UNIVERSITÀ DELLA CALABRIA

DEPARTMENT OF COMPUTER SCIENCE, MODELING, ELECTRONICS AND SYSTEMS ENGINEERING (DIMES)

PhD program in *Information and Communication Technologies (ICT)*

Model Predictive Control and Reinforcement Learning: a unifying approach

Francesco Giannini

Francesco Giannini 24.01.2025 16:37:25 GMT+02:00

Supervisor: **Professor Giuseppe Franzé**

GIUSEPPE FRANZE' 24.01.2025 15:43:07 GMT+01:00

Co-supervisor: **Professor Francesco Pupo**

FRANCESCO PUPO 25.01.2025 16:32:49 GMT+02:00

PhD Coordinator: **Professor Giancarlo Fortino**

GIANCARLO FORTINO 26.01.2025 23:15:17 GMT+02:00

ACADEMIC YEAR 2024/25

"Un homme s'engage dans sa vie,

dessine sa figure,

et en dehors de cette figure il n'y a rien".

(Jean-Paul Sartre - L'existentialisme est un humanisme - Gallimard, 1946.)

Dédié à ceux qui s'engagent dans l'amélioration constante de cette figure.

" In his life, a man commits himself,

shapes his own figure,

and there is nothing but that figure".

Dedicated to those who are committed to the constant improvement of that figure.

Contents

Abstract

The aim of this thesis is to propose new insights in the field of Model Predictive Control (MPC) combined with Reinforcement Learning (RL) algorithms, in the following research contexts:

- *Set-theoretic Receding Horizon Control for unknown dynamical systems.* The proposed methodology is based on three elements: Linear Time-Invariant system behaviour; Data-Driven modelling, and Reinforcement Learning technicalities. These are appropriately combined in order to derive an accurate outer convex approximation of the nonlinear evolution of an unknown system. The effectiveness of the obtained uncertain polytopic model has been tested using a probabilistic approach.

- *Platoons on Urban Road Networks.* Here, the main goals are to alleviate traffic congestion and reduce the CO_2 emissions of autonomous vehicles. The core of this part of the project is the formal definition of new strategies, comprehensive of routing algorithms and distributed control systems within a theoretical formulation. To efficiently evaluate the performance of the proposed control architecture, a co-simulation between SUMO (Simulation of Urban MObility) and MATLAB has been implemented.

- *An intelligent multi-layer control architecture for logistics operations of autonomous vehicles in manufacturing systems.* Similar to the case of autonomous vehicles on Urban Road Networks, Reinforcement Learning and Model Predictive Control are combined to solve the problem of task scheduling, routing and control.

- *Autonomous vehicle platoons subject to cyber-attacks.* The core of this problem is how to detect and to deal with false data injections in the field of intelligent transportation systems.

Abstract in Italiano

Lo scopo di questa tesi è proporre nuove strategie di controllo predittivo, combinato con algoritmi di apprendimento per rinforzo, nei seguenti contesti di ricerca:

- *Controllo set-theoretic ad orizzonte recedente per sistemi dinamici ignoti.* La metodologia proposta si basa su tre elementi: comportamento lineare e tempo-invariante di sistemi, utilizzo di dati e apprendimento per rinforzo. L'obiettivo è derivare un'accurata approssimazione convessa esterna dell'evoluzione non lineare di sistemi ignoti. L'efficacia del modello incerto politopico ottenuto è stata testata attraverso un approccio probabilistico.

- *Plotoni su reti stradali urbane.* Gli obiettivi principali consistono nell'alleviare la congestione del traffico e nel ridurre le emissioni di CO_2 di veicoli urbani. La definizione formale di nuove strategie, comprensive di algoritmi di routing e sistemi di controllo distribuiti, costituisce il nucleo di questa parte della tesi. Per valutare in modo efficiente le prestazioni dell'architettura di controllo proposta è stata implementata una procedura di co-simulazione tra SUMO (Simulation of Urban MObility) e MATLAB.

- *Una architettura intelligente di controllo multi-layer per operazioni logistiche di veicoli in sistemi di produzione.* Similmente al caso dei veicoli autonomi su reti stradali urbane, apprendimento per rinforzo e controllo predittivo sono combinati per risolvere i problemi di scheduling, routing e controllo.

- *Plotoni di veicoli autonomi soggetti a cyber attacchi.* Il rilevamento e la gestione dell'iniezione di dati falsi nel contesto dei sistemi di trasporto intelligenti rappresentano il focus dell'ultima parte della tesi.

List of Symbols

$\mathbb{Z}_+$	Set of non-negative integer numbers
$\mathbb{R}$	Set of real numbers
$\mathbb{C}$	Set of complex numbers
$t \in \mathbb{Z}_+$	(Discrete) time
T_s	Sampling time
$x(t) \in \mathbb{R}^n$	State of plant at time t
$u(t) \in \mathbb{R}^m$	Control input at time t
$y(t) \in \mathbb{R}^{n_y}$	Measurement vector at time t
$[A, B; C, D]$	Dynamical evolution matrices of a linear plant
$\not{f}$	State-evolution function of a non-linear dynamical system
I_n	Identity matrix of size n
0_r	Column vector of r zero entries
$(0, 1)$	Open interval of all the real numbers between 0 and 1
$\{0, 1\}$	Set composed of two elements: 0 and 1
$\rho \in \Lambda \subseteq \mathbb{R}^L$	Unknown parameters vector in a polytopic description belonging to the unitary simplex
Ω	Polytopic set with vertices $[A_j, B_j; C_j, D_j], j = 1, \dots L$
$Co\{\cdot\}$	Convex hull operator
$\mathcal{U} \subseteq \mathbb{R}^m$	Constraint set on $u(\cdot)$
$\mathcal{X} \subseteq \mathbb{R}^n$	Constraint set on $x(\cdot)$
u_{max}	Maximum admissible norm of $u(\cdot)$
x_{max}	Maximum admissible norm of $x(\cdot)$
$\Xi \subseteq \mathcal{X}$	A RPI set, also indicated as $\mathcal{T}_0$
$\mathcal{E}$	An ellipsoid, defined by its center $c_\mathcal{E}$ and shaping matrix $P_\mathcal{E}$
$K \in \mathbb{R}^{m \times n}$	State-feedback gain
$Q_\mathcal{E}, Y_\mathcal{E}, \gamma_\mathcal{E}$	LMI variables used to compute $\mathcal{E}$
$\mathcal{T}^T$	Controllability set to $\mathcal{T}$ in time $T < \infty$

$Pre(\mathcal{T})$	Predecessor set of $\mathcal{T}$, denoted as $\mathcal{T}^1$
$\{\mathcal{T}_k\}_{k=1}^{H}$	Family of H predecessor sets, $\mathcal{T}_k := Pre(\mathcal{T}_{k-1})$, with $\mathcal{T}_0$ RPI
$In[\mathcal{T}_k] =: \mathcal{E}_k$	Inner ellipsoidal approximation of a set $\mathcal{T}_k$
h	MPC prediction horizon
$f(\cdot)$	A function
∇	Nabla operator
$\mathcal{C}^1$	Class of continuously differentiable functions
$\bar{x}$	Equilibrium point corresponding to $\bar{u}$
ξ	A generic point belonging to the state space
$\Delta x(t) := x(t) - x(t-1)$	One-step state difference
$Proj_x$	Projection on x operator
$\#(\cdot)$	Cardinality of a set
DoA	Domain of attraction of an algorithm
$\tilde{\mathcal{E}}_i \in \mathbb{R}^{n+m}$	Ellipsoidal set in the extended space (x, u)
$E[\cdot]$	Expected value operator
$rand(\cdot)$	Function extracting a random element from a set
ν	A random number. If it is known, the set $\mathcal{A}$ from where it is extracted: $\nu = rand(\mathcal{A})$
νG	Pseudo-random number generator unit
σ	Standard deviation of a random variable
β	A scalar
$\theta \in \mathbb{R}^{W_\theta}$	Neural network weights
$\alpha \in (0, 1)$	Learning rate
$\mathcal{B}(x, l)$	Ball centered in the point x and with radius l
$\inf(\Xi)$	Infimum of a set Ξ, i.e., the greatest lower bound of the set
$\mathcal{L}_x \in \mathbb{R}^+$	Lipschitz constant
$\otimes$	Concatenation operator between sequences
$Type[\cdot]$	Operator that, given an expression, returns its type
$s(\cdot) \in \mathcal{S}$	Reinforcement learning state
$a(\cdot) \in \mathcal{A}$	Action
$r(\cdot)$	Reward function
$\hat{r}(\cdot)$	Expected reward, $\hat{r}(t) := E[r(t)]$

$Pr(Event_1 \mid Event_2)$ Probability of $Event_1$ given $Event_2$

$env : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ Probabilistic transition function describing the environment

$\mathcal{M}$ A Markov Decision Process (MDP)

G_r Return function

$\gamma \in (0, 1)$ Discount factor

$V(s(\cdot))$ State value function

$\mathcal{Q}(s(\cdot), a(\cdot))$ State-action value function, known as Q-function

$\mathcal{Q}^i(s(\cdot), a(\cdot))$ Q-function of the i-th agent

$\mathcal{Q}(s(\cdot), a(\cdot); \theta)$ Q-function, when it is defined as a neural network

π Reinforcement learning policy

a^* Greedy action

$\epsilon_\pi(\cdot) \in (0, 1)$ Probability function

$\epsilon_\delta > 0$ A given threshold scalar

$n_a(t) \sim \mathcal{N}(E[n_a(t)], \sigma_a(t)^2)$ Gaussian noise, described by its expected value and standard deviation

$\delta_\pi(\cdot)$ Temporal difference error

$\Gamma := \left\{ \{u^d(t), x^d(t)\}_{t=0}^{N_s} \right\}_{d=1}^{M}$, A family of input-output data sequences

$U_{0,1,T}$ Input data in the time interval [0 T]

$X_{0,T}$ State data in the time interval [0 T]

$X_{1,T}$ State data in the time interval [1 T]

$\dagger$ Penrose pseudo-inverse operator

$[\mathscr{A}\ \mathscr{B}]$ Linear description arising from data

$\mathcal{T}(\Omega)$ State trajectory tube of a polytopic model

$\mathscr{P}$ Probability measure

$\mathcal{X}_\mu$ Set of μ independent identically distributed (i.i.d.) samples from $\mathcal{X}$

$1 - \circ$ Confidence level

$th \in \mathbb{R}_+$ Threshold on the desired confidence level

$\mathcal{U}_{prec}$ Stored applied inputs

ψ_u Heuristic function, part of the RL framework in Chapter 3

Q_e, P_e, E_e Matrices used in the proof of the embedding algorithm of Chapter 3

η Tolerance on the inclusion of the non-linear evolution into the actual trajectories tube

$W(z)$ A transfer function

υ Singular values vector

$\mathcal{P}(\cdot)$ Polyhedral operator

Ω_{exact} Exact linear polytopic embedding of a given unknown non-linear system

e_r^k Relative error index introduce to evaluate Ω w.r.t. Ω_{exact}

l, $\mathfrak{z}$, CS_i, cs_i, l_i, $\mathfrak{r}$, $\mathcal{K}_i$ Parameters of the four tanks model, described in Table 3.2

$AV = \{AV_i\}_{i=1}^{N}$ Set of autonomous vehicles

$LF^k = AV^k = \{AV_i^k\}_{i=1}^{N_k}$ $k-$th platoon of autonomous vehicles, $k = 1, \dots q$, with length N_k

URN Urban Road Network, composed of M links

$\#^j(t) \in \mathbb{Z}_+$ Number of vehicles on the link j

$\bar{w}^j$ and l^j Average width and length of the link j

DZ^j Decision zone associated with the link j

O_{free}^j Obstacle-free regions on link j

$\mathcal{O}_t$ Time-varying obstacle occurrence at time t

R_{vis} Detection radius

R_{min}^c Minimum curvature radius of a vehicle

x_f Assigned target

RG Reference generator unit

$z^i(\cdot)$ Reference signal for the i-th vehicle

t_{dec} Decision time, when AV^i enters the DZ

$\Psi(\cdot)$ Global reward

$\psi_{goal}(\cdot)$, $\psi_{wt}(\cdot)$, $WT(\cdot)$, $f_{loops}(\cdot)$ Heuristics used to compute the reward in Chapter 4

p^i, v^i, p_{max}^i, v_{max}^i x-y position, x-y velocity of AV^i and relative constraints

$\mathcal{X}_p^i$, $\mathcal{X}_v^i$ Constraint sets on x-y position and velocity

x_{in}^i and x_{fin}^i Initial and final positions of the i-th agent

$(\mathcal{X}_c^i)^j$ State constraint set relative to AV^i on edge j

$(\mathcal{T}_0^i)_k^j \subset \mathbb{R}^n$, $(\bar{x}^1)_k^j$, $(K^i)_k^j$ k-th PI region of AV^i, its center, and corresponding feedback gain

J, J^* Cost and optimal cost

$R_x^i > 0$ and $R_u^i \geq 0$ Symmetric state and input weighting matrices, respectively

h_i, $i = 1, \dots N$ Prediction horizon length of the i-th agent

$\mathbf{x}(t) := \{x(t+k \mid t)\}_{k=1}^h$ Predicted state trajectory

$\mathbf{u}(t) := \{u(t+k \mid t)\}_{k=1}^h$ Predicted control inputs

$\mathbf{x}^*(t) := \{x^*(t+k \mid t)\}_{k=1}^h$ Optimal state trajectory

$\hat{\mathbf{x}}(t) := \{x^*(t+k \mid t)\}_{k=1}^h$ Assumed state trajectory

d_{min}, d_{max} Platoon constraints

FMS Flexible manufacturing system

$Ma = \{Ma^k\}_{k=1}^{\beth}$ Set of machines in a FMS

L/U Load/unload station

$\mathcal{J}^k$ A job, i.e., an ordered sequence of operations $op_i, i = 1, \dots, n_k$

PN Petri net

$TCPN$ Timed colored Petri net

Π Set of places in a PN

Φ Set of transitions in a PN

$Arcs, Post, Pre, CS, Var, PC, Gu, AE, Ini$ Elements of a TCPN

CO^j $j-$th aisle in a FMS

$\mathcal{H}$, g Variables of proper dimensions used to define hyperplanes

$\mathcal{H}^j$, g^j Hyperplane relative to the DZ on the edge j

$(\mathcal{H}^j)_k$, g_k^j $k-$th hyperplane relative to the CO^j

$\daleth$ Number of hyperplanes defining CO^j

OF and JA Functions used in computing the reward r_{TCPN}

FDI False Data Injection

$\tilde{\Xi}$ Corrupted target set $\tilde{\Xi} \leftarrow \Xi + \Xi_a$

$\tilde{\mathbf{x}}$	Corrupted predicted state trajectory $\tilde{\mathbf{x}} \leftarrow \hat{\mathbf{x}} + \mathbf{x_a}$
$\mathcal{I}^{i-1}(t)$	Information set transmitted by AV_{i-1} to AV_i at time t
$\tilde{\mathcal{I}}^{i-1}(t)$	Information set received by AV_i at time t
$\mathcal{D}^i$	Digital twin model of the agent AV_i
$\hat{\mathbf{x}}^{\mathcal{D}^i}(t)$	State predictions generated by means of $\mathcal{D}^i$
$\hat{\mathcal{X}}_i^k(t)$	k-step ahead state predictions polyhedral sets compatible with the constraints
$\mathbf{D}_i(t) \in 0, 1$	Attack detector output
k_o	Maximum time steps to wait for a guaranteed attack-free communication

List of Figures

Introduction

The aim of this thesis is to propose new insights in the field of Model Predictive Control (MPC) [1] combined with Reinforcement Learning (RL) [2] algorithms.

Model Predictive Control was first proposed in the 60's [3][4] and it has been used in industries since the 80's [5]. Many approaches to MPC have been used over the years [6], *Chapter* 1 of this thesis provides a brief introduction about the set-theoretic one [7][8]. Today the cutting-edge advancements in MPC [9][10] lie in its integration with machine learning (ML). *Chapter* 2 outlines essential Reinforcement Learning (RL) [2][11] algorithms that support the analytical framework of this research. Such algorithms make use of Neural Networks, therefore some essential ideas about them are introduced in the Appendix *A*.

The best way to combine ML and MPC is the extensive use of data of the paradigm named Data-driven MPC [12][13]. While a robust theoretical framework exists for linear systems, based on the "fundamental lemma" [14], controlling nonlinear unknown systems by means of data trajectories is still an open research paradigm. *Chapter* 3 of this thesis proposes new insights in this field of studies. There, three methodologies - linear time-invariant system behavior, data-driven modelling and RL - have been used to derive an accurate outer convex approximation of the non linear evolution of an unknown system.

In *Chapter 4*, the main focus is on Autonomous Vehicles (AVs) platoons on Urban Road Networks (URNs). This area of research is today of great interest because of the increasing number of vehicles world-wide and the corresponding need for innovative solutions to deal with road traffic issues [15][16]. The core of the proposal given here is the formal definition of new strategies, comprehensive of routing algorithms and distributed control systems within a theoretical formulation. To efficiently evaluate the performance of the proposed control architecture, a co-simulation between SUMO (Simulation of Urban MObility) and MATLAB has been implemented.

The control architecture proposed in *Chapter 4* can be applied to many different research contexts. Among these is the field of Autonomous Vehicles in smart manufacturing systems. In that context, the challenge is to optimize the production efficiency by involving automation to enhance logistics issues [17].

The last *Chapter* of the thesis refers to the case of Autonomous vehicles platoons on smart road environments subject to cyber-attacks. This, because the use of open and unreliable communication platforms has enormously increased the chance that data shared among AVs could be tampered with by malicious agents [18]. Therefore, in this context, cyber-security appears to be relevant in terms of attack detection capabilities and adequate resilient actions [19][20].

Finally, I underline that - while *Chapter* 1, and 2 resume precedent established results - *Chapter* 3 and those following represent the core of the present proposal. They adopt the same structure: (1) an introduction, comprehensive of the context of interest where the topic is presented, a specific literature review and a summary of the main contribution made; (2) problem formulation; (3) proposed solution; (4) numerical results.

Chapter 1

Set-theoretic Model Predictive Control

MPC [1] is a philosophy to control dynamical system based on the use of state predictions. In order to compute these predictions, in the classical formulation of MPC, a mathematical model of the system is required. As an example, given a dynamical system described by

$$\begin{cases} x(t+1) = \not{f}(x(t), u(t)) \\ y(t) = Cx(t) \end{cases} \tag{1.1}$$

- where $x(t) \in \mathbb{R}^n$ is the state of plant at time t, $u(t) \in \mathbb{R}^m$ the control input, $y(t) \in \mathbb{R}^{n_y}$ the measurement vector, $\not{f} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ the non-linear state-evolution function, $C \in \mathbb{R}^n \times \mathbb{R}^{n_y}$ the output matrix -, MPC can be formulated as a finite horizon open-loop optimal control problem, solved at each time instant, using the current state of the plant as the initial state. This optimization yields an optimal control sequence and only the first control action in this sequence is applied to the plant [1][21][22]:

$$\mathbf{u}^*(x(t)) = \arg\min_{\mathbf{u}} J(\mathbf{x}(t), \mathbf{u}(\mathbf{t})) \tag{1.2}$$

s.t.

$$
\begin{aligned}
&x(t|t) = x(t) \\
&x(t+k+1|t) = \not{f}(x(t+k|t), u(t+k|t)) \\
&x(t+k|t) \in \mathcal{X} \\
&u(t+k|t) \in \mathcal{U}
\end{aligned}
$$

where $\mathbf{x}(t) = \{x(t+k|t)\}_{k=0}^{h}$ predicted state of the system, $\mathbf{u}(t) = \{u(t+k|t)\}_{k=0}^{h}$ predicted inputs, $J(\mathbf{x}(t), \mathbf{u}(\mathbf{t}))$ a cost index to be defined, $\mathcal{X}$ and $\mathcal{U}$ constraints sets on the state and the input. These could be described in an ellipsoidal set-membership fashion:

$$
\begin{aligned}
&u(t) \in \mathcal{U} := \{u \in \mathbb{R}^m \mid u^T u \leq u_{max}^2\}, \\
&x(t) \in \mathcal{X} := \{x \in \mathbb{R}^n \mid x^T x \leq x_{max}^2\} \quad \forall t \geq 0,
\end{aligned}
\tag{1.3}
$$

with $\mathcal{U}$, $\mathcal{X}$ convex and compact subsets of $\mathbb{R}^m$ and $\mathbb{R}^n$, respectively.

Figure 1.1 shows the logic behind MPC and the fundamental ingredients: current time t, when the computations are performed; *prediction horizon length* h (the future horizon within computing the predictions of the state); sampling time T_s; *predicted inputs* and *predicted outputs*; the signal to be tracked (named reference); the applied control move (called *current input*).

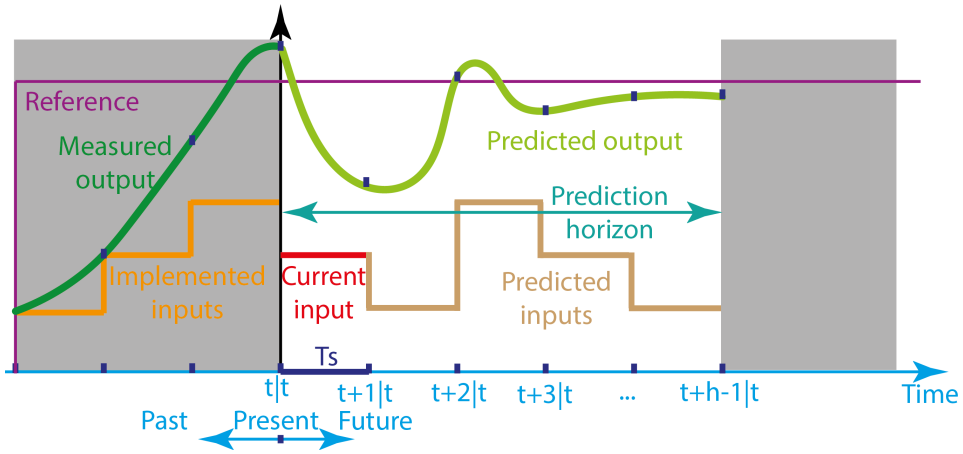

Figure 1.1: Example of model predictive control

The effectiveness of MPC to explicitly lead with state and input hard constraints and plant uncertainty has been proved [23]. The price that must be paid for these benefits is its computational demands [24]. In order to solve this drawback, set-theoretic ideas [7][8][25] have been introduced into the MPC framework. The main goal is to reduce the computational load by moving most of the computations in an off-line phase aiming at

synthesizing a convex inner approximation of the feasible state-space with feasibility and stability guarantees on the possible trajectories of the plant. In what follows, a set-theoretic approach to MPC is introduced.

1.1 Set-theoretic approach in MPC

The aim is to compute a law $u(\cdot) = g(x(\cdot))$ with stability guarantees to control the plant under fulfillment of the set-membership constraints 1.3 without solving any sampling time the optimization problem 1.2.

The main idea is to find a way to steer the state of the plant to a positively invariant region:

Definition 1 *[7]. A set $\Xi \subseteq \mathcal{X}$ is said Positively Invariant (PI) for (1.1) under (1.3) if there exists a control law $u(t) := g(x(t)) \in \mathcal{U}$ such that $\forall\, x(t) \in \Xi$ one has*

$$\not{f}(x(t), u(t)) \in \Xi$$

□

If a state $x(0) \in \Xi$ is controlled by the above control law $u(\cdot) = g(x(\cdot))$, the evolution $x(t) \in \Xi$, $\forall t \in \mathbb{Z}_+$.

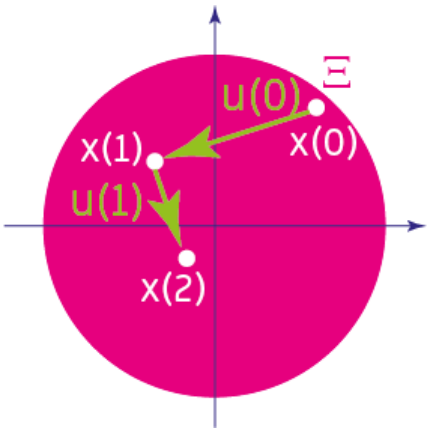

Figure 1.2: A simple representation of the dynamical evolution of a system in a RPI region.

Figure 1.2 provides a simple representation of the dynamical evolution of a system when the initial state belongs to the PI region Ξ, under the action of the corresponding state-feedback control law.

Notice that if a disturbance is included in the dynamic evolution of the system (1.1), i.e., $x(t+1) = \not{f}(x(t), u(t), d(t))$, where $d(t) \in \mathscr{D}$ and $\mathscr{D}$ is a bounded set, the definition of PI region can be extended to each possible realization of $d(t)$. In such a case the region is called Robustly Positively Invariant (RPI).

The Domain of Attraction (DoA) of this Receding Horizon Control (RHC) algorithm is given by the set $\Xi = \mathcal{T}_0$. In order to increase the DoA, some definitions are required.

Definition 2 *[7] Given a set $\mathcal{T} \subseteq \mathcal{X}$, the controllability set $\mathcal{T}^T$ to $\mathcal{T}$ in time $T < \infty$ is the set of all vectors x for which there exists a control law $u(\cdot) \in \mathcal{U}$ such that if $x(0) = x$ then $x(T) \in \mathcal{T}$ under the action of $u(t)$, $t = 0, \dots T$.* □

A special case is the predecessor set:

Definition 3 *Given the set $\mathcal{T} \subseteq \mathcal{X}$, the predecessor set $Pre(\mathcal{T})$, is the set of states for which there exists a causal control $u(t) \in \mathcal{U}$ such that the one-step state evolution belongs to $\mathcal{T}$:*

$$Pre(\mathcal{T}) := \{x \in \mathcal{X} : \exists u \in \mathcal{U} : \not{f}(x(t), u(t)) \in \mathcal{T}\} \tag{1.4}$$

□

Notice that $Pre(\mathcal{T}) = \mathcal{T}^1$. Then, let $\mathcal{E}$ be the target set, it is possible to determine the sets of states i-step controllable to $\mathcal{E}$ via the following recursion:

$$\mathcal{T}_0 := \mathcal{E}, \quad \mathcal{T}_i := Pre(\mathcal{T}_{i-1}) \tag{1.5}$$

given a RPI set $\mathcal{T}_0$, the core of the strategy is to compute a family of sets $\mathcal{T}_i$, $i = 1, \dots H$ such that it is easy to determine an admissible control strategy capable to steer in finite time any initial state $x(0) \in \bigcup_{i=0}^{H} \mathcal{T}_i$ to the terminal (target) set $\mathcal{T}_0$. Such a problem is stated as follows:

MPC Problem - Given the system (1.6)-(1.7), a sequence of regions $\{\mathcal{T}_i\}_{i=0}^{H}$ and an initial state $x(0) \in \bigcup_{i=0}^{H} \mathcal{T}_i$, compute at each time instant t and on the basis of the current state $x(t)$ a control strategy , compatible with (1.3), such that there exists a finite time instant $\bar{t} \geq 0$ so that $x(\bar{t}) \in \mathcal{T}_0$ is achieved and a convenient performance index is minimized. □

1.2 Ellipsoidal off-line MPC approach

In order to solve the introduced **MPC problem** in a simple computational way, few hypothesis are introduced:

- the dynamical system is described by a discrete-time multi-model linear time-invariant (LTI) model:

$$\begin{aligned} x(t+1) &= A(\rho(t))x(t) + B(\rho(t))u(t) \\ y(t) &= Cx(t) \end{aligned} \tag{1.6}$$

  where $t \in \mathbb{Z}_+$, $x(t) \in \mathbb{R}^n$ is the state, $u(t) \in \mathbb{R}^m$ the control input, $\rho(t) = [\rho_1(t), \ldots, \rho_L(t)]$ an unknown parameter and $y(t) \in \mathbb{R}^{n_y}$ the measurement vector. The system evolution matrices are such that

$$[A(\rho(t)), B(\rho(t))] := \sum_{j=1}^{L} \rho_j(t)[A_j, B_j] \quad \text{with} \quad \rho \in \Lambda := \left\{ \rho \in \mathbb{R}^L : \sum_{j=1}^{L} \rho_j = 1,\ \rho_j \geq 0 \right\} \tag{1.7}$$

  (A_j, B_j) and ρ_j therefore are, respectively, the polytope vertices and coordinates. Notice that $\Omega := Co\{[A_1 \ \ B_1] \ldots [A_L \ \ B_L]\}$ is the set of all system pairs $(A(t), B(t)), \forall t$, with $Co\{\cdot\}$ denoting the convex hull operator.

- The sets are ellipsoids[26], therefore the definition is required:

  Definition 4 *[27]. An ellipsoid $\mathcal{E} \in \mathbb{R}^n$ is described as*

$$\mathcal{E} := \{x \in \mathbb{R}^n \,|(x - c_\mathcal{E})' P_\mathcal{E} (x - c_\mathcal{E}) \leq 1\},$$

  where $c_\mathcal{E}$ is the center and $P_\mathcal{E} = P_\mathcal{E}' > 0$ the shaping matrix. □

  ***Remark* 1** *The same ellipsoid $\mathcal{E}$ can be described by means of the support function*

$$sup_{x \in \mathcal{E}} l'x = l'c_\mathcal{E} + \sqrt{l' P_\mathcal{E}^{-1} l}$$

  where sup denotes the least upper bound of the set. □

- $\mathcal{U}$ can be expressed as intersection of (possibly degenerate) ellipsoidal sets - $\mathcal{U} = \bigcap_i \mathcal{E}_i^\mathcal{U}$

Under these hypothesis, the terminal set $\mathcal{E} = \mathcal{T}_0$ and the stabilizing local control law $u(\cdot) = g(x(\cdot))$ defined as a pair $(K, \mathcal{E})$ with $\mathcal{E} \subset \mathbb{R}^n$ an ellipsoidal set and $K \in \mathbb{R}^{m \times n}$ the feedback gain $u(t) = Kx(t)$ satisfying

$$(A_j + B_j K)\mathcal{E} \subset \mathcal{E} \subseteq \mathcal{X}, \quad \forall j = 1, \ldots, L,$$

can be computed by solving the following linear matrix inequality (LMI) optimization [23]:

$$\min_{Q_{\mathcal{E}}, Y_{\mathcal{E}}, \gamma_{\mathcal{E}}} \gamma_{\mathcal{E}} \tag{1.8}$$

subject to

$$\begin{bmatrix} 1 & x(t)^T \\ x(t) & Q_{\mathcal{E}} \end{bmatrix} \geq 0\,, \tag{1.9}$$

$$\begin{bmatrix} Q_{\mathcal{E}} & Q_{\mathcal{E}} A_j' + Y_{\mathcal{E}}' B_j' & Q_{\mathcal{E}} R_x^{1/2} & Y_{\mathcal{E}}' R_u^{1/2} \\ A_j Q_{\mathcal{E}} + B_j Y_{\mathcal{E}} & Q_{\mathcal{E}} & 0 & 0 \\ R_x^{1/2} Q_{\mathcal{E}} & 0 & \gamma_{\mathcal{E}} I_n & 0 \\ R_u^{1/2} Y_{\mathcal{E}} & 0 & 0 & \gamma_{\mathcal{E}} I_m \end{bmatrix} \geq 0 \tag{1.10}$$

$$\begin{bmatrix} Q_{\mathcal{E}} & Q_{\mathcal{E}} A_j' + Y_{\mathcal{E}}' B_j' \\ A_j Q_{\mathcal{E}} + B_j Y_{\mathcal{E}} & x_{max}^2 I_n \end{bmatrix} \geq 0, \quad \forall j = 1, \ldots, L, \tag{1.11}$$

$$\begin{bmatrix} u_{max}^2 I_n & Y_{\mathcal{E}} \\ Y_{\mathcal{E}}' & Q_{\mathcal{E}} \end{bmatrix} \geq 0, \tag{1.12}$$

Imposing

$$K = Y_{\mathcal{E}} Q_{\mathcal{E}}^{-1} \qquad \text{and} \qquad \mathcal{E} := \{x \in \mathbb{R}^n \mid x^T P_{\mathcal{E}} x \leq 1\}\,, \quad P_{\mathcal{E}} = \gamma_{\mathcal{E}} Q_{\mathcal{E}}^{-1} \tag{1.13}$$

the RPI region for the closed-loop state evolution

$$x(t+1) = (A(\rho(t)) + B(\rho(t))K)x(t)\,, \quad \forall \rho(t) \in \Lambda,$$

complies with the prescribed constraints: $\mathcal{E} \subset \mathcal{X}$ and $K\mathcal{E} \subset \mathcal{U}$.

As the set sequence $\{\mathcal{T}_i\}_{i=1}^H$ is concerned, the recursion (1.5) can be implemented via LMIs that lead to inner ellipsoidal approximations of $\{\mathcal{T}_i\}_{i=1}^N$. Notice that

$$\begin{aligned} \mathcal{T}_{i+1} &= Pre\,(\mathcal{T}_i) = \{x \in \mathcal{X} \mid \exists u \in \mathcal{U} \,:\, \forall \rho \in \Lambda, A(\rho)x + B(\rho)u \in \mathcal{T}_i\} \\ &= \{x \in \mathcal{X} \mid \exists u \in \mathcal{U} \,:\, \forall j = 1, \ldots, L, A_j x + B_j u \in \mathcal{T}_i\} \\ &\supset \{x \in \mathcal{X} \mid \exists u \in \mathcal{U} \,:\, \forall j = 1, \ldots, L, A_j x + B_j u \in In[\mathcal{T}_i]\} \\ &= \mathrm{Proj}_x \left\{ [x\ u] \mid x \in \mathcal{X}\,,\, u \in \mathcal{U} \text{ and } \forall j = 1, \ldots, L, [x\ u] \in \tilde{\mathcal{E}}_i^j \right\} \end{aligned}$$

where $In[\cdot]$ is the inner ellipsoidal approximation operator [27], $Proj_x$ is the projection on x operator and $\tilde{\mathcal{E}}_i$ is the ellipsoidal set defined in the extended space (x, u). Then

$$\begin{aligned} &\mathcal{T}_{i+1} = \{x \in \mathcal{X} \mid \exists u \in \mathcal{U} \,:\, \forall \rho \in \Lambda, A(\rho)x + B(\rho)u \in \mathcal{T}_i\} \\ &\supset \mathrm{Proj}_x \left[In \left[\left(\bigcap_{j=1}^{L} \tilde{\mathcal{E}}_{i-1} \right) \cap \bigcap_i (\mathbb{R}^n \times \mathcal{E}_i^{\mathcal{U}}) \right] \right] =: \mathcal{E}_{i+1} \end{aligned} \tag{1.14}$$

Chapter 2

Reinforcement Learning

The focus of this thesis is to apply Machine Learning schemes to advanced MPC applications in a unified framework. The aim of this *Chapter* is to introduce some fundamentals [28][29] used in this work. A definition is required:

Definition 5 *[11] An agent - figure 2.1 - is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators.* □

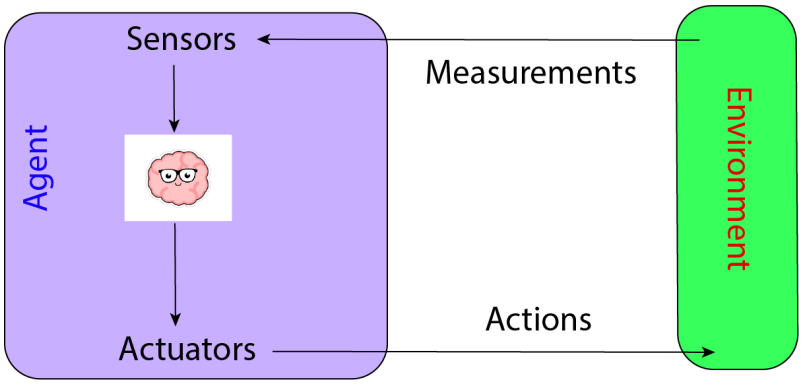

Figure 2.1: Simple representation of an intelligent agent

In order to evaluate the success or failure of an agent's behavior, a numerical criterion of success of its actions is used. In this sense, an agent is said to be *rational* if it tries by its actions to maximize the expected value of the performance criterion. Clearly, to be rational doesn't imply to be omniscient - because measurements may not supply all relevant

information about the environment - or clairvoyant - because action outcomes may not be as expected-. Anyway, a rational agent is learning if it improves its performance after making observations about the world.

2.1 Fundamentals

Definition 6 *[2] Reinforcement Learning (RL) is simultaneously a problem, a class of solution methods that work well on the class of problems, and the field that studies these problems and their solution methods.* □

As illustrated in see figure 2.2, the agent and the environment interact in a closed-loop fashion without any direct instructions about how to act. The most important ingredients of RL are:

- the **state space** $\mathcal{S}$ that contains the states s, describing the actual situation of the environment;
- the **action space** $\mathcal{A}$ containing the admissible actions;
- the **reward function** $r(\cdot)$ evaluating the performances of the agent.

Note that both state and action sets can be continuous or discrete.

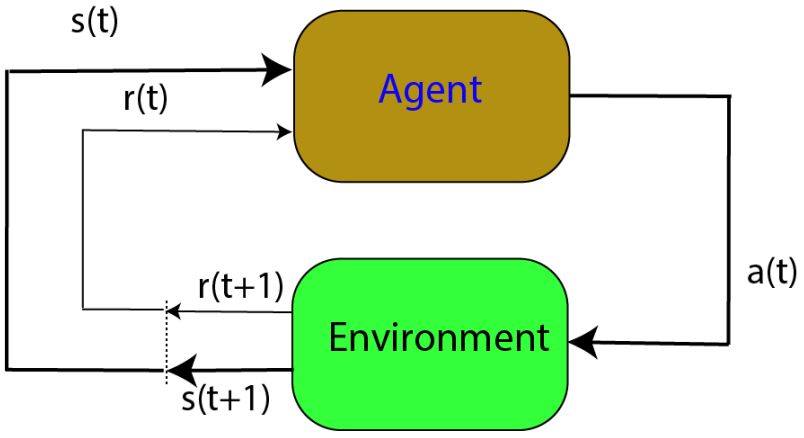

Figure 2.2: RL framework

Figure 2.2 shows how the agent interacts with the environment by means of the actions and receives the current state and the performance criterion.

2.1.1 The environment

Regarding the environment, it can be characterized by accomplishment of some properties. First of all, it is fully observable if the measurement allows the complete reconstruction of the state. Otherwise, it is said to be not fully observable. Notice that, in this introduction to RL, the measurement are assumed to comprehend all the state because the focus is on decision making. Another important property regards the dynamical evolution of the environment, described as a state evolution function, unknown to the agent. Such a function can be stochastic if the next state is not fully determined by past states and actions. Usually the sequence of previous states and actions is called "run". The only hypothesis about the environment is the markovian transition model:

$$Pr(s(t+1)|a(t), s(t), a(t-1), s(t-1), \ldots, a(0), s(0)) = Pr(s(t+1)|a(t), s(t)) \tag{2.1}$$

where $Pr(Event_1|Event_2)$ denote the probability of $Event_1$ given $Event_2$. The last equation can be rewritten by considering that the action $a(t)$ is not known:

$$Pr(s(t+1)|s(t), a(t-1), s(t-1), \ldots, a(0), s(0)) = \sum_{a \in \mathcal{A}} Pr(s(t+1), a|s(t)) \tag{2.2}$$

Under this hypotesis, the process can be represented as a markov decision one:

Definition 7 *A Markov Decision Process (MDP) is a tuple* $\mathcal{M} = \{\mathcal{S}, s(0), \mathcal{A}, env(\cdot,\cdot,\cdot)\}$ *where* $s(0)$ *is the initial state and* $env : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0,1]$ *is a probabilistic transition function describing the dynamic evolution of the environment.*

2.1.2 Main elements

In order to solve a RL problem, some elements need to be introduced:

- the **return** G_r is the weighted accumulated future rewards, starting from the current state $s(t)$:
$$G_r(t) := \sum_{k=0}^{\infty} \gamma^k r(t+k+1) \tag{2.3}$$
the scalar $\gamma \in (0,1)$ is called *discount factor*;

- the **value function** V is the total amount of discounted reward an agent can *expect* to accumulate over the future. There are two main paradigms:

– **state-action value function**:

$$\mathcal{Q}(s(t), a(t)) := E\left[\sum_{k=0}^{\infty} \gamma^k r(t+k+1) | s(t), a(t)\right] \tag{2.4}$$

where $E[\cdot]$ denotes the expected value operator. Usually the state-action value function is called *Q-function*;

– **state value function**:

$$V(s(t)) := E\left[\sum_{k=0}^{\infty} \gamma^k r(t+k+1) | s(t)\right] \tag{2.5}$$

- both the introduced value functions depend on the **policy**:

$$\pi : \mathcal{S} \to \mathcal{A} \tag{2.6}$$

This function determines the agent behaviour given the actual state $s(t)$. It is the core of the RL framework, to be determined (and trained) to solve the learning problem.

The policy can be defined in terms of finding the *greedy action* - usually indicated as a^* - in a given state, i.e., the action that maximize the value function in such a state. This attempt of maximization largely depends on the knowledge of the agent about the environment. There are many strategies to improve it. The ones used in this thesis work are:

- $\epsilon-$**greedy policy** [30]:

$$a(t) = \begin{cases} rand(\mathcal{A}) & \text{with probability } \epsilon_\pi(t) \\ a^* & \text{otherwise} \end{cases} \tag{2.7}$$

where $\epsilon_\pi(t) \in (0, 1)$ is a monotonically decreasing function of time. The reason behind this choice is the hypotheses that the knowledge about the environment increases over time.

- **additive action noise** [31] :

$$a(t) = a^*(t) + n_a(t) \tag{2.8}$$

The quantity $n_a(t)$ is the noise, extracted from a probability distribution, for example:

$$n_a(t) \sim \mathcal{N}(0, \sigma_a(t)^2) \tag{2.9}$$

where $\mathcal{N}$ denotes a normal distribution with zero mean s_0 and $\sigma_a(t)$ standard deviation. Again, $\sigma_a(t)$ is a monotonically decreasing function of time.

Literature refers to the trade-off between applying the greedy action and improving the knowledge as *dichotomy between exploitation of existing knowledge and exploration of uncharted territory* [32].

Starting form the value function it is possible to compute the expected value of the reward at the next-time instant:

Theorem 1 *Given the value function $\mathcal{Q}(s,a)$, the actual state $s(t)$ and the applied action $a(t)$, the expected reward is given by*

$$\hat{r}(t+1) := E[r(t+1)] = \mathcal{Q}(s(t),a(t)) - \gamma\mathcal{Q}(E[s(t+1)],\pi(E[s(t+1)])) \tag{2.10}$$

Proof

$$\begin{aligned}
\mathcal{Q}(s(t),a(t)) &:= E\left[\sum_{k=0}^{\infty}\gamma^k r(t+k+1)\right] \\
&= E[r(t+1) + \sum_{k=1}^{\infty}\gamma^k r(t+k+1)] \\
&= \hat{r}(t+1) + E\left[\sum_{k=0}^{\infty}\gamma^{k+1} r(t+k+2)\right] \\
&= \hat{r}(t+1) + \gamma E\left[\sum_{k=0}^{\infty}\gamma^{k} r(t+k+2)\right] \\
&= \hat{r}(t+1) + \gamma\mathcal{Q}(s(t+1),a(t+1))
\end{aligned}$$

□

Notice that if the state value function is given instead of the state-action value function, the following formula holds:

$$\hat{r}(t+1) = V(s(t)) - \gamma V(E[s(t+1)]) \tag{2.11}$$

When the agent receives, from the environment, $r(t+1)$ and $s(t+1)$ the quantity

$$\delta_\pi(t) := r(t+1) + \gamma\mathcal{Q}(s(t+1),\pi(s(t+1))) - \mathcal{Q}(s(t),a(t)) \tag{2.12}$$

called *temporal difference error*, assumes a crucial role in the learning process. Therefore, the level of the knowledge about the environment of the agent can be evaluated by means of $\delta_\pi(\cdot)$. This quantity is evaluated several time during the learning process which is composed of *episodes.* An episode is defined as the time from an initial state to a terminal state:

- The initial state is described by means of a probability distribution, for example:

$$s(0) \sim \mathcal{N}(s_0, \sigma_s^2) \tag{2.13}$$

- The final instant depends on the satisfaction of a criterion that evaluate a state as terminal. Examples of terminal state are the instant when the desired state is reached or the instant when it is not anymore reachable.

The process that re-initialize the environment for the next episode is called *reset.*

2.2 Reinforcement Learning algorithms

In literature, there are many algorithms to deal with RL problems. Mostly they can be grouped as:

- **policy gradient** [33][34] that directly differentiate the value function;
- **value based** estimating the value function of the optimal policy, like Q-learning [35][36] or State-Action-Reward-State-Action (SARSA) [37][38] ;
- **actor critic methods** [39][40] estimating both a policy and a value function;
- **model-based** [41][42] using the dynamic evolution model of the environment.

In what follows, two paradigms are presented: Q-learning and actor critic architecture. The first class of methods is suitable to discrete action space, while the second one to the continuous case.

2.2.1 Deep Q-learning

Q-learning [35][36] is a RL approach used to find an optimal action-selection for a given MDP. It is defined by the policy:

$$\pi(s) = arg\max_{a \in \mathcal{A}} \mathcal{Q}(s, a) \tag{2.14}$$

Figure 2.3 represents the training of a Q-learning agent. The main idea is to repeatedly go from the initial state $s(0)$ to the terminal state $s(t_{end})$ - that can be a good state or not - while controlling the evolution of the system by means of an $\epsilon-$greedy policy.
It has been proved that by updating $\mathcal{Q}$ as

$$\mathcal{Q}(s(t), a(t)) \leftarrow \mathcal{Q}(s(t), a(t)) + \alpha(r(t+1) + \gamma \max_{a \in \mathcal{A}} \mathcal{Q}(s(t+1), a) - \mathcal{Q}(s(t), a(t))) \tag{2.15}$$

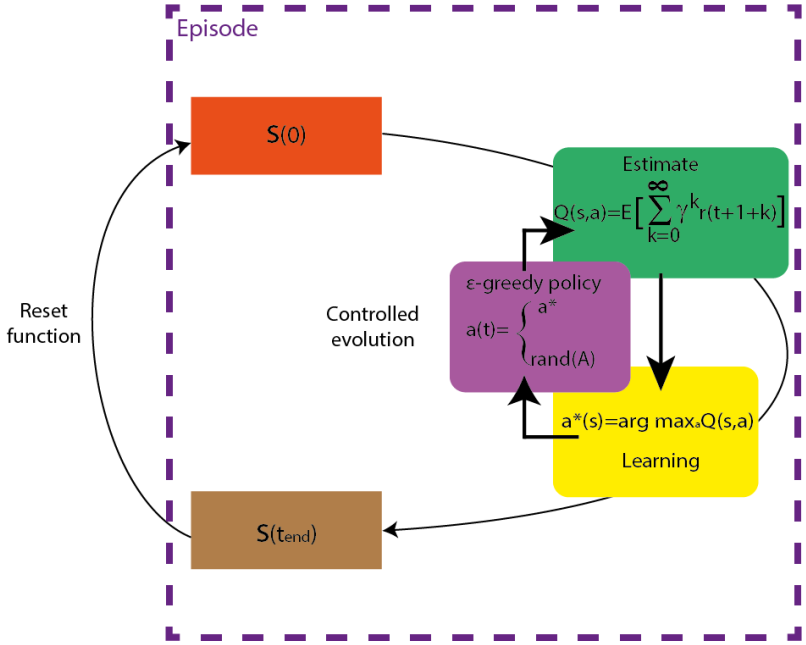

Figure 2.3: Q learning loop

where $\alpha \in (0,1)$ the learning rate, the approximation will converges to the return, but it may visit possible state-action pairs many times [43].

All the above developments allow to write down the following computable algorithm.

Q learning -Algorithm

Initialization

1: **Initialize** $\mathcal{Q}(s,a)$;

For each episode

1: **Initialize** $s(0)$ as in (2.13)

For each time step

1: **Select** an admissible action, by means of an $\epsilon-$greedy policy:

$$\pi : a(t) = \begin{cases} rand(\mathcal{A}) & \text{with probability } \epsilon_\pi(t) \\ arg\max_{a\in\mathcal{A}} \mathcal{Q}(s(t),a) & \text{otherwise} \end{cases} \tag{2.16}$$

2: **Apply** the chosen action $a(t)$;

3: **Observe** the next state of the environment $s(t+1)$;

4: **Receive** the reward $r(t+1)$

5: **Update** $\mathcal{Q}(s,a)$ as in (2.15);

6: **Decrease** $\epsilon_\pi(t)$

7: **Until** $s(t+1)$ is terminal.

8: **Until** it is assumed that the temporal difference error $\delta_\pi(t)$ defined in (2.12) is definitely such that $|\delta_\pi(t)| < \epsilon_\delta$ where $\epsilon_\delta > 0$ is a given threshold scalar.

Usually the best way to structure the Q-function is by using a neural network (see Appendix 1) as approximation function. In such a case, Q-learning is called *Deep Q learning* - to underline the presence of the network - and the Q-function depends on the neural networks weights θ: $\mathcal{Q}(s,a;\theta)$. The aim of the learning process is to find θ^* :

$$\theta^* := arg\min_\theta |G_r(s,a) - \mathcal{Q}(s,a;\theta)|, \quad \forall (s,a) \in \mathcal{S} \times \mathcal{A} \tag{2.17}$$

where $G_r(s,a)$, as in (2.3), is the (unknown) return, therefore this formula cannot be straightforwardly used. In order to find a good approximation of θ^*, the training process continues to update the Q-function until the temporal difference error $\delta_\pi(t)$ defined in (2.12) converges.

2.2.2 Actor-Critic approach

Actor-Critic [39][40] is an RL paradigm where the agent is designed by means of two main parts - as depicted in figure 2.4-:

- the **Actor** directly implements the RL policy (2.6), i.e. , it is in charge to generate the greedy action a^* given the actual knowledge about the environment and the actual state $s(t)$;
- the **Critic** is the value function - (2.4) or (2.5) - therefore its role consists in evaluating the policy provided by the **Actor**.

The scheme of figure 2.4 shows this paradigm in case of an additive action noise on the greedy action as in formula (2.8) to explore the environment.
All the above developments allow to write down the following computable algorithm.

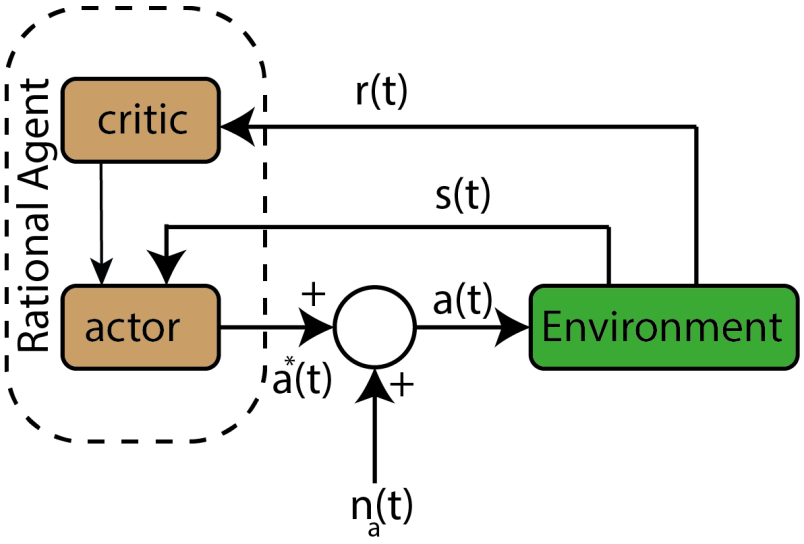

Figure 2.4: Actor-critic scheme

Actor Critic -Algorithm

Initialization

1: **Initialize** $\mathcal{Q}(s,a)$ and $\pi(s)$

For each episode

1: **Initialize** $s(0)$ as in (2.13)

For each time step

1: **Select** an action, by means of the actor:

$$a(t) = \pi(s(t)) \tag{2.18}$$

2: **Perturb** the chosen action as in (2.8) in order to explore the state space;

3: **Apply** the action;

4: **Observe** the next state of the environment $s(t+1)$;

5: **Receive** the reward $r(t+1)$

6: **Update** $Q(s,a)$ as in (2.15);

7: **Update** the policy $\pi(s)$;

8: **Until** $s(t+1)$ is terminal.

9: **Until** it is assumed that the temporal difference error δ_t defined in (2.12) is definitely

such that $|\delta_t| < \epsilon_\delta$ where $\epsilon_\delta > 0$ is a given threshold scalar.

Updating the policy The policy is a function depending on some parameters θ_π :

$$\pi = f(s; \theta_\pi) \tag{2.19}$$

This is a general assumption that holds true also when the actor is defined as a neural network. A classical way [39] to update the policy is by gradient descent [44]:

$$\theta_\pi = \theta_\pi + \alpha_\pi \nabla_{\theta_\pi} \mathcal{Q} \tag{2.20}$$

where $\alpha_\pi \in (0, 1)$ is a small enough learning rate for the actor.

Chapter 3

Set-theoretic Receding Horizon Control for nonlinear systems: a data-driven approach

This *Chapter* is based on the work presented in [45] and [46]. Here, the problem of generating multi-model state space descriptions in a data-driven context to embed the dynamic behavior of nonlinear systems is addressed. The proposed methodology takes advantage of three methodologies: linear time-invariant system behavior, data-driven modeling and reinforcement learning technicalities. These elements are properly combined to develop a data-driven algorithm capable to derive an accurate outer convex approximation of the nonlinear evolution of an unknown system. At each iteration, the resulting algorithm computes a polytopic model and evaluates its effectiveness using a probabilistic approach.

As the main merits of the proposed approach are concerned, the following aspect clearly stands up: the development of an interdisciplinary methodology that takes advantage of system theory, probabilistic arguments and reinforcement learning capabilities giving rise to an harmonized architecture in charge to deal with a vast class of nonlinear systems.

Finally, the proposed approach is tested by resorting to benchmark examples that allow to quantify the level of accuracy of the computed polytopic outer approximations.

3.1 Context of interest

In recent years, the acquirement, processing, and analysis of "big data" has become a relevant topic in science and engineering, also reviving the interest of using data-driven techniques for control engineering purposes [47, 48]. As a consequence, learning from data has assumed a key role in statistics and artificial intelligence areas, as well as in the control theory domain [49], [50], [51], [52]. On the other hand, it is worth to recall that the most sophisticated control strategies, such as adaptive control [53], model predictive control [54, 21], sliding-mode control [55] and so on, require a mathematical model of the system. Although this is an advantage for the controller design theory, it represents a drawback when model-based approaches are applied in real contexts because the mathematical model is not always available, especially in complex systems. For example, most of the processes arising in several industrial domains (chemistry, metallurgy, machinery, power electronics, transportation to cite a few) show increasing complex behaviors (unmodeled dynamics, large disturbances, uncertainty sources) that make difficult or even impossible the use of standard model identification techniques, often restricted to linear and deterministic systems, see [56]. Even if exact mathematical models were available, they may be not applicable for controller design purposes because, due to high orders and strong nonlinearities, the resulting control architectures could be not suitable for practical implementations.

Based on this analysis, an idea, often investigated in past years, to overcome such difficulties relies on the derivation of proper state trajectory embeddings such that the behavior of the nonlinear plant is included into that of an uncertain linear time invariant (LTI) system [27]. Although appealing, this *modus operandi* crashes with the intrinsic complexity of process dynamics that require an effective handling of uncertainties and nonlinearities. Therefore, the adoption of data-driven approaches [57] appears the more suitable way to pursue in this context. As a matter of fact, the development of data-driven techniques is based on the exploitation of "big data", either collected and stored from the process history or generated by a "digital twin" [58]. Therefore, the need of *ad-hoc* machine learning algorithms [11] becomes fundamental to efficiently address computational complexity issues, see e.g. [59].

3.1.1 Literature review

The derivation of outer convex approximations of the state trajectories pertaining to specific classes of nonlinear systems is a relevant topic collecting many contributions in the last three decades. The discussion about the technical literature can be divided into two main categories: model-based approaches and data-driven ones. The so-called model-based schemes have a long history starting from the seminal paper [60] and as testified by the numerous contributions concerning the use of linear time-varying uncertain system descriptions, see [61]-[62] and references therein. Although these methods have successfully applied in different contexts, they suffer of an unavoidable methodological/computational drawback concerning with the derivation of mathematical models by means of first principles [63]. Conversely, the use of data-driven simulation approaches [64] to describe the dynamics of nonlinear systems via LTI embeddings takes inspiration from the well-known *fundamental lemma* [14]. There, the authors have established the methodological criteria under which the whole state trajectory tube arising from a linear system can be represented by a finite number of excited dynamics. By properly adapting this result, in recent years LTI embeddings have been first derived for Wiener systems [65] and then a generalization to the class of bilinear systems, such as Hammerstein, finite-lag Volterra, have been provided in [13]. An alternative method to the data-driven modeling relies on the Koopman operator that has attracted considerable attention [66], [67]. Unfortunately, a critical issue inherent to the Koopman-based method is the selection of basis functions, which are a set of nonlinear real-valued functions with respect to the measurement coordinates. As a consequence, obtaining representations is proven to be difficult in all but the simplest systems, and they are often complex or are the output of uninterpretable black-box optimizations, see [68]. In this context, the introduction of deep learning methods enable a data-driven architecture for finding Koopman eigenfunctions, providing intrinsic linear representations of strongly nonlinear systems, see e.g., [69].

Along similar lines, it is worth to mention those methods that combine the use of multi-modelling framework based on the Linear Parameter Varying (LPV) approach with machine learning techniques that have started to appear, see [70] for a recent overview. Neural networks have been used to develop an approach to simultaneously estimate the states and structural dependency of LPV matrices from input/output data [71].

Although interesting from a theoretical point of view, all these items share at least two drawbacks. From one hand, the achieved methodological results are restricted to specific classes of nonlinear systems that are unlikely extendable to different configurations. From the other hand, computable algorithms are absent or even impossible to write down.

3.1.2 Main contribution

Starting from the above analysis, the main novelties of the proposed approach can be summarized as follows.

- An *ad-hoc* data-driven algorithm for embedding the state trajectory tube of a wide class of nonlinear systems within a multi-model state space description whose key ingredients are: the behavioral approach coupled with a RL scheme under the satisfaction of statistical requirements. At the best authors' knowledge, this idea can be considered as one of the first attempts to combine into a single framework these two methodologies to derive outer convex approximations of the exact differential inclusions characterizing the dynamical behavior of nonlinear systems;

- Since an outer convex approximation of the exact differential inclusion of a nonlinear system should be provided, the lack of knowledge on the mathematical structure of the plant can lead to an exhaustive search on the admissible (according to the prescribed constraints) extended (state-input) state space. Then, two strategies are properly combined to obtain an affordable computational scheme to overcome the difficulties arising from the lack of knowledge on the mathematical structure of the plant:

    1. from one side, the analysis of the nonlinear black-box is moved towards a linear time-invariant scenario. This is achieved by formally deriving conditions under which several system properties, such as equilibrium points, linear behaviors around the equilibria, polytope vertex reduction (see e.g., [72]), become valid under data analysis processes;

    2. from the other side a reinforcement learning algorithm, specifically an actor-critic based scheme, see Chapter 2, combined with a statistical analysis at each event [73].

- The accuracy of the obtained system multi-model is formally characterized by evaluating pre-specified confidence levels according to statistical tests. This task is coupled with the resulting actor-critic architecture by defining the reward function as a linear function of the level of confidence pertaining to the current convex approximation.
- Convergence properties are proven by resorting to set-theoretic arguments.

In conclusion, the proposed convex outer embedding algorithm takes advantages of the behavioral approach in terms of linear time-invariant description of the nonlinear system, the actor-critic reinforcement learning architecture for search parameter purposes and the statistics criteria for accuracy evaluations. On the other hand, the methodological interest to derive multi-model descriptions is strictly connected to design issues. Since systematic and computational efficient tools, such as those based on Linear Matrix Inequalities (LMIs) [23] or set-invariance arguments [8], are not directly applicable in nonlinear data-based settings, the availability of outer convex approximations, allow to use flexible, though a slightly more conservative, control algorithms.

3.1.3 Outline

A introductory section recalls notations and some preliminary results on differential inclusions. In Section 3.3, the state trajectory embbeding problem is stated in the discrete-time setting. Hence, Section 3.4 recalls and customizes to the considered framework the data-based LTI description. Section 3.5 provides insights to the probabilistic measure adopted to infer the accuracy of the computed uncertain linear embedding. A Deep Machine Learning characterization is presented in Section 3.6 with the aim to efficiently perform an exhaustive search on the parameter space. Hence, Section 3.7 collects all the methodological results to write down the computable polytopic embedding algorithm. Finally, two benchmark models, namely the quadruple-tank system and the Van der Pol oscillator, are used for evaluating the effectiveness of the proposed approach in Section 3.8.

3.2 Preliminary considerations

Definition 8 *A differential inclusion is described by [27]:*

$$\dot{x}(t) \in f(x(t), u(t), t), \;\; t \in \mathbb{R}, \tag{3.1}$$

where $x \in \mathbb{R}^n$, $u \in \mathbb{R}^m$ *and* $f(x(t), u(t)) : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ *is a set-valued map satisfying the following conditions for each* $u \in \mathbb{R}^m$

1. *it is upper semi-continuous, i.e., for each* $x \in \mathbb{R}^n$ *and each* $\beta > 0$ *there exists* $\eta > 0$ *such that, for all* $\xi \in \mathbb{R}^n$ *complying with* $|\xi - x| \leq \eta$, *one has* $f(\xi, u) \subseteq f(x, u) + \beta\mathcal{B}(0, 1)$, *with* $\mathcal{B}(0, 1)$ *the unit hyperball in* $\mathbb{R}^n$;

2. *for each* $x \in \mathbb{R}^n$ *the set* $f(x, u)$ *is nonempty, compact and convex.* □

As shown in [74], conditions 1)-2) guarantee that for each fixed $u \in \mathbb{R}^n$ there exists at least one solution to (3.1).

Definition 9 *The discrete-time model of the differential inclusion (3.1) is*

$$x(t+1) \in f_{T_s}(x(t), u(t)), \tag{3.2}$$

with $T_s > 0$ *the sampling time and* $f_{T_s}(x, u)$ *the set of values that the solutions to (3.1) can take at time* $t = kT_s$, $k \in \mathbb{Z}_+ := \{0, 1, \ldots\}$, *when starting at* x *and with the constant input* $u(t) = u, \forall t$, *applied.* □

3.3 Problem formulation

Consider the class of non-linear systems described by the following discrete-time state space model

$$\begin{aligned} x(t+1) &= \not{f}(x(t), u(t)) \\ y(t) &= Cx(t) \end{aligned} \tag{3.3}$$

where $t \in \mathbb{Z}_+$, $x(t) \in \mathbb{R}^n$ is the plant state, $u(t) \in \mathbb{R}^m$ the command input and $y(t) \in \mathbb{R}^{n_y}$ the measurement vector with C the output map. Moreover, the set-membership constraints 1.3 are prescribed.

As the discrete-time nonlinear function map $\not{f} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ in (3.3) is concerned, the following standard assumptions [27, 75] guarantee the existence of convex outer approximations for the state trajectories of the non-linear dynamics (3.3):

Assumption 1 $\not{f}(\cdot, \cdot)$ *is a continuously differentiable function - therefore defined as* $\mathcal{C}^1$ *function - where* $\not{f}(0_n, 0_m) = 0_n$. □

Assumption 2 $\not{f}(\cdot,\cdot)$ *is uniformly Lipschitz in* $u \in \mathcal{U}$, *i.e.,*

$$\|\not{f}(x,u) - \not{f}(\xi,u)\| \leq \mathcal{L}_x \|x-\xi\|, \forall (x,\xi) \in \mathcal{X} \times \mathcal{X},$$

with $\mathcal{L}_x \in \mathbb{R}^+$ *the Lipschitz constant.* □

In the sequel, the following operating conditions are considered:

Assumption 3 *The state vector* x *is fully measurable., i.e.* $C = I_n$. □

Data-driven scenario - The map $\not{f}(\cdot,\cdot)$ is unknown while the upper-bound scalars x_{max} and u_{max} are available. Input-output data sequences $\{u^d(t), x^d(t)\}_{t=0}^{N_s}$, $N_s < \infty$, complying with (1.3), have been obtained by performing experiments on the real plant. □

Then, the problem of interest can be stated as follows.

Polytopic embedding of nonlinear dynamics (PE-ND) - *Given a family of input-output data* $\Gamma := \left\{ \{u^d(t), x^d(t)\}_{t=0}^{N_s} \right\}_{d=1}^{M}$, *with the apex* d *referring to available data and* M *to the data family length, determine a polytopic outer approximation (1.6)-(1.7) of the differential inclusion (3.2)*

$$\not{f}((x(t),u(t)) = f_{T_s}(x(t),u(t)) \begin{bmatrix} x(t) \\ u(t) \end{bmatrix}, \tag{3.4}$$

with $f_{T_s}(x(t),u(t)) \in \Omega,\ \forall t$, *such that*

$$f_{T_s}(x,u) \subseteq \Omega,\ \forall (x,u) \in \mathcal{X} \times \mathcal{U} \tag{3.5}$$

□

The **PE-ND** problem has been addressed by conceiving a novel procedure capable to exploit input-output data sequences and to take advantage of the following three methodologies:

1. the behavioral approach,
2. the statistical analysis of dynamical systems,
3. the reinforcement learning (RL) framework.

The next sections will be first devoted to customize these ingredients within the **PE-ND** prescriptions.

3.4 Data based LTI description

For the subsequent developments, the concepts of equilibrium and linear system behavior are of interest.

Definition 10 *A pair* $\{u^s, x^s\} \in \mathbb{R}^{m+n}$ *is an equilibrium of the nonlinear system (3.3), if the input-output sequence* $\{u^d(t), x^d(t)\}_{t=0}^{\infty}$, *with* $\{u^d(t), x^d(t)\} = \{u^s, x^s\}$, $\forall t \in \mathbb{Z}_+$, *is a state trajectory of (3.3).* □

Conversely, if the nonlinear plant (3.3) behaves as a LTI system, e.g., the minimal realization (A, B, C), i.e., reachable and observable [76], resulting from (1.6) when $\rho(t) = \bar{\rho}, \forall t$, with $\bar{\rho} \in \Lambda$ a fixed multi-model realization, the *Definition 10* simplifies as follows.

Definition 11 *A pair* $\{u^s, x^s\} \in \mathbb{R}^{m+n}$ *is an equilibrium of the LTI system* (A, B, C), *if the input-output sequence* $\{u^d(t), x^d(t)\}_{t=0}^{n}$, *with* $\{u^d(t), x^d(t)\} = \{u^s, x^s\}$, $\forall t \in \mathbb{Z}_+$, *is one of its state trajectories.* □

As it is well-known, the plant dynamics (3.3) can be approximated around the equilibrium $\{u^s, x^s\}$ via linearization arguments [72]. Since in the considered data-driven setting the knowledge of the map $\not{f}(\cdot, \cdot)$ is not available, a LTI model of (3.3) can be derived by using the Fundamental Lemma [14] and the formulas presented in [49]. Let

$$\begin{aligned} U_{0,1,T} &:= \begin{bmatrix} u^d(0) & u^d(1) & \cdots & u^d(T) \end{bmatrix} \\ X_{0,T} &:= \begin{bmatrix} x^d(0) & x^d(1) & \cdots & x^d(T) \end{bmatrix} \\ X_{1,T} &:= \begin{bmatrix} x^d(1) & x^d(2) & \cdots & x^d(T) \end{bmatrix} \end{aligned} \tag{3.6}$$

be sequences of sample data of length $T > 0$ such that $U_{0,1,T}$ is a persistently exciting input sequence of order $n + 1$, i.e. the following rank condition is satisfied

$$Rank \begin{bmatrix} U_{0,1,T} \\ \hline X_{0,T} \end{bmatrix} = n + m \tag{3.7}$$

As outlined in [49], one derives that

$$x(t+1) = X_{1,T} \begin{bmatrix} U_{0,1,T} \\ \hline X_{0,T} \end{bmatrix}^{\dagger} \begin{bmatrix} u(t) \\ x(t) \end{bmatrix} \tag{3.8}$$

with † the Penrose pseudo-inverse operator. Then the following linear time-invariant description comes out:

$$[\mathscr{A} \; \mathscr{B}] = X_{1,T} \begin{bmatrix} U_{0,1,T} \\ \hline X_{0,T} \end{bmatrix}^{\dagger} \tag{3.9}$$

If the system generating the trajectories is linear - therefore described by the matrices (A, B, C) - and the input is persistently exciting, the fundamental lemma stands that $[A\ \ B] = [\mathscr{A}\ \mathscr{B}]$.

In order to make computable the above reasoning, two questions must be addressed:

1. determine an equilibrium for the unknown dynamic system (3.3);
2. built admissible data sequences (3.6) complying with (3.7).

Notice that, according to *Definition 10*, an infinite sequence of data should be experimentally obtained to assess the existence of an equilibrium point. Then, the issue *1.* is solved by means of the following result.

Proposition 1 *Let $\beta_a > 0$ be an arbitrary small scalar. Given a constant input $u^s \in \mathbb{R}^m$, iteratively applied to (3.3) from $t = 0$ onward, and $\Delta x(t) := x(t) - x(t-1)$, $t \geq 1$, the resulting one-step state difference. If starting from a certain time instant $\bar{t} > 0$ one has that*

$$\|\Delta x(\bar{t} + k)\| \leq \beta_a, \ \ k = 0, 1, \ldots, h, \tag{3.10}$$

then $x^s \leftarrow x(\bar{t} + h)$ is an equilibrium for (3.3).

Proof - If the repetitive application of a constant input u^s leads to a convergent state trajectory, this implies that $\Delta x(\infty) = 0_x$ as prescribed by *Definition 10*. Assume that this is true, then at $t \to \infty$ an equilibrium will be reached,. As a consequence, the nonlinear system (3.3) has a linear behavior within the admissibility region $\mathcal{B}([x^{s^T}\ u^{s^T}]^T, l_s),\ l_s > 0$, see [72]. In virtue of this reasoning, if at $\bar{t}$ one has that

$$\|\Delta x(\bar{t})\| \leq \beta_a \tag{3.11}$$

then it is sufficient to check the validity of (3.11) for the successive samples $x(\bar{t} + k)$, $k = 1, \ldots, h$, to comply with the requirements of *Definition 11*. Finally, since the tolerance β_a is vanishing the computed equilibrium is given by $x(\bar{t} + h) =: x^s$. □

As the second question is concerned, the data sequences (3.6) must be obtained by satisfying the following set-membership condition:

$$[x^{d^T}(t)\ u^{d^T}(t)]^T \in \mathcal{B}([x^{s^T}\ u^{s^T}]^T, l_s), \forall t.$$

Unfortunately the radius l_s of the hyperball $\mathcal{B}([x^{s^T}\ u^{s^T}]^T, l_s)$ is unknown and cannot be explicitly computed because the map $\not{\mathscr{A}}(\cdot, \cdot)$ is not available. As a consequence, the next

proposition identifies the points belonging to this hyperball where the nonlinear system behaves as a linear model. Again, it is possible to resort to system theory arguments.

Proposition 2 *Given the equilibrium* $(u^s,\ x^s)$, *consider the following state evolution sequences of length* h *obtained by applying the constant input* $\bar{u}$ *to the system starting from the initial condition* $\bar{x}$:

$$\bar{x}(k) = \mathscr{A}^k \bar{x} + \sum_{i=0}^{k-1} \mathscr{A}^{k-1-i}\, \mathscr{B}(\bar{u}) \quad , k = 1, \ldots, h, \tag{3.12}$$

and

$$\tilde{x}(k) = \mathscr{A}^k (\bar{x} - x^s) + \sum_{i=0}^{k-1} \mathscr{A}^{k-1-i}\, \mathscr{B}(\bar{u} - u^s) \quad , k = 1, \ldots, h. \tag{3.13}$$

If

$$[\bar{x}^T\ \bar{u}^T]^T \in \mathcal{B}([x^{s^T}\ u^{s^T}]^T, l_s)$$

then

$$\bar{x}(k) - x^s \equiv \tilde{x}(k),\ \ k = 1, \ldots, h. \tag{3.14}$$

Proof - It is sufficient to exploit the case of linear systems since the state trajectory arising from the application of the pair $(u^s,\ x^s)$ belongs to $\mathrm{Proj}_x \left(\mathcal{B}([x^{s^T}\ u^{s^T}]^T, l_s) \right)$. In order to prove that the same holds true for $(\bar{u},\ \bar{x})$, the additive property of the superimposition principle is applied to the LTI model (3.9) as shown in (3.13) and the resulting h samples compared by (3.14). □.

3.5 A probabilistic measure of performance

Given a multi-model description (1.6)-(1.7) obtained as the convex hull of a set of LTI models computed according to the *Propositions 1-2* prescriptions - since it has been achieved by exploiting input-output data -, a key issue consists in understanding how much this outer polytopic approximation of (3.3) is effective.

Ideally, this should prescribe to check that all the admissible (compatible with (1.3)) state trajectories of (3.3) lie in the state trajectory tube, say $\mathcal{T}(\Omega)$, associated to (1.6)-(1.7). Evidently this is not viable for computational reasons. A possible way to overcome such a hitch consists in using a probabilistic approach within the Monte Carlo simulation setting, see e.g. [77]. Then, the effectiveness of the outer approximation can be formalized as follows.

Probabilistic analysis - *Let $\mathcal{X}$ be equipped with a probability measure $\mathscr{P}$, the problem is to estimate the measure $\mathscr{P}(\mathscr{S})$ of the subset $\mathscr{S} \subset \mathcal{X}$ consisting of the elements of $\mathcal{X}$ for which a "successful" event occurs: the state trajectory of (3.3), starting from a randomly chosen initial state $x \in \mathcal{X}$ under the action of a random constant input $u \in \mathcal{U}$, belongs to $\mathcal{T}(\Omega)$.* □

As it is well-known, if a set $\mathcal{X}_\mu$ of μ independent identically distributed (i.i.d.) samples is drawn from $\mathcal{X}$ an empirical estimate of $\mathscr{P}(\mathscr{S})$ is given by the fraction of "successful" events $\hat{P}(\mathscr{S}, \mathcal{X}_\mu)$, and the Chernoff bound holds

$$\begin{cases} \Pr\{|\hat{P}(\mathscr{S}, \mathcal{X}_\mu) - \mathscr{P}(\mathscr{S})| > \epsilon_p\} \leq 2^{-2\mu\epsilon_p^2} \\ \text{or equivalently} \\ \Pr\{|\hat{P}(\mathscr{S}, \mathcal{X}_\mu) - \mathscr{P}(\mathscr{S})| \leq \epsilon_p\} > 1 - 2^{-2\mu\epsilon_p^2} \end{cases} \tag{3.15}$$

Essentially, the inequality (3.15) allows to evaluate $\mathscr{P}(\mathscr{S})$ on the basis the accuracy ϵ_p with which $\hat{P}(\mathscr{S}, \mathcal{X}_\mu)$ estimates $\mathscr{P}(\mathscr{S})$ by using μ i.i.d. samples and the confidence level $1 - \circledcirc = 1 - 2^{-2\mu\epsilon_p^2}$.

According to (3.15), the probability of success is determined by extracting μ i.i.d. samples from the following sets:

- $\hat{\mathcal{X}}$ uniform distribution on $\mathcal{X}$;
- $\hat{\mathcal{U}}$ uniform distribution on $\mathcal{U}$.

Hence, these distributions generate μ state trajectories of (3.3), say $\{\hat{x}_k(u; x)\}$ to be read as the sequence of data emanating from the initial state condition x when the constant input u is applied, that are checked by following the prescriptions of the **Probabilistic analysis**:

$$\hat{x}_k(x; u) \in \mathcal{T}(\Omega), \forall k \in \mathbb{Z}_+ \tag{3.16}$$

3.6 Reinforcement Learning approach

In this section, the **PE-ND** problem is framed within a reinforcement learning scheme. It is worth that the derivation of a polytopic embedding (3.4)-(3.5) for the nonlinear state trajectories prescribes the exploration of the whole admissible extended space $\mathcal{X} \times \mathcal{U}$. This, though in principle achievable, is not viable from a computational point of view, therefore a

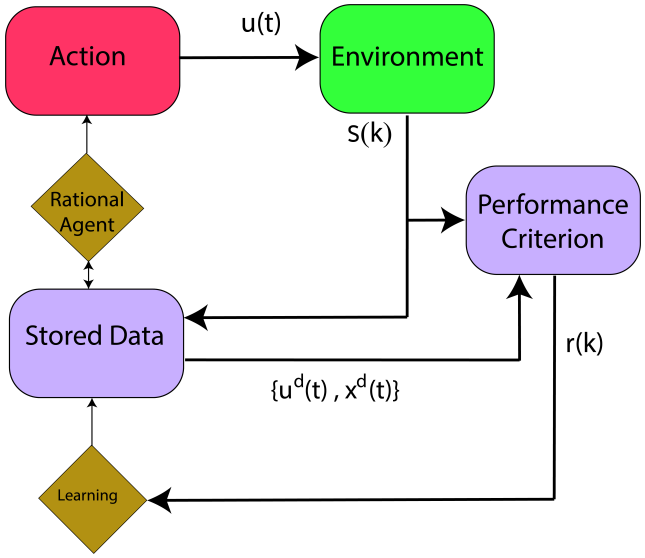

Figure 3.1: Reinforcement learning: the diagram flow

machine-learning approach results to be the more adequate way to proceed in consideration of the intrinsic uncertainty of the considered problem.

Figure 3.1 shows the RL diagram flow for the considered setting. Here, the non-linear model (3.3) represents the environment where the RL algorithm operates while the rational agent is designed as an actor–critic architecture in charge to select the action u by using the stored applied input data $\mathcal{U}_{prec}$ (the set of actions used in the past) and the polytopic description (1.6)-(1.7). The main RL ingredients are below summarized:

- **Action space** - the set of admissible inputs:

$$\mathcal{A} := \mathcal{U} \tag{3.17}$$

  Notice that the action set $\mathcal{A}$ is a continuous space and this is a reason behind choosing the actor-critic architecture.

- **State space** -

$$\mathcal{S} := \{s = \Omega\} \tag{3.18}$$

  where a state realization is a multi-model description (1.6)-(1.7) of (3.3);

- **Reward function** (Performance criterion) - $r : \mathcal{S} \times \mathcal{S} \to \mathbb{R}$ is the weighted sum of two components:

$$r(s,u) := (1 - \circledcirc(s)) + \beta_r \psi_u(s, s^+) \tag{3.19}$$

- $(1 - \circledcirc(s))$ is the confidence level computed via (3.15) and, as a consequence, checks the effectiveness of the current embedding (3.4)-(3.5),
- $\psi_u : \mathcal{S} \times \mathcal{S} \to \{0, 1\}$ - penalized by the scalar $\beta_r \in [0, 1]$ - accounts for the capability of success:

$$\psi(s, s^+) := \begin{cases} 1, & if\ s \neq s^+ \\ 0, & otherwise \end{cases} \tag{3.20}$$

where s^+ is the one-step state evolution arising from s under the chosen action u.

- **Return function**

$$G_r(k) := \sum_{k_1=0}^{\infty} \gamma^{k_1}\, r(s(k + k_1), u(k + k_1)) \tag{3.21}$$

with $\gamma \in (0, 1)$ the discount factor.

- **The policy**

$$\pi(s(k), \mathcal{U}_{prec}; \theta_a) = a(k) \tag{3.22}$$

It depends on $\mathcal{U}_{prec}$ because it is not allowed to use twice an action. θ_a are the **Actor** neural network weights.

- **State-action value function**

$$\mathcal{Q}(s, u; \theta_c) := E\left[G_r | \pi\right] \tag{3.23}$$

with θ_c the **Critic** neural network weights.

It is worth to underline that the **Critic** is the value function $\mathcal{Q}(s, u; \theta_c)$ in charge to approximate the return G_r while the **Actor** is the policy $u(k) = \pi(s(k); \theta_a)$. Notice that in this specific framework, the action $a(k)$ coincides with the control input $u(k)$ to be applied. As in formula 2.10, the expected value of $r(k + 1)$ is:

$$\hat{r}(k + 1) := E[r(k + 1)] = \mathcal{Q}(s(k), u(k); \theta_c) - \gamma \mathcal{Q}(s(k + 1), \pi(s(k + 1); \theta_a); \theta_c) \tag{3.24}$$

and the temporal difference error

$$\delta_\pi = r(k + 1) - \hat{r}(k + 1) \tag{3.25}$$

Finally, the scheme of figure 2.4 prescribes the use of the Gaussian noise technique, i.e. random values $n_a(k)$, normally distributed with a zero mean and a standard deviation $\psi_a(k)$, are added to the selected action u^*. This has the aim to make the RL system more able to explore the action space.

In the present context, the above scheme has been implemented by resorting to a reset-free approach, see [78] for details. The main reason behind this choice relies on the need to take trace at each time step of the computed multi-model description (1.6)-(1.7), i.e. the state realization $s(k)$. In particular, the rational agent learns by increasing the variance of the state distribution as a result of the application of the perturbation $n_a(k)$ to the policy (3.22). For exploration purposes, the **Actor** is iteratively executed for a finite number of steps, say T_a, according to intermittent noise realizations. This leads to the desirable consequence that the exploration of $\mathcal{A}$ is encouraged in order to avoid local minima. According to the above developments, the following **Actor** procedure comes out.

Reset-Free Actor (RF-A) Procedure

INPUT: $s(k)$;

GLOBAL: T_a; $active$; j; $\mathcal{U}_{prec}$;

OUTPUT: $u(k)$

1: **if** $j < T_a$ and $active == false$ **then** $n_a(k) \equiv 0$ & $j \leftarrow j + 1$

2: **else if** $j < T_a$ and $active == true$ **then** $n_a(k) \sim \mathcal{N}(0, \sigma_a(k))$ & $j \leftarrow j + 1$

3: **end if**

4: **if** $j == T_a$ and $active == false$ **then** $active \leftarrow true$ & $j \leftarrow 0$

5: **else if** $j == T_a$ and $active == true$ **then** $active \leftarrow false$ & $j \leftarrow 0$

6: **end if**

7: **Determine** $\pi(s(k); \theta_a)$ as the output of the underlying neural network

8: **Compute**

$$u(k) = \pi(s(k), \mathcal{U}_{prec}; \theta_a) + n_a(k) \tag{3.26}$$

9: **Update** $\mathcal{U}_{prec}$

Remark 1 The logical variable *active* is instrumental to take trace of T_a consecutive non-null realizations of the signal noise $n_a(k)$. □

3.7 State trajectory embedding algorithm

In this section, the developments of the above sections are harmonized to write down a computable algorithm. To this end, it is mandatory to formally take into account the unavoidable noise deriving from the exclusively use of input-output data [79]. In particular, two questions have to be adequately addressed:

1. How can false occurrences on the set-membership condition (3.16) be mitigated?

2. How can the number of vertices characterizing the convex hull Ω be kept at a reasonable level as the number of LTI realizations $[A_j \;\; B_j]$ grows up?

The first issue can be attacked by resorting to the perturbation bound of the Moore-Penrose inverse under the spectral norm proposed in [80, 81]:

In view of this, the condition (3.16) can be relaxed as follows.

Proposition 3 *Let Ω be a convex hull describing the multi-model (1.6)-(1.7). Then, any state trajectory $\{\hat{x}_k(x;u)\}$ of (3.3) is also a state trajectory of (1.6)-(1.7) if*

$$\hat{x}_k(x;u) \in \mathcal{T}(\Omega) + \mathcal{B}(0,\eta) \quad ,\forall k \in \mathbb{Z}_+ \tag{3.27}$$

with

$$\eta := \frac{1}{\sqrt{n}\,\beta_e \min\limits_{j\in\{1,\dots,L\}} \left\{ \left\| \begin{bmatrix} U^j_{0,1,T} \\ \hline X^j_{0,T} \end{bmatrix}^{\dagger} \right\|_2^2 \right\}} \tag{3.28}$$

The *proof* of this result is available in [45].

The second question is addressed by taking advantage from model reduction arguments [82]. The following result holds.

Proposition 4 *Given the multi-model system (1.6)-(1.7). Let $[\mathscr{A} \;\; \mathscr{B}]$ be a LTI realization computed by formulas (3.8)-(3.9). Then,*

$$[\mathscr{A} \;\; \mathscr{B}] = \sum_{j=1}^{L} \rho_j \, [A_j \;\; B_j], \text{ for some } \rho = [\rho_1, \dots \rho_L] \in \Lambda$$

if there exists a polytope vertex $[A_j \;\; B_j]$, $j \in \{1,\dots,L\}$, such that

$$\| \; \mathscr{W}(z) - W^j(z) \|_\infty \leq 2 \sum_{i=1}^{n} (\tilde{\upsilon}_i - \upsilon_i^j) \tag{3.29}$$

where $\mathscr{W}(z)$ *denotes the transfer function of the system* $(\mathscr{A}, \mathscr{B}, I_n)$, W_ρ *while* $W^j(z)$ *the transfer function of the system realization* (A_j, B_j, I_n), *and* $\tilde{v}, v^j$, *the associated singular values vectors.*

Proof - As shown in [83], a generic system $W(z)$ can be model reduced if the following arguments are valid. Let $W_{n-k}(z)$ and $W_k(z)$ be the subsystems of $W(z)$ split according to its n singular values. If (3.29) holds with $W(z)$ and $W_k(z)$ in place of $\mathscr{W}(z)$ and $W^j(z)$, respectively, then the related dynamical system can be described by simply referring $W_k(z)$, because the two transfer functions numerically coincide. Therefore, by adapting this reasoning, the LTI realization $[\mathscr{A}\ \mathscr{B}]$ is a convex combination of Ω if (3.29) is verified. □

Let $th \in \mathbb{R}_+$ be a positive scalar understood as the threshold on the desired confidence level $(1 - \circ)$ arising from (3.15). Then, the following algorithm can be derived.

Polytopic Embedding (PE) Algorithm

INPUT: β_r; γ; T_a; th; Γ

OUTPUT: Ω

Initialization

1: **Determine** two equilibrium points $\{u^{s_1}, x^{s_1}\}$ and $\{u^{s_2}, x^{s_2}\}$ according to the prescriptions of *Proposition 1*;

2: **Generate** two admissible data sequences compatible with (3.12)-(3.14) under the satisfaction of (3.7) for $\{u^{s_1}, x^{s_1}\}$ and $\{u^{s_2}, x^{s_2}\}$, respectively;

3: **Compute** $[A_1\ \ B_1]$ and $[A_2\ \ B_2]$ by (3.9). Let $\Omega = \{[A_1\ \ B_1],\ [A_2\ \ B_2]\}$ and $L = 2$;

4: **Set** $active := false$;

Embedding

Statistical performance -

1: **Compute** the tolerance η by formula (3.28);

2: **Evaluate** the polytopic embedding Ω using the stored state trajectories Γ by means of the formula (3.27);

3: **Determine** the confidence level $(1 - \circ)$ via (3.15);

4: **if** $(1 - \circledcirc) \geq th$ **then Exit**

5: **end if**

INPUT SPACE EXPLORATION -

1: **Choose** $u(k)$ by (3.26)

2: **Update** $\mathcal{U}_{prec} \leftarrow \mathcal{U}_{prec} \cup \{u_k\}$;

CONVEX HULL UPDATING -

a) **Perform** the prescriptions of *Proposition 1*;

b) **If** u_k is an equilibrium **then**

1. **Generate** an admissible data sequence compatible with (3.12)-(3.14) under the satisfaction of (3.7) for $\{u_k, x_k\}$;
2. **Compute** $[\mathscr{A}\ \mathscr{B}]$ by (3.9);
3. **If** $\nexists \rho \in \Lambda$ such that $[\mathscr{A}\ \mathscr{B}] = \sum_{j=1}^{L} \rho_j [A_j\ \ B_j]$ and (3.29) fails **then update** $s(k+1) = \Omega \leftarrow Co\{\Omega, [\mathscr{A}\ \mathscr{B}]\}$ and $L \leftarrow \#(s(k+1))$;

c) **else** $s(k+1) = s(k)$;

3: **Perform** the STATISTICAL PERFORMANCE procedure with $s(k+1)$ in place of Ω;

4: **Get** the reward $r(s(k), u(k))$;

5: **Compute** the expected reward (3.24) and the estimation error (3.25)

6: $k \leftarrow k + 1$

7: **Goto** Step 1 of the INPUT SPACE EXPLORATION procedure;

Finally, it is worth to underline that the **PE** algorithm enjoys the following convergence property.

Proposition 5 *Given a nonlinear system (3.3)-(1.3) complying with* **Assumptions** *1-3. Let Ω_{exact} be its exact linear polytopic embedding. Then, the* **PE** *algorithm asymptotically converges to the following convex hull*

$$\Omega_{exact} \setminus \mathcal{P}\left(\mathcal{B}(0, \eta)\right) \tag{3.30}$$

with $\mathcal{P}(\cdot)$ the polyhedral operator that provides a polyhedron that embeds the set that is provided as argument.

Proof - In **Step** 2 of the STATISTICAL PERFORMANCE procedure, the validity of the polytopic embedding Ω is checked by means of the set-membership (3.27). As outlined in *Proposition*

3, this depends on the tolerance η, accounting for the measurement error on the noisy plant realizations as outlined in *Proposition 3*. Then, as the number of events goes towards infinity ($k \to \infty$), one has that the best achievable outer approximation of (3.3)-(1.3) satisfies the set inclusion $\Omega \subseteq \Omega_{exact}$. □

3.8 Numerical results

In this section, the effectiveness of the **PE** algorithm is verified by considering two cases of study for which an exact polytopic model is also analytically derived. First, outer convex approximations are derived for both the plants. Then, the level of accuracy is assessed by using an error fitting index defined in the experiments below when addressing a constrained zero regulation problem by using the LMI-based receding horizon controller proposed in [23].

3.8.1 Van der Pol oscillator

As done in other contexts, to evaluate the performance of sophisticated nonlinear control approaches [84], the Van Der Pol oscillator model is instrumental to check the algorithm performance on a well-reputed benchmark example. The considered model [85] is the following:

$$\begin{aligned} \dot{x}_1(t) = &\quad x_2(t) \\ \dot{x}_2(t) = &\quad -x_1(t) - (1 - x_1^2(t))x_2(t) + u(t) \end{aligned} \tag{3.31}$$

subject to the following component-wise constraints

$$|u| \leq 0.2 \ , \quad |x_i| \leq 0.25, \ i = 1, 2 \tag{3.32}$$

The discrete-time state space description is obtained by applying the forward Euler method with the sampling time $T_s = 0.1\,[s]$.

The following choices have been made:

- $M = 10^5$ input-output sequences stored as $\Gamma := \left\{\{u^d(t), x^d(t)\}\right\}_{d=1}^{M}$;
- $\beta_a = 0.001$ the accuracy level on the one-step difference (3.10);
- $\gamma = 0.9$ the discount factor in (3.21);
- $\beta_r = 0.1$ the scalar characterizing the reward (3.19);

- $T_a = 2$ the number of consecutive non-null realizations of the noise $n_a(k)$ within the **RF-A** procedure;

- the **Critic** neural network is is reported in figure 3.2. There, "FC" stands for Fully Connected layer, and the number of neurons in the layer is reported. See, Appendix A for further details about Neural Networks. Notice that the whole **Critic** computes the state-action value function where the last layer accounts for the output and, as a consequence, a single neuron is required. The activation functions are ReLU [86];

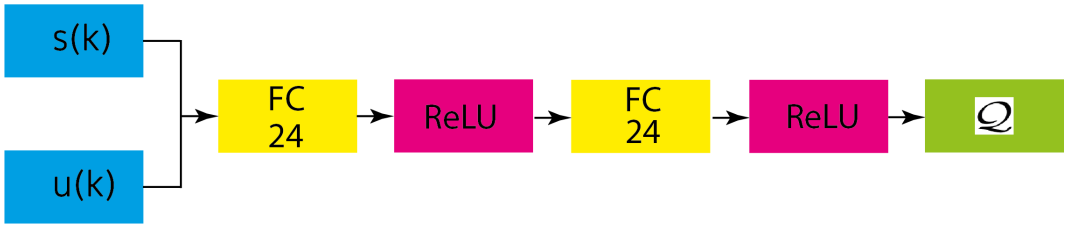

Figure 3.2: Critic network

- the **Actor** neural network is reported in figure 3.3. As discussed in Section 3.6 (see the **Reset-Free Actor (RF-A) Procedure**), the output branches provide the mean and standard deviation of each component of the action $u(k)$.

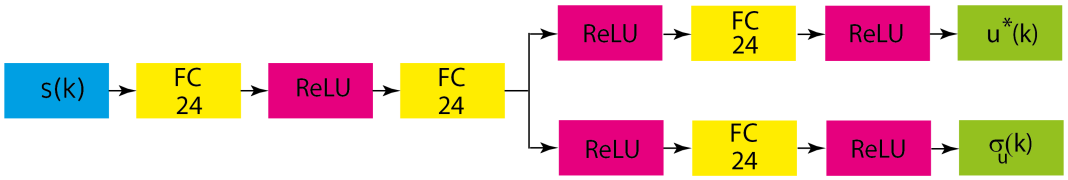

Figure 3.3: Actor network

By imposing the confidence level threshold equal to $th = 0.9$, the **PE** algorithm provides the convex hull Ω defined by:

$$[A_1, B_1] = \begin{bmatrix} 1 & 0.1 & 1 \times 10^{-5} \\ 0 & 0.9038 & 0.1 \end{bmatrix}$$

$$[A_2, B_2] = \begin{bmatrix} 1 & 0.1 & 0 \\ 0 & 0.9 & 0.1 \end{bmatrix}$$

$$[A_3, B_3] = \begin{bmatrix} 1 & 0.1 & 0 \\ 0 & 0.9023 & 0.1 \end{bmatrix}$$

Figure 3.4 depicts the reward evolution as the number of events k increases. As expected, a monotonically non-decreasing behavior comes out until the stopping criterion (**Step** 4 of the STATISTICAL PERFORMANCE procedure) holds true. Notice also that the flat zones of the $(1-\circ(k))$ evolution account for new linear realizations $[\mathscr{A}\ \mathscr{B}]$ that comply with (3.29) or are convex combinations of the convex hull Ω determined at the $(k-1)-th$ event.

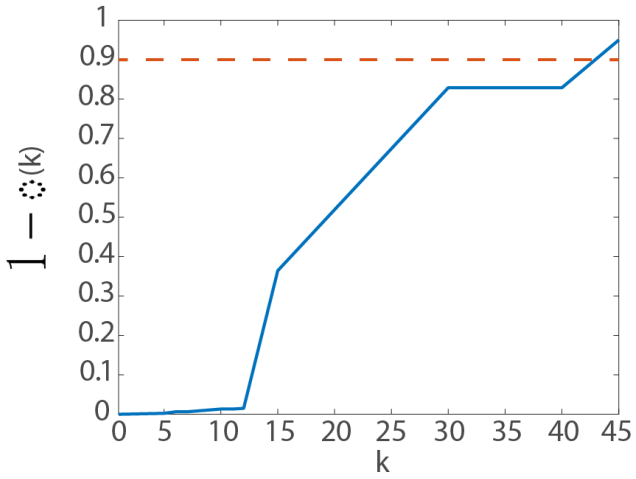

Figure 3.4: Van der Pol oscillator - Confidence level

An exact multi-model outer approximation of (3.31)-(3.32) is defined by the convex hull Ω_{exact} consisting of the following vertices:

$$[A_1^{exact}, B_1^{exact}] = \begin{bmatrix} 1 & 0.1 & 0 \\ -0.1 & 1.03 & 0.1 \end{bmatrix}$$

$$[A_2^{exact}, B_2^{exact}] = \begin{bmatrix} 1 & 0.1 & 0 \\ -0.1 & 1.06 & 0.1 \end{bmatrix}$$

For comparison purposes, 1000 feasible initial conditions have been randomly chosen and the state trajectories $\hat{x}_{exact}(k;x)$ and $\hat{x}_{\Omega}(k;x)$ resulting from the receding horizon controller pertaining to Ω_{exact} and Ω, respectively, have been generated. The evaluation is performed according to the relative error index:

$$\begin{aligned} & e_r^k = \max_{1\leq i\leq n} \frac{|\hat{x}_{\Omega}(k;x(k))(i) - \hat{x}_{exact}(k;x(k))(i)|}{|\hat{x}_{exact}(k;x(k))(i)|}, \\ & k = 1, \ldots, 1000 \end{aligned} \tag{3.33}$$

and the related statistics are reported in Table 3.1. From such data, one can claim that the achieved convex hull Ω is numerically comparable with the exact one Ω_{exact} in terms of achievable control performance.

Table 3.1: Statistics of the relative error e_r^k

Mean value	7.4234×10^{-4}
Max value	0.0044
Standard deviation	3.5749×10^{-7}

Notice that the **PE** algorithm selects 43 equilibrium points before overcoming the prescribed threshold $th = 0.9$ on the confidence level, see figure 3.4. As expected, the convex hull operator, the reduction conditions (3.29) allows to achieve the polytopic outer approximation characterized by the system vertexes $[A_i, B_i], i = 1, 2, 3$.

Finally, the results reported in figure 3.5 are instrumental to show the effectiveness of the achieved outer approximation Ω. To this end, four initial state conditions have been considered within the admissible state space defined by the constraints (3.32). Hence, the corresponding state evolutions under the action of constant inputs pertaining to the nonlinear model (continuous red line) always lie within the tube defined by the computed polytope vertexes (continuous blue lines).

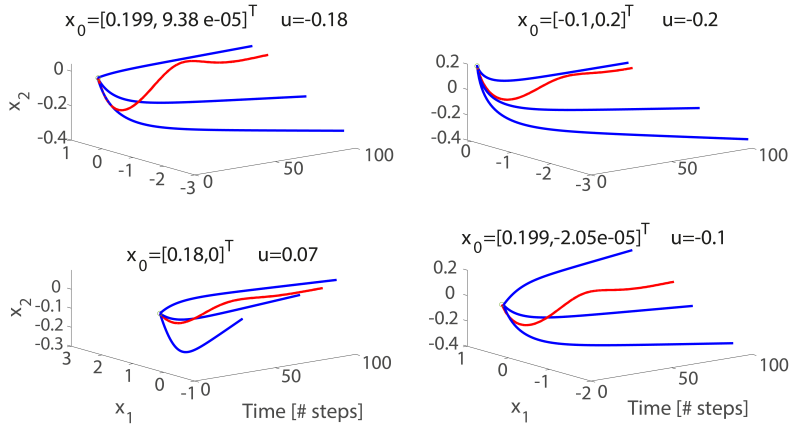

Figure 3.5: State trajectories: Non-linear *vs* Polytopic embedding

3.8.2 Four tanks

This second example is instrumental because of two reasons:

- the four-tanks system is a laboratory well-reputed benchmark used in different contexts;
- it allows one to show the capability of this proposed scheme to efficiently deal with the dimensional curse.

The dynamics of the four tanks model is described by the following continuous-time mathematical model taken from [87]:

$$\begin{aligned}
\dot{l}_1(t) &= -\tfrac{cs_1}{CS_1}\sqrt{2\mathfrak{g}l_1(t)} + \tfrac{cs_3}{CS_1}\sqrt{2\mathfrak{g}l_3(t)} + \tfrac{2\mathfrak{r}_1\mathcal{K}_1}{CS_1}u_1(t) \\
\dot{l}_2(t) &= -\tfrac{cs_2}{CS_2}\sqrt{2\mathfrak{g}l_2(t)} + \tfrac{cs_4}{CS_2}\sqrt{2\mathfrak{g}l_4(t)} + \tfrac{\mathfrak{r}_2\mathcal{K}_2}{CS_2}u_2(t) \\
\dot{l}_3(t) &= -\tfrac{cs_3}{CS_3}\sqrt{2\mathfrak{g}l_3(t)} + \tfrac{(1-\mathfrak{r}_2)\mathcal{K}_2}{CS_3}u_2(t) \\
\dot{l}_4(t) &= -\tfrac{cs_4}{CS_4}\sqrt{2\mathfrak{g}l_4(t)} + \tfrac{(1-\mathfrak{r}_1)\mathcal{K}_1}{CS_4}u_1(t)
\end{aligned} \tag{3.34}$$

All the parameters are reported in Table 3.2.

Table 3.2: Four tanks parameters

Parameter	Meaning	Unit	$i=1$	$i=2$	$i=3$	$i=4$
$\mathfrak{g}$	Gravity constant	$[m/s^2]$	9.81	9.81	9.81	9.81
CS_i	Cross-section of tank i	$[cm^2]$	28	32	28	32
cs_i	Cross-section of outlet hole i	$[cm^2]$	0.071	0.057	0.071	0.057
l_i	Water level i	$[cm]$	/	/	/	/
$\mathfrak{r}_i$	Set of valve i	/	0.43	0.34	/	/
$\mathcal{K}_i$	Flow/voltage i	$[cm^3/Vs]$	3.33	3.35	/	/

Moreover, the system (3.34) is subject to the following constraints:

$$0 \leq l_i(t) \leq \bar{l}_i,\ i=1,\dots 4\ ,\quad 0 \leq u_j(t) \leq u_{\max},\ j=1,2,\ \forall t, \tag{3.35}$$

with $\bar{l}_1 = 2.5[cm]$, $\bar{l}_2 = 2.5[cm]$, $\bar{l}_3 = 1.5[cm]$, $\bar{l}_4 = 1.5[cm]$, $u_{max} = 8[V]$. The system has been discretized using forward Euler method with sampling time $T_s = 0.5\,[s]$.

By using the **Actor** and **Critic** neural networks defined in the previous section and under

the choices: $M = 10^5$, $\beta_a = 0.001$, $\gamma = 0.9$, $\beta_r = 0.1$, $T_a = 2$ and $th = 0.9$, the **PE** algorithm provides the following convex hull Ω :

$$\begin{cases} A_1 = \begin{bmatrix} 0.96 & -0.17 & 0.25 & 0.1 \\ 8.7 \times 10^{-3} & 0.87 & 0.04 & 0.11 \\ -0.05 & 0.05 & 0.84 & -0.21 \\ 1.8 \times 10^{-3} & -4.2 \times 10^{-3} & 1.4 \times 10^{-3} & 0.73 \end{bmatrix} \\ B_1 = \begin{bmatrix} 8.0 \times 10^{-3} & -4.3 \times 10^{-3} \\ -7.8 \times 10^{-4} & 4.0 \times 10^{-3} \\ -3.7 \times 10^{-3} & 0.02 \\ 0.01 & -1.1 \times 10^{-3} \end{bmatrix} \end{cases}$$

$$\begin{cases} A_2 = \begin{bmatrix} 0.91 & 4.6 \times 10^{-4} & 0.09 & 4.9 \times 10^{-3} \\ 4.9 \times 10^{-3} & 0.94 & -0.01 & 0.12 \\ -0.07 & 5.2 \times 10^{-3} & 0.9 & 0.03 \\ -0.04 & -2.0 \times 10^{-3} & 6.0 \times 10^{-3} & 0.91 \end{bmatrix} \\ B_2 = \begin{bmatrix} 5.3 \times 10^{-3} & -2.6 \times 10^{-5} \\ -5.1 \times 10^{-3} & 4.0 \times 10^{-3} \\ -3.0 \times 10^{-6} & 0.02 \\ 0.02 & 2.4 \times 10^{-4} \end{bmatrix} \end{cases}$$

$$\begin{cases} A_3 = \begin{bmatrix} 0.9 & 4.7 \times 10^{-3} & 0.22 & 0.02 \\ -2.3 \times 10^{-6} & 0.95 & -2.1 \times 10^{-3} & 0.05 \\ -0.02 & 8.2 \times 10^{-3} & 0.81 & -0.04 \\ -7.9 \times 10^{-3} & 0.01 & -4.6 \times 10^{-3} & 0.89 \end{bmatrix} \\ B_3 = \begin{bmatrix} -8.1 \times 10^{-4} & -1.7 \times 10^{-3} \\ 8.4 \times 10^{-5} & 4.5 \times 10^{-3} \\ 1.2 \times 10^{-3} & 0.02 \\ 0.01 & -4.8 \times 10^{-4} \end{bmatrix} \end{cases}$$

$$\begin{cases} A_4 = \begin{bmatrix} 0.91 & -3.8 \times 10^{-4} & 0.09 & -0.01 \\ -0.37 & 0.92 & 0.33 & 0.35 \\ -0.02 & -6.6 \times 10^{-4} & 0.78 & 4.5 \times 10^{-3} \\ 0.16 & -4.9 \times 10^{-3} & -0.21 & 0.8 \end{bmatrix} \\ B_4 = \begin{bmatrix} 5.1 \times 10^{-3} & -3.7 \times 10^{-4} \\ 1.2 \times 10^{-4} & 7.5 \times 10^{-3} \\ -3.5 \times 10^{-4} & 0.01 \\ 0.01 & -2.0 \times 10^{-3} \end{bmatrix} \end{cases}$$

$$\begin{cases} A_5 = \begin{bmatrix} 1.0 & 0.02 & 0.37 & -0.25 \\ -0.02 & 0.9 & -0.09 & 0.21 \\ -0.11 & -0.01 & 0.71 & 0.15 \\ 1.7 \times 10^{-3} & -1.6 \times 10^{-3} & 0.03 & 0.88 \end{bmatrix} \\ B_5 = \begin{bmatrix} 3.7 \times 10^{-3} & -2.6 \times 10^{-4} \\ -1.7 \times 10^{-3} & 2.6 \times 10^{-3} \\ -5.0 \times 10^{-3} & 0.01 \\ 8.3 \times 10^{-3} & -3.7 \times 10^{-3} \end{bmatrix} \end{cases}$$

$$\begin{cases} A_6 = \begin{bmatrix} 0.87 & -4.8 \times 10^{-3} & 0.13 & 0.04 \\ -0.07 & 0.9 & 0.01 & 0.37 \\ 8.0 \times 10^{-3} & -5.4 \times 10^{-3} & 0.79 & -0.1 \\ 0.01 & -4.7 \times 10^{-4} & -0.04 & 0.78 \end{bmatrix} \\ B_6 = \begin{bmatrix} 5.3 \times 10^{-3} & -9.5 \times 10^{-5} \\ 3.7 \times 10^{-4} & 4.1 \times 10^{-3} \\ -7.4 \times 10^{-4} & 0.02 \\ 0.01 & -1.9 \times 10^{-4} \end{bmatrix} \end{cases}$$

Figure 3.6 reports the confidence level that reaches the desired value of $th = 0.9$ after 105 iterations.

Finally, for the sake of completeness, a similar set set of experiments have been performed to evaluate the effectiveness of Ω with respect to the exact polytopic linear description Ω_{exact},

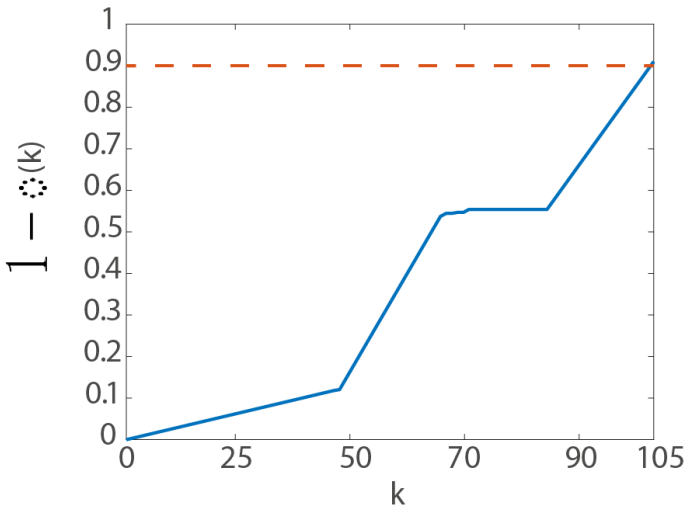

Figure 3.6: Four tanks model - Confidence level

that is derived e.g., in [88], and characterized by the following vertices:

$$[A_1^{exact}, B_1^{exact}] = \begin{bmatrix} 0.957 & 0 & 0.086 & 0 & 0.01 & 0 \\ 0 & 0.975 & 0 & 0.05 & 0. & 0.009 \\ 0 & 0 & 0.914 & 0 & 0 & 0.031 \\ 0 & 0 & 0 & 0.95 & 0.027 & 0 \end{bmatrix}$$

$$[A_2^{exact}, B_2^{exact}] = \begin{bmatrix} 0.98 & 0.000 & 0.027 & 0 & 0.01 & 0 \\ 0 & 0.988 & 0 & 0.05 & 0 & 0.009 \\ 0 & 0 & 0.973 & 0 & 0 & 0.031 \\ 0 & 0 & 0 & 0.95 & 0.027 & 0 \end{bmatrix}$$

$$[A_3^{exact}, B_3^{exact}] = \begin{bmatrix} 0.98 & 0 & 0.086 & 0 & 0.01 & 0 \\ 0 & 0.988 & 0 & 0.016 & 0 & 0.009 \\ 0 & 0 & 0.914 & 0 & 0 & 0.031 \\ 0 & 0 & 0 & 0.984 & 0.027 & 0 \end{bmatrix}$$

$$[A_4^{exact}, B_4^{exact}] = \begin{bmatrix} 0.987 & 0 & 0.027 & 0 & 0.01 & 0 \\ 0 & 0.992 & 0 & 0.016 & 0 & 0.009 \\ 0 & 0 & 0.973 & 0 & 0 & 0.031 \\ 0 & 0 & 0 & 0.984 & 0.027 & 0 \end{bmatrix}$$

As done for the Van der Pol oscillator $k = 10000$ random feasible initial conditions have been considered and the results checked by means of (3.33). As it can be observed in Table

3.3, the proposed **PE** algorithm gives rise to quite similar results as the exact model-based approach.

Table 3.3: Statistics of the relative error e_r^k

Mean	4.3621×10^{-3}
Max	0.031
Standard deviation	1.5743×10^{-4}

Chapter 4

Vehicular routing under traffic congestion and sustainability requirements: a DRL-based MPC approach

This *Chapter* resumes the works presented in [89] -[90]. The result is a sustainable routing algorithm for vehicle platoons operating on smart Urban Road Networks (URNs). The proposed approach makes use of the introduced tools: DRL and set-theoretic MPC. Here, the learning process aims at reducing traffic congestion and CO_2 emissions, whereas the MPC unit allows to adequately track the assigned path by using real-time traffic data.

4.1 Context of interest

In recent years, sustainable transportation issues have gained an increasing interest mainly for their capability to have deep socio-economic and environmental impacts in the human life across the world, see e.g., [91] -[92]. Accordingly, novel vehicle technologies and intelligent vehicle transportation systems play an important role because they help in reducing traffic undesired effects on the environment while improving the city long-term

sustainability [93]- [94].

In this scenario, increasing the level of autonomy in Intelligent Transportation Systems (ITS) gives rise to several advantages: time savings for drivers, energy savings for the environment and road safety improvement. In particular, travel time savings can be achieved by coordinated and connected vehicle configurations that can be more efficiently managed by means of *ad-hoc* control strategies for autonomous multi-vehicles systems [95].

4.1.1 Literature review

In the last decade, several innovative solutions have been developed with the aim of addressing environmental sustainability and traffic congestion specifications [96]. As a matter of fact, today's vehicles are also connected devices [97], [98] and Internet of Vehicles (IoV) paradigm has been successfully applied in high-tech roads scenarios as testified by [99], [100] and references therein. It is worth underlining that the IoV technology can provide vehicle services oriented to sustainability goals, such as information about traffic congestion and CO_2 emission levels.

By referring to the field of autonomous vehicle systems, the Vehicle Routing Problem (VRP) [101] is by now relevant as testified by the significant number of strategies conceived for its solution: ranging from optimization-based techniques [102], [103] to Machine Learning [15] -[104]. By considering the more recent advancements, an interesting contribution on the adopted practices for dealing with the sustainable urban vehicle routing problem is presented in [105]. There, a carefully analysis of different routing problem customizations, complying with mathematical models, sustainability levels and proposed solutions, has been provided. Moreover, the contribution [106] focuses on the so-called vehicle routing problem with backhauls (VRPB): a variant of the VRP that analyzes the vehicle efficiency *versus* the time interval of use. Conversely, the traffic-induced air pollution is addressed in [107], where an innovative scheme for detecting and counting vehicles has been proposed to mitigate the undesired effects of pollution.

By taking a look to the autonomous multi-vehicle systems, the coordination and control have gained more and more interest, see e.g., [108], [109]. In the last years most of the contributions were focused on analyzing stabilization [110], [111] and path following [112] issues, that are strictly connected with the computation of feasible state trajectories of

the leader vehicle and the formal definition of control schemes capable to confine the vehicle positions within secure bounds.

According to these premises, the MPC philosophy has acquired an increasing reputation in solving problems involving unavoidable physical constraints as testified by analyzing some pertinent and recent results. First, the contribution [113] where a distributed receding horizon controller is designed for regulating leader-follower (LF) configuration described by nonlinear dynamics. In [114], the authors propose a distributed MPC for multi-vehicle systems subject to nonholonomic constraints. Moreover, platoons of constrained nonlinear vehicles are managed by resorting to the concept of γ-gain stability, while state trajectories tubes of multi-vehicle systems are characterized in terms of high-order polynomials. Finally, the contributions [115] and [116] propose set-theoretic distributed MPC schemes for addressing formation requirements of LF systems.

4.1.2 Main contribution

The main aim of this chapter is to propose new insights in the field of path planning for constrained autonomous vehicles moving in the cluttered environments of urban road networks. Here, the attention is devoted to address the Sustainable Urban Vehicle Routing (SUVR) problem. To this end, the guideline is to make the resulting control architecture scalable, flexible and computational low demanding in order to be capable of addressing time-varying operating scenarios along the path. From a methodological perspective two aspects deserve particular attention:

1. the computation of routing decisions for mitigating traffic congestion phenomena and reducing CO_2 emissions;

2. a constrained control strategy in charge of adequately exploiting the routing decisions for sustainability purposes.

These considerations naturally lead to consider a distributed framework, necessary to fulfill scalability and flexibility specifications, where the first issue is addressed by resorting to an innovative DRL scheme, while the point 2) is solved by means of a MPC approach adequately designed to reduce as much as possible computational loads pertaining to the underlying optimizations.

To comply with this reasoning, the two approaches are combined into a single distributed control architecture where the capability of the DRL scheme to derive routing decisions by mitigating the undesired traffic effects allows to drive safely the vehicles until the assigned target without any occurrence of vehicle collisions. On the control side, this translates into the formalization of local MPC controllers that leverage the receding horizon philosophy and the platoon topology. Most computations are performed by the leader, while the followers rely on information propagated along the vehicle chain and set-membership conditions to reduce their computational complexity. Besides this, the periodic RL activation is the job of the leader while the rest of platoon only accesses and evaluates the reward function for routing decision purposes. Further, it is worth remarking that the platoon configuration allows to reduce fuel consumption because the followers' air resistance can be mitigated, see [117] and references therein.

4.2 Problem Formulation

In order to represent a real city scenario - as the one in figure 4.1 - , three definitions are useful:

- a *junction* is where two or more roads meet;
- a *link* (or *edge*) is the portion of road between two junctions;
- an *Urban Road Network (URN)* is a collection of links and junctions.

The links are assumed to be M.

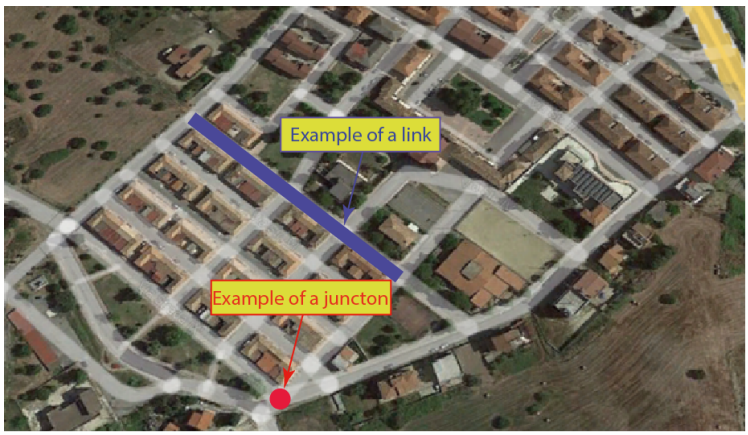

Figure 4.1: An example of URN

Consider a team of autonomous vehicles described by the following state space descriptions

$$AV_i : \quad x^i(t+1) = A^i x^i(t) + B^i u^i(t) \quad , i = 1, \dots, N \tag{4.1}$$

with $x^i \in \mathbb{R}^n,\ n = 4$, accounting for the state of AV_i (composed by the x-y position p^i and by the x-y velocity v^i). It is assumed that the structure of (4.1) includes integral actions in charge to ensure that any planar position is an equilibrium under zero velocity and $u^i(t) = 0$.

Figure 4.2: The leader-follower topology

The vehicles are topologically organized as a platoon, see figure 4.2. This formation prescribes the satisfaction of the constraint:

$$d_{min} < ||p^{i-1}(t) - p^i(t)|| < d_{max} \quad , i = 2, \dots N_1 \, , \forall t \tag{4.2}$$

Notice that d_{min} accounts for the geometrical dimension of the vehicles and the minimum safe distance between them, while d_{max} refers only to the maximum distance at which the vehicles should still be considered a platoon.

The vehicles operate in a URN, where each vehicle has to select the more adequate action when approaching a junction. To this end, the so-called decision zone is introduced

$$DZ^j : \ \mathcal{H}^j p > g^j \, , \quad p \in \mathbb{R}^2 \, , \quad j = 1, \dots, M \tag{4.3}$$

with $\mathcal{H}^j$ and g^j characterizing the hyper-plane associated to the j-link.

For the sake of comprehension consider figure 4.3. In this sketch, the dashed green zones account for those road constraints where a vehicle can move, while dotted cyan areas are the DZ for the two considered links.

Then, the vehicle routing problem can be straightforwardly defined in terms of a path search on a connected graph, see the illustrative sketch in figure 4.4.

Perception capabilities: the vehicles are equipped with a perception module capable of detecting obstacles within a pre-specified radius $R_{vis} > 0$. Let R^c_{min} be the minimum curvature radius of the given vehicle, then the perception module is such that the field of

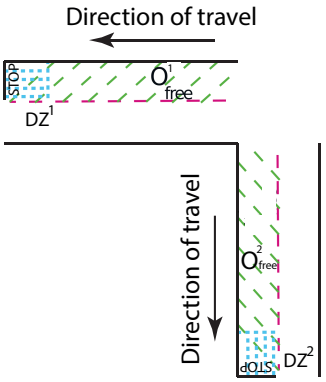

Figure 4.3: O^1_{free} and O^2_{free} (dashed green zones) account for obstacle-free regions on the 1−st and 2−nd links, respectively.

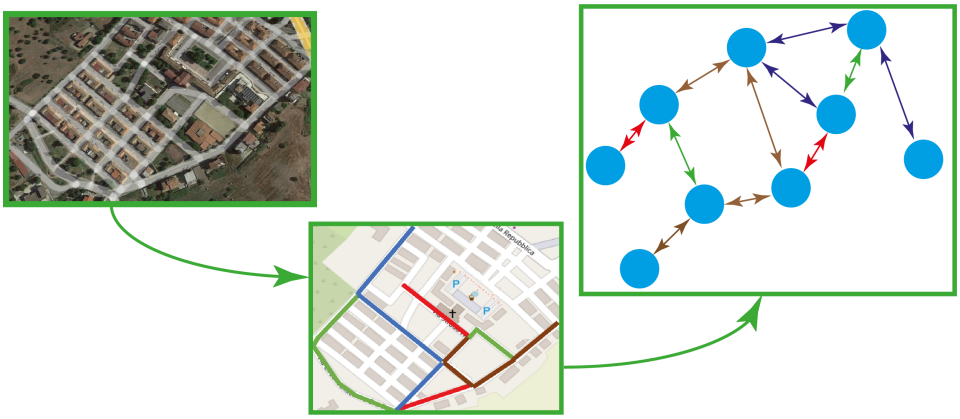

Figure 4.4: A simple example of conversion of a real city map into a decision graph

view is 360^o and $R_{vis} > R^c_{min}$. In this respect, the ball centered at the current agent planar position $\mathcal{B}(p^i(t), R_{vis})$, $\forall i = 1, \ldots, N$ - where N is the number of autonomous vehicles -, represents the detected region, as shown in figure 4.5.

The problem of interest can be stated as follows.

Autonomous Vehicles in Urban Environments (AV-UE): *Given admissible initial condition $x(0) = x_{in}$, a target x_{fin}, and a group of N autonomous vehicles, derive a path planning strategy complying with the following requirements:*

- *the team is driven from $x(0)$ towards x_{fin},under constraints fulfillment;*
- *traffic congestion events are mitigated as much as possible;*
- *CO_2 emissions are kept under a prescribed threshold.*

□

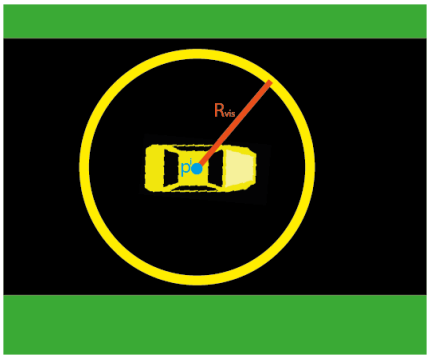

Figure 4.5: A representation of the detected region $\mathcal{B}(p^i(t), R_{vis})$

To formally address the **AV-UE** problem, the following assumptions are made.

Assumption 4 *An admissible path between $x(0)$ and x_{fin} exists.*

Assumption 5 *Each vehicle is aware about the traffic density along URN edges.*

4.3 An overview of the proposed solution

The **AV-UE** problem is addressed by conceiving a distributed scheme incorporating into a single framework the features of DRL and MPC methodologies. Specifically, at each junction the output of the DRL algorithm, namely the routing decision, is exploited as the set-point of the MPC unit, see the overall architecture of figure 4.6. It is composed of three main elements: the **Path Planner** (i.e., the DRL unit), the **Reference Generator** and the **Controller** defined according to the vehicles $AV = \{AV_i\}_{i=1}^{N}$.

At each t, the following tasks are performed:

- from the autonomous vehicles AV_i, the **Path Planner** acquires vehicle position coordinates, the associated edge on the graph and the edge vehicle density;
- the **Path Planner** computes the routing decision $a^i(t)$ to be applied by AV_i at the next junction;
- then, the **Reference Generator** generates the reference $z^i(t)$ according to $a^i(t)$ to be used by the controller;
- hence, the Receding Horizon Control (RHC) units, namely **MPC Controllers**, compute feasible inputs $u^i(t)$;

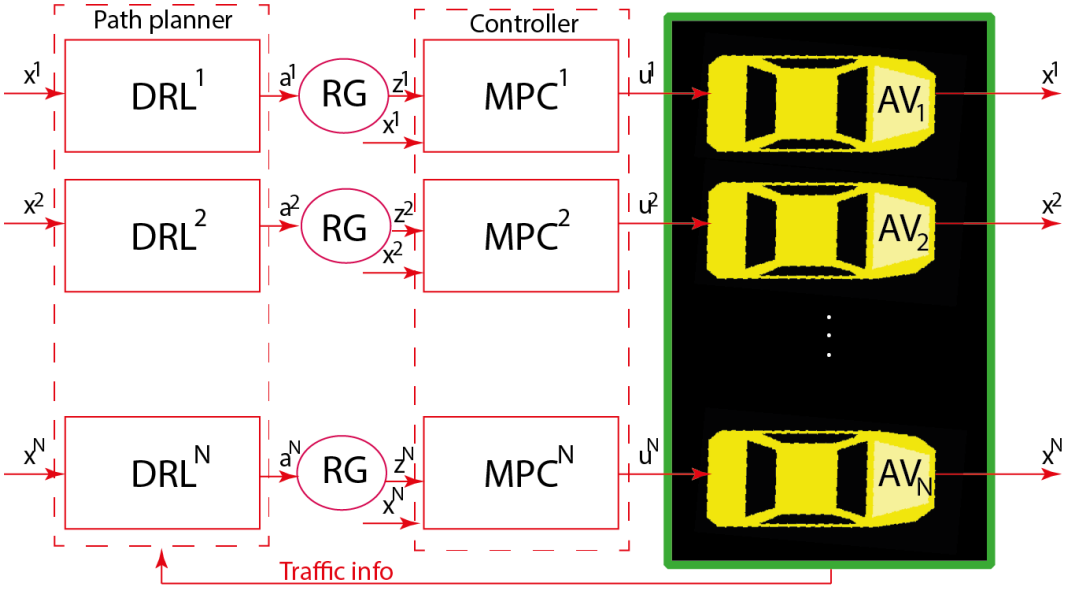

Figure 4.6: Proposed control architecture

- finally, the vehicles AV_i receive the control inputs and send their data to the **Path Planner**.

4.4 Reinforcement Learning scheme

In the sequel, a distributed deep reinforcement learning algorithm is developed with the aim of addressing the routing decision problem. By resorting to the Q-learning methodology - see chapter 2 -, a multi-agent reinforcement learning description is considered:

$$< AV, \mathcal{S}, \mathcal{A}, r(\cdot) > \tag{4.4}$$

where

- $AV = \{AV_1, \ldots, AV_N\}$ is the set of involved autonomous vehicles (agents);
- $\mathcal{S} := \{s^i\}_{i=1}^N$ is the set of RL states (consisting, for each AV_i, of the edge j where it is driving, the vehicle density there and the vehicle densities of the neighbor edges);
- $\mathcal{A}$ is a finite set of integer numbers accounting for the possible vehicle actions, see the example in figure 4.8;
- $r(\cdot)$ is the reward function.

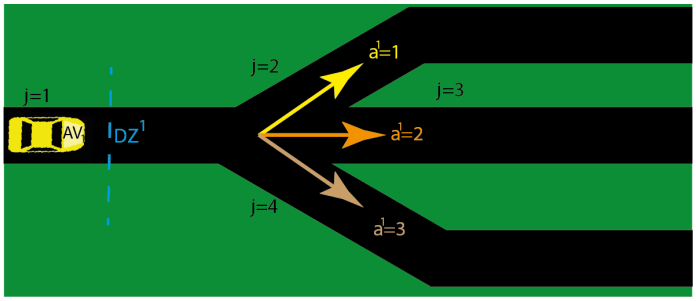

Figure 4.7: A representation of the meaning of $\mathcal{A}$

Let t_{dec} be the decision time (or trigger time), i.e. the time instant when an autonomous vehicle enters the DZ, where the RL unit starts, see figure 4.3. Here, the following reward function has been used:

$$\begin{aligned} r^i(t_{dec}+\Delta t_{dec}(t)) = \ & \Psi\left(\bigcup_k s^k(t_{dec}+\Delta t_{dec}(t))\right)+ \\ & +\psi^i_{goal}(t_{dec}+\Delta t_{dec}(t))+\psi^i_{wt}(t_{dec}+\Delta t_{dec}(t)), \quad i=1,\dots N \end{aligned} \tag{4.5}$$

where $\Delta t_{dec}(t) := t - t_{dec}$ with $x^i(t_{dec}) \in (\mathcal{X}^i_c)^{j_1}$ and $x^i(t) \in (\mathcal{X}^i_c)^{j_2}, j_1 \neq j_2$. Notice that the reward (5.13) is computed only when the vehicle AV_i has effectively performed the imposed action. Essentially, the meaning of formula (5.13) can be summarized as follows:

- $\Psi\left(\bigcup_k s^k(t_{dec}+\Delta t_{dec}(t))\right)$ is called global reward because it is the same for all the involved vehicles. It takes care of traffic congestion phenomena occurring over the URN that in turn is evaluated by

$$\Psi\left(\bigcup_k s^k(t_{dec}+\Delta t_{dec}(t))\right) = \sum_{j=1}^{M} \frac{\#^j(t_{dec}) - \#^j(t_{dec}+\Delta t_{dec}(t))}{\bar{w}^j l^j} \tag{4.6}$$

  where $\#^j(t) \in \mathbb{Z}_+$ the number of vehicles on the link j, and $\bar{w}^j \in \mathbb{R}^+$ and l^j the average width and road length, respectively. Notice that the quantity $\frac{\#^j(t)}{\bar{w}^j l^j}$ represents the vehicle density of the $j-th$ link at each time instant t. Therefore, it is important to underline: lower traffic densities higher reward realizations.

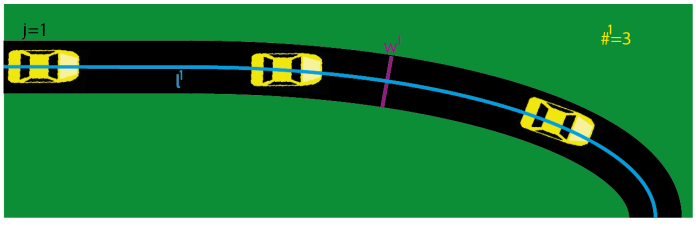

Figure 4.8: A representation of the quantities in formula (4.6)

- the heuristics $\psi^i_{goal}(t_{dec} + \Delta t_{dec}(t))$ accounts for the current longitudinal velocity of AV_i and the distance from the target:

$$\psi^i_{goal}(t) = \frac{||v^i(t)||}{||p^i(t) - p^i_{fin}||} \tag{4.7}$$

- $\psi^i_{wt}(t_{dec} + \Delta t_{dec}(t))$ is given by:

$$\psi^i_{wt}(t) = -WT^i(t) + f^i_{loops}(t) \tag{4.8}$$

where WT is the total waiting time of the vehicle AV_i since t_0 to t, while f^i_{loops} is a function that analyzes the route since the initial time instant t_0 until t and penalizes the possibly presence of loops (like turning around a roundabout). Essentially, ψ^i_{wt} takes into account those sequences of actions driving AV_i to queue scenarios or repetitive routes.

Figure 4.9 summarizes the meaning of the reward.

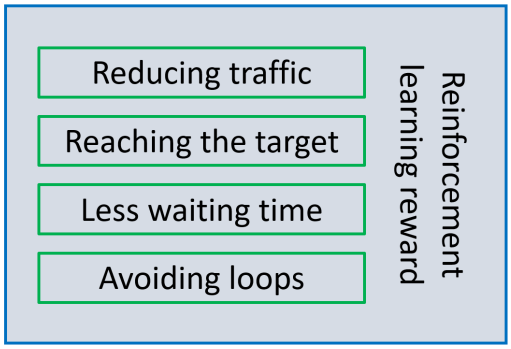

Figure 4.9: A representation of the reward of formula (5.13)

Notice that $\mathcal{Q}^i : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ refers to the $i-th$ vehicle and it is defined by means of a Neural Network (NN). Therefore, the used notation is $\mathcal{Q}^i(s^i, a^i; \theta^i)$ where $\theta^i \in \mathbb{R}^W$ the neural network weights.

It is worth to underline that the Q-function is initialized with random weights under the operating assumption that no information are available on the traffic behavior before the NN training. During the training, the trade-off between exploration and exploitation is managed by an epsilon-greedy policy.

4.4.1 Sustenaibility remark

Notice that the above reward function (5.13) does not explicitly take into account of emission phenomena. Then, in order to make the DRL process emission sustainable, the arguments proposed in [118] are exploited. Specifically, the idea is to support the AV^i action with a low-level decision maker that can modify the assumed decision according to the environment emission of a specific URN district. This because a vehicle AV^i can access a certain road link if a pre-assigned emission threshold is not overcome. Therefore, according to $s^i(t)$, if the action $\bar{a}^i$ reveals to be not applicable because of the emission threshold, the vehicle AV_i must comply with the following sub-optimal decision:

$$a^i(t_{dec}) = \arg \max_{a \in \mathcal{A} \setminus \{\bar{a}^i\}} \mathcal{Q}^i(s^i(t_{dec}), a; \theta^i) \tag{4.9}$$

Notice that if the action $\tilde{a}^i$ coming from (4.9) results to be again unusable, then a new sub-optimization has to be performed with the research space reduced as $\mathcal{A} \setminus \{\bar{a}^i, a^i\}$. Moreover, it is important to underline that, since the threshold is a function of the platoon length, longer platoons lead to lower thresholds for keeping emissions constant.

4.4.2 Algorithm

The above developments allow to write down the following computable algorithm.

DQL-Algorithm

Initialization

1: **Measure** $s^i(0), i = 1, \ldots, h$, from the URN;

2: **Initialize** $\mathcal{Q}(s^i, a; \theta)$, $i = 1, \ldots, h$;

At each t_{dec} *do*

1: **Select** an admissible action:

$$a^i(t_{dec}) = \begin{cases} rand(\mathcal{A}), & \text{with probability } \epsilon(t_{dec}) \\ \arg \max_{a \in \mathcal{A}} \mathcal{Q}^i(s^i(t_{dec}), a; \theta^i), & \text{otherwise} \end{cases}$$

2: **Check** if the action $\bar{a}^i = a^i(t_{dec})$ conducts AV_i into an edge with limitation on CO_2 emissions:

if FALSE go to STEP 3

else ask to the infrastructure if AV_i (and its followers) can enter inside such road:

if TRUE go to STEP 3

else pick a sub-optimal action. Go to STEP 2 with reduced research space

3: **Observe** the global reward $\Phi\left(\bigcup_k s^k(t_{dec}+\Delta t_{dec}(t))\right)$ and $s^i(t_{dec}+\Delta t_{dec}(t)), \forall i = 1,\dots,N$;

4: **Compute** the reward $r^i(t_{dec}+\Delta T_{dec}(t)),, \forall i = 1,\dots,N$;

5: **Update** $\theta^i, \forall i = 1,\dots,N$.

6: **Decrease** $\epsilon_\pi(t)$ as $\epsilon_\pi(t) = 0.9\epsilon_\pi(t)$

Until it is assumed that the temporal difference error $\delta_\pi(t)$ defined in (2.12) is definitely such that $|\delta_\pi(t)| < \epsilon_\delta$ where $\epsilon_\delta > 0$ is a given threshold scalar.

4.5 RHC units

In the sequel, and the following constraints imposed:

$$\begin{cases} p^i(t) \in \mathcal{X}_p^i := \{p^i \in \mathbb{R}^2 : p^{i\prime} p^i \leq (p_{max}^i)^2\} \subseteq \mathbb{R}^2 \\ v^i(t) \in \mathcal{X}_v^i := \{v^i \in \mathbb{R}^2 : v^{i\prime} v^i \leq (v_{max}^i)^2\} \subseteq \mathbb{R}^2 \\ u^i(t) \in \mathcal{U}^i := \{u^i \in \mathbb{R}^2 : u^{i\prime} u^i \leq (u_{max}^i)^2\} \subseteq \mathbb{R}^2 \end{cases} , \quad \forall t \geq 0, i = 1,\dots,N. \tag{4.10}$$

First of all, it is mandatory a formal characterization of feasible state trajectories complying with the prescriptions of **AV-UE** problem. By exploiting set-theoretic ideas [7], the problem to travel a link can be solved by computing sequences of overlapped positively invariant (PI) sets - see definition 1 - according to the multi-agent system dynamics (4.1)-(4.10), where the link-free region $\mathcal{O}_{free}^j$ is added so giving rise to the following state constraint description:

$$(\mathcal{X}_c^i)^j := \mathcal{X}^i \cap \mathcal{O}_{free}^j \; i = 1,\dots,N, \; j = 1,\dots,M, \tag{4.11}$$

See figure 4.10 for an example.

In the following,$(\mathcal{T}_0^i)_k^j \subset \mathbb{R}^n$ denotes the $k-th$ PI region for the dynamical evolution of the subsystem AV_i under the action of the state-feedback law:

$$u^i(t) = (K^i)_k^j x^i(t), \forall t \geq 0, \tag{4.12}$$

as usually, the apex j denoting the $j-th$ road.

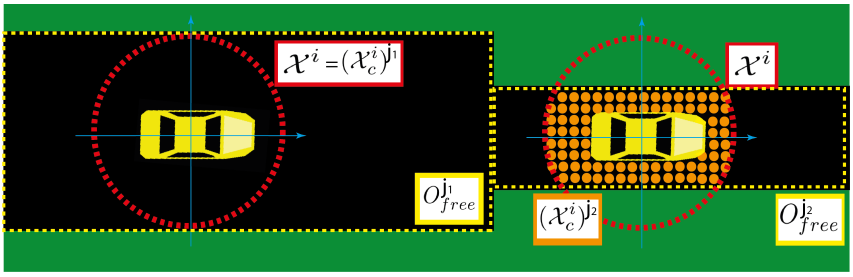

Figure 4.10: Example of intersection between state constraints and link-free region

The developed idea is illustrated in figure 4.11 and can be briefly summarized. A sequence of families of overlapped PI sets, each one associated to the used vehicle AV_i, the link j and the direction of travel, are off-line computed with the aim to define an inner convex approximation of the link-free region $\mathcal{O}_{free}^j$. According to this reasoning, feasible switchings along the pre-computed sets are always viable and, as a consequence, there exists an admissible path until the target, that in turn can be reached by using the routing decisions obtained via the DQL scheme. A simple example of this is depicted in figure 4.11 where, from a sequence of overlapped PI sets (the yellow ones), three different decisions can arise: turn right (red family), go straight (green family) or turn left (blue family).

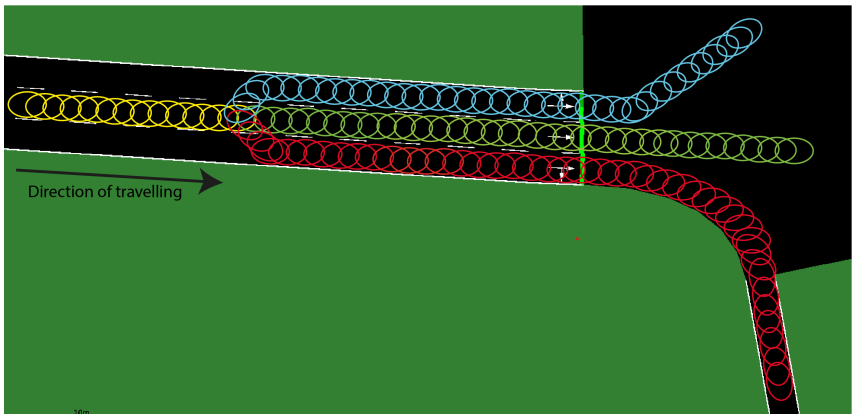

Figure 4.11: Example of families of overlapped sets

Within this context, it is clear that the family of PI sets must be properly designed by also taking into account of safe distances between the vehicles along the platoon. Therefore, a distance constraint is introduced:

$$d_{min} \leq dist(x^i, (\mathcal{T}_0^{i-1})_k^j) \leq d_{max} \tag{4.13}$$

with d_{min}, $d_{max} \in \mathbb{R}^+$ chosen in accordance with the current operating scenario. Figure

4.12 represents formula (5.23).

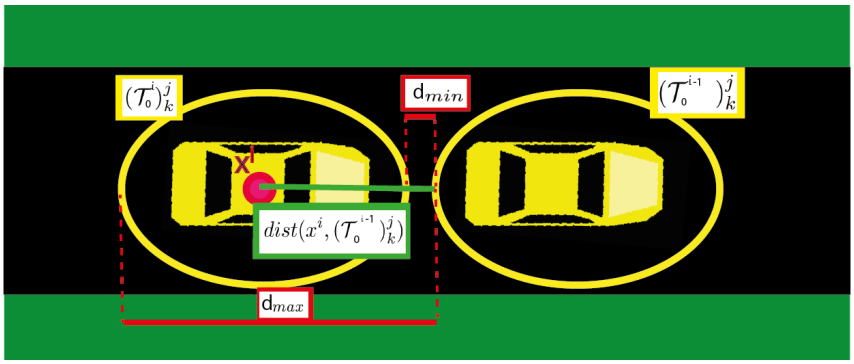

Figure 4.12: Illustration of formula (5.23)

Therefore, one has that

$$(\mathcal{T}_0^i)_k^j \cap (\mathcal{T}_0^{i-1})_k^j = \emptyset$$

In view of this analysis, the PI sets pertaining to the leader AV_1 are achieved as the solution of the following semi-definite programming problem:

DMPC-$\mathcal{P}_L$:

$$\min_{(K^1)_k^j, (\mathcal{T}_0^1)_k^j} J_\infty^1(t) \text{ s.t} \tag{4.14}$$

$$(A^1 + B^1(K^1)_k^j)(\mathcal{T}_0^1)_k^j \subset (\mathcal{T}_0^1)_k^j \subseteq (\mathcal{X}_c^1)^j \tag{4.15}$$

$$(K^1)_k^j(\mathcal{T}_0^1)_k^j \subset \mathcal{U}^1, \tag{4.16}$$

$$(\bar{x}^1)_k^j \in (\mathcal{T}_0^1)_{k-1}^j \cap (\mathcal{T}_0^1)_k^j \tag{4.17}$$

On the other, when a follower is concerned, one has:

DMPC-$\mathcal{P}_F$:

$$\min_{(K^i)_k^j, (\mathcal{T}_0^i)_k^j} J_\infty^i(t) \text{ s.t} \tag{4.18}$$

$$(A^i + B^i(K^i)_k^j)(\mathcal{T}_0^i)_k^j \subset (\mathcal{T}_0^i)_k^j \subseteq (\mathcal{X}_c^i)^j \tag{4.19}$$

$$(K^i)_k^j(\mathcal{T}_0^i)_k^j \subset \mathcal{U}^i, \tag{4.20}$$

$$(\bar{x}^i)_k^j \in (\mathcal{T}_0^i)_{k-1}^j \cap (\mathcal{T}_0^i)_k^j \tag{4.21}$$

$$d_{min} \leq dist(x^i, (\mathcal{T}_0^{i-1})_k^j) \leq d_{max}, \forall x^i \in (\mathcal{T}_0^i)_k^j \tag{4.22}$$

where $(\bar{x}^i)_k^j, i = 1, \ldots, h$ is the equilibrium selected at each $k-th$ iteration as follows

$$\begin{aligned} (\bar{x}^i)_k^j := \quad & \arg \max_{x \in (\mathcal{T}_0^i)_{k-1}^j} \|(\bar{x}^i)_{k-1}^j - x^i\|_2 \\ & \text{s.t. } \exists \bar{u}^i \in \mathcal{U}^i \,,\; x^i = (I_n - A^i)^{-1} B^i \bar{u}^i \end{aligned} \tag{4.23}$$

and

$$J^i_\infty(t) := \sum_{\tau=0}^{\infty} \left[\|x^i(t+\tau|t) - x^i_{fin}\|^2_{R^i_x} + \|u^i(t+\tau|t)\|^2_{R^i_u} \right], \quad i = 1, \ldots, N, \tag{4.24}$$

with $R^i_x > 0$ and $R^i_u \geq 0$ symmetric state and input weighting matrices, respectively. Figure 4.13 resumes the meaning of the constraints in **DMPC**-$\mathcal{P}_L$ and **DMPC**-$\mathcal{P}_F$.

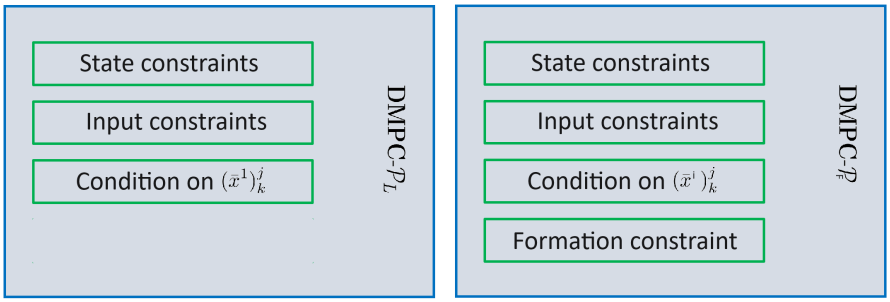

Figure 4.13: A representation of the constraints in **DMPC**-$\mathcal{P}_L$ and **DMPC**-$\mathcal{P}_F$

It is worth to underline that in virtue of (5.35), the set $(\mathcal{T}^i_0)^j_k$ is such that

$$(\mathcal{T}^i_0)^j_k \cap (\mathcal{T}^i_0)^j_{k-1} \neq \emptyset \tag{4.25}$$

As the on-line phase is concerned, this must guarantee that there always exist a control action capable to move the platoon amongst overlapped regions. The next results provides a solution to this requirement.

During the on-line operations, the following viable property along the overlapped families must be ensured.

Proposition 6 *Let $(\mathcal{T}^i_0)^j_k$ and $(\mathcal{T}^i_0)^j_{k+1}$ be two PI sets complying with (5.37). Let $x^i(t) \in (\mathcal{T}^i_0)^j_k$, $\tilde{x}^i := x^i - (\bar{x}^i)^j_{k+1}$ and $\tilde{u} := u - (\bar{u}^i)^j_{k+1}$ be the current state, the shifted state and the shifted input with respect to the equilibrium $((\bar{x}^i)^j_{k+1}, (\bar{u}^i)^j_{k+1})$, respectively. Then, the tracking one-step MPC problem*

$$\tilde{u}^i(t) := \arg\min_{u^i} \|A^i\tilde{x}^i(t) + B^i\tilde{u}^i\|^2_{((P^i_{\mathcal{T}})^j_{k+1})^{-1}} \tag{4.26}$$

such that

$$A^i x^i(t) + B^i u^i \in (\mathcal{T}^i_0)^j_{k+1}, \; u^i \in \mathcal{U}^i, \tag{4.27}$$

with $(P^i_{\mathcal{T}})^j_{k+1}$ the shaping positive definite matrix of $(\mathcal{T}^i_0)^j_{k+1}$, has a solution at each time instant t.

The proof of this result is available in [119].

As a consequence of the above discussion, the following distributed algorithm results.

DRL-MPC Algorithm - Vehicle $i-th$

Input: d_{min}, d_{max}, x^i_{in}, x^i_{fin};

Initialization-

1: **Compute** $\{(\mathcal{T}^1_0)^j_k, (K^1)^j_k\}$ by solving (5.24)-(5.28);

2: **Compute** $\{(\mathcal{T}^i_0)^j_k, (K^i)^j_k\}$ complying with (5.29)-(5.34);

3: **Derive** the equilibria by (5.35);

4: $k \leftarrow k+1$; goto **Step 1**;

On-line phase -

1: **Receive** $x^i(t)$;

2: **Determine** the routing decision $z^i(t)$;

3: **Solve** the optimization (4.26)-(4.27);

4: **Apply** $u^{i^*}(t|t)$;

5: $t \leftarrow t+1$ and goto **Step 1**;

The main properties of the **DRHC** Algorithm are collected in the next result.

Proposition 7 *Given x^i_{in} and $x^i_{fin}, i = 1, \ldots, N$, the* **DRL-MPC** *Algorithm always fulfills the imposed constraints, satisfies the* **AV-UE** *problem specs and guarantees asymptotic stability of the closed-loop system.*

Proof - The feasibility retention is achieved by showing the solvability of (4.26)-(4.27) at each sampling time.

By hypothesizing that one has an optimal solution for the vehicle AV^i at t and $x^i(t) \in (\mathcal{T}^i_0)^j_k$, at $t+1$ an admissible, though not optimal, command input can be derived because the state evolution $x^i(\cdot)$ driven by $u^i(\cdot) = (K^i)^j_k x^i(\cdot)$ indefinitely lies within $(\mathcal{T}^i_0)^j_k$. On the other hand, the optimal action $u^{i^*}(t|t)$ conducts AV^i within $(\mathcal{T}^i_0)^j_{k+1}$. Same arguments can be used for the other vehicles.

As the asymptotic closed-loop stability is concerned, a similar analysis can be straightforwardly pursued by taking care that, thanks to the routing decision $z^i(t)$, each vehicle converges to d^i_{fin} and the velocity $v^i(t)$ settles down as $t \to \infty$. □

4.6 An improved control strategy

In order to improve the performances of the autonomous vehicles, the topology has been supposed to be *time-varying*. Therefore, the main change in the control requirements is that all vehicles within each platoon here have the capability to decide to split the platoon into two, whenever different routing decisions arise. Conversely, two platoons can be combined into a single configuration only if the leader of the first platoon reconnects to the leaf node of the second one.

As in the previously presented control scheme, the **Controller** - see figure (4.6) - computes an admissible command $u^i(t)$ in a distributed receding horizon fashion, but in this case, each $i-th$ vehicle is equipped with a distributed MPC:

1. the leader (namely $i = 1$) implements a local RHC controller with the prediction horizon length $h_1 = 0$;

2. each follower $i = 2, \ldots, N$, uses an MPC controller with $h_i = i - 1$;

Nevertheless of the time-varying topology, the feasibility is preserved if the following actions are performed :

- Let $\bar{t}$ be the time instant when an $i-th, i > 1$, vehicle leaves the initial LF configuration - see figure (4.14)-. At $\bar{t}+1$ it becomes the leader of a new platoon and the prediction horizon is set to $h_i = 0$. If $0 < i < N$ (not leaf node), the prediction horizon length of all the followers along the LF tree is decremented by $i - 1$.

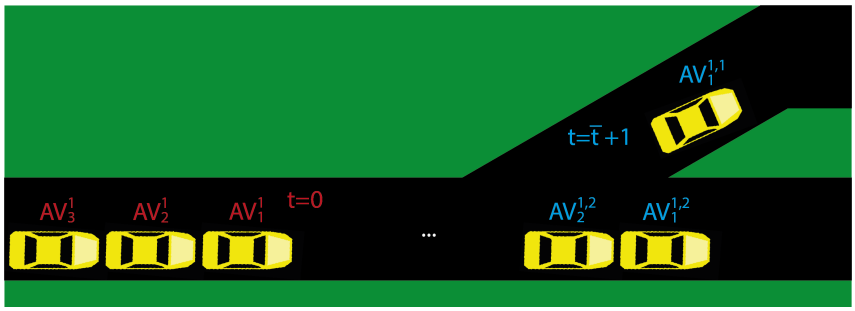

Figure 4.14: Illustration of platoon splitting

- Two platoons $LF^1 = \{AV_k^1\}_{k=1}^{N_1}$ and $LF^2 = \{AV_k^2\}_{k=1}^{N_2}$ join if the second platoon is added as the leaf node of the first configurations. The platoon LF^2 queues to LF^1

under the following reasoning: the prediction horizons of the local MPC controllers of the vehicles previously belonging to the second platoon are incremented of N_2.

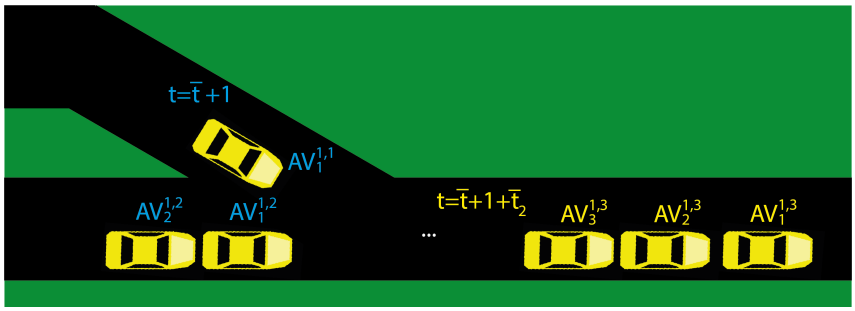

Figure 4.15: Illustration of platoon queuing

***Remark* 2** *In order to ensure that the LF formation is preserved, each vehicle should be aware of the position of its predecessor from the current time instant onward. Within an MPC framework, this prescribes that each element of the LF configuration informs followers about its future state trajectory (predicted sequence) in order to properly define the LF formation constraints within the time interval defined by the prediction horizon length.* □

Therefore, the -*information exchange* hypothesis is enforced: at each time instant t, each $i-th$ vehicle sends to $(i-1)-th$ its predicted future state trajectory.

***Remark* 3** *The need for time-varying LF configurations arises from the aim of improving as much as possible the control architecture flexibility. Leader switching is intended to mitigate traffic congestion phenomena through routing decisions and, as a result, enhance the vehicle formation mission.* □

In order to solve the **AV-UE** problem and guarantee asymptotic stability of the closed-loop system while allowing a time-varying topology, the following methodological questions:

1. Given an initial LF configuration, define local MPC problems for the N involved vehicles;

2. Characterize the conditions under which the platoon split is feasible;

3. Determine the conditions under which two platoons can be combined into a single LF configuration.

4.6.1 DMPC control units: single platoon

In the sequel the following notations is used. For any vehicle $i = 1, \ldots, N$, let the $k - th$ predicted state and predicted control within the horizon at time t be denoted by $x^i(t+k|t)$ and $u^i(t+k|t)$, respectively.

Let $\mathbf{x}^i(t) := \{x^i(t+k|t)\}_{k=0}^{h_i}$ and $\mathbf{u}^i(t) := \{u^i(t+k|t)\}_{k=0}^{h_i}$, $i = 1, \ldots, N$, be the predicted states and predicted controls within the control horizon, respectively. Moreover, over any prediction interval $[t+k,\ t+k+h_i]$ and for any $i-th$ vehicle the following further state trajectories are defined:

- the optimal state trajectory: $\mathbf{x}^{i^*}(t) := \{x^{i^*}(t+k|t)\}_{k=0}^{h_i}$, $i = 1, \ldots, N$;
- the assumed state trajectory: $\hat{\mathbf{x}}^i(t) := \{\hat{x}^i(t+k|t)\}_{k=0}^{h_i}$, $i = 1, \ldots, N$;

Notice that the sequence $\hat{\mathbf{x}}^i$ is transmitted to the follower $i+1$ which in turn hypothesizes that the $i - th$ vehicle will implement during the update prediction time interval $[t+k+1,\ t+k+h_i+1]$.

In order to formally define the optimal control problem underlying the MPC strategy, the following ingredients are required:

- Input sequence parametrization:

$$u^i(\cdot|t) = \begin{cases} u^i(t+k|t), & k = 0, \ldots, h_i - 1 \\ K^i x^i(t+k|t), & k \geq h_i \end{cases} \tag{4.28}$$

  with $K^i \in \mathbb{R}^{m \times n}$ a stabilizing and admissible state feedback law;

- Cost-to-go:

$$\begin{array}{l} J_i(x(t|t), x^i_{fin}, \mathbf{u}^i(t)) := \\ \displaystyle\sum_{k=1}^{h_i-1} \Big[\|x^i(t+k|t) - x^i_{fin}\|^2_{R^i_x} + \|u^i(t+k|t)\|^2_{R^i_u} \Big] + \|x^i(t+h_i|t) - x^i_{fin}\|^2_{R^i_{fin}} \end{array} \tag{4.29}$$

  where $R^i_x > 0$ and $R^i_u \geq 0$ are symmetric state and input weighting matrices and $R^i_{fin} \geq 0$.

- Terminal constraint:

$$x^i(t+h_i|t) \in \Xi^i \subset \mathbb{R}^n \tag{4.30}$$

  The pair (Ξ^i, K^i) is computed such that Ξ^i is a positively invariant region for the state evolutions of the closed-loop system.

It is necessary to deal with the fact that the LF formation moves towards x_{fin} : this prescribes that *the terminal condition is time-varying* and the related constraint must be accordingly modified. Moreover, the vehicle must take into account of routing decisions at each junction, so that the resulting action $a^i(t)$ comes to play when the terminal region is derived. First, the following conditions have to be satisfied for all involved vehicles:

$$x^i(t+h_i|t) \in \Xi^i(t-1) \cup \Xi^i(t) \tag{4.31}$$

where $\Xi^i(t)$ is computed such that

$$\bar{x}^i_{t-1} \in \Xi^i(t-1) \cap \Xi^i(t) \tag{4.32}$$

with $\bar{x}^i_{t-1}$ denoting an equilibrium point selected at the time instant $t-1$. Notice that the leader vehicle computes admissible sets $\Xi^1(t)$ complying with

$$\text{Proj}_d(\Xi^1(t)) \subseteq \mathcal{B}(p^1(t), R_{vis}), \forall t \geq 0. \tag{4.33}$$

As the followers are concerned, an admissible computation of the sets $\Xi^i(t)$, $i = 2, \ldots, N$, prescribes that each vehicle transmits to its direct follower $i+1$ the PI computed at the previous time instant, i.e. $\Xi^i(t-1)$

Then, one has that

- If $i = 1$ (leader), then $\Xi^1(t)$ is such that

$$\begin{cases} \Xi^1(t) := \arg\min\limits_{\Xi^1} \; dist(x^1_{fin}, \Xi^1) \\ \quad \text{subject to} \quad (4.32), (4.33) \end{cases} \tag{4.34}$$

- else

  - If $z^i(t) \equiv z^{i-1}(t)$ then

$$\begin{cases} \Xi^i(t) := \arg\min\limits_{\Xi^i} \; dist(\Xi^i, \Xi^{i-1}(t-1)) \\ \quad \text{subject to} \quad (4.32) \end{cases} \tag{4.35}$$

  - else

$$\begin{cases} \Xi^i(t) := \arg\min\limits_{\Xi^i} \; dist(x^i_{fin}, \Xi^i) \\ \quad \text{subject to} \quad (4.32) \end{cases} \tag{4.36}$$

Finally, it is mandatory to preserve the LF configuration at each time instant. In the present context, it is required that each follower vehicle must maintain a safe distance from its predecessor and neighbors. This translates into LF formation constraints in terms of desired separation and the desired relative bearing between the $i-th$ vehicle, its predecessor [120].

Hence, given the current state measurement $x^i(t|t) \in \mathcal{X}^i$ and the predecessor assumed state sequence $\hat{\mathbf{x}}^{i-1}(t)$ computed according to the following strategy:

$$\hat{\mathbf{x}}^{i-1}(t) = \begin{cases} x^{(i-1)^*}(t-1+k|t-1), & k = 1, \ldots, h_{i-1}; \\ (A^{i-1} + B^{i-1}K^{i-1})^{t-1+k} x^{(i-1)^*}(t-1+N_{i-1}|t-1), & k = h_{i-1}+1, \ldots, h_i. \end{cases} \tag{4.37}$$

the optimization problem for the $i-th$ follower, hereafter denoted as $\mathcal{P}_F^i(t)$, is

DMPC-$\mathcal{P}_F^i(t)$:

$$\begin{aligned} &\min_{\mathbf{u^i(t)}} \; J_i(x(t|t), x^i_{fin}, \mathbf{u}^i(t)) \\ &\text{subject to} \end{aligned} \tag{4.38}$$

$$x^i(t+k+1|t) = A^i x^i(t+k|t) + B^i u^i(t+k|t) \tag{4.39}$$

$$x^i(t+k|t) \in \mathcal{X}^i_j,\; k = 0, 1, \ldots, h_i - 1 \tag{4.40}$$

$$u^i(t+k|t) \in \mathcal{U}^i,\; k = 0, 1, \ldots, h_i - 1 \tag{4.41}$$

$$x^i(t+h_i|t) \in \Xi^i(t) \tag{4.42}$$

$$d^c_{min} \leq \|x^i(t+k|t) - \hat{x}^{i-1}(t+k|t)\| \leq d^c_{max},\; k = 0, 1, \ldots, h_i \tag{4.43}$$

where $d^c_{max} \in \mathbb{R}^+$ and d^c_{min} are guaranteed bounds to be chosen by the designer and depending on the LF formation requirements.

Figure 4.16 resumes the meaning of the constraints in **DMPC**-$\mathcal{P}_L$ and **DMPC**-$\mathcal{P}_F$.

Conversely, the **DMPC** optimization pertaining to the leader vehicle, named $\mathcal{P}_L(t)$, reduces to the off-line computation of an admissible pair (Ξ^1, K^1) which is then adequately shifted via (4.44)-(4.45) within the vision radius when no obstacles are detected. On the other hand, an obstacle occurrence prescribes the re-computation of (Ξ^1, K^1) via the following semidefinite programming (SDP) problem (see e.g. [23]):

DMPC-$\mathcal{P}_L(t)$:

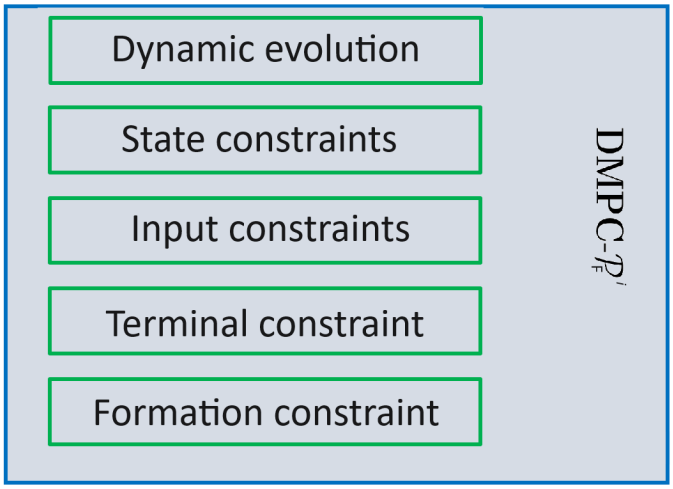

Figure 4.16: A representation of the constraints in **DMPC**-$\mathcal{P}_F^i$

$$\min_{K^1,\Xi^1} \sum_{k=t}^{\infty}\Big[\|x^1(t+k|t) - x_f^1\|^2_{R_x^1} + \|u^1(t+k|t)\|^2_{R_u^1}\Big] \tag{4.44}$$

subject to

$$x^1(t+k+1|t) = A^1x^1(t+k|t) + B^1u^1(t+k|t),\ \forall k \tag{4.45}$$

$$u^1(t+k|t) = K^1x^1(t+k+1|t) \in \mathcal{U}^1,\ \forall k \tag{4.46}$$

$$x^1(t+k|t) \in \mathcal{X}_j^1,\ \forall k \tag{4.47}$$

$$\bar{x}_{t-1}^1 \in \Xi^1(t-1) \cap \Xi^1(t) \tag{4.48}$$

Notice that it does not involve the satisfaction of (4.43) because this vehicle leads the team, while the LF configuration is exclusively preserved by its follower nodes.

***Remark* 4** *Although l.h.s. of (4.43) gives rise to non-convex constraints, they are addressed by straightforwardly adapting the convexification results of [121] (Proposition 1, pg. 292).* □

4.6.2 Platoon splitting and queuing

Along the road links, the initial platoon can split in two or more sub-LF configurations and two any platoons (even singletons) can regroup.

As the platoon splitting is concerned and for the sake of clarity, we shall consider that at a certain time instant $\bar{t}$, the routing decision sequence $\{a^1(\bar{t}), a^2(\bar{t}), \ldots, a^N(\bar{t})\}$ is such that $a^{i_1}(\bar{t}) \neq a^{i_2}(\bar{t})$ for $i_1 \neq i_2$. As a consequence two LF configurations, namely LF^1 and LF^2, result. Generally speaking, this leads to the following topological scenario. Notice

that the vehicles belonging to the LF^1 configuration keep the same control horizon lengths $h_i = i-1, i = 1\ldots, N_1$, while for the new platoon LF^2 one has that:

$$h_{N_2} = 0,\ h_i = i - N_2,\ i = N_2 + 1, \ldots, N.$$

As a consequence of such an event, it is assumed that an adequate *re-numbering* procedure is implemented so that the new platoons LF^1 and LF^2 are renamed as follows $\{AV_i^1\}_{i=1}^{N_1}$ and $\{AV_i^2\}_{i=1}^{N_2}$.

Conversely, two platoons could regroup according to the following reasoning. The idea is that LF^1 is added to LF^2 or *viceversa* at the leaf node of the LF chain. As a consequence, the associated **DMPC** controllers will be implemented over the horizon of lengths $h_i = level(leaf) + i$, $i = 1, \ldots, N_2$. The main difficulty with such an aim relies on the capability of the external platoon to detect leaf node of the platoon to regroup. Without any *a-priori* information about the roles and positions of vehicles within the LF chain, external vehicles face significant limitations. To address this, the following technical approach can be employed.

Statement 1 *At each time instant t, each $i-th$ vehicle of the LF^1 formation makes available a data packet whose structure is the form $Count^i = \{Count^i.child(i), Count^i.h_i\}$, where the integer $child(i)$ accounts for its follower, i.e. $child(i) = 0$ implies no follower, and the horizon length h_i. Such a packet can be acquired by another $i_2 - th$ vehicle if there exists a time instant $\bar{t}$ such that*

$$d_{min}^c \leq \|p^{i_2}(\bar{t}) - p^i(\bar{t})\|_2 \leq d_{max}^c \tag{4.49}$$

Then, the LF^2 platoon is able to regroup to LF^1 if condition (4.49) becomes valid for some $i-th$ vehicle for which the $Count^i.child(i) = 0$. This means that the AV_1^2 vehicle will be a leaf node of the LF^1 formation and, as a consequence, communication facilities will be enabled and a new LF configuration will come out.

The key question to investigate is if splitting and queuing operations allow to preserve the overall feasibility property of the **DMPC** strategy. To this end the following arguments are considered:

- *splitting mode* - the robust Bellman equation with constraints [122] to prove that if at the time instant t there exists a solution to $\mathcal{P}_F^i(t)$ with $h_i > 0$, then at the next time instant $t+1$ an admissible solution there exists for the same problem with $h_i - k$, $k > 0$;

- *queuing mode* - Positively invariant regions computed under time-delay scenarios to prove that if at the time instant t there exists a solution to $\mathcal{P}_{(F^2)^i}(t)$ (respectively $\mathcal{P}_{(L^2)^i}(t)$) with $h_i \geq 0$, then at the next time instant $t+1$ an admissible solution there exists for the same problem with $h_i + k$, $k > 0$.

Splitting mode

Let $J^*_{N_i}(x(t|t)) = J_i(x(t|t), x^i_f, \mathbf{u}^{i^*}(t))$ be the minimum of the cost at the optimal solution $\mathbf{u}^{i^*}(t)$ of $\mathcal{P}^i_F(t)$. The Bellman optimality principle states that an optimal sequence $\mathbf{u}^{i^*}(t)$ is such that: given $x^i(t+k|t)$ along the optimal system trajectory (by applying the input sub-sequence $\{u^{i^*}(t|t), \ldots, u^{i^*}(t+k-1|t)\}$, then the subsequent input sequence $\{u^{i^*}(t+k-1|t), \ldots, u^{i^*}(t+N_i-1|t)\}$ is optimal for the cost-to-go over the horizon $[k, h_i]$. Moreover, the *Bellman equation* is:

$$J^*_{N_i}(x(t|t)) = \min_{\mathbf{u^i(t)}} \left\{ \sum_{i=0}^{k-1} \left[\|x^i(t|t)\|^2_{R^i_x} + \|u^i(t|t)\|^2_{R^i_u} \right] + J^*_{N_i-k}(x(t|t)) \right\} \tag{4.50}$$

According to the above developments, the following result comes out.

Proposition 8 *Given the optimal solution $\mathbf{u}^{i^*}(t)$ of the optimization $\mathcal{P}^i_F(t)$. Then at the next time instant $t+1$, there always exists an admissible solution of $\mathcal{P}^i_F(t+1)$ with the control horizon equals $h_i - k$.*

Proof - The optimal solution $\mathbf{u}^{i^*}(t) = \{u^{i^*}(t|t), \ldots, u^{i^*}(t+h_i-k|t)\}$ can be rewritten as $\mathbf{u}^{i^*}(t) = \{u^{i^*}(t|t), \ldots, u^{i^*}(t+k-1|t)\} \otimes \{u^{i^*}(t+k+1|t), \ldots, u^{i^*}(t+h_i-1|t)\}$, where $\otimes$ denotes the concatenation operator between sequences. By applying the sequence of virtual moves $\mathbf{u}^{i^*}(t) = \{u^{i^*}(t|t), \ldots, u^{i^*}(t+k-1|t)\}$ the following k-step state predictions are achieved:

$$x^i(t+k) = (A^i)^k x^i(t) + \sum_{j=0}^{k-1} (A^i)^{k-j-1} B^i u^{i^*}(t+k-1|t)$$

which exactly match the k-step predicted states $x^{i^*}(t+j|t), j = 1, \ldots, k-1$. Then in virtue of both the Bellman optimality principle and (4.50), the following input sequence of $h_i - k$ length

$$\mathbf{u}^i(t+1) = \{u^{i^*}(t+1+k|t), \ldots, u^{i^*}(t+h_i-1|t), \overbrace{K^i x^{i^*}(t+h_i-1+j|t)\}}^{j=1,\ldots,k}$$

results to be always admissible, though not optimal, for $\mathcal{P}^i_F(t+1)$. □

Queuing mode

Let $\bar{t}$ the time instant when the platoon has been split for the first time, the terminal regions $\Xi^i(t), i = 1, \ldots, N$, $\forall t \geq \bar{t}$, must be computed by assuming that a constant time delay τ occurs on the vehicle dynamics. This is instrumental to ensure that at each time instant t the computed command $u^i(t)$, $i = 1, \ldots, N$, can be consecutively applied, i.e. $u^i(t+k) = u^i(t), k = 1, \ldots, \tau$. In the present context, the delay is imposed equal to the number of followers of the initial platoon

$$\tau := N - 1 \tag{4.51}$$

and this always allows to provide an admissible sequence when the platoon LF^1 and LF^2 regroup. According to this reasoning, first the positively invariant regions are derived by resorting to the so-called **Delay Dependent (DD)** time-delay scenario and the prescribed constraints (4.10). To this end, by considering the state-feedback control law

$$u^i(t) = K^i \, x^i(t-\tau) \tag{4.52}$$

and the regulated plant

$$x^i(t+1) = A^i x^i(t) + B^i \, K^i \, x^i(t-\tau), \tag{4.53}$$

the delayed system technicalities of [123][124] are hereafter used. The feedback control law (4.52) stabilizes the plant and satisfies the prescribed constraints if the following matrix inequalities in the objective variables K^i, Q^i_τ, Ψ^i_τ, R^i_τ and $\bar{\tau}$, are satisfied

$$\begin{bmatrix} E'_\tau \, Q^i_\tau \, E_\tau - \mathcal{S}^i_\tau & 0 & \mathcal{A}^{i\prime}{}_\tau \, Q^i_\tau \\ 0 & \tau(R^i_\tau + \Psi^i_\tau) & \mathcal{B}^{i\,\prime}_\tau \, Q^i_\tau \\ Q^i_\tau \, \mathcal{A}^i_\tau & Q^i_\tau \, \mathcal{B}^i_\tau & Q^i_\tau \end{bmatrix} \geq 0, \tag{4.54}$$

$$\begin{bmatrix} \bar{u}^2 \, E'_\tau \, q^i_\tau \, E_\tau & \begin{bmatrix} K^{i\prime}{}_\tau \\ 0 \end{bmatrix} \\ \begin{bmatrix} K^i_\tau & 0 \end{bmatrix} & I \end{bmatrix} \geq 0 \tag{4.55}$$

$$E'_\tau Q^i_\tau E_\tau \geq \frac{1}{\bar{x}^2_j} \tag{4.56}$$

where $Q^i_\tau = Q^{i\prime}{}_\tau \geq 0$, $R^i_\tau = R^{i\prime}{}_\tau \geq 0$, $\Psi^i_\tau = \Psi^{i\prime}{}_\tau \geq 0$ $E_\tau = \mathrm{diag}\{I, 0\}$, $\mathcal{S}^i{}_\tau := \mathrm{diag}\left\{0, \bar{\tau}\,\left(R^i_\tau + \Psi^i_\tau\right)\right\}$ and

$$\mathcal{A}^i_\tau = \begin{bmatrix} I & I \\ A^i - I + B^i K^i & -I \end{bmatrix}, \; \mathcal{B}^i_\tau = \begin{bmatrix} 0 \\ -B^i K^i \end{bmatrix},$$

Hence, the ellipsoidal set

$$\Xi^i := \{x^i \in \mathbb{R}^n \,|x^{i\prime} P_\tau^i x^i \leq 1\}, P_\tau^i = P^{i\prime}{}_\tau > 0,\, P_\tau^i = \left(Q_\tau^i\right)^{-1}$$

arising from the inequalities (4.54), (4.55) is a positively invariant region for the closed-loop state evolution (4.53) complying with (4.10), viz. $\Xi^i \subset \mathcal{X}^i$ and $K^i \Xi^i \subset \mathcal{U}^i$.

Hence, the following result holds true.

Proposition 9 *Given the optimal solution* $\mathbf{u}^{i^*}(t)$ *of the optimization* $\mathcal{P}_F^i(t)$ *(resp.* $\mathcal{P}_L^i(t)$*), with* $\Xi^i(t)$ *computed satisfying (4.54)-(4.56). Then at the next time instant* $t+1$*, there always exists an admissible solution of* $\mathcal{P}_F^i(t+1)$ *(resp.* $\mathcal{P}_L^i(t+1)$*) with the control horizon equals* $h_i + k$, $k \leq N-1$.

The *proof* of this result is available in [125].

4.6.3 The distributed DRL-MPC algorithm *with platoon splitting and queuing*

The developments of the previous sections are here collected to define a computable algorithm based on the availability of an urban map network characterized in terms admissible directions of travel for each road link.

DRL-MPC-Algorithm *with platoon splitting and queuing* **- Vehicle** $i-th$

Input: d_{min}, d_{max}, $x^i(0)$, x_{fin}^i;

Output: $u^{i^*}(t|t)$;

Initialization: $Splitting\_mode := false$;

1: **if** $i \geq 2$ **then receive** $\hat{\mathbf{x}}^{i-1}(t)$ and $a^{i-1}(t)$ and $Splitting\_mode$;

2: **end if**

3: **Acquire** $x^i(t)$;

4: **if** $x^i(t+h_i|t) \in DZ^j$ for some $j \in \{1,\ldots,M\}$ **then update** the reward $r^i(t)$ via the $\mathbf{DRL}^i$ unit;

5: **Generate** the reference $z^i(t)$ by using the $\mathbf{RG}^i$ device;

6: **end if**

7: **if** $i = 1$ **then**

updates $\Xi^1(t)$ via (4.34)

8: **else**

9: **if** $a^i(t) \equiv a^{i-1}(t)$ **then** updates $\Xi^i(t)$ via (4.35)

10: **else** *Splitting_mode* := *true*; ▷ Splitting mode

11: **Renumber** the resulting platoons by assigning $1 \rightarrow i$ and $h_i = 0$;

12: computes $\Xi^i(t)$ by solving inequalities (4.54)-(4.56)

13: **end if**

14: **end if**

15: **if** *Splitting_mode* == *true* **then**

updates $\Xi^i(t)$ via (4.36)

16: **end if**

17: **if** $i = 1$ **then solve** the optimization (4.44)-(4.48)

18: **else solve** (4.38)-(4.43) ;

19: **end if**

20: **if** a packet $Count^{i1}$ has been received **then set** $h_i = h_{i1} + 1$; ▷ Queuing mode

21: *Splitting_mode* := $false$

22: **end if**

23: **Apply** $u^{i^*}(t|t)$;

24: $t \leftarrow t + 1$ and goto **Step 1**;

Recursive feasibility and asymptotic stability of the **DRL-MPC** Algorithm are stated in the following proposition.

Proposition 10 *Let* $x^i(0)$ *and* $x^i_{fin}, i = 1, \ldots, N$, *initial and final conditions be given. Then, the* **DRL-MPC** *Algorithm always satisfies the prescribed constraints, complies with the requirements of the* **PL-UE** *problem and ensures that the regulated state trajectories are asymptotically stable.*

Proof - Under the hypothesis that feasible input sequences are off-line available both for $\mathcal{P}_L$ and $\mathcal{P}^i_F$ optimization problems, the following arguments are exploited for feasibility purposes. The following scenarios can arise:

1) *single platoon phase:* the **leader** feasibility arises from the fact that if an optimal solution there exists at the time instant t, namely $u^{1^*}(t|t) = K^1(t)x^1(t)$, then at the

next time instant $t+1$ the positively invariance property of $\Xi^1(t)$ ensures that the state trajectory at least will remain confined within $\Xi^1(t)$ and, in virtue of (4.31)-(4.32), the transition to the new controller is asymptotically guaranteed. Similar arguments can be exploited for the generic **follower** because after h_i-steps state ahead predictions the state trajectory $x^i(\cdot)$ keeps confined within the positively invariant set Ξ^i, in turn updated by (4.35);

2) *splitting mode:* without loss of generality, let us assume that two platoons arise, namely N_1 and N_2. At $t+1$ the local optimization $\mathcal{P}_{N^1}$ remains feasible because the domain of attraction $\Xi^1(t)$ is invariant w.r.t. the state feedback $K^1(t)$, while the feasibility of the leader of the new platoon LF_1^2 is not affected because admissible, though not optimal, solutions there always exists in virtue of the results of *Proposition* 1. Again, similar ideas apply to the follower vehicles;

3) *queuing mode:* at $t+1$ the feasibility of the platoon LF^2 is guaranteed in virtue of *Proposition* 2 that allows to define an admissible, though not optimal, control moves sequence by exploiting the delay property of the PI region $\Xi^i(t)$ computed by complying with (4.54)-(4.56).

Finally, the closed-loop asymptotic stability property derives from the same analysis because, according to each routing decision $a^i(t)$, the vehicle position converges to x^i_{fin} as $t \to \infty$.

4.7 SUMO-MATLAB co-design

In this *Section*, a simulation framework is proposed with the aim of showing the capabilities of the presented control architecture in complex environments. Specifically, two software platforms are considered: **S**imulation of **U**rban **MO**bility®(SUMO) software [126] and *MATLAB 2022a*®.

SUMO is an open source, globally accepted by scientific community, highly portable, microscopic and continuous multi-modal traffic simulation package designed to handle large road networks [127]. Although from a macroscopic point of view, traffic is usually described by means of departure times and routes with certain durations, in a real scenario it is highly conditioned by various other factors like drivers mobility, driving styles, weather, infrastructure within the region or other incidents affecting the environment. As it is well-known, the

environment complexity prevents any traffic modeling via explicit mathematical expressions [128], therefore the only chance consists in performing reliable simulations instrumental to show critical waypoints or to predict traffic behaviors [126]. This software is particularly suitable to simulate URNs where different classes of vehicles (such as cars, trains, buses) lie. The simulation engine is based on an hybrid description: time-discrete and space-continuous. Moreover, the SUMO community provides all the URN data, traffic demanding and the resulting simulation in XML formats.

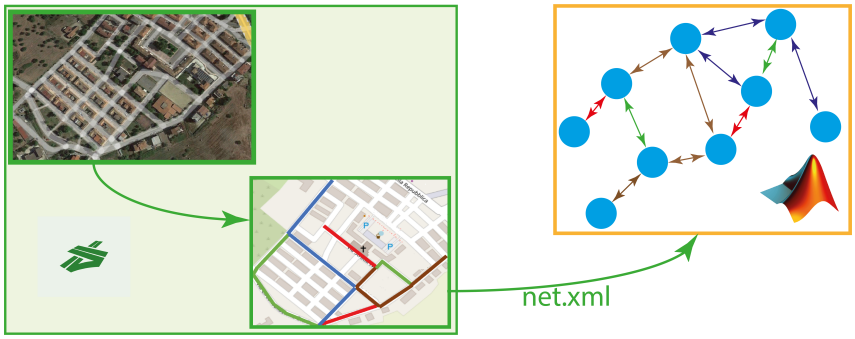

Figure 4.17: Conversion of a real city map into a decision graph

Before the SUMO engine starts, a properly encoded MATLAB object is obtained by exploiting the SUMO xml files. This object contains the graph representing the URN - created from the net xml file provided by SUMO, see figure 4.17 - and others properties and methods useful to develop the proposal.

Then, a run-time communication layer is needed in order to exchange information between SUMO and MATLAB while their two engines run. Therefore, a co-design procedure has been developed. Essentially, as shown in figure 4.18, the core of the proposal has been implemented in MATLAB, while the AVs live in SUMO. The software communication between the two is allowed by the functions provided by the TraCI4Matlab®[129] toolbox, properly customized to comply with the **DRL-MPC Algorithm**. More in detail, TraCI application level protocol is based on the client-server paradigm and it is built on top of the TCP/IP stack: the application developed in MATLAB plays the role of a client and it can access and modify the simulation environment provided by SUMO that acts as a server.

A set of Application Programmable Interfaces (APIs) have been then developed to get, modify or add information about vehicles within the simulation environment. For instance these routines can be used to query SUMO where a vehicle is currently moving or to add

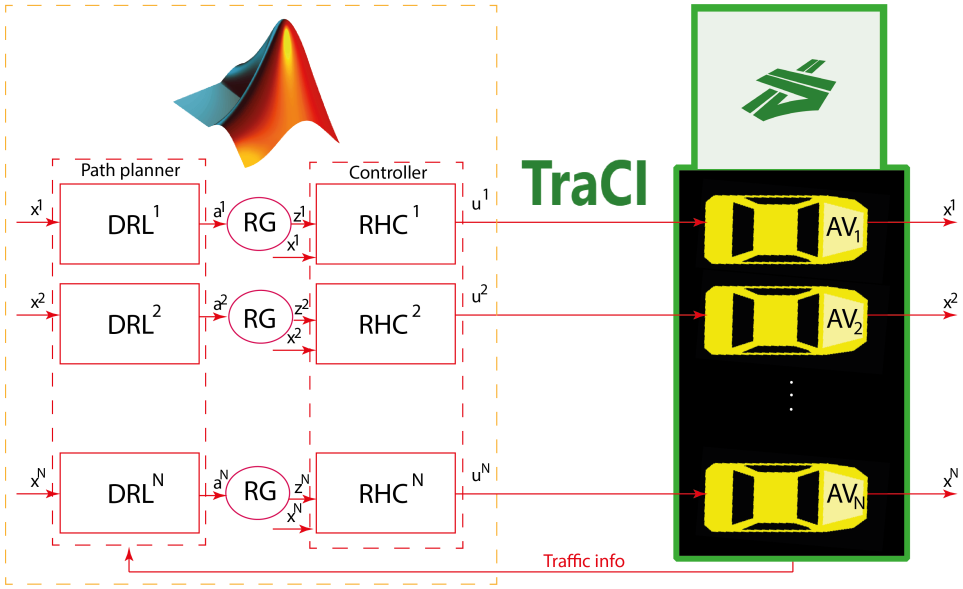

Figure 4.18: A representation of the simulation framework

a new vehicle in terms of its position, speed, attitude and driving features in the URN reference frame. In addition, it is possible to obtain data regarding road vehicles density, spatial constraints (i.e. road width, length and topology), obstacles and so on.

The proposed co-design adapts to new operating scenarios by simply selecting a new SUMO xml describing file that in turn increases its flexibility features for routing decision purposes. Specifically, the designed APIs also allow to compute the reward (5.13), while the optimization of NN weights is achieved by means of the MATLAB Reinforcement Learning toolbox®and the MATLAB Deep learning toolbox®.

4.8 Numerical results

Several simulations have been performed in order to show the effectiveness and the peculiarities of the proposed control architecture. Thereafter, two simulation scenarios are depicted: the first one [89] implements the **DRL-MPC** algorithm, while the second one [125] the **DRL-MPC** algorithm with platoon splitting and queuing. In order to show the adaptability of the **DRL-MPC** algorithm and of the developed toolbox, to different scenarios, two URNs are considered. Both of them are correlated with real traffic data,

provided by the TAVF project [1] throught the SUMO documentation about scenarios[2].

Each AV is described by a point mobile robot whose dynamics is described as in (4.1) where $x^i = [p_x^i,\, p_y^i,\, v_x^i,\, v_y^i]^T \in \mathbb{R}^4$ is the state vector and

$$A^i = \begin{bmatrix} I_2 & T_s I_2 \\ 0_2 & I_2 \end{bmatrix},\; B^i = \begin{bmatrix} \frac{T_s^2}{2} I_2 \\ T_s I_2 \end{bmatrix},\; \forall i = 1, \ldots, N$$

where $T_s = 0.1\,[s]$ the sampling time. Moreover, the following input constraints

$$\mathcal{U} := \{u^i(\cdot) \in \mathbb{R}^2 : \|u^i(t)\|^2 \leq u_{max} = 5\,[m/s^2],\; \forall t \geq 0, \forall i = 1, \ldots, N\} \tag{4.57}$$

are prescribed on the acceleration components.

The perception and steering capabilities of the vehicles are defined by $R_{vis} = 5\,[m]$, $R^c_{min} = 2\,[m]$. Moreover, the decision zones (4.3) are straightforwardly derived according to the assumption the decision is made $3.5\,[m]$ before each junction. Finally, the schematic of figure 4.19 provides an overview of the neural network used to implement the Q-function:

- the blue blocks refer to the NN inputs (s^i and a^i);
- the yellow ones are Fully Connected Layers - the numbers in the figure indicate the chosen number of neurons -;
- the red ones are ELU activation functions;
- the grey block is an addition layer;
- the last block is in charge to compute $\mathcal{Q}^i(s^i, a^i; \theta^i)$ and, therefore, it consists of one neuron.

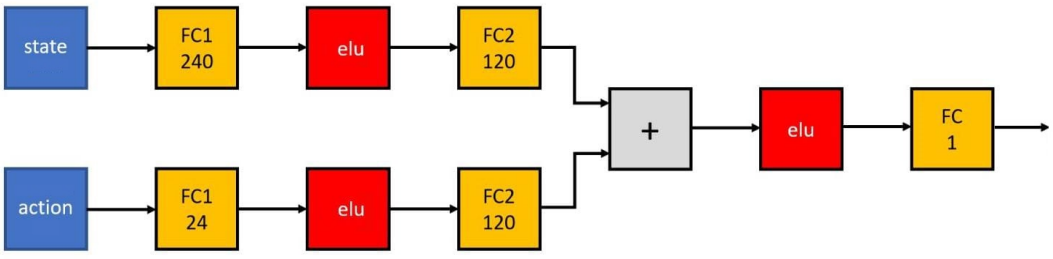

Figure 4.19: Neural Network description

[1] https://tavf.hamburg/en/

[2] (https://sumo.dlr.de/docs/Data/Scenarios.html#dublin_-_irish_motorway_national_road_and_city_centre)

4.8.1 DRL-MPC algorithm

The operating scenario is a map of a district in Hamburg in figure 4.4. The reference initial position is located at the latitude $53.540365°N$ and longitude $9.946923°E$ coordinates. By using a local tangent plane East-North-Up reference frame [130], the AVs initial positions have been set as reported in Table I with $v_x^i = 0$ and $v_y^i = 0, \forall i$. Moreover, a traffic limitation is imposed on the edge highlighted in ciano. The aim consists in driving a platoon of $N = 5$ vehicles towards the target p_{fin}. The geometrical displacement between two consecutive vehicles is defined by the bounds $d_{min} = 1[m]$ and $d_{max} = 6[m]$.

Figure 4.20: On the left, the URN from the city of Hamburg in Google Earth. On the right, the same URN in SUMO. The leader initial (yellow) and target (red) locations and the edge with traffic limitation (ciano) are highlighted.

Table 4.1: Vehicle initial positions

Agent	AV_1	AV_2	AV_3	AV_4	AV_5
$p_x^i(0)$	1495.8	1494.9	1492.1	1490.3	1488.4
$p_y^i(0)$	667.1	665.4	663.7	662	660.3

The achieved results are collected in figs. 4.22-4.24. As the Q-function is concerned, around 1800 traffic episodes have been used for its training. Notice that at t_0 the vehicles are located to their initial positions reported in Table I, while at t_{fin} they are on the target x_{fin} (parking area).

First, the evolution of the training is reported in the left hand side of figure 4.22, where

the episode reward (the sum of all the received rewards during each episode) and the expected episode reward are depicted. This graph explicitly shows how proper routing decisions are selected according to the training evolution: the episode reward (blue line) settles down to an high performance value, while the expected reward (orange line) asymptotically converges to it. On the other hand, the r.h.s. of figure 4.22 accounts for the CO_2 emissions of the platoon for each episode. As expected, the emission curve decreases as the training process evolves

To analyze the behavior of the training process *versus* the emission threshold on the edge with traffic limitation, the results reported in figure 4.22 are discussed more in depth. As the threshold increases, the chance of the platoon to enter that edge clearly increases. Accordingly, the expected reward shows less settling times as the threshold increases.

Finally, by taking a look to the regulated multi-agent system behavior, in figure 4.21 the regulated state trajectories show that the platoon comply with the prescribed mission as testified by means of four relevant snapshots, while all the prescribed constraints are always fulfilled, see figs. 4.23, 4.24.

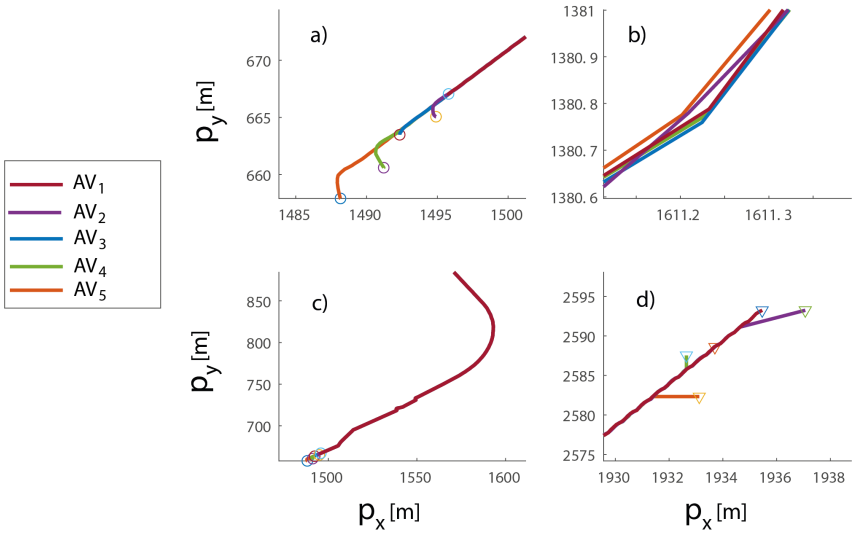

Figure 4.21: Regulated state trajectories: a) the circles denote the initial positions of the five vehicles; b) under the action of the **DRL-MPC** Algorithm the vehicles move along the path indicated by the **DQL** Algorithm; c) the vehicles are geometrical organized as a platoon; d) the vehicle are approaching the parking locations depicted as triangles.

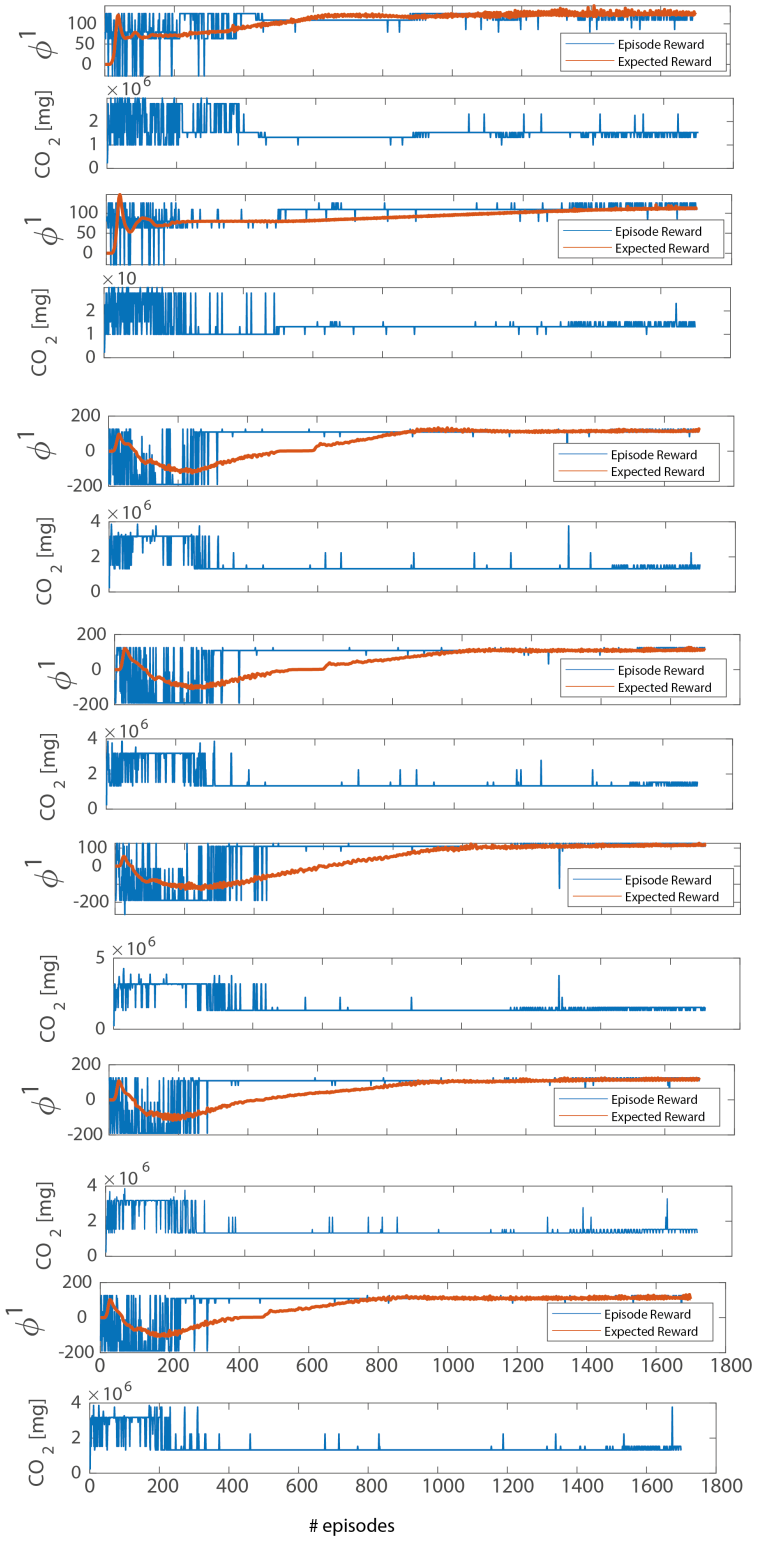

Figure 4.22: DRL training and emissions. Emission thresholds: 50000 55000 80000 5500000 6000000 7000000 20000000 [mg/s].

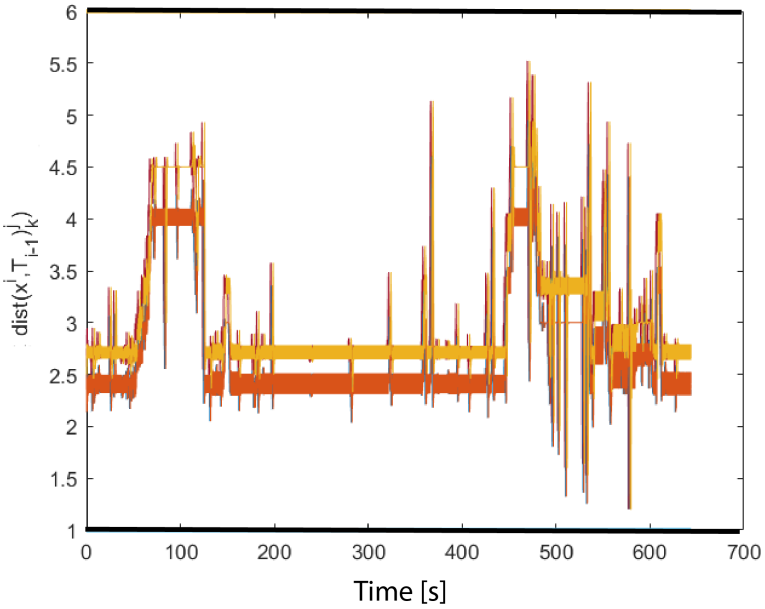

Figure 4.23: State constraints (5.34): safe distance evolution between neighbouring vehicles along the platoon.

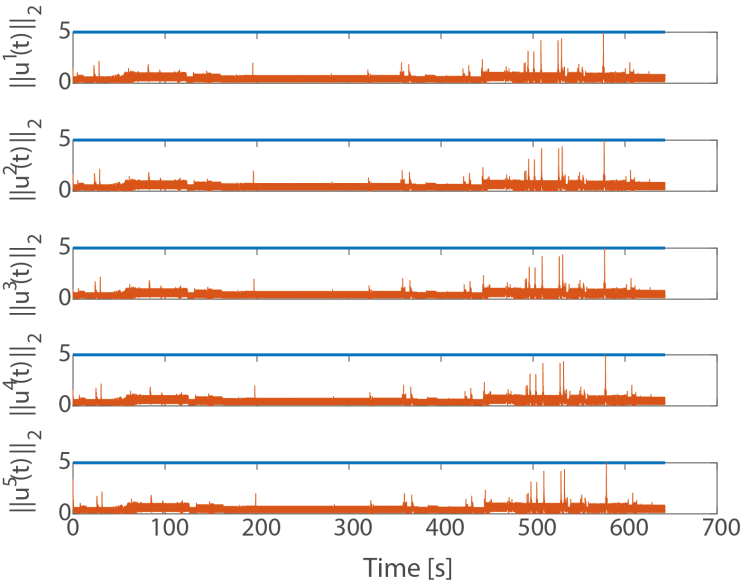

Figure 4.24: Acceleration constraints compatible with routing and sustainable requirements.

4.8.2 DRL-MPC algorithm *with platoon splitting and queuing*

In this section, a set of simulations is proposed aiming at highlighting the performance of the proposed **DRL- MPC** algorithm *with platoon splitting and queuing*.

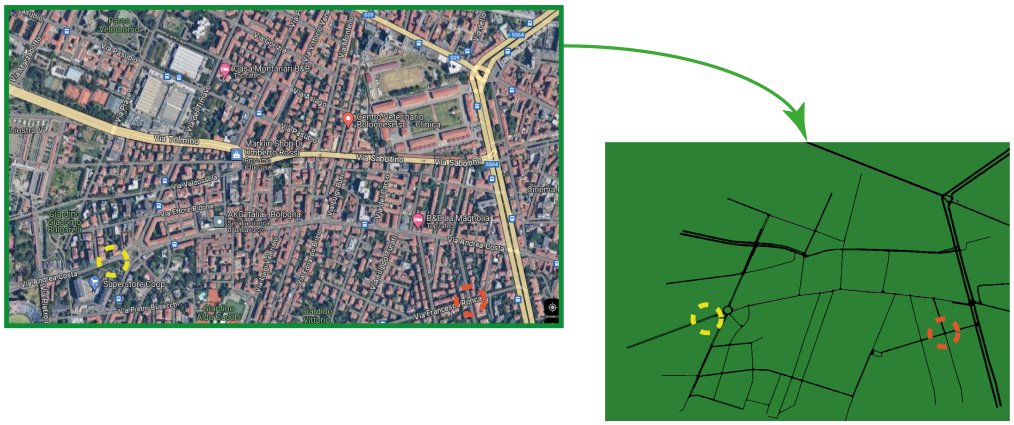

Figure 4.25: On the left, the URN from the A. Costa district in Bologna in Google Earth. On the right, the same URN in SUMO. The leader initial (yellow) and target (red) locations are highlighted.

Here, the operating scenario is the district around the "Andrea Costa" road in Bologna (IT), see figure 4.25. The traffic data refers to an area in the proximity of the city football stadium that well adapts to simulate the vehicles mobility during big and popular events such as football matches or concerts [131]. In particular, these traffic data are related to the Bologna's peak hour (8:00 am - 9:00 am) and have been generated by considering the highest traffic demand during a football match on March 3rd 2010 and the daily flow one week before. The reference initial position is located at the latitude $44°29'23.6''North$ and longitude $11°18'30.2''East$ coordinates. Within a local tangent plane East-North-Up reference frame [130], the AVs initial positions have been set as reported in Table 4.2 with $v_x^i = 0$ and $v_y^i = 0, \forall i$. The aim consists in driving a platoon of $N = 11$ vehicles towards the target p_{fin}It is required that the geometrical displacement between two consecutive vehicles is defined by $d_{min} = 3\,[m]$ and $d_{max} = 10\,[m]$. The following knobs are used for the **DRL-MPC** algorithm implementation: $R_x^i = I_{n_i},\ R_u^i = I_{m_i},\ i = 1, \ldots, N$, $\alpha = 0.01$ and $\gamma = 0.99$.

Table 4.2: Platoon initial positions

Agent	AV^1	AV^2	AV^3	AV^4	AV^5	AV^6
$p_x^i(0)$	70.0195	63.955	57.8905	51.826	45.7615	39.697
$p_y^i(0)$	386.4212	384.082	381.7428	379.4036	377.0644	374.7252

	AV^7	AV^8	AV^9	AV^{10}	AV^{11}
$p_x^i(0)$	33.6325	27.568	21.5035	15.439	9.3745
$p_y^i(0)$	372.386	370.0468	367.7076	365.3684	363.0292

All the achieved results are collected in figs. 4.26 -4.32. As the Q-function is concerned, the training has been done by generating 2000 traffic episodes and it is depicted in figure 4.26, where the episode reward (the sum of all the rewards received from the agent during each episode), the moving average reward and the expected episode reward are reported. There, it can be observed that the capability of the vehicle to select the more adequate decision improves as the training process evolves: the average reward (red line) increases, while the expected one (orange line) shows an asymptotic convergent behavior to the effective reward.

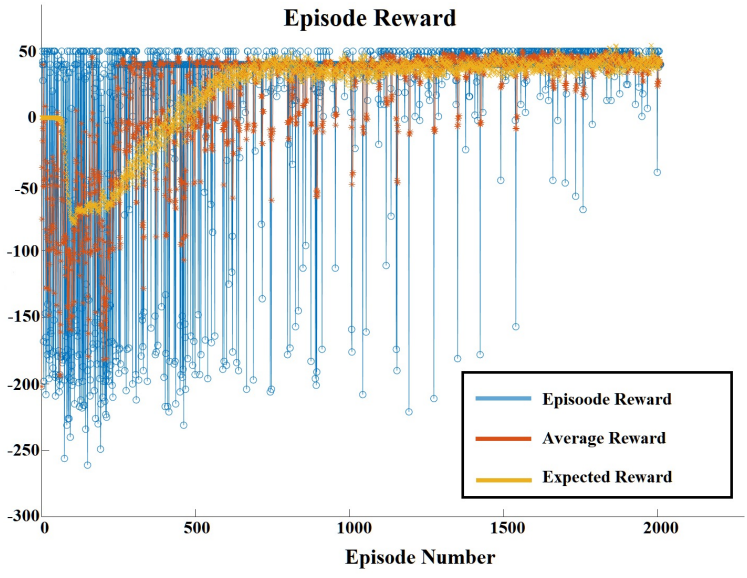

Figure 4.26: Training from MATLAB Reinforcement Learning toolbox

The first experiment is performed by considering the traffic data in the time interval $8:00am - 8:30am$. In this case, the initial platoon configuration is kept the same until

around $t \approx 482\,[s]$, when the greedy actions $a^i(482) \neq a^1(482)$, $i = 9, 10, 11$, suggest to the vehicles $AV_i, i = 9, 10, 11$, to turn on the right while the rest of the platoon continues on the straight line, see figure 4.27 and the sub-graph b) of figure 4.29. As prescribed in Section 4.6.2, two platoons $LF^1 = \{AV_1, \cdots, AV_8\}$ and $LF^2 = \{AV_9, AV_{10}, AV_{11}\}$ arise with $AV_9 = AV_1^2$ the leader of LF^2. This event is detailed in figure 4.29 where four snapshots, described within the SUMO environment, at different time instants show the dynamical behavior of the platoon: 1) before the splitting; 2) at the splitting time instant; 3) with two platoons along the path; and 4) when the leader AV_1 accomplishes the mission. Notice that the two platoons never queue until the parking area is reached: in particular, AV_1 arrives at $t \approx 690\,[s]$ while the second leader AV_9 at $t \approx 740\,[s]$.

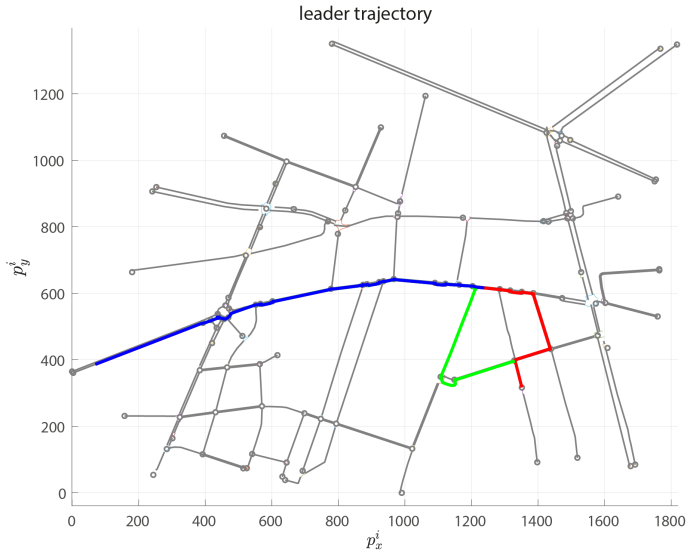

Figure 4.27: $8:00am - 8:30am$ traffic data set: single leader (blue line) and two leaders (AV_1 red line, AV_2 green line)

The second experiment considers the time interval $8:30am - 9:00am$. At $t \approx 123\,[s]$ the vehicles AV_i, $i = 6, \ldots, 11$, make a different routing decisions with respect to the leader AV_1 and two new platoons arise, see the sub-graph b) of figure 4.31. This situation persists until $t \approx 350[s]$ when LF^2 queues to LF^1 because $a^i(350) = a^j(350), i = 1, \ldots, 5, j = 6, \ldots, 11$, and $\|p^6(350) - p^5(350)\| = 7.8\,[m]$. Hence, the platoon reaches the target at $t \approx 715\,[s]$. For the sake of completeness constraint evolutions and platoon behavior snapshots within SUMO are summarized in figs. 4.30, and 4.31, respectively. Finally, figure 4.32 depicts the achieved paths during the prescribed mission.

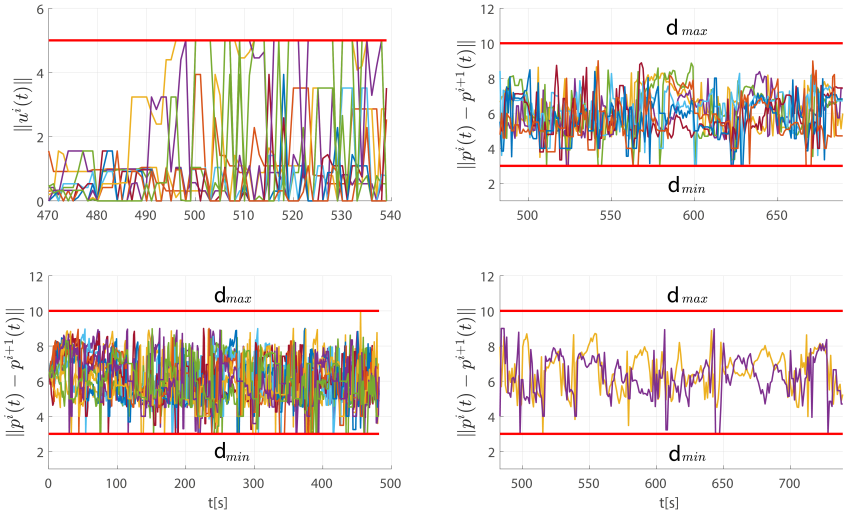

Figure 4.28: $8:00am - 8:30am$ traffic data set: constraints fulfillment. Left bottom graph refers to a single platoon, while the right column to the two platoons scenario (LF^1 (top) and LF^2 (bottom)).

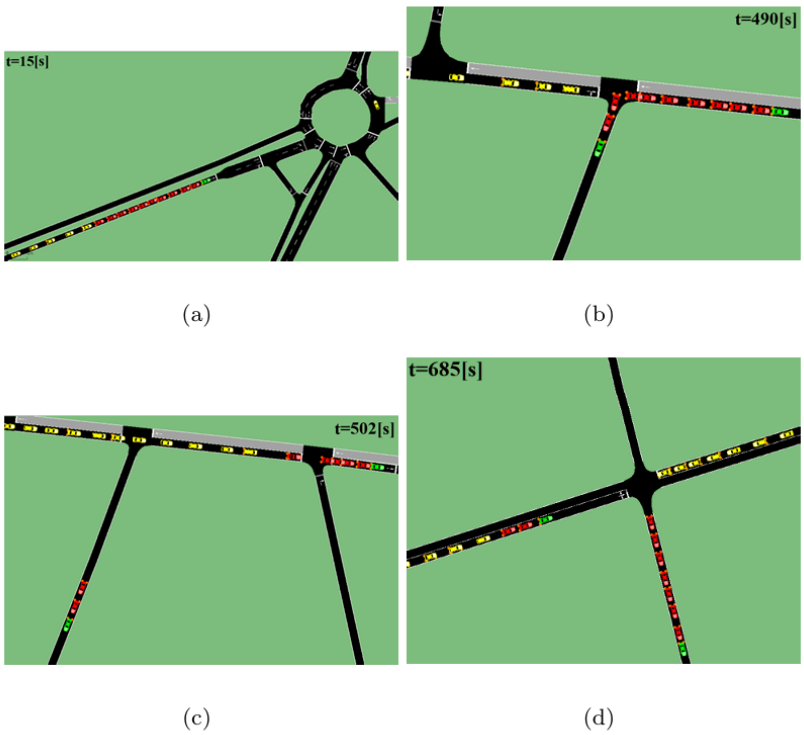

Figure 4.29: $8:00am - 8:30am$ traffic data set: (a) single platoon before split (b) splitting (c) two platoons moving in the URN (d) platoons reaching the target position. Yellow cars denote the traffic along the path.

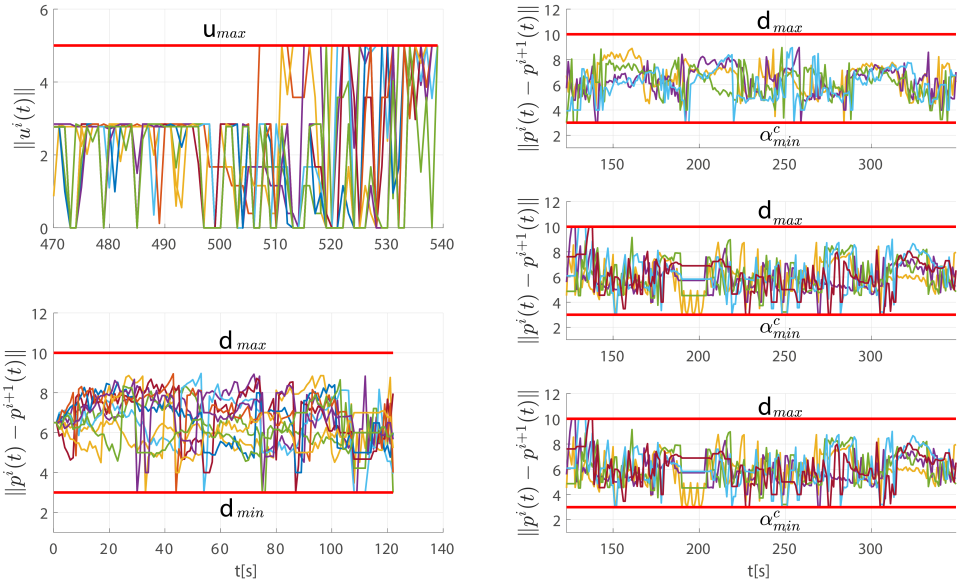

Figure 4.30: $8:30am - 9:00am$ traffic data set: constraints fulfillment. Left bottom graph refers to a single platoon, right top and middle graphs to the two platoons configuration and the right bottom one to the single platoon after queuing.

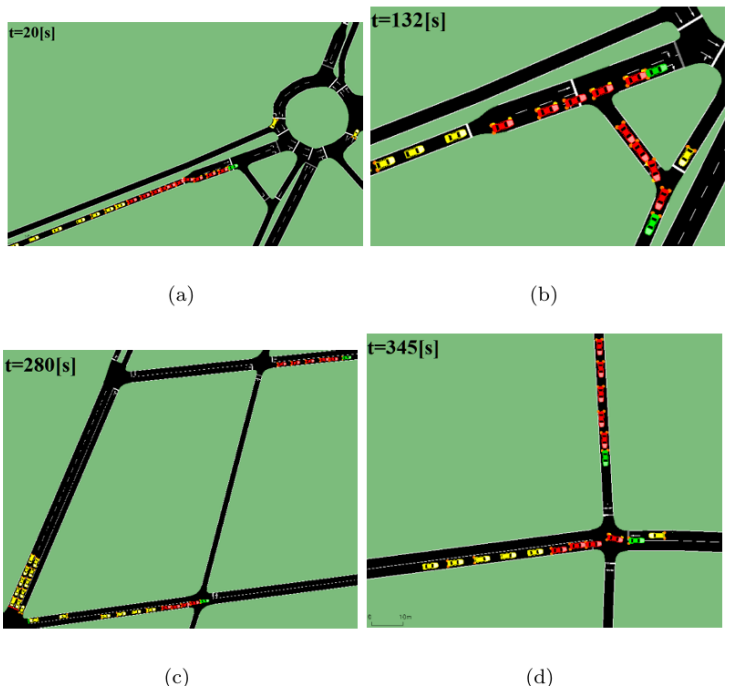

Figure 4.31: $8:30am - 9:00am$ traffic data set: (a) single platoon before split (b) splitting (c) two platoons moving in the URN (d) platoons reaching the joining condition. In all the pictures, green vehicles are the leaders and red vehicles are the followers. Yellow cars denote the traffic along the path.

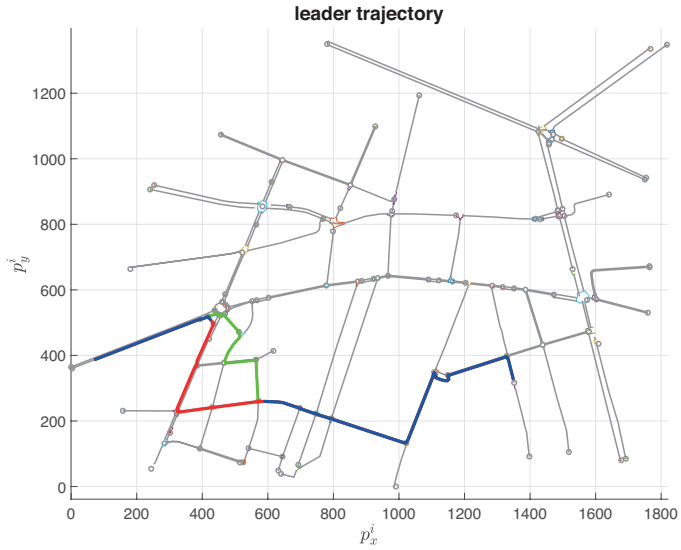

Figure 4.32: $8:30am - 9:00am$ traffic data set: leaders trajectory in the cases of a single platoon, before the split and after the join (blue line), and of two platoons (red line for LF^1 and green line for LF^2)

Chapter 5

An intelligent multi-layer control architecture for logistics operations of autonomous vehicles in manufacturing systems

This *Chapter* is based on the work presented in [132][133]. Here, the framework proposed in *Chapter* 3 is applied to a different research context. The idea is the development of a multi-layer control architecture for autonomous vehicles in smart manufacturing systems. These vehicles are organized in a finite set of platoons in charge to accomplish prescribed job(s). Three aspects are then needed to be formally outlined: task scheduling, routing decisions and command inputs computations. Therefore, a distributed multi-layer architecture has been conceived by using three methodologies: Timed Colored Petri Nets (TCPNs), DRL and MPC. In a nutshell, TCPNs are exploited to formally model the manufacturing system so that an optimal scheduling task complying with the required jobs and the available vehicles is derived; then, run-time routing decisions are obtained by using a DRL approach which exploits the available information provided by the vehicle sensor module; finally, the

distributed MPC algorithm is built by resorting to a set-theoretic approach where most of the computations are off-line performed.

5.1 Context of interest

Nowadays, industries have to deal with a high competitive market. Therefore, any process optimization devoted to improve the production efficiency has a crucial role. Accordingly, current industrial systems move toward an increasingly involvement of automation for enhancing challenging logistics issues: streamline operations, customer service, transportation cost, planning and risk management [134]. Along these lines, the technological innovations in the field of autonomous vehicles has put in light an inherent capability to sense and interact with the surrounding environment in a manner that allows to rationalize costs and increase societal benefits such as safety and accessibility [135]. As a matter of fact, autonomous vehicles are a radical innovation that could efficiently assist the management of production lines, warehouse inventories handling and the support of intra- and inter-logistics services in several economic sectors [136]-[137].

In a nutshell, production logistics settings with several transportation tasks, (e.g., between warehousing or material supply stations and production locations within larger production sites) are usually addressed by autonomous vehicles swarms in virtue of their distributed decision-making capabilities. Therefore, in this context an interdisciplinary study involving logistics, artificial intelligence and control of dynamical systems naturally comes out [138]. In turn, this gives rise to three key issues:

1. definition of a near-optimal task scheduling,

2. computation of adequate routing decision actions,

3. development of an efficient distributed control scheme to accomplish prescribed production jobs.

In what follows, these questions will be addressed in a multi-layer way by taking advantage of timed colored Petri nets (TCPN) for the scheduling design, Deep Reinforcement Learning (DRL) for routing decision purposes and Model Predictive Control (MPC) arguments for driving the platoon within the Flexible Manufacturing Systems (FMS).

5.1.1 Literature review

The coordination of complex manufacturing systems involves challenging constraints, and several approaches have been proposed in literature to address the scheduling issue. One of the first literature contributions on the Petri Net approach is from [139] where a modeled FMS scheduling has been described by a PN firing sequence from an initial marking to a final one and, thanking to a branch-and-bound algorithm, an optimal schedule of the FMS then obtained. Also, Timed marked graphs, a special timed PN category have been exploited to analyze the long-term FMS performance. Deep reinforcement learning approaches are more recent and noticeable contributions are [140] and [141] where a Cuckoo Search Algorithm is proposed: the first exploits a knowledge base, initially trained off-line before operation whereas the second exploits a learning based approach. All the approaches are based on reinforcement learning and hybrid heuristics to store scheduling information and appropriate parameters. In [142] a learning-based grey wolf optimizer tailored for two scenarios in real manufacturing environments is instead proposed. This approach exploits optimal computing budget allocation to intelligently allocate resources and enhance search efficiency.

Regarding the field of industrial logistics supported by the operation of autonomous vehicles, as extensively discussed, for instance, in [143]-[144], significant interest has been observed among researchers approaching the subject from various perspectives. In broad terms, literature contributions on the matter can be partitioned in centralized [145] and distributed approaches [146]. By fixing the attention on the second ones, the initial solutions were characterized by local coordination-based [147] and priority-based algorithms [148]. Dynamic optimization algorithms based on enhanced rule-based heuristic algorithm and the fusion of the Dijkstra and Q-Learning algorithms, has been developed in [149] to tackle optimal autonomous vehicle scheduling and path planning. Conflict avoidance strategy rooted in graph theory has been also introduced to mitigate path collisions among agents. Unfortunately, high demands for robot perception modules and communication facilities turned into significant applicability prevention of these methods. Therefore, hierarchical strategies have been exploited for the coordination of automated guided vehicles, see e.g., [150] and references therein. In turn, these approaches suffered for scalability issues that in recent years have been addressed by resorting to learning-based approaches [151] particularly suitable for their intrinsic capability to train the autonomous vehicles by acquiring local

information. On the other hand, until now the resulting dynamical performance are not comparable with those pertaining to the traditional methods above reported. Currently, more effective and appealing solutions fall down in the MPC category [152]. Specifically, the NP-hard traffic scheduling problem is significantly simplified in [153] by taking advantage of the MPC properties: receding horizon principle and optimality of distributed solutions.

By summarizing, the above analysis puts in light that this complex problem, namely logistic operations in industrial contexts via multi-vehicle configurations, needs a mutli-disciplinary approach capable to take advantage of the properties of any single strategy and to mitigate as much as possible the related deficiencies by mixing each other the involved methodologies.

5.1.2 Main contribution

The core of this chapter relies on the design of an original architecture, based on deep interactions among scheduling, routing and control objectives, that has to be capable of efficiently accomplishing logistic tasks by using a group of autonomous vehicles moving within a generic FMS labyrinth. Accordingly, flexibility and production performance enhancement are the guidelines along which the approach is developed. The starting idea is to *a-priori* organize the vehicles as a set of platoons, each one in charge to accomplish a specific job of the overall task: this prescribes to formally introduce the following ingredients:

1. a centralized unit providing a full required operations scheduling scheme to be used during the on-line phase;

2. a distributed routing decision unit for mitigating vehicles congestion phenomena;

3. a distributed control strategy to take advantage of the scheduling and routing decisions according to the constraints imposed by the FMS planimetry.

In view of these premises, the resulting framework enjoys a hierarchical structure and has a twofold feature: parallel with respect to the mission assigned at each platoon and distributed *versus* the vehicles of each leader-follower configuration. Specifically, the first question is addressed by using the TCPN modeling capabilities that allow to define a so-called twin description of the overall FMS and to extrapolate a near-optimal scheduling; the routing decisions are obtained in a distributed fashion by resorting to DRL tools, while the constrained

control scheme is developed by using set-theoretic arguments along the lines indicated in [154].

The overall *modus operandi* can be summarized as follows. In the off-line phase the TCPN formalization allows to compute the scheduling of the prescribed jobs that are provided as inputs to the distributed units during the on-line phase. At each time instant, for each vehicle the DRL determines a routing decision in charge to accomplish its own operation within the platoon job by mitigating as much as possible collision occurrences amongst platoons. Conversely, the distributed MPC strategy jointly exploits the scheduling and the decisions coming from the DRL in order to take the more adequate control action according to the receding horizon philosophy. Since by construction the platoons are completely disconnected each other (no information are shared amongst them), an anti-collision protocol must be conceived. Here, the idea is to associate to each job a priority (this is without loss of generality within a real-like FMS context) that is exploited once the vehicle sensor module detects a competitor platoon along the same line: the low priority platoon stops and the high level one overcomes it by on-line building an overlapped sequence of safe regions. Unlike existing literature contributions, the proposed approach enjoys several properties: interdisciplinary translated into a flexible multi-layer architecture; full exploitation of the FMS information (task and planimetry) to define a low-demanding distributed control scheme; parallelization of the assigned jobs by proper platoons partition of the available set of autonomous vehicles; distribution among the platoon vehicles of the operations pertaining to each single job.

5.1.3 Preliminaries: Timed Colored Petri Nets

First of all, it is worth to recall that PNs and Colored Petri Nets (CPNs) are discrete-event paradigms successfully used for modeling concurrent, asynchronous, distributed, parallel, non deterministic and stochastic processes [155, 156]. As a matter of fact, they have been extensively applied in the FMS domain for their natural ability to address model precedence relations, deadlocks, conflicts, and resource constraints [157, 158] by resorting to logic properties, such as boundedness and liveness [159]. CPNs can represent parts with attributes as well as temporal activities, i.e., TCPNs [160].

In a Petri net (PN), a *token* is an abstract representation of a discrete quantity, often used to model resources, data, or control flow within the system being modeled. The state

of the PN is described by the *marking* that refers to the actual distribution of tokens. Timed coloured Petri nets (TCPN) extend Petri nets (PNs) [160] with two concepts: colours and time.

- In TCPN, tokens can carry different data values (*token color*). Also, places can carry data types (*colour sets*). Places can only contain token colour corresponding to their colour sets.
- The second extension is that tokens can carry a *timestamp*, the time at which the token is ready to be used. Transitions can be defined with time delay inscription. Also, there is a *global clock* representing the model time.

Definition 12 *Formally, a timed colored Petri net is a tuple*

$$TCPN = (\Pi, \Phi, Arcs, Post, Pre, CS, Var, PC, Gu, AE, Ini) \tag{5.1}$$

where, for the sake of clarity, the meaning of the symbols is hereafter reported:

- Π *denotes a finite set of places;*
- Φ *is a finite set of transitions, such that* $\Pi \cap \Phi = \emptyset$ *;*
- $Arcs \subseteq \Pi \times \Phi \cup \Phi \times \Pi$ *is a set of directed arcs;*
- *Post and Pre are the* $|\Pi| \times |\Phi|$ *post-incidence and pre-incidence matrices, respectively;*
- CS *is a finite set of non-empty colour sets, each colour set is either untimed or timed;*
- Var *is a finite set of typed variables such that* $Type[Var_i] \in CS$ *for all variables* $Var_i \in Var$*. The operator* $Type[\cdot]$*, given an expression, returns its type;*
- $PC : \Pi \to CS$ *is a colour set function that assigns a colour set to each place. A place* Π_i *is timed if* $PC(\Pi_i)$ *is timed, otherwise* Π_i *is untimed;*
- $Gu : \Phi \to EXPR_{Var}$ *is a function that assigns a guard to each transition* Φ_i*.* $EXPR_{Var}$ *is the set of expressions provided by the inscription language and such that* $Type[Gu(\Phi_i)] = Bool$;
- $AE : Arcs \to EXPR_{Var}$ *is an arc expression function that assigns an arc expression to each arc* $Arcs_i$ *such that* $Type[AE(Arcs_i)] = PC(\Pi_i)_{MS}$ *where* Π_i *is the place*

connected to the arc $Arcs_i$. This means that each evaluation of the arc expression must yeld a multi-set (MS)[160] over the colour set that is attached to the corresponding place. MS becomes a timed multi-set (TMS) if Π_i is a timed place;

- *$Ini : \Pi \rightarrow EXPR_{Var}$ is a function that assigns an initialisation expression to each place Π_i such that $Type[Ini(\Pi_i)] = PC(\Pi_i)_{MS}$.* □

5.2 Problem formulation

In the sequel, the focus consists in the description of the logistic operations in a manufacturing system using a team of autonomous ground vehicles organized in terms of leader-follower configurations. To this end, the first issue relies on a formal definition of the operating environment. This prescribes the characterization of two aspects:

- *Factory layout*: without loss of generality, it is assumed that it is exactly known in terms of the accessible region. Let M be the number of corridors characterizing the planar structure. Each aisle CO^j, $j = 1, \dots M$ is formally defined by resorting to polyhedral arguments, i.e.

$$CO^j : \bigcap_{k=1}^{\natural} \left\{ (\mathcal{H}^j)'_k p > (g^j)_k \right\} \tag{5.2}$$

where $p \in \mathbb{R}^2$ are the planar components of the state space $x \in \mathbb{R}^n$, $(\mathcal{H}^j)'_k$ characterizes the wall directions in the j-th corridor/aisle and, in a planar representation, is a 2-elements row vector, $(c^j)_k$ denotes the shift vs. origin term of the j-corridor walls (scalar quantity in a planar representation). The p-plane point locus $(\mathcal{H}^j)'_k p = (g^j)_k$ denotes a linear manifold characterizing the walls boundary whereas $(\mathcal{H}^j)'_k p > (g^j)_k$ are half-spaces whose intersection generates the corridor (see figure 5.1). $\natural$ is the number of hyperplanes defining CO^j that in the example of figure 5.1 is equal to 2. The accessible region is

$$\mathcal{O}_{free} := \bigcup_{j=1}^{M} \left\{ x \in \mathbb{R}^n : p \in CO^j \right\} \tag{5.3}$$

notice that the integer M takes care of same drive-through corridors in opposite directions.

- *Task description* - the following statements are exploited:

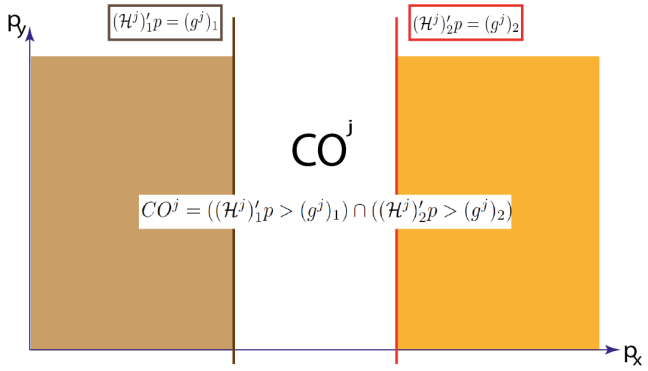

Figure 5.1: Aisle/Corridor polyhedral representation

- A FMS is an integrated, computer-controlled factory whose elements are automated material handling devices and numerically controlled machine tools that can simultaneously handle medium-sized volumes of pieces and parts to be processed. It consists of a finite set of machines $Ma = \{Ma^k\}_{k=1}^{\beth}$ - $\beth$ is the number of machines - and a load/unload (L/U) station [161];
- A job $\mathcal{J}^k$ is an ordered sequence of operations $op_i, i = 1, \ldots, n_k$, carried out on dedicated machines $\{Ma^1, \ldots, Ma^{n_k}\} \subseteq Ma$ [162, 163].

Then, a task is defined as the collection of a finite number of jobs $\{\mathcal{J}^1, \ldots, \mathcal{J}^{\daleth}\}$.

The second issue concerns with the autonomous vehicles (AVs) modeling. The AV dynamics is described by means of discrete-time linear time invariant (LTI) models

$$x^i(t+1) = A^i x^i(t) + B^i u^i(t), \;\; i = 1, \ldots, N, \tag{5.4}$$

Like in the case of the transportation systems, these vehicles are subject to set-membership constraints:

$$\begin{aligned} x^i(t) \in \;\; & \mathcal{X}^i := \{x^i \in \mathbb{R}^{n_i} : {x^i}^T x^i \leq x^2_{max}\}, \\ u^i(t) \in \;\; & \mathcal{U}^i := \{u^i \in \mathbb{R}^{m_i} : {u^i}^T u^i \leq u^2_{max}\}, \; \forall t \geq 0, \end{aligned} \tag{5.5}$$

Moreover, the vehicle geometry is defined by resorting to a polyhedral description in the 2-D space:

$$AV_i : \begin{bmatrix} (\mathcal{H}^{AV_i})'_1 \\ \vdots \\ (\mathcal{H}^{AV_i})'_{l_i} \end{bmatrix} p \leq \begin{bmatrix} (g^{AV_i})_1 \\ \vdots \\ (g^{AV_i})_{l_i} \end{bmatrix} \tag{5.6}$$

where $(\mathcal{H}^{AV_i})'_j p \leq (g^{AV_i})_j$, $j = 1, \ldots, l_i$ half-plane portions whose intersection is the vehicle shell. Then, the problem of interest can be stated as follows.

Platoons in manufacturing systems (PL-MS) - *Given a FMS* $(Ma, L/U)$ *geometrically defined by (5.2)-(5.3) and a group of AVs (5.4) organized as q platoons of* N_j, $j = 1, \ldots, q$, *agents such that* $\sum_{j=1}^{q} N_j = N$, *determine a distributed state-feedback control policy*

$$\begin{aligned} u_j^1(t) &= f(x_j^1(t)) \\ u_j^i(t) &= f(x^i(t), x_j^{i-1}(t)), i = 2, \ldots, N_j, j = 1, \ldots, q, \end{aligned} \tag{5.7}$$

compatible with (5.3) and (5.5) such that an assigned task $\{\mathcal{J}^1, \ldots, \mathcal{J}^{\urcorner}\}$, *with associated job priorities* $\{pr_1, \ldots, pr_{\urcorner}\}$, $0 \leq pr_k \leq 1, pr_i \neq pr_j, \forall i \neq j$, *is accomplished regardless any time-varying obstacle occurrence* $\mathcal{O}_t := \{AV_k\}$. □

The **PL-MS** problem requires a formal characterization of three issues:

- scheduling according to the assigned jobs and priorities,
- routing decision at each junction occurrence within the FMS,
- path palling /obstacle avoidance control action during the platoon traveling.

The next sections will be devoted to address such points.

5.3 The proposed multi-layer control architecture

The developed idea consists in the integration into a unique framework of the previously mentioned methodologies: TCPN, RL and Set-theoretic based MPC. From an abstract point of view, the proposed architecture is reported in figure 5.2. There, three main actions are conceived:

- the task scheduling $\{\mathcal{J}^1, \ldots, \mathcal{J}^{\urcorner}\}$ performed in an off-line phase by exploiting the FMS structure, i.e. , the involved machines M, the factory layout (5.2)-(5.3), and the available AVs team (5.4)-(5.6)[1];
- the routing decision $a^i(t)$, $i = 1, \ldots, N$, updated, whenever necessary during the on-line operations, on the basis of the scheduling path $\{p_{goal}^1, \ldots, p_{goal}^{\urcorner}\}$ and the current state measurement $x^i(t)$, $i = 1, \ldots, N$;

[1]Since the exploration of the resulting reachability graph could be computationally impracticable [160], a near-optimal solution based on reinforcement learning ideas is here developed.

- the control action $u^i(t)$, $i = 1, \ldots, N$, computed in a distributed fashion and taking into account the scheduling sequence $\{p^1_{goal}, \ldots, p^{\urcorner}_{goal}\}$, the list of job priorities $\{pr_1, \ldots, pr_{\urcorner}\}$ and the routing decisions $\{a^1(t), \ldots, a^N(t)\}$.

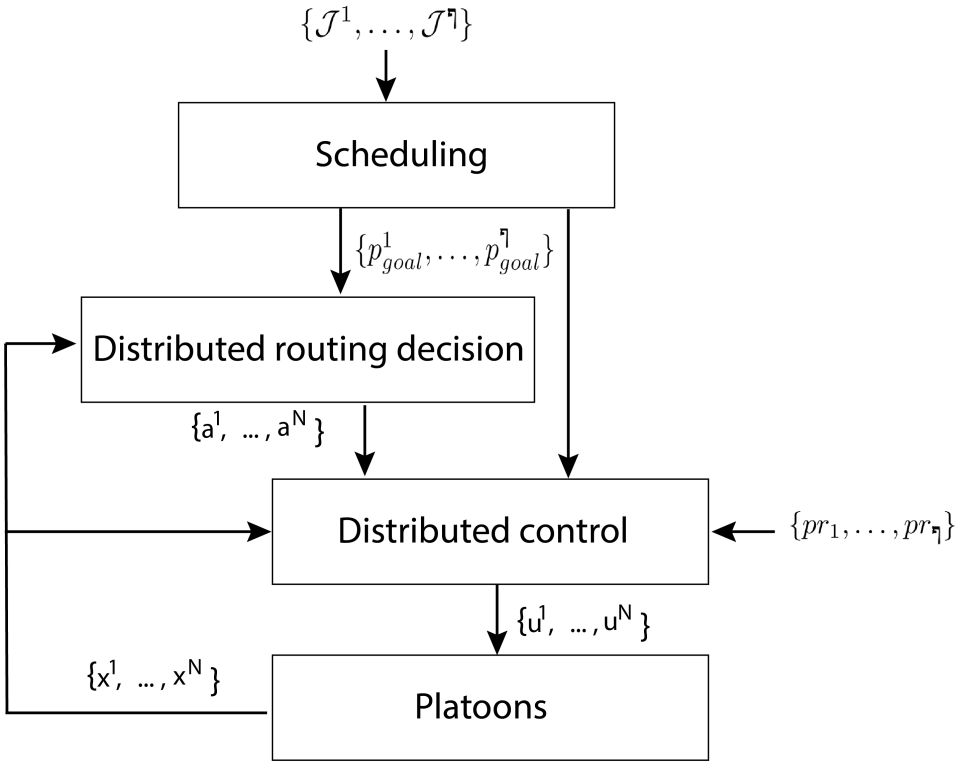

Figure 5.2: Multi-layer control architecture

The *modus operandi* can be summarized as follows. At each time instant t, the routing decision layer generates the action $a^i(t)$ (corridor selection) by using the state measurement $x^i(t)$ to be executed until the next junction. Such a selection is transformed into as a set-point $z^i(t)$ for the control computation purposes by using a RG unit that receives in input the current routing decision $a^i(t)$ and computes the set-point according to the following rule

$$z^i(t) := \arg \max_{z^i \in CO^{a^i(t)}} \|p^i(t) - z^i\|^2 \tag{5.8}$$

For the sake of a formal description, the routing decision is activated along any $j-$th corridor once the platoon enters DZ^j.

The last layer concerns with the computation of feasible command inputs in charge to take care of the platoon framework, to deal with the prescribed input/state constraints and to avoid obstacles along the path. This is done by tailoring to the distributed setting the set-theoretic receding horizon control arguments proposed in [154] for a single vehicle.

The path planning and anti-collision tasks are jointly accomplished by carefully formalizing off-line and on-line phases of the set-theoretic based control strategy. First of all, according to the scheduling path $p^i_{goal},\ i = 1, \ldots, ¶$, the AVs team (5.4)-(5.6) is partitioned into $q \leq ¶$ platoons $LF^k,\ k = 1, \ldots q$.

- **MPC Off-line -** For each corridor CO^j, the state trajectory tube compatible with (5.3), (5.5) is computed for each leader-follower configuration LF^k, namely DoA^k. These regions are then stored in an assigned repository to be used during the on-line operations. Since the latter are performed in indoor environments, without loss of generality it can be assumed that data are safely and instantaneously made available to the AVs side, e.g., by an *ad-hoc* wi-fi network.

- **MPC On-line -** As in the architecture of figure 4.6, each vehicle is equipped with a distributed model predictive control (DMPC) unit according to the platoon partitioning imposed in the off-line phase. At each time instant t, the q platoons proceed in a parallel fashion to accomplish their own missions (namely p^k_{goal}): this is done by computing $u^i(t)$ subject to the information provided by the routing decision layer $a^i(t)$ and to the prescribed priorities vector $\{pr_1, \ldots, pr_¶\}$. In particular, the latter is exploited whenever two platoons $LF^{k_1} = \{AV^{k_1}_1, AV^{k_1}_2, \ldots\}$ and $LF^{k_2} = \{AV^{k_2}_1, AV^{k_2}_2, \ldots\}$ travel in the same corridor CO^j in opposite directions. In such a case, once the two platoons recognize each other the associated priorities pr_{k_1}, pr_{k_2} are compared and the platoon with the lower one, for example $pr_{k_1} < pr_{k_2}$, acts as an obstacle to be bypassed by the platoon LF^{k_2}. Essentially, the lower priority platoon LF^{k_1}, lets LF^{k_2} go ahead.

5.4 Task scheduling: a combined TCPN and Reinforcement Learning scheme

In this section, the design of the scheduling layer is outlined by means of Petri Nets (PNs) based arguments. The scheduling is an issue because the TCPN modeling seems best fitted to explore the whole set of firing sequences of transitions by determining the reachability graph. According to this, the best schedule is then computed by optimizing a given performance index. Unfortunately, an exhaustive enumeration is computationally

intractable due to the intrinsic dimensionality curse. As a consequence, scheduling approaches exploit artificial intelligence heuristics to partially inspect the reachability graph, see [164]-[165] to mention a few.

Moving from this analysis, the FMS is formally described as a TCPN resulting in a digital twin capable of abstractly reproducing the production time evolution based on the involved machines Ma, the load/unload station L/U and the prescribed task $\{J_1, \ldots, J_{\mathfrak{h}}\}$. TCPN exploits a non-deterministic mechanism to execute a transition per time whenever multiple operations are enabled, see [166] for details. The TCPN representation of a FMS is shown in figure 5.3, where:

- A token in Π_1 is described by four scalars: $(\mathcal{J}, i, src, Ma)@t$, where $\mathcal{J}$ accounts for the job, i refers to the operation, src and Ma are the prescribed source and destination related to operation i of job $\mathcal{J}$ respectively. During the evolution of the TCPN model, the scalar i is incremented until it reaches the number of operations of the job $\mathcal{J}$ and then the token is removed. The notation $@t$ is a standard way to indicate that the token $(\mathcal{J}, i, src, Ma)$ is available at time t.

- A token in Π_2 - $(AV, pos)@t$ - accounts for the availability of AV in position pos at time t. Notice that the initial numbers of tokens in Π_2 represents the number of autonomous vehicles N.

- If at least an operation is ready to be executed and a vehicle is available, the transition Φ_1 fires. It is without duration and describes the vehicle assignment event. Notice that, in case more than one token are available in its Pre, the need of a scheduling decision arises.

- A token in Π_3 - $(\mathcal{J}, i, src, Ma, AV, pos)@t$ - represents that the vehicle AV will go from pos to src. Transition Φ_3 is a timed transition and accounts for the time Δt needed by AV to reach src.

- A token in Π_4 - $(\mathcal{J}, i, src, Ma, AV)@t + \Delta t$ -represents that the vehicle AV is ready in src.

- Similarly to Φ_3, transition Φ_4 accounts for the time Δt_2 needed by AV to go from src to Ma. These two transitions are strictly related with the geometry of the FMS and the dynamics of the vehicles.

- Once the vehicle AV reaches Ma, a new token in Π_2 is generated because the vehicle can be used to help a different operation while the operation i of $\mathcal{J}$ is performed in Ma.

- A token in Π_5 - $(\mathcal{J}, i, Ma)@t$ -represents that a piece corresponding to operation i of $\mathcal{J}$ is waiting for machine Ma.

- A token in Π_6 - $(Ma^i)@t$ - stands for machine Ma^i available at time t.

- When a token is ready in Π_5 and the corresponding machine is available in Π_6, the timed transition Φ_5 accounts for working time of machine Ma on operation i of $\mathcal{J}$. Φ_5 produces a token in Π_1 if i is not the last operation of $\mathcal{J}$.

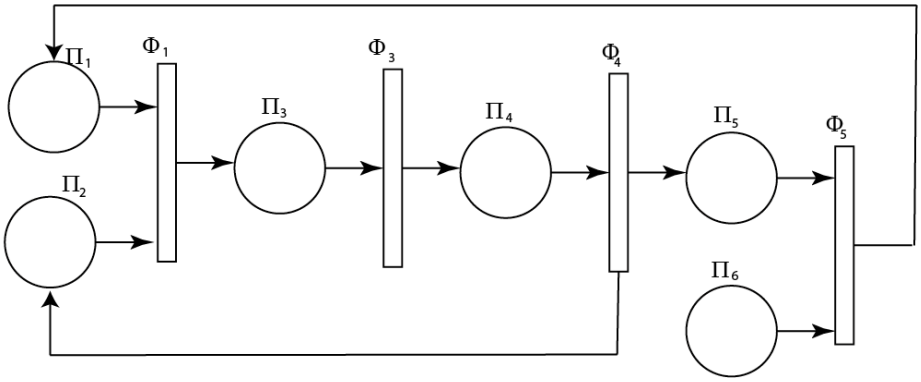

Figure 5.3: The TCPN model

Notice that this structure of TCPN is general: a different task is is described by a different marking in Π_1 and update rule of Φ_5.

.

Hence, the scheduling optimality issue is here addressed by resorting to reinforcement learning (RL) technicalities [2], where the main aim consists in avoiding the generation of the reachability graph and selecting the more adequate operation to be executed according to the current reward function. Specifically, a deep Q-learning approach [167] is adopted and customized as follows:

- **Environment** - the TCPN

$$TCPN = (\Pi, \Phi, Arcs, Post, Pre, CS, Var, PC, Gu, AE, Ini)$$

- **RL state** - $s_{TCPN} \in \mathcal{S}_{TCPN}$ the colored marking of the TCPN;

- **Actions** -

$$\mathcal{A}_{TCPN}(t) = \{a_{TCPN} = op_j \in \mathcal{A}_t(s_{TCPN})\} \tag{5.9}$$

where $\mathcal{A}_t(s_{TCPN})$ is the set of the operations ready to be executed at time t;

- **Reward**:

$$r_{TCPN}(t) := \beta_1 OF(t) + \beta_2 JA(t) \tag{5.10}$$

with $\beta_i \in \mathbb{R},\ i = \{1, 2\}$ positive scalars and

  - $OF(t)$ accounting for the set occupancy factor, i.e., the percentage of machines active at the current time instant;
  - $JA(t)$ an heuristic characterizing the job accomplishment with respect to the time evolution;

  The role of the positive scalar parameters β_1 and β_2 exhibits a trade-off characteristic in the reward implementation. Parameter β_1 weights the transient aspect, specifically the occupancy factor of the machines at each time instant, while β_2 is related the overall time taken to complete the task. Parameter β_2 can be kept fixed, β_1 used as a tunable design knob and vice-versa: as an example (first case, α_2 fixed and β_1 tunable) the lower β_1, the larger the percentage of active machines, and the *faster* the time job accomplishment. An *opposite direction* trend, $alpha_1$ larger, means less occupancy factor and **slower** time job accomplishment.

- **State-action value function** -

$$\mathcal{Q}_{TCPN}(s_{TCPN}(t), a_{TCPN}(t); \theta_{TCPN}) := E\left[\sum_{k=1}^{\infty} \gamma_{TCPN}^{(k-1)}\, r_{TCPN}(t + k\Delta(t))\right] \tag{5.11}$$

where $\Delta(t)$ the time at which a new action has to be chosen;

- **Policy** - epsilon-greedy approach [30]:

$$a_{TCPN}(t) = \begin{cases} \text{rand}(\mathcal{A}_p(s_{TCPN})), & \text{probability } \epsilon(t) \\ a^*_{TCPN}, & \text{otherwise} \end{cases} \tag{5.12}$$

where $\mathcal{A}_p(s_{TCPN}) \subseteq \mathcal{A}_{TCPN}$ is the set of possible actions on $s_{TCPN} \in \mathcal{S}_{TCPN}$ and

$$a^*_{TCPN} := arg \max_{a_{TCPN} \in \mathcal{A}_p(s_{TCPN})} \mathcal{Q}_{TCPN}(s_{TCPN}, a_{TCPN})$$

Based on these developments, the following algorithm provides a near-optimal scheduling complying with the FMS pair $(Ma, L/U)$ and the task $\{J_1, \ldots, J_\eta\}$.

DQL-TCPN-Algorithm

Initialize $\mathcal{Q}_{TCPN}(s_{TCPN}(t), a_{TCPN}(t); \theta_{TCPN})$

For *each episode*

1: **Read** $s_{TCPN}(0)$

For *each decision time t*

1: **Select** by (5.12) the operation to be executed;

2: **Read** the RL state $s_{TCPN}(t + \Delta(t))$;

3: **Compute** the reward $r_{TCPN}(t + \Delta(t))$;

4: **Update** θ_{TCPN};

5.5 A distributed Deep Reinforcement Learning algorithm for routing decisions

Inspired by the ideas developed in Chapter 4, the routing decision layer is implemented by resorting to a multi-agent deep Q-learning (DQL) approach. Accordingly, the model has been customized as follows:

- **Agents** - $AV := \{\{AV_j^i\}_{i=1}^{N_j}\}_{j=1}^{q}$ the set of vehicles in charge to accomplish the DQL task;
- **Environment** - the manufacturing system layout described as the connected graph derived by the geometry of the corridors $\{CO^k\}_{k=1}^{M}$;
- **DQL state** - $s_j^i(t)$ - the edges where the vehicle AV_j^i is located at the time instant t;
- **Actions** - the set $\mathcal{A}$ a finite set of integer numbers accounting for all the possible vehicle actions;
- **Global reward** - $\Psi : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$ is the same for all the AVs and takes care of vehicle congestion occurrences over the FMS;
- **Reward** -

$$r_j^i(t_{dec} + \Delta t_{dec}(t)) = \Psi\left(\bigcup_k (s_j^k)(t_{dec} + \Delta t_{dec}(t))\right) + (\psi_j^i)_{goal}(t_{dec} + \Delta t_{dec}(t)) + (\psi_j^i)_{wt_j^i}(t_{dec} + \Delta t_{dec}(t)) \tag{5.13}$$

where the symbols have the same meaning of Section 4.4.

- **State-action value function** -

$$\mathcal{Q}_j^i((s_j^i)(t),(a_j^i)_R(t);\theta_j^i) := E\left[\sum_{k=1}^{\infty}\gamma^{(k-1)}\, r^i(t+k\Delta t_{dec}(t))\right] \tag{5.14}$$

where $(a_j^i)_R(t) \in \Lambda$, θ_j^i is the NN weighting vector associated to each vehicle AV_j^i and $\gamma \in (0,1)$ the discount factor.

- **Policy** - Again an epsilon-greedy approach:

$$(a_j^i)_R = \begin{cases} rand(\Lambda), & \text{with probability } \epsilon \\ \arg\max_{a\in\mathcal{A}} \mathcal{Q}_j^i((s_j^i)(t_{dec}),a;\theta_j^i), & \text{otherwise} \end{cases} \tag{5.15}$$

In view of the above customization, a distributed deep Q-learning for routing decisions (**D-DQL-DR**) algorithm is as follows:

D-DQL-RL Algorithm (Agent AV_j^i)

Initialize $\mathcal{Q}_j^i((s_j^i),a;\theta_j^i)$

For *each episode*

1: **Read** $(s_j^i)(0)$;

For *each* t_{dec}

1: **Select** by (5.15) the action a_j^i to be executed;

2: **Read** the DQL state $(s_j^i)(t_{dec}+\Delta t_{dec}(t))$;

3: **Compute** the reward $r_j^i(t_{dec}+\Delta t_{dec}(t))$;

4: **Update** θ_j^i;

5.6 A set-theoretic control scheme for path planning and anti-collision

In this section, the core of the control architecture of figure 5.2 is designed. This is done by resorting to a set-theoretic model predictive control approach whose main properties have been deeply discussed in [7], [8] and here are combined with scheduling and routing decision layers in a distributed/parallel fashion. According to the prescriptions of the **PL-MS** problem, three issues have to be properly addressed:

1. compute state trajectory tubes in accordance with the q platoons LF^j;

2. develop an efficient anti-collision procedure complying with the job priorities $\{pr_1, \ldots, pr_{\overline{\jmath}}\}$;

3. derive a parallel/distributed control architecture to accomplish the prescribed task $\{\mathcal{J}_1, \ldots, \mathcal{J}_{\overline{\jmath}}\}$.

5.6.1 Platoon path tubes

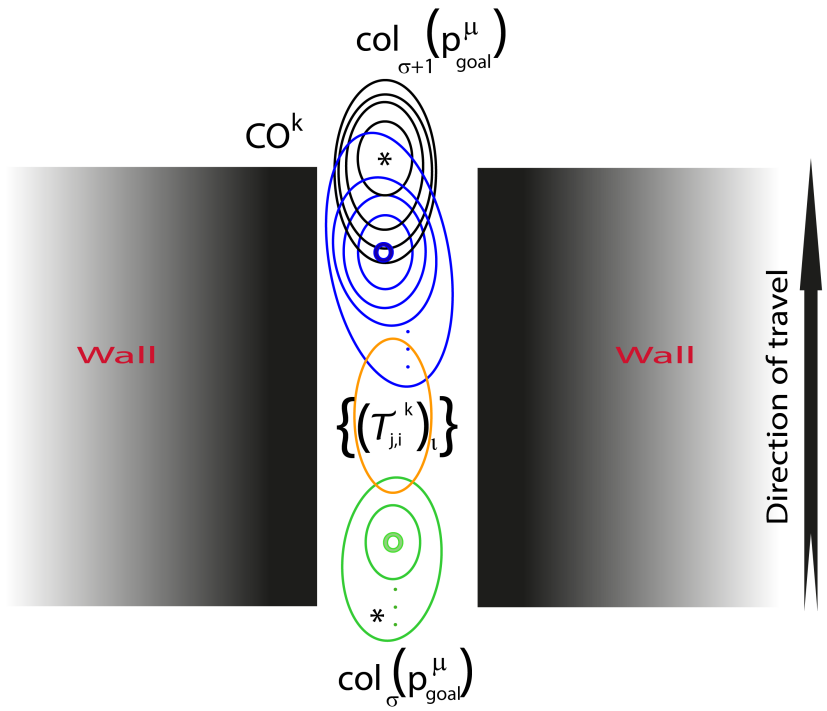

Figure 5.4: Corridor state trajectory tube

A solution for question 1) is provided in the off-line phase by using set-theoretic arguments [154]. For the sake of comprehension, the simplified scenario of figure 5.4 is considered to begin with: a platoon LF^j, a corridor CO^k and a job $\mathcal{J}_\mu$ whose two scheduled way-points are $col_\sigma(p^\mu_{goal})$ and $col_{\sigma+1}(p^\mu_{goal})$, respectively. Then, the state trajectory tube complying with the direction of travel and with the initial and final equilibria $col_\sigma(p^\mu_{goal})$

and $col_{\sigma+1}(p^{\mu}_{goal})$, is obtained by computing a family of one-step state ahead controllable sets, hereafter named $\{(\mathcal{T}^k_{j,i})_\iota\}$. To this end, the state constraints of the vehicle AV^i_j are re-defined by taking into account the corridor CO^k :

$$\mathcal{X}^k_{j,i} := \mathcal{X}^i_j \cap CO^k,\ i = 1, \ldots, N_j \tag{5.16}$$

By considering the One-step Controllable Set Procedure (**OCSP**) presented in [154], the tube $\{(\mathcal{T}^k_{j,i})_\iota\}$ is computed according the following reasoning. An initialization phase prescribes the computation of the pair $((\mathcal{T}^k_{j,i})_0, (K^k_{i,j})_0)$, consisting of a PI region centered at $col_{\sigma+1}(p^{\mu}_{goal})$ (see the first black ellipsoid in figure 5.4) and its associated stabilizing state-feedback control law. Then, a sequence of one-step state ahead controllable sets $(\mathcal{T}^k_{i,j})_\iota$ is determined until a saturation effect comes out, i.e., the last black ellipsoid. Hence, an intermediate equilibrium (the blue circle) is chosen and the above steps are done again until the way point $col_{\sigma}(p^{\mu}_{goal})$ is covered by the last green ellipsoid. Since a multi-agent configuration is considered, both PI regions and controllable sets must be adequately computed:

Positively invariant regions

They are designed by preserving the Cartesian structure

$$(\mathcal{T}^k_j)_0 := \prod_{i=1}^{N_j} (\mathcal{T}^k_{j,i})_0 \tag{5.17}$$

where $(\mathcal{T}^k_j)_0 \subset \mathbb{R}^{n_i N_j}$ accounts for the PI set of the centralized system achievable by collecting all the N_j vehicles (5.4). Since such sets are independent each other, the pairs $((\mathcal{T}^k_{j,i})_0, (K^k_{j,i})_0)$, $i = 1, \ldots, N_j$, are achievable by standard technicalities, see e.g. [7]. Details regarding the computation of the PI sets can be found in [8] (Ellipsoidal case) and [168] (Polyhedral case).

One-step state ahead controllable sets

These regions are instead computed by carefully taking into account that one-step state predictions are evaluated along interacting vehicles. Then, the sequences $\{(\mathcal{T}^k_{j,i})_\iota\}$, $i = 1, \ldots, N_j$, are obtained by means of the following recursions:

$$(\mathcal{T}^k_{j,1})_\iota = \{x^1_j \in \mathcal{X}^{j,1}\ \exists u \in \mathcal{U}^1_j, A^1_j x^1_j + B^1_j u^1_j \in (\mathcal{T}^k_{j,1})_{\iota-1}\} \tag{5.18}$$

$$
\begin{aligned}
(\mathcal{T}^k_{j,i})_\iota = \{ & x^i_j \in \mathcal{X}_{j,i} : \exists u \in \mathcal{U}^i_j, A^i_j x^i_j + B^i u^i_j \in (\mathcal{T}^k_{j,i})_{\iota-1} \\
& d_{min} \leq dist(x^i_j, (\mathcal{T}^k_{j,i-1})_\iota) \leq d_{max} \\
& d_{min} \leq dist(A^i_j x^i_j + B^i_j u^i_j, (\mathcal{T}^k_{j,i-1})_{\iota-1}) \leq d_{max} \}, \\
& i = 2, \ldots, N_j;
\end{aligned}
\tag{5.19}
$$

$$
(\mathcal{T}^k_{j,i})_{\iota-1} \subset (\mathcal{T}^k_{j,i})_\iota, \, \forall \iota, \, i = 1, \ldots, N_j \tag{5.20}
$$

where d_{min}, $d_{max} \in \mathbb{R}^+$ are *a-priori* known and guaranteed bounds. The argument behind (5.18)-(5.20) can be summarized as follows. At each step, the leader one-step controllable region is computed by complying with the corridor constraints (5.16) and irrespective of any LF configuration requirement. Conversely, the AV followers must take care of coordination constraints to avoid any collisions with their predecessors along the chain. The latter translates into the following set-containment conditions:

- $d_{min} \leq dist(x^i_j, \mathcal{T}^{i-1}_j) \leq d_{max}$: the set of state x^i_j is selected such that the distance from the same-level controllable region of the predecessor AV along the LF chain is bounded by d_{min} and d_{max}, i.e $(\mathcal{T}^k_{j,i})_{\iota-1} \cap (\mathcal{T}^k_{j,i-1})_\iota$;

- $d_{min} \leq dist(A^i_j x^i_j + B^i_j u^i_j, (\mathcal{T}^k_{j,i-1})_{\iota-1}) \leq d_{max}$: the set of one-step ahead state prediction $X^{i^+}_j := A^i_j x^i_j + B^i_j u^i_j$ is such that $X^{i^+}_j \cap (\mathcal{T}^k_{j,i-1})_{\iota-1} = \emptyset$.

In the above developments, the simplified scenario of figure 5.4 is in charge to formalize the technicalities under which the tube $\{(\mathcal{T}^k_{j,i})_\iota\}$ has to be built up. Anyway, a key aspect has been overruled: in real contexts, the sequence of way-points P^μ_{goal} could be located everywhere along the FMS planimetry and not necessarily at the entrance and exit of each involved corridor. To comply with this issue, the sketch of figure 5.5 is hereafter considered, where the way points $col_\sigma(p^\mu_{goal})$ and $col_{\sigma+1}(p^\mu_{goal})$ are located along the two corridors CO^{k_1} and CO^{k_2}. Again, the off-line aim is to define a path tube to be used during the on-line operations for driving the platoon $\mathbf{LF}^j$ from $col_\sigma(p^\mu_{goal})$ to $col_{\sigma+1}(p^\mu_{goal})$. According to the previous analysis, the family $\{(\mathcal{T}^{k_1}_{j,i})_\iota\}$ is first computed by choosing as the target an equilibrium $\bar{\mathbf{x}}$ located at the junction just after the end of CO^{k_1} (this is always viable because the structured environment is known in advance). Then, the family $\{(\mathcal{T}^{k_2}_{j,i})_\iota\}$ is derived according to the following further requirements:

- the equilibrium $\bar{\mathbf{x}} \in CO^{k_1} \cap CO^{k_2}$ acts as the initial way-point;

- $$\bar{\mathbf{x}} \in \{(\mathcal{T}_{j,i}^{k_1})_\iota\} \cap \{(\mathcal{T}_{j,i}^{k_2})_\iota\} \neq \emptyset \tag{5.21}$$

Essentially, the condition (5.21) allows to formally connect CO^{k_1} and CO^{k_2} by making viable the actions provided by the routing decision unit.

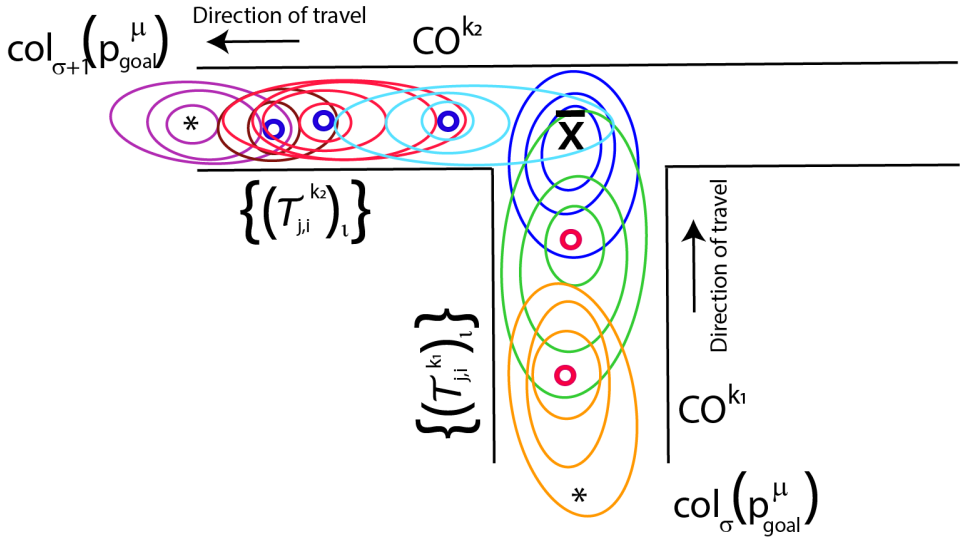

Figure 5.5: Path state trajectory tube

By collecting these arguments, the path state trajectory tubes for the involved platoons can be defined as follows:

$$\bigcup_{i=1}^{L_j} \bigcup_{k} \bigcup_{\iota} (\mathcal{T}_{j,i}^{k})_\iota, j = 1, \ldots, q \tag{5.22}$$

5.6.2 Anti-collision procedure

The second aspect to address concerns with the on-line operations: the capability to avoid collisions among platoons when they travel along the same corridor in opposite directions. To better clarify the idea here developed, the illustrative example of figure 5.6 is analyzed. There, the platoons $\mathbf{LF}^{j_1}$ and $\mathbf{LF}^{j_2}$ are considered with $pr_{j_1} > pr_{j_2}$ prescribed priorities. When the two platoons recognize each other, $\bar{t}$, the leader vehicles, $AV_1^{j_1}$ and $AV_1^{j_2}$, share the respective information about platoon lengths and priorities. Accordingly, the low priority platoon $\mathbf{LF}^{j_2}$ stops and from $\bar{t}$ onward acts as a static obstacle for $\mathbf{LF}^{j_1}$ until it is worked around.

As the control strategy is concerned, the idea is to generate a sequence of overlapped PI regions (see continuous red line ellipsoids in figure 5.6) until the obstacle platoon $\mathbf{LF}^{j_2}$ is

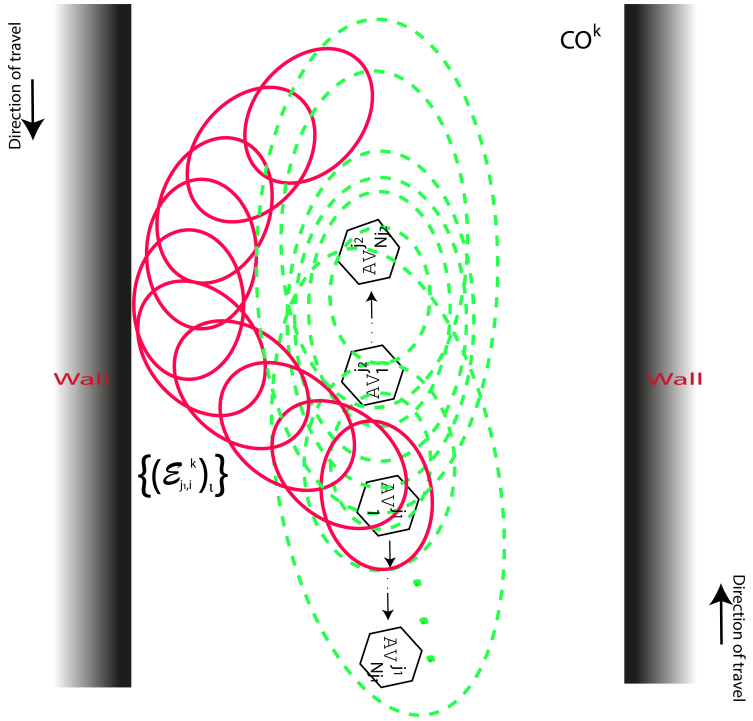

Figure 5.6: Avoiding low-priority platoon

overcome. This choice guarantees that during the online operations each vehicle can move by avoiding the obstacles thanks to admissible swtichings between two regions.

Since the PI sets must take care of safe distances amongst the vehicles, therefore they have to be derived by a level-by-level procedure. Specifically, first the PI leader is derived without adding any distance constraint, then the direct follower (second agent) computes its PI under a distance requirement w.r.t. the leader region, and so on until the leaf node. According to this reasoning, feasible switchings along the pre-computed sets are always viable and, as a consequence, there exists an admissible path until bypassing the obstacle $\mathbf{LF}_b$. Then by denoting with $(\mathcal{E}^k_{j_1,i})_\iota$ the PI set of $AV^i_{j_1}$, the distance constraint is:

$$d_{min} \leq dist(x^i_{j_1}, (\mathcal{E}^k_{j_1,i-1})_\iota) \leq d_{max} \tag{5.23}$$

and, as a consequence,

$$(\mathcal{E}^k_{j_1,i})_\iota \cap (\mathcal{E}^k_{j_1,i-1})_\iota = \emptyset$$

In view of this reasoning, the regions $(\mathcal{E}^k_{j_1,1})_\iota$ are computed by the following optimization:

DMPC-Leader:

$$\min_{(K^k_{j_1,1})_\iota, (\mathcal{E}^k_{a,1})_\iota} J^1_{j_1}(t) \text{ s.t} \tag{5.24}$$

$$(A^{j_1}_1 + B^{j_1}_1 (K^k_{j_1,1})_\iota)(\mathcal{E}^k_{j_1,1})_\iota \subset (\mathcal{E}^k_{j_1,1})_\iota \subseteq \mathcal{X}^k_{j_1,1} \tag{5.25}$$

$$(K^k_{j_1,1})_\iota(\mathcal{E}^k_{j_1,1})_\iota \subset \mathcal{U}^1_{j_1} \tag{5.26}$$

$$Proj_p\left((\mathcal{E}^k_{j_1,1})_\iota\right) \subseteq \mathcal{B}(p^1_{j_1}(t) + R_{max}, R) \tag{5.27}$$

$$(\bar{x}^k_{j_1,1})_\iota \in (\mathcal{E}^k_{j_1,1})_{\iota-1} \cap (\mathcal{E}^k_{j_1,1})_\iota \tag{5.28}$$

Conversely, as the followers are concerned, one has:

DMPC-Follower:

$$\min_{(K^k_{j_1,i})_\iota,(\mathcal{E}^k_{a,i})_\iota} J^i_{j_1}(t) \text{ s.t} \tag{5.29}$$

$$(A^{j_1}_i + B^{j_1}_i(K^k_{j_1,i})_\iota)(\mathcal{E}^k_{j_1,i})_\iota \subset (\mathcal{E}^k_{j_1,i})_\iota \subseteq \mathcal{X}^k_{j_1,i} \tag{5.30}$$

$$(K^k_{j_1,i})_\iota(\mathcal{E}^k_{j_1,i})_\iota \subset \mathcal{U}^i_{j_1} \tag{5.31}$$

$$Proj_p\left((\mathcal{E}^k_{j_1,i})_\iota\right) \subseteq \mathcal{B}(p^i_{j_1}(t) + R_{max}, R) \tag{5.32}$$

$$(\bar{x}^k_{j_1,i})_\iota \in (\mathcal{E}^k_{j_1,i})_{\iota-1} \cap (\mathcal{E}^k_{j_1,i})_\iota \tag{5.33}$$

$$d_{min} \leq dist(x^i_{j_1}, (\mathcal{E}^k_{j_1,i-1})_\iota) \leq d_{max}, \forall x^i_{j_1} \in (\mathcal{E}^k_{j_1,i})_\iota \tag{5.34}$$

where $(\bar{x}^k_{j_!,i})_\iota$ is the equilibrium selected at each $\iota - th$ iteration as follows

$$\begin{aligned}(\bar{x}^k_{j_!,i})_\iota := \quad & \arg \max_{x^i_{j1} \in (\mathcal{E}^k_{j1,i-1})_{\iota-1}} \|(\xi^k_{j_1,i})_{\iota-1} - x^i_{j_1}\|_2 \\ & \text{s.t. } \exists (u^k_{j_1,i})_\iota \in \mathcal{U}^i_{j_1}, x^i_{j_1} = (I_n - A^{j_1}_i)^{-1} B^{j_1}_i{}^{(u^k_{j_1,i})_\iota}\end{aligned} \tag{5.35}$$

and

$$J^i_{j_1}(t) := \sum_{h=0}^{\infty} [\|x^i_{j_1}(t+h|t) - (\xi^k_{j_1,i})_\iota\|^2_{R_{x^i_{j_1}}} + \|u^i_{j_1}(t+h|t) - (u^k_{j_1,i})_\iota\|^2_{R_{u^i_{j_1}}}], \forall i, \tag{5.36}$$

with $R_{x^i_{j_1}} > 0$ and $R_{u^i_{j_1}} \geq 0$ symmetric shaping matrices. In view of these developments, each PI region $(\mathcal{E}^k_{j_1,i})_\iota$ is centered at $(\bar{x}^k_{j_1,i})_\iota$ and

$$(\mathcal{E}^k_{j_!,i})_\iota \cap (\mathcal{E}^k_{j_1,i-1})_\iota \neq \emptyset \tag{5.37}$$

Hence, the results of Preposition 6, guarantees that at each time instant t there exists a control action capable of moving the platoon $\mathbf{LF}_{j_1}$ along the overlapped sets.

5.6.3 Minimum-time control strategy

For the sake of comprehension, a single platoon $\mathbf{LF}^j$ moving along a single corridor CO^k is considered at first: there $z^j_1(\hat{t})$ and $z^j_1(\bar{t})$, with $\bar{t} \geq \hat{t}$, denote the platoon leader routing decisions at the entrance and the exit of CO^k, respectively.

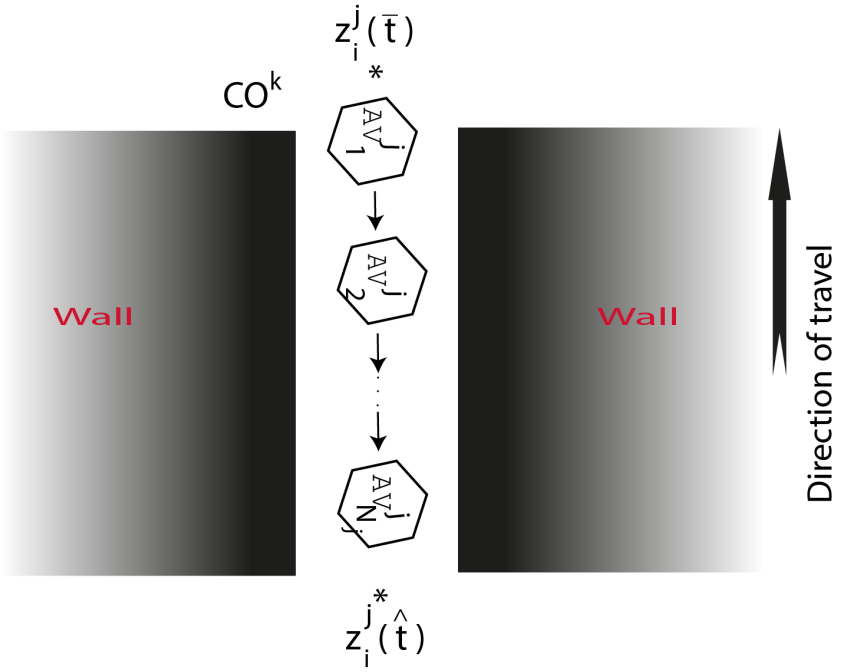

Figure 5.7: On-line corridor crossing

During the on-line operations, the local commands for each vehicle AV_i^j are obtained by solving the following semi-definite programming problem (SDP):

$$u_i^j(t) = \arg\min_{u_i^j} J_{\iota(t)}(x_i^j(t), u_i^j) \tag{5.38}$$

subject to

$$A_i^j x_i^j(t) + B_i^j u_i^j \in (\mathcal{T}_{j,i}^k)_{\iota(t)-1}, u_i^j \in \mathcal{U}_i^j \tag{5.39}$$

In this context, the running cost $J_{\iota(t)}(x^i(t), u^i)$ accounts for the approximate minimum time criterion:

$$J_{\iota(t)}(x_i^j(t), u_i^j) := \| Proj_p \left(A_i^j x_i^j(t) + B_i^j u_i^j \right) - z_i^j(\bar{t}) \|^2_{(P_{j,i}^k)^{-1}_{\iota(t)-1}} \tag{5.40}$$

with $(P_{j,i}^k)_{\iota(t)-1}$ the shaping matrix of $(\mathcal{T}_{j,i}^k)_{\iota(t)-1}$ along the corridor CO^k.

Then, the following parallel algorithm is adopted to address the **PL-MS** prescriptions.

Parallel Deep Reinforcement Learning Receding Horizon Control (P-DRL-RHC) Algorithm - Platoon PL$_j$

Input: d_{min}; d_{max}; p_{goal}^{μ}; pr_{μ}; $x_i^j(0)$;

Output: $u_i^{j^*}(t|t)$;

Initialization -

1: **compute** $(\mathcal{T}_{j,i}^k)_0$, $i = 1, \ldots, N_j$, $\forall$ viable k; ▷ the PI regions

2: **compute** $(K_{j,i}^k)_0$, $i = 1, \ldots, N_j$, ▷ the stabilizing state feedback gains such that (5.5) are fulfilled;

3: **generate** the tubes $\bigcup_k \left\{(\mathcal{T}_{j,i}^k)_\iota\right\}$, $i = 1, \ldots, N_j$, via recursions (5.18)-(5.20) such that

$$x^j(0) \in \bigcup_\iota \bigcup_k \left\{(\mathcal{T}_{j,1}^k)_\iota \times \cdots \times (\mathcal{T}_{j,L_j}^k)_\iota\right\}$$

4: **store** $\bigcup_k \left\{\left(\mathcal{T}_{j,i}^k\right)_\iota\right\}$ and $(K_{j,i}^k)_0, i = 1, \ldots, L_j$.

P-DRL-RHC - On-line - Vehicle AV_j^i

1: **Identify** $CO^{k(t)}$;

2: **Acquire** $x_i^(t)$;

3: **if** $x_i^j(t) \in DZ^{k(t)}$ **then update** the action $a_i^j(t)$ via the **D-DQL-RL** unit;

4: **Generate** the routing $z_j^i(t)$ by (5.8);

5: **end if**

6: **if** for some $\mathbf{LF}^{j_1}$, $\mathbf{LF}^{j_2}$ travel along the same corridor in opposite directions **then goto** Step 17 ▷ a collision may happen

7: **Find** $\iota(t) = min\left\{\iota : x_i^j(t) \in \left(\mathcal{T}_{j,i}^{k(t)}\right)_\iota\right\}$

8: **if** $i > 1$ **then receive** $\iota(t-1)$ from AV_{i-1}^j and **assign** $\iota_{pre} := \iota(t-1)$;

9: **if** $\iota(t) < \iota_{pre}$ **then** $\iota(t) \leftarrow \iota_{pre}$

10: **end if**

11: **end if**

12: **if** $\iota(t) == 1$ **then** $u_i^{j^*}(t) = \left(K_{j,i}^k(t)\right)_0 x_i^j(t)$

13: **else**

Solve the SDP (5.38)-(5.39);

14: **end if**

15: **if** $i < N_j$ **then transmit** $\iota(t)$ to AV_{i+1}^j;

16: **end if**

17: **if** $pr_{j_1} > pr_{j_2}$ **then Block** the platoon $\mathbf{PL}_{j_2}$

18: **if** $i = 1$ **then compute** the PI region $(\mathcal{E}_{j,i}^k)_\iota$ by solving the **DMPC**-Leader optimization (5.24)-(5.28);

19: **else**

solve the **DMPC**-Follower optimization (5.29)-(5.34);

20: **end if**

21: **Compute** $u_i^{j^*}(t) = \left(K_{j,i}^k(t)\right)_0 x_i^j(t)$

22: **else**

Block the platoon $\mathbf{PL}_{j_2}$;

23: **Repeat** Steps 18-21;

24: **end if**

25: **end if**

26: **Apply** $u_j^{i^*}(t)$;

27: $t \leftarrow t+1$ and **goto** Step 1;

The main properties of the **P-DRL-RHC** Algorithm are summarized in the following proposition.

Proposition 11 *Given the task* $\{\mathcal{J}^1, \ldots, \mathcal{J}^{\urcorner}\}$, *the associated priorities* $\{pr_1, \ldots, pr_{\urcorner}\}$ *and a set of vehicles organized as* q *platoons* $\mathbf{LF}^j = \{AV_i^j\}_{i=1}^{N_j}$, $j = 1, \ldots, q$. *Then, the* **P-DRL-RHC** *Algorithm accomplishes all the prescribed jobs by always ensuring constraints satisfaction and enhancing anti-collision capabilities of all the involved platoons. Moreover, the regulated state trajectories are asymptotically stable.*

Proof - Starting from the assumption that the optimal job scheduling $\{p_{goal}^1, \ldots, p_{goal}^{\urcorner}\}$ has been obtained by using the **DQL-TCPN** Algorithm, the feasibility property results by separately analyzing two operation modes.

Normal mode: at each time instant t the leader AV_1^j identifies the corridor $k(t)$ so that the family of one-step state ahead controllable sets $\left\{\left(\mathcal{T}_{j,i}^{k(t)}\right)_\iota\right\}$ is considered. After selecting the set-membership level $\iota(t)$, the leader command $u_i^{j^*}(t)$ is computed and the same $\iota(t)$ is transmitted to all the followers in order to ensure that the same level is used. In virtue of this, the optimization (5.38)-(5.39) has always a solution for any vehicle AV_i^j, $i = 1, \ldots, N_j$;

Anti-collision mode: by construction the sequence $\left\{(\mathcal{E}_{j,i}^k)_\iota\right\}$ is such that

$$\left\{(\mathcal{E}_{j,i}^k)_\iota\right\} \cap \left\{\left(\mathcal{T}_{j,i}^{k(t)}\right)_\iota\right\} \neq \emptyset$$

for some $\iota > 0$. Let $x_i^j(t) \in \left(\mathcal{E}_{j,i}^k\right)_{\tilde{t}}, \tilde{t} \geq t$ be the current state measurement, then in a finite number of steps the state trajectory is driven into $\left\{\left(\mathcal{T}_{j,i}^{k(t)}\right)_\iota\right\}$ because *Proposition* 6 ensures that the state trajectory is driven to $\left(\mathcal{E}_{j,i}^k\right)_{\tilde{t}+1}$ and there always

exists an index $s \geq \tilde{\iota} + 1$ such that

$$\left(\mathcal{E}_{j,i}^{k}\right)_{s} \cap \left\{\left(\mathcal{T}_{j,i}^{k(t)}\right)_{\iota}\right\} \neq \emptyset$$

Finally, the closed-loop asymptotic stability derives from the same analysis because, according to the set of routing decisions $\{z_j^i(t)\}$ the platoon $\mathbf{LF}^j$ accomplishes the scheduling p_{goal}^{μ} as $t \to \infty$. □

5.7 Numerical results

In this section, the effectiveness of the proposed **P-DRL-RHC** algorithm is evaluated by means of a simulation campaign, whose setup is implemented within the Matlab environment. For the subsequent developments, autonomous vehicles are described by the following model:

$$A^i = \begin{bmatrix} I_2 & T_s\, I_2 \\ 0_2 & I_2 \end{bmatrix},\ B^i = \begin{bmatrix} \frac{(T_s)^2\, I_2}{2} \\ T_s I_2 \end{bmatrix},$$

with $T_s = 0.5$ and subject to the saturation constraint

$$|row_v(u^i(t))| \leq 1, v = 1, 2,\ \forall t \geq 0, \forall i. \tag{5.41}$$

The geometrical displacement between two consecutive vehicles is $d_{min} = 1[m]$ and $d_{max} = 10[m]$. Moreover, $R = 5[m]$ and $R_{min}^c = 1[m]$.

Operating scenario- *Given the FMS of figure 5.8, defined on an area of* $2500\,[m^2]$*, the task of five jobs* $\{\mathcal{J}_1, \ldots, \mathcal{J}_5\}$ *fully described in Table 5.1 with priorities* $pr_1 = 0.45, pr_2 = 0.2, pr_3 = 0.1, pr_4 = 0.25$*, is required to be accomplished.* □

As the scheduling phase is concerned,the TCPN of figure 5.3 has been initialized with 5 tokens in Π_1. The update rules of transition Φ_5 have been set accordingly to Table 5.1. The number of tokens in Π_2 depends on the vehicles configuration. Figure 5.9 represents the TCPN used to compute the scheduling when the number of vehicles is 5.

The training of the Q-function (5.11) has been addressed within the MATLAB environment by using Reinforcement Learning Toolbox built-in routines. In particular, it has been carried out by generating 12000 episodes, where each of them consists of a time ordered sequence of all the prescribed operations reported in Table 5.1. The resulting training evolution is shown in figure 5.10, where one can see that the capability to optimize the

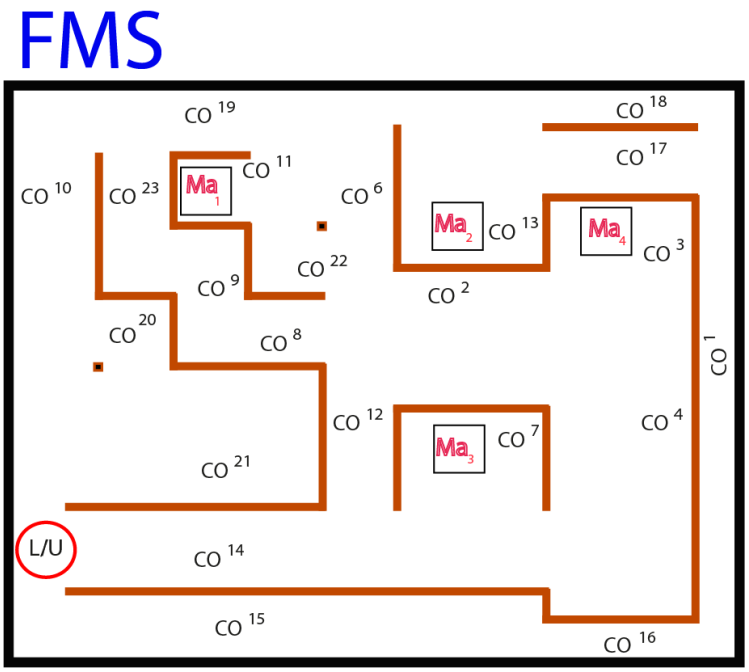

Figure 5.8: FMS planimetry: machines and L/U locations

sequence of jobs asymptotically improves, i.e., the expected reward (orange line) converges to the episode one.

Conversely, the training for the distributed routing decision units has been performed by considering 1500 episodes for each vehicle. By referring to a platoon $\mathbf{LF}^j$ whose aim is to accomplish a subset of jobs, say $\mathcal{J}_{sub} \subseteq \{\mathcal{J}_1, \ldots, \mathcal{J}_5\}$, then the episode for AV_i^j consists in a sequence of time instants with initial, partial and final AV_i^j positions complying with the operations pertaining to $\mathcal{J}_{sub}$ as indicated in Table 5.1. For the sake of completeness, a training process is reported in figure 5.11.

Job	**Operations**	**Idle time units**
$\mathcal{J}_1$	$op_1^1 : L/U \rightarrow Ma_1$	$M_1 = 20$
	$op_1^2 : Ma_1 \rightarrow Ma_2$	$Ma_2 = 30$
	$op_1^3 : Ma_2 \rightarrow Ma_4$	$Ma_4 = 20$
	$op_1^4 : Ma_4 \rightarrow L/U$	$L/U = 20$
$\mathcal{J}_2$	$op_2^1 : L/U \rightarrow Ma_3$	$Ma_3 = 30$
	$op_2^2 : Ma_3 \rightarrow Ma_2$	$Ma_2 = 20$
	$op_2^3 : Ma_2 \rightarrow L/U$	$L/U = 20$
$\mathcal{J}_3$	$op_3^1 : L/U \rightarrow Ma_4$	$Ma_4 = 30$
	$op_3^2 : Ma_4 \rightarrow Ma_1$	$Ma_1 = 40$
	$op_3^3 : Ma_1 \rightarrow L/U$	$L/U = 20$
$\mathcal{J}_4$	$op_4^1 : L/U \rightarrow Ma_1$	$Ma_1 = 20$
	$op_4^2 : Ma_1 \rightarrow Ma_3$	$Ma_3 = 20$
	$op_4^3 : Ma_3 \rightarrow L/U$	$L/U = 20$
$\mathcal{J}_5$	$op_5^1 : L/U \rightarrow Ma_4$	$Ma_4 = 20$
	$op_5^2 : Ma_4 \rightarrow Ma_3$	$Ma_3 = 30$
	$op_5^3 : Ma_3 \rightarrow Ma_1$	$Ma_1 = 40$
	$op_5^4 : Ma_1 \rightarrow L/U$	$L/U = 20$

Table 5.1: Task operations

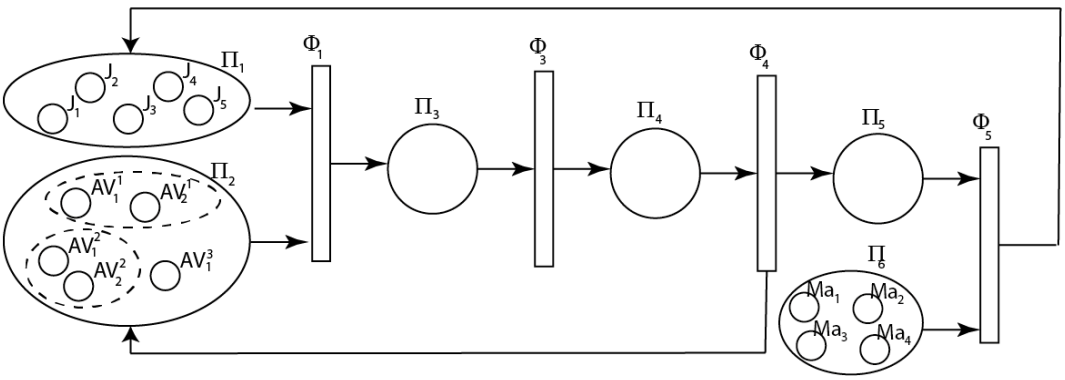

Figure 5.9: The TCPN model when $\#Vehicles = 5$

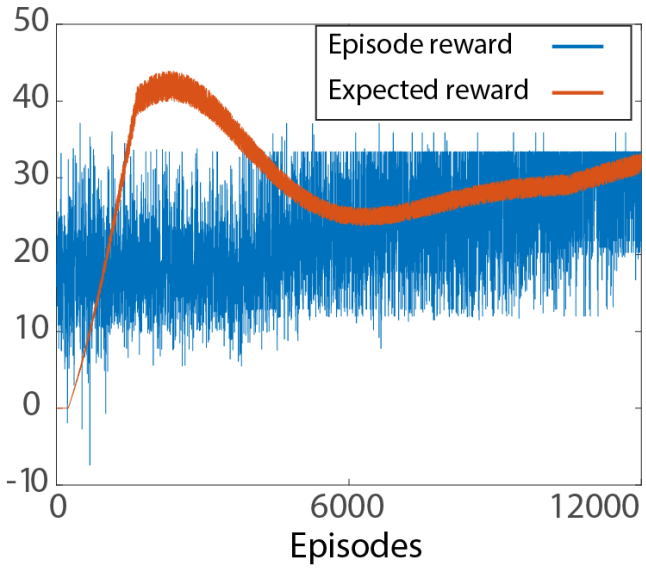

Figure 5.10: DQL training

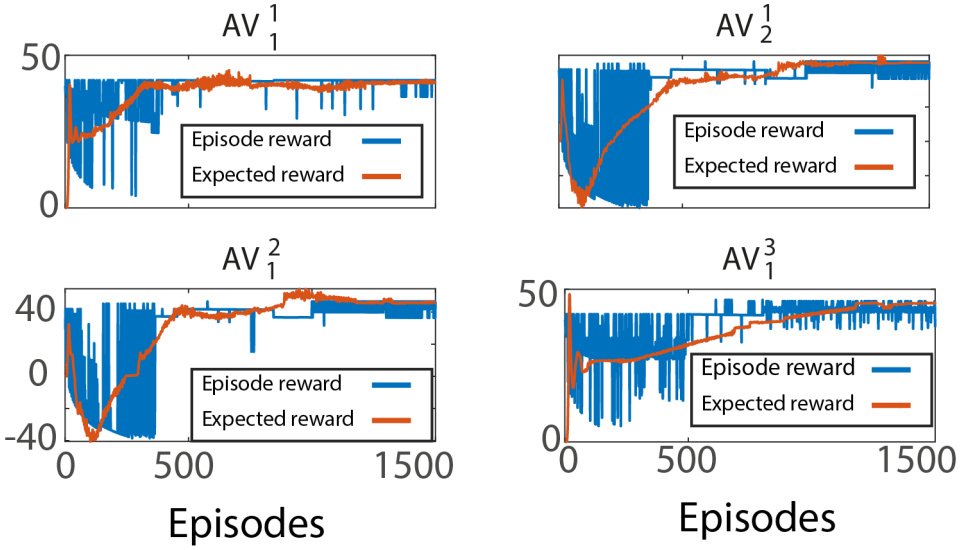

Figure 5.11: DRL training: $\mathbf{LF}^1 = \{AV_1^1, AV_2^1\}$, $\mathbf{LF}^2 = \{AV_1^2\}$, $\mathbf{LF}^3 = \{AV_1^3\}$

The aim of the simulation is to determine an optimal configuration in terms of number of involved autonomous vehicles and leader-follower organization according to the following criterion:

$$\uparrow := \max_{j=1,\dots,q,\ i=1,\dots,N_j} \sum_{k} \Delta T^{j,i}_{\mathcal{J}_k},$$

where $\Delta T^{\mathcal{J},i}_{J_k}$ is the time interval required to accomplish the job $\mathcal{J}_k$ associated to the vehicle AV^j_i. Accordingly, numerical scenarios has been defined by increasing the number of vehicles:

# **Vehicles**	**Platoons**	**Scheduling**
1	$\mathbf{LF}^1 = \{AV^1_1\}$	$\mathcal{J}_1 \to \mathcal{J}_2 \to \mathcal{J}_3$
		$\mathcal{J}_3 \to \mathcal{J}_4 \to \mathcal{J}_5$
2	$\mathbf{LF}^1 = \{AV^1_1\}$	$\mathcal{J}_1 \to \mathcal{J}_3 \to J_5$
	$\mathbf{LF}^2 = \{AV^1_2\}$	$\mathcal{J}_2 \to \mathcal{J}_4$
3	$\mathbf{LF}_1 = \{AV^1_1, AV^1_2\}$	$\{\mathcal{J}_1 \to J_3, J_2 \to \mathcal{J}_4\}$
	$\mathbf{LF}^2 = \{AV^2_1\}$	$\mathcal{J}_5$
4	$\mathbf{LF}^1 = \{AV^1_1, AV^1_2\}$	$\{\mathcal{J}_1, \mathcal{J}_2 \to \mathcal{J}_4\}$
	$\mathbf{LF}^2 = \{AV^2_1\}$	$\mathcal{J}_5$
	$\mathbf{LF}^3 = \{AV^3_1\}$	$\mathcal{J}_3$
5	$\mathbf{LF}^1 = \{AV^1_1, AV^1_2\}$	$\{\mathcal{J}_1, \mathcal{J}_2\}$
	$\mathbf{LF}^2 = \{AV^2_1, AV^2_2\}$	$\{\mathcal{J}_5, \mathcal{J}_3\}$
	$\mathbf{LF}^3 = \{AV^3_1\}$	$\mathcal{J}_4$

Table 5.2: Vehicles, platoons and scheduling

The meaning of Table 5.2 is as follows. Given a number of vehicles (first column), *a-priori* organized as platoons (second column), the **DQL-TCPN** algorithm provides the near-optimal scheduling (third column) where the arrow ($\to$) accounts for the serial execution of two jobs while the others are paralleled or distributed within the same platoon. For example, if a single vehicle is available all the jobs are sequentially performed, while, by considering two AVs, one can parallelize $\mathcal{J}_1 \to \mathcal{J}_3 \to \mathcal{J}_5$ w.r.t. $\mathcal{J}_2 \to \mathcal{J}_4$. The same reasoning hols true for all the other scenarios.

By proceeding step-by-step, the proposed **P-DRL-RHC** algorithm obtains the results collected in Table 5.3. There, it clearly appears the capability of the proposed strategy to take

advantage of two aspects: the parallel nature of the proposed approach and the distributed feature of the leader-follower arrangement. When the number of vehicles increases the cumulative time interval ϳ progressively reduces until traffic congestion occurrences come out (the last scenario). This represents a relevant property usable in real contexts to identify the minimum number of AVs compatible with the FMS structure and the assigned task.

Platoon scenario	**Jobs duration (Time units)**	ϳ
AV_1^1	$\Delta T_{\mathcal{J}_1}^{1,1} = 302$; $\Delta T_{\mathcal{J}_2}^{1,1} = 262$	
	$\Delta T_{\mathcal{J}_3}^{1,1} = 268$; $\Delta T_{\mathcal{J}_4}^{1,1} = 220$	1444
	$\Delta T_{\mathcal{J}_5}^{1,1} = 391$	
AV_1^1	$\Delta T_{\mathcal{J}_1}^{1,1} = 300$; $\Delta T_{\mathcal{J}_3}^{1,1} = 260$	
	$\Delta T_{\mathcal{J}_5}^{1,1} = 384$	944
AV_1^2	$\Delta T_{\mathcal{J}_2}^{2,1} = 257$; $\Delta T_{\mathcal{J}_4}^{2,1} = 208$	
AV_1^1	$\Delta T_{\mathcal{J}_1}^{1,1} = 300$; $\Delta T_{\mathcal{J}_3}^{1,1} = 257$	
AV_2^1	$\Delta T_{\mathcal{J}_2}^{1,2} = 258$; $\Delta T_{\mathcal{J}_4}^{1,2} = 204$	557
AV_1^2	$\Delta T_{\mathcal{J}_5}^{2,1} = 382$	
AV_1^1	$\Delta T_{\mathcal{J}_1}^{1,1} = 299$	
AV_2^1	$\Delta T_{\mathcal{J}_2}^{1,2} = 293$; $\Delta T_{\mathcal{J}_4}^{1,2} = 202$	495
AV_1^2	$\Delta T_{\mathcal{J}_5}^{2,1} = 374$	
AV_1^3	$\Delta T_{\mathcal{J}_3}^{3,1} = 277$	
AV_1^1	$\Delta T_{\mathcal{J}_1}^{1,1} = 362$	
AV_2^1	$\Delta T_{\mathcal{J}_2}^{1,2} = 304$	
AV_1^2	$\Delta T_{\mathcal{J}_5}^{2,1} = 498$	498
AV_2^2	$\Delta T_{\mathcal{J}_3}^{2,2} = 334$	
AV_1^3	$\Delta T_{\mathcal{J}_4}^{3,1} = 312$	

Table 5.3: Required time intervals according to the task scheduling

The dynamical behavior of AVs is reported in figs. 5.12-5.14 by considering the best platoon configuration (see Table 5.3): $\mathbf{LF}^1 = \{AV_1^1, AV_2^1\}$, $\mathbf{LF}_2 = \{AV_1^2\}$, $\mathbf{LF}^3 = \{AV_1^3\}$. First, notice that the prescribed constraints are always satisfied and the required jobs accomplished. Then, it is worth to underline that, whenever a vehicle reaches the machine associated to its prescribed operation, an idle phase comes out. For example take a look

to the behavior of AV_1^1, whose scheduled job is $\mathcal{J}_1$ (Table 5.3) and the first operation is $L/U \rightarrow Ma_1$ (Table 5.1): in this case, once the vehicle arrives at Ma_1 an idle phase takes place as shown in the shadow orange zones of Figs. 5.12-5.13. For the sake of completeness, figure 5.14 reports the planar trajectories of AV_2^1 according to its sequential jobs $\mathcal{J}_2$ and $\mathcal{J}_4$. Moreover, a further simulation concerns with the platoon configuration $\mathbf{LF}^1 = \{AV_1^1, AV_2^1\}$, $\mathbf{LF}^2 = \{AV_1^2, AV_2^2\}$ and $\mathbf{LF}^3 = \{AV_1^3\}$. As pointed out in Table 5.3, the cumulative time interval ⌇ increment is obtained. The reason behind is that, in this case, the platoons $\mathbf{LF}^2$ and $\mathbf{LF}^3$ move in opposite directions along the corridor CO^{12} and they detect each other at $t = 83\,[steps]$. Since $pr_2 > pr_1$, the vehicle AV_1^3 stops and, according to the developments of Section 5.6.2, AV_1^2 overcomes it.

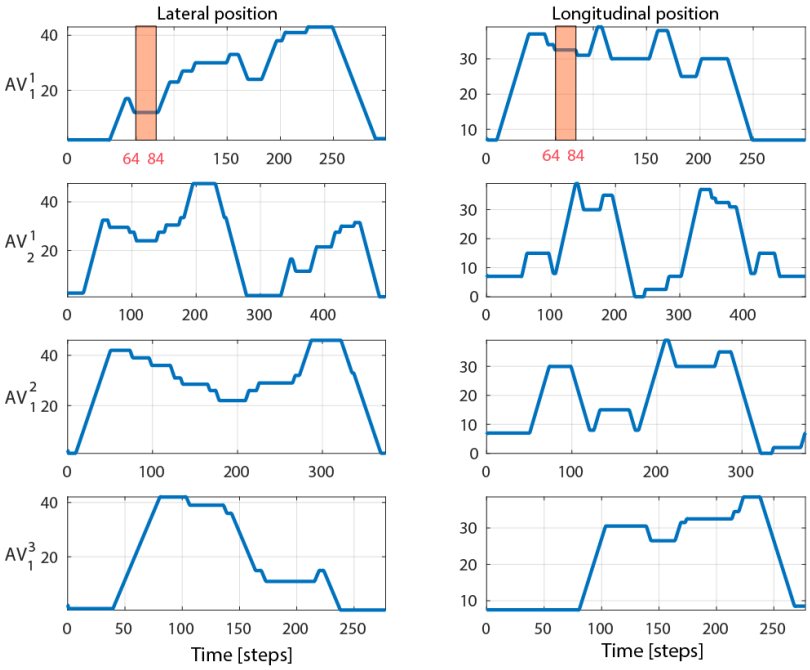

Figure 5.12: Planar components

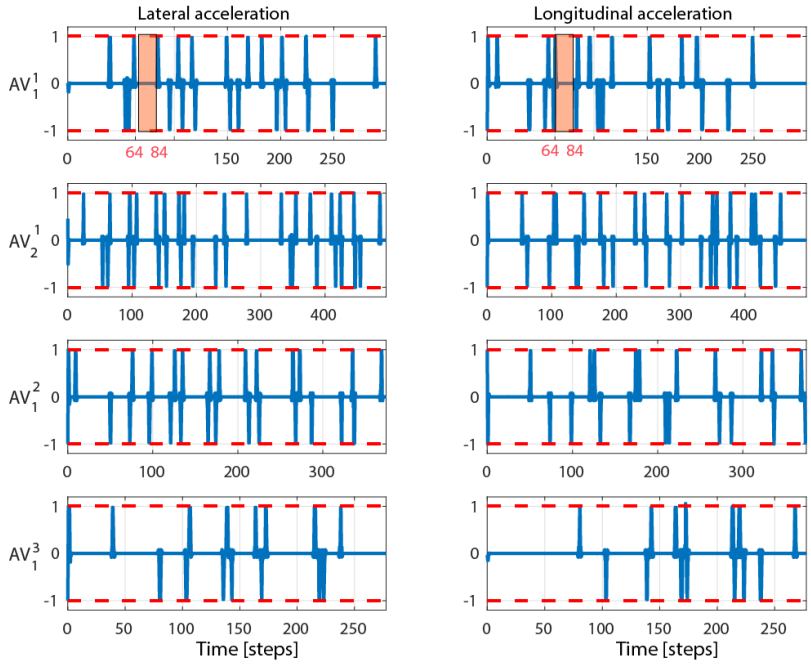

Figure 5.13: Constrained command inputs

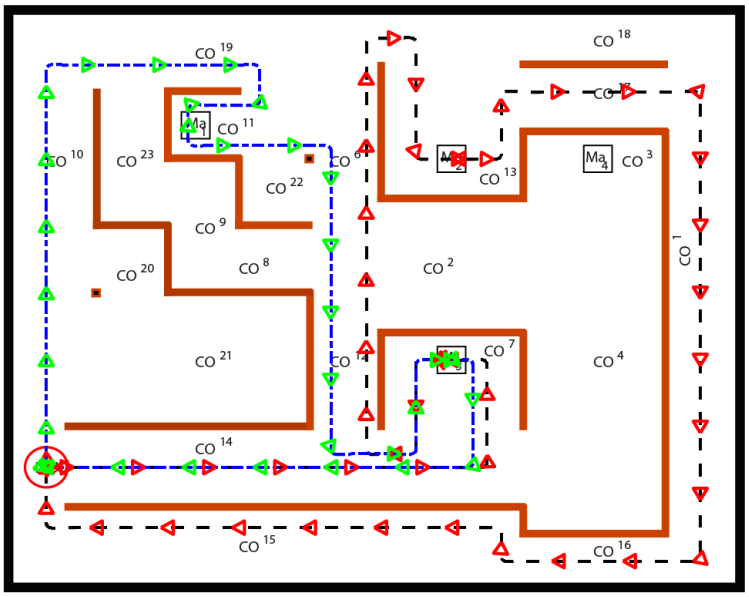

Figure 5.14: AV_2^1 trajectory: $\mathcal{J}_2$ (black dashed line) and J_4 (blue dashed-dotted line)

Finally, numerical comparisons have been carried out by contrasting the **DQL** and Dijkstra [169] schemes when used for routing decision purposes within the **P-DRL-RHC** algorithm. The results are collected in Table 5.4 in terms of the cumulative time interval]
. Notice the first platoon configuration of Table 5.3, i.e., a single vehicle, has been omitted

because it is a trivial case and gives rise to the same performance. As expected, the proposed **DRL** solution outperforms the Dijkstra-based one in all the considered scenarios in virtue of its intrinsic capability to take care of the current traffic conditions: the discrepancy between the two competitors increases as the number of vehicles grows.

Table 5.4: Cumulative time [time units] *versus* platoon configurations

Platoon configuration	Djkstra	DRL	Reduction in percent
$\mathbf{LF}^1 = \{AV_1^1\}$ $\mathbf{LF}^2 = \{AV_1^2\}$	963	944	2%
$\mathbf{LF}^1 = \{AV_1^1, AV_2^1\}$; $\mathbf{LF}^2 = \{AV_1^2\}$	579	557	3.8%
$\mathbf{LF}^1 = \{AV_1^1, AV_2^1\}$ $\mathbf{LF}^2 = \{AV_1^2\}$ $\mathbf{LF}^3 = \{AV_1^3\}$	523	495	5.3%
$\mathbf{LF}^1 = \{AV_1^1, AV_2^1\}$ $\mathbf{LF}^2 = \{AV_1^2, AV_2^2\}$ $\mathbf{LF}^3 = \{AV_1^3\}$	533	498	6.9%

Chapter 6

Autonomous vehicles platoons subject to cyber attacks

As a development of the framework presented in Chapter 4, here the focus is on resilience issues for platoons of autonomous agents operating in URNs. This work has been presented in [170].

In order to maintain the platoon formation a communication between vehicles is needed and the communication medium can be unreliable, as shown in figure 6.1. In this *Chapter*, the channel can be subject to false data injections affecting the information exchanged among the neighbors. Therefore, the problem of interest is not anymore limited in formally characterizing vehicle state trajectory tubes by means of routing decisions complying with traffic congestion criteria. Here, the distributed model predictive control scheme exploiting deep reinforcement learning capabilities for decision-making purposes is used for dealing with the regulation task, whereas, as cyber-security issues are concerned, an efficient anomaly detector and viable attack countermeasures are accordingly designed.

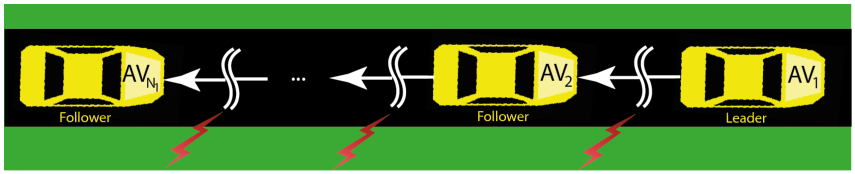

Figure 6.1: Platoon under attack

6.1 Context of interest

In the last decade, the use of open and unreliable communication platforms has enormously increased the chance that data shared among AVs could be tampered by malicious agents [18]. In this context, cybersecurity appears to be relevant in terms of attack detection capabilities and adequate resilient actions, see e.g., [19], [20] [171] and references therein.

In this *Chapter*, the distributed MPC of *Chapter* 4 has been properly customized to the present context. Specifically, the structure of the local controllers is exploited to prove the existence of a new anomaly detector in charge to reveal in finite time stealthy false data injections (FDIs) occurrences. The proposed detector exploits feasibility retention arguments to identify anomalies along the state prediction tubes. On the other hand, as the control countermeasures are concerned, two actions can be pursued without compromising the overall feasibility property and by keeping the look towards the satisfaction of the control requirement (i.e., driving the platoon to a prescribed target): use stored and feasible command inputs as soon as the attack is detected; split the platoon in sub-LF configurations. The first step leverages the fact that in attack-free scenarios, each agent stores and updates the computed MPC sequence within a dedicated buffer. Whereas the second countermeasure *enters the game* when all the feasible stored control moves have been timely used and concerns with the disconnection and re-activation in finite time of the communication link between an agent and its predecessor, see [172].

The effectiveness of the proposed approach is evaluated by means of a platoon operating in the city center of Hamburg in Germany.

6.2 Problem formulation and proposed solution

First of all, it is important to formally define the hypotheses about the communication network and later on the problem will be formulated.

Communication network

- data exchange between the autonomous vehicle AV_i and its controller MPC^i, $i = 1, \dots, N$, is not affected by any communication anomalies (induced time delays, false data injections and so on);

- at each time instant t, the agent AV_i, $i = 1, \ldots, N-1$, transmits to its follower AV_{i+1} along the LF chain the predicted state trajectory $\hat{\mathbf{x}}^i(t)$, under the action of MPC^i, and the positively invariant region $\Xi^i(t)$ complying with the current target $\bar{x}_t^i$, hereafter named as the *target set*;

- communication network may be unreliable: the information sent from AV_i to AV_{i+1} may be corrupted by malicious intruders as follows:

  - *target set:*

$$\tilde{\Xi}^i(t) \leftarrow \Xi^i(t) + \Xi_a^i(t) \tag{6.1}$$

  - *predicted state trajectory:*

$$\tilde{\mathbf{x}}^i(t) \leftarrow \hat{\mathbf{x}}^i(t) + \mathbf{x}^a(t) \tag{6.2}$$

  Hence, the problem to solve can be stated as follows:

Resilient Platoons in Urban Road Networks (RP-URN) Given a road map and a platoon of AVs (4.1) and a target x_{fin} design a distributed state-feedback control policy

$$\begin{aligned} u^1(t) &= g(x^1(t), x_{fin}^1), \\ u^i(t) &= g(x^i(t), \hat{\mathbf{x}}^{i-1}(t), x_f^i), \quad i = 2, \ldots, N, \end{aligned} \tag{6.3}$$

satisfying constraints (4.10) and such that, starting from an admissible initial condition $x(0) = x_{ini}$, the team is driven towards the target x_{fin} regardless of any junction occurrence and of any admissible occurrence of FDI attacks (6.1)-(6.2) on the communication network along the leader-follower formation. □

The proposed solution is the one introduced in chapter 4 and represented in figure 4.6, properly customized to address the security issues. In what follows, a defense strategy, capable to deal with the possible occurrence of FDI attacks on the communication channel along the LF chain, is presented.

6.3 Attack detector

According to the problem formulation and DMPC structure, the agent AV_{i-1} transmits the following information set

$$\mathcal{I}^{i-1}(t) := \{\nu(t-1)\Xi_{i-1}(t-1),\ \nu(t-1)\mathbf{z}^{i-1}(t-1),\ \nu(t-1)K_{i-1}\} \tag{6.4}$$

where $\nu = rand((0,1)) \in \mathbb{R}$ is generated by a cryptographically secure pseudo-random number generator unit νG^{i-1}. On the other hand, the agent AV_i, $i \geq 2$, receives at each t the following information set

$$\tilde{\mathcal{I}}^i(t) := \{\tilde{\Xi}_{i-1}(t-1)/\nu(t-1),\, \tilde{\mathbf{z}}^{i-1}(t-1)/\nu(t-1), \tilde{K}_{i-1}/\nu(t-1)\} \tag{6.5}$$

that should be exploited by the local controller for regulation purposes. Notice that the pseudo-random number generators νG^i and νG^{i-1} have the same seed, and are correctly synchronized with each other.

Since the communication medium is unreliable, the data packets in (6.5) have to be checked in order to verify their integrity. To this end, the architecture of figure 6.2 is conceived and its *modus operandi* hereafter discussed. At the current time instant t, the agent AV_i receives from the predecessor the packet $\tilde{\mathcal{I}}^{i-1}(t)$ to be used by the local controller $DMPC^i$ for computing the new sequence $\mathbf{u}^i(t)$. The latter is correctly done if the integrity of $\tilde{\mathcal{I}}^{i-1}(t)$ is not impaired by malicious actions on the communication channel. Here two set-membership tests are stated and the capability to formally reveal stealthy attacks proven. First, the received PI region $\tilde{\Xi}_{i-1}(t-1)$ must satisfies the following condition:

$$\tilde{\Xi}_{i-1}(t-2) \cap \tilde{\Xi}_{i-1}(t-1) \neq \emptyset \tag{6.6}$$

in order to comply with (4.32). Then, the sequence $\mathbf{u^i(t)} = \{\mathbf{u(t+j|t)}\}_{\mathbf{j=1^h}}$ is checked and, if admissible, exploited at the next time instant $t+1$. To this end, a so-called *twin* model $\mathcal{D}^i$ of the agent AV_i is defined with the aim to generate the state predictions $\hat{\mathbf{x}}^{\mathcal{D}^i}(t)$ that must fulfill the following inclusions

$$\begin{aligned}
\hat{x}_0^{\mathcal{D}^i}(t+1) \in &\quad \hat{\mathcal{X}}_i^1(t) \\
\hat{x}_1^{\mathcal{D}^i}(t+1) \in &\quad \hat{\mathcal{X}}_i^2(t) \\
&\quad \vdots \\
\hat{x}_{h_i-2}^{\mathcal{D}^i}(t+1) \in &\quad \hat{\mathcal{X}}_i^{h_i-1}(t) \\
\hat{x}_{h_i-1}^{\mathcal{D}^i}(t+1) \in &\quad \Xi^i(t) \\
\hat{x}_{h_i}^{\mathcal{D}^i}(t+1) \in &\quad \Xi^i(t)
\end{aligned} \tag{6.7}$$

where

$$\begin{aligned}
\hat{\mathcal{X}}_i^k(t) := &\quad \{x \in \mathcal{X}^i \,|x = A^i x(t) + \sum_{j=0}^{k-1} (A^i)^{k-i-j} B^i u^i(t+j|t) \\
&\quad \forall u^i(t+j|t) \in \mathcal{U}^i, j = 0, \ldots, k-1\}
\end{aligned} \tag{6.8}$$

account for the $k-step$ ahead $k = 1, \ldots, h_i - 1$ state predictions polyhedral sets compatible with the imposed constraints (4.10). Then, the following detection logic comes out

$$\mathbf{D}_i(t) := \begin{cases} 1 \leftarrow \text{attack} & \text{if (6.6)}OR\text{(6.7) fail} \\ 0 \leftarrow \text{no attack} & \text{otherwise} \end{cases} \tag{6.9}$$

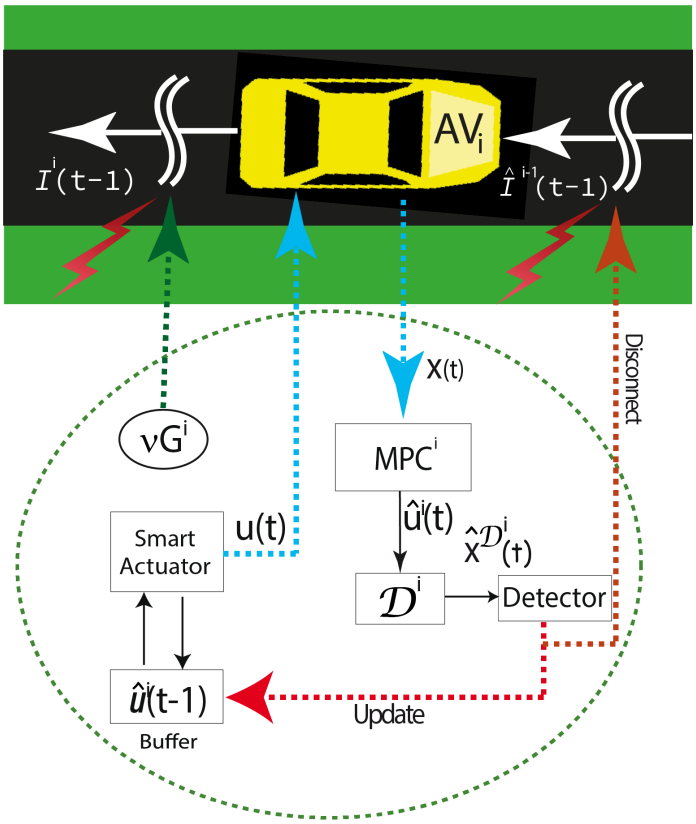

Figure 6.2: Resilient control architecture

6.4 On-line resilient actions

At each time instant t along the LF chain, the following operating scenarios can occur:

1. **Nominal mode**: attacks are not underway.

   The stored sequence $\hat{\mathbf{u}}^i(t-1)$ is updated with the currently computed sequence $\hat{\mathbf{u}}^i(t)$, while the plant is driven by $\hat{u}^i((t-1)+1|t-1)$.

2. **Viable mode**: an anomaly has been detected but a control action can be still applied without compromising constraints fulfillment and closed-loop stability.

As soon as the attack is detected, the **Smart Actuator** is instructed to apply the feasible stored moves $\hat{u}^i((t-1)+k|t-1)$, $k=1,\ldots,h_i-1$, while the newly computed sequence $\hat{\mathbf{u}}^i(t+\tau)$, $\tau \geq 1$, is discarded until the attack is underway.

3. **Attack mode**: feasible command inputs are no longer available.

   When all the stored commands $\hat{u}^i(t-1+k|t-1)$, $k=1,\ldots,h_i-1$, have been used, a possible countermeasure to keep the resilience capabilities of the overall formation consists in disconnecting the attacked agent AV_i and its successors AV_j, $j \geq i+1$, from the rest of the platoon. This means that two or more platoons (even singletons) could come out. They are controlled as presented in section 4.6.

   A further countermeasure arises from the fact that the communication link between two agents can be safely excluded and re-activated in a finite time. This translates into the following operating assumption.

   Assumption 6 *A guaranteed attack-free communication between two disconnected agents AV_i and AV_{i+1} can be reestablished in at most k_o time steps.* □

   Therefore, after at maximum k_o time steps, the two platoons can re-join the initial configuration.

6.5 Numerical results

In this subsection, the effectiveness of the proposed approach is evaluated by considering a platoon of two vehicle subject to the following constraints

$$|v_x^i(t))| \leq 6[m/sec], |v_y^i(t))| \leq 6[m/sec], i=1,2,\ \forall t \geq 0 \tag{6.10}$$

The geometrical displacement between them is $\alpha_{min} = 1[m]$ and $\alpha_{max} = 10[m]$. All the other knobs are the same as in section 4.8. The map and the traffic data are related to the district in Hamburg of figure 4.20. In this example, the threshold on CO_2 emissions is $20000000[mg/s]$.

The simulation results are collected in figs. 6.3-6.4. First, it worth to underline that the mission is accomplished despite of FDI occurrences and under the fulfillment of the prescribed constraints. In particular, at the time instant $t = 290$ *sec* an FDI attack on the

target set (6.1) takes place along the communication channel between the two vehicles. It is detected at $t = 300$ *sec* by the detection logic $\mathbf{D}_2(300)$, and, according to the countermeasure in Section III.B.3), the LF topology is disrupted and they proceed as singletons, see figure 6.3. Then at 1100 *sec* the attack ends and this is promptly recognized by the detection unit. A recovery phase takes place so that the two vehicles move to reconnect and this happens at $t = 1130\,\text{sec}$. As expected the vehicle AV_2 is significantly affected by the malicious intrusion, as it is forced to modify and consequently extend its path to mitigate the anomaly according to the new routing decisions (see figure 6.4).

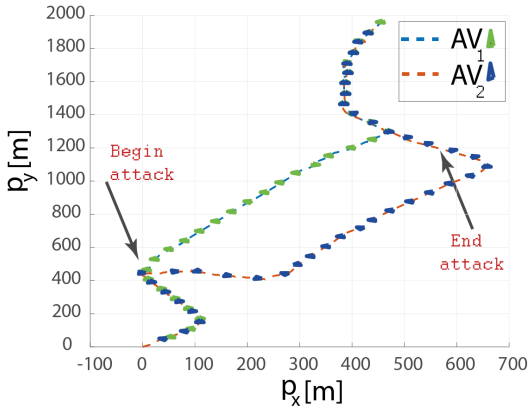

Figure 6.3: Vehicles planar trajectories

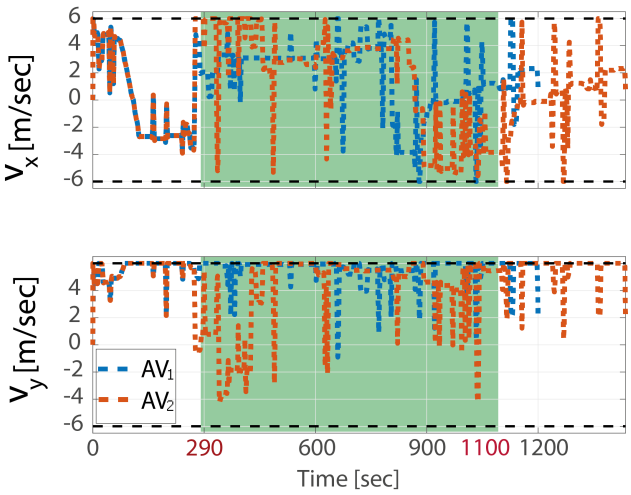

Figure 6.4: Constrained evolution

Appendix A

Neural Networks fundamentals

A neural network (NN) [11][173][174] is an interconnected assembly of simple processing elements, called neurons, whose functionality is based on the biological structure of brain. A neuron is a function $f : \mathbb{R}^n -> \mathbb{R}^m$, whose model is represented in figure A.1. The output is given by

$$f(x;\theta) = f_1(\sum_{i=1}^{n} \theta_i x_i + \theta_0) \tag{A.1}$$

where θ is a vector of parameters called weights while f_1 is a non-linear function called activation function. Such a function can have many different structures [175], some of them are introduced here:

- ReLU (Rectifying Linear Unit):

$$f_1(x) = max(0, x)$$

- leaky ReLU:

$$f_1(x) = \begin{cases} 0.01x & if\, x < 0 \\ x & otherwise \end{cases}$$

- ELU (Exponential Linear Unit):

$$f_1(x) = \begin{cases} exp(x) - 1 & if\, x < 0 \\ x & otherwise \end{cases}$$

- step:

$$f_1(x) = \begin{cases} 0 & if\, x < 0 \\ 1 & otherwise \end{cases}$$

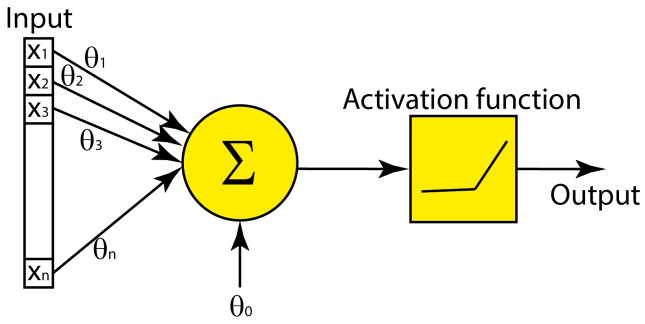

Figure A.1: Basic mathematical model for a neuron

- sigmoid:

$$f_1(x) = \frac{1}{1 + exp(-x)}$$

Neurons are connected together to form neural networks. There are two fundamentally distinct ways to do this:

- a **feed-forward network** is structured in layers and each of them is composed by neurons that receive input from the previous layer and send to the next one, without presence of loops;
- the presence of loops identifies a **recurrent network** that therefore have an inner state depending on the past inputs.

In this thesis, the used NNs are fully-connected feed-forward networks. The attribute "fully-connected" means that the input of each neuron of a layer is the same as the other neurons of that layer and it is the whole output of the previous layer (or the whole input for the first layer). Using different networks architectures and compare different designs of networks goes beyond the scope of this work.

Conclusions

This thesis has proposed new insights in the field of Model Predictive Control combined with Reinforcement Learning algorithms. Some new achievements have been reached, although there is still a lot of work to be done.

In *Chapter* 3, the problem of describing a class of nonlinear systems by means of multi-model state space descriptions has been analyzed in a data-driven context. There, the available information about an unknown nonlinear system has been carefully exploited by mitigating as much as possible the unavoidable noise arising from experimental data. Moreover, the asymptotically convergence property of the algorithm has been formally proven. The performed simulations have been devoted to the evaluation of the proposed approach. The widely recognized four-tank process and the Van der Pol oscillator have been employed as benchmark cases to evaluate the accuracy of the Polytopic Embedding algorithm. From an engineering point of view, these simulations show that the proposal can be used to derive an accurate linear model of the plant dynamics without resorting to the constitutive equations that in turn would give rise to unavoidable identification processes.

Future studies will be directed along the following lines: first of all, a formal analysis will be pursued to explicitly characterize problems of robustness, thus helping to find a trade-off between the accuracy of the obtained polytopic embedding and the resulting control performance. As a consequence, these arguments will be exploited to address the design of model predictive controllers in a nonlinear setting. Finally, it will be important to evaluate the reliability in real-life applications, such as those arising in the field of autonomous vehicles.

In *Chapter* 4, the sustainable routing decision problem for vehicle platoons operating on Urban Road Networks has been addressed. By jointly exploiting Deep Reinforcement Learning and set-theoretic Model Predictive Control arguments, a novel distributed machine

learning-control architecture has been conceived and its main properties outlined. Many simulations have been performed using real traffic data to show the main features of the proposed algorithms. These simulations testify that the proposed strategy reduces CO_2 emissions and traffic congestion, and avoids the increase in pollution of those areas subject to CO_2 emission thresholds, while satisfying input and state constraints.

Future studies will extend the proposed approach to comply with obstacle avoidance requirements and roads with more than one lane, where the lane changing problem must be addressed.

In *Chapter* 5, a multi-layer architecture has been designed to address the path planning problem, customized to comply with logistic operations in flexible manufacturing systems. Here, three methodologies have been exploited: Timed Colored Petri Nets, Deep Reinforcement Learning and Distributed Receding Horizon Control. The core of the proposed strategy relies on dividing the available group of vehicles into platoons capable of minimizing the time necessary to accomplish each single job. This has meant adding coordination and anti-collision capabilities to the distributed control structure, according to the references provided by the scheduling unit that is updated online by the Reinforcement Learning algorithm.

Future studies will be devoted to considering a more complex operating environment where, in addition to the other vehicles, the presence of human operators, understood as moving obstacles, has to be taken into account.

In *Chapter* 6, the path planning problem for vehicle platoons subject to external attacks and operating on Urban Road Networks has been addressed by integrating the control architecture described in *Chapter 3* with a detection mechanism, designed by utilizing feasibility arguments pertaining to the Receding Horizon Control philosophy that have been translated into set-containment conditions.

Future research will focus on addressing more severe attack scenarios and developing advanced detection strategies tailored to these challenges.

My work over the last three years has began to shed light on some of the most pressing problems encountered by research. Once again, further concentration on specific-case scenarios will, hopefully, lead to new successes.

Bibliography

[1] Francesco Borrelli, Alberto Bemporad, and Manfred Morari. *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.

[2] Richard S Sutton. *Reinforcement learning: An introduction*. A Bradford Book, 2018.

[3] Marshall D Rafal and William F Stevens. Discrete dynamic optimization applied to on-line optimal control. *AIChE Journal*, 14(1):85–91, 1968.

[4] AI Propoi. Application of linear programming methods for the synthesis of automatic sampled-data systems. *Avtomat. i Telemekh*, 24(7):912–920, 1963.

[5] S Joe Qin and Thomas A Badgwell. A survey of industrial model predictive control technology. *Control engineering practice*, 11(7):733–764, 2003.

[6] Max Schwenzer, Muzaffer Ay, Thomas Bergs, and Dirk Abel. Review on model predictive control: An engineering perspective. *The International Journal of Advanced Manufacturing Technology*, 117(5):1327–1349, 2021.

[7] Franco Blanchini, Stefano Miani, et al. *Set-theoretic methods in control*. Birkhäuser, second edition, 2008.

[8] David Angeli, Alessandro Casavola, Giuseppe Franzè, and Edoardo Mosca. An ellipsoidal off-line mpc scheme for uncertain polytopic discrete-time systems. *Automatica*, 44(12):3113–3119, 2008.

[9] Lea Bold, Lars Grüne, Manuel Schaller, and Karl Worthmann. Data-driven mpc with stability guarantees using extended dynamic mode decomposition. *IEEE Transactions on Automatic Control*, 2024.

[10] Armin Norouzi, Hamed Heidarifar, Hoseinali Borhan, Mahdi Shahbakhti, and Charles Robert Koch. Integrating machine learning and model predictive control for automotive applications: A review and future directions. *Engineering Applications of Artificial Intelligence*, 120(105878):1–31, 2023.

[11] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson, 4th edition, 2022.

[12] Claudio De Persis and Pietro Tesi. Formulas for data-driven control stabilization, optimality, and robustness. *IEEE Transactions on Automatic Control*, 65(3):909–924, 2019.

[13] Ivan Markovsky. Data-driven simulation of generalized bilinear systems via linear time-invariant embedding. *IEEE Transactions on Automatic Control*, 68(2):1101–1106, 2022.

[14] Jan C Willems, Paolo Rapisarda, Ivan Markovsky, and Bart LM De Moor. A note on persistency of excitation. *Systems & Control Letters*, 54(4):325–329, 2005.

[15] Kai Lin, Chensi Li, Yihui Li, Claudio Savaglio, and Giancarlo Fortino. Distributed learning for vehicle routing decision in software defined internet of vehicles. *IEEE Transactions on intelligent transportation systems*, 22(6):3730–3741, 2020.

[16] Guillaume Bono, Jilles S Dibangoye, Olivier Simonin, Laëtitia Matignon, and Florian Pereyron. Solving multi-agent routing problems using deep attention mechanisms. *IEEE Transactions on Intelligent Transportation Systems*, 22(12):7804–7813, 2020.

[17] ZhengCai Cao, ChengRan Lin, and MengChu Zhou. A knowledge-based cuckoo search algorithm to schedule a flexible job shop with sequencing flexibility. *IEEE Transactions on Automation Science and Engineering*, 18(1):56–69, 2019.

[18] Dan Zhang, Gang Feng, Yang Shi, and Dipti Srinivasan. Physical safety and cyber security analysis of multi-agent systems: A survey of recent advances. *IEEE/CAA Journal of Automatica Sinica*, 8(2):319–333, 2021.

[19] Giuseppe Franzè, Francesco Tedesco, and Domenico Famularo. Resilience against replay attacks: A distributed model predictive control scheme for networked multi-agent systems. *IEEE/CAA Journal of Automatica Sinica*, 8(3):628–640, 2020.

[20] Francesco Tedesco, Domenico Famularo, and Giuseppe Franzè. A resilient control strategy for networked multi-agent systems subject to covert attacks. *Transactions of the Institute of Measurement and Control*, 2021.

[21] D.Q. Mayne, J.B. Rawlings, C.V. Rao, and P.O.M. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789 – 814, 2000.

[22] Carlos E. García, David M. Prett, and Manfred Morari. Model predictive control: Theory and practice-a survey. *Automatica*, 25(3):335 – 348, 1989.

[23] Mayuresh V. Kothare, Venkataramanan Balakrishnan, and Manfred Morari. Robust constrained model predictive control using linear matrix inequalities. *Automatica*, 32(10):1361 – 1379, 1996.

[24] Pierre OM Scokaert and David Q Mayne. Min-max feedback model predictive control for constrained linear systems. *IEEE Transactions on Automatic control*, 43(8):1136–1142, 1998.

[25] Eugenio Alcalá, Vicenç Puig, Joseba Quevedo, and Olivier Sename. Fast zonotope-tube-based lpv-mpc for autonomous vehicles. *IET Control Theory & Applications*, 14(20):3676–3685, 2020.

[26] Alexander Kurzhanski and István Vályi. *Ellipsoidal calculus for estimation and control*. Springer, 1997.

[27] Stephen Boyd, Laurent El Ghaoui, Eric Feron, and Venkataramanan Balakrishnan. *Linear matrix inequalities in system and control theory*. SIAM, 1994.

[28] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[29] Yann Lecun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436 – 444, 2015.

[30] Chris Dann, Yishay Mansour, Mehryar Mohri, Ayush Sekhari, and Karthik Sridharan. Guarantees for epsilon-greedy reinforcement learning with function approximation. In *Proceedings of the 39th International Conference on Machine Learning, 162:4666–4689*, 2022.

[31] Jakob Hollenstein, Sayantan Auddy, Matteo Saveriano, Erwan Renaudo, and Justus Piater. Action noise in off-policy deep reinforcement learning: Impact on exploration and performance. *Transactions on Machine Learning Research*, 11:1–33, 2022.

[32] Haoran Wang, Thaleia Zariphopoulou, and Xunyu Zhou. Exploration versus exploitation in reinforcement learning: A stochastic control approach. *arXiv preprint arXiv:1812.01552*, 2018.

[33] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

[34] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *Proceedings of the 31st International Conference on Machine Learning, 32(1):387–395*. PMLR, 214.

[35] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8:279–292, 1992.

[36] Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 30(1), 2016.

[37] Harm van Seijen, Hado van Hasselt, Shimon Whiteson, and Marco Wiering. A theoretical and empirical analysis of expected sarsa. In *2009 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning: 177–184*, 2009.

[38] Satinder Singh, Tommi Jaakkola, Michael L Littman, and Csaba Szepesvári. Convergence results for single-step on-policy reinforcement-learning algorithms. *Machine learning*, 38:287–308, 2000.

[39] Ivo Grondman, Lucian Busoniu, Gabriel AD Lopes, and Robert Babuska. A survey of actor-critic reinforcement learning: Standard and natural policy gradients. *IEEE Transactions on Systems, Man, and Cybernetics, part C (applications and reviews)*, 42(6):1291–1307, 2012.

[40] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning: 1587–1596*. PMLR, 2018.

[41] Thomas M Moerland, Joost Broekens, Aske Plaat, Catholijn M Jonker, et al. Model-based reinforcement learning: A survey. *Foundations and Trends® in Machine Learning*, 16(1):1–118, 2023.

[42] Wen Sun, Nan Jiang, Akshay Krishnamurthy, Alekh Agarwal, and John Langford. Model-based rl in contextual decision processes: Pac bounds and exponential improvements over model-free approaches. In *Conference on learning theory, 99:2898–2933*. PMLR, 2019.

[43] Md Zahangir Alom, Tarek M Taha, Chris Yakopcic, Stefan Westberg, Paheding Sidike, Mst Shamima Nasrin, Mahmudul Hasan, Brian C Van Essen, Abdul AS Awwal, and Vijayan K Asari. A state-of-the-art survey on deep learning theory and architectures. *electronics*, 8(3):292, 2019.

[44] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. Learning to learn by gradient descent by gradient descent. *Advances in neural information processing systems*, 29, 2016.

[45] Giuseppe Franzè, Francesco Giannini, Vicenç Puig, and Giancarlo Fortino. Embedding the state trajectories of nonlinear systems via multimodel linear descriptions: A data-driven-based algorithm. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 54(11):7143–7155, 2024.

[46] Francesco Giannini, Giuseppe Franzè, Francesco Pupo, and Giancarlo Fortino. Set-theoretic receding horizon control for nonlinear systems: a data-driven approach. In *IEEE EUROCON 2023-20th International Conference on Smart Technologies: 579–584*. IEEE, 2023.

[47] Zhong-Sheng Hou and Zhuo Wang. From model-based control to data-driven control: Survey, classification and perspective. *Information Sciences*, 235:3–35, 2013.

[48] Shen Yin and Okyay Kaynak. Big data for modern industry: challenges and trends [point of view]. *Proceedings of the IEEE*, 103(2):143–146, 2015.

[49] Claudio De Persis and Pietro Tesi. Formulas for data-driven control: Stabilization, optimality, and robustness. *IEEE Transactions on Automatic Control*, 65(3):909–924, 2019.

[50] Zhiqiang Pu, Tianle Zhang, Xiaolin Ai, Tenghai Qiu, and Jianqiang Yi. A deep reinforcement learning approach combined with model-based paradigms for multiagent formation control with collision avoidance. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 53(7):4189–4204, 2023.

[51] Weiwei Bai, Tieshan Li, Yue Long, and CL Philip Chen. Event-triggered multigradient recursive reinforcement learning tracking control for multiagent systems. *IEEE Transactions on Neural Networks and Learning Systems*, 34(1):366–379, 2021.

[52] Weiwei Bai, Tieshan Li, Yue Long, CL Philip Chen, Yang Xiao, Wenjiang Li, and Ronghui Li. A novel adaptive control design for a class of nonstrict-feedback discrete-time systems via reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2023.

[53] Brian DO Anderson and Arvin Dehghani. Challenges of adaptive control–past, permanent and future. *Annual reviews in control*, 32(2):123–135, 2008.

[54] Frank Allgöwer, Thomas A Badgwell, Joe S Qin, James B Rawlings, and Steven J Wright. Nonlinear predictive control and moving horizon estimation—an introductory overview. *Advances in control: 391–449*, 1999.

[55] Christopher Edwards and Sarah Spurgeon. *Sliding mode control: theory and applications*. Crc Press, 1998.

[56] T. Söderström and P. Stoica. *System Identification*. Prentice Hall International, 1989.

[57] Jose Nathan Kutz. *Data-driven modeling & scientific computation: methods for complex systems & big data*. OUP Oxford, 2013.

[58] Fei Tao, He Zhang, Ang Liu, and Andrew YC Nee. Digital twin in industry: State-of-the-art. *IEEE Transactions on industrial informatics*, 15(4):2405–2415, 2018.

[59] Huai-Ning Wu and Zhou-yang Liu. Data-driven guaranteed cost control design via reinforcement learning for linear systems with parameter uncertainties. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 50(11):4151–4159, 2019.

[60] Ruey-Wen Liu. Convergent systems. *IEEE Transactions on Automatic Control*, 13(4):384–391, 1968.

[61] Alain Rapaport and Abdelmalek Maloum. Design of exponential observers for nonlinear systems by embedding. *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal*, 14(3):273–288, 2004.

[62] Arash Sadeghzadeh and Roland Tóth. Improved embedding of nonlinear systems in linear parameter-varying models with polynomial dependence. *IEEE Transactions on Control Systems Technology*, 31(1):70–82, 2022.

[63] Mark Meerschaert. *Mathematical modeling*. Academic press, 2013.

[64] Ivan Markovsky and Paolo Rapisarda. Data-driven simulation and control. *International Journal of Control*, 81(12):1946–1959, 2008.

[65] Ivan Markovsky. On the behavior of autonomous wiener systems. *Automatica*, 110:108601, 2019.

[66] Steven L Brunton, Bingni W Brunton, Joshua L Proctor, and J Nathan Kutz. Koopman invariant subspaces and finite linear representations of nonlinear dynamical systems for control. *PloS one*, 11(2):1–19, 2016.

[67] Suddhasattwa Das, Shakib Mustavee, Shaurya Agarwal, and Samiul Hasan. Koopman-theoretic modeling of quasiperiodically driven systems: Example of signalized traffic corridor. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 53(7):4466–4476, 2023.

[68] Petar Bevanda, Stefan Sosnowski, and Sandra Hirche. Koopman operator dynamical models: Learning, analysis and control. *Annual Reviews in Control*, 52:197–212, 2021.

[69] Fletcher Fan, Bowen Yi, David Rye, Guodong Shi, and Ian R Manchester. Learning stable koopman embeddings. In *2022 American Control Conference (ACC): 2742–2747*. IEEE, 2022.

[70] Yajie Bao and J Mohammadpour Velni. An overview of data-driven modeling and learning-based control design methods for nonlinear systems in lpv framework. In *Proc. 5th IFAC Workshop Linear Parameter Varying Syst.*, 2022.

[71] Yajie Bao and Javad Mohammadpour Velni. Data-driven linear parameter-varying model identification using transfer learning. *IEEE Control Systems Letters*, 5(5):1579–1584, 2020.

[72] Hassan Khalil. *Nonlinear Systems*. Pearson, 2001.

[73] Mathukumalli Vidyasagar. *Learning and generalisation: with applications to neural networks*. Springer Science & Business Media, 2013.

[74] J-P Aubin and Hélène Frankowska. Set-valued analysis, viability theory and partial differential inclusions. *World Congress on Nonlinear Analysis*, 1992.

[75] Miguel Bernal, Antonio Sala, Zsófia Lendek, and Thierry Marie Guerra. *Analysis and synthesis of nonlinear control systems*. Springer, 2022.

[76] Panos J Antsaklis and Anthony N Michel. *Linear systems*, volume 8. Springer, 1997.

[77] Giuseppe C Calafiore, Fabrizio Dabbene, and Roberto Tempo. Research on probabilistic methods for control system design. *Automatica*, 47(7):1279–1293, 2011.

[78] Henry Zhu, Justin Yu, Abhishek Gupta, Dhruv Shah, Kristian Hartikainen, Avi Singh, Vikash Kumar, and Sergey Levine. The ingredients of real-world robotic reinforcement learning. *arXiv preprint arXiv:2004.12570*, 2020.

[79] Vinay A Bavdekar, Anjali P Deshpande, and Sachin C Patwardhan. Identification of process and measurement noise covariance for state and parameter estimation using extended kalman filter. *Journal of Process control*, 21(4):585–601, 2011.

[80] Per-Åke Wedin. Perturbation theory for pseudo-inverses. *BIT Numerical Mathematics*, 13:217–232, 1973.

[81] Lingsheng Meng and Bing Zheng. The optimal perturbation bounds of the moore–penrose inverse under the frobenius norm. *Linear algebra and its applications*, 432(4):956–963, 2010.

[82] Ricardo S Sanchez-Pena and Mario Sznaier. *Robust systems theory and applications*. John Wiley & Sons, Inc., 1998.

[83] Keith Glover. All optimal hankel-norm approximations of linear multivariable systems and their l,-error bounds. *International journal of control*, 39(6):1115–1193, 1984.

[84] Weehong Tan and Andrew Packard. Stability region analysis using polynomial and composite polynomial lyapunov functions and sum-of-squares programming. *IEEE Transactions on Automatic Control*, 53(2):565–571, 2008.

[85] Giuseppe Franzè. A nonlinear sum-of-squares model predictive control approach. *IEEE Transactions on Automatic Control*, 55(6):1466–1471, 2010.

[86] AF Agarap. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018.

[87] Karl Henrik Johansson. The quadruple-tank process: A multivariable laboratory process with an adjustable zero. *IEEE Transactions on control systems technology*, 8(3):456–465, 2000.

[88] Saúl Montes de Oca, Vicenç Puig, and Joaquim Blesa. Robust fault detection based on adaptive threshold generation using interval lpv observers. *International Journal of Adaptive Control and Signal Processing*, 26(3):258–283, 2012.

[89] Francesco Giannini, Giuseppe Franzè, Francesco Pupo, and Giancarlo Fortino. A sustainable multi-agent routing algorithm for vehicle platoons in urban networks. *IEEE Transactions on Intelligent Transportation Systems*, 24(12):14830–14840, 2023.

[90] Francesco Giannini, Giuseppe Franzè, Francesco Pupo, and Giancarlo Fortino. Autonomous vehicles in smart cities: A deep reinforcement learning solution. In *2022 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*. IEEE, 2022.

[91] Dastan Bamwesigye and Petra Hlavackova. Analysis of sustainable transport for smart cities. *Sustainability*, 11(7):2140, 2019.

[92] Todd Litman and David Burwell. Issues in sustainable transportation. *International Journal of Global Environmental Issues*, 6(4):331–347, 2006.

[93] MC Bell. Environmental factors in intelligent transport systems. In *IEE Proceedings-Intelligent Transport Systems, 153(2):113–128*. IET, 2006.

[94] Tolga Ercan. Sustainability analysis of intelligent transportation systems. *Electronic Theses and Dissertations*, 2013.

[95] Ammar Haydari and Yasin Yılmaz. Deep reinforcement learning for intelligent transportation systems: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 23(1):11–32, 2020.

[96] Walid Besbes, Diala Dhouib, Niaz Wassan, and Emna Marrekchi. *Solving transport problems: towards green logistics*. Wiley Online Library, 2019.

[97] Andre Maia Pereira, Hossam Anany, Ondřej Přibyl, and Jan Přikryl. Automated vehicles in smart urban environment: A review. In *2017 Smart City Symposium Prague (SCSP)*. IEEE, 2017.

[98] Qiu Jin, Guoyuan Wu, Kanok Boriboonsomsin, and Matthew Barth. Multi-agent intersection management for connected vehicles using an optimal scheduling approach. In *2012 International Conference on Connected Vehicles and Expo (ICCVE), 185–190*. IEEE, 2012.

[99] JiuJun Cheng, JunLu Cheng, MengChu Zhou, FuQiang Liu, ShangCe Gao, and Cong Liu. Routing in internet of vehicles: A review. *IEEE Transactions on Intelligent Transportation Systems*, 16(5):2339–2352, 2015.

[100] Kai Lin, Yihui Li, Jing Deng, Pasquale Pace, and Giancarlo Fortino. Clustering-learning-based long-term predictive localization in 5g-envisioned internet of connected vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 22(8):5232–5246, 2020.

[101] Marlin W Ulmer, Justin C Goodson, Dirk C Mattfeld, and Barrett W Thomas. On modeling stochastic dynamic vehicle routing problems. *EURO Journal on Transportation and Logistics*, 9(2):100008, 2020.

[102] Paolo Toth and Daniele Vigo. *Vehicle Routing: Problems, Methods and Applications*. Society for Industrial and Applied Mathematics and the Mathematical Optimization Society, 2014.

[103] Marino Mangeruga, Alessandro Casavola, Francesco Pupo, and Fabio Bruno. An underwater pathfinding algorithm for optimised planning of survey dives. *Remote Sensing*, 12(23):3974, 2020.

[104] Kai Lin, Jian Gao, Yihui Li, Claudio Savaglio, and Giancarlo Fortino. Multi-granularity collaborative decision with cognitive networking in intelligent transportation systems. *IEEE Transactions on Intelligent Transportation Systems*, 24(1):1088–1098, 2022.

[105] Hasan Dündar, Mine Ömürgönülşen, and Mehmet Soysal. A review on sustainable urban vehicle routing. *Journal of Cleaner Production*, 285:125444, 2021.

[106] Maria João Santos, Pedro Amorim, Alexandra Marques, Ana Carvalho, and Ana Póvoa. The vehicle routing problem with backhauls towards a sustainability perspective: A review. *Top*, 28(2):358–401, 2020.

[107] V.S. Harilakshmi and P. Arockia Jansi Rani. Intelligent vehicle counter-a road to sustainable development and pollution prevention (p2). In *2016 International Conference on Energy Efficient Technologies for Sustainability (ICEETS): 877–880*, 2016.

[108] Mehran Mesbahi. Graph theoretic methods in multiagent networks. *Princeton University Press*, 2010.

[109] Kwang-Kyo Oh, Myoung-Chul Park, and Hyo-Sung Ahn. A survey of multi-agent formation control. *Automatica*, 53:424–440, 2015.

[110] Kristian Hengster-Movrić, Stjepan Bogdan, and Ivica Draganjac. Multi-agent formation control based on bell-shaped potential functions. *Journal of Intelligent and Robotic Systems*, 58:165–189, 2010.

[111] Wenjie Dong. Robust formation control of multiple wheeled mobile robots. *Journal of Intelligent & Robotic Systems*, 62:547–565, 2011.

[112] Jawhar Ghommam, Hasan Mehrjerdi, Maarouf Saad, and Faiçal Mnif. Formation path following control of unicycle-type mobile robots. *Robotics and Autonomous Systems*, 58(5):727–736, 2010.

[113] William B Dunbar and Derek S Caveney. Distributed receding horizon control of vehicle platoons: Stability and string stability. *IEEE Transactions on Automatic Control*, 57(3):620–633, 2011.

[114] Peng Wang and Baocang Ding. Distributed rhc for tracking and formation of nonholonomic multi-vehicle systems. *IEEE Transactions on Automatic Control*, 59(6):1439–1453, 2014.

[115] Giuseppe Franzè, Walter Lucia, and Francesco Tedesco. A distributed model predictive control scheme for leader-follower multi-agent systems. *International Journal of Control*, 91(2):369–382, 2018.

[116] Giuseppe Franzè and Giuseppe Fedele. A distributed model predictive control strategy for finite-time synchronization problems in multi-agent double-integrator systems. *European Journal of Control*, 55:56–67, 2020.

[117] Chen Chen, Yuru Zhang, Mohammad R Khosravi, Qingqi Pei, and Shaohua Wan. An intelligent platooning algorithm for sustainable transportation systems in smart cities. *IEEE Sensors Journal*, 21(14):15437–15447, 2020.

[118] Junjie Wang, Qichao Zhang, Dongbin Zhao, and Yaran Chen. Lane change decision-making through deep reinforcement learning with rule-based constraints. In *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019.

[119] Giuseppe Franze and Walter Lucia. A receding horizon control strategy for autonomous vehicles in dynamic environments. *IEEE Transactions on Control Systems Technology*, 24(2):695–702, 2015.

[120] Zhaoxia Peng, Guoguang Wen, Ahmed Rahmani, and Yongguang Yu. Leader–follower formation control of nonholonomic mobile robots based on a bioinspired neurodynamic based approach. *Robotics and autonomous systems*, 61(9):988–996, 2013.

[121] Alessandro Casavola, Domenico Famularo, and Giuseppe Franze. Robust fault detection of uncertain linear systems via quasi-lmis. In *Proceedings of the 2005, American Control Conference, 1654–1659*. IEEE, 2005.

[122] David Q Mayne. Control of constrained dynamic systems. *European Journal of Control*, 7(2-3):87–99, 2001.

[123] Emilia Fridman and Uri Shaked. Delay-dependent h_∞ control of uncertain discrete delay systems. *European Journal of Control*, 11(1):29–37, 2005.

[124] Emilia Fridman. *Introduction to time-delay systems*. Birkhäuser, 2014.

[125] Luigi D'Alfonso, Francesco Giannini, Giuseppe Franzè, Giuseppe Fedele, Francesco Pupo, and Giancarlo Fortino. Autonomous vehicle platoons in urban road networks: A joint distributed reinforcement learning and model predictive control approach. *IEEE/CAA Journal of Automatica Sinica*, 11(1):141–156, 2024.

[126] Pablo Alvarez Lopez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Leonhard Lücken, Johannes Rummel, Peter Wagner, and Evamarie Wießner. Microscopic traffic simulation using sumo. In *2018 21st international conference on intelligent transportation systems (ITSC): 2575–2582*. IEEE, 2018.

[127] Daniel Krajzewicz, Jakob Erdmann, Michael Behrisch, and Laura Bieker. Recent development and applications of sumo-simulation of urban mobility. *International journal on advances in systems and measurements*, 5(3), 2012.

[128] Michael Behrisch, Laura Bieker, Jakob Erdmann, and Daniel Krajzewicz. Sumo-simulation of urban mobility: an overview. In *Proceedings of SIMUL 2011, The Third International Conference on Advances in System Simulation*. ThinkMind, 2011.

[129] Andrés F Acosta, Jorge E Espinosa, and Jairo Espinosa. Traci4matlab: Enabling the integration of the sumo road traffic simulator and matlab® through a software re-engineering process. In *Modeling Mobility with Open Data: 2nd SUMO Conference, 155–170*. Springer, 2015.

[130] Geoff Henderson. Applied mathematics in integrated navigation systems–third edition rm rogers american institute of aeronautics and astronautics, 1801 alexander bell drive, suite 500, reston, va 20191-4344, usa. 2007. 408pp. illustrated. 94.95 (non-members). isbn 1-56347-927-3. *The Aeronautical Journal*, 113(1141):202–202, 2009.

[131] Laura Bieker, Daniel Krajzewicz, AntonioPio Morra, Carlo Michelacci, and Fabio Cartolano. Traffic simulation for all: a real world traffic scenario from the city of bologna. In *Modeling Mobility with Open Data: 2nd SUMO Conference, 47–60*. Springer, 2015.

[132] Domenico Famularo, Giancarlo Fortino, Francesco Pupo, Francesco Giannini, and Giuseppe Franzè. An intelligent multi-layer control architecture for logistics operations of autonomous vehicles in manufacturing systems. *IEEE Transactions on Automation Science and Engineering*, 2024.

[133] Domenico Famularo, Giuseppe Franzè, Francesco Giannini, Francesco Pupo, Giancarlo Fortino, and Francesco Tedesco. A multi-tiered control framework designed for managing the logistical activities of self-driving vehicles within manufacturing environment. In *20th International Conference on Automation Science and Engineering*. IEEE, 2024.

[134] Petri Helo and Yuqiuge Hao. Artificial intelligence in operations management and supply chain management: An exploratory case study. *Production Planning & Control*, 33(16):1573–1590, 2022.

[135] Ben Kolosz, Susan Grant-Muller, and Karim Djemame. Modelling uncertainty in the sustainability of intelligent transport systems for highways using probabilistic data fusion. *Environmental Modelling & Software*, 49:78–97, 2013.

[136] Héctor J Carlo, Iris FA Vis, and Kees Jan Roodbergen. Transport operations in container terminals: Literature overview, trends, research directions and classification scheme. *European journal of operational research*, 236(1):1–13, 2014.

[137] Andrea Ferrara, Elisa Gebennini, and Andrea Grassi. Fleet sizing of laser guided vehicles and pallet shuttles in automated warehouses. *International Journal of Production Economics*, 157:7–14, 2014.

[138] Sunderesh S Heragu. *Facilities design*. CRC Press, forth edition, 2016.

[139] MengChu Zhou, Hua-Sheng Chiu, and H Henry Xiong. Petri net scheduling of fms using branch and bound method. In *Proceedings of IECON'95-21st Annual Conference on IEEE Industrial Electronics, 1:211–216*. IEEE, 1995.

[140] ZhengCai Cao, ChengRan Lin, and MengChu Zhou. A knowledge-based cuckoo search algorithm to schedule a flexible job shop with sequencing flexibility. *IEEE Transactions on Automation Science and Engineering*, 18(1):56–69, 2019.

[141] Chengran Lin, Zhengcai Cao, and MengChu Zhou. Learning-based grey wolf optimizer for stochastic flexible job shop scheduling. *IEEE Transactions on Automation Science and Engineering*, 19(4):3659–3671, 2022.

[142] ChengRan Lin, ZhengCai Cao, and MengChu Zhou. Learning-based cuckoo search algorithm to schedule a flexible job shop with sequencing flexibility. *IEEE Transactions on Cybernetics*, 53(10):6663–6675, 2022.

[143] Juliane Fischer, Christian Lieberoth-Leden, Johannes Fottner, and Birgit Vogel-Heuser. Design, application, and evaluation of a multiagent system in the logistics domain. *IEEE Transactions on Automation Science and Engineering*, 17(3):1283–1296, 2020.

[144] Sai-Ho Chung. Applications of smart technologies in logistics and transport: A review. *Transportation Research Part E: Logistics and Transportation Review*, 153:102455, 2021.

[145] Jingjin Yu and Steven M LaValle. Optimal multirobot path planning on graphs: Complete algorithms and effective heuristics. *IEEE Transactions on Robotics*, 32(5):1163–1177, 2016.

[146] Nenad Smolic-Rocak, Stjepan Bogdan, Zdenko Kovacic, and Tamara Petrovic. Time windows based dynamic routing in multi-agv systems. *IEEE Transactions on Automation Science and Engineering*, 7(1):151–155, 2009.

[147] Ruochen Tai, Jingchuan Wang, and Weidong Chen. A prioritized planning algorithm of trajectory coordination based on time windows for multiple agvs with delay disturbance. *Assembly Automation*, 39(5):753–768, 2019.

[148] Michal Čáp, Peter Novák, Alexander Kleiner, and Martin Seleckỳ. Prioritized planning algorithms for trajectory coordination of multiple mobile robots. *IEEE transactions on automation science and engineering*, 12(3):835–849, 2015.

[149] Lijun Yue and Houming Fan. Dynamic scheduling and path planning of automated guided vehicles in automatic container terminal. *IEEE/CAA Journal of Automatica Sinica*, 9(11):2005–2019, 2022.

[150] Valerio Digani, Lorenzo Sabattini, Cristian Secchi, and Cesare Fantuzzi. Ensemble coordination approach in multi-agv systems applied to industrial warehouses. *IEEE Transactions on Automation Science and Engineering*, 12(3):922–934, 2015.

[151] Zhe Liu, Hesheng Wang, Huanshu Wei, Ming Liu, and Yun-Hui Liu. Prediction, planning, and coordination of thousand-warehousing-robot networks with motion and communication uncertainties. *IEEE Transactions on Automation Science and Engineering*, 18(4):1705–1717, 2020.

[152] Yang Shi and Kunwu Zhang. Advanced model predictive control framework for autonomous intelligent mechatronic systems: A tutorial overview and perspectives. *Annual Reviews in Control*, 52:170–196, 2021.

[153] Spyros Reveliotis. An mpc scheme for traffic coordination in open and irreversible, zone-controlled, guidepath-based transport systems. *IEEE Transactions on Automation Science and Engineering*, 17(3):1528–1542, 2020.

[154] Giuseppe Franzè and Walter Lucia. The obstacle avoidance motion planning problem for autonomous vehicles: A low-demanding receding horizon control scheme. *Systems & Control Letters*, 77:1–10, 2015.

[155] Tadao Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.

[156] Wolfgang Reisig. *Petri nets: an introduction*, volume 4. Springer Science & Business Media, 2012.

[157] Naiqi Wu. Necessary and sufficient conditions for deadlock-free operation in flexible manufacturing systems using a colored petri net model. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 29(2):192–204, 1999.

[158] Gonca Tuncel and G Mirac Bayhan. Applications of petri nets in production scheduling: a review. *The International Journal of Advanced Manufacturing Technology*, 34:762–773, 2007.

[159] James T Lin and Chia-Chu Lee. A petri net-based integrated control and scheduling scheme for flexible manufacturing cells. *Computer Integrated Manufacturing Systems*, 10(2):109–122, 1997.

[160] Kurt Jensen. *Coloured petri nets*, volume 1. Springer, 1997.

[161] Jim J Browne, Didier Dubois, Keith Rathmill, Suresh Sethi, and Kathrin Stecke. Classification of flexible manufacturing systems. *The FMS magazine*, 2(2):114–117, 1984.

[162] Olatunde T Baruwa and Miquel A Piera. A coloured petri net-based hybrid heuristic search approach to simultaneous scheduling of machines and automated guided vehicles. *International Journal of Production Research*, 54(16):4773–4792, 2016.

[163] Johann Hurink and Sigrid Knust. Tabu search algorithms for job-shop problems with a single transport robot. *European journal of operational research*, 162(1):99–111, 2005.

[164] H Yu, A Reyes, S Cang, and S Lloyd. Combined petri net modelling and ai based heuristic hybrid search for flexible manufacturing systems—part 1. petri net modelling and heuristic search. *Computers & Industrial Engineering*, 44(4):527–543, 2003.

[165] Pengling Wang, Lei Ma, Rob MP Goverde, and Qingyuan Wang. Rescheduling trains using petri nets and heuristic search. *IEEE Transactions on intelligent transportation systems*, 17(3):726–735, 2015.

[166] Liang Hu, Zhenyu Liu, Weifei Hu, Yueyang Wang, Jianrong Tan, and Fei Wu. Petri-net-based dynamic scheduling of flexible manufacturing system via deep reinforcement learning with graph convolutional network. *Journal of Manufacturing Systems*, 55:1–14, 2020.

[167] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, et al. Deep q-learning from demonstrations. In *Proceedings of the AAAI conference on artificial intelligence, 32(1):3223–3230*, 2018.

[168] David Q Mayne, María M Seron, and Saša V Raković. Robust model predictive control of constrained linear systems with bounded disturbances. *Automatica*, 41(2):219–224, 2005.

[169] E. W. Dijkstra. *A Note on Two Problems in Connexion with Graphs*. Edsger Wybe Dijkstra: His Life,Work, and Legacy, 287–290, Association for Computing Machinery, New York, NY, USA, first edition, 2022.

[170] Domenico Famularo, Francesco Giannini, Giancarlo Fortino, and Giuseppe Franzè. A distributed control architecture for sustainable rouing decisions of autonomous vehicle platoons subject to cyber attacks. In *2024 10th International Conference on Control, Decision and Information Technologies (CoDIT)*. IEEE, 2024.

[171] Florin Stoican and Sorin Olaru. *Set-theoretic fault-tolerant control in multisensor systems*. John Wiley & Sons, 2013.

[172] Giuseppe Franzè, Walter Lucia, and Francesco Tedesco. Resilient model predictive control for constrained cyber-physical systems subject to severe attacks on the communication channels. *IEEE Transactions on Automatic Control*, 67(4):1822–1836, 2021.

[173] Peter A Jansson. Neural networks: An overview. *Analytical chemistry*, 63(6):357A–362A, 1991.

[174] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[175] Andrea Apicella, Francesco Donnarumma, Francesco Isgrò, and Roberto Prevete. A survey on modern trainable activation functions. *Neural Networks*, 138:14–32, 2021.

Acknowledgments

Special thanks to my family for their constant support and backing me during this PhD journey. I specifically thank my parents, my uncles and aunts, my cousins and their children. I would like to include my appreciation of all the books that I have garnered so much from, their authors have undoubtedly helped me to develop as a sentient human being.

I want to express my gratitude to all the examiners encountered during my masters degree course in Automation engineering. The work necessary to obtain the degree has been the foundation for my growing knowledge and continued curiosity.

Many thanks to all the people who helped me to grow professionally: my supervisor, professor Giuseppe Franzé with his daily support; my co-supervisor, professor Francesco Pupo; my supervisor at Universitat Politecnica de Catalunya, professor Vicenç Puig; and all the co-authors of our publications. Among them, special thanks to professor Giancarlo Fortino. Furthermore, I would like to thank professor Sorin Olaru and professor Teodoro Alamo for reviewing my manuscript.

Finally, thanks to my friends, for all the wonderful experiences we have enjoyed together during these three years.