	Positive semidefinite weight matrix for the control inputs in the linear quadratic regulator.
K	Constant used in the log-sum exponential approximation for max and min operators.
M	Number of tasks to create in the task generator procedure.
N	Number of formulas used to create the task in conjunctive normal form. In nested form, N corresponds to the number of levels of the task.
$\mathcal{S}$	Spatial parameter of the nested task in the innermost level.
η	Trust region in the sequential convex programming.
x_{ref}	Reference point of the state variables in the linear quadratic regulator settings.

Decision Variables

x	State variables in the system dynamics f.
u	Control inputs that determine the behavior of the system dynamics along the time horizon.

Expressions

$f(x, u)$	System dynamics with state variable x and control input u.
$\rho(x)$	Robustness degree of the task in terms of the state variables x.
$\hat{f}(x, u)$	Linearized system dynamics.
ρ_{av}	Average robustness degree for the tasks used in the training stage.

1. Introduction

New challenges are present with the increasing number of unmanned aerial vehicles (UAVs) involved in air traffic control (ATC), which leverages safe flight operations in controlled airspaces. From the ATC perspective, the most significant problem is the collision between two aircraft or the aircraft with surrounding obstacles. In the particular case of UAVs, a sufficient guarantee must be provided to integrate the UAVs in the conventional airspace [1], since the autonomous operation has emerged as a task-strategic framework for applications such as goods delivery, aerial mapping, agriculture, and surveillance. Under the context of the UAVs Traffic Management (UTM), smooth operation of UAVs is required in the open air space in high-density urban cities [2]. In this sense, tasks related to obstacle avoidance, reaching way-points, and flight altitude limits, are part of the learning process of the UAV's task-compliance to establish a harmonized autonomous operation in the airspace. This paper addresses efficient learning and control of aerial vehicle systems.

In the learning process, developing a new skill is based on prior knowledge obtained based on other related skills. This principle leads to learning multiple tasks simultaneously, from which transferrable skills can be obtained and utilized for new tasks. This knowledge transfer mechanism allows an individual to learn new concepts with limited data. Motivated by humans' ability to learn, multi-task learning (MTL) aims to learn multiple tasks simultaneously such that the experience obtained when developing one task gathers useful information that can be used to generalize the performance for other related tasks [3]. MTL is especially useful for transferring skills to situations with limited data for online fast adaptation, since a performant controller is often trained by using a noticeable amount of data [4]. This data scarcity issue is more severe in applications requiring manual labor to label data, i.e., medical image analysis, speech recognition, and natural language processing [5]. Furthermore, when the available labeled data is very limited for training purposes, few-shot learning (FSL) results in a useful approach that corresponds to the ability of the machine learning models to generalize from only a few training examples [6]. Nevertheless, MTL is not typically classified within the FSL framework, as its core aim involves learning multiple tasks simultaneously, considering that these tasks share a common representation, rather than necessitating few training examples.

Additionally, MTL not only improves the performance of specific tasks but also increases the adversarial robustness, decreasing the model vulnerabilities under adversarial attacks [7]. Early contributions presented in [8] defined MTL as an inductive transfer mechanism to improve the generalization performance by leveraging the domain-specific information contained in training signals of related tasks. The training signals process is developed in parallel using a shared representation. Different approaches have been proposed for MTL, from multi-objective optimization [9] and dynamic task prioritization [10], to task-dependent weighting [11]. With the introduction of deep learning [12], significant progress has been made during the last two decades for MTL.

MTL can be properly formulated using signal temporal logic (STL) to incorporate the temporal behavior of tasks in autonomous systems due to its expressiveness and interoperability from its similarity to natural language. The task is defined by one or several formulas with spatiotemporal parameters and logical operators. The symbolic control problem is composed of a continuous-time signal trajectory that meets the task formulas, providing a quantitative notion of time and space [13]. Several studies

have been conducted on the STL application in the optimal control context. This encompasses robust control, control barrier function, and control Lyapunov function, applied in energy management systems, temperature control, autonomous driving, and trajectory control for unmanned ground and aerial vehicles. Reference [14] introduced and maximized the robustness degree, a metric to measure how robustly the formulas are satisfied or violated, using smooth approximations of STL semantics. The methodology is applied to the control of heating, ventilation, and air conditioning buildings. Similarly, in [13] a new robustness under-approximation was developed, based on arithmetic and geometric means that makes the positive robustness a sufficient condition to meet the formula. STL has been extended to the time-varying control barrier function (CBF) and its temporal properties within STL task satisfaction. A control law is derived so that the state variables' solution with an initial condition met the specification. This approach is especially useful in multi-robot systems, where some methods do not scale computationally [15]. In [16] a methodology was proposed to construct a CBF for a fragment of STL tasks by solving an optimization problem, applied in dynamically coupled multi-agent systems. Other approaches learn a neural network controller to satisfy an STL specification. The parameters of the CBF are obtained through training, substantially decreasing the conservatism [17]. Considering the trajectory planning for continuous linear systems, a mixed integer quadratic program was formulated in [18] with STL constraints with linear predicates and a CBF, to synthesize a discrete sequence of controls. In [19], the STL formulas considered nonlinear state constraints using high order control Lyapunov barrier functions. This led to developing controllers for system stabilization inside a set within a specified time while guaranteeing the system to remain in the set.

In this article, an MTL-based approach for fast online adaptation with STL formulas is developed using STL semantics and robustness degree maximization. Due to the non-convex terms that arise from the STL semantics, such as the log-sum exponential approximation for the min and max operators, a sequential convex programming (SCP) algorithm is used to solve the non-convex problem, considering the system dynamics and the trust region as constraints. There are two stages in the methodology: the training stage, in which the SCP finds an optimal solution for multiple tasks perturbed under the nominal STL formula, given that the robustness degree of the final solution must be positive for task compliance, and the testing stage, where new related tasks are generated by applying a larger perturbation compared with the training stage. The solution obtained in the training stage is the initializer for the testing stage, solving the new perturbed tasks with only a few SCP iterations. To the best of our knowledge, our work is the first to apply MTL under the STL framework, since STL-based optimal control problems can be properly addressed with MTL principles.

The contributions of this work are summarized as follows:

- Introduction of a novel approach that applies multi-task learning under the optimal control context using signal temporal logic specifications.
- Utilizing sequential convex programming to efficiently solve STL-based optimal control problems, considering smooth approximations of max and min operators with log-sum exponential expressions.
- Development of a two-stage learning framework for STL-based optimal control problems. In the first stage, the system is trained over a set of perturbed tasks while ensuring a positive robustness degree. In the second stage, the learned initialization enables fast adaptation to new, more perturbed tasks.
- Practical application on two case studies, e.g., the spring-mass-damper system and air traffic

control for UAVs, demonstrating the feasibility and efficiency of applying MTL principles to STL-based optimal control problems.

The remainder of this article is organized as follows. Section 2 presents the mathematical formulation, considering the optimal control problem and how the STL framework is introduced in the robustness degree. In Section 3 the problem formulation and solution methodology is explained, within the SCP algorithm and MTL training and testing stages. The results of two case studies are addressed in Section 4. Finally, the concluding remarks and some avenues of research are presented in Section 5.

2. Preliminaries and mathematical formulation

Let $x \in \mathbb{R}^n$ and $u \in \mathcal{U} \subseteq \mathbb{R}^m$ be the state and control input of a discrete-time system presented as follows:

$$x_{k+1} = f(x_k, u_k) \tag{2.1}$$

where $f(x_k, u_k)$ represents the discretized system dynamics. Given the initial state x_0 and a control sequence $(u_0, u_1, ..., u_{N_T-1})$, a trajectory $(x_0, x_1, ..., x_{N_T})$ that satisfies (2.1) can be generated, and in this work is named *signal*.

2.1. Signal temporal logic (STL) and the robustness degree

STL is a formal language to describe a broad range of real-valued, temporal properties in cyber-physical systems [20]. An STL formula comprises predicates, logical connectives, and temporal operators, using the atomic propositions presented in the Backus-Naur form, as shown below:

$$\phi := True \left| \mathcal{B}_\S \geq 0 \right| \sim \phi \,|\phi_1 \wedge \phi_2|\, \phi_1 \mathsf{U}_{[a,b]} \phi_2 \,|\phi_1 \Rightarrow \phi_2| \tag{2.2}$$

where $\sim$ means "negation", $\wedge$ is the "And" operator, U means "Until", $[a, b]$ is the time interval from a to b, and ϕ_1 and ϕ_2 are STL formulas. From the elementary propositions above, additional operators can be written, such as $\phi_1 \vee \phi_2$, $\mathsf{F}_{[a,b]}\phi$, $\mathsf{G}_{[a,b]}\phi$, and $\phi_1 \Rightarrow \phi_2$, that stand for "Or", "Eventually", "Always", and "Infer" operators, respectively. The predicate $\mathcal{B}_\S$ is a function h in terms of the signal x, which is expressed as $h(x) - C_\phi$. If $\mathcal{B}_\S \geq 0$, then $\mathcal{B}_\S$ is True, otherwise, $\mathcal{B}_\S$ is False. Considering that $(x, t) \models \phi$ indicates if the signal x satisfies ϕ at time t, the formula satisfaction is recursively computed in the formula semantics, as presented in [18] and shown below:

$$\begin{aligned}
&(x,t) \models \mathcal{B}_\S \quad \Leftrightarrow \quad h(x(t)) \geq 0 \\
&(x,t) \models \sim \phi \quad \Leftrightarrow \quad \sim ((x,t) \models \phi) \\
&(x,t) \models \phi_1 \wedge \phi_2 \quad \Leftrightarrow \quad (x,t) \models \phi_1 \wedge (x,t) \models \phi_2 \\
&(x,t) \models \phi_1 \mathsf{U}_{[a,b]} \phi_2 \quad \Leftrightarrow \quad \exists t_1 \in [t+a, t+b] \; s.t. \\
&\qquad\qquad\qquad\qquad\qquad (x,t_1) \models \phi_2 \wedge \forall t_2 \in [t, t_1], \\
&\qquad\qquad\qquad\qquad\qquad (x,t_2) \models \phi_1 \\
&(x,t) \models \phi_1 \Rightarrow \phi_2 \quad \Leftrightarrow \quad \text{if } (x,t) \models \phi_1 \text{ then } (x,t) \models \phi_2
\end{aligned} \tag{2.3}$$

To measure how well the STL formula is satisfied by the signal x, a quantitative semantics named the robustness degree, denoted by ρ, is used. If $\rho > 0$, the STL formula is satisfied, and $\rho < 0$ otherwise. If

$\rho = 0$, the formula satisfaction is inconclusive. The robustness of ϕ with respect to signal x at time t is recursively defined as follows:

$$\begin{aligned}
\rho^{\mathcal{B}_{\S}}(x,t) &= h(x) - C_{\phi} \\
\rho^{\sim\phi}(x,t) &= -\rho^{\phi}(x,t) \\
\rho^{\phi_1 \wedge \phi_2}(x,t) &= \min\left(\rho^{\phi_1}(x,t), \rho^{\phi_2}(x,t)\right) \\
\rho^{\phi_1 \vee \phi_2}(x,t) &= \max\left(\rho^{\phi_1}(x,t), \rho^{\phi_2}(x,t)\right) \\
\rho^{\mathsf{F}_{[a,b]}\phi}(x,t) &= \max_{t' \in [t+a, t+b]}\left(\rho^{\phi}(x,t)\right) \\
\rho^{\mathsf{G}_{[a,b]}\phi}(x,t) &= \min_{t' \in [t+a, t+b]}\left(\rho^{\phi}(x,t)\right) \\
\rho^{\phi_1 \Rightarrow \phi_2}(x,t) &= \max\left(-\rho^{\phi_1}(x,t), \rho^{\phi_2}(x,t)\right)
\end{aligned} \tag{2.4}$$

2.2. Mathematical model

Given the STL semantics, the following optimal control problem is formulated in (2.5) along a horizon N_T, maximizing ρ subject to the system dynamics. The objective is to find a control input and state variable sequence that satisfies a set of formulas.

$$\begin{aligned}
&\max_{x,u} \rho(x) - \alpha \sum_{k=1}^{N_T} \left[x_k^{\top} Q x_k + u_k^{\top} R u_k\right] \\
s.t.\ \ &x_{k+1} = f(x_k, u_k), \forall k \in \{0, 1, ..., N_T - 1\} \\
&x_k \in \mathcal{X}, u_k \in \mathcal{U},
\end{aligned} \tag{2.5}$$

where Q and R are positive semidefinite weight matrices for the state variables and control inputs, respectively, to form the linear quadratic regulator (LQR) term weighted with α. Although this term penalizes the objective function, it plays an important role in stabilizing the signal trajectory.

Additionally, the computation of ρ^{ϕ} often lacks smoothness due to the max and min operators commonly used in robustness degree semantics. A smooth (infinitely differentiable) approximation is achieved by replacing the max and min operators in ρ^{ϕ} with the log-sum-exponential (LSE) functions [21], as shown as follows:

$$\widetilde{\max}\left[(a_1, ..., a_m)\right]^{\top} := \frac{1}{K} \log\left(\sum_{i=1}^{m} e^{k a_i}\right) \tag{2.6}$$

$$\widetilde{\min}\left[(a_1, ..., a_m)\right]^{\top} := -\frac{1}{K} \log\left(\sum_{i=1}^{m} e^{-k a_i}\right) \tag{2.7}$$

The max and min operators' approximation in (2.6) and (2.7) are smooth when using LSE expressions, and the gradient can be analytically derived. As the parameter $K \to \infty$, the maximum and minimum values approach to their true values. Although the resulting mathematical model is still non-convex, employing convex approximations such as Taylor series expansion and SCP allows us to obtain a positive local maximum.

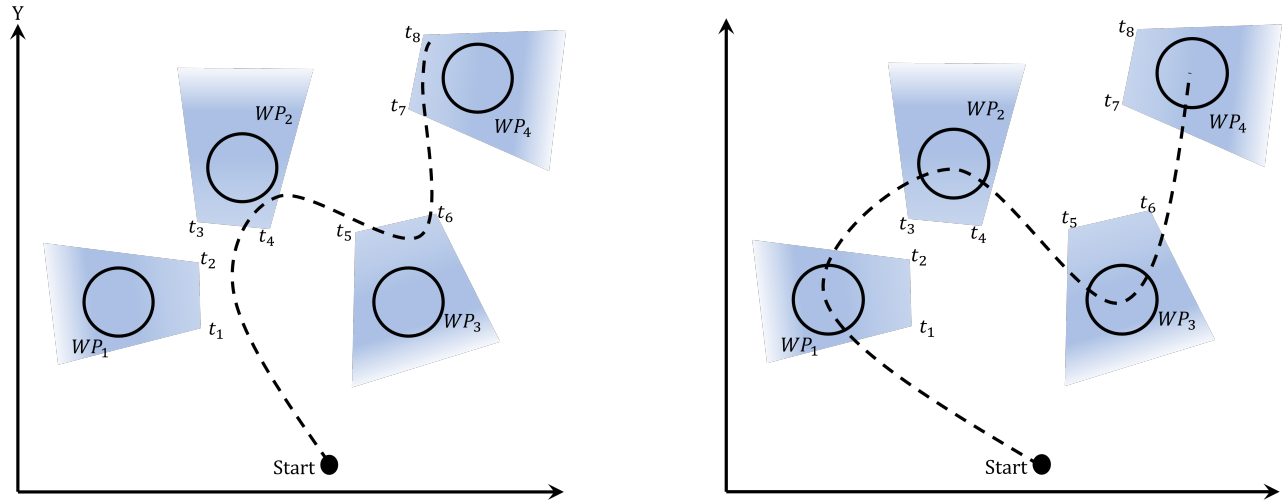
3. Problem formulation and implementation of multi-task learning on STL specifications

An STL formula ϕ is composed by a predicate $\mathcal{B}_{\S}$ of the form $h(x) - C_\phi$, where $h(x)$ and C_ϕ describe the behavior of the state variables in the task in the standard form of $h(x) - C_\phi \geq 0$. The function $h(x)$ depends on one or several state variables. Additionally, a time interval $[t_a, t_b]$ is also included, in which the formula $h(x) - C_\phi \geq 0$ should be satisfied. The time interval $[t_a, t_b]$ can encompass the entire horizon N_T or a subset of N_T. In general, the parameters C_ϕ, t_a, and t_b are the spatial and temporal parameters of the STL formula, respectively, formally expressed as $\phi_{[t_a,t_b]}$.

Example 1. *A robot is required to traverse four way-points along a time-horizon. Each way-point has to be visited by the robot at a specified time window, as described by the following task:*

$$\phi(\alpha) = \mathsf{F}_{[t_1,t_2]}(\boldsymbol{P} \in WP_1) \wedge \mathsf{F}_{[t_3,t_4]}(\boldsymbol{P} \in WP_2) \wedge \mathsf{F}_{[t_5,t_6]}(\boldsymbol{P} \in WP_3) \wedge \mathsf{F}_{[t_7,t_8]}(\boldsymbol{P} \in WP_4) \tag{3.1}$$

where $\boldsymbol{P}$ denotes the robot's position at each way-point WP. Figures 1(a) and 1(b) present the robot's trajectory when it is not ($\rho < 0$) and it is ($\rho > 0$) in compliance with the task specifications, respectively. Each formula within the task establishes the behavior of the robot at the corresponding way-point, considering a temporal and spatial parameter.



(a) Robot's trajectory with $\rho < 0$.

(b) Robot's trajectory with $\rho > 0$.

Figure 1. Trajectory of the robot for the task in (3.1).

Considering the task's formulation using the STL semantics, the MTL framework simultaneously trains a model with multiple tasks, given that each task involves several formulas. Through spatial and temporal parameter perturbation within these tasks, the aim is to obtain a sequence of control inputs and state variables that satisfy the set of training tasks serving as a strong initializer to address other related tasks. This stage is referred to as the "training stage". Subsequently, a phase known as the "testing stage" is applied, in which the solutions of the tasks are achieved, i.e., ρ becomes positive, in only a few iterations of the SCP algorithm. This concept involves a quick online adaptation to new tasks, as the training is performed offline on a batch of related training tasks. The term "related" implies that the temporal and spatial parameters are affected by a similar level of perturbation.

Algorithm 1 presents the procedure to generate M tasks given N formulas in conjunctive normal form (CNF). If the tasks are in nested form, N corresponds to the number of levels of the nested task, from the innermost to the outermost level. The base task includes the input data $\underline{\mathcal{T}}$ and $\overline{\mathcal{T}}$, which are the lower and upper bounds, respectively, for the temporal parameters, and C is the spatial parameter. These are included in the sets $\underline{\mathcal{T}} = \left\{\underline{\mathcal{T}_1}, \ldots, \underline{\mathcal{T}_N}\right\}$, $\overline{\mathcal{T}} = \left\{\overline{\mathcal{T}_1}, \ldots, \overline{\mathcal{T}_N}\right\}$, and $C = \{C_1, \ldots, C_N\}$. The set $O = \{O_1, \ldots, O_N\}$ contains the temporal operators F and/or G for "eventual" and "always" operators, respectively; and the set $\mathcal{H}(x, C) = \{\mathcal{H}_1(x, C_1), \ldots, \mathcal{H}_N(x, C_N)\}$ corresponds to the predicates, which are in terms of the state variables and the spatial parameters, and that describe the system dynamics for each formula throughout the task. Additionally, the Boolean operators $\wedge$ and $\vee$ to conform the task are in the set $\mathcal{B} = \{\mathcal{B}_1, \mathcal{B}_2 \ldots, \mathcal{B}'_{N-1}, \varnothing\}$. For CNF task construction purposes, the last element in $\mathcal{B}$ is $\varnothing$ since no more specifications are ahead to conform the CNF task. In the nested task, a unique predicate described as $\mathcal{H}(x, \mathcal{S})$ is assumed where $\mathcal{S}$ is the spatial parameter for the nested base task, and no Boolean operators are incorporated along the levels. For task generation purposes, N formulas/levels are created by using the probability distribution of each temporal and spatial parameter, until M tasks are generated. The perturbation on the parameters is performed using a normal distribution $\mathcal{N}(\mu, \sigma^2)$, where μ and σ are the mean and the standard deviation, respectively. Given that the tasks generated in Algorithm 1 are related to each other, the resulting tasks are in conjunctive or nested form, without a combination between forms.

Example 2. *The linear position x of a mass along a time horizon of 20 seconds is described by the task in conjunctive normal form below:*

$$\phi = \mathsf{G}_{[4,6]}(x \geq 9) \wedge \mathsf{F}_{[10,12]}(x \leq -10) \wedge \mathsf{G}_{[16,18]}(x \leq -10) \tag{3.2}$$

In accordance with the base task in (3.2)*, a perturbed task is generated using the pseudo-code in Algorithm 1. The lower and upper limits of the temporal parameters, i.e., the time-windows for the three specifications, are represented by* $\underline{\mathcal{T}} = \{4, 10, 16\}$ *and* $\overline{\mathcal{T}} = \{6, 12, 18\}$*, respectively. The sets of spatial parameters, Boolean operators, and temporal operators are* $C = \{9, -10, -15\}$*,* $\mathcal{B} = \{\wedge, \wedge, \varnothing\}$*, and* $O = \{\mathsf{G}, \mathsf{F}, \mathsf{G}\}$*, respectively. Following the conjunctive normal form section in Algorithm 1, and considering a normal distribution for the perturbation of the spatiotemporal parameters, the first specification, this is* $j = 1$*, of the CNF task is described by* $\underline{\mathcal{T}_1}' = 3.92$*,* $\overline{\mathcal{T}_1}' = 5.87$*,* $C_1' = 8.75$*, using lines 5 to 7 in the pseudo-code. Line 8 joins the temporal operator* $O_1 \leftarrow \mathsf{G}$ *with the predicate* $\mathcal{H}_1(x, C_1') \leftarrow (x - 8.75 \geq 0)$ *and the Boolean operator* $\mathcal{B}_1 \leftarrow \wedge$ *at the end. This results in the first portion of the CNF task given by* $\phi_1 = \mathsf{G}_{[3.92,5.87]}(x - 8.75 \geq 0) \wedge$*. Later, the content in* ϕ_1 *is assigned to Task* $\leftarrow \mathsf{G}_{[3.92,5.87]}(x - 8.75 \geq 0) \wedge$ *in line 9. Similarly, the second specification* $\phi_2 = \mathsf{F}_{[10.08,11.93]}(-x - 9.73 \geq 0) \wedge$ *is appended in Task* $\leftarrow \mathsf{G}_{[3.92,5.87]}(x - 8.75 \geq 0) \wedge \mathsf{F}_{[10.08,11.93]}(-x - 9.73 \geq 0) \wedge$*. The last specification* $\phi_3 = \mathsf{G}_{[16.58,17.45]}(x + 15.28 \geq 0)$ *is finally appended in the Task, which results in the following conjunctive normal form task:*

$$\mathit{Task} \leftarrow \mathsf{G}_{[3.92,5.87]}(x - 8.75 \geq 0) \wedge \mathsf{F}_{[10.08,11.93]}(-x - 9.73 \geq 0) \wedge \mathsf{G}_{[16.58,17.45]}(x + 15.28 \geq 0) \tag{3.3}$$

Example 3. *The linear position of a mass is addressed in this case as a nested STL task along a 10-second horizon, as shown below:*

$$\phi = \mathsf{F}_{[0,10]}\left(\mathsf{G}_{[0,2]}\left(\mathsf{F}_{[0,2]}(x > 8)\right)\right) \tag{3.4}$$

Algorithm 1 Task Generator

function TaskGenerator(M, N, $\underline{\mathcal{T}}$, $\overline{\mathcal{T}}$, C, O, $\mathcal{B}$)
— Conjunctive Normal Form —
  if isCNF(Task) **then**
    for $i = 1$ to M **do**
      for $j = 1$ to N **do**
        $\underline{\mathcal{T}_j}' \sim \mathcal{N}(\underline{\mathcal{T}_j}, \sigma^2_{\underline{\mathcal{T}_j}})$ ▷ Lower time limit of spec j
        $\overline{\mathcal{T}_j}' \sim \mathcal{N}(\overline{\mathcal{T}_j}, \sigma^2_{\overline{\mathcal{T}_j}})$ ▷ Upper time limit of spec j
        $C_j' \sim \mathcal{N}(C_j, \sigma^2_{C_j})$ ▷ Spatial param of spec j
        $\phi_j \leftarrow O_{j[\underline{\mathcal{T}_j}', \overline{\mathcal{T}_j}']}(\mathcal{H}_j(x, C_j'))\mathcal{B}_j$ ▷ Build spec j
        $\text{Task}_i \leftarrow \overline{\text{Task}}_i.\text{append}(\phi_j)$ ▷ Add spec j to task i
      end for
      Tasks $\leftarrow$ Tasks.append(Task_i) ▷ Add task i to Tasks
    end for
  end if
— Nested form —
  if isNested(Task) **then**
    for $i = 1$ to M **do**
      $\mathcal{S}' \sim \mathcal{N}(\mathcal{S}, \sigma^2_{\mathcal{S}})$ ▷ Perturbation on spatial parameter
      $\text{Task}_i \leftarrow \mathcal{H}(x, \mathcal{S}')$ ▷ Task i initialized with predicate
      for $j = 1$ to N **do**
        $\underline{\mathcal{T}_j}' \sim \mathcal{N}(\underline{\mathcal{T}_j}, \sigma^2_{\underline{\mathcal{T}_j}})$ ▷ Lower time limit of level j
        $\overline{\mathcal{T}_j}' \sim \mathcal{N}(\overline{\mathcal{T}_j}, \sigma^2_{\overline{\mathcal{T}_j}})$ ▷ Upper time limit of level j
        $\text{Task}_i \leftarrow O_{j[\underline{\mathcal{T}_j}', \overline{\mathcal{T}_j}']}(\text{Task}_i)$ ▷ Add previous expression to Task i
      end for
      Tasks $\leftarrow$ Tasks.append(Task_i) ▷ Add task i to Tasks
    end for
  end if
  return Tasks
end function

The task in (3.4) *specifies that within the 10 seconds horizon, there must eventually exist a time-window of 2 seconds during which, at every instant during the next two seconds, it is guaranteed that* $x \geq 8$ *will occur at least once. Considering this behavior as the base task and following the nested form section in Algorithm 1, a perturbed task is generated using the lower and upper limits for the temporal parameters, i.e., the time-windows for the three levels, which are contained in* $\underline{\mathcal{T}} = \{0, 0, 0\}$ *and* $\overline{\mathcal{T}} = \{2, 2, 10\}$*, respectively. The set for temporal operators is* $\mathcal{O} = \{\mathsf{G}, \mathsf{F}, \mathsf{G}\}$*. Temporal parameters and temporal operator sets start from the innermost to the outermost level in the nested task. The spatial parameter corresponds to* $\mathcal{S} = 8$*. Given a normal distribution for the perturbation of the spatiotemporal parameters, the perturbed spatial parameter* $\mathcal{S}' = 7.55$*. Line 17 in the pseudo-code provides the predicate* $(x - 7.55 > 0)$ *which initializes the Task. The perturbation of temporal parameters is performed in lines 19 and 20 of the pseudo-code, providing* $\underline{\mathcal{T}_1}' = 0.35$ *and* $\overline{\mathcal{T}_1}' = 1.93$*, for the lower and upper limits in the first level time-window, respectively. The temporal parameter is* $\mathcal{O}_1 = \mathsf{F}$*. This results in the first portion of the nested task, i.e.,* $\mathsf{F}_{[0.35,1.93]}$*, which is appended with the already initialized expression. Then,* $Task \leftarrow \mathsf{F}_{[0.35,1.93]}(x - 7.55 > 0)$*. Later, the perturbed lower and upper limits in the time-window for the second level are* $\underline{\mathcal{T}_2}' = 0.8$ *and* $\overline{\mathcal{T}_2}' = 2.14$*, respectively, and the temporal operator is* $\mathcal{O}_2 = \mathsf{G}$*. Now the expression for the task is* $Task \leftarrow \mathsf{G}_{[0.8,2.14]}(\mathsf{F}_{[0.35,1.93]}(x - 7.55 > 0))$*. Similarly, for the outermost level the perturbed temporal parameters are* $\underline{\mathcal{T}_3}' = 0.27$ *and* $\overline{\mathcal{T}_3}' = 10.85$*. The temporal operator is* $\mathcal{O}_3 = \mathsf{F}$*. Finally the perturbed nested task is generated as follows:*

$$Task \leftarrow \mathsf{F}_{[0.27,10.85]}(\mathsf{G}_{[0.8,2.14]}(\mathsf{F}_{[0.35,1.93]}(x - 7.55 > 0))) \tag{3.5}$$

As part of the training and testing stages, the SCP algorithm is used to solve the non-convex mathematical model in (2.5). Since the robustness degree in its current form is non-convex due to negative LSE for the max operator, an affine approximation is applied on $\rho(x)$ using the Taylor series expansion, which yields the linearized expression $\hat{\rho}(x)$, as presented below:

$$\hat{\rho}(x) = \rho(x_h) + \nabla\rho(x_h)^\top (x - x_h) + \frac{1}{2}(x - x_h)^\top \nabla^2\rho(x_h)(x - x_h) \tag{3.6}$$

where ∇ and ∇^2 are the gradient and the Hessian operators, respectively. The linearization of $\rho(x)$ is around a known operation point x_h, where h is the index of the SCP iterations. The expansion is performed until the second-order term that includes $\nabla^2\rho(x_h)$. However, this term can be neglected or approximated by the identity matrix. In this sense, the mathematical model in (2.5) is rewritten in its linearized version as follows:

$$\begin{aligned} & \max_{x,u} \hat{\rho}(x) - \alpha \sum_{k=1}^{N_T} [x_k^\top Q x_k + u_k^\top R u_k] \\ s.t. \quad & x_{k+1} = \hat{f}(x_k, u_k), \forall k \in \{0, 1, ..., N_T - 1\} \\ & \|x - x_h\| \leq \eta \\ & x_k \in \mathcal{X}, u_k \in \mathcal{U}, \end{aligned} \tag{3.7}$$

where $\hat{f}(x_k, u_k)$ is the linearized system dynamics, η is the trust region, and $\mathcal{X}$ and $\mathcal{U}$ are the domains for the x and u variables, respectively.

Algorithm 2 presents the SCP pseudo-code. The input set is conformed by: the ρ expression after applying LSE approximation for the min and max operators, linearized system dynamics $\hat{f}(x_k, u_k)$,

initial point x_0 for the state variables along the time horizon N_T, initial guesses x_{init} and u_{init} for the state variables' trajectory and control inputs, respectively, along the time horizon under study, and a maximum number of iterations N_h in the SCP. Initial guess values are assigned to x_h and u_h in the first iteration as nominal vectors [22] to linearize $\rho(x)$ and obtain $\hat{\rho}(x)$. Then, x^* and u^* are the solutions obtained for x and u, respectively, in the current iteration h. Prior to the next SCP iteration $h+1$, choosing the nominal vector depends on the trust region update. This is based on two performance metrics: the relative decrease $\delta = \rho(x_h) - \rho(x^*)$ and the relative error $\hat{\delta} = \rho(x_h) - \hat{\rho}(x^*)$. Trust region η shrinks/grows contingent upon the performance metric comparison [23]. If $\delta \geq \alpha'\hat{\delta}$, the trust region is increased for the next iteration $h+1$, that is, $\eta_{h+1} = \beta\eta_h$, with $0 < \alpha' < 1$ and $\beta > 1$, and $x_{h+1} \leftarrow x^*$, meaning that the solution x^* is accepted and becomes the nominal vector for the next iteration. If $\delta < \alpha'\hat{\delta}$, the trust region is decreased for the next iteration $h+1$, that is, $\eta_{h+1} = \gamma\eta_h$, with $0 < \gamma < 1$ and $x_{h+1} \leftarrow x_h$, meaning that the solution x^* is rejected and the current solution remains as the nominal vector for the next iteration.

Convergence criteria is in terms of a maximum number of iterations N_h and moreover, the SCP algorithm stops when the robustness degree of signal x is positive and the signal meets the task formulas. The SCP returns the solutions x^{sol} and u^{sol} in terms of the value sequence for x and u variables, respectively. The corresponding robustness degree ρ^{sol} value for the x trajectory is also provided.

Algorithm 2 Sequential Convex Programming (SCP)

1: **function** SCP(ρ, $\hat{f}(x_k, u_k)$, x_0, x_{init}, u_{init}, N_h)
2: $x_h \leftarrow x_{\text{init}}$ ▷ Initial guess for state trajectory
3: $u_h \leftarrow u_{\text{init}}$ ▷ Initial guess for control trajectory
4: **for** $h = 1$ to N_h **do**
5: Approximate ρ as $\hat{\rho}$ using (3.6)
6: $x^* \leftarrow \max_{x,u} \rho$ ▷ Solve convex problem (3.7)
7: **if** trust region is increased **then**
8: $x_{h+1} \leftarrow x^*$ ▷ Solution obtained becomes the nominal vector for $h+1$
9: **else if** trust region is decreased **then**
10: $x_{h+1} \leftarrow x_h$ ▷ Current nominal vector is preserved for $h+1$
11: **end if**
12: **if** convergence criteria met **or** $\rho > 0$ and task satisfied **then**
13: **break** ▷ Stop if solution is feasible and robust
14: **end if**
15: **end for**
16: **return** $\rho^{\text{sol}}, x^{\text{sol}}, u^{\text{sol}}$ ▷ Return final solution
17: **end function**

The training stage is described in Algorithm 3. Using the task generator function in Algorithm 1, M_L training tasks are created in line 2 and their corresponding symbolic expressions for the robustness degree are obtained. Later in line 3, the average robustness degree ρ_{av} is conformed which overcomes all the training tasks. Notice that ρ_{av} is also a symbolic expression and is used in the linearized optimal control problem in (3.7). Line 4 runs the SCP to solve (3.7) considering the expression ρ_{av}. This yields the sequence of state variables x_{learn} and control inputs u_{learn} that is in compliance with the specifications

of all the M_L training tasks, that is, ($\rho^{\text{sol}}_{\text{learn}} > 0$), as long as these tasks are not in conflict with each other. The solution x_{learn} with its sequence of control inputs u_{learn} are used as initialization to solve the tasks in the testing stage.

Algorithm 3 Training Stage

1: **function** TRAINING STAGE($M_L, N, \underline{\mathcal{T}}, \overline{\mathcal{T}}, \mathcal{C}, \mathcal{O}, \mathcal{B}$)
2: $\quad$ $\text{Tasks}_{\text{learn}} \leftarrow \text{TaskGenerator}(M_L, N, \underline{\mathcal{T}}, \overline{\mathcal{T}}, \mathcal{C}, \mathcal{O}, \mathcal{B})$ ▷ Task generation for training
3: $\quad$ $\rho_{\text{av}} \leftarrow \frac{1}{M_L} \sum_{h=1}^{M_L} \rho(\text{Tasks}_{\text{learn}_h})$ ▷ Compute average robustness degree for training tasks
4: $\quad$ $\rho^{\text{sol}}_{\text{learn}}, x_{\text{learn}}, u_{\text{learn}} \leftarrow \text{SCP}(\rho_{\text{av}}, \hat{f}(x_k, u_k), x_0, x_{\text{init}}, u_{\text{init}}, N_h)$ ▷ Obtain initializer for testing stage
5: $\quad$ **return** $x_{\text{learn}}, u_{\text{learn}}$
6: **end function**

In the testing stage described in Algorithm 4, the input data are the x_{learn} and u_{learn} previously obtained in Algorithm 3. Parameter M_T is the number of testing tasks to generate in the testing stage. In lines 4 and 5, x_{learn} and u_{learn} are assigned to the initial solution x_h and u_h to run the SCP. Once the convergence in the SCP algorithm is reached, a solution for the testing task is obtained and stored in $\rho^{\text{sol}}_{\text{test}}$, in addition to the signal x_{test} and the control sequence u_{test}.

Algorithm 4 Testing Stage

1: **function** TESTINGSTAGE($x_{\text{learn}}, u_{\text{learn}}, M_T, N, \underline{\mathcal{T}}, \overline{\mathcal{T}}, \mathcal{C}, \mathcal{O}, \mathcal{B}$)
2: $\quad$ $\text{Tasks}_{\text{test}} \leftarrow \text{TaskGenerator}(M_T, N, \underline{\mathcal{T}}, \overline{\mathcal{T}}, \mathcal{C}, \mathcal{O}, \mathcal{B})$ ▷ Tasks generation for testing
3: $\quad$ **for** $j = 1, \ldots, M_T$ **do**
— Initializer assigned to nominal vector —
4: $\quad\quad$ $x_h \leftarrow x_{\text{learn}}$
5: $\quad\quad$ $u_h \leftarrow u_{\text{learn}}$
6: $\quad\quad$ $\rho^{\text{sol}}_{\text{test}_j}, x_{\text{test}_j}, u_{\text{test}_j} \leftarrow \text{SCP}(\rho(\text{Tasks}_{\text{test}_j}), \hat{f}(x_k, u_k), x_0, x_h, u_h, N_h)$ ▷ Solve the task via SCP
7: $\quad$ **end for**
8: $\quad$ **return** $\rho^{\text{sol}}_{\text{test}}, x_{\text{test}}, u_{\text{test}}$
9: **end function**

The MTL framework is in the flow chart illustration of Figure 2. The enclosed numbers over the arrows between blocks are the steps of the overall methodology. Initially, the input involves M_L and M_T for the number of training and testing tasks, respectively, the number of formulas N for each task, upper $\overline{\tau}$ and lower $\underline{\tau}$ limits for temporal parameters, σ_{train} and σ_{test}, and spatial parameters C. Once M_L training tasks are generated through Algorithm 1 using the probability distribution for perturbation parameters, the average ρ_{av} and the approximation $\hat{\rho}_{\text{av}}$ are computed, based on LSE and Taylor series expansion. Later, initial guesses x_{init} and u_{init} are assigned to the state variables x and control inputs u, respectively, since the SCP relies on this initial point to iteratively refine the solutions x_{learn} and u_{learn}. At this point of the flow chart, the following is the testing stage, where $\sigma = \sigma_{\text{test}}$, being $\sigma_{\text{test}} > \sigma_{\text{train}}$, meaning that the perturbation in the testing stage is larger than that used in the training. Notice that for each testing task, the initial point for the SCP corresponds to x_{learn} and u_{learn}, previously obtained in the training stage. For each task, once the SCP reaches the convergence, the values for the robustness degree, state variables trajectories and control inputs are presented in $\rho^{\text{sol}}_{\text{test}}$, x_{test}, and u_{test}, respectively.

The task generator (Algorithm 1), sequential convex programming (Algorithm 2), and ρ approximation modules are involved in both training (Algorithm 3) and testing (Algorithm 4) stages.

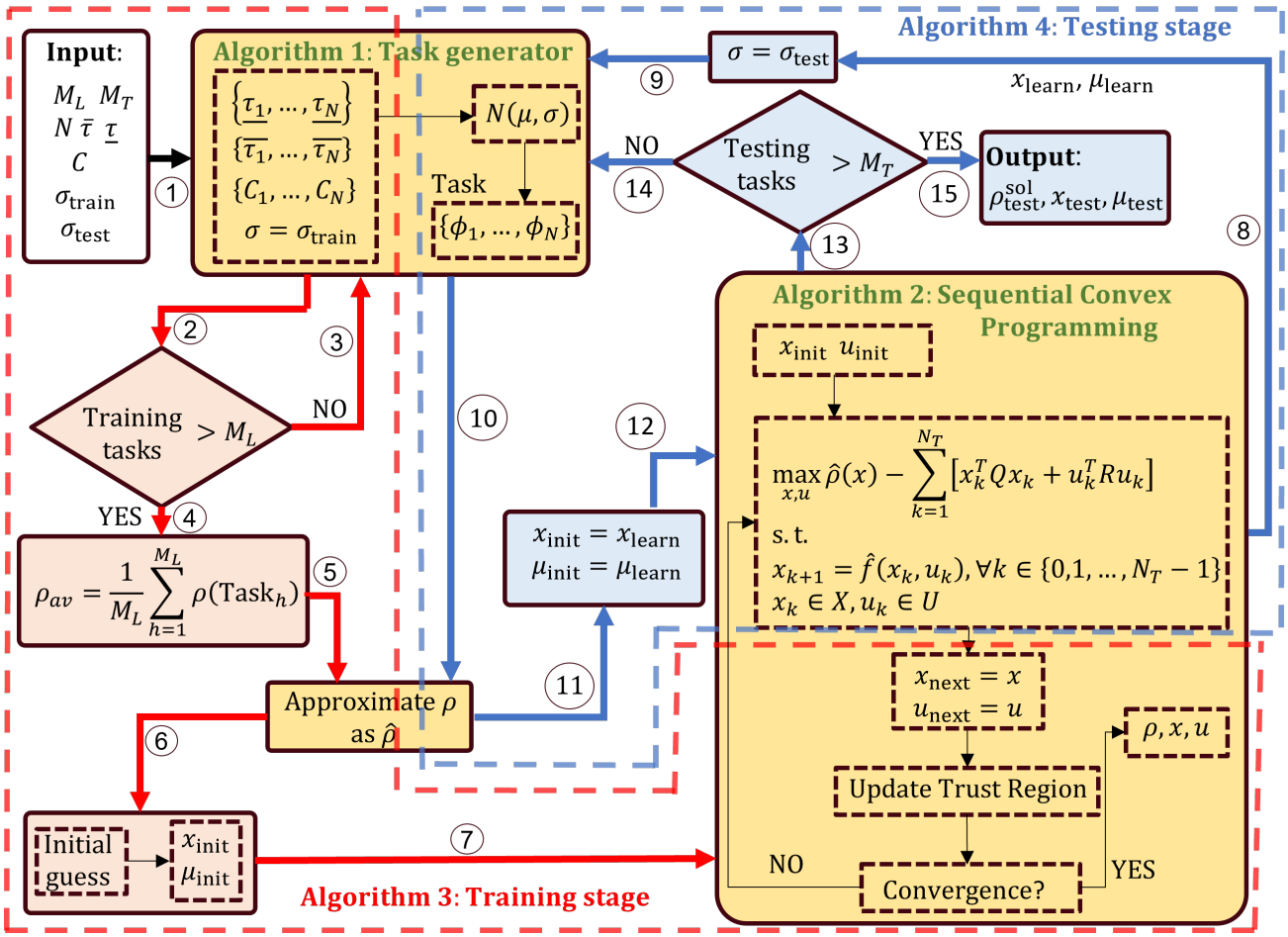


Figure 2. The overall framework of multi-task learning. Enclosed numbers denote the steps along the procedure.

4. Case studies and results

For validation purposes, the MTL methodology is proved in two case studies. The first case corresponds to the spring-mass-damper system where the mass position is controlled along the time horizon given a task with different STL formulas. In the second case, the air traffic control (ATC) problem is addressed to generate a control sequence on a quad-copter, subject to preset rules in the STL framework. Training and testing stages are run in MATLAB R2023A using the Gurobi solver in CVX modeling language, equipped with an AMD Ryzen 3 2300U processor and 12 GB of RAM.

4.1. Spring-mass-damper system

The first scenario is the spring-mass-damper system with the discretized dynamics shown below.

$$x_{k+1} = x_k + \left(\begin{bmatrix} 0 & 1 \\ \frac{-k_s}{m} & \frac{-b}{m} \end{bmatrix} x_k + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} u_k \right) \Delta_t \tag{4.1}$$

where $x \in \mathbb{R}^2$ represents the variables x_1 and x_2: position and velocity of the mass, respectively. Control input is denoted by $u \in \mathbb{R}$. Mass, spring, and damping constants are m, k_s, and b, respectively, and are set at $m = 1\,\text{kg}$, $k_s = 2\,\text{N/m}$, and $b = 0.2\,\text{Ns/m}$. The time step is chosen as $\Delta_t = 0.1\,\text{s}$. The base task is related to a desired behavior in the mass position x_1, along a time horizon of 30 seconds and initial values for the state variables $[x_1 \quad x_2]_0^\top = [\pi \quad 2]^\top$. The base task is described as follows:

$$\mathsf{G}_{[4,6]}(x_1 \geqslant 9) \wedge \mathsf{F}_{[10,12]}(x_1 \leqslant -10) \wedge \mathsf{G}_{[16,18]}(x_1 \leqslant -12) \wedge \mathsf{F}_{[22,24]}(x_1 \geqslant 13) \wedge \mathsf{G}_{[28,30]}(x_1 \leqslant -15) \tag{4.2}$$

Given the task specifications in (4.2), the mass is required to always reach a position farther from $x_1 = 9$ between $t = 4$ and $t = 6$; then within $t \in [10, 12]$ eventually to be placed in $x_1 \leq -10$; later, always within $t = 16$ and $t = 18$ to a position at $x_1 \leq -12$; then, eventually reach beyond $x_1 = 13$ between $t \in [22, 24]$; and during the last two seconds to be always positioned at $x_1 \leq -15$. Once the SCP is applied on the task in (4.2), the mass position along the time horizon is shown in Figure 3. The specifications are met considering the spatiotemporal parameter settings applied in the task.

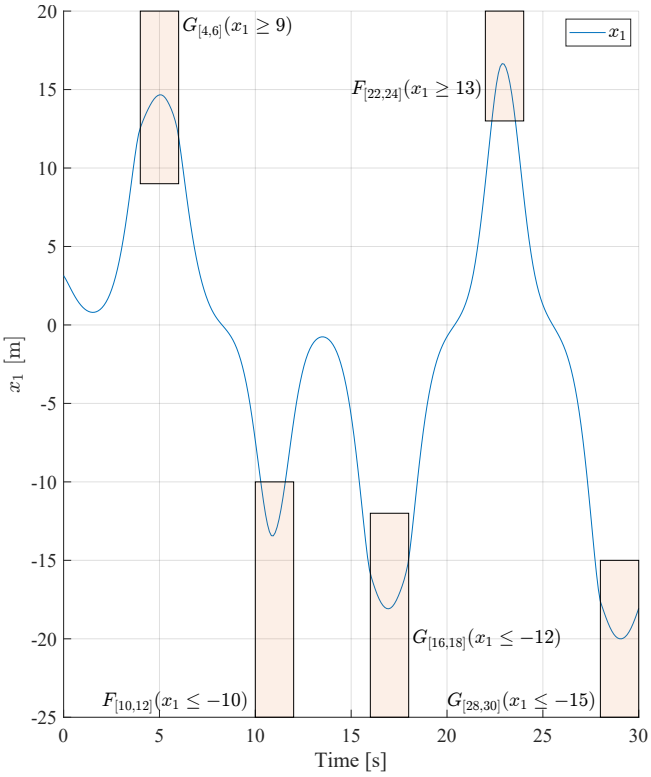
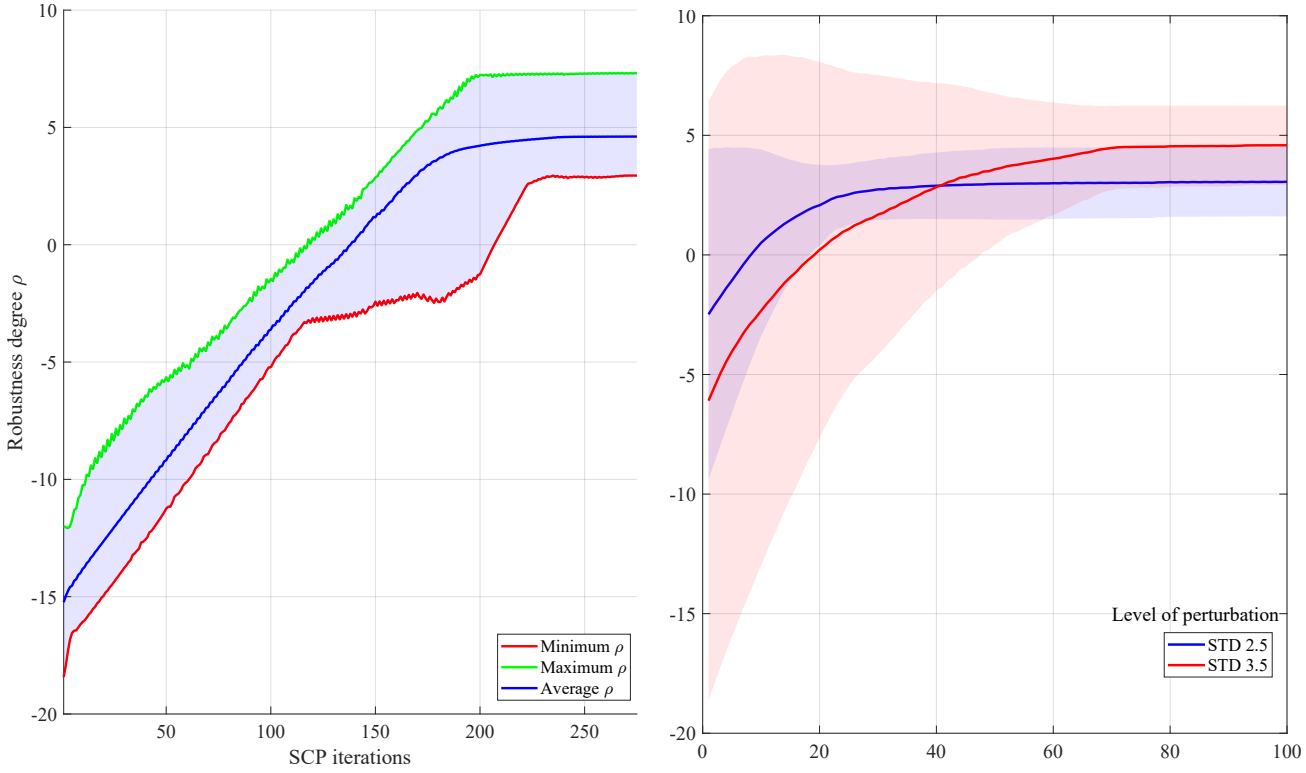


Figure 3. Mass position (x_1) along the time horizon.

Figure 4(a) presents the training stage over 25 tasks randomly generated with a standard deviation (STD) of 1.1 in the temporal and spatial parameters. It is worth mentioning that the tasks generated in the training stage are not in conflict with each other, meaning that the spatiotemporal specifications are overlapped between tasks, and that the solution x_{learn} obtained satisfies all the training tasks. Notice that the average ρ over all the tasks becomes positive after 140 SCP iterations and converges after 250

iterations. For the least and most complex tasks, 119 and 207 SCP iterations were needed, respectively. A higher STD along the task generation provides a more diverse set of tasks in the training stage, and furthermore, more SCP iterations are required to solve all the tasks. The solution obtained at this point is used as an initializer in the testing stage.

For testing purposes, 10 tasks are randomly generated by applying perturbations using standard deviations in the temporal and spatial parameters. As opposed to the training, in testing, the tasks randomly generated are not required to be non-conflicting with each other, as they are solved separately. The behavior of the robustness degree is plotted in Figure 4(b) for two STD values, along with the corresponding dispersion across the tasks. Compared to the training stage, it is observed that in the testing stage, the average robustness degree becomes positive after a few SCP iterations: For perturbation levels of STD = 2.5 and STD = 3.5, only 9 and 20 SCP iterations were required, representing reductions of approximately 94% and 86%, respectively, compared to the training stage.



(a) Robustness degree ρ: training stage for study case 1.

(b) Robustness degree ρ: testing stage for study case 1.

Figure 4. Results for the spring-mass-damper system.

4.2. ATC problem for an autonomous quad-rotor

The air traffic control (ATC) problem involves complex rules to ensure safety in airspace used by aircraft. In the context of unmanned aerial vehicles, the ATC paradigm seeks to enable safe, efficient operation and route optimization in drone operations [24]. Flight constraints are generally related to altitude ranges for flying in certain areas and prohibited zones where aerial operations are not allowed, such as airports and military facilities [25]. In this work, the ATC problem is scaled to control a

quad-rotor to reach a way-point and a final destination, i.e., addressing battery recharge and package delivery, following some altitude rules and flight avoidance of prohibited zones.

For simulation purposes, the linearized and discretized quad-rotor dynamics [26] are presented as follows:

$$x_{k+1} = Ax_k + Bu_k \tag{4.3a}$$

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0.2 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0.2 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0.2 & 0 & 0 & 1 \end{bmatrix} \quad B = \begin{bmatrix} 1.96 & 0 & 0 \\ 0 & -1.96 & 0 \\ 0 & 0 & 0.4 \\ 0.196 & 0 & 0 \\ 0 & -0.196 & 0 \\ 0 & 0 & 0.04 \end{bmatrix} \tag{4.3b}$$

where $x \in \mathbb{R}^6$ represents the variables x_1, x_2, and x_3 corresponding to the velocities, and x_4, x_5, and x_6 the positions, in x, y, and z coordinates respectively; and $u \in \mathbb{R}^3$ represents the control inputs u_1, u_2, and u_3 corresponding to the roll angle, pitch angle, and thrust, respectively. Following the STL semantics, the base task of an autonomous quad-rotor in the ATC framework is described in (4.4a) to (4.4e) along a time horizon of 2 seconds.

$$\mathsf{G}_{[0,2]}(q \in \text{Zone}_1 \Rightarrow 3 \leqslant x_6 \leqslant 7) \tag{4.4a}$$

$$\mathsf{G}_{[0,2]}(q \in \text{Zone}_2 \Rightarrow 0 \leqslant x_6 \leqslant 4) \tag{4.4b}$$

$$\mathsf{G}_{[0,2]}(x_4^2 + x_5^2 > 1.5^2) \tag{4.4c}$$

$$\mathsf{F}_{[0,2]}((x_4 + 5)^2 + (x_5 + 2)^2 + (x_6 - 3)^2 \leqslant 0.5^2) \tag{4.4d}$$

$$\mathsf{F}_{[0,2]}((x_4 + 2.5)^2 + (x_5 - 2.5)^2 + (x_6 - 1)^2 \leqslant 0.5^2) \tag{4.4e}$$

where q is the quad-rotor position in x, y, and z coordinates. The task involves (4.4a) $\wedge$ (4.4b) $\wedge$ (4.4c) $\wedge$ (4.4d) $\wedge$ (4.4e). Formulas (4.4a) and (4.4b) require that the altitude is within a lower and upper limit, depending on the zone. During the time horizon, the quad-rotor is required to avoid a cylindrical unsafe zone, which is described in (4.4c). Way-point and terminal point are represented as spheres in (4.4d), and (4.4e) respectively, that are eventually reached by the quad-rotor along N_T.

Considering the unsafe zone centered at the origin, and the task in (4.4a) to (4.4e), the trajectory performed by the quad-rotor is presented in Figure 5, once the robustness degree reaches a positive value in the SCP iterations. Given that the SCP requires an initial operation point to deploy the optimization, the initial trajectory for the state variables in the ATC problem is obtained by solving the optimal control problem in (4.5), considering a reference point x_{ref}. This trajectory is not required to satisfy the task specifications in (4.4a)–(4.4e); rather, it provides a consistent initial point for executing the SCP. In this regard, and considering the increased complexity of the system dynamics with more variables, providing a well-defined initial signal trajectory is crucial for the SCP to properly perform. A random initial solution can lead to poor SCP performance, including numerical instability during optimization and limited improvement in the objective function across iterations.

$$\begin{aligned} &\min_{x,u} \sum_{k=1}^{N_T} \left[(x_k - x_{\text{ref}})^\top Q (x_k - x_{\text{ref}}) + u_k^\top R u_k \right] \\ s.t.\ \ &x_{k+1} = f(x_k, u_k), \forall k \in \{0, 1, ..., N_T - 1\} \\ &x_k \in \mathcal{X}, u_k \in \mathcal{U}. \end{aligned} \tag{4.5}$$

In some cases and depending on the level of perturbation assigned to the task, the way-point and terminal point might be visited in a different order, since there is no visit preference for these two points in the STL formulas.

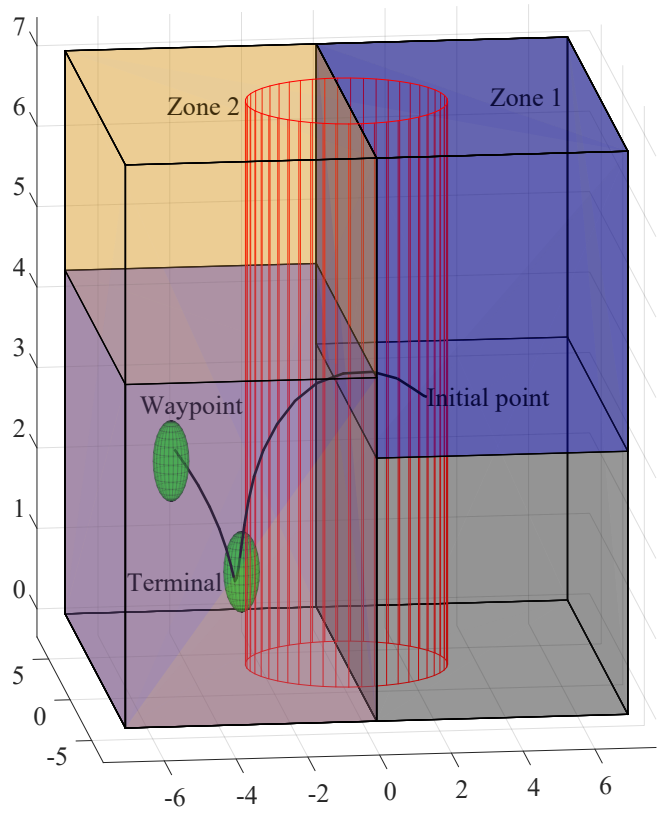
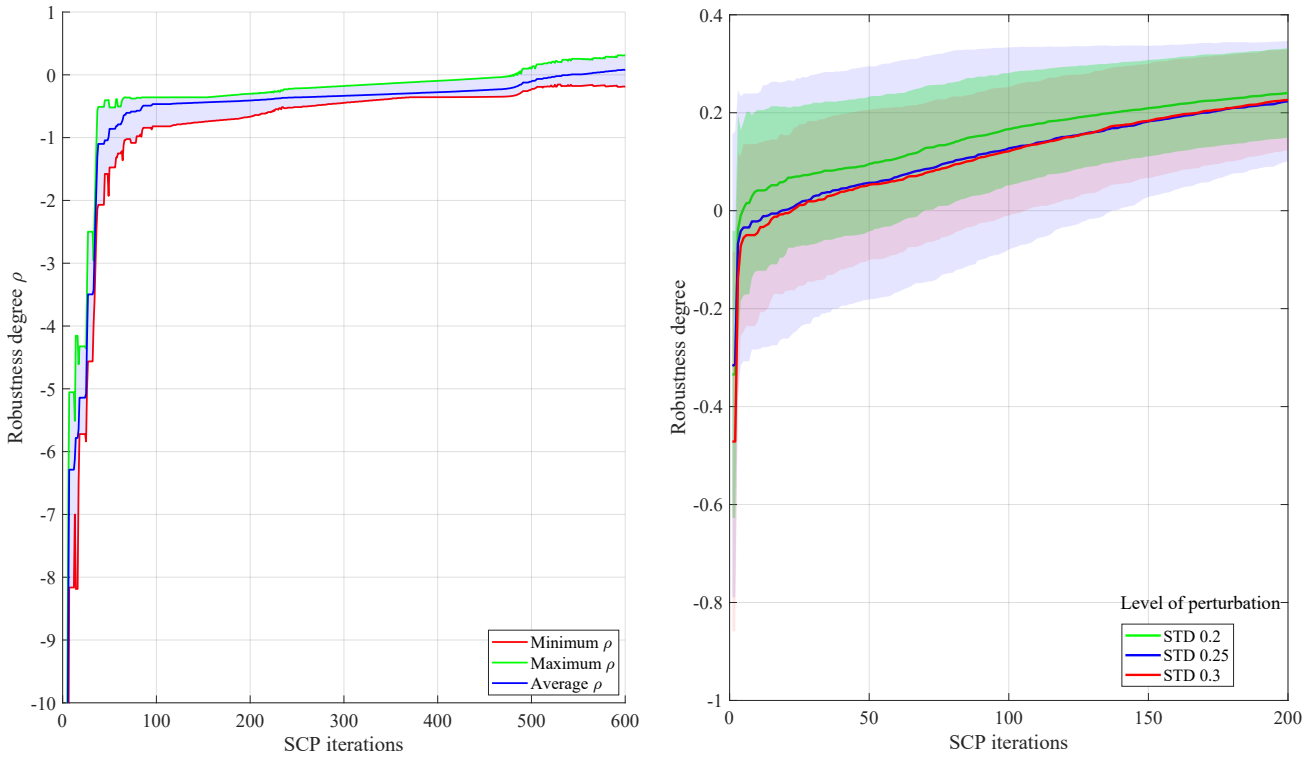


Figure 5. Quad-rotor trajectory.

The training stage is performed for the ATC problem considering 5 tasks, randomly generated with an STD of 0.1 applied on the spatial parameters, i.e., radii of the unsafe zone, way-point and terminal point, and allowable altitude on the zones. Temporal parameters are not perturbed as there is no sequence-based preference to execute the formulas. Similar to the previous case, in the ATC problem the training tasks must not be in conflict each other and the obtained solution x_{learn} should satisfy all the training tasks. The evolution of the maximum, average, and minimum ρ in this stage is depicted in Figure 6(a), which become positive after 486, 542, and 1738 SCP iterations, respectively.

Then the solution obtained in the training stage is used as an initializer for the testing stage. Three STD values are considered to perturb the position of the way-point and terminal point, which results in highly complex tasks. Ten tasks are randomly generated, and for each perturbation level, the average ρ is shown in Figure 6(b) with dispersion across the tasks. Notice that the larger the perturbation level, the more SCP iterations are needed to obtain a positive average ρ. In testing, SCP iterations required to meet the tasks are at most 4% compared with the training stage.



(a) Robustness degree ρ: training stage for study case 2.

(b) Robustness degree ρ: testing stage for study case 2.

Figure 6. Results for the ATC problem.

5. Conclusions

A multi-task learning (MTL) with signal temporal logic (STL) framework is effectively integrated for fast online adaptation using sequential convex programming (SCP). Approximations for min and max operators in terms of log-sum exponential (LSE) expressions in the STL specifications are used to smooth the objective function in the mathematical model maximization, which overcomes the robustness degree of the overall STL task. Based on the smooth approximation and linearizing around a known operation point, SCP results in an affordable methodology to solve the training and testing stages of the MTL approach. For the two dynamical systems under study, e.g., the spring-mass-damper system and air traffic control applied to UAVs, with temporal and spatial formulas, and considering different levels of perturbation in the STL formulas, the training stage provided a warm start to solve related tasks in a few SCP iterations during the testing stage, even for highly perturbed tasks.

Given that the MTL framework is able to address multiple training tasks using a shared representation, it is often faster to solve a set of similar tasks together, in comparison with solving a single task from a scratch solution. This implies that when a difficult task is trained with other easier tasks, the SCP can more quickly find a solution that satisfies all task specifications, as long as the tasks are not in conflict with each other.

A future research direction involves the meta-optimization framework. In this context, using only a small number of training tasks can be an effective approach for enabling rapid adaptation to more perturbed tasks. Additionally, within gradient-based methods, differentiable programming in meta-

optimization settings may offer a powerful means to efficiently compute how the objective function changes with respect to problem parameters for training purposes.

Author contributions

Andres Arias: Conceptualization, methodology, computational implementation, validation of results, and writing. Chuangchuang Sun: Conceptualization, methodology, supervision, review, and editing.

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Conflict of interest

Chuangchuang Sun is an editorial boardmember for Metascience in Aerospace and was not involved in the editorial review or the decision to publish this article. All authors declare that there are no competing interests.

References

1. Sándor Z (2019) Challenges caused by the unmanned aerial vehicle in the air traffic management. *Period Polytech Transp Eng* 47: 96–105. https://doi.org/10.3311/PPtr.11204
2. Shrestha R, Bajracharya R, Kim S (2021) 6g enabled unmanned aerial vehicle traffic management: A perspective. *IEEE Access* 9: 91119–91136. https://doi.org/10.1109/ACCESS.2021.3092039
3. Zhang Y, Yang Q (2021) A survey on multi-task learning. *Ieee T Knowl Data Eng* 34: 5586–5609. https://doi.org/10.1109/TKDE.2021.3070203
4. Zhang Y, Yang Q (2018) An overview of multi-task learning. *Natl Sci Rev* 5: 30–43. https://doi.org/10.1093/nsr/nwx105
5. Thung KH, Wee CY (2018) A brief review on multi-task learning. *Multimed Tools Appl* 77: 29705–29725. https://doi.org/10.1007/s11042-018-6463-x
6. Parnami A, Lee M (2022) Learning from few examples: A summary of approaches to few-shot learning. *arXiv preprint*. https://doi.org/10.48550/arXiv.2203.04291
7. Mao C, Gupta A, Nitin V, et al. (2020) Multitask learning strengthens adversarial robustness. In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, Proceedings, Part II 16*, 158–174. Springer. https://doi.org/10.1007/978-3-030-58536-5_10
8. Caruana R (1997) Multitask learning. *Mach learn* 28: 41–75. https://doi.org/10.1023/A:1007379606734
9. Sener O, Koltun V (2018) Multi-task learning as multi-objective optimization. *Adv Neur Inform Process Syst* 31.
10. Guo M, Haque A, Huang DA, et al. (2018) Dynamic task prioritization for multitask learning. In: *Proceedings of the European conference on computer vision (ECCV)*, 270–287.

11. Kendall A, Gal Y, Cipolla R (2018) Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 7482–7491.

12. Vafaeikia P, Namdar K, Khalvati F (2020) A brief review of deep multi-task learning and auxiliary task learning. *arXiv preprint*. https://doi.org/10.48550/arXiv.2007.01126

13. Gilpin Y, Kurtz V, Lin H (2020) A smooth robustness measure of signal temporal logic for symbolic control. *Ieee Contr Syst Lett* 5: 241–246. https://doi.org/10.1109/LCSYS.2020.3001875

14. Pant YV, Abbas H, Mangharam R (2017a) Smooth operator: Control using the smooth robustness of temporal logic. In: *2017 IEEE Conference on Control Technology and Applications (CCTA)*, IEEE, 1235–1240. https://doi.org/10.1109/CCTA.2017.8062628

15. Lindemann L, Dimarogonas DV (2018) Control barrier functions for signal temporal logic tasks. *IEEE Contr Syst Lett* 3: 96–101. https://doi.org/10.1109/LCSYS.2018.2853182

16. Lindemann L, Dimarogonas DV (2020) Barrier function based collaborative control of multiple robots under signal temporal logic tasks. *Ieee T Control Netw Syst* 7: 1916–1928. https://doi.org/10.1109/TCNS.2020.3014602

17. Liu W, Xiao W, Belta C (2023) Learning robust and correct controllers from signal temporal logic specifications using barriernet. In: *2023 62nd IEEE Conference on Decision and Control (CDC)*, 7049–7054. IEEE. https://doi.org/10.1109/CDC49753.2023.10383857

18. Yang G, Belta C, Tron R (2020) Continuous-time signal temporal logic planning with control barrier functions. In: *2020 American Control Conference (ACC)*, 4612–4618. IEEE. https://doi.org/10.23919/ACC45564.2020.9147387

19. Xiao W, Belta CA, Cassandras CG (2021) High order control lyapunov-barrier functions for temporal logic specifications. In: *2021 American Control Conference (ACC)*, 4886–4891. IEEE. https://doi.org/10.23919/ACC50511.2021.9483028

20. Madsen C, Vaidyanathan P, Sadraddini S, et al. (2018) Metrics for signal temporal logic formulae. In: *2018 IEEE Conference on Decision and Control (CDC)*, 1542–1547. IEEE. https://doi.org/10.1109/CDC.2018.8619541

21. Mao Y, Acikmese B, Garoche P L, et al. (2022) Successive convexification for optimal control with signal temporal logic specifications. In: *Proceedings of the 25th ACM International Conference on Hybrid Systems: Computation and Control*, 1–7. https://doi.org/10.1145/3501710.3519518

22. Morgan D, Chung SJ, Hadaegh FY (2014) Model predictive control of swarms of spacecraft using sequential convex programming. *J Guid Control Dynam* 37: 1725–1740. https://doi.org/10.2514/1.G000218

23. Reynolds TP, Mesbahi M (2020) The crawling phenomenon in sequential convex programming. In: *2020 American Control Conference (ACC)*, 3613–3618. IEEE. https://doi.org/10.23919/ACC45564.2020.9147550

24. Hutchins AR, Cummings M, Aubert MC, et al. (2015) Toward the development of a low-altitude air traffic control paradigm for networks of small, autonomous unmanned aerial vehicles. In: *AIAA infotech@ aerospace*, 1110. https://doi.org/10.2514/6.2015-1110

25.Foina AG, Sengupta R, Lerchi P, et al. (2015) Drones in smart cities: Overcoming barriers through air traffic control research. In: *2015 workshop on research, education and development of unmanned aerial systems (RED-UAS)*, 351–359. IEEE. 10.1109/RED-UAS.2015.7441027

26.Pant YV, Abbas H, Mangharam R (2017b) Technical report: Control using the smooth robustness of temporal logic.

AIMS Press