Binary Combinatorial Optimization-based Path Planning and Optimal Transition Control in Piecewise Linear Neural Abstraction Domain

Yousef Farid[a], Raphaël Jungers[a]

[a] *Department of Mathematical Engineering, ICTEAM, UCLouvain, Louvain-la-Neuve, Belgium*

Abstract

This paper addresses the construction of a finite-state neural abstraction (NA) for a broad class of unknown, uncertain nonlinear discrete-time systems. The framework relies on an ε-approximate alternating simulation relation (ASR), where the neural network (NN) approximation error provides a quantitative bound between the abstracted model and the real system. We first formalize a state-space partitioning (SSP) induced by a NN with an activation function of type rectified linear unit (ReLU), whereby the continuous state space is decomposed into polyhedral cells, and a discrete-time piecewise linear (PWL) dynamics model is associated with each input symbol. To efficiently compute transitions among cells, we introduce a binary combinatorial optimization (BCO) algorithm that finds the shortest paths between initial and target cells. A hybrid control law is then designed for optimal cell-to-cell transitions in PWL subsystems, combining tube model predictive control (MPC) with minimax-based feedback to ensure robustness. The result is a weighted automaton over polyhedral cells, where edge weights correspond to tube-MPC costs used within the BCO procedure. Establishing a feedback-refinement relation between the real unknown system and its data-driven finite abstraction model enables certified controller synthesis on the abstraction, with guaranteed refinement back to the real system. The proposed approach is validated on three benchmarks: autonomous vehicle path planning, a double-pendulum robotic system, and an inverted pendulum.

Keywords: Neural Abstraction, Binary combinatorial optimization, Tube-MPC, Minimax control.

1. Introduction

Control and path planning for cyber–physical systems (CPSs) require handling nonlinear dynamics, ensuring safety guarantees, and achieving real-time execution. Symbolic abstractions and formal methods have emerged as powerful frameworks to guarantee safety and correctness by constructing finite-state models that approximate the original continuous-state systems [1, 2]. In these approaches, controllers synthesized on the abstraction can be refined into hybrid feedback strategies for the original system when an alternating simulation relation (ASR) holds [3]. However, traditional abstraction techniques often rely on grid-based discretization of the state and input sets, leading to the curse of dimensionality and high computational complexity. Meanwhile, modern data-driven approaches can approximate unknown dynamics but frequently lack interpretability, struggle to offer formal guarantees, and impose heavy computational or memory demands. This gap motivates the development of structured neural abstractions (NA) and control architectures that are both theoretically sound and computationally efficient.

1.1. Related Works

To address computational challenges in complex control systems, finite abstractions have been proposed as approximate representations of continuous-space systems, where each finite state corresponds to a set of continuous states [4]. Compositional symbolic finite-state abstractions for temporal logic specifications in control networks were developed in [5], while [6] employed reinforcement learning tailored for non-deterministic finite transition systems in symbolic control. A data-driven gridding approach was introduced in [7] to optimize state-set discretization and minimize transitions during abstraction construction, reducing computational effort. Parallel to these efforts, data-driven controller synthesis has gained attention for monitoring and controlling complex systems, particularly when deriving explicit mathematical models is infeasible, such as in highly nonlinear systems or human-in-the-loop applications. [8] proposed data-driven control design with regular language specifications for systems defined as assemblies of input-state functions, while [9] introduced a learning-based method to construct symbolic abstractions by estimating unmodeled dynamics and defining an ε-approximate ASR. [10] developed a Gaussian process-based approach to safely learn unknown dynamics and build high-confidence abstractions via Markov decision processes. However, these methods often struggle with fully unknown nonlinear dynamics, as their performance heavily depends on accurate system knowledge.

An alternative approach for safety-critical systems is to employ a neural network (NN) to model the controller and learn the system's unknown dynamics [11, 12]. In [13], a convex relaxation-based algorithm was proposed to compute forward

Email addresses: y.farid.e.control@gmail.com (Yousef Farid), raphael.jungers@uclouvain.be (Raphaël Jungers)

reachable and back-projection sets for neural feedback loops. [14] introduced a method to construct a global-interval NA induced by a NN with rectified linear unit (ReLU) activation functions, achieving center-exact reconstruction. Safety verification of nonlinear dynamical systems using NA has been studied in [15], while [16] proposed a memory-efficient approach to reduce the high memory demands of such abstraction-based techniques. More recently, [17] examined the trade-offs among NA templates in terms of precision, synthesis time, and verification time via reachability analysis.

In symbolic control, a key challenge is finding optimal paths between origin and destination cells while avoiding obstacles and respecting motion constraints. Various path-planning algorithms have been proposed, including geometry-based methods (e.g., Voronoi diagrams [18]), graph-based algorithms (e.g., Dijkstra [19], A^* [20], RRT and its variants [21]), meta-heuristic approaches (e.g., genetic algorithms, simulated annealing [22]), artificial potential fields [23], and AI-based methods (e.g., recurrent NNs [24], Q-learning [25]). However, these approaches often struggle with abstracted or high-dimensional systems due to the exponential growth of state-space samples, graph complexity, solution suboptimality, and extensive training requirements in learning-based methods. Combinatorial optimization offers a robust framework for path planning, enhancing route calculation and decision-making in complex and dynamic environments [26]. Applications include UAV path planning with QoS constraints in 5G networks [27] and optimal switch selection in radial distribution systems considering cost and reliability [28], demonstrating its effectiveness in minimizing costs while ensuring system performance.

Many challenging control problems require both performance and safety guarantees in the presence of model uncertainty. Model predictive control (MPC), a widely used optimization-based strategy in industry [29], computes an optimal input sequence over a finite horizon by treating the current plant state as the initial condition of a nominal model. Tube-MPC, a robust variant, guarantees that all trajectories of the uncertain system remain within a bounded "tube" around the nominal trajectory by enforcing tightened constraints. This framework has been successfully applied to robust output-feedback control [30], linear parameter-varying and nonlinear systems [31], safe learning [32], and planning for robotic and autonomous systems [33]. For abstracted systems, incorporating a robust term within tube-MPC enhances disturbance rejection and ensures closed-loop stability despite modeling errors. Reinforcement learning has also been employed to compute optimal control policies under uncertainty [34], including feedback designs for discrete- and continuous-time systems as well as zero-sum dynamic game settings [35]. In particular, policy iteration methods have proven effective for solving linear–quadratic regulator problems [36].

1.2. Motivation and Representative Scenarios

Real-time control of systems with unknown nonlinear dynamics, strict safety constraints, and limited onboard computation remains a central challenge in robotics and CPSs. Learning-based methods offer expressive modeling power, but their opacity, limited certifiability, and computational burden hinder use in safety-critical domains [37]. Classical robust and adaptive control techniques provide formal guarantees, yet often assume structural model knowledge, smooth uncertainty bounds, and do not scale efficiently to high-dimensional data-driven settings [38]. This motivates control architectures that jointly learn system behavior from data, enforce formal safety guarantees, and remain computationally viable on embedded platforms. The control problem is governed by a critical trade-off among three factors:

1. uncertain nonlinear discrete-time dynamics arising from real-world data rather than exact analytical models,
2. safety-critical state and input constraints that must hold at all times,
3. real-time computational limits that prevent reliance on expensive online optimization or large NN architectures.

These challenges appear in practice. Two representative settings include:

(a) Urban autonomous driving: vehicle dynamics involve uncertainty from interaction forces and friction variability, while ensuring lane-keeping, collision avoidance, and actuator limits under tight timing constraints. Key issues: partial observability, bounded model mismatch, real-time planning, and guaranteed constraint satisfaction in dense traffic.

(b) Underactuated pendulum swing-up and stabilization: physical deviations from nominal models due to friction, elasticity, and backlash require robust regulation of an unstable system with actuator limits and fast control cycles. Key issues: underactuation, sensitivity to model error, energy regulation, and limited computation budget.

In both scenarios, the goal is not only accurate prediction, but certified safety under uncertainty, reliable operation despite model mismatch, and real-time feasibility. The proposed framework is designed to address this three-way tension through data-driven piecewise linear (PWL) abstraction, constraint-aware symbolic planning, and tube-based feedback control.

1.3. Contribution

Inspired by the preceding discussion and to the best of our knowledge, there is no comprehensive work on the construction of finite-state weighted automaton-based NNs for discrete-time nonlinear nonaffine control systems. This abstraction system inherently provides a structured modeling platform suitable for CPSs functioning within environments characterized by uncertainty or noise. Leveraging automated optimal controller synthesis methods can significantly enhance system development by improving reliability while reducing design effort, cost, and computational burden. The main contribution of this paper is listed below:

1. Proposing a NN-based approach for the construction of an abstraction model for uncertain nonlinear controlled systems;
2. Developing an algorithm to identify and label the essential polyhedral cells within the NN–induced partitioned state space;

3. Proposing an efficient binary combinatorial optimization (BCO) algorithm to determine the optimal path between initial and target cells;
4. Designing a hybrid robust optimal transition controller, combining tube-MPC with minimax feedback controller, to guarantee reachability between successive polyhedral cells given the optimal path on the weighted automaton;
5. The effectiveness of the proposed approach is illustrated by its implementation on a wheeled mobile robot, a double-pendulum robot, and an inverted pendulum.

1.4. Outline

The remainder of the paper is organized as follows. Section 2 presents the preliminaries and formally defines the problem. Section 3 describes the finite-state NA and its associated state-space partitioning (SSP) algorithm. Our path planning approach using the BCO algorithm is detailed in Section 4, followed by the robust optimal transition control procedure in Section 5. Section 6 provides numerical simulations on a double pendulum robot, path planning benchmark for a mobile robot, and an inverted pendulum. Finally, Section 7 concludes the paper with a summary and outlook.

2. Notations and Preliminaries

2.1. Notations on Polyhedra

Definition 1 (Polytope: $\mathcal{H}$- and $\mathcal{V}$-representations)**.** *A convex bounded polytope $\mathcal{P} \subset \mathbb{R}^n$ can be represented in two equivalent forms:*

a) ***$\mathcal{H}$-representation (half-space representation):***

$$\mathcal{P} = \{x \in \mathbb{R}^n : \mathcal{A}x \leq \boldsymbol{\mu}\} \tag{1}$$

where $\mathcal{A} = \{\boldsymbol{a}_1, ..., \boldsymbol{a}_n\} \in \mathbb{R}^{m\times n}$, $\boldsymbol{\mu} = \{\mu_1, ..., \mu_n\} \in \mathbb{R}^m$, and each pair $(\boldsymbol{a}_i^\top, \mu_i)$ defines a half-space constraint.

b) ***$\mathcal{V}$-representation (vertex representation):***

$$\mathcal{V} = \text{conv}\{\boldsymbol{\nu}_1, \ldots, \boldsymbol{\nu}_k\} \tag{2}$$

where $\boldsymbol{\nu}_1, \ldots, \boldsymbol{\nu}_k \in \mathbb{R}^n$ are the vertices of $\mathcal{P}$.

Lemma 1. *(Farkas' Lemma [39]). If there exists a vector $\zeta \geq 0$ such that $\zeta^T\mathcal{A} = 0^T$ and $\zeta^T b \geq 0$, then the convex polytope defined by* (1) *is nonempty.*

Introducing ϵ, Lemma 1 can be formulated in terms of the following linear program (LP):

$$\begin{aligned} \max \quad & \epsilon \\ \text{s.t.} \quad & \epsilon \geq 0, \;\; \zeta \geq 0 \\ & \mathcal{A}^T\zeta = 0 \\ & \boldsymbol{\mu}^T\zeta - \epsilon \leq 0 \end{aligned} \tag{3}$$

If LP (3) is infeasible, the original convex polytope is empty. If it is feasible and the solution satisfies $\epsilon^* > 0$, the convex polytope is nonempty. The case $\epsilon^* = 0$ is inconclusive.

Definition 2. *(Duplicate Constraints): A constraint $\boldsymbol{a}_i^T x \leq \mu_i$ is called* duplicate *if there exists another constraint with a lower index $\boldsymbol{a}_j^T x \leq \mu_j$, $j < i$, such that $\alpha_c[\boldsymbol{a}_i^T, \mu_i] = [\boldsymbol{a}_j^T, \mu_j]$, where α_c is a positive scalar.*

Definition 3. *(Redundant and Essential Constraints): A constraint is* redundant *when its elimination does not alter the feasible set. Conversely, an* essential *constraint is characterized by its non-redundancy, signifying that its removal results in a modification of the feasible set.*

In the Appendix A, we show that the essential constraints in the H-representation of a polytope can be identified by solving the linear programming problem given in (A.2).

Another key consideration is determining whether a hyperplane intersects a polytope. Let:

$$\mathcal{H} = \{x \mid \boldsymbol{\eta}^T x = \rho\}, \tag{4}$$

denote a hyperplane, where $\boldsymbol{\eta} \in \mathbb{R}^n$ is a nonzero normal vector and $\rho \in \mathbb{R}$ is a scalar constant. If, for every vertex $\boldsymbol{\nu}_i$ of the polytope,

$$\boldsymbol{\eta}^T\boldsymbol{\nu}_i - \rho \geq 0 \qquad \text{or} \qquad \boldsymbol{\eta}^T\boldsymbol{\nu}_i - \rho \leq 0, \tag{5}$$

then the polytope lies entirely on one side of the hyperplane $\mathcal{H}$. Otherwise, the hyperplane intersects the polytope $\mathcal{V}$ and partitions it into two smaller polytopes.

2.2. Deep NNs

Definition 4. *(Deep NNs). A* deep NN *is a tuple $\mathcal{N} = (\mathcal{L}, \mathcal{W}, \Theta)$, where $\mathcal{L} = \{\mathcal{L}_k | k \in \{0, .., L+1\}\}$ is a set of layers, $\mathcal{W}\{\mathcal{W}_k | k \in \{1, .., L+1\}\}$ denotes a set of connection weights between layers, and $\Theta = \{\Theta_k | k \in \{1, ..., K+1\}\}$ is a set of activation functions, one for each non-input layer. In a* DNN*, $\mathcal{L}_0$ is the input layer, $\mathcal{L}_{L+1}$ is the output layer, and other layers are called hidden layers. Each layer $\mathcal{L}_k$ is constituted by l_k neurons.*

Definition 5. *(Feedforward NN). A* feedforward NN *$\mathcal{NN}_{\mathcal{FF}}(\chi_0)$ is a combination of m fully-connected (hidden) layers and a final output layer:*

$$\mathcal{NN}_{\mathcal{FF}}(\chi_0) = W_m\sigma(W_{m-1}\sigma(...\sigma(W_1\chi_0 + b_1)) + b_{m-1}) + b_m \tag{6}$$

where $\chi_0 \in \mathbb{R}^{n_0}$ is the input vector of NN, W_m denotes the connection weight matrix between layer m and $m-1$, b_m stands for the bias vector of the neurons in the layer m, and $\sigma(.)$ Indicates for activation function.

Definition 6. *(ReLU Activation Function). A* ReLU *is a PWL* activation function $\sigma : \mathbb{R} \to \mathbb{R}_{\geq 0}$*, where $\sigma(z) = \max(z, 0)$.*

Definition 7. *(ReLU NN) Consider definitions 4 and 5. For a* ReLU NN*, the mapping from input space $\mathbb{R}^{n_x}$ to the output space $\mathbb{R}^{n_o}$ is characterized by:*

$$\mathbb{R}^{n_x} \xrightarrow[ReLU]{W_1, b_1} \mathbb{R}^{l_1} \cdots \mathbb{R}^{l_{L-1}} \xrightarrow[ReLU]{W_L, b_L} \mathbb{R}^{l_L} \xrightarrow[ReLU]{W_{L+1}, b_{L+1}} \mathbb{R}^{n_o} \tag{7}$$

In this model, $l_0 = n_x$ and $l_{L+1} = n_o$), and $W_i \in \mathbb{R}^{l_i \times l_{i-1}}$ and $b_i \in \mathbb{R}^{l_i}$ denote the weight matrix and bias vector of layer i, respectively. In this chain, it is assumed that the activation function of the output layer operates as an identity mapping.

According to definitions 5 and 6, $\sigma_{L+1}(.)$ is an identity map, and the output of the layer i is defined as:

$$o_i = \begin{bmatrix} \max(0, o_{i1}) & \dots & \max(0, o_{il_i}) \end{bmatrix}^T \tag{8}$$

The Eq. (8) can be represented in a homogeneous form as:

$$\begin{bmatrix} o_i \\ 1 \end{bmatrix} = \sigma_i\Big(\Theta_i \begin{bmatrix} o_{i-1} \\ 1 \end{bmatrix}\Big) \tag{9}$$

where $\Theta_i = \begin{bmatrix} W_i & b_i \\ 0 & 1 \end{bmatrix}$ for $i = 1, ..., L$, and $\Theta_{L+1} = \begin{bmatrix} W_{L+1} & b_{L+1} \end{bmatrix}$. Consider χ as an input vector to a ReLU NN. Based on the preceding explanation, the complete NN function o can be explicitly expressed as a mapping from x to o as follows:

$$o = \Theta_{[}L+1]\Big(\prod_{i=1}^{L-1} \sigma_i \Theta_i \begin{bmatrix} \chi \\ 1 \end{bmatrix}\Big) \tag{10}$$

2.3. Transition System and ASR

Definition 8. *[1]. A system S is a sextuple $(X, X_0, U, \rightarrow, Y, H)$ consisting of:*

- *a set of states X;*
- *a set of initial states $X_0 \subseteq X$;*
- *a set of inputs U;*
- *a transition relation $\rightarrow \subseteq X \times U \times X$;*
- *a set of outputs Y;*
- *an output map $H : X \rightarrow Y$.*

A system is said to be:

- *metric, if the output set Y is equipped with a metric $textbf{d} : Y \times Y \rightarrow \mathbb{R}_0^+$;*
- *countable, if X is a countable set;*
- *finite, if X is a finite set.*

We adopt the notation $(x, u, y) \in \rightarrow$ by $x \xrightarrow{u} y$. In the context of a transition $x \xrightarrow{u} y$, the state y is referred to as a u-successor.

Definition 9. *[1] Let $S_a = (X_a, x_{a_0}, U_a, T_a)$ and $S_b = (X_b, x_{b_0}, U_b, T_b)$ be two transition systems. Given $\varepsilon \in \mathbb{R}^+$, a relation $\mathcal{R}(\varepsilon) \subseteq X_a \times X_b$ is called an ε-approximate ASR from S_a to S_b, if the following conditions are satisfied:*

1. *$(x_{a_0}, x_{b_0}) \in \mathcal{R}(\varepsilon)$;*
2. *For every $(x_a, x_b) \in \mathcal{R}(\varepsilon)$, we have $\|x_a - x_b\|_\infty \leq \varepsilon$;*
3. *For every $(x_a, x_b) \in \mathcal{R}(\varepsilon)$ and for every $u_a \in U_a(x_a)$, there exist $u_b \in U_b(_xb)$, such that the following holds: for every $(x\prime_b \in T_b(x_b, u_b)$, there exists $(x\prime_a \in T_a(x_a, u_a)$, such that $(x\prime_a, x\prime_b) \in \mathcal{R}(\varepsilon)$.*

The transition system S_a serves as the abstract expression of S_b, in the sense that every transition of S_b can be approximately simulated by those of S_a according to (1)–(3) in Definition 9. The concept of an ε-approximate ASR is particularly useful to synthesize a controller for the transition system S_b, based on the controller for S_a.

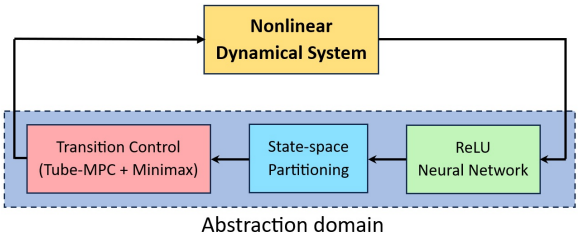

Figure 1: Abstract representation of nonlinear dynamical systems

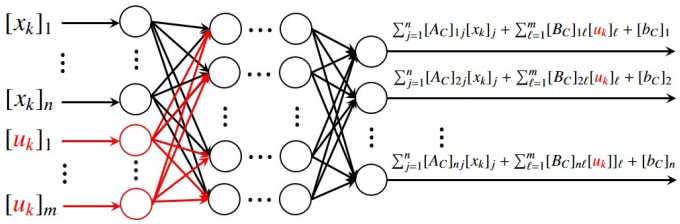

Figure 2: Learning the unknown dynamics $f(x_k, u_k)$ via ReLU NNs.

2.4. Problem Statement

Let us consider the following nonlinear discrete-time system:

$$x_{k+1} = f(x_k, u_k) + \delta_k, \tag{11}$$

with state $x_k \in \mathbb{R}^n$, control input $u_k \in \mathbb{R}^m$, unknown nonlinear function $f(x_k, u_k) \in \mathbb{R}^n$, and exogenous input $\delta_k \in \mathbb{R}^n$. The main objective of this paper is to construct an abstraction model of the control system (11), as illustrated in Fig. 1. Due to the presence of the unknown function $f(x_k, u_k)$ and the external disturbance δ_k, we propose a learning-based approach in which the symbolic model is constructed by learning the function $f(x_k, u_k)$ from training data. Based on the resulting abstraction, we then develop an algorithm to extract the essential polyhedral inequalities and label the resulting polyhedral cells. In particular, we introduce a safe and computationally efficient exploration strategy based on BCO, which identifies the optimal path toward the target polyhedral cell. Finally, leveraging optimal control techniques, we design a hybrid transition controller that regulates the system trajectory as it evolves between the desired polyhedral cells.

3. NA and SSP

In a ReLU NN, every hidden neuron has an associated binary neuron activation of zero or one, corresponding to whether the preactivation value is nonpositive or positive, respectively. Specifically, the activation flag for neuron j in layer i is given by:

$$\lambda_{ij} = \begin{cases} 1 & if \quad y_{ij} \geq 0 \\ 0 & if \quad y_{ij} \leq 0 \end{cases} \tag{12}$$

We define the activation pattern at layer i as a binary vector $\lambda_i = \begin{bmatrix} \lambda_{i1} & \dots & \lambda_{il_i} \end{bmatrix}^T$ and the activation pattern of the whole network as $\lambda = \begin{bmatrix} \lambda_1 & \dots & \lambda_{L+1} \end{bmatrix}$. Also, define the diagonal activation matrix for layer i as $\Lambda_i = \text{diag}(\lambda_i^T, 1)$, where a 1 is appended at the end to match the dimensions of the homogeneous form in (9).

Exploiting the universal approximation property of NNs [40], the unknown nonlinear dynamics $f(x_k, u_k)$ in Eq. (11) can be represented by a ReLU NN (Fig. 2). This yields the following equivalent affine state-space form:

$$x_{k+1} = A_C x_k + B_C u_k + b_C + \delta_k + \varepsilon_k, \tag{13}$$

where ε_k denotes the approximation error. The matrices of this linear model are directly obtained from the NN weights and biases as:

$$\begin{cases} A_C = W_{k+1}\text{diag}(\lambda_1)W_{11}\prod_{i=2}^{k}\text{diag}(\lambda_i)W_i, \\ B_C = W_{k+1}\text{diag}(\lambda_1)W_{12}\prod_{i=2}^{k}\text{diag}(\lambda_i)W_i, \\ b_C = b_{k+1} + \sum_{i=1}^{k}(W_{k+1}\prod_{j=i+1}^{k}\text{diag}(\lambda_j)W_j)\text{diag}(\lambda_i)b_i, \end{cases} \tag{14}$$

with $W_1 = [W_{11}\ W_{12}]$, where W_{11} maps the state x to the first layer and W_{12} maps the input u to the first layer. Since A_C, B_C, and b_C are explicitly constructed from the NN parameters, the full linear model (13) is known.

3.1. NN–Driven Hybrid Abstraction

NN–based abstraction couples learned system dynamics with state–space partitioning. A ReLU NN approximates the nonlinear dynamics, and the state space is partitioned into regions defining a hybrid automaton. This unified representation improves interpretability and enables efficient planning and control in high-dimensional nonlinear systems.

Definition 10. *(Neural Abstraction (NA)) Consider the discrete-time nonlinear system in* (11)*, where $f(x_k, u_k)$ denotes the true dynamics and δ_k is a bounded unknown input with $\|\delta_k\| \leq \bar{\delta}$. Let $\mathcal{X} \subseteq \mathbb{R}^n$ denote the operating domain. A ReLU NN $\mathcal{NN}_R(x_k, u_k) : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is called a NA of the system if it approximates $f(x_k, u_k)$ with a guaranteed uniform bound $\varepsilon > 0$ such that*

$$\|f(x_k, u_k) - \mathcal{NN}_R(x_k, u_k)\| \leq \varepsilon - \bar{\delta}, \qquad \forall x \in \mathcal{X}. \tag{15}$$

In Definition 9, assuming that all states of the transition system S are measurable, we can represent it as a quadruple $S = (X, X_0, U, \to)$. A symbolic model of S in a NA domain is constructed using a triple $\varrho = (\mathcal{D}_T, \theta_{\mathcal{NN}}, \varepsilon)$, where:

- $\mathcal{D}_T = \{\mathcal{D}_{T,1}, ..., \mathcal{D}_{T,n_x}\}$ denotes the set of training data;
- $\theta_{\mathcal{NN}}$ stands for the structure parameter of a ReLU NN including weights and biases;
- $\varepsilon \in \mathbb{R}^+$ is the approximation error parameter of a ReLU NN.

Given $\varrho = (\mathcal{D}_T, \theta_{\mathcal{NN}}, \varepsilon)$, we define a symbolic representation of S is as a quadruple $S_\varrho^N = (X_\varrho, X_{\varrho_0}, U_\varrho, \underset{\varrho}{\to})$, where

- X_ϱ represents the set of states;
- $X_{\varrho_0} \in X_\varrho$ denotes the set of initial state;
- U_ϱ is the set of inputs;
- $\underset{\varrho}{\to}$ is the transition map.

The following theorem establishes that an ε-approximate ASR exists from S_ϱ^N to S.

Theorem 1. *Let S be the original system and S_ϱ^N be its NA–based symbolic model. Define the relation*

$$\mathcal{R}(\varepsilon) = \{(x_k^N, x_k) \in X_\varrho \times X \mid \|x_k^N - x_k\|_\infty \leq \varepsilon\}. \tag{16}$$

Then $\mathcal{R}(\varepsilon)$ serves as an ε-approximate ASR from S_ϱ^N to S.

Proof: The proof proceeds by verifying the three requirements of Definition 9.

Condition 1: For every abstract state $x_k^N \in X_\varrho$, a corresponding state $x_k \in X$ exists such that $(x_k^N, x_k) \in \mathcal{R}(\varepsilon)$. Since S_ϱ^N is constructed with an approximation precision bounded by ε, each symbolic state x_k^N is guaranteed to satisfy $\|x_k^N - x_k\|_\infty \leq \varepsilon$ for some $x_k \in X$.

Condition 2: Whenever $(x_k^N, x_k) \in \mathcal{R}(\varepsilon)$, the definition of $\mathcal{R}(\varepsilon)$ in (16) ensures that $\|x_k^N - x_k\|_\infty \leq \varepsilon$ holds. Thus, the distance bound is respected.

Condition 3: Consider a transition $x_k^N \xrightarrow{u_k^\varrho} x_k'^N$ in S_ϱ^N for some input $u_k^\varrho \in U_\varrho$. We must show the existence of a matching transition $x_k \xrightarrow{u} x_k'$ in S such that $(x_k'^N, x_k') \in \mathcal{R}(\varepsilon)$. Since the NA introduces an approximation error ε, the transition map $\underset{\varrho}{\to}$ satisfies:

$$\|f_\varrho(x_k^N, u_k^\varrho) - f(x_k, u_k)\|_\infty \leq \varepsilon, \tag{17}$$

where $f_\varrho = A_C x_k^N + B_C u_k^\varrho + b_C$ describes the abstract dynamics (13), and $f(x_k, u_k)$ represents the true nonlinear map (11). Selecting an input u_k compatible with u_k^ϱ (e.g., via tube-MPC) yields a corresponding transition $x_k \xrightarrow{u_k} x_k'$ satisfying

$$\|x_k'^N - x_k'\|_\infty \leq \varepsilon. \tag{18}$$

Thus, $(x_k'^N, x_k') \in \mathcal{R}(\varepsilon)$, and the condition is satisfied.

Since all three requirements are satisfied, $\mathcal{R}(\varepsilon)$ defines an ε-approximate ASR from S_ϱ^N to S. □

In a ReLU NN with multiple hidden layers, each neuron in a hidden layer induces a hyperplane in the state space associated with the previous layer. This hyperplane partitions the space into two regions: one corresponding to the neuron being active and the other to the neuron being inactive. To be specific, for the j^{th} neuron in the i^{th} layer, these segments delineate the separate regions within the hyperplane created:

$$W_{i,j}o_{i-1} + b_{i,j} = 0 \tag{19}$$

where $W_{i,j}$ represents the j^{th} row in the weight matrix W_i, and $b_{i,j}$ refers to the j^{th} element within the bias vector b_i. Consequently, each activation pattern of the ReLU NN defines an intersection of half-spaces, yielding a polyhedral region in the input state space. Additionally, every configuration delineates a linear function originating from input neurons and extending to output neurons. These configurations collectively partition the input space, assigning each region to an associated affine function. When translated into a NA, this framework resembles a hybrid automaton, wherein each mode corresponds to a specific configuration of the hidden neurons. Furthermore, these configurations induce a system of affine Ordinary Differential Equations, as illustrated in Fig. 3.

Definition 11 (Neighboring Polyhedra)**.** *Two polyhedra, $\mathcal{P}_i$ and $\mathcal{P}_j$, are said to be* neighbors *if their intersection has nonzero Euclidean volume in $n-1$ dimensions, that is:*

$$|\mathcal{P}_i \cap \mathcal{P}_j|_{n-1} > 0, \tag{20}$$

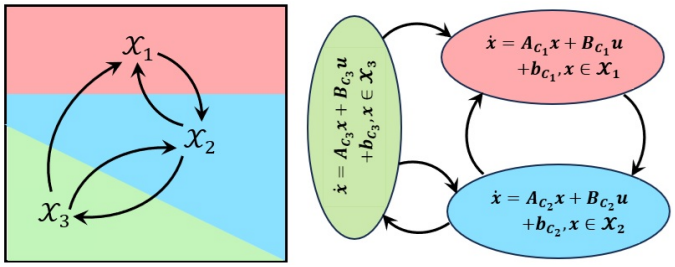

Figure 3: Illustration of a hybrid automaton derived from an SSP and its PWL dynamics formulation.

where $|\cdot|_{n-1}$ denotes the $(n-1)$-dimensional Euclidean volume of a set, computed with respect to the Lebesgue measure.

Definition 12. *Let $\mathcal{X} \subset \mathbb{R}^{n_x}$ be a compact set. A finite collection of sets $\chi_{cs} = \{\mathcal{X}_1, \ldots, \mathcal{X}_\mathcal{M}\}$ is called a* partition *of $\mathcal{X}$ if: (1) $\mathcal{X}_i \subseteq \mathcal{X}$ for all $i = 1, \ldots, \mathcal{M}$; (2) $\text{Int}(\mathcal{X}_q) \cap \text{Int}(\mathcal{X}_p) = \emptyset$ for all $q \neq p$; (3) $\mathcal{X} = \bigcup_{i=1}^{\mathcal{M}} \mathcal{X}_i$. Each set $\mathcal{X}_i$ in the partition χ_{cs} is called a* cell.

Definition 13. *(Convexity of Activation Regions). Consider a ReLU NN $\mathcal{NN}_R$. For any activation pattern λ_i and a vector $\theta_{\mathcal{NN}}$ representing the trainable parameters of $\mathcal{NN}_R$, each activation region $\mathcal{R}(\lambda_i, \theta_{\mathcal{NN}})$ is defined to be* convex.

Definition 14. *A* NN-based hybrid automaton *is defined by a tuple $(\mathcal{X}, \mathcal{X}_i, \mathcal{X}_0, \mathcal{U}, \rightarrow, \mathcal{F})$ in which the components are defined by:*

- *Cells: $\mathcal{X} \equiv \{\mathcal{X}_1, ..., \mathcal{X}_\mathcal{M}\}$ is a finite set of cells;*
- *State Variables: $\mathcal{X}_i \subset \mathbb{R}^n$, $i = 1, ..., \mathcal{M}$ is the set of state variables;*
- *Initial conditions: $\mathcal{X}_0$ is the initial state set;*
- *Inputs: $\mathcal{U} \equiv \{\mathcal{U}_1, ..., \mathcal{U}_m\}$ is the set of optimal controllers;*
- *Transitions $\rightarrow$: $\mathcal{X}_i \xrightarrow{\mathcal{U}_i} \mathcal{X}_j$ is the set of transitions from cell i to cell j;*
- *Set of piecewise dynamical systems: $\mathcal{F} \equiv \{\mathcal{F}_1, ..., \mathcal{F}_\mathcal{M}\}$ is the set of dynamical systems for each given cell $\mathcal{F}_i \in \mathcal{F}$.*

3.2. SSP and labeling

In this section, we present an algorithm for constructing the abstract state space. Leveraging the fact that ReLU NNs partition the continuous state space into polyhedral regions, our method avoids explicitly enumerating all possible polyhedra. Instead, we employ a binary space partitioning (BSP) strategy to incrementally generate only the regions necessary for analysis. This on-demand construction significantly reduces computational complexity compared to exhaustive region enumeration.

The continuous state space $\mathcal{X}$ is partitioned into polyhedral regions and represented as a BSP tree, whose structure is induced by the NN with N ReLU neurons. The root node corresponds to the entire domain $\mathcal{X}$. Construction begins by applying the first half-space constraint $\mathcal{H}_1$ (associated with the first neuron), which splits $\mathcal{X}$ into two convex polyhedra forming the left and right child nodes. The process then proceeds recursively: at each node, the next half-space constraint is imposed to further

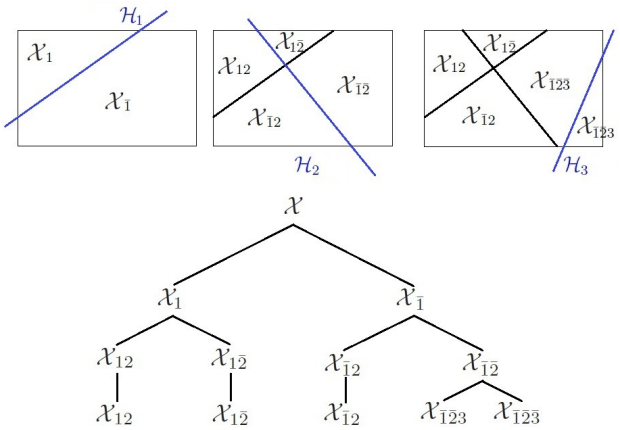

Figure 4: State-space abstraction using BSP

subdivide each non-empty polyhedron. Through this iterative refinement, the BSP tree encodes only the feasible polyhedral regions induced by the network's activation pattern, avoiding explicit enumeration of the full exponential region set. After all neuron-generated half-spaces are processed, each non-empty leaf corresponds to a valid abstract region. This process is illustrated in Fig. 4. The 2D rectangular space $\mathcal{X}$ is successively partitioned by $\mathcal{H}_1$, $\mathcal{H}_2$, and $\mathcal{H}_3$. First, $\mathcal{H}_1$ divides $\mathcal{X}$ into $\mathcal{X}_1$ and $\mathcal{X}_{\bar{1}}$. Applying $\mathcal{H}_2$ yields four regions: $\mathcal{X}_{12}$, $\mathcal{X}_{1\bar{2}}$, $\mathcal{X}_{\bar{1}2}$, and $\mathcal{X}_{\bar{1}\bar{2}}$. Only $\mathcal{X}_{\bar{1}\bar{2}}$ intersects $\mathcal{H}_3$, so it alone is further partitioned. The resulting BSP tree simultaneously provides the geometric state-space partition and an efficient search structure over the abstract regions.

Fig. 5 demonstrates an instance of partitioning a convex polytope using a line. As shown in Fig. 5(b), at first, the constraint of active neurons is added to the main polyhedral constraint. Then, the essential constraints of the portioned polyhedral are checked, and the constraints that violate the inequality are removed (Fig. 5 (c) and (d)). According to the explanations given in previous subsections, the main SSP algorithm is given in Fig. 6.

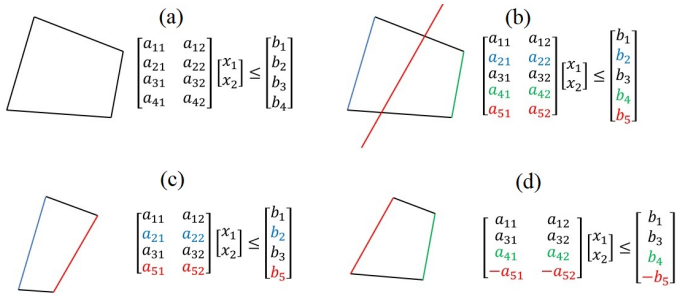

Figure 5: Polyhedral cell partitioning: (a) base cell with inequalities, (b) main intersections, redundant edges, and added line constraints, (c) left cell with inequalities, and (d) right cell with inequalities after partitioning.

Remark 1. *Previous studies have shown that ReLU NNs partition the input space into convex polyhedral regions [41, 42]. In contrast, our work introduces a distinct approach: we use a BSP-based construction, driven by the NN's weights and activation patterns, to explicitly generate an abstract state-space.*

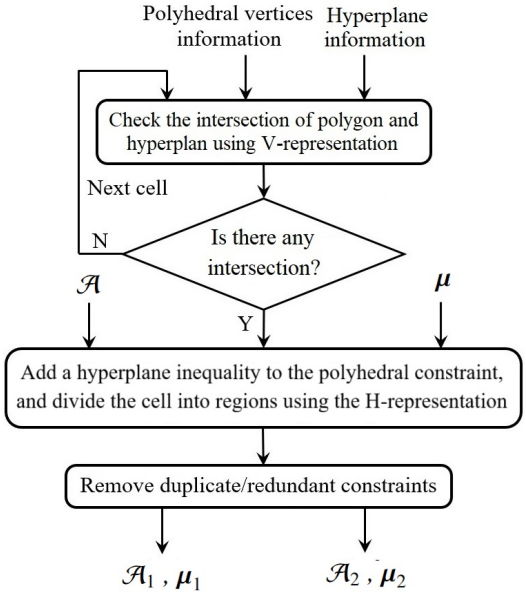

Figure 6: The main SSP algorithm.

This abstraction is then leveraged for BCO in global path planning and safety-aware hybrid control synthesis. While building on the known geometric properties of ReLU NNs, the integration of BSP-driven NA with BCO and robust optimal reach control represents a novel contribution of this study.

4. Efficient Safe Path Exploration Algorithm

In this section, a BCO algorithm is introduced for finding the optimal path between the initial and target cells, which are represented by polyhedral templates and characterized by binary values, while avoiding obstacles.

4.1. Path Planning

Consider $\mathcal{X} \subset \mathbb{R}^n$ as the state space, where the obstacle space is denoted as $\mathcal{X}_{obs}$ and the free space as $\mathcal{X}_{free} = \mathcal{X} \setminus \mathcal{X}_{obs}$. The objective of path planning is to discover a feasible path $\phi(t)$ satisfying:

$$\phi(t) : [t_0, T] \to \mathcal{X}_{free},\ \phi(t_0) \in \mathcal{X}_{init},\ \phi(T) \in \mathcal{X}_{goal} \tag{21}$$

where $\mathcal{X}_{init}$ represents the initial state set, $\mathcal{X}_{goal}$ denotes the goal set, t_0 is the initial time, and T is the final time.

Let Φ_f denote the set of all feasible paths. Introducing a cost function $c_c : \phi \to \mathbb{R}^+$, which assigns each feasible path $\phi \in \mathcal{X}_{free}$ a positive real number, the optimal path planning problem can be formulated as follows:

$$\begin{aligned} \phi^* &= \arg\min_{\phi \in \Phi} \ \ c_c(\phi) \\ &subject\ \ to: \quad (21) \end{aligned} \tag{22}$$

The cost function between two cells is defined as:

$$c_c(\phi) = \sum_{t=0}^{T} \text{Cost}(\phi(t), \phi(t+T)) \tag{23}$$

Problem 1. *(Feasible Path Solution). Find a path $\phi_f : [T_0, t]$, if one exists, within the obstacle-free space $\mathcal{X}_{free} \subset \mathcal{X}$ satisfying $\phi_f(0) = \mathcal{X}_{init} \in \mathcal{X}_{free}$ and $\phi_f(T) = \mathcal{X}_{goal}$. If no such path exists, report failure.*

Problem 2. *(Optimal Path Solution). Find an optimal path $\phi_f^* : [t_0, T]$ connecting $\mathcal{X}_{init}$ and $\mathcal{X}_{free}$ in obstacle-free space $\mathcal{X}_{free} \subset \mathcal{X}$, such that the cost of the path ϕ_f^* is minimum, i.e., $c_c(\phi_f^*) = \{\min_{\phi_T} c_c(\phi_f) : \phi_f \in \Phi_f\}$.*

4.2. BCO-based Safe Path Planning

Definition 15. ***(Combinatorial Optimization Problem (COP))*** *A COP is formally defined as a quadruple $C_{\text{OP}} = (\mathcal{I}, \mathcal{Z}, d, \mathcal{G})$, whose components are given as follows:*

- *$\mathcal{I}$ is the set of problem instances;*
- *For each instance $\beta \in \mathcal{I}$, $\mathcal{Z}(\beta)$ denotes the finite set of feasible solutions;*
- *$d(\beta, z)$ is the cost (or objective) associated with instance β and solution $z \in \mathcal{Z}(\beta)$, typically a real-valued function;*
- *$\mathcal{G}$ is an optimization operator, either* min *or* max.

The goal is to find an optimal solution $z^* \in \mathcal{Z}(\beta)$ for a given instance β such that:

$$d(\beta, z^*) = \mathcal{G}\{d(\beta, z) \mid z \in \mathcal{Z}(\beta)\}. \tag{24}$$

In the partitioned state space composed of polyhedral cells, the proposed BCO algorithm begins by constructing feasible transition sequences from the initial to the target region. The algorithm explores all admissible paths that move toward the target while respecting the optimality constraints defined over the abstract graph of polyhedral regions. An illustrative example of transitions between polyhedral cells using binary flags in the proposed BCO framework is shown in Fig. 7. It's assumed that the state space is partitioned using a ReLU NN with 12 neurons. $\mathcal{X}_0$ and $\mathcal{X}_\mathcal{M}$ stand for initial and target polytopes, in which some neurons are active and some are inactive. There are two common bits between initial and target polyhedrons. The total number of transitions is determined by applying the proposed BCO algorithm. Notably, a majority of transitions demonstrate that altering one bit leads to minimum cost, except for the transition from cell $\mathcal{X}_1$ to cell $\mathcal{X}_2$, where a 2-bit change results in a lower cost compared to bit-by-bit alteration. A pivotal characteristic of this approach lies in the design of an optimal path between successive middle cells at each step, facilitating its applicability in online path planning. Ultimately, the shortest optimal path between the initial and target is chosen by connecting the devised trajectories amid the middle cells that the agent will traverse.

4.3. Algorithm Description

Input parameters for the algorithm include:

1. $\mathcal{S}_n$, The set of flags representing neuron activation within the next polyhedral cell. For instance, in a ReLU NN with 10 neurons, this set is denoted as 0100111001, where 0 signifies inactive and 1 signifies active flags.
2. $\mathcal{S}_m$, the set of neuron flags within of the current cell.
3. $\mathcal{S}_T$, the set of neuron flags within of the target cell.

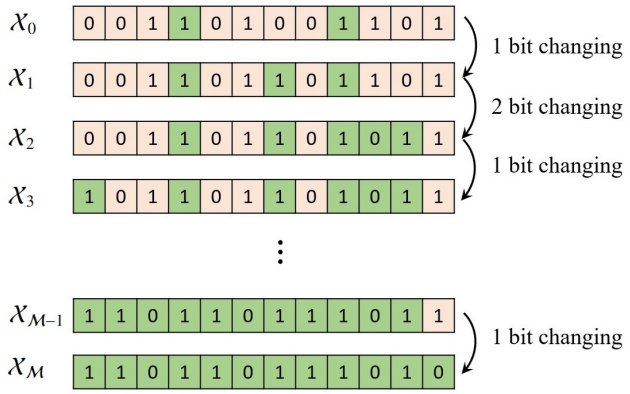

Figure 7: Illustration of polyhedral cell transitions using binary activation flags in the BCO framework. A ReLU NN with 12 neurons partitions the state space, with $\mathcal{X}_0$ and $\mathcal{X}_{\mathcal{M}}$ denoting the initial and target cells. In this example, two bits match, and the BCO evaluates all feasible bit-flip sequences. The optimal transition uses a 2-bit flip, yielding lower cost than sequential single-bit changes.

4. $\mathcal{S}_b$, the set of neuron flags within of the barrier cell.
5. N_{max}, The maximum allowable number of bit changes.
6. N, The total number of neurons in the NN structure, equivalent to the number of bits.

Therefore, the components of COP in Definition 15 can be specified as:

- Instance Set $\mathcal{I}$:
$$\mathcal{I} = \{\mathcal{S}_0, \mathcal{S}_T\},$$
- Feasible Solutions $\mathcal{Z}(\mathcal{S}_0)$:
$$\mathcal{Z}(\mathcal{S}_0) = \{\{\mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_k = \mathcal{S}_T\} \mid |\mathcal{S}_i \oplus \mathcal{S}_{i+1}| \leq N_{max}, \forall i\}$$
where $\oplus$ denotes the bitwise XOR operation.
- Cost Function $d(\mathcal{S}_i, \mathcal{S}_{i+1})$:
$$d(\mathcal{S}_i, \mathcal{S}_{i+1}) = \text{cost of transitioning from } \mathcal{S}_i \text{ to } \mathcal{S}_{i+1},$$
- Goal Function $\mathcal{G}$:
$$\mathcal{G} = \min \sum_{i=0}^{k-1} d(\mathcal{S}_i, \mathcal{S}_{i+1}).$$

Algorithm 1 provides a detailed explanation of the BCO algorithm.

4.4. Complexity Analysis and Algorithmic Guarantees

4.4.1. Computational Complexity

At each step, the algorithm enumerates all possible bit–flip combinations up to N_{max}, yielding:

$$C_{tot} = \sum_{i=1}^{N_{max}} \binom{N}{i}$$

candidate transitions. For each candidate cell $\mathcal{S}_n$, the cost Q_c is computed once and compared to the current minimum. Hence, the per-iteration computational cost is $O(C_{tot})$, and in the worst case (when $N_{max} = N$), the complexity is $O(2^N)$, which corresponds to exhaustive enumeration of all binary states (blue line in Fig. 8). However, in practice $N_{max} \ll N$, making the algorithm quasi–linear with respect to N since $\sum_{i=1}^{N_{max}} \binom{N}{i} \approx O(N^{N_{max}})$ (red line in Fig. 8). Thus, the proposed BCO can operate online for moderate-dimensional partitions because only a limited subset of neighboring cells is evaluated at each step. The memory requirement is proportional to the number of candidate cells, i.e., $O(C_{tot})$.

Algorithm 1 BCO algorithm

```
// Q_c is the cost from the current cell to the next cell.
// S_o is the optimal set of neuron's flags. The target related to
this cell is selected as the next transition cell.
while S_o ≠ S_T
    // Q_min is the minimum cost for transition between
       cells.
    Q_min ← ∞,
    for i=1: N_max
        for j=1: (N choose i)
            S_n = FindNextCell(i, j, S_m),
            if S_n ≠ S_b
                u* = ū + K_e*(x − x̄),   // take action
                Q_c = cost(S_m, S_n, u*),   // calculate cost
                if Q_c < Q_min
                    Q_min ← Q_c,
                    S_o ← S_n,
                end
            end
        end
    end
    S_m ← S_o,
end
```

4.4.2. Correctness Guarantee

At every iteration, the algorithm selects the feasible neighboring cell $\mathcal{S}_n$ that minimizes the local transition cost $d(\mathcal{S}_m, \mathcal{S}_n)$ while ensuring $\mathcal{S}_n \neq \mathcal{S}_b$ (barrier) and $|\mathcal{S}_m \oplus \mathcal{S}_n| \leq N_{max}$. Since each move strictly reduces the accumulated cost and $\mathcal{S}_T$ is reachable in a finite number of transitions, the algorithm terminates in at most $\mathcal{M}$ steps, where $\mathcal{M}$ is the number of polyhedral cells along the feasible path. Therefore, the procedure is guaranteed to find a valid sequence of cells connecting $\mathcal{S}_0$ to $\mathcal{S}_T$ whenever such a sequence exists under the bit–change and barrier constraints.

4.4.3. Optimality

The resulting trajectory is *sub–optimal with respect to the discrete transition cost* defined over the graph of polyhedral cells. Global optimality in the original continuous domain is not guaranteed, as the abstraction maps continuous dynamics into a finite symbolic structure and uses approximate cell–to–cell cost evaluations. However, when (i) each transition cost $d(\mathcal{S}_i, \mathcal{S}_{i+1})$ is *conservative* with respect to the true system cost (i.e., it upper–bounds the continuous cost for that transition), and (ii) all feasible neighboring cells are evaluated at each expansion

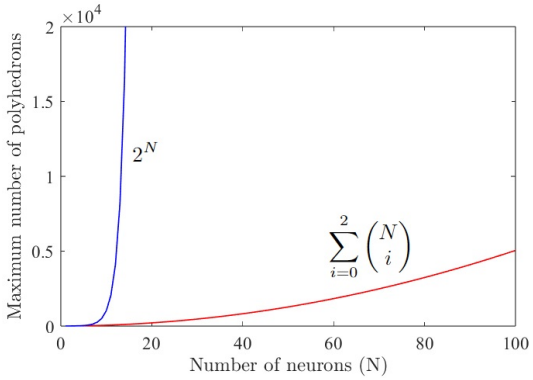

Figure 8: Comparison between the exponential growth of binary partitions 2^N (blue line) and the restricted neighbor expansion $\sum_{i=0}^{2}\binom{N}{i}$ used in the proposed BCO method (red line).

step, Algorithm 1 yields the *shortest path on the induced discrete graph*. Accordingly, the proposed planner solves a CBO consistent with Definition 15, exploring sequences in $\mathcal{Z}(\mathcal{S}_0)$ and accumulating transition costs $d(\mathcal{S}_i, \mathcal{S}_{i+1})$ to minimize the discrete objective $\mathcal{G}$. Under conservative transition costs and exhaustive evaluation of neighbors satisfying $|\mathcal{S}_i \oplus \mathcal{S}_{i+1}| \leq N_{\max}$, the method provides an optimal solution within the discrete abstraction while remaining *sub–optimal in the continuous domain*.

5. Robust Optimal Transition Control Design

The dynamics in Eq. (13) depend on ReLU activation patterns, where each pattern corresponds to an affine model valid in a specific polyhedral region. This yields a PWL affine system of the form:

$$x_{k+1} = A_i x_k + B_i u_k + b_i + \omega_k, \qquad i = 1, \ldots, \mathcal{M}, \tag{25}$$

where $\omega_k = \varepsilon_k + \delta_k$, and each tuple (A_i, B_i, b_i) corresponds to the affine dynamics associated with cell i. For the purpose of transition–controller synthesis, we focus on the local dynamics within a specific region and across its neighboring cells. Thus, the dynamics used for control design are expressed as:

$$x_{k+1} = Ax_k + Bu_k + b + \omega_k, \tag{26}$$

representing the affine model governing the transition between two adjacent polyhedral cells. Ignoring ω_k yields the nominal system:

$$\bar{x}_{k+1} = A\bar{x}_k + B\bar{u}_k + b. \tag{27}$$

Define the deviation between real and nominal trajectories as:

$$\tilde{x}_k = x_k - \bar{x}_k. \tag{28}$$

Substitution gives the error system:

$$\tilde{x}_{k+1} = A\tilde{x}_k + B\tilde{u}_k + \omega_k, \tag{29}$$

where $\tilde{u}_k = u_k - \bar{u}_k$. Let x_r denote the desired position of the target polytope. The nominal error relative to x_r evolves as:

$$\bar{e}_{k+1} = A\bar{x}_k + B\bar{u}_k + b - x_r. \tag{30}$$

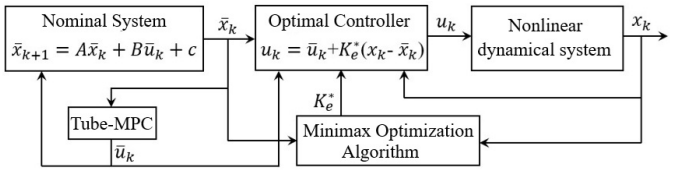

Figure 9: Hybrid control framework combining tube-MPC for nominal guidance and a minimax control for robust error correction.

The control input is decomposed as:

$$u_k = \bar{u}_k + \tilde{u}_k, \tag{31}$$

where the nominal term $\bar{u}_k$ is generated by a tube–MPC, and the corrective term is:

$$\tilde{u}_k = K_e \tilde{x}_k, \tag{32}$$

with K_e obtained via Minimax control policy. Fig. 9 illustrates the proposed control system architecture: the tube-MPC drives the nominal dynamics toward the next region, while the minimax controller compensates modeling errors and ensures robustness.

5.1. Tube-MPC Formulation

The tube–MPC regulates the nominal system while satisfying tightened constraints to preserve feasibility under disturbances. The finite–horizon cost to be minimized is defined as:

$$J(\bar{x}_k, \bar{u}_k) = J_T(\bar{x}_{k+N|k}) + \sum_{i=k}^{N+k-1} J_s(\bar{x}_{i|k}, \bar{u}_{i|k}) \tag{33}$$

where N is the prediction horizon, $J_s(\bar{x}_{i|k}, \bar{u}_{i|k}) = \|\bar{x}_{i|k}\|^2_{Q_{MPC}} + \|\bar{u}_{i|k}\|^2_{R_{MPC}}$ stands for the stage cost, $J_T(\bar{x}_{k+N|k}) = \|\bar{x}_{k+N|k}\|^2_{P_{MPC}}$ is the terminal cost, and $P_{MPC}, Q_{MPC}, R_{MPC} \in \mathbb{R}^{n\times n}$ are symmetric and positive semi-definite weighting matrices. The finite-horizon tube-MPC problem is formulated as:

$$\begin{aligned}
(\bar{x}_k^*, \bar{u}_k^*) &= \arg\min_{\{\bar{x}_{i|k}, \bar{u}_{i|k}\}} J(\bar{x}_k, \bar{u}_k) \\
\text{s.t.} \quad \bar{x}_{i+1|k} &= A\bar{x}_{i|k} + B\bar{u}_{i|k} + b, \\
e_{i+1|k} &= A\bar{x}_{i|k} + B\bar{u}_{i|k} + b - x_r, \\
\bar{x}_{i|k} &\in \bar{\mathcal{X}}_{\text{tube}}, \\
\bar{u}_{i|k} &\in \bar{\mathcal{U}}_{\text{tube}}, \\
\bar{x}_{k+N|k} &\in \bar{\mathcal{X}}_{\text{tube},T}, \\
\bar{x}_{k|k} &= \bar{x}_k,
\end{aligned} \tag{34}$$

where $i = k, \ldots, k+N-1$. The optimization enforces tube constraints on both the predicted nominal states $\bar{\mathcal{X}}_{\text{tube}}$ and inputs $\bar{\mathcal{U}}_{\text{tube}}$ to guarantee constraint satisfaction while avoiding obstacles. The terminal condition $\bar{x}_{k+N|k} \in \bar{\mathcal{X}}_{\text{tube},T}$ with terminal cost J_T ensures recursive feasibility and exponential convergence, driving $\bar{x}_k \to x_r$.

5.2. Minimax control policy design

In this section, the minimax optimal policy design procedure is formulated for the linear discrete-time systems Eq. (29). For

system (29), the infinite-horizon H_∞ tracking problem is to find a control policy u_k such that:

1) The full measured states of system (29) converge to the origin with $\omega_k = 0$,

2) The disturbance attenuation condition is satisfied:

$$\frac{\sum_{i=k}^{\infty} \alpha^{i-k}[\tilde{x}_i^T Q \tilde{x}_i + u_i^T R u_i]}{\sum_{i=k}^{\infty} \alpha^{i-k} \omega_i^T \omega_i} \leq \gamma^2 \tag{35}$$

where $\alpha \in (0, 1)$ represents the discount factor, γ is the upper bound denoting the desired $\mathcal{L}_2$ gain disturbance attenuation level, and Q and R are both positive symmetric weighting matrices.

Assumption 1. *In system* (29)*, it is assumed that the pairs* (A, B) *and* $(A, Q^{\frac{1}{2}})$ *are controllable and observable, respectively, where* Q *is defined in Eq.* (35)*.*

Assumption 2. *The positive semi-definite matrix* P *is the solution of the following algebraic Riccati equation (GARE) [43]:*

$$\begin{aligned} P =& Q + \alpha A^T P A - \begin{bmatrix} \alpha A^T P B & \alpha A^T P \end{bmatrix} \\ & \times \begin{bmatrix} R + \alpha B^T P B & \alpha B^T P \\ \alpha P B & \alpha P - \gamma^2 I \end{bmatrix}^{-1} \begin{bmatrix} \alpha B^T P A \\ \alpha P A \end{bmatrix}. \end{aligned} \tag{36}$$

The H_∞ control problem (35) can be considered as a two-player zero-sum linear quadratic dynamic game, in which the controller and disturbance are regarded as minimizing and maximizing players, respectively. The reward function is defined as:

$$r(\tilde{x}_i, u_i, \omega_i) = \tilde{x}_i^T Q \tilde{x}_i + u_i^T R u_i - \gamma^2 \omega_i^T \omega_i \tag{37}$$

The primary goal is to address the subsequent minimax optimization problem:

$$\begin{aligned} u_k^*, \omega_i^* &= \min_{u_k} \max_{\omega_k} V(\tilde{x}_k) \\ &= \min_{u_k} \max_{\omega_k} \sum_{i=k}^{\infty} \alpha^{i-k} r(\tilde{x}_k, u_i, \omega_i) \end{aligned} \tag{38}$$

where $V(\tilde{x}_k) = \alpha^{i-k} r(\tilde{x}_k, u_i, \omega_i)$ represents the value function corresponding to admissible control policy u_k and ω_k. The objective of the zero-sum dynamic game is to find the feedback saddle solution (u_k^*, ω_i^*) that satisfies the following inequality for any admissible control policies u_k and ω_k:

$$V(\tilde{x}_k, u_k^*, \omega_k) \leq V(\tilde{x}_k, u_k^*, \omega_k^*) \leq V(\tilde{x}_k, u_k, \omega_k^*) \tag{39}$$

The inequality (39) indicates that no player will deviate from (u_k^*, ω_i^*) because a unilateral change of strategy will cause a loss of revenue for both players.

According to Bellman's principle of optimality, the feedback saddle solution (u_k^*, ω_i^*) is expected to meet the following Bellman optimality equation:

$$V(\tilde{x}_k) = r(\tilde{x}_k, u_k, \omega_k) + \alpha \sum_{i=k}^{\infty} \alpha^{i-k-1} r(\tilde{x}_i, u_i, \omega_i) \tag{40}$$

which can be expressed as:

$$V(\tilde{x}_k) = r(\tilde{x}_k, u_k, \omega_k) + \alpha V(\tilde{x}_{k+1}) \tag{41}$$

Here, the main goal is to find an optimal solution for the state-dependent value function $V(x_k)$ which satisfies the GARE introduced in Assumption 2. Assuming that the game has a value and is solvable, then the value function takes a quadratic form in the state and is represented as:

$$V(\tilde{x}_k) = \tilde{x}_k^T P \tilde{x}_k \tag{42}$$

Thus, the Bellman equation (41) is transformed into:

$$\tilde{x}_k^T P \tilde{x}_k = r(\tilde{x}_i, u_i, \omega_i) + \alpha \tilde{x}_{k+1}^T P \tilde{x}_{k+1} \tag{43}$$

Define the Hamiltonian function as:

$$\mathcal{H}(\tilde{x}_k, u_k, \omega_k) = r(\tilde{x}_k, u_k, \omega_k) + \alpha V(\tilde{x}_{k+1}) - V(\tilde{x}_k) \tag{44}$$

The following theorem shows the result of the minimax controller design for the discrete-time linear system (29).

Theorem 2. *The minimax optimal policy control* u^* *and the worst-case disturbance take the following form:*

$$u_k^* = -K_e^* \tilde{x}_k \tag{45}$$

$$w_k^* = -K_w^* \tilde{x}_k \tag{46}$$

where the coefficients K_v^* *and* K_w^* *are calculated by:*

$$\begin{aligned} K_e^* =& [R + \alpha B^T P B - \alpha B^T P (\alpha P - \gamma^2)^{-1} \alpha P B]^{-1} \\ & \times [\alpha B^T P A - \alpha B^T P (\alpha P - \gamma^2 I)^{-1} \alpha P A] \end{aligned} \tag{47}$$

$$\begin{aligned} K_w^* =& [\alpha P - \gamma^2 I - \alpha P B (R + \alpha B^T P B)^{-1} \alpha B^T P]^{-1} \\ & \times [\alpha P A - \alpha P B (R + \alpha B^T P B)^{-1} \alpha B^T P A] \end{aligned} \tag{48}$$

where I *denotes the identity matrix with an appropriate dimension.*

Proof. An essential requirement for optimality is that the control policy and the disturbance should satisfy the conditions $\frac{\partial \mathcal{H}}{\partial u_k} = 0$ and $\frac{\partial \mathcal{H}}{\partial w_k} = 0$. Therefore, from (44), we obtain:

$$\begin{cases} (R + \alpha B^T P B) u_k + \alpha B^T P w_k + \alpha B^T P A \tilde{x}_k = 0 \\ \alpha B^T P B u_k + (\alpha P - \gamma^2 I) w_k + \alpha P A \tilde{x}_k = 0 \end{cases} \tag{49}$$

By simultaneously solving the two equations presented in (49), we can derive the optimal control policy and worst-case disturbance, as expressed in Eqs. (45) and (46). Additionally, upon substituting Eqs. (45) and (46) into the Bellman equation (43), we obtain the GARE Eq. (36). □

Remark 2. *In order to establish unique feedback-stabilizing policies as depicted in* (45) *and* (46)*, the following inequalities need to be satisfied:*

$$I - \alpha \gamma^{-2} P > 0 \tag{50}$$

$$R + \alpha B^T P B > 0 \tag{51}$$

From Eqs. (47) and (48), it is evident that determining K_v^* and ω_v^* necessitates solving the GARE, which constitutes a nonlinear matrix equation. In Algorithm 2, a policy iteration algorithm is developed to solve the GARE equation (36) and extract the optimal values for K_e^* and K_w^*.

Algorithm 2 Policy iteration algorithm for solving GARE

Input: Initial values for iteration number, $j = 0$, and feedback gains K_e^0, K_ω^0.

1. Policy evaluation: find P^{j+1} using Bellman equation:

$$P^{j+1} = Q + (K_e^j)^T R K_e^j - \gamma^2 (K_w^j)^T K_w^j + (A - BK_e^j - K_w^j)^T P^{j+1} (A - BK_e^j - K_w^j)$$

2. Policy improvement:

$$K_e^{j+1} = (R + \alpha B^T P^{j+1} B - \alpha B^T P^{j+1} (\alpha P^{j+1} - \gamma^2 I)^{-1} \alpha P^{j+1} B)^{-1} (\alpha B^T P^{j+1} A - \alpha B^T P^{j+1} (\alpha P^{j+1} - \gamma^2 I)^{-1} \alpha P^{j+1} A)$$

$$K_w^{j+1} = (\alpha P^{j+1} - \gamma^2 I - \alpha P^{j+1} B (R + \alpha B^T P^{j+1} B)^{-1} \alpha B^T P^{j+1})^{-1} (\alpha P^{j+1} A - \alpha P^{j+1} B (R + \alpha B^T P^{j+1} B)^{-1} \alpha B^T P^{j+1} A)$$

3. Stop if $\|K_i^{j+1} - K_i^j\| \leq \kappa$, $i \in \{v, w\}$, where κ is a threshold; otherwise, set $j = j + 1$ and go to step 1.

Note that in both tube-MPC and the minimax optimization problem, the optimal decision variables $\bar{x}_{k+1}$ and $\tilde{x}_{k+1}$ are expressed as piecewise affine functions of the current states $\bar{x}_k$ and $\tilde{x}_k$. Consequently, this indicates that the control law (31) is also piecewise affine. To be more precise, we can express:

$$u_k = \begin{cases} \bar{u}_{k,1}^* + K_{e,1}^* (x_{k,1} - \bar{x}_{k,1}), & if \quad x \in \mathcal{P}_1 \\ \quad \vdots & \\ \bar{u}_{k,\mathcal{M}}^* + K_{e,N}^* (x_{k,\mathcal{M}} - \bar{x}_{k,\mathcal{M}}), & if \quad x \in \mathcal{P}_\mathcal{M} \end{cases} \tag{52}$$

The sets $\mathcal{P}_1, \ldots, \mathcal{P}_\mathcal{M}$ represents convex polytopes with pairwise disjoint interiors, and their union $\bigcup_{i=1}^{\mathcal{M}} \mathcal{P}_i$ forms the set $\mathcal{X}$.

Remark 3. *The adoption of tube-MPC in the proposed hybrid framework is essential to ensure robust constraint satisfaction within the polyhedral regions determined by BCO algorithm. In the ReLU NN-based NA, the true nonlinear dynamics are approximated by a PWL affine model, which introduces bounded modeling uncertainties, particularly near the boundaries of adjacent polyhedral cells. A standard MPC formulation, which assumes perfect model knowledge, may therefore lead to constraint violations or unstable transitions between cells. Tube-MPC addresses this issue by constructing a robust invariant "tube" around the nominal trajectory, guaranteeing that all possible realizations of the perturbed system remain within the corresponding polyhedral cell prescribed by the BCO. This mechanism ensures that both the nominal and actual trajectories remain feasible and consistent with the planned transitions, preserving stability and safety during the entire path planning process.*

Remark 4. *In contrast to conventional nonlinear approximation approximation such as Gaussian Processes, fuzzy models, or Koopman-based lifting [44], the proposed ReLU-based SSP algorithm produces an explicit PWL abstraction of the unknown dynamics within polyhedral regions (Fig. 10). This structure enhances safety, interpretability, and real-time feasibility by enabling direct enforcement of state and input constraints in each region, supporting global path planning through BCO, and permitting real-time control synthesis via hybrid tube-MPC with minimax error compensation. Such properties are particularly beneficial in safety-critical and time-sensitive scenarios—including autonomous vehicle navigation, pendulum swing-up stabilization, and dynamic legged-robot locomotion—where nonlinearities, contacts, and actuator limits require transparent and certifiable models. The approach further resolves key challenges associated with balancing approximation accuracy and partition complexity, maintaining tractable online computation, and guaranteeing safe transitions across polyhedral cells, limitations that traditional black-box learning techniques do not adequately address.*

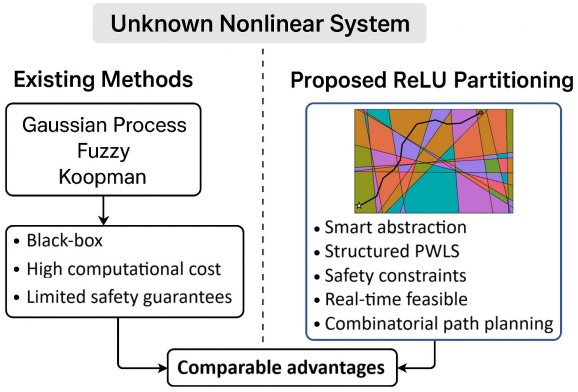

Figure 10: Comparison of existing nonlinear modeling methods and the proposed ReLU-based partitioning.

Remark 5. *The proposed hybrid control framework, which integrates tube-MPC with a minimax feedback policy, is intentionally designed to balance* optimality, robustness, *and* constraint satisfaction *in the NA domain. The tube-MPC component optimizes the nominal trajectory while guaranteeing recursive feasibility and constraint satisfaction across polyhedral partitions induced by the ReLU NN. However, since the ReLU-based piecewise affine approximation inevitably introduces bounded modeling errors and external disturbances, the minimax control law is incorporated to attenuate these uncertainties and ensure closed-loop robustness. This combination allows the overall controller to maintain stability and safety during transitions between polyhedral cells, even when the underlying nonlinear dynamics are unknown. In contrast, a pure tube-MPC scheme would be sensitive to unmodeled dynamics, while a standalone minimax controller would neglect the optimization and constraint-handling aspects. Therefore, the proposed hybrid design provides a systematic way to achieve both disturbance rejection and trajectory optimality under NA-based PWL dynamics.*

6. Numerical Evaluation

In this section, the efficacy of the proposed data-driven NA and hybrid control approach is investigated through three benchmark examples: (i) vehicle path planning, (ii) dynamic identification and control of double-pendulum robot, and (iii) robustness analysis of tube-MPC on inverted pendulum robot. Each model employed a one layer ReLU NN trained with the Adam optimizer using a learning rate of 10^{-3}, batch size of 64, and a maximum of 200 epochs.

6.1. Vehicle path Planning Benchmark

The first example considers the design of an optimal controller for a nonholonomic vehicle tasked with reaching a target polyhedral region while avoiding obstacles. The vehicle state is defined as $\xi = [x, y, \theta]^T$, where (x, y) denote the vehicle position and θ its orientation. The discrete-time dynamic model of the vehicle is as [45, Example A]:

$$\begin{cases} x_{k+1} = x_k + \Delta t v_k \dfrac{\cos(\theta_k + \varphi_k)}{\cos(\varphi_k)} \\ y_{k+1} = y_k + \Delta t v_k \dfrac{\sin(\theta_k + \varphi_k)}{\cos(\varphi_k)} \\ \theta_{k+1} = \theta_k + \Delta t v_k \tan(w_k), \end{cases} \tag{53}$$

where $\varphi_k = \arctan\left(\frac{\tan(w_k)}{2}\right)$, the state satisfies $\xi_k \in [-2.5, 2.5]^2 \times [-\pi/2, \pi/2]$, and the control input is $u_k = [v_k, w_k]^T \in [-1, 1]^2$. The variables v_k and w_k correspond to the rear-wheel velocity and steering angle, respectively. Here, x, y are in metres (m), θ, φ_k, w_k are in radians (rad), and v_k is in m/s. The sampling period is $\Delta t = 0.05$s. Within the GARE-based optimization framework, the parameters are selected as $Q = \text{diag}(1000, 1200, 1500)$, $P = \text{diag}(100, 100, 150)$, $R = 5$, $\gamma = 5$, $\alpha = 0.025$, and the maximum iteration number is 100.

A ReLU NN with five inputs $[x_k, y_k, \theta_k, v_k, w_k]^T$, three outputs $[x_{k+1}, y_{k+1}, \theta_{k+1}]^T$, a dataset of 1500 samples, and a single hidden layer of 25 neurons (Fig. 11(a)) was trained to approximate the nonlinear dynamics in (53) with a PWL representation. The training loss is shown in Fig. 11(b).

Fig. 12 depicts the resulting polyhedral state-space partition, highlighting the initial cell (red), the target cell (green), and the obstacle regions (blue). The blue trajectory demonstrates successful collision-free navigation of the vehicle to the goal using the BCO and hybrid control scheme. Table 1 presents the tracking error statistics from simulation. The results show small position and orientation deviations, demonstrating that the controller achieves reliable trajectory tracking despite the PWL model approximation.

Table 1: Error Statistics for vehicle.

Error Statistic	Max. Error	Variance	Mean Error
Δx	0.171 m	0.049	0.192 m
Δy	0.132 m	0.061	0.083 m
$\Delta\theta$	0.112 rad	0.054	0.072 rad

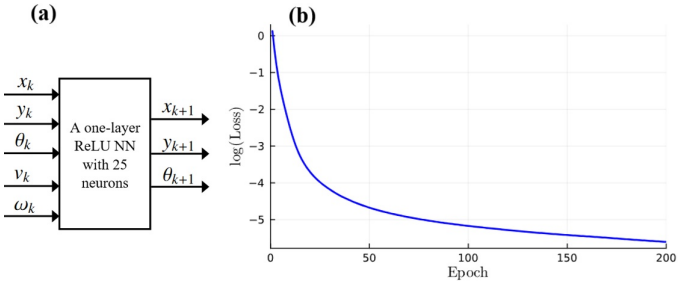

Figure 11: (a) Structure of the ReLU NN employed to learn the vehicle dynamics. (b) Training loss plotted on a logarithmic scale.

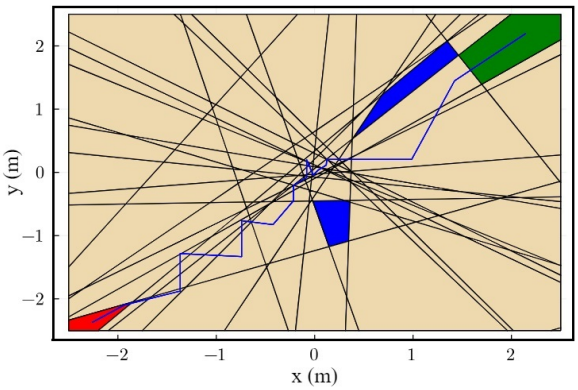

Figure 12: Polyhedral partition of the vehicle state space. Red, green, and blue denote the initial cell, target cell, and obstacles, respectively. The blue line shows the trajectory generated by BCO.

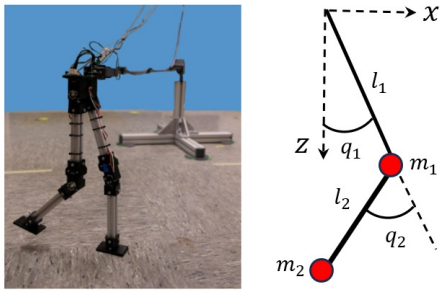

Figure 13: Experimental setup of a small-scale bipedal robot (left) and the simplified dynamic model of one leg as a double pendulum (right).

6.2. Dynamic Identification and Control of Double-Pendulum

In this example, we aim to model and identify the dynamics of one leg of a bipedal robot, as shown in Fig. 13. Each leg is represented as a planar double pendulum with two revolute joints. The physical parameters of each leg are: link lengths $L_1 = 0.2$,m and $L_2 = 0.172$,m, with point masses $m_1 = 0.211$,kg and $m_2 = 0.141$,kg.

To collect data from each joint of one leg of the robot, chirp torque inputs are applied, with amplitudes of 0.075 Nm and 0.028 Nm, and maximum frequencies of 3 Hz and 2.5 Hz for the two joints, respectively. Joint angular positions (q_1, q_2) are obtained from encoder measurements, while joint velocities $(\dot{q}_1, \dot{q}_2)$ and accelerations $(\ddot{q}_1, \ddot{q}_2)$ are computed using Euler discretization and subsequently smoothed with a moving-average filter. The end-effector position of the double-pendulum can be

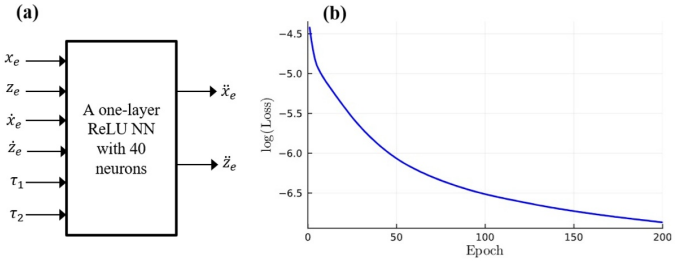

Figure 14: (a) Architecture of the ReLU NN used to model the double-pendulum dynamics. (b) Log-scaled loss curve during training.

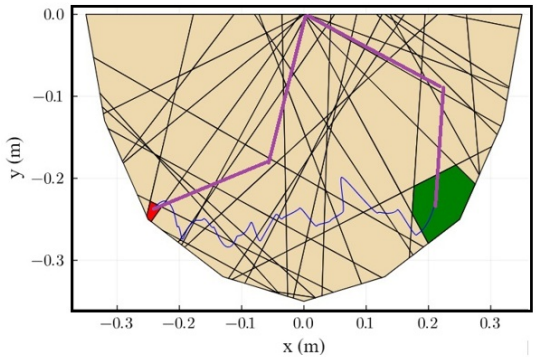

Figure 15: Polyhedral partition of the double-pendulum end-effector space. The red and green regions denote the initial and target cells, and the blue line shows the trajectory obtained using BCO.

specified by:

$$\begin{cases} x_e = L_1 \sin(q_1) + L_2 \sin(q_1 + q_2) \\ z_e = L_1 \cos(q_1) + L_2 \cos(q_1 + q_2) \end{cases} \tag{54}$$

The end-effector velocity $(\dot{x}_e, \dot{z}_e)$ and acceleration $(\ddot{x}_e, \ddot{z}_e)$ are computed using Euler discretization and subsequently denoised via the moving-average filter. Therefore, the dynamic equation of the pendulum in Cartesian space can be formulated as:

$$\ddot{\Upsilon} = f(\Upsilon, \dot{\Upsilon}, \tau) \tag{55}$$

where $\Upsilon = [x_e, z_e]$ and $\dot{\Upsilon} = [\dot{x}_e, \dot{z}_e]$. Since, in practice, the nonlinear function $f(\Upsilon, \dot{\Upsilon}, \tau)$ includes unknown dynamic uncertainties and friction effects, a single-hidden-layer NN with 40 neurons was trained using a dataset of 8000 samples (Fig. 14(a)). The training convergence is shown in Fig. 14(b).

The learned model induces a polyhedral decomposition of the end-effector state space, enabling path planning through the BCO framework. Starting from the initial region (red) and progressing toward the target region (green), the tube-MPC controller guarantees safe transitions between neighboring cells. The resulting trajectory in Fig. 15 shows precise path following and smooth regulation despite the system's nonlinear behavior.

Table 2 reports the tracking error statistics. The small angular deviations for both joints demonstrate that the proposed hybrid planning-and-control framework maintains high tracking accuracy, validating the effectiveness of the learned PWL model.

Table 2: Error Statistics for double-pendulum.

Error Statistic	Max. Error	Variance	Mean Error
q_1	7.34°	0.043°	0.20°
q_2	5.75°	0.021°	0.15°

6.3. Polyhedral Tube-MPC Robustness on an Inverted Pendulum

The final example examines the robustness of the proposed polyhedral tube-MPC on an inverted pendulum robot governed by:

$$\ddot{\theta} = \frac{g}{l} \sin(\theta) + \frac{1}{ml^2} u + w, \tag{56}$$

where θ (rad) is the pendulum angle, u (N·m) the control torque, $g = 9.8\,\text{m/s}^2$, $l = 0.172\,\text{m}$, $m = 0.141\,\text{kg}$, and $|w| \leq w_{\max}$ represents bounded disturbance. The viscous friction coefficient is $k = 0.1$, and the sampling time is $\Delta t = 0.001\,\text{s}$. A feed-forward ReLU neural network with three inputs, $[\theta, \dot{\theta}, u]^T$, one output, $\ddot{\theta}$, and a single hidden layer of 16 neurons (Fig. 16(a)), was trained to approximate the nonlinear pendulum dynamics described in (56). The model was trained using 1000 uniformly sampled data points from the domain $[-\pi/2, , \pi/2] \times [-2.5, , 2.5] \times [-1, , 1]$, and the logarithmic training loss is shown in Fig. 16(b).

The pendulum was commanded to swing from $\theta(0) = 0°$ to $\theta_f = 45°$ under disturbance magnitudes $w_{\max} \in 0.01, 0.05, 0.10$. As shown in Fig. 17, the tube-MPC design ensured constraint satisfaction and bounded deviations between nominal and disturbed trajectories across all disturbance levels. The phase portraits in Fig. 18 further demonstrate that the disturbed trajectories consistently remain within their corresponding polyhedral tubes.

Table 3 reports the angular tracking error statistics for different tube sizes and disturbance levels in the inverted pendulum. As expected, the mean, variance, and maximum errors increase with both the tube size and the disturbance intensity. Specifically, for a small tube under low disturbance, the mean error is 0.35,rad, variance 0.04,rad^2, and maximum error 0.82,rad. For a medium tube under medium disturbance, the mean, variance, and maximum errors increase to 1.07,rad, 0.26,rad^2, and 2.45,rad, respectively. Under a large tube with high disturbance, the errors further increase to a mean of 2.84,rad, variance of 0.88,rad^2, and a maximum of 5.33,rad. Importantly, all observed errors remain within the designed tube bounds, confirming that the conservatively chosen tube radii—set as 1.5× the maximum learned-model error—are sufficient to guarantee robust invariance and safe tracking.

Together, these results validate the ability of the ReLU-based model and polyhedral tube-MPC framework to ensure robust motion execution despite model approximation errors and external disturbances.

7. Conclusion

This paper introduced a novel data-driven framework for synthesizing robust and optimal controllers for a broad class of unknown nonlinear discrete-time systems. The proposed

Table 3: Angular Error Statistics for Inverted Pendulum.

Tube Size	Disturbance Level	$\overline{\Delta\theta}$	Var($\Delta\theta$)	$\Delta\theta_{\max}$
Small	Low	0.35	0.04	0.82
Medium	Medium	1.07	0.26	2.45
Large	High	2.84	0.88	5.33

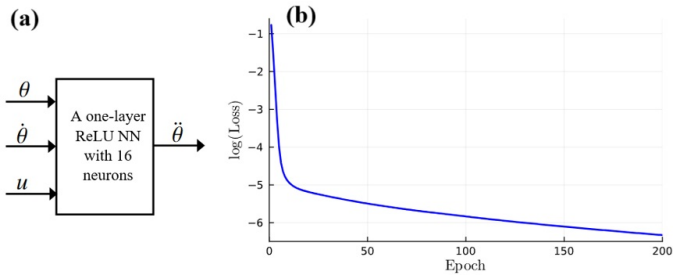

Figure 16: (a) ReLU NN configuration for identifying the dynamics of the inverted pendulum. (b) Logarithmic plot of the training loss.

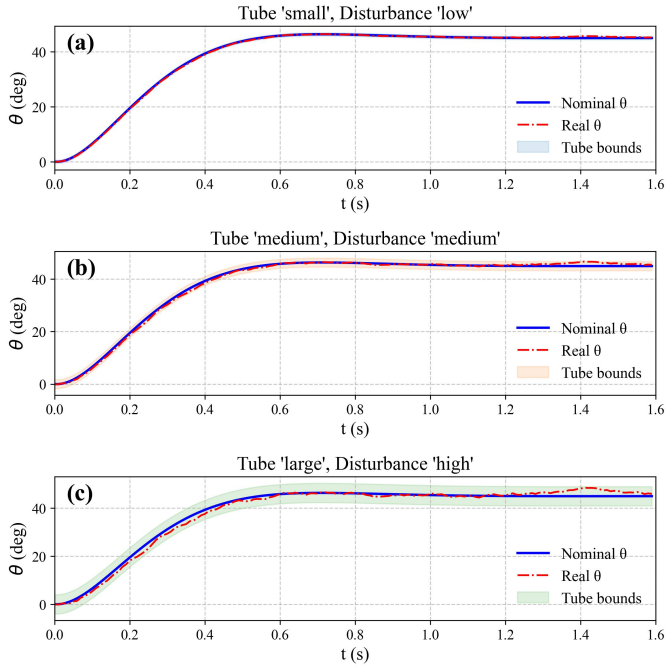

Figure 17: Time evolution of θ under three disturbance levels. Blue: nominal trajectory; red: real trajectory; shaded area: tube bounds.

method partitions the state space into polyhedral cells using ReLU NNs, allowing unknown nonlinear dynamics to be represented as PWL systems. An ε-approximate relation between the real system and its abstracted model was established, enabling efficient and formally grounded computational modeling. A BCO-based safe path planning algorithm was developed to identify optimal transition sequences between initial and target cells while minimizing the number of transitions within the abstraction, thereby reducing computational complexity. For transitions between polyhedral regions, a hybrid control strategy was designed that integrates tube-MPC for nominal dynamics with a minimax feedback controller for error compensation. Furthermore, a weighted automaton independent of the system dynamics was constructed to represent the overall abstraction. The effectiveness of the proposed framework was validated through case studies, including an autonomous vehicle path-planning benchmark, a double-pendulum robotic system, and an inverted pendulum. Future work will extend this methodology to higher-dimensional systems and implement the controller on a ten-degree-of-freedom bipedal robot with unknown dynamics to further demonstrate its practical applicability.

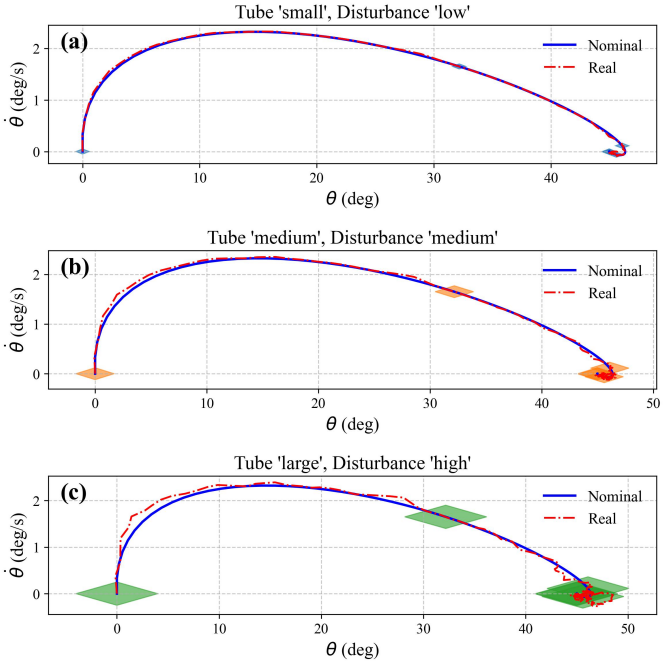

Figure 18: Phase portraits of θ and $\dot{\theta}$ for varying tube sizes. Blue: nominal, red: real, polygons: tube boundaries.

Acknowledgments

RJ is an FNRS honorary Research Associate. This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under grant agreement No. 864017 – L2C, from the Horizon Europe programme under grant agreement No. 101177842 – Unimaas, and from the ARC (French Community of Belgium), project name: SIDDARTA.

Declaration of Interests

The authors declare that they have no conflict of interest.

Appendix A. Finding Essential Constraints

To check whether constraint i is necessary or redundant, we create a new set of constraints by excluding the i^{th} constraint as:

$$\mathcal{A} = \begin{bmatrix} \boldsymbol{a}_1 & \dots & \boldsymbol{a}_{i-1} & \boldsymbol{a}_{i+1} & \dots & a_m \end{bmatrix}^T \tag{A.1a}$$

$$\boldsymbol{\mu} = \begin{bmatrix} \mu_1 & \dots & \mu_{i-1} & \mu_{i+1} & \dots & \mu_m \end{bmatrix}^T \tag{A.1b}$$

and proceed to solve the following LP problem:

$$\begin{aligned} \max_{x} \quad & \boldsymbol{a}_i x \\ \text{s.t.} \quad & \mathcal{A}x \leq \boldsymbol{\mu} \end{aligned} \tag{A.2}$$

An inequality $\boldsymbol{a}_i x \leq \mu_i$ within (A.2) is considered redundant if removing it from the system does not alter the solution set of (A.2). Equivalently, there is no x that satisfies $\boldsymbol{a}_i x > \mu_i$ while simultaneously meeting $\boldsymbol{a}_j x \leq \mu_j$ for all $j \neq i$.

References

[1] P. Tabuada, Verification and Control of Hybrid Systems: A Symbolic Approach, Springer, US, 2009. doi:https://doi.org/10.1007/978-1-4419-0224-5.

[2] J. Calbert, L. N. Egidio, R. M. Jungers, Smart abstraction based on iterative cover and non-uniform cells, IEEE Control Syst. Lett. 8 (2024) 2301–2306. doi:10.1109/LCSYS.2024.3409106.

[3] M. Rungger, M. Zamani, Compositional construction of approximate abstractions of interconnected control systems, IEEE Trans. Control Netw. Syst. 5 (1) (2018) 116–127. doi:10.1109/TCNS.2016.2583063.

[4] M. Zamani, P. M. Esfahani, R. Majumdar, A. Abate, J. Lygeros, Symbolic control of stochastic systems via approximately bisimilar finite abstractions, IEEE Trans. Autom. Control 59 (12) (2014) 3135–3150. doi:10.1109/TAC.2014.2351652.

[5] K. Mallik, A. K. Schmuck, S. Soudjani, R. Majumdar, Compositional synthesis of finite-state abstractions, IEEE Trans. Autom. Control 64 (6) (2019) 2629–2636. doi:10.1109/TAC.2018.2869740.

[6] A. Borri, C. Possieri, Reinforcement learning for non-deterministic transition systems with an application to symbolic control, IEEE Control Syst. Lett. 7 (2023) 1610–1615. doi:10.1109/LCSYS.2023.3252823.

[7] D. Ajeleye, A. Lavaei, M. Zamani, Data-driven controller synthesis via finite abstractions with formal guarantees, IEEE Control Syst. Lett. 7 (2023) 3453–3458. doi:10.1109/LCSYS.2023.3331385.

[8] G. Pola, T. Masciulli, E. D. Santis, M. D. D. Benedetto, Data-driven controller synthesis for abstract systems with regular language specifications, Automatica 134 (2021) 109903. doi:https://doi.org/10.1016/j.automatica.2021.109903.

[9] K. Hashimoto, A. Saoud, M. Kishida, T. Ushio, D. V. Dimarogonas, Learning-based symbolic abstractions for nonlinear control systems, Automatica 146 (2022) 110646. doi:https://doi.org/10.1016/j.automatica.2022.110646.

[10] J. Jiang, Y. Zhao, S. Coogan, Safe learning for uncertainty-aware planning via interval mdp abstraction, IEEE Control Syst. Lett. 6 (2022) 2641–2646. doi:10.1109/LCSYS.2022.3173993.

[11] V. Debauche, A. Edwards, R. M. Jungers, A. Abate, Formal synthesis of lyapunov stability certificates for linear switched systems using relu neural networks, in: Proc. Int. Conf. Neuro-symb. Syst., Vol. 288, PMLR, 2025, pp. 352–364.
URL https://proceedings.mlr.press/v288/debauche25a.html

[12] Y. Farid, A. Ramezani, A wavelet-based robust adaptive t-s fuzzy controller design for synchronization of faulty chaotic gyrostat systems, J. Control Autom. Electr. Syst. 32 (2021) 57–69. doi:https://doi.org/10.1007/s40313-020-00647-z.

[13] M. Everett, G. Habibi, C. Sun, J. P. How, Reachability analysis of neural feedback loops, IEEE Access 9 (2021) 163938–163953. doi:10.1109/ACCESS.2021.3133370.

[14] E. Manino, I. Bessa, L. C. Cordeiro, Towards global neural network abstractions with locally-exact reconstruction, Neural Networks 165 (2023) 344–357. doi:https://doi.org/10.1016/j.neunet.2023.06.002.

[15] A. Abate, A. Edwards, M. Giacobbe, Neural abstractions, in: Proc. 36th Conf. Neural Inf. Process. Syst. (NeurIPS 2022), 2022.

[16] R. Majumdar, M. Salamati, S. Soudjani, Neural abstraction-based controller synthesis and deployment, ACM Trans. Embed. Comput. Syst. 22 (5s) (2023) 1–25. doi:https://doi.org/10.1145/3608104.

[17] A. Edwards, M. Giacobbe, A. Abate, On the trade-off between efficiency and precision of neural abstraction, in: Quantitative Evaluation of Systems, Springer Nature Switzerland, 2023, pp. 152–171. doi:https://doi.org/10.1007/978-3-031-43835-6_12.

[18] C. Katrakazas, M. Quddus, W. H. Chen, L. Deka, Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions, Transp. Res. Part C Emerg. Technol. 60 (2015) 416–442. doi:https://doi.org/10.1016/j.trc.2015.09.011.

[19] N. L. Prasad, B. Ramkumar, 3-d deployment and trajectory planning for relay based uav assisted cooperative communication for emergency scenarios using dijkstra's algorithm, IEEE Trans. Veh. Technol. 72 (4) (2023) 5049–5063. doi:10.1109/TVT.2022.3224304.

[20] G. Dong, F. Yang, K.-L. Tsui, C. Zou, Active balancing of lithium-ion batteries using graph theory and a-star search algorithm, IEEE Trans. Ind. Informat. 17 (4) (2021) 2587–2599. doi:10.1109/TII.2020.2997828.

[21] O. Salzman, D. Halperin, Asymptotically near-optimal rrt for fast, high-quality motion planning, IEEE Trans. Robot. 32 (3) (2016) 473–483. doi:10.1109/TRO.2016.2539377.

[22] H. Zheng, L. Li, F. Xu, F. Sun, M. Ding, Evolutionary route planner for unmanned air vehicles, IEEE Trans. Robot. 21 (4) (2005) 609–620. doi:10.1109/TRO.2005.844684.

[23] J. Ji, A. Khajepour, W. W. Melek, Y. Huang, Path planning and tracking for vehicle collision avoidance based on model predictive control with multi-constraints, IEEE

Trans. Veh. Technol. 66 (2) (2017) 952–964. doi:10.1109/TVT.2016.2555853.

[24] J. Wang, W. Chi, C. Li, C. Wang, M. Q.-H. Meng, Neural RRT*: Learning-based optimal path planning, IEEE Trans. Autom. Sci. Eng. 17 (4) (2023) 1748–1758. doi:10.1109/TASE.2020.2976560.

[25] G. P. Kontoudis, K. G. Vamvoudakis, Kinodynamic motion planning with continuous-time q-learning: An online, model-free, and safe navigation framework, IEEE Trans. Neural Netw. Learn. Syst. 30 (12) (2019) 3803–3817. doi:10.1109/TNNLS.2019.2899311.

[26] M. Z. Zgurovsky, A. A. Pavlov, Combinatorial Optimization Problems in Planning and Decision Making: Theory and Applications, Vol. 173 of Studies in Systems, Decision and Control, Springer, Cham, 2019. doi:https://doi.org/10.1007/978-3-319-98977-8.

[27] L. Shi, S. Xu, Uav path planning with qos constraint in device-to-device 5g networks using particle swarm optimization, IEEE Access 626 (2020) 137884–137896. doi:10.1109/ACCESS.2020.3010281.

[28] H. Mavalizadeh, O. Homaee, R. Dashti, J. M. Guerrero, H. H. Alhelou, P. Siano, Robust switch selection in radial distribution systems using combinatorial optimization, CSEE J. Power Energy Syst. 8 (3) (2020) 933–940. doi:10.17775/CSEEJPES.2020.05660.

[29] B. Legat, P. Tabuada, R. M. Jungers, Sum-of-squares methods for controlled invariant sets with applications to model-predictive control, Nonlinear Anal. Hybrid Syst. 36 (2020) 100858. doi:https://doi.org/10.1016/j.nahs.2020.100858.

[30] S. V. Raković, The implicit rigid tube model predictive control, Automatica 157 (2023) 111234. doi:https://doi.org/10.1016/j.automatica.2023.111234.

[31] S. V. Raković, L. Dai, Y. Xia, Homothetic tube model predictive control for nonlinear systems, IEEE Trans. Autom. Control 68 (8) (2023) 4554–4569. doi:10.1109/TAC.2022.3207415.

[32] K. Zhang, Y. Shi, Adaptive model predictive control for a class of constrained linear systems with parametric uncertainties, Automatica 117 (2020) 108974. doi:https://doi.org/10.1016/j.automatica.2020.108974.

[33] A. Nikou, D. V. Dimarogonas, Decentralized tube-based model predictive control of uncertain nonlinear multiagent systems, Int. J. Robust Nonlinear Control 29 (10) (2019) 2799–2818. doi:https://doi.org/10.1002/rnc.4522.

[34] Y. Song, et al., Generalized model and deep reinforcement learning-based evolutionary method for multitype satellite observation scheduling, IEEE Trans. Syst., Man, Cybernetics: Syst. 54 (4) (2024) 2576–2589. doi:10.1109/TSMC.2023.3345928.

[35] K. G. Vamvoudakis, Non-zero-sum nash q-learning for unknown deterministic continuous-time linear systems, Automatica 61 (2015) 274–281. doi:https://doi.org/10.1016/j.automatica.2015.08.017.

[36] D. Wang, J. Ren, M. Ha, Discounted linear q-learning control with novel tracking cost and its stability, Information Sciences 626 (2023) 339–353. doi:https://doi.org/10.1016/j.ins.2023.01.030.

[37] M. Mersha, K. Lam, J. Wood, A. K. AlShami, J. Kalita, Explainable artificial intelligence: A survey of needs, techniques, applications, and future direction, Neurocomputing 599 (2024) 128111. doi:https://doi.org/10.1016/j.neucom.2024.128111.

[38] H. J. van Waarde, M. K. Camlibel, M. Mesbahi, From noisy data to feedback controllers: Nonconservative design via a matrix s-lemma, IEEE Trans. Autom. Control 67 (1) (2022) 162–175. doi:10.1109/TAC.2020.3047577.

[39] N. Dinh, M. A. Goberna, M. A. Lopez, From linear to convex systems: consistency, farkas' lemma and applications, J. Convex Anal. 13 (1) (2006) 113–133. doi:http://hdl.handle.net/10045/8805.

[40] Y. Farid, N. Bigdeli, Robust adaptive intelligent sliding model control for a class of uncertain chaotic systems with unknown time-delay, Nonlinear Dyn. 67 (2012) 2225—-2240. doi:https://doi.org/10.1007/s11071-011-0141-0.

[41] G. F. Montúfar, R. Pascanu, K. Cho, Y. Bengio, On the number of linear regions of deep neural networks, in: Adv. Neural Inf. Process. Syst., Vol. 27, 2014, pp. 2924–2932.
URL https://api.semanticscholar.org/CorpusID:5941770

[42] T. Serra, C. Tjandraatmadja, S. Ramalingam, Bounding and counting linear regions of deep neural networks, in: Proc. Int. Conf. Mach. Learn., Vol. 80, 2018, pp. 4558–4566.
URL https://api.semanticscholar.org/CorpusID:34019680

[43] F. Zhao, K. You, Minimax q-learning control for linear systems using the wasserstein metric, Automatica 149 (2023) 110850. doi:https://doi.org/10.1016/j.automatica.2022.110850.

[44] R. Ke, J. Tang, Z. Zuo, Y. Shi, Koopman-based robust model predictive control with online identification for nonlinear dynamical systems, IEEE/CAA J. Autom. Sin. 12 (9) (2025) 1947–1949. doi:10.1109/JAS.2025.125546.

[45] G. Reissig, A. Weber, M. Rungger, Feedback refinement relations for the synthesis of symbolic controllers, IEEE Trans. Autom. Control 62 (2015) 1781–1796. doi:10.1109/TAC.2016.2593947.