Discrete Event Dynamic Systems (2026) 36:18
https://doi.org/10.1007/s10626-026-00433-0

RESEARCH

On Robust Supervisory Control of Metric Discrete Event Systems for scLTL Specifications

Peiran Liu[1] · Shaowen Miao[1] · Yiding Ji[1,2] · Xiang Yin[3]

Received: 8 January 2025 / Accepted: 18 February 2026

Abstract
This work presents a comprehensive investigation into the robust supervisory control of metric discrete event systems, which are modeled as finite state automata and incorporate metric functions to measure state distances. While existing studies on supervisory control primarily focus on qualitative analysis and provide a binary answer regarding whether the controlled system satisfies the given specifications, our work introduces Metric Discrete Event Systems to quantitatively investigate robust supervisory control. Specifications are defined using a fragment of Linear Temporal Logic (LTL), specifically syntactically co-safe LTL (scLTL). Environmental disturbances can cause deviations from the system's nominal behaviors, thereby hindering the successful accomplishment of the original tasks. To mitigate this issue, we design supervisors to ensure that the controlled system degrades gracefully under adverse conditions. We formally define the robustness of supervisors in a topological context and formulate two key problems: verification of the existence of robust supervisors and synthesis of optimal robust supervisors. We propose a two-player game framework to address both problems. First, we introduce a bipartite structure called distance bipartite transition system (DBTS) as the arena of the game between the supervisor and the environment. The verification problem is then reformulated and solved as a reachability game on a special DBTS, where the set of target states is properly defined. For the synthesis problem, we define a sequence of vectors to track the shortest distance to the accepting states under disturbances and compute their fixed-point using dynamic programming techniques. Then we synthesize the optimal winning strategy of the supervisor, which eventually yields the optimal robust supervisor. Finally, we provide a case study of robot task planning to validate the performance of our proposed methods that demonstrates persuasive results in real-world scenarios.

Keywords Discrete event systems · Metric automata · Supervisory control · Robust control · Syntactically co-safe linear temporal logic · Algorithmic games

Peiran Liu and Shaowen Miao contributed equally to this work.

Yiding Ji and Xiang Yin are both corresponding authors of the work.

Extended author information available on the last page of the article

Published online: 01 May 2026

1 Introduction

Supervisory control theory is a pivotal topic of discrete event systems (DES) ever since it was first proposed in the 1980s. The plant under control is usually modeled as a finite state machine and a specification is defined to represent the admissible behaviors. The underlying control paradigm for the supervisor is to properly enable and disable certain events so that the behaviors of the controlled system are restricted within the specification (Cassandras and Lafortune 2021; Wonham and Cai 2019).

Recently, supervisory control has been intensively investigated in various DES settings, including but not limited to networked control (Luo et al. 2025), control for timed DES (Miao et al. 2025b), optimal control (Fokkink and Goorden 2024), compositional control (Malik et al. 2023), learning based control (Zhang et al. 2018), decentralized control (Ritsuka and Rudie 2024), hierarchical control (Komenda and Masopust 2023), modular control (Miao et al. 2025c), control of nondeterministic DES (Takai 2021), control of switched DES (Reveliotis and Fei 2016), control of stochastic DES (Deng et al. 2021), control of epistemic DES properties (Miao et al. 2025a), mean payoff quantitative control (Ji et al. 2022) and its variations (Lv et al. 2024; Ji et al. 2021), to name a few.

As DES are increasingly widely applied to address problems of complex dynamical systems such as robots, autonomous systems, and cyber-physical systems, it is crucial to specify system properties by formal languages. Temporal logics, such as Linear Temporal Logic (LTL) and Signal Temporal Logic (STL), provide a structured language for specifications due to their rich expressiveness and resemblance to natural languages (Baier and Katoen 2008). As an extension, Seow (2020) defines DES as transition systems with fairness constraints, allowing for verification through canonical LTL classes. Yu et al. (2024) investigates model predictive monitoring of dynamical systems with STL tasks. Additionally, Lacerda and Lima (2012) investigates supervisory control on Petri nets with LTL formulas, demonstrating its applicability in robot soccer coordination. Furthermore, Jiang and Kumar (2006) employs full branching-time logic (CTL*) to express specifications that specify constraints on both linear sequences and branching state structures. A special fragment of LTL called syntactically co-safe LTL (scLTL) is extensively used to define properties for various applications such as robot motion planning, task scheduling, and safety-critical control, since their satisfaction is guaranteed in finite time, which turns out to be less computationally expensive for system verification and design (Belta et al. 2017; Tellex et al. 2020).

In many real-world applications, dynamical systems are usually deployed in an uncertain environment and subject to disturbances, inaccurate measurements, or cyber attacks, therefore may fail to achieve the original tasks. This challenge raises the problem of robust supervisory control where supervisors are designed to tolerate or mitigate the adversaries so that the specification is still (partially) achievable. Some representative works are as follows: Alves et al. (2021) considers robust control under intermittent loss of observations; Meira-Góes et al. (2023) studies tolerance of DES when the transitions are perturbed; Wang et al. (2020) investigates robust control of manufacturing systems modeled by DES. Those works focus on qualitative analysis of relevant properties and only give a binary answer about whether the controlled system is resilient.

Recently, supervisory control against cyber attacks has also been extensively studied from the perspectives of both the attacker and the defender. For instance, Tai et al. (2022) designs covert attackers against unknown supervisors; Fritz and Zhang (2023) detects stealthy cyber attacks by

introducing permutation matrices; You et al. (2021) studies supervisory control against replacement attacks for Petri Nets; Zheng et al. (2023) proposes a framework for modeling and control of DES under joint sensor and actuator attacks; Miao et al. (2026) generalizes hierarchical control framework to networked and cyber-attacked DES; Ma and Cai (2021) develops a secret protection mechanism of DES, which involves two optimality criteria. Our problem settings are also in contrast with the above works since we model both the system and disturbances quantitatively, and consider robust control for more involved tasks defined by temporal logics.

Motivated by robust control theory for continuous systems (Sontag 1998), we develop a paradigm of robust supervisory control of metric DES where the specifications are scLTL formulas. To be more specific, we first introduce a metric function to quantify the distance between states of the system and consider a generic model of disturbances. Next, we translate the scLTL formula into a deterministic finite state automaton (DFA), and make the product between the metric DES and the DFA to integrate the system dynamics with the formula. Then we consider the effects of disturbances and introduce σ-*robust supervisors* where the positive constant σ measures the deviation from the accepting states of the product automaton under disturbances. Note that our notion of robustness is in the *topological* sense of deviation and a smaller σ indicates a more robust supervisor. Our control paradigm is to ensure graceful degradation of system performance under bounded disturbances, thus no catastrophic failures are incurred. Based on those concepts, we formulate two key problems:

(i) Verification of robust supervisors: determine if there exists a sufficiently robust supervisor under the given disturbances;
(ii) Synthesis of optimal robust supervisors: design a supervisor that achieves the best possible robustness measure under the given disturbances.

To solve these problems, we develop a two-player game-theoretic control framework where the supervisor acts as the protagonist and the environment (disturbances) as the antagonist. A novel bipartite structure named *distance bipartite transition system (DBTS)* is defined to serve as the game arena. The robustness verification problem is reformulated as a reachability game, which can be solved in the standard *fixed-point* computation. As for the synthesis problem, we introduce a sequence of vectors to track the distance from each state of the arena to the accepting states under the strategies of the supervisor and disturbances. Then, a dynamic programming method is developed to compute the fixed-point of the operators, which yields the optimal robust supervisor. Finally, we provide a running example of robot task planning to illustrate the overall algorithmic process and demonstrate the performance of our methods in practical applications.

Some ideas of robust synthesis for cyber-physical systems also contribute to our framework, see, e.g., Majumdar et al. (2013); Girard and Eqtami (2021). In contrast to symbolic formal synthesis, we consider supervisory control problems where uncontrollable events are involved and multiple events are enabled simultaneously. Our work is also related with Sakakibara and Ushio (2020); Sakakibara et al. (2021), which investigate temporal logic-based supervisory control, however, they do not consider robust control, thus are incomparable with this work in both problem settings and solutions.

The remainder of the paper is organized as follows. Section 2 introduces the metric DES model, the basic syntax and semantics of scLTL, and the supervisory control theory. Section 3 formulates the two robust supervisory control problems. Section 4 presents the two player game structure and some relevant properties. Section 5 proposes systematic game-theoretic solutions to

the problems. Section 6 illustrates the application of the proposed paradigm in a robot task planning example. Finally, Section 7 concludes the paper and lists several future research directions.

The preceding conference version (Ji and Yin 2024) presents some preliminary results that are extended in the following perspectives. First, we consider more complex tasks expressed as temporal logic formulas instead of the simple reachability manner objectives. In addition, we formulate and solve a new problem, i.e., verification of the existence of sufficiently robust supervisors, which is not covered in Ji and Yin (2024). A more comprehensive game-theoretic synthesis procedure for optimal robust supervisors is also included, together with the relevant analysis and proofs. Finally, we also provide a detailed case study to demonstrate the application of our framework in robot task planning scenarios.

2 Preliminary knowledge

2.1 System model

Let $\mathbb{R}_0^+$, $\mathbb{N}_0^+$, and $\mathbb{N}^+$ denote the set of non-negative real numbers, non-negative integers, and positive integers, respectively. For a finite set Σ, the notation $|\Sigma|$ stands for the cardinality of Σ. Given a finite set Σ, we let Σ^* be the set of finite sequences of Σ, namely finite words, and Σ^ω be the set of infinite sequences of Σ, namely infinite words. For two words $w, w' \in \Sigma^*$, we write $w' \preceq w$ if w' is a *prefix* of w, and denote by ww' the *concatenation* of w and w'.

The DES is modeled as a labeled finite state automaton:

$$G = (X, E, f, x_0, AP, L),$$

where X is the finite state space; E is the set of events; $f : X \times E \to X$ is the partial transition function whose domain may be extended to $X \times E^*$ in a recursive manner as $f(x, se) = f(f(x, s), e)$ for $x \in X$, $s \in E^*$ and $e \in E$, where E^* denotes the set of all finite *strings* of elements of E, including the empty string ϵ; $x_0 \in X$ is the initial state; AP is the set of atomic propositions, i.e., statements about certain properties that are either true or false; and $L : X \to 2^{AP}$ is the labeling function where for a state $x \in X$, $p \in L(x)$ implies that the atomic proposition $p \in AP$ holds true at x. Without loss of generality, we only consider the *accessible part* of the automata in this work.

Given a DES G, we introduce the following notions and concepts. If an event $e \in E$ is defined at a state $x \in X$, we call it *active* or *feasible* at x. We write $f(x, e)!$ if $e \in E$ is active at $x \in X$, where ! stands for being defined. For $x_1, x_2 \in X$ and $e \in E$, we also write $x_1 \xrightarrow{e} x_2$ if $f(x_1, e) = x_2$. Then we define $R_G(x) = \{x' \in X : (\exists e \in E)[x \xrightarrow{e} x']\}$ as the set of *direct successor states* of x reachable via one transition. The *language* generated by G is a set of strings defined as $\mathcal{L}(G) = \{s \in E^* : f(x_0, s)!\}$.

Additionally, a *run* generated by G is a sequence of alternating states and events of the form: $r = x_1 \xrightarrow{e_1} x_2 \xrightarrow{e_2} \cdots$, which is either finite or infinite. In addition, a run is termed *initial* if it starts from the initial state x_0 of G. We denote by $Run(G)$ the set of all runs generated by G, which also includes the empty run (a single state without events). We write $x \in r$ if state x is included in run r. A run $r = x_1 \xrightarrow{e_1} x_2 \xrightarrow{e_2} \cdots$ of G generates a *trace* $L(r) = L(x_1)L(x_2)\cdots$ over 2^{AP} by concatenating the labels of each state in the run. We denote by $Trace(G)$ the set of all *infinite traces* generated by G.

We quantitatively measure the *distance* between states of a DES G and introduce *metric function* $d : X \times X \to \mathbb{R}_0^+$ where $\forall x, y, z \in X$, the following conditions hold: (i) $d(x, y) = 0 \Leftrightarrow x = y$ (identity); (ii) $d(x, y) = d(y, x)$ (symmetry); (iii) $d(x, z) \leq d(x, y) + d(y, z)$ (triangle inequality). In addition, the distance between a state to a set of states is defined as for any $x \in X$ and $Q \subseteq X$: $d(x, Q) = \min_{x' \in Q} d(x, x')$, i.e., the shortest distance from x to a state in Q. If a DES G is equipped with d defined on its state space, it is called a *metric DES* and denoted by (G, d).

Example 1 Consider the automaton G in Fig. 1. The event set is $E = \{a, b, c, d, u\}$ and the state space is $X = \{x_0, \cdots, x_7\}$. Let the set of atomic propositions be $AP = X$, and the labeling function be defined as: for all $x_i \in X$, $L(x_i) = \{x_i\}$. The metric function $d : X \times X \to \mathbb{R}_0^+$ is also defined and the distance between every two states is summarized in the following table, which satisfies the above three conditions.

2.2 Formal specifications

The specifications, in this work, of the DESs are defined by a special subclass of *linear temporal logic (LTL)*, namely *syntactically co-safe LTL (scLTL)*, see, e.g., Belta et al. (2017). Formally, an scLTL formula over a set of atomic propositions AP is defined as

$$\varphi := true \mid a \mid \neg a \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \bigcirc \varphi \mid \varphi_1 \mathcal{U} \varphi_2,$$

where $a \in AP$ is an atomic proposition, φ, φ_1 and φ_2 are scLTL formulas, $\bigcirc$ and $\mathcal{U}$ are the "next" and "until" operators, respectively. We also define the temporal operator *eventually* as $\Diamond \varphi := true\, \mathcal{U} \varphi$.

Given a set of atomic propositions AP, a *word* is a finite or infinite sequence of atomic propositions. For an scLTL formula φ and an infinite word $w = A_0 A_1 A_2 \cdots \in (2^{AP})^\omega$, we denote by $w[j \cdots] = A_j A_{j+1} A_{j+2} \cdots$, and write $w \vDash \varphi$ if w satisfies φ. Then the scLTL semantics are formally defined as follows:

- $w \vDash true$;
- $w \vDash a$ iff $a \in A_0$, i.e., $A_0 \vDash a$;
- $w \vDash \neg a$ iff $a \notin A_0$, i.e., $A_0 \nvDash a$;
- $w \vDash \varphi_1 \wedge \varphi_2$ iff $w \vDash \varphi_1$ and $w \vDash \varphi_2$;
- $w \vDash \bigcirc \varphi$ iff $w[1 \cdots] = A_1 A_2 \cdots \vDash \varphi$;
- $w \vDash \varphi_1 \mathcal{U} \varphi_2$ iff $\exists j \geq 0 : w[j \cdots] \vDash \varphi_2$ and $\forall 0 \leq i < j : w[i \cdots] \vDash \varphi_1$.

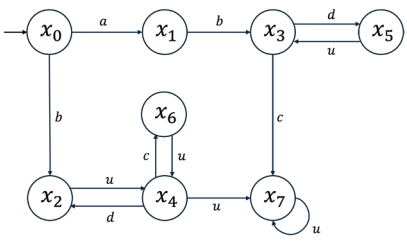

Fig. 1 A metric DES (G, d)

Table 1 Distance between states of G in Fig. 1

	x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7
x_0	0	4	2	5	5	5	6	6
x_1		0	3	3	3	3	3	4
x_2			0	4	4	5	5	5
x_3				0	2	3	3	1
x_4					0	2	2	2
x_5						0	2	3
x_6							0	3
x_7								0

The set of *infinite words* that satisfy φ is defined as $Word(\varphi) = \{w \in (2^{AP})^\omega : w \vDash \varphi\}$. The traces of a DES are evaluated by scLTL formulas and we say that G satisfies φ, denoted by $G \vDash \varphi$, if $L(r) \vDash \varphi$ for all infinite run r, i.e., $Trace(G) \subseteq Word(\varphi)$.

Checking whether an scLTL formula holds is known to be equivalent with inspecting if an infinite word has a *good prefix* such that all its infinite continuations satisfy the formula. Additionally, any scLTL formula can be translated into a DFA that accepts such good prefixes, see Kupferman and Vardi (2001) for details.

Definition 1 (Translate scLTL to DFA). An scLTL formula φ over the set of atomic propositions AP is translated to a DFA $A_\varphi = (X_A, \Sigma_A, f_A, x_{A,0}, F_A)$, where X_A is the state space; $\Sigma_A = 2^{AP}$ is the set of events; $f_A : X_A \times \Sigma_A \to X_A$ is the transition function; $x_{A,0} \in X_A$ is the initial state; and $F_A \subseteq X_A$ is the set of accepting states.

By definition, $\mathcal{A}_\varphi$ accepts all good prefixes of words that satisfy φ, thus the satisfaction of φ can be restated in a *reachability* manner. For an infinite word $w \in (2^{AP})^\omega$, we have $w \vDash \varphi \Leftrightarrow \exists w' \in (2^{AP})^\omega : w' \preceq w \wedge f_A(x_{A,0}, w') \in F_A$. We also assume that there are no deadlock or livelock states in A_φ, which is without loss of generality since these states can be removed by the trim operation (Cassandras and Lafortune 2021).

2.3 Supervisory control

We briefly review the supervisory control theory of DESs (Cassandras and Lafortune 2021). Given a DES G, a supervisor is a function $S : \mathcal{L}(G) \to 2^E$ that dynamically enables and disables events to manipulate the behaviors of the system. We let $\mathbb{S}$ be the set of supervisors and $\Gamma \subseteq 2^E$ be the set of control decisions. By convention, a control decision $\gamma \in \Gamma$ is said to be the set of events enabled by the supervisor. The event set E is partitioned as $E = E_c \cup E_{uc}$, where E_c and E_{uc} represent the sets of controllable and uncontrollable events, respectively. A control decision is termed *admissible* if $E_{uc} \subseteq \gamma$, i.e., no uncontrollable event is disabled, and we only consider admissible control decisions in this work. The supervised system under $S \in \mathbb{S}$ is denoted by S/G, and its generated language, denoted by $\mathcal{L}(S/G)$, is defined recursively as follows:

(i) $\epsilon \in \mathcal{L}(S/G)$;

(ii) $[s \in \mathcal{L}(S/G) \wedge se \in \mathcal{L}(G) \wedge e \in S(s)] \Leftrightarrow [se \in \mathcal{L}(S/G)]$.

A supervisor generally makes decisions based on the whole *history* of past states and enabled events. In other words, memory is necessary for decision making. A special class of supervi-

sors are called *memoryless* where $\forall s, s' \in \mathcal{L}(S/G) : f(x_0, s) = f(x_0, s') \Rightarrow S(s) = S(s')$. That is, the supervisor always issues the same control command whenever the same state is reached, regardless of strings reaching the state. Intuitively, the supervisor does not have to remember how a state is reached.

In this work, we do not consider the non-blockingness property, and thus do not include marked states in the DES model. Instead, we consider a (weaker) *liveness* property. Formally, a DES G is (weakly) live, aka deadlock-free, if $(\forall x \in X)(\exists e \in E)[f(x, e)!]$, that is, every state has at least one active event, which also implies that every string generated by G can be continued by non-trivial strings (not ϵ). Since liveness is ubiquitously required in various engineering scenarios such as robotics, smart manufacturing, and software systems, we assume that G is live, which is without loss of generality since the assumption can be relaxed by adding self-loops at the terminal states, as done in Sampath et al. (1995). In addition, we will also guarantee that the controlled system is live and consider more complex properties specified by scLTL formulas as stated in the following definition.

Definition 2 (Supervisory control with scLTL specifications). Given a DES $G = (X, E, f, x_0, AP, L)$, an scLTL formula φ over AP, design a supervisor S such that the controlled system S/G satisfies φ ($S/G \vDash \varphi$), i.e., $Trace(S/G) \subseteq Word(\varphi)$.

In order to solve the above problem, we first introduce the *product system* which synchronizes the behaviors of a DES with a given scLTL formula.

Definition 3 (Product system). Given a DES G and an scLTL formula expressed as a DFA A_φ, then the product system of G and A_φ is a tuple $G_\otimes = G \times A_\varphi = (X_\otimes, E_\otimes, f_\otimes, x^0_\otimes, F_\otimes, L_\otimes)$, where

- $X_\otimes = X \times X_A$ is the product state space;
- $E_\otimes = E$ is the set of events;
- $f_\otimes : X_\otimes \times E_\otimes \to X_\otimes$ is the transition function: $\forall (x_G, x_A) \in X_\otimes, \forall e \in E_\otimes$,

$$f_\otimes((x_G, x_A), e) = \{(x'_G, x'_A) : x_G \xrightarrow{e} x'_G, x_A \xrightarrow{L(x'_G)} x'_A\}.$$

- $x^0_\otimes = (x_0, f_A(x_{A,0}, L(x_0)))$ is the initial state;
- $F_\otimes = X \times F_A$ is the set of accepting states;
- $L_\otimes : X_\otimes \to X_A$ is the labeling function, where $\forall x_\otimes = (x_G, x_A) \in X_\otimes$: $L_\otimes(x_\otimes) = x_A$, i.e., each state is labeled by its component from A_φ.

The domain of function $f_\otimes$ is extended to $X_\otimes \times E^*_\otimes$ in a recursive manner, similarly with the case of G. Then for a metric function d defined for a DES G, we extend its domain to $X \times X_A$ and redefine a metric function for the product system as $d_\otimes : X \times X_A \to \mathbb{R}^+_0$, where for all $(x_G, x_A), (x'_G, x'_A) \in X_\otimes$: $d_\otimes((x_G, x_A), (x'_G, x'_A)) = d(x_G, x'_G)$, that is, the distance only depends on the first state component from G.

In general, the scLTL synthesis problem can be solved as a *reachability game* on the product automata (Belta et al. 2017). There are two players, namely, the supervisor (controller) and the environment (adversary), where the supervisor wins the game by *reaching the accepting states*, conversely, the environment aims to spoil the supervisor's objective.

Both players take turns to play following their *strategies*, which determine the corresponding actions at each position. Specifically, it is sufficient for both players to apply *positional*, i.e., *memoryless* strategies to win the reachability game (Apt and Grädel 2011). A *winning strategy* of the controller is then converted to a feasible supervisor in Definition 2. By definition, the product $G_\otimes$ is also a DES and subject to control, therefore we write $S/G_\otimes$ for the controlled product system by the supervisor S. The detailed mechanism of game-theoretic supervisor synthesis will be covered in Sections 5.

2.4 Environmental disturbances

Disturbances arise from the uncertain environment and may intervene with the dynamics of the system. Here we consider a simple model of disturbances and do not specify their physical nature or origins, which are beyond the scope of this work. Given a metric DES (G, d), a function $\delta : X \to \mathbb{R}^+$ is introduced to quantify the disturbingeffects at each state in G.

Then we elaborate on how disturbances interfere with the dynamics of a DES G. For an event e defined in state x and a transition $x \xrightarrow{e} y$, we consider two cases. First, if $\delta(y) \leq d(x, y)$, i.e., the disturbance bound is no greater than the distance between x and y, the transition is not perturbed; otherwise, when $\delta(y) > d(x, y)$, we first define

$$R_G^\delta(x) = \{x' \in X : x' \in R_G(x),\ d(x', y) < \delta(y)\}$$

as the set of states that are reachable from x and within the distance of $\delta(y)$ from y. Then we choose one particular state from $R_G^\delta(x)$ as the state reached under disturbances. Specifically, we require that only transitions labeled by controllable events are potentially disturbed, as stated in the following definition.

Definition 4 (Disturbed transition function). Given a metric DES (G, d) under disturbances δ, the disturbed transition function of G is defined as $f^\delta : X \times E \to X$, where for all transition $x \xrightarrow{e} y$ with $e \in E_c$, we have that:

$$f^\delta(x, e) = \begin{cases} y & \text{if } \delta(y) \leq d(x, y); \\ \arg\max_{x' \in R_G^\delta(x)} d(x', y) & \text{if } \delta(y) > d(x, y), \end{cases}$$

and for all transition $x \xrightarrow{e} y$ with $e \in E_{uc}$, we have $f^\delta(x, e) = y$.

Definition 4 describes the dynamics under disturbances. Note that since $d(y, y) = 0$ by identity property, the transition under disturbances necessarily goes to a state other than the original target state y if $R_G^\delta(x)$ includes not only y. Here we compare the disturbance effect $\delta(y)$ in the target state with the distance $d(x, y)$. This is simply our choice and it is straightforward to reformulate the comparison with respect to $\delta(x)$ and $d(x, y)$. When the disturbance effect is sufficiently large, the reachable state is chosen to be the one with the largest distance from the original reachable state, reflecting the worst potential deviation caused by the disturbances. We assume that disturbances take effects *immediately* after the event occurs. When controllable events are subject to disturbances, the supervisor is still capable of disabling those events preemptively. Then we replace the transition function of a DES G with its disturbed version in Definition 4, and denote the resulting DES by G^δ.

Recall that the scLTL specification is enforced if there exists a control strategy that reaches the accepting states to win the game. However, these states may no longer be reachable when disturbances intervene with the product system. Given a metric DES (G, d), a DFA $A_\varphi = (X_A, \Sigma_A, f_A, x_{A,0}, F_A)$ representing an scLTL formula φ and their product $G_\otimes$, then for a disturbance effect function $\delta : X \to \mathbb{R}^+$ defined for (G, d), we extend its domain to $X \times X_A$ and redefine the function as $\delta_\otimes : X \times X_A \to \mathbb{R}^+$ where $\forall (x_G, x_A) \in X \times X_A$: $\delta_\otimes(x_G, x_A) = \delta(x_G)$, i.e., disturbance effects only depend on the first state component from G. Again, we require that only transitions labeled by controllable events are disturbed. Then the *disturbed transition function* of $G_\otimes$ is defined as $f^\delta_\otimes : X_\otimes \times E \to X_\otimes$, where for all transition $x_\otimes \xrightarrow{e} y_\otimes$ with $e \in E_c$, we have:

$$f^\delta_\otimes(x_\otimes, e) = \begin{cases} y_\otimes & \text{if } \delta(y_\otimes) \leq d(x_\otimes, y_\otimes) \\ \arg\max_{x'_\otimes \in R^\delta_{G_\otimes}(x_\otimes)} d_\otimes(x'_\otimes, y_\otimes) & \text{if } \delta(y_\otimes) > d(x_\otimes, y_\otimes) \end{cases}, \tag{1}$$

where $R^\delta_{G_\otimes}(x_\otimes) = \{x'_\otimes \in X_\otimes : x'_\otimes \in R_{G_\otimes}(x_\otimes),\ d_\otimes(x'_\otimes, y_\otimes) < \delta(y_\otimes)\}$, and for $x_\otimes \xrightarrow{e} y_\otimes$ with $e \in E_{uc}$, we have $f^\delta_\otimes(x_\otimes, e) = y_\otimes$. If we replace the original transition function with the disturbed version defined above, then we denote by $G^\delta_\otimes$ the product system $G_\otimes$ under disturbances δ, and $S/G^\delta_\otimes$ the controlled product system under S.

A supervisor that nominally achieves the given scLTL specification may fail to do so when the system is disturbed. This raises the issue of designing supervisors that are resilient to disturbances such that the strings of the controlled system may reach states proximal to the accepting states, which will be investigated in later sections.

3 Problem formulation

Since the given specification may no longer be enforced when the system is disturbed, it is essential to design supervisors that operate in a *robust* manner to *tolerate* disturbances, thus avoids catastrophic failures. For that reason, we investigate robust supervisory control and formulate two problems for investigation: verification of the existence of sufficiently robust supervisors and synthesis of optimal robust supervisors.

Given a metric DES (G, d), a DFA A_φ representing an scLTL formula φ, their product $G_\otimes$, and disturbance effect function δ, we have the metric function $d_\otimes$ and disturbance effect function $\delta_\otimes$ for $G_\otimes$. Next, we quantify the deviation from the accepting states caused by disturbances and introduce the concept of robust supervisors.

Definition 5 (Inflated accepting states). Given a metric DES (G, d), a DFA A_φ representing an scLTL formula φ, their product $G_\otimes$ with accepting state set $F_\otimes$ and a metric function $d_\otimes$, then for a constant $\sigma \in \mathbb{R}^+_0$, the σ-inflated accepting state set of $G_\otimes$ is defined as $F^\sigma_\otimes = \{(x_G, x_A) \in X \times Q : d_\otimes((x_G, x_A), F_\otimes) \leq \sigma\}$.

Definition 6 (Robust supervisors). Given a metric DES (G, d), a DFA A_φ representing an scLTL formula φ, their product $G_\otimes$ and a disturbance effect function δ, then for a constant $\sigma \in \mathbb{R}^+_0$, supervisor S is called σ-robust with respect to δ if for all $\rho \in Trace(S/G^\delta_\otimes)$, there exists $x_\otimes \in F^\sigma_\otimes$, such that $L_\otimes(x_\otimes)$ appears in ρ.

Our fundamental inspiration comes from classic robust control theory, see, e.g., Sontag (1998), where bounded disturbances only cause a modest deviation from desired behaviors. The scLTL formula may become unachievable subject to disturbances, i.e., the original accepting states $F_\otimes$ of the product system are not guaranteed to be reached. However, the traces of the controlled system may still reach states that are "not too far from" $F_\otimes$ when the disturbances are bounded. The robustness of supervisors is in a topological sense where σ measures the degree of tolerance against adversarial perturbation. A smaller σ indicates that the behaviors of the controlled system deviate less substantially from the original accepting states $F_\otimes$, which means the supervisor is more robust. Note that if a supervisor is σ-robust, it is naturally σ'-robust for all $\sigma' > \sigma$. When there is no confusion, we simply call a supervisor robust if it is σ-robust for some σ. Next, we formulate the two key problems of robust supervisory control in this work.

Problem 1 (Existence of sufficiently robust supervisors). Given a metric DES (G, d), a DFA A_φ for an scLTL formula φ, their product $G_\otimes$, disturbances with effect function δ and constant $\sigma \geq 0$, verify if there exists a live and σ-robust supervisor.

Problem 2 (Synthesis of optimal robust supervisors). Given a metric DES (G, d), a DFA A_φ for an scLTL formula φ, their product $G_\otimes$, and a disturbance effect function δ, then synthesize an optimal robust supervisor S_{opt} such that S_{opt} is live and σ_{min}-robust, where $\sigma_{min} = \min_{S \in \mathbb{S}}\{\sigma \in \mathbb{R}_0^+ : S \text{ is } \sigma\text{-robust }\}$.

Remark 1 Here σ_{min} is the optimal robustness measure for all supervisors under the disturbance δ in the sense that there do not exist supervisors with a smaller σ. Also, S_{opt} may not be unique since different supervisors share the same σ_{min} and drive the system to reach the same inflated set of accepting states.

In the remainder of the work, we will propose a game-theoretic framework to solve the above two problems. The following example concludes this section.

Example 2 Consider the system in Example 1 and let $E_{uc} = \{u\}$. The scLTL specification is $\Diamond x_7$, i.e., eventually reaching x_7. The disturbance effect function δ is defined as: $\delta(x_0) = \delta(x_1) = \delta(x_2) = \delta(x_3) = \delta(x_4) = \delta(x_6) = 1$, $\delta(x_5) = 2$ and $\delta(x_7) = 4$. Suppose that there is a supervisor S, which enables a at x_0; b at x_1; c at x_3; u at x_7, see Fig. 2. Obviously, if there are no disturbances, S/G achieves the specification. However, if there are disturbances, transition $x_3 \xrightarrow{c} x_7$ will be disturbed since $\delta(x_7) > d(x_3, x_7)$, and will

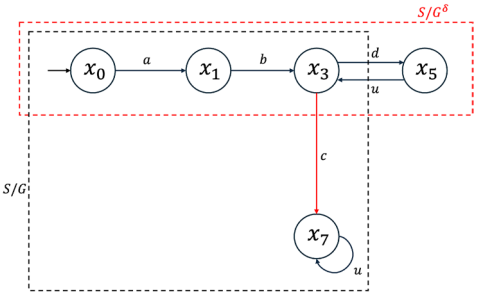

Fig. 2 In the absence of disturbances, the supervisor S/G enables actions that successfully reach the target state x_7. However, when disturbances are introduced into the system G, the modified system S/G^δ may enable action c at state x_3, resulting in a transition to x_5 instead of the intended x_7. This deviation from the optimal path illustrates how disturbances can negatively impact system performance

be replaced by $x_3 \xrightarrow{d} x_5$ due to $R_G^{\delta}(x_3) = R_G(x_3) = \{x_5, x_7\}$ and $d(x_5, x_7) > d(x_7, x_7)$, making x_7 unreachable under S.

4 Two-player game structure

In this section, we first transform the automaton model in Section 2 to a bipartite structure called *disturbance bipartite transition system (DBTS)* which represents the game between the supervisor and the disturbances (environment). We then analyze the properties of DBTSs and construct the largest DBTS which includes live supervisors and will serve as the basis for addressing Problems 1 and 2 in the following section.

Definition 7 (Disturbance bipartite transition system). A DBTS with respect to a metric DES (G, d), a DFA A_φ representing an scLTL formula φ, their product $G_\otimes$, and a disturbance effect function δ, is a tuple $T = (Q_Y, Q_Z, f_{yz}, f_{zy}, E, \Gamma, q_y^0)$ such that

- $Q_Y \subseteq X_\otimes \times \mathbb{R}_0^+$ is the set of Y-states, and for $q_y \in Q_Y$, we write $q_y = (\mathcal{S}(q_y), \mathcal{D}(q_y))$, where $\mathcal{S}(q_y)$ and $\mathcal{D}(q_y)$ denote the state and distance components of q_y, respectively;
- $Q_Z \subseteq X_\otimes \times \mathbb{R}_0^+ \times \Gamma$ is the set of Z-states, and for $q_z \in Q_Z$, we write $q_z = (\mathcal{P}(q_z), \Gamma(q_z))$, where $\mathcal{P}(q_z) = (\mathcal{S}(\mathcal{P}(q_z)), \mathcal{D}(\mathcal{P}(q_z)))$ and $\Gamma(q_z)$ denote the state-distance pair and control decision components of q_z, respectively;
- $f_{yz} : Q_Y \times \Gamma \to Q_Z$ is the transition from Q_Y to Q_Z states, and it is defined as $\forall q_y \in Q_Y, \forall \gamma \in \Gamma$, and $\forall q_z \in Q_Z$:

$$f_{yz}(q_y, \gamma) = q_z \Rightarrow [\mathcal{P}(q_z) = q_y] \wedge [\Gamma(q_z) = \gamma];$$

- $f_{zy} : Q_Z \times E \to Q_Y$ is the transition from Q_Z to Q_Y states, and it is defined as $\forall q_z \in Q_Z, \forall e \in E$, and $\forall q_y \in Q_Y$:

$$f_{zy}(q_z, e) = q_y \Rightarrow \\ [e \in \Gamma(q_z)] \wedge [\mathcal{S}(q_y) = f_\otimes^{\delta}(\mathcal{S}(\mathcal{P}(q_z)), e)] \wedge [\mathcal{D}(q_y) = d_\otimes(\mathcal{S}(q_y), F_\otimes)];$$

- E is the set of events of G;
- Γ is the set of admissible control decisions;
- $q_y^0 = (x_\otimes^0, d_\otimes(x_\otimes^0, F_\otimes)) \in Q_Y$ is the initial state.

In essence, a DBTS constitutes a two-player game arena, where the supervisor plays against the environment (disturbances). The supervisor plays at Y-states by issuing control decisions and the environment plays at Z-states by determining the new reachable states under the last enabled events and potential disturbances. A transition from a Y-state q_y to a Z-state q_z records the last issued control command. At this stage, the state-distance pair remains unchanged, i.e., $\mathcal{P}(q_z) = q_y$, since we assume that the enabled events have not occurred yet. The control command $\Gamma(q_z) = \gamma$ is issued at the preceding Y-state. A transition from a Z-state q_z to a Y-state q_y updates the reachable state-distance pair under control decisions and disturbances. To be more specific, a f_{zy} transition labeled by event e leads from a

Z-state q_z to a *Y*-state q_y, following the disturbed transition function $f^{\delta}_{\otimes}$ of the product system $G_{\otimes}$. Meanwhile, the distance component is updated accordingly to track the distance from q_y to the set of accepting states $F_{\otimes}$ of the product system $G_{\otimes}$.

Given a DBTS *T*, $q_y \in Q_Y$ and $q_z \in Q_Z$, we let $Post(q_y) = \{q_z \in Q_Z : (\exists \gamma \in \Gamma)[f_{yz}(q_y, \gamma) = q_z]\}$ and $Post(q_z) = \{q_y \in Q_Y : (\exists e \in E)[f_{zy}(q_z, e) = q_y]\}$ be the set of direct successors of q_y and q_z, respectively. A run in *T* is a sequence of alternating states, control decisions, and events, which is either finite or infinite and of the form:

$$r = q_y^1 \xrightarrow{\gamma_1} q_z^1 \xrightarrow{e_1} q_y^2 \xrightarrow{\gamma_2} q_z^2 \xrightarrow{e_2} q_y^3 \cdots$$

We denote by *Run*(*T*) the set of runs in *T* and by *Run*(*T*, *q*) the set of runs in *T* starting from a state $q \in Q_Y \cup Q_Z$, also write $Run_y(T)$ (respectively $Run_z(T)$) as the set of runs ending with a *Y*-state (respectively a *Z*-state).

Definition 7 is generic and the following conditions are imposed on a DBTS *T* for the purpose of supervisor synthesis:

(i) $\forall q_y \in Q_Y$: $C_T(q_y) = \{\gamma \in \Gamma : f_{yz}(q_y, \gamma)!\} \neq \emptyset$;
(ii) $\forall q_z \in Q_Z$: $(\forall e \in \Gamma(q_z))[f^{\delta}_{\otimes}(\mathcal{S}(\mathcal{P}(q_z)), e)!] \Leftrightarrow f_{zy}(q_z, e)!$.

The first condition says that the supervisor is capable to choose at least one feasible control decision at any *Y*-state of *T*. The second states that all enabled events should occur from any *Z*-state since the supervisor is not allowed to decide which event to occur once they are enabled. A DBTS *T* is termed *complete* if the above two conditions are met, and the game never terminates in states where no actions are available for the two players.

In a DBTS *T*, both player follow their *strategies* to select the next action in their respective positions. In general, strategies are based on runs of *T*, i.e., the whole history of past states, events, and control decisions. Accordingly, we define the *supervisor's strategy (control strategy)* as $\pi_s : Run_y(T) \to \Gamma$ and the *environment's strategy* as $\pi_e : Run_z(T) \to E$. We denote the set of all supervisor's and environment's strategies by Π_s and Π_e, respectively. A player selects the corresponding transition at its positions following its strategy. It turns out that a control strategy included in a DBTS works in the same way as a standard supervisor, thus we will use the terms "supervisor" and "supervisor's strategy (control strategy)" interchangeably in the following discussions. Moreover, a strategy $\pi_i \in \Pi_i$ of player $i \in \{s, e\}$ is *positional* if its decisions depend only on the current state. Specifically, a control strategy π_s is positional if $\forall r, r' \in Run(T)$, $Last_Y(r) = Last_Y(r') \Rightarrow \pi_s(r) = \pi_s(r')$, where $Last_Y(r)$ stands for the last *Y*-state of run *r*. Positional strategies for the environment are defined analogously and omitted here for simplicity of notions.

At a state in a DBTS, a set of runs is generated if a player's strategy is fixed. Specifically, given a *Y*-state q_y and a control strategy π_s, we define

$$Run(q_y, \pi_s) = \{q_y \xrightarrow{\gamma_1} q_z^1 \xrightarrow{e_1} q_y^2 \xrightarrow{\gamma_2} q_z^2 \xrightarrow{e_2} \cdots \mid \forall n \in \mathbb{N}^+ : \gamma_n = \pi_s(q_y \xrightarrow{\gamma_1} q_z^1 \xrightarrow{e_1} q_y^2 \xrightarrow{\gamma_2} \cdots q_y^n)\}$$

as the set of runs starting from q_y and *consistent* with π_s, i.e., each control decision is specified by π_s. Additionally, given a *Z*-state q_z, we define

$$Run(q_z, \pi_s) = \{q_z \xrightarrow{e_1} q_y^1 \xrightarrow{\gamma_1} q_z^2 \xrightarrow{e_2} q_y^2 \xrightarrow{\gamma_2} \cdots \mid \forall n \in \mathbb{N}^+ : \gamma_n = \pi_s(q_z \xrightarrow{e_1} q_y^1 \xrightarrow{\gamma_1} q_z^2 \xrightarrow{e_2} \cdots q_y^n)\}$$

as set of runs starting from q_z and *consistent* with π_s. Runs consistent with the environment's strategies are defined in a similar manner.

Next, we "decode" a supervisor following the control strategies and generated runs from a complete DBTS T. First, we let $Q_Y^S(q_y, s)$ be the Y-state resulting from the occurrence of string s starting from Y-state q_y. Specifically, for $s \in E^*$ and $e \in E$, $Q_Y^S(q_y, s)$ is computed in a recursive manner: (i) $Q_Y^S(q_y, \epsilon) = q_y$; (ii) $Q_Y^S(q_y, se) = f_{zy}(f_{yz}(Q_Y^S(q_y, s), S(s)), e)$ if $e \in S(s)$. Then a supervisor S is said to be *included* in T if $\forall s \in \mathcal{L}(S/G_\otimes^\delta), S(s) \in C_T(Q_Y^S(q_y, s))$. That is, after string $s$occurs, the next control decision is chosen from the reached Y-state. Also, $Q_Z^S(q_z, s)$ is computed similarly: (i) $Q_Z^S(q_z, \epsilon) = q_z$; (ii) $Q_Z^S(q_z, se) = f_{yz}(f_{zy}(Q_Z^S(q_z, s), e), S(s))$ if $e \in S(s)$.

A DBTS T is termed *deterministic* if $\forall q_y \in Q_Y$, $|C_T(q_y)| = 1$, i.e., a unique control decision, denoted by $c_T(q_y)$, is defined at each Y-state. It is straightforward that a deterministic DBTS T induces a unique supervisor, which is denoted by S_T. In addition, the *realization* of S_T is represented by an automaton $G_T = (Q_Y, E, \xi, q_0)$, where Q_Y is the set of Y-states in T; $\xi : Q_Y \times E \to Q_Y$ is the (partial) transition function defined as: $\forall q_y \in Q_Y$, $\xi(q_y, e) = f_{zy}(f_{yz}(q_y, c_T(q_y)), e)$ if $e \in c_T(q_y)$. Then the language of the supervised system is denoted by $\mathcal{L}(S_T/G_\otimes^\delta) = \mathcal{L}(G_T \times G_\otimes^\delta)$, where $\times$ is the product operation of automata, see, e.g., Cassandras and Lafortune (2021).

In order to enforce liveness by the supervisors induced by DBTSs, we introduce deadlock-free Z states. Then, inspired by Lemma V.1 of Yin and Lafortune (2016), we also show that the liveness property can be transformed into a Z-state property.

Definition 8 (Deadlock-free Z-states). In a DBTS T, a Z-state q_z is called deadlock-free if $\exists e \in \Gamma(q_z) : f_\otimes^\delta(\mathcal{S}(\mathcal{P}(q_z)), e)!$; otherwise it is called a deadlock Z-state.

Lemma 1 *For any supervisor S included by a complete DBTS T, the supervised system $S/G_\otimes^\delta$ is live if and only if all Z-states in T are deadlock-free. Formally,*

$$(\forall q_z \in Q_Z)(\exists e \in \Gamma(q_z))[f_\otimes^\delta(\mathcal{S}(\mathcal{P}(q_z)), e)!] \iff (\forall s \in \mathcal{L}(S/G_\otimes^\delta))(\exists e \in S(s))[f_\otimes^\delta(x_0, se)!].$$

Proof "$\Rightarrow$": By contrapositive. Assume that $S/G_\otimes^\delta$ is not live, i.e., $(\exists s \in \mathcal{L}(S/G_\otimes^\delta))(\forall e \in S(s))[f_\otimes^\delta(x_0, se)\not!]$ where $\not!$ stands for "not defined". Since the product in Definition 3 and disturbances do not affect the liveness of the supervised system, by Definition 7, the above formula is reformulated as $(\exists s \in \mathcal{L}(S/G_\otimes^\delta))(\exists q_z = Q_Z^S(f_{yz}(q_y^0, S(\epsilon)), s) \in Q_Z)(\forall e \in \Gamma(q_z))[f_\otimes^\delta(\mathcal{S}(\mathcal{P}(q_z)), e)\not!]$, which completes the proof.

"$\Leftarrow$": By contrapositive. Suppose that $(\exists q_z \in Q_Z)(\forall e \in \Gamma(q_z))[f_\otimes^\delta(\mathcal{S}(\mathcal{P}(q_z)), e)\not!]$. By definitions of f_{yz} and f_{zy} in Definition 7, we know that there exists $s \in \mathcal{L}(S/G_\otimes^\delta)$ such that $q_z = Q_Z^S(f_{yz}(q_y^0, S(\epsilon)), s)$ and $\Gamma(q_z) = S(s)$, where S is a supervisor *included* in T. Note that by Definition 7, $\mathcal{S}(\mathcal{P}(q_z)) = f_\otimes^\delta(x_0, s)$. In conclusion, there exists a supervisor

S included in T, such that $(\exists s \in \mathcal{L}(S/G^{\delta}_{\otimes}))(\forall e \in S(s))[f^{\delta}_{\otimes}(x_0, se)!]$, which completes the proof. □

Lemma 1 entails that if we build a DBTS that is "as large as possible" and has all its Z-states being deadlock-free, then it should include all live supervisors. Then we compare the size of DBTSs in a graph merging sense: given $T_i = (Q^i_Y, Q^i_Z, E, \Gamma, f^i_{yz}, f^i_{zy}, y^i_0)$ for $i \in \{1, 2\}$, we say that T_1 is a *subgraph* of T_2, denoted by $T_1 \sqsubseteq T_2$, if (i) $Q^1_Y \subseteq Q^2_Y, Q^1_Z \subseteq Q^2_Z$; and (ii) for all $q_y \in Q^1_Y$, $q_z \in Q^1_Z$, $\gamma \in \Gamma$, and $e \in E$, we have that $f^1_{yz}(q_y, \gamma) = q_z \Rightarrow f^2_{yz}(q_y, \gamma) = q_z$ and $f^1_{zy}(q_z, e) = q_y \Rightarrow f^2_{zy}(q_z, e) = q_y$. Intuitively, T_2 is also said to be "larger" than T_1. Inspired by this notion, we introduce the concept of the *all live structure (ALS)*.

Definition 9 (All live structure). Given a metric DES (G, d), a DFA A_φ representing an scLTL formula φ, their product $G_\otimes$, and a disturbance effect function δ, the ALS, denoted by $T_{max} = (Q^m_Y, Q^m_Z, E, \Gamma, f^m_{yz}, f^m_{zy}, q^0_y)$, is the largest DBTS such that:

1) $\forall q_y \in Q^m_Y$: $C_{T_{max}}(q_y) \neq \emptyset$;
2) $\forall q_z \in Q^m_Z$: q_z is deadlock-free and $(\forall e \in \Gamma(q_z))[f^{\delta}_{\otimes}(\mathcal{S}(\mathcal{P}(q_z)), e)!] \Leftrightarrow f_{zy}(q_z, e)!$;
3) for all DBTS T satisfying 1) and 2), we have that $T \sqsubseteq T_{max}$.

ALS is a special from of DBTS that enables efficient and effective derivation of valid robust supervisors. Conditions 1) and 2) in Definition 9 jointly guarantee that the ALS is complete. If two DBTSs T_1 and T_2 satisfy these conditions, then their union, denoted by $T_1 \cup T_2$, also satisfies them, where the union is defined as: (i) $Q^{1\cup2}_Y = Q^1_Y \cup Q^2_Y, Q^{1\cup2}_Z = Q^1_Z \cup Q^2_Z$; (ii) for all $q_y \in Q^{1\cup2}_Y$, $q_z \in Q^{1\cup2}_Z$, $\gamma \in \Gamma$, and $e \in E$, we have that $f^{1\cup2}_{yz}(q_y, \gamma) = q_z \Leftrightarrow \exists i \in \{1, 2\} : f^i_{yz}(q_y, \gamma) = q_z$ and $f^{1\cup2}_{zy}(q_z, e) = q_y \Leftrightarrow \exists i \in \{1, 2\} : f^i_{zy}(q_z, e) = q_y$. Hence, the notion of being the largest is well-defined in the sense of graph merging.

Algorithm 1 Build the All Live Structure.

Input: G, φ, d, δ
Output: $T_{max} = (Q^m_Y, Q^m_Z, E, \Gamma, f^m_{yz}, f^m_{zy}, q^0_y)$
Build DFA A_φ to represent φ and take the product $G_\otimes = G \times A_\varphi$;
$Q^m_Y = \{q^0_y\} = \{(x^0_\otimes, d(x^0_\otimes, F_\otimes))\}$, $Q^m_Z = \emptyset$;
$T^{pre}_{max} = DoDFS(q^0_y, G_\otimes, d_\otimes, \delta_\otimes)$;
while *there exist Y-states without successors* **do**
  Remove such states and their predecessor Z-states;
Take the accessible part of the remaining structure and return T_{max};
Procedure: $DoDFS(q_y, G_\otimes, d_\otimes, \delta_\otimes)$
for $\gamma \in \Gamma$ **do**
  $q_z = f_{yz}(q_y, \gamma)$ by Definition 7;
  if q_z *is deadlock-free* **then**
    Add transition $q_y \xrightarrow{\gamma} q_z$ to f^m_{yz};
    if $q_z \notin Q^m_Z$ **then**
      $Q^m_Z = Q^m_Z \cup \{q_z\}$;
      for $e \in \gamma$ **do**
        $\tilde{q}_y = f_{zy}(q_z, e)$ by Definition 7;
        Add transition $q_z \xrightarrow{e} \tilde{q}_y$ to f^m_{zy};
        if $\tilde{q}_y \notin Q^m_Y$ **then**
          $Q^m_Y = Q^m_Y \cup \{\tilde{q}_y\}$;
          $DoDFS(\tilde{q}_y, G_\otimes, d_\otimes, \delta_\otimes)$;

Now we are ready to present Algorithm 1 to construct the ALS, which follows directly from Definition 9 and proceeds in two steps. First, a depth-first search of procedure *DoDFS* is initiated from the initial state q_y^0 and called recursively until no new states and transitions are available, eventually resulting in a DBTS T_{max}^{pre}. Specifically, only deadlock-free *Z*-states are added to the structure to ensure the liveness property. After *DoDFS*, there may be *Y*-states without successors, i.e., no feasible control actions. This is not allowed in the settings of supervisory control theory, thus the next step from Line 5 is to remove such states as well as their predecessor *Z*-states since enabled events in *Z*-states should not be blocked from occurring. The removal process may be interpreted as calculating the supremal controllable sublanguage in supervisory control theory of DES if we view states without successors as undesired, transitions for f_{yz} as controllable, and transitions for f_{zy} as uncontrollable. Finally, states that are no longer accessible from the initial state are removed before the final ALS is returned. The algorithm converges after a finite number of steps since the number of *Y*-states and *Z*-states is finite by Definition 9.

The following result states that the ALS T_{max} built by Algorithm 1 contains all live supervisors *S*. Intuitively, this is because Algorithm 1 initially includes all possible control actions, thus accounting for all potential *Z*-states, and subsequently removes only those *Z*-states that are identified as non-live.

Lemma 2 *A supervisor S is live if and only if it is included in* T_{max}*. Formally,*

$$(\forall s \in \mathcal{L}(S/G_{\otimes}^{\delta}))(\exists e \in S(s))[f_{\otimes}^{\delta}(x_0, se)!] \Leftrightarrow \forall s \in \mathcal{L}(S/G_{\otimes}^{\delta}), S(s) \in C_{T_{max}}(Q_Y^S(q_y, s)).$$

Proof The "⇐" part follows directly from Lemma 1 and Definition 9.

"⇒": Suppose that *S* is supervisor such that $(\forall s \in \mathcal{L}(S/G_{\otimes}^{\delta}))(\exists e \in S(s))[f_{\otimes}^{\delta}(x_0, se)!]$ holds. By contradiction, we assume that $\exists s \in \mathcal{L}(S/G_{\otimes}^{\delta}), S(s) \notin C_{T_{max}}(Q_Y^S(q_y, s))$. First, we know that there exists a complete DBTS *T* such that $\forall s \in \mathcal{L}(S/G_{\otimes}^{\delta}), S(s) \in C_T(Q_Y^S(q_y, s))$. Specifically, the complete DBTS *T* can be constructed as follows: $Q_Y := \{q_y \in X_{\otimes} \times \mathbb{R}_0^+ : \exists s \in \mathcal{L}(S/G_{\otimes}^{\delta}) \text{ s.t. } q_y = Q_Y^S(q_y^0, s)\}$, $Q_Z := \{q_z \in X_{\otimes} \times \mathbb{R}_0^+ \times \Gamma : \exists s \in \mathcal{L}(S/G_{\otimes}^{\delta}) \text{ s.t. } q_z = Q_Z^S(f_{yz}(q_y^0, S(\epsilon)), s)\}$ and for any $q_y \in Q_Y$, $C_T(q_y) := \{\gamma \in \Gamma : \exists s \in \mathcal{L}(S/G_{\otimes}^{\delta}) \text{ s.t. } q_y = Q_Y^S(q_y^0, s) \wedge \gamma = S(s)\}$. In other words, any *Y* or *Z*-state in *T* are a *Y* or *Z*-state reached by supervisor *S* under some string $t \in \mathcal{L}(S/G_{\otimes}^{\delta})$, respectively. By Lemma 1, we know that *T* satisfies the conditions in Definition 9. In addition, $\exists s \in \mathcal{L}(S/G_{\otimes}^{\delta}), S(s) \notin C_{T_{max}}(Q_Y^S(q_y, s))$. In this case, the union of *T* and T_{max} is strictly larger than T_{max}, since control decision $S(s)$ is defined at *Y*-state $Q_Y^S(q_y^0, s)$ in $T \cup T_{max}$ but not in T_{max}. This is a contradiction since by definition, T_{max} is the largest DBTS satisfying the conditions in Definition 9. □

5 Game-theoretic verification and synthesis

This section proposes a game-theoretic framework to comprehensively address Problems 1 and 2. Initially, we formulate a reachability game with properly defined target states on the ALS, allowing us to compute the winning region of the game and determine the existence

of robust supervisors. Subsequently, we develop a dynamic programming method to evaluate the optimal robustness measure at each state of the ALS, which ultimately leads to an optimal robust supervisor.

5.1 Verification of the existence of robust supervisors

For the verification problem, we first formulate a *reachability game* with corresponding objectives on the ALS. Then we compute the *winning region* of the game to determine if robust supervisors exist under the given disturbances, thus solving problem 1.

A reachability game is played between the supervisor and the environment (disturbances) on the ALS, where the supervisor wins by reaching the target states, and the environment wins otherwise. We define the set of *target states* as $\mathcal{Q}_R = \{q_y \in Q_Y^m : \mathcal{S}(q_y) \in F_\otimes^\sigma\}$, where $F_\otimes^\sigma = \{(x_G, x_A) \in X \times Q : d_\otimes((x_G, x_A), F_\otimes) \leq \sigma\}$ is the set of σ-inflated accepting states defined in Definition 5 and $F_\otimes$ is the set of accepting states of $G_\otimes$. In addition, we let $\mathcal{Q}_R^c = Q_Y^m \setminus \mathcal{Q}_R$ be the complement of $\mathcal{Q}_R$. Then an initial (infinite) run $r = q_y^0 \xrightarrow{\gamma_0} q_z^0 \xrightarrow{e_0} q_y^1 \xrightarrow{\gamma_1} q_z^1 \xrightarrow{e_1} q_y^2 \cdots$ is called *winning* with respect to $\mathcal{Q}_R$ if there exists $q_y^i \in r$ for some $i \geq 0$ such that $q_y^i \in \mathcal{Q}_R$.

The strategies for both players follow from the definitions in Section 4 and it is known that positional (memoryless) strategies suffice for both players to win the reachability games (Apt and Grädel 2011). Therefore, we will restrict our attention to positional strategies in the remainder of this work. Given a state q of the ALS, a control strategy π_s is said to win at q if every run starting from q and consistent with π_s is winning for the supervisor. Consequently, q is called a winning state for the supervisor. The winning region of the supervisor, denoted by $\mathcal{W}_s(T_{max})$, is the set of states where the supervisor has a strategy to achieve $\mathcal{Q}_R$ successfully regardless of the strategies played by the environment. Similarly, the winning states and winning regions are defined for the environment, which are not repeated for simplicity. Also, the winning regions of the two players are disjoint, see, e.g., Apt and Grädel (2011) for more technical discussions.

Reachability games on a finite game graph are solved via a fixed-point manner computation of *attractors* for the set of target states. Intuitively, the attractor of $\mathcal{Q}_R$ with respect to a player contains all states from which the player is capable of forcing a run to eventually reach $\mathcal{Q}_R$. Then the attractors for the supervisor is defined recursively, where a run is enforced to reach $\mathcal{Q}_R$ within at most $i \in \mathbb{N}_0^+$ steps.

Definition 10 (Attractors for the supervisor). Given the ALS T_{max} and the set of target states $\mathcal{Q}_R$, the controlled predecessor $CPre_s(\mathcal{Q}_R)$ of $\mathcal{Q}_R$ is:

$$\begin{aligned} CPre_s(\mathcal{Q}_R) = &\{q_y \in Q_Y^m \mid \exists \gamma \in \Gamma : f_{yz}^m(q_y, \gamma) \in \mathcal{Q}_R\} \cup \\ &\{q_z \in Q_Z^m \mid \forall e \in \Gamma(q_z) : f_{zy}^m(q_z, e) \in \mathcal{Q}_R\} \end{aligned} \tag{2}$$

then the attractor is defined by inductively applying the controlled predecessor:

$$\begin{aligned} Attr_s^0(\mathcal{Q}_R) &= \mathcal{Q}_R; \\ Attr_s^{i+1}(\mathcal{Q}_R) &= Attr_s^i(\mathcal{Q}_R) \cup CPre_s(Attr_s^i(\mathcal{Q}_R)); \\ Attr_s(\mathcal{Q}_R) &= \bigcup_{n \in \mathbb{N}_0^+} Attr_s^n(\mathcal{Q}_R). \end{aligned} \tag{3}$$

We start the computation from a given $\mathcal{Q}_R$ and subsequently expand the attractor set by including all *Z*-states where the supervisor will eventually reach $\mathcal{Q}_R$ *for sure*, as well as all *Y*-states where the supervisor has some strategies to reach $\mathcal{Q}_R$. The computation is guaranteed to converge after a finite number of iterations, as shown below, since T_{max} has a finite number of states and transitions (Apt and Grädel 2011):

Lemma 3 *Given the ALS T_{max} and the set of target states $\mathcal{Q}_R$, then for the reachability game, there exists an integer $k \in \mathbb{N}_0^+$ with $k \leq |Q_Y^m \cup Q_Z^m|$ such that:*

$$Attr_s^0(\mathcal{Q}_R) \subseteq Attr_s^1(\mathcal{Q}_R) \subseteq \cdots \subseteq Attr_s^k(\mathcal{Q}_R) = Attr_s^{k+1}(\mathcal{Q}_R) = Attr_s(\mathcal{Q}_R).$$

We leverage the standard fixed-point algorithm in Apt and Grädel (2011) to compute the winning region of the supervisor, whose complexity is known to be linear with respect to the number of states and transitions of the game graph T_{max}.

Lemma 4 *For the reachability game on the ALS T_{max} with the set of target states $\mathcal{Q}_R$, the supervisor's winning region is $\mathcal{W}_s(T_{max}) = Attr_s(\mathcal{Q}_R)$.*

Proof We first define an iteration counter

$$\mathcal{C}(q) = \min\{n \in \mathbb{N}_0^+ \mid q \in Attr_s^n(\mathcal{Q}_R)\}$$

for any state $q \in Q_Y^m \cup Q_Z^m$, which indicates the number of iterations required for a state that is initially in $Attr_s^0(\mathcal{Q}_R)$ to be included in the subsequent attractors. In plain words, $\mathcal{C}(q)$ indicates the distance from q to $\mathcal{Q}_R$. In particular, we let $\min \emptyset = \infty$.

By the definition of the attractors, the following properties hold:

1) $\mathcal{C}(q) = 0$ if and only if $q \in \mathcal{Q}_R = Attr_s^0(\mathcal{Q}_R)$.
2) If $q \in Q_Y^m$ satisfies $0 < \mathcal{C}(q) < \infty$, then q has a successor q' such that $\mathcal{C}(q') < \mathcal{C}(q)$, indicating that q' is one step closer to $\mathcal{Q}_R$. Specifically, such a q satisfies $q \in Attr_s^{\mathcal{C}(q)}(\mathcal{Q}_R) \setminus Attr_s^{\mathcal{C}(q)-1}(\mathcal{Q}_R)$, by Eq. 3, q is in $CPre_s(Attr_s^{\mathcal{C}(q)-1}(\mathcal{Q}_R))$. Since $q \in Q_Y^m$, it follows that

$$q \in \{q_y \in Q_Y^m \mid \exists \gamma \in \Gamma : f_{yz}^m(q_y, \gamma) \in Attr_s^{\mathcal{C}(q)-1}(\mathcal{Q}_R)\},$$

which implies that there is a successor q' such that $\mathcal{C}(q') < \mathcal{C}(q)$.
3) If $q \in Q_Z^m$ satisfies $0 < \mathcal{C}(q) < \infty$, then every successor q' of q satisfies $\mathcal{C}(q') < \mathcal{C}(q)$. Specifically, we again have $q \in Attr_s^{\mathcal{C}(q)}(\mathcal{Q}_R) \setminus Attr_s^{\mathcal{C}(q)-1}(\mathcal{Q}_R) = CPre_s(Attr_s^{\mathcal{C}(q)-1}(\mathcal{Q}_R))$. Since $q \in Q_Z^m$, it follows that

$$q \in \{q_z \in Q_Z^m \mid \forall e \in \Gamma(q_z) : f_{zy}^m(q_z, e) \in Attr_s^{\mathcal{C}(q)-1}(\mathcal{Q}_R)\},$$

which indicates that every successor q' satisfies $\mathcal{C}(q') < \mathcal{C}(q)$.

If $0 < \mathcal{C}(q) < \infty$ for some $q \in Q_Y^m$ (resp. $q \in Q_Z^m$), then we define $\pi_s(q) = \gamma$ (resp. $\pi_s(q \xrightarrow{e} q_y) = \gamma$ for any $e \in \Gamma(q)$) such that $f_{yz}^m(q, \gamma) = q'$ (resp. $f_{yz}^m(f_{zy}^m(q, e), \gamma) = q'$) for some successor $q' \in Q_Z^m$ of q with $\mathcal{C}(q') < \mathcal{C}(q)$. If $\mathcal{C}(q) \in \{0, \infty\}$, we define q move to any of its arbitrary successors. Specifically, $\mathcal{C}(q) = 0$ means that $q \in \mathcal{Q}_R$, i.e., the supervisor has already won and can therefore move arbitrarily. On the other hand, $\mathcal{C}(q) = \infty$ means that $q \in (Q_Y^m \cup Q_Z^m) \setminus Attr_s(\mathcal{Q}_R)$. We note that states with $\mathcal{C}(q) = \infty$ are encountered only after visiting $\mathcal{Q}_R$ and arbitrarily moving, starting from $Attr_s(\mathcal{Q}_R)$. Hence, in this case, q can also move arbitrarily.

It remains to prove that π_s is a winning strategy. Let $r = q_0 \to q_1 \to q_2 \to \cdots \in Run(q, \pi_e)$ be an arbitrary run that is consistent with π_s, starting from $q \in Q_Y^m \cup Q_Z^m$. We aim to show that there exist an index k such that $q_k \in \mathcal{Q}_R$, by induction on $\mathcal{C}(q_0)$.

Base step: for $\mathcal{C}(q_0) = 0$, we know that $q_0 \in \mathcal{Q}_R \subseteq Attr_s(\mathcal{Q}_R)$ by property 1 stated above. Induction step: assume $0 < \mathcal{C}(q_0) < \infty$. By applying properties 2 and 3, we know that $\mathcal{C}(q_1) < \mathcal{C}(q_0)$, meaning $q_1 \in Attr_s(\mathcal{Q}_R)$. Furthermore, since r is positional, the subrun $q_1 \to q_2 \to \cdots$ is also consistent with π_s. Thus, the induction hypothesis is applicable to $q_1 \to q_2 \to \cdots$ and yields $q_k \in Attr_s(\mathcal{Q}_R)$. Hence, the position $k+1$ of r is in $\mathcal{Q}_R$, which concludes the inclusion step.

Thus, π_s is a positional winning strategy for the supervisor from each state in $Attr_s(\mathcal{Q}_R)$, i.e., $Attr_s(\mathcal{Q}_R) \subseteq \mathcal{W}_s(T_{max})$.

Let $\mathcal{W}_e(T_{max})$ denote the winning region for the environment, by arguments similar to the above, we have $(Q_Y^m \cup Q_Z^m) \setminus Attr_s(\mathcal{Q}_R) \subseteq \mathcal{W}_e(T_{max})$. By combining with $\mathcal{W}_s(T_{max}) \cap \mathcal{W}_e(T_{max}) = \emptyset$, we conclude that $\mathcal{W}_s(T_{max}) = Attr_s(\mathcal{Q}_R)$. □

Theorem 1 *The solution to Problem 1 is yes when* $\mathcal{W}_s(T_{max}) \neq \emptyset$.

Proof By definition, any state in the winning region $\mathcal{W}_s(T_{max})$ is reachable from the initial state q_y^0 in the DBTS T_{max}. Therefore, there exist initial runs that are capable of reaching $\mathcal{W}_s(T_{max})$. Since any such initial run can induce a winning strategy for the supervisor, which corresponds to a feasible robust supervisor satisfying the given scLTL formula under disturbances, the proof is thereby completed. □

5.2 Optimal robust supervisor synthesis

In this subsection, we will formulate a game on the ALS, where the supervisor aims to minimize the distance to the accepting states, while the disturbance strives in the opposite direction. Then we introduce a sequence of vectors and develop a dynamic programming based approach to search for the optimal winning strategy of the supervisor, which is then mapped to a supervisor solving Problem 2.

We define a sequence of vectors $V^i \in (\mathbb{R}_0^+)^{|Q_Y^m \cup Q_Z^m| \times 1}$ for $i \in \mathbb{N}_0^+$, where i represents the iteration number. Each element of V^i is a scalar corresponding to a state in T_{max} at iteration i. This sequence is used to calculate the minimum distance for each state in the ALS $T_{\max}$ to reach the target states after a finite number of actions. Initially, we let $V^0(q_y) = \mathcal{D}(q_y) = d_\otimes(\mathcal{S}(q_y), F_\otimes)$ for all $q_y \in Q_Y^m$, and $V^0(q_z) = \mathcal{D}(\mathcal{P}(q_z)) = d_\otimes(\mathcal{S}(\mathcal{P}(q_z)), F_\otimes)$ for all $q_z \in Q_Z^m$. Next, for all $q_y \in Q_Y^m, q_z \in Q_Z^m$, and $i \geq 0$, we have the following equations:

$$V^{i+1}(q_y) = \min\{V^i(q_y), \min_{q_z \in Post(q_y)} \max_{\tilde{q}_y \in Post(q_z)} V^i(\tilde{q}_y)\}; \tag{4}$$

$$V^{i+1}(q_z) = \max_{\tilde{q}_y \in Post(q_z)} V^i(\tilde{q}_y). \tag{5}$$

The above equations immediately imply that $\forall q_y \in Q_Y^m$:

$$V^1(q_y) = \min\{V^0(q_y), \min_{q_z \in Post(q_y)} \max_{\tilde{q}_y \in Post(q_z)} V^0(\tilde{q}_y)\}. \tag{6}$$

That is, $V^1(q_y)$ encodes the minimum distance to reach the accepting states $F_\otimes$ through at most one step of action from the current state *Y*-state q_y. Note that the distance $V^i(q_y)$ will be updated in the next iteration if a smaller value is obtained when some actions are taken. Globally, the supervisor aims to minimize the distance to reach the accepting states, while the disturbances aim to maximize it. Iterating the sequence of vectors eventually leads to the *fixed-point* (vector), which is denoted by V^* and defined as: for all $q_y \in Q_Y^m$ and $q_z \in Q_Z^m$,

$$V^*(q_y) = \min_{r \in Run(T, q_y)} \max_{\tilde{q}_y \in Post(Last_Z(r))} \mathcal{D}(\tilde{q}_y); \tag{7}$$

$$V^*(q_z) = \max_{\tilde{q}_y \in Post(q_z)} V^*(\tilde{q}_y). \tag{8}$$

The fixed-point value of each state in the ALS also indicates the best achievable robustness bound of any control strategy starting from that state. The supervisor has to consider the *worst* deviation caused by disturbances reflected by the *max* operator in the above equations. In other words, the optimal robustness of the supervisor is achievable under all possible adversaries. Especially, the robustness measure of any control strategy at a *Y*-state q_y can not exceed $\mathcal{D}(q_y)$, since the supervisor is able to secure a distance (from the accepting states) of at most $\mathcal{D}(q_y)$ by simply staying at the current state. Since the state space of T_{max} is finite, the fixed-point is guaranteed to be calculated after a finite number of iterations. Then we present a dynamic programming-based approach to compute the fixed-point as described in Algorithm 2, which is inspired by Bellman-Ford shortest path algorithm (Cormen et al. n.d.). Note that the elements in the fixed-point vector V^* are all identical by calculation. The convergence of Algorithm 2 is formally discussed in the following lemma.

Algorithm 2 Compute the fixed-point.

Input: G, φ, d, δ
Output: $T_{max} = (Q_Y^m, Q_Z^m, E, \Gamma, f_{yz}^m, f_{zy}^m, q_y^0)$
1 Build DFA A_φ to represent φ and take the product $G_\otimes = G \times A_\varphi$;
2 $Q_Y^m = \{q_y^0\} = \{(x_\otimes^0, d(x_\otimes^0, F_\otimes))\}$, $Q_Z^m = \emptyset$;
3 $T_{max}^{pre} = DoDFS(q_y^0, G_\otimes, d_\otimes, \delta_\otimes)$;
4 **while** *there exist Y-states without successors* **do**
5 └ Remove such states and their predecessor Z-states;
6 Take the accessible part of the remaining structure and return T_{max};
Procedure: $DoDFS(q_y, G_\otimes, d_\otimes, \delta_\otimes)$
7 **for** $\gamma \in \Gamma$ **do**
8 $q_z = f_{yz}(q_y, \gamma)$ by Definition 7;
9 **if** q_z *is deadlock-free* **then**
10 Add transition $q_y \xrightarrow{\gamma} q_z$ to f_{yz}^m;
11 **if** $q_z \notin Q^m$ **then**

Lemma 5 *The fixed-point calculation in Algorithm 2 converges in finite step.*

Proof Given Eq. 5, the convergence of $V^i(q_z)$ is readily apparent. Since the iterative update operation of $V(\cdot)$ is in real numbers $\mathbb{R}$, the Monotone Convergence Theorem can be applied. To demonstrate the convergence of Eq. 4, specifically $V^i(q_y)$, it is sufficient to establish its boundedness and monotonicity. We first show its boundedness by induction.

Base Step: By Line 2 of Algorithm 2, we initialize the vector as $V^0(q_y) = \mathcal{D}(q_y)$. Thus, it follows that $\min \mathcal{D}(q_y) \leq V^0(q_y) \leq \max \mathcal{D}(q_y)$. From Eq. 5, we obtain $V^1(q_z) = \max_{\tilde{q}_y \in Post(q_z)} V^0(\tilde{q}_y)$, which implies that $\min \mathcal{D}(q_y) \leq V^1(q_z) \leq \max \mathcal{D}(q_y)$.

Induction Step: Suppose $\min \mathcal{D}(q_y) \leq V^i(q_y) \leq \max \mathcal{D}(q_y)$ and $\min \mathcal{D}(q_y) \leq V^i(q_z) \leq \max \mathcal{D}(q_y)$ hold. By substituting Eq. 5, we rewrite Eq. 4 as

$$V^{i+1}(q_y) = \min\{V^i(q_y), \min_{q_z \in Post(q_y)} V^{i+1}(q_z)\},$$

where $V^i(q_y)$ is bounded by induction hypothesis and $V^{i+1}(q_z)$ is bounded by Eq. 5. Thus, $V^{i+1}(q_y)$ is also bounded.

We then show the update rule in Eq. 4 is monotonically decreasing since $V^{i+1}(q_y) = \min\{V^i(q_y), \min_{q_z \in Post(q_y)} \max_{\tilde{q}_y \in Post(q_z)} V^i(\tilde{q}_y)\} \leq V^i(q_y)$. Thus, the convergence of $V^i(q_y)$ follows from its boundedness and monotonicity. Additionally, $V^i(q_z)$ converges due to the convergence of $V^i(q_y)$. □

Based on the above results, we present Algorithm 3 to synthesize optimal robust supervisors by tracking the fixed-point values and control strategies from T_{max}. The procedure *Expand* iteratively adds new states to T_{opt}, which share the same fixed-point value with the initial state q_y^0. This expansion continues until no additional states are available and T_{opt} encodes a control strategy guaranteed to reach inflated accepting states $F_\otimes^\sigma$ where $\sigma = V^*(q_y^0)$. Eventually, the algorithm returns an optimal robust supervisor with $\sigma_{min} = V^*(q_y^0)$ and the following theorem ensures its correctness.

Algorithm 3 Synthesize an optimal robust supervisor.

Input: T_{max}
Output: V^*
for $q_y \in Q_Y^m$ **do**
  $V(q_y) \leftarrow \mathcal{D}(q_y)$;
for $q_y \in Q_Y^m$ **do**
  for $q_z \in Post(q_y)$ **do**
    $V(q_z) \leftarrow \max_{\tilde{q}_y \in Post(q_z)} V(\tilde{q}_y)$;
    if $\min_{q_z \in Post(q_y)} \max_{\tilde{q}_y \in Post(q_z)} V(\tilde{q}_y) < V(q_y)$ **then**
      $V(q_y) \leftarrow \min_{q_z \in Post(q_y)} \max_{\tilde{q}_y \in Post(q_z)} V(\tilde{q}_y)$;
for $q \in Q_Y^m \cup Q_Z^m$ **do**
  $V^*(q) \leftarrow V(q)$;
return V^*;

Theorem 2 *The supervisor synthesized by Algorithm 3 is live and $V^*(q_y^0)$-robust.*

Proof The liveness holds by Lemma 1. We prove the $V^*(q_y^0)$-robustness by contradiction. Let $\sigma_{min} = V^*(q_y^0)$. Suppose there exist a σ'-robust supervisor with $\sigma' < \sigma_{min}$. Then, there necessarily exist a run $r \in Run(T, q_y)$ and a state $\tilde{q}'_y \in r$, starting from q_y^0, such that $\tilde{q}'_y \in F_{\otimes}^{\sigma'}$, which leads to a contradiction with Eq. 7. □

Remark 2 Note that the metric function d is defined *a priori*, depending on the specific dynamics of the system in the practical applications. From the definition of the disturbed transition function of the product system $G_{\otimes}^{\delta}$ (Eq. 1), the definition of DBTS (Definition 7), Algorithm 1, and the results in Section 5.1, it follows that the choice of d directly influences the structure of $G_{\otimes}^{\delta}$, and consequently, the structure of DBTS/ALS, the supervisor's winning region over ALS, and the existence of robust supervisors. Furthermore, according to the results in Section 5.2, the minimal robustness is also sensitive to the selected metric d.

5.3 Complexity analysis

In this section, we provide a detailed analysis of the computational complexity of our framework of robust supervisory control, which consists of three main procedures: the construction of the ALS T_{max}, the verification of the existence of a robust supervisor S, and the synthesis of an optimal controller S_{opt}.

The construction of T_{max} is presented in Algorithm 1, which involves two key procedures: $DoDFS$ and the pruning of redundant states. The $DoDFS$ procedure, executed from Line 7 to Line 18, constructs a DBTS. In the worst-case scenario, this DBTS contains $|X_{\otimes}| + |X_{\otimes}| \cdot |\Gamma| = |X||X_A| + |X||X_A|2^{|\Sigma_c|}$ states, where $|X|$ and $|X_A|$ are the state spaces of the plant G and the DFA A_{φ} translated from the scLTL formula, respectively. This follows directly from Definition 7 on the DBTS. The pruning procedure, as detailed in

Lines 4–6, reduces this state space by eliminating redundant states, which has its complexity quadratic in the size of the DBTS.

The existence of a robust supervisor is determined by computing the winning region through the reachability game outlined in Section 5.1. This procedure has a complexity that is linear with respect to the size of T_{max} Apt and Grädel (2011).

Lastly, before synthesizing the optimal robust supervisor, the fixed-point vector V^* is computed using Algorithm 2. This algorithm is a variant of the Bellman-Ford shortest path algorithm. The complexity of this computation is determined by the product of the number of states and transitions in T_{max} (Cormen et al. n.d.). Finally, Algorithm 3 is employed to synthesize the optimal robust supervisor. This algorithm operates linearly with respect to the size of the ALS, which, in the worst case, shares the same state space as the DBTS.

The most computationally intensive procedure of our synthesis algorithm is the calculation of the fixed-point vector V^* in Algorithm 2. Consequently, the overall complexity of our synthesis procedure is

$$\begin{aligned} & O((|Q_Y^m| \times |\Gamma| + |Q_Z^m| \times |E|) \times (|Q_Y^m| + |Q_Z^m|)) \\ & = O\big((|X||X_A|2^{|\Sigma_c|} + |X||X_A|2^{|\Sigma_c|} \cdot |\Sigma|) \\ & \times (|X||X_A| + |X||X_A|2^{|\Sigma_c|})\big) \\ & = O(|X|^2|X_A|^2|\Sigma|2^{2|\Sigma_c|}) \end{aligned}$$

, that is the product of the number of states and the number of transitions in the ALS T_{max}.

It was shown in Hu et al. (2020) that synthesizing a supervisor that ensures the supervised system remains live is NP-hard. Therefore, this exponential complexity appears to be unavoidable. However, in the practical systems, it is often the case that $|\Sigma_c| \ll |X|$, which can mitigate the computational burden in real-world applications.

6 Illustrative example

In this section, we demonstrate the application of our robust supervisory control framework by a robot task planning example. A single robot is deployed to navigate an environment comprising eight rooms to accomplish two distinct tasks.

An office environment is modeled by a DES G shown in Fig. 3, with rooms (state space) $X = \{q_1, q_2, q_3, q_4, q_5, q_6, q_c, q_m\}$, where q_c and q_m are coffee room and mail room, respectively. Then the labeling function L assigns labels as follows: $L(q_1) = \cdots = L(q_6) = \emptyset$, indicating that these states have no specific task. The state q_c is labeled as "coffee", i.e., $L(q_c) = \{\text{coffee}\}$, denoting the capability of the robot to perform the coffee-making task, and the state q_m is labeled as "mail", i.e., $L(q_m) = \{\text{mail}\}$, indicating the robot's ability to execute the mail-sending task.

The event set of the system is partitioned as$E = E_c \cup E_{uc}$ where $E_c = \{a, b, c\}$ is the set of controllable events and $E_{uc} = \{u\}$ is the set of uncontrollable events. The metric function d, which is detailed in Table 2, quantifies the distance associated with transitions between states (rooms). Additionally, the system is subject to disturbances, characterized by the disturbance function δ. Specifically, the disturbances are $\delta(q_c) = 3$, $\delta(q_3) = 2$, and $\delta(q_1) = \delta(q_2) = \delta(q_4) = \delta(q_5) = \delta(q_6) = \delta(q_m) = 1$.

The robot initiates its operation in the state q_1, with the task of sending mail and making coffee. To avoid mishaps, such as spilling coffee on the mail, the robot is required to send the mail before making the coffee. This sequential requirement is encapsulated in a control specification formally expressed through an scLTL formula:

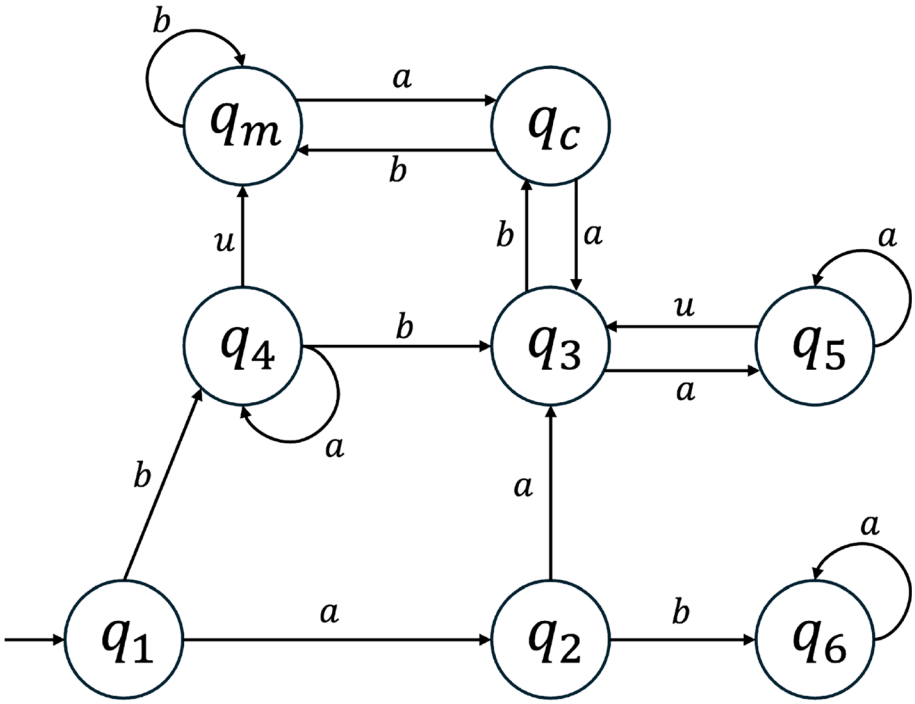

Fig. 3 DES G

Table 2 Distance between states of G in Fig. 3

	q_1	q_2	q_3	q_4	q_5	q_6	q_m	q_c
q_1	0	3	4	2	5	4	3	5
q_2		0	1	3	2	2	4	2
q_3			0	2	1	3	3	1
q_4				0	3	5	1	3
q_5					0	4	4	2
q_6						0	6	4
q_m							0	2
q_c								0

$$\varphi = \neg\text{coffee}\ \mathcal{U}\ \text{mail} \wedge \Diamond\text{coffee} \wedge \Diamond\text{mail}. \tag{9}$$

Next, we convert φ into a DFA A_ϕ using the toolbox LTLf2DFA Fuggitti (2019); De Giacomo et al. (2020). A_ϕ is shown in Fig. 4 and we subsequently build the product $G_\otimes$ in Fig. 5, which has 11 states. The accepting set is $F_\otimes = \{(q_c, 3), (q_m, 3), (q_3, 3), (q_5, 3)\}$ as 3 is the accepting state of A_φ and we only consider accessible states.

Then we will address both Problems 1 and 2 under the disturbances mentioned above. The disturbed product system $G_\otimes^\delta$ is depicted in Fig. 6, where perturbed transitions are highlighted in red. Note that $G_\otimes^\delta$ has a reduced state space compared with $G_\otimes$ since the disturbances render some states unreachable. For example, the original transition from state $(q_m, 2)$ to $(q_c, 3)$, represented as $(q_m, 2) \xrightarrow{a} (q_c, 3)$, is interrupted because $a \in E_c$

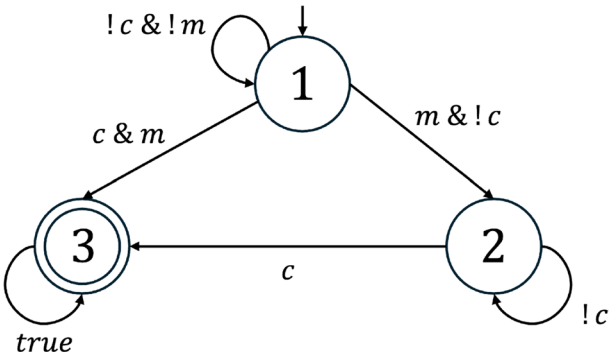

Fig. 4 DFA A_φ, where letter c stands for "coffee" and letter m stands for "mail"

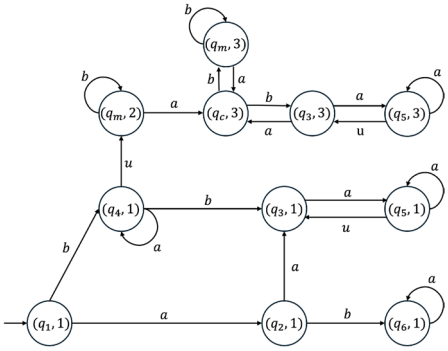

Fig. 5 Product system $G_\otimes$

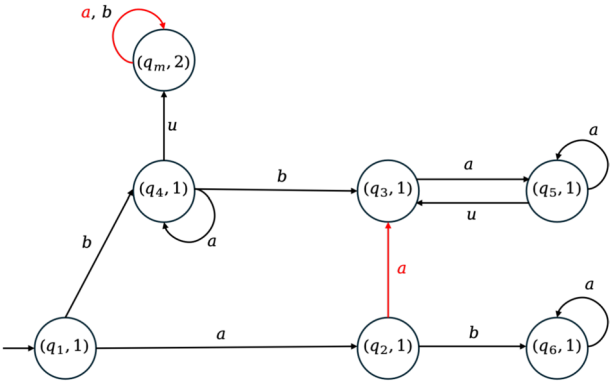

Fig. 6 Disturbed product system $G^\delta_\otimes$. The disturbed transitions

and the distance $d(q_m, q_c) = 2$ is less than the disturbance effect $\delta(q_c) = 3$. Consequently, this transition is modified to $(q_m, 2) \xrightarrow{a} (q_m, 2)$ and the original successor states $(q_c, 3), (q_m, 2), (q_3, 3), (q_5, 3)$ become no longer reachable. The transition $(q_2, 1) \xrightarrow{a} (q_3, 1)$ is also affected since $d(q_2, q_3) = 1 < \delta(q_3) = 2$. However, the disturbed transition still leads to $(q_3, 1)$ since q_3 is the unique element within the set $R^\delta_{G_\otimes}((q_2, 1))$, given that $d(q_6, q_3) = \delta(q_3)$. Given $\sigma = 2$ and by Definition 5, the set of inflated accepting states is $F^\sigma_\otimes = F_\otimes \cup \{(q_2, 1), (q_3, 1), (q_4, 1), (q_5, 1), (q_m, 2)\}$, which quantifies the potential deviation from the original accepting states of $G_\otimes$.

Then we construct the ALS T_{max} illustrated in Fig. 7, where the rectangular states represent *Y*-states, i.e., supervisor's places for actions and the ellipsoid states represent *Z*-states, i.e., the environment's places to play. According to Definition 7, each *Y*-state is represented as $q_y = (\mathcal{S}(q_y), \mathcal{D}(q_y))$ and $\mathcal{D}(q_y)$ is omitted from the Fig. for simplicity. For a similar reason, the distance component $\mathcal{D}(\mathcal{P}(q_z))$ is also not shown in each *Z*-state q_z. We do not label transitions from *Y*-states to *Z*-states since the labeling, i.e., set of enabled events by the supervisor, is simply retrieved from the second component of the source *Z*-state. Transitions from *Z*-states to *Y*-states are labeled by the enabled events that are allowed to occur. The set of target states $\mathcal{Q}_R$, i.e., the *Y*-states with left components in the σ-inflated accepted states, where $\sigma = 2$, are marked with double-rectangles.

We first solve Problem 1 following the method outlined in Subsection 5.1. Initially, the attractor $Attr_s^0(\mathcal{Q}_R)$ consists of four double-rectangular σ-inflated states. The iterative computation using Eq. 3 eventually returns the winning region $\mathcal{W}_s(T_{max})$ of the supervisor, which is shaded by a grey area in Fig. 7. Since the initial state of T_{max} is preserved in $\mathcal{W}_s(T_{max})$, we conclude that there exist 2-robust supervisors for the system (*G*, *d*), task φ and disturbances δ, according to Theorem 1.

Then we proceed to address Problem 2 by implementing the algorithms in Subsection 5.2. The initial values of the operators $V^0(q_y)$ for each $q_y \in Q_Y^m$ are assigned through Line 2 of Algorithm 2. The remaining lines of the algorithm compute the fixed-point values of the operator for each *Y*-state, which are appended to the right of each state of T_{max} in Fig. 7. The value at the initial state is 1, which indicates that the optimal supervisor synthesized from T_{max} is 0-robust, i.e., $\sigma_{min} = 0$. Note that a 0-robust supervisor is a supervisor synthesized under disturbances whose minimal robustness value equals 0. This indicates that, under disturbance, the supervised system can reach a state at zero distance from the nominal target set—i.e., it

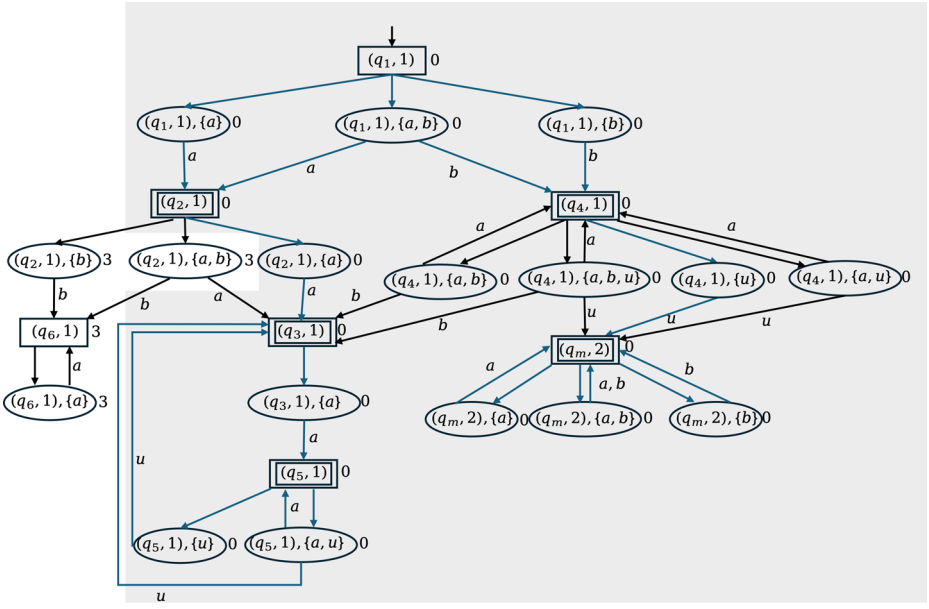

Fig. 7 ALS T_{max}: grey shaded area is the winning region of the supervisor when $\sigma = 2$; *Y*-states q_y with $\mathcal{S}(q_y) \in F_{\otimes}^{\sigma}$ are marked using double-rectangle; values on the right of the states are fixed-point values

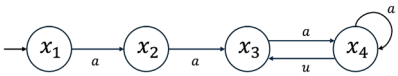

Fig. 8 An optimal robust supervisor $S/G^{\delta}_{\otimes}$ synthesized from Algorithm 3

can approach the original target states as closely as the metric allows. However, the system may still fail to reach the exact nominal target state itself, meaning that a 0-robust supervisor can still behave differently from the supervisor synthesized in the absence of disturbances. Next, we run Algorithm 3 to obtain the control strategy that adheres to the fixed-point values and the specific actions at each state are marked by blue lines in Fig. 7. Subsequently, the optimal robust supervisor S is built from T_{opt} and depicted in Fig. 8, with a renamed state space.

7 Conclusion

This study presents a robust supervisory control framework for metric discrete event systems (DES), which aims at achieving syntactically co-safe Linear Temporal Logic (scLTL) specifications in the presence of environmental disturbances. We formally model those disturbances and define the robustness of supervisors in a topological context. Two central problems are formulated and then addressed using two-player game-theoretic approaches: verification of the existence of robust supervisors and synthesis of optimal robust supervisors. To facilitate the analysis, we introduce disturbance bipartite transition system (DBTS), which serves as the game arena where the supervisor interacts with the environment. A specialized DBTS, termed the all live structure (ALS), is constructed under specific requirements. For the verification problem, we formulate and solve a reachability game on the ALS to ascertain the existence of robust supervisors. In the synthesis phase, we develop a dynamic programming method on the ALS to compute the optimal robustness value for each state. By tracking the actions that yield these optimal values, we derive the optimal winning strategy for the supervisor, which ultimately results in the optimal supervisor. To demonstrate the practical applicability and effectiveness of our proposed framework, we provide a running example of robot task planning.

Looking ahead, we plan to extend this work to address robust supervisory control under partial observation and more general control specifications expressed by full LTL or other temporal logic formulas. For the extension to partial observation, the main challenge lies in appropriately defining the states of DBTS/ALS in a powerset form and refining the corresponding fixed-point computation algorithm. As for general temporal logic specifications beyond reachability objectives, such as full LTL, the reachability game formulation used in this paper is no longer applicable. Hence, developing suitable algorithmic game formulations and synthesis procedures for such specifications presents another important challenge.

Acknowledgements This work is partially supported by National Natural Science Foundation of China grants 62303389, 62373289, 62573291, 62173226; Guangdong Basic and Applied Basic Research Funding grant 2024A1515012586; Guangdong Scientific Research Platform and Project Scheme grant 2024KTSCX039; Guangzhou-HKUST(GZ) Joint Funding Program grants 2023A03J0678, 2023A03J0011; Youth Talent Support Program of Guangdong Association for Science and Technology grant SKXRC2025463; together with Science Center Program of National Natural Science Foundation of China under Grant 62188101.

Author Contributions Y. J. and X.Y developed the initial idea of the work and outlined the directions for extension. P. L and S. M proposed most of the technical methods and finished the relevant contents. All authors contributed to the writings and reviewed the manuscript.

Data Availability No datasets were generated or analysed during the current study.

Declarations

Competing Interests The authors declare no competing interests.

References

Alves MV, da Cunha AE, Carvalho LK et al (2021) Robust supervisory control of discrete event systems against intermittent loss of observations. Int J Control 94(7):2008–2020

Apt KR, Grädel E (2011) Lectures in game theory for computer scientists. Cambridge University Press

Baier C, Katoen JP (2008) Principles of model checking. MIT press

Belta C, Yordanov B, Gol EA (2017) Formal methods for discrete-time dynamical systems, vol 89. Springer

Cassandras CG, Lafortune S (2021) Introduction to discrete event systems. Springer

Cormen TH, Leiserson CE, Rivest RL, et al. (2022) Introduction to algorithms. MIT press

De Giacomo G, Di Stasio A, Fuggitti F, et al. (2020) Pure-past linear temporal and dynamic logic on finite traces. In: 29th International Joint Conference on Artificial Intelligence, pp 4959–4965

Deng W, Qiu D, Yang J (2021) Intersection-based decentralized supervisory control of probabilistic discrete event systems. IEEE Trans Autom Control 66(12):6171–6178

Fokkink W, Goorden M (2024) Offline supervisory control synthesis: taxonomy and recent developments. Discrete Event Dynamic Systems pp 1–53

Fritz R, Zhang P (2023) Detection and localization of stealthy cyberattacks in cyber-physical discrete event systems. IEEE Trans Autom Control 68(12):7895–7902

Fuggitti F (2019) LTLf2DFA. DOI: https://doi.org/10.5281/zenodo.3888410

Girard A, Eqtami A (2021) Least-violating symbolic controller synthesis for safety, reachability and attractivity specifications. Automatica 127:109543

Hu Y, Ma Z, Li Z (2020) Design of supervisors for active diagnosis in discrete event systems. IEEE Trans Autom Control 65(12):5159–5172

Ji Y, Yin X (2024) Robust control of metric discrete event systems against bounded disturbances. In: proceedings of 17th IFAC International Workshop on Discrete Event Systems, pp 108–113

Ji Y, Yin X, Lafortune S (2021) Optimal supervisory control with mean payoff objectives and under partial observation. Automatica 123:109359

Ji Y, Yin X, Lafortune S (2022) Local mean payoff supervisory control for discrete event systems. IEEE Trans Autom Control 67(5):2282–2297

Jiang S, Kumar R (2006) Supervisory control of discrete event systems with CTL* temporal logic specifications. SIAM J Control Optim 44(6):2079–2103

Komenda J, Masopust T (2023) Hierarchical supervisory control under partial observation: Normality. IEEE Trans Autom Control 68(12):7286–7298

Kupferman O, Vardi M (2001) Model checking of safety properties. Formal methods in system design 19:291–314

Lacerda B, Lima PU (2012) Designing petri net supervisors from LTL specifications. Robotics: Science and Systems VII p 169

Luo Y, Miao S, Yu X et al (2025) Supervisory control of weighted automata with control delays. IEEE Control Systems Letters 9:1952–1957

Lv P, Xu Z, Ji Y et al (2024) Optimal supervisory control of discrete event systems for cyclic tasks. Automatica 164:111634

Ma Z, Cai K (2021) Optimal secret protections in discrete-event systems. IEEE Trans Autom Control 67(6):2816–2828

Majumdar R, Render E, Tabuada P (2013) A theory of robust omega-regular software synthesis. ACM Trans on Embedded Computing Systems 13(3):1–27

Malik R, Mohajerani S, Fabian M (2023) A survey on compositional algorithms for verification and synthesis in supervisory control. Discrete Event Dynamic Systems Theory and Applications 33(3):279–340

Meira-Góes R, Kang E, Lafortune S et al (2023) On tolerance of discrete systems with respect to transition perturbations. Discrete Event Dynamic Systems Theory and Applications 33:395–424

Miao S, Cui B, Ji Y, et al. (2025a) Protect your knowledge: Epistemic property enforcement of discrete event systems with asymmetric information. IEEE Control Systems Letters 9:1832–1837

Miao S, Komenda J, Lai A (2025b) Active diagnosis of time-interval automata: Time perspectives. IEEE Transactions on Automation Science and Engineering 22:11239–11249

Miao S, Komenda J, Masopust T, et al. (2025c) Enforcement of critical observability in modular discrete-event systems. IEEE Transactions on Automatic Control 10.1109/TAC.2025.3622478
Miao S, Komenda J, Lin F (2026) Hierarchical supervisory control of networked and cyber-attacked discrete-event systems. Automatica 183:112578
Reveliotis S, Fei Z (2016) Invariant-based supervisory control of switched discrete event systems. IEEE Trans Autom Control 62(2):921–927
Ritsuka K, Rudie K (2024) A uniform approach to compare architectures in decentralized discrete/event systems. Automatica 165:111683
Sakakibara A, Ushio T (2020) On-line permissive supervisory control of discrete event systems for scltl specifications. IEEE Control Systems Letters 4(3):530–535
Sakakibara A, Urabe N, Ushio T (2021) Finite-memory supervisory control of discrete event systems for LTL $[\mathcal{F}]$ specifications. IEEE Trans Autom Control 67(12):6896–6903
Sampath M, Sengupta R, Lafortune S et al (1995) Diagnosability of discrete-event systems. IEEE Trans Autom Control 40(9):1555–1575
Seow KT (2020) Supervisory control of fair discrete-event systems: A canonical temporal logic foundation. IEEE Trans Autom Control 66(11):5269–5282
Sontag ED (1998) Mathematical control theory: deterministic finite dimensional systems. Springer
Tai R, Lin L, Zhu Y et al (2022) Synthesis of the supremal covert attacker against unknown supervisors by using observations. IEEE Trans Autom Control 68(6):3453–3468
Takai S (2021) Synthesis of maximally permissive supervisors for nondeterministic discrete event systems with nondeterministic specifications. IEEE Trans Autom Control 66(7):3197–3204
Tellex S, Gopalan N, Kress-Gazit H et al (2020) Robots that use language. Annual Review of Control Robotics and Autonomous Systems 3(1):25–55
Wang X, Hu H, Zhou M (2020) Discrete event approach to robust control in automated manufacturing systems. IEEE Transactions on Systems Man and Cybernetics Systems 52(1):123–135
Wonham WM, Cai K (2019) Supervisory control of discrete-event systems. Springer
Yin X, Lafortune S (2016) A uniform approach for synthesizing property-enforcing supervisors for partially-observed discrete-event systems. IEEE Trans Autom Control 61(8):2140–2154
You D, Wang S, Zhou M et al (2021) Supervisory control of petri nets in the presence of replacement attacks. IEEE Trans Autom Control 67(3):1466–1473
Yu X, Dong W, Li S et al (2024) Model predictive monitoring of dynamical systems for signal temporal logic specifications. Automatica 160:111445
Zhang H, Feng L, Li Z (2018) A learning-based synthesis approach to the supremal nonblocking supervisor of discrete-event systems. IEEE Trans Autom Control 63(10):3345–3360
Zheng S, Shu S, Lin F (2023) Modeling and control of discrete event systems under joint sensor-actuator cyber attacks. IEEE Transactions on Control of Network Systems 11(2):782–794

Peiran Liu received the B.S. degree in Mathematics from the University of California, San Diego (UC San Diego), La Jolla, CA, USA, and the M.S. degree in Electrical and Computer Engineering from UC San Diego, where he was a member of the Existential Robotics Laboratory. He is currently pursuing the Ph.D. degree in the Robotics and Autonomous Systems Thrust at the Hong Kong University of Science and Technology (Guangzhou), Guangzhou, China. His research interests include robotics, reinforcement learning, and autonomous systems.

Shaowen Miao was born in Fujian, China, in 2000. He received the B.Eng. degree in automation from Fuzhou University, Fuzhou, China, in 2022, and the M.Eng. degree in control engineering from Xiamen University, Xiamen, China, in 2025. He is currently pursuing the Ph.D. degree in robotics and autonomous systems at the Systems Hub, The Hong Kong University of Science and Technology (Guangzhou), Guangzhou, China. From April 2024 to June 2024, he was a visiting student with the Institute of Mathematics, Czech Academy of Sciences. He is also currently a visiting student with the School of Automation and Intelligent Sensing at Shanghai Jiao Tong University. His current research interests include formal methods and machine learning, with applications to cyber-physical systems and robotics.

Yiding Ji Yiding Ji earned his Bachelor's degree of Electrical Engineering and Automation from Tianjin University, China, in 2014. He subsequently obtained both a Master's degree (2016) and a Ph.D. degree (2019) of Electrical and Computer Engineering from the University of Michigan, USA. From 2019 to 2021, he served as a postdoctoral researcher at Boston University, USA, then worked as a research scientist at Siemens Corporation, USA. Since 2022, he has been with the Hong Kong University of Science and Technology (Guangzhou), China, where he is now an assistant professor in Robotics and Autonomous Systems Thrust of Systems Hub. He is also affiliated with the Department of Electronic and Computer Engineering at the Hong Kong University of Science and Technology. His research interests span control systems, formal methods, discrete event systems, game theory, machine learning, intelligent robots and autonomous systems. He is a member of the Institute of Electrical and Electronics Engineers (IEEE), where he participates in the IEEE Control Systems Society Technical Community on Discrete Event Systems and the IEEE Control Systems Society Conference Editorial Board Committee. Additionally, he serves as a reviewer and associate editor for multiple leading academic journals and conferences.

Xiang Yin Xiang Yin is a Full Professor in the School of Automation and Intelligent Sensing, Shanghai Jiao Tong University. He received the B.Eng degree from Zhejiang University, China in 2012, the M.S. degree from the University of Michigan, Ann Arbor, USA, in 2013, and the Ph.D degree from the University of Michigan, Ann Arbor, USA, in 2017, all in electrical engineering. Since 2017, he has been with Shanghai Jiao Tong University, China. His research interests include formal methods, discrete-event systems, robotics, artificial intelligence and cyber-physical systems. Dr. Yin is serving as the chair of the IEEE CSS Technical Committee on Discrete Event Systems, Associate Editors for the Journal of Discrete Event Dynamic Systems: Theory & Applications, Nonlinear Analysis: Hybrid Systems, IEEE Control Systems Letters, IEEE Transactions on Automation Science and Engineering, and a member of the IEEE CSS Conference Editorial Board.

Authors and Affiliations

Peiran Liu[1] · Shaowen Miao[1] · Yiding Ji[1,2] · Xiang Yin[3]

✉ Yiding Ji
jiyiding@hkust-gz.edu.cn

Peiran Liu
pliu868@connect.hkust-gz.edu.cn

Shaowen Miao
smiao585@connect.hkust-gz.edu.cn

Xiang Yin
yinxiang@sjtu.edu.cn

[1] Robotics and Autonomous Systems Thrust, The Hong Kong University of Science and Technology (Guangzhou), Guangzhou, China

[2] Department of Electronic and Computer Engineering, The Hong Kong University of Science and Technology, Hong Kong, China

[3] School of Automation and Intelligent Sensing, Shanghai Jiao Tong University, Shanghai, China