KTH Electrical Engineering

Cooperative Planning Control and Formation Control of Multi-Agent Systems

ALEXANDROS NIKOU

Licentiate Thesis
Stockholm, Sweden, 2017

KTH School of Electrical Engineering
Automatic Control Lab
SE-100 44 Stockholm
SWEDEN

TRITA-EE 2017:041
ISSN 1653-5146
ISBN: 978-91-7729-420-7

Akademisk avhandling som med tillstånd av Kungliga Tekniska högskolan framlägges till offentlig granskning för avläggande av teknologie licenciatexamen i reglerteknik fredagen den 2 Juni 2017 klockan 10.00 i sal Q2 Kungliga Tekniska högskolan, Osquldas väg 10, KTH, Stockholm.

Tryck: Universitetsservice US AB

Abstract

Cooperative planning control is an active topic of research, with many practical applications including multi-robot systems, transportation, multi-point surveillance and biological systems. The contributions of this thesis lie in the scope of three topics: formation control, time-constrained cooperative planning control and probabilistic control synthesis, all of them in the framework of multi-agent systems.

In the first part of the thesis, given a team of rigid-bodies, we propose decentralized control protocols such that desired position and orientation-based formation between neighboring agents is achieved. Inter-agent collisions and collisions between agents and static obstacles of the workspace are guaranteed to be avoided by the proposed control scheme. Furthermore, the connectivity between the agents that are initially connected is preserved. In the second part of the thesis, we consider a team of agents, modeled by coupled single-integrator dynamics. Each agent is assigned with individual high-level tasks, given in Metric Interval Temporal Logic (MITL). By abstracting the motion of each agent into Transition Systems (TS), we propose decentralized control methodologies that guarantee the satisfaction of the desired tasks of each agent. In the final part, a coupled multi-agent system under the presence of uncertainties and model errors is considered. Each agent is modeled by a Markov Decision Process (MDP) and is assigned with a high-level task given in Probabilistic Computational Tree Logic (PCTL). The goal is to design control policies such that each agent is performing a desired task. By clustering the agents into dependency clusters, we propose control algorithms that guarantee that the desired specifications are fulfilled. Numerical simulations conducted in MATLAB verify the claimed results.

Sammanfattning

Kooperativ planering och reglering är ett aktivt forskningsfält, med många lovande praktiska tillämpningar, t.ex; system med flera robotar som måste samarbeta, transport- och logistikproblem, distribuerad övervakning samt diverse biologiska system. Den här avhandlingen syftar att behandla följande ämnen: reglering för formationpositionering, kooperativ planering under tids-bivillkor, samt probabilistisk reglersyntes, allt under det gemensamma taket av fler-agents system.

I den första delen av avhandlingen föreslås decentraliserade reglerprotokoll, för system med agenter som modelleras enligt stelkroppsdynamik, som garanterar att önskade positions- och orientationsangivelser mellan närliggande agenter uppfylls. Reglerprotokollet garanterar att kollisioner mellan agenter, samt mellan agenter och statiska hinder i omgivningen, undviks. Vidare sa garanteras även att agenter som ursprungligen är sammankopplade förblir så. I den andra delen av avhandlingen behandlas ett system av agenter som modelleras enligt kopplad singelintegrator dynamik. Varje agent har en individuell uppgift som modelleras med Metric Interval Temporal Logic (MITL). Genom att abstrahera rörelsen hos varje agent till ett transitionssystem (TS) så kan vi föreslå decentraliserade regleringsmetoder som garanterar att varje agent uppfyller sin tilldelade uppgift. I den avslutande delen av avhandlingen behandlas ett kopplat system bestående av ett flertal agenter med osäkerhet och modellfel. Varje agent modelleras som en Markoviansk beslutsprocess (Markov Decision Process, MDP), och ges en uppgift angiven i Probabilistic Computational Tree Logic (PCTL). Vi söker reglerprotokoll som garanterar att varje agent uppfyller sitt tilldelade mål. Problemet behandlas genom att klustra agenterna med avseende på inbördes beroende. Detta låter oss i sin tur föreslå algoritmer som garanterar att alla önskade specifikationer på systemet uppfylls. Vi verifierar algoritmernas korrekthet genom numeriska simuleringar utförda i MATLAB.

Acknowledgements

I would like first of all to express my gratitude to my supervisor Prof. Dimos Dimarogonas for his invaluable support, guidance and encouragement, both in my life and my work; his continuous feedback and inspiration has made this thesis possible, and it has been an excellent experience to work with him; my co-supervisor Prof. Jana Tumova for her great insight, knowledge and assistance; my colleagues Dr. Dimitris Boskos, Christos Verginis and Shahab Heshmati-alamdari for the continuous offer of assistance and inspiration to me, in the projects that we are working together, and for being good friends.

During the past two years, Department of Automatic Control has been a joyful place to stay and work. I would like through this text to thank first of all Robert Mattila for being a good friend and collaborator and for his continuous support for everything that I needed. Among the great people I have met at the department, I would like to thank, in particular Vahan Petrosyan, Lars Lindemann, Rong Du, Linnea Persson, Valerio Turri, Jezdimir Milosevic, Mairton Da Silva Junior, Manne Henriksson, Kaveh Paridari and Xinlei Yi for not only being my colleagues but also part of my life as true friends. Sofie Andersson and Alexandros Filotheou for the great collaboration and work during their master thesis. I would also like to extend special thanks to my colleagues Pedro Pereira, Antonio Adaldo, Meng Guo and Sebastian van de Hoef.

Many thanks also to the rest of my colleagues, the professors and the administrators of the Department: Hanna, Sylvia, Anneli and Karin.

Finally, I would like to thank my family and my friends for always believing in me and for their constant moral support.

Alexandros Nikou
May 12, 2017.

Contents

Abbreviations

CTL	Computational Tree Logic
DTMC	Discrete Time Markov Chain
LTL	Linear Temporal Logic
MDP	Markov Decision Process
MILP	Mixed-Integer Linear Programming
MTL	Metric Temporal Logic
MITL	Metric Interval Temporal Logic
NMPC	Nonlinear Model Predictive Control
PCTL	Probabilistic Computational Tree Logic
ROCP	Robust Optimal Control Problem
TBA	Timed Büchi Automata
TS	Transition System
WTS	Weighted Transition System

List of Figures

Chapter 1

Introduction

1.1 Motivation

The unprecedented development of digital processing units has boosted the manufacture and installation of industrial and especially domestic robots. They become more powerful in terms of computing speed and capacity, and at the same time more affordable. They are expected to accomplish various tasks specified by non-expert end-users autonomously, without or with minimal human intervention.

Additionally, wireless communication technology enables almost all robots to be connected and equipped also with internal or external smart sensors, meaning that they can have more accurate and up-to-date information about their operation space. This type of communication should be modeled and encoded in a formal and correct way to save bandwidth and improve efficiency. All these issues bring the need for a new framework for modeling, designing and analyzing interconnected multi-agent systems.

During the last decade, decentralized control of multi-agent systems has gained a significant amount of attention due to the great variety of its applications, including multi-robot systems, transportation, multi-point surveillance and biological systems. The main focus of multi-agent systems is the design of distributed control protocols in order to achieve global tasks, such as *consensus* [1–5], in which all the agents are required to converge to a specific point and *formation* [6, 7], in which all the agents aim to form a predefined geometrical shape. At the same time, the agents might need to fulfill certain transient properties, such as *network connectivity* [8–10] and/or *collision avoidance* [11]. In parallel, another topic of research is the control of multi-robot systems such that each robot (each robot can be seen as an agent) is fulfilling desired tasks given in high-level specifications.

In particular, consider the robot in Figure 1.1 operating in a workspace which is partitioned into 6 rooms and a corridor consisting of three regions. A high-level task for the robot might have the following form: “Periodically visit rooms R_1, R_4, R_6, in this order, while avoiding rooms R_2, R_3 and R_5”, or “Grab the ball that lies in room R_6 and deliver it in room R_3 between 10 and 20 time units” or “The

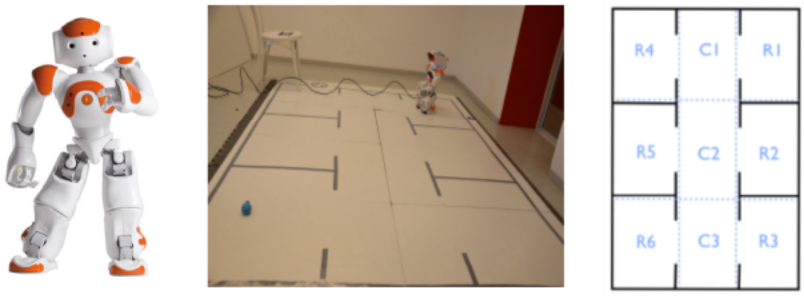

Figure 1.1: A humanoid robot moving to an environment consisting of 6 rooms and 3 corridor regions. In room $R6$ there exists a ball that the robot can grab and throw.

probability of the robot to visit rooms R_1, R_2 and R_3, in this order, is more than 0.8". The aforementioned specifications include complex tasks in which *time* and *probability* play important role. The need of imposing to robots complex tasks that have to be fulfilled, renders the control design a challenging task.

Motivated by the above, the main contribution of this thesis is to propose novel control design methodologies which solve the following three general classes of problems that arise in multi-agent systems:

1. Consider a multi-agent system modeled by Lagrangian dynamics operating in a bounded workspace with obstacles. The goal is to design controllers that use only local information for the neighboring agents such that a predefined formation between the initially connected agents, is achieved. In parallel, inter-agent collisions as well as collisions between the agents and the obstacles, should be avoided.

2. Consider a multi-agent system modeled by coupled dynamics i.e., coupling terms of neighboring agents. By assigning an individual high-level task to each agent, in which time constraints of satisfying the task occur, design decentralized control laws such that each agent fulfills the desired specification, within the desired time bounds.

3. Consider a multi-agent system under the presence of uncertainties and modeling errors. By assigning a specification task to each agent, which involves other agents as well, design control policies such that the specifications are guaranteed.

Taking the aforementioned into consideration, this thesis is divided intro three parts. Every part deals with methodologies and control algorithms for solving Problems 1, 2 and 3. The research areas that this thesis lies in is Multi-Agent Systems, Control Theory and Formal Verification, as it can be observed in Figure 1.2. Finally, the work developed in this thesis, was also inspired by the proposal of

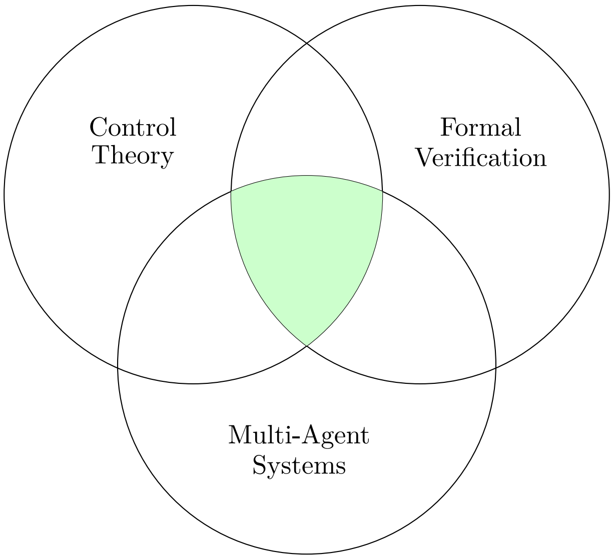

Figure 1.2: The research fields on which this thesis lies on are the following: Multi-Agent Systems, Control Theory and Formal Verification.

the European Research Project BUCOPHSYS [12]. The next section considers the outline of this thesis.

1.2 Thesis Outline and Contributions

In this Section, we provide the outline of the thesis and indicate the contributions of each chapter. Chapter 2 is devoted for notations that will be adopted in this thesis and preliminary background knowledge. The thesis is divided into *three main parts* which solves the Problems 1-3 that were previously mentioned. In particular,

- The *first part* consists of Chapter 3. In this part, we propose a novel decentralized control protocol for each agent of a multi-agent system consisting of N rigid bodies governed by Lagrangian dynamics. The proposed control scheme guarantees position and orientation based formation of the neighbors of the initial graph, inter-agent collision avoidance, agents-obstacles collision avoidance as well as connectivity maintenance with the initial connected agents.

- The *second part* consists of Chapters 4, 5. In the second part we deal with the problem of decentralized abstractions and control synthesis of multi-agent systems. By using abstraction techniques, each agent's motion is captured through a Weighted Transition System (WTS). In the sequel, we provide control synthesis tools which guarantee the satisfaction of individual formulas which are assigned to each agent. In this part, we work both in continuous and discrete level.

- The *third part* consists of Chapter 6. This part addresses the problem of probabilistic verification of multi-agent systems i.e., the design of control policies such that each agent satisfies a high-level formula given in Probabilistic Computational Tree Logic (PCTL).

Chapter 3

In this chapter, we address the problem of position- and orientation-based formation control of a class of 2nd order nonlinear multi-agent systems in a 3D workspace with obstacles. More specifically, we design decentralized control protocols such that each agent achieves a predefined geometric formation with its initial neighbors, while only using local information based on a limited sensing radius. The latter implies that the proposed scheme guarantees that the initially connected agents remain always connected. In addition, by introducing certain distance constraints, we guarantee inter-agent collision avoidance as well as collision avoidance with the obstacles and the boundary of the workspace. The proposed controllers employ a novel class of potential functions and do not require a priori knowledge of the dynamical model, except for gravity-related terms. The covered material is based on the following contributions [13–15]:

- [C7] [1] Alexandros Nikou, Christos K. Verginis and Dimos V. Dimarogonas, "Robust Distance-Based Formation Control of Multiple Rigid Bodies with Orientation Alignment", 20th World Congress of the International Federation of Automatic Control (IFAC WC), Toulouse, France, 2017.
- [C9] Alexandros Nikou, Christos K. Verginis and Dimos V. Dimarogonas, "On the Position and Orientation Based Formation Control of Multiple Rigid Bodies with Collision Avoidance and Connectivity Maintenance", 56th IEEE Conference on Decision and Control, (CDC), 2017. (Under Review)
- [J2] Christos K. Verginis, Alexandros Nikou and Dimos V. Dimarogonas, "Formation Control of Rigid Bodies with Collision Avoidance and Connectivity Maintenance", IEEE Transactions on Control of Network Systems (CONES), 2017. (Under preparation)

Chapter 4

In this chapter the problem of cooperative task planning of multi-agent systems when timed constraints are imposed to the system is investigated. We consider timed constraints given by Metric Interval Temporal Logic (MITL). We propose a method

[1]The notations C, J stands for conference and journal publications, respectively, enumerated as appeared in author's web page: https://people.kth.se/∼anikou/publications.html

for automatic control synthesis in a two-stage systematic procedure. Under the proposed method, it is guaranteed that all the agents satisfy their own individual task specifications as well as that the team satisfies a team task specification. The covered material is based on the following contributions [16, 17]:

- [C2] Alexandros Nikou, Jana Tumova, Dimos V. Dimarogonas, "Cooperative Task Planning Synthesis for Multi-Agent Systems Under Timed Temporal Specifications", American Control Conference (ACC), 2016, Boston, MA, USA.

- [C5] Sofie Andersson, Alexandros Nikou and Dimos V. Dimarogonas, "Control Synthesis for Multi-Agent Systems under Metric Interval Temporal Logic Specifications", 20th World Congress of the International Federation of Automatic Control (IFAC WC), Toulouse, France, July 2017.

Chapter 5

A fully automated procedure for controller synthesis for a general class of multi-agent systems under coupling constraints is presented in this chapter. Each agent is modeled with dynamics consisting of two terms: the first one models the coupling constraints and the other one is an additional bounded control input. We aim to design these inputs so that each agent meets an individual high-level specification given as a Metric Interval Temporal Logic (MITL). Furthermore, the connectivity of the initially connected agents, is required to be maintained. First, assuming a polyhedral partition of the workspace, a novel decentralized abstraction that provides controllers for each agent that guarantee the transition between different regions is designed. The controllers are the solution of a Robust Optimal Control Problem (ROCP) for each agent. Second, by utilizing techniques from formal verification, an algorithm that computes the individual runs which provably satisfy the high-level tasks is provided. These results presented in this chapter are based on [18–20]:

- [C3] Alexandros Nikou, Dimitris Boskos, Jana Tumova and Dimos V. Dimarogonas, "Cooperative Planning Synthesis for Coupled Multi-Agent Systems Under Timed Temporal Specifications", American Control Conference (ACC), 2017, Seattle, WA, USA.

- [J1] Alexandros Nikou, Dimitris Boskos, Jana Tumova and Dimos V. Dimarogonas, "On the Timed Temporal Logic Planning of Coupled Multi-Agent Systems", Automatica, 2017. (Under Review)

- [C10] Alexandros Nikou, Shahab Heshmati-alamdari, Christos K. Verginis and Dimos V. Dimarogonas, "Decentralized Abstractions and Timed Constrained Planning of a General Class of Coupled Multi-Agent Systems", 56th IEEE Conference on Decision and Control, (CDC), 2017. (Under Review)

Chapter 6

In this chapter we present a fully automated procedure for controller synthesis for multi-agent systems under the presence of uncertainties. We model the motion of each of the agents in the environment as a Markov Decision Process (MDP) and we assign to each agent one individual high-level formula given in Probabilistic Computational Tree Logic (PCTL). Each agent may need to collaborate with other agents in order to achieve a task. The collaboration is imposed by sharing actions between the agents. We aim to design local control policies such that each agent satisfies its individual PCTL formula. The proposed algorithm builds on clustering the agents, MDP products construction and controller policies design. We show that our approach has better computational complexity than the centralized case, which traditionally suffers from very high computational demands. These results are based on [21]:

- [C4] Alexandros Nikou, Jana Tumova and Dimos V. Dimarogonas, "Probabilistic Plan Synthesis for Coupled Multi-Agent Systems", 20th World Congress of the International Federation of Automatic Control (IFAC WC), Toulouse, France, 2017.

Finally, in Chapter 7, conclusions of this thesis as well as future research directions are discussed.

Contributions not included in this thesis

The following publications are not covered in this thesis, but contain material that motivates the work presented here [22–24]:

- [C8] Alexandros Nikou, Christos K. Verginis, Shahab Heshmati-alamdari and Dimos V. Dimarogonas, "A Nonlinear Model Predictive Control Scheme for Cooperative Manipulation with Singularity and Collision Avoidance", 25th IEEE Mediterranean Conference on Control and Automation (MED), Valletta, Malta, 2017.

- [C6] Shahab Heshmati-Alamdari, Alexandros Nikou, Kostas J. Kyriakopoulos and Dimos V. Dimarogonas, "A Robust Control Approach for Underwater Vehicle Manipulator Systems in Interaction with Compliant Environments", 20th World Congress of the International Federation of Automatic Control (IFAC WC), Toulouse, France, 2017.

- [C1] Alexandros Nikou, Georgios Gavridis and Kostas J. Kyriakopoulos, "Mechanical Design, Modelling and Control of a Novel Aerial Manipulator", IEEE International Conference on Robotics and Automation (ICRA), May 26-30, 2015, Washington State Convention Center, Seattle, Washington, USA.

Chapter 2

Notation and Preliminaries

In this chapter, the notation that will be used hereafter as well as the necessary background, are provided.

We denote by $\mathbb{R}, \mathbb{Q}_+, \mathbb{N}$ the set of real, nonnegative rational and natural numbers including 0, respectively. $\mathbb{R}^n_{\geq 0}$ and $\mathbb{R}^n_{>0}$ are the sets of real n-vectors with all elements nonnegative and positive, respectively. Define also $\mathbb{T}_\infty = \mathbb{T} \cup \{\infty\}$ for a set $\mathbb{T} \subseteq \mathbb{R}$. Given a set S, denote by $|S|$ its cardinality, by $S^n = S \times \cdots \times S$ its n-fold Cartesian product, and by 2^S the set of all its subsets; ∂S stands for the boundary of the set S. The notation $\|x\|$ is used for the Euclidean norm of a vector $x \in \mathbb{R}^n$; $\|A\| = \max\{\|Ax\| : \|x\| = 1\}$ stands for the induced norm of a matrix $A \in \mathbb{R}^{n \times n}$. The absolute value of the maximum singular value and the absolute value of the minimum eigenvalue of a matrix $A \in \mathbb{R}^{n\times n}$ are denoted by $\sigma_{\max}(A), \lambda_{\min}(A)$, respectively. Define by $\mathbb{1}_n \in \mathbb{R}^n$, $I_n \in \mathbb{R}^{n\times n}$ and $0_{m\times n} \in \mathbb{R}^{m\times n}$ the column vector with all entries 1, the unit matrix and the $m \times n$ matrix with all entries zeros, respectively. A matrix $S \in \mathbb{R}^{n\times n}$ is called skew-symmetric if and only if $S^\top = -S$. $A \otimes B$ denotes the Kronecker product of the matrices $A, B \in \mathbb{R}^{m\times n}$ (see [25]). The set-valued function $\mathcal{B} : \mathbb{R}^3 \times \mathbb{R}_{>0} \rightrightarrows \mathbb{R}^3$, given as

$$\mathcal{B}(c, \underline{r}) = \{x \in \mathbb{R}^3 : \|x - c\| \leq \underline{r}\},$$

represents the 3D sphere with center $c \in \mathbb{R}^3$ and radius $\underline{r} \in \mathbb{R}_{>0}$. Given a scalar function $y : \mathbb{R}^n \to \mathbb{R}$ and a vector $x \in \mathbb{R}^n$, denote by

$$\nabla_x y(x) = \left[\frac{\partial y(x)}{\partial x_1}, \ldots, \frac{\partial y(x)}{\partial x_n}\right]^\top \in \mathbb{R}^n,$$

the gradient of y. The derivative of a matrix $M \in \mathbb{R}^{m\times n}$, with $M = [m_{ij}]$, is defined by $\frac{\partial M}{\partial x} = [\frac{\partial m_{ij}}{\partial x}]$, with $i \in \{1, \ldots, m\}$, $j \in \{1, \ldots, n\}$.

Definition 2.1. Given the sets S_1, S_2, their *Minkowski addition* is defined by:

$$S_1 \oplus S_2 = \{s_1 + s_2 : s_1 \in S_1, s_2 \in S_2\}.$$

Definition 2.2. Consider two sets $S_1, S_2 \subseteq \mathbb{R}^n$. Then, the *Pontryagin difference* is defined by:

$$S_1 \sim S_2 = \{x \in \mathbb{R}^n : s_1 + s_2 \in S_1, \forall\ s_2 \in S_2\}.$$

Definition 2.3. Given a set S, we say that a family of sets $\{S_\ell\}_{\ell \in \mathbb{I}}$ forms a *partition* of S if $S \neq \emptyset$, $\bigcup\limits_{\ell \in \mathbb{I}} S_\ell = S$ and for every $S, S' \in S$ with $S \neq S'$ it holds $S \cap S' = \emptyset$; $\mathbb{I}$ is a set of indexes which stands for the enumeration of the members of the partition.

Definition 2.4. ([26]) A continuous function $\alpha : [0, a) \to \mathbb{R}_{\geq 0}$ is said to belong to *class* $\mathcal{K}$, if it is strictly increasing and $\alpha(0) = 0$. It is said to belong to class $\mathcal{K}_\infty$ if $a = \infty$ and $\alpha(r) \to \infty$, as $r \to \infty$.

Definition 2.5. ([26]) A continuous function $\beta : [0, a) \times \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is said to belong to *class* $\mathcal{KL}$, if:

- For each fixed s, $\beta(r, s) \in \mathcal{K}$ with respect to r.
- For each fixed r, $\beta(r, s)$ is decreasing with respect to s and $\beta(r, s) \to 0$, at $s \to \infty$.

Definition 2.6. ([27]) A nonlinear system $\dot{x} = f(x, u)$ with initial condition $x(t_0)$ is said to be *Input to State Stable (ISS)* if there exist functions $\beta \in \mathcal{KL}$ and $\sigma \in \mathcal{K}_\infty$ such that:

$$\|x(t)\| \leq \beta(\|x(t_0)\|, t) + \sigma(\|u\|).$$

Definition 2.7. ([27]) A Lyapunov function $V(x, u)$ for the nonlinear system $\dot{x} = f(x, u)$ with initial condition $x(t_0)$ is said to be *ISS-Lyapunov function* if there exist functions $\alpha, \sigma \in \mathcal{K}_\infty$ such that:

$$\dot{V}(x, u) \leq -\alpha(\|x\|) + \sigma(\|u\|), \forall x, u. \tag{2.1}$$

Theorem 2.1. *A nonlinear system $\dot{x} = f(x, u)$ with initial condition $x(t_0)$ is said to be ISS if and only if it admits a ISS-Lyapunov function.*

Proof. The proof can be found in [28]. □

Theorem 2.2. *Consider the system $\dot{x} = f(x)$ where $f : D \to \mathbb{R}^n$ is piecewise continuous and locally Lipschitz on $D \subseteq \mathbb{R}^n$; D is a domain that contains the origin. Let $V : D \to \mathbb{R}$ be a continuously differentiable function such that $\alpha_1(\|x\|) \leq V(x) \leq \alpha_2(\|x\|)$ and $\dot{V} \leq -w(x)$, $\forall \|x\| \geq \mu > 0$ for every $t \geq 0$ and $x \in \mathcal{D}$, where α_1, α_2 are class $\mathcal{K}$ functions and w_3 is a continuous positive definite function. Take $r > 0$ such that $\mathcal{B}(0, r) \subseteq D$ and suppose that $\mu < \alpha_2^{-1}(\alpha_1(r))$. Then, there exist a class $\mathcal{K}_\infty$ function α_3 and for every initial state $x(t_0)$ satisfying $\|x(t_0)\| \leq \alpha_2^{-1}(\alpha_1(r))$, there exists $T \geq 0$ such that*

$$\|x(t)\| \leq \alpha_3(\|\zeta_0\|), \forall\ t_0 \leq t \leq T,$$
$$\|x(t)\| \leq \alpha_1^{-1}(\alpha_2(\sigma)), \forall t > T.$$

Proof. The proof can be found in [26, Appendix C.9]. □

Lemma 2.1. *(Grönwall-Bellman Inequality) Let $\bar{y} : [a, b] \to \mathbb{R}$ be continuous and $\widetilde{y} : [a, b] \to \mathbb{R}$ be continuous and nonnegative. If a continuous function $y : [a, b] \to \mathbb{R}$ satisfies*

$$y(t) \leq \bar{y}(t) + \int_a^t \widetilde{y}(s) y(s) ds,$$

for $t \in [a, b]$, then on the same interval it holds that:

$$y(t) \leq \bar{y}(t) + \int_a^t \bar{y}(s)\widetilde{y}(s) \exp\left[\int_s^t \widetilde{y}(\tau) d\tau\right] ds.$$

Proof. The proof can be found in [26, Appendix A]. □

2.1 Time Sequence, Timed Run and Weighted Transition System

In the next three sections, we include definitions from computer science that are required to analyze the framework of this thesis.

An infinite sequence of elements of a set S is called an *infinite word* over this set and it is denoted by $w = w(0)w(1)\ldots$. The j-th element of a sequence is denoted by $w(j)$. For certain technical reasons that will be clarified in the sequel, we define $\mathbb{T} \triangleq \mathbb{Q}_+$. An *atomic proposition* σ is a statement that is either True ($\top$) or False ($\bot$).

Definition 2.8. ([29]) A *time sequence* $\tau = \tau(0)\tau(1)\ldots$ is an infinite sequence of time values $\tau(j) \in \mathbb{T}$, satisfying the following properties:

- Monotonicity: $\tau(j) < \tau(j + 1)$ for all $j \geq 0$.
- Progress: For every $t \in \mathbb{T}$, there exists $j \geq 1$, such that $\tau(j) > t$.

Definition 2.9. ([29]) Let Σ be a finite set of atomic propositions. A *timed word* w over the set Σ, is an infinite sequence $w^t = (w(0), \tau(0))(w(1), \tau(1))\ldots$ where $w(0)w(1)\ldots$ is an infinite word over the set 2^Σ, and $\tau(0)\tau(1)\ldots$ is a time sequence with $\tau(j) \in \mathbb{T},\ j \geq 0$.

Definition 2.10. A *Weighted Transition System* (WTS) is a tuple $(S, S_0, Act, \longrightarrow, d, \Sigma, L)$ where:

- S is a finite set of states;
- $S_0 \subseteq S$ is a set of initial states;
- Act is a set of actions;
- $\longrightarrow \subseteq S \times Act \times S$ is a transition relation;

- $d : \longrightarrow \to \mathbb{T}$ is a map that assigns a positive weight to each transition;
- Σ is a finite set of atomic propositions;
- $L : S \to 2^{\Sigma}$ is a labeling function, which maps each state with the atomic propositions that are true in this state.

The notation $s \xrightarrow{\alpha} s'$ is used to denote that $(s, \alpha, s') \in \longrightarrow$ for $s, s' \in S$ and $\alpha \in Act$. For every $s \in S$ and $\alpha \in Act$ define $\text{Post}(s, \alpha) = \{s' \in S : (s, \alpha, s') \in \longrightarrow\}$. Hereafter, $\longrightarrow$ will denote transitions and $\to$ will denote function mappings.

Definition 2.11. A *timed run* of a WTS is an infinite sequence

$$r^t = (r(0), \tau(0))(r(1), \tau(1))(r(2), \tau(2)) \ldots,$$

such that $r(0) \in S_0$, and for all $j \geq 0$, it holds that $r(j) \in S$ and $(r(j), \alpha(j), r(j+1)) \in \longrightarrow$ for a sequence of actions $\alpha(0)\alpha(1)\alpha(2)\ldots$ with $\alpha(j) \in Act, \forall\, j \geq 0$. The *time stamps* $\tau(j), j \geq 0$ are inductively defined as:

1. $\tau(0) = 0$;
2. $\tau(j+1) = \tau(j) + d((r(j), \alpha(j), r(j+1))),\ \forall\, j \geq 0$.

Definition 2.12. Every timed run r^t of a WTS generates a *timed word*

$$w(r^t) = (w(0), \tau(0))(w(1), \tau(1))(w(2), \tau(2)) \ldots,$$

over the set 2^{Σ}, where $w(j) = L(r(j)), \forall\, j \geq 0$ is the subset of atomic propositions that are true in state $r(j)$.

2.2 Metric Interval Temporal Logic (MITL)

The syntax of *Metric Interval Temporal Logic (MITL)* over the set of atomic propositions Σ is defined by the grammar:

$$\varphi := \sigma \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \bigcirc_I \varphi \mid \Diamond_I \varphi \mid \Box_I \varphi \mid \varphi_1\, \mathcal{U}_I\, \varphi_2,$$

where $\sigma \in \Sigma$ is an atomic proposition, $\neg, \wedge$ are boolean negation and conjunction operators, respectively, and $\bigcirc$, $\Diamond$, $\Box$ and $\mathcal{U}$ are the next, eventually (eventually in the future), always (now and forever in the future) and until temporal operators, respectively, as they are defined in [30, Chapter 5]; $I = [a, b] \subseteq \mathbb{T}$ where $a, b \in [0, \infty]$ with $a < b$ is a non-empty timed interval. MITL can be interpreted either in continuous or point-wise semantics [31]. In this thesis, the latter approach is utilized, since the consideration of point-wise (event-based) semantics is more suitable for the automata-based specifications considered in a discretized state-space. The MITL formulas are interpreted over timed words like the ones produced by a WTS as is given in Definition 2.12.

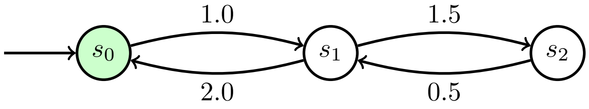

Figure 2.1: An example of a WTS with 3 states.

Definition 2.13. ([31], [32]) Given a timed word $w^t = (w(0), \tau(0))(w(1), \tau(1))\ldots$, an MITL formula φ and a position i in the timed word, the satisfaction relation $(w^t, i) \models \varphi$, for $i \geq 0$ (read w^t satisfies φ at position i) is inductively defined as follows:

$$\begin{aligned}
&(w^t, i) \models \sigma \Leftrightarrow \sigma \in w(i),\\
&(w^t, i) \models \neg\varphi \Leftrightarrow (w^t, i) \not\models \varphi,\\
&(w^t, i) \models \varphi_1 \wedge \varphi_2 \Leftrightarrow (w^t, i) \models \varphi_1 \text{ and } (w^t, i) \models \varphi_2,\\
&(w^t, i) \models \bigcirc_I\ \varphi \Leftrightarrow (w^t, i+1) \models \varphi \text{ and } \tau(i+1) - \tau(i) \in I,\\
&(w^t, i) \models \Diamond_I \varphi \Leftrightarrow \exists j \geq i, \text{ such that } (w^t, j) \models \varphi, \tau(j) - \tau(i) \in I,\\
&(w^t, i) \models \Box_I \varphi \Leftrightarrow \forall j \geq i,\ \tau(j) - \tau(i) \in I \Rightarrow (w^t, j) \models \varphi,\\
&(w^t, i) \models \varphi_1\ \mathcal{U}_I\ \varphi_2 \Leftrightarrow \exists j \geq i, \text{ s.t. } (w^t, j) \models \varphi_2,\\
&\qquad\qquad\qquad\qquad \tau(j) - \tau(i) \in I \text{ and } (w^t, k) \models \varphi_1, \forall\ i \leq k < j.
\end{aligned}$$

We say that a timed run $r^t = (r(0), \tau(0))(r(1), \tau(1))\ldots$ satisfies the MITL formula φ (we write $r^t \models \varphi$) if and only if the corresponding timed word $w(r^t) = (w(0), \tau(0))(w(1), \tau(1))\ldots$ with $w(j) = L(r(j)), \forall j \geq 0$, satisfies the MITL formula ($w(r^t) \models \varphi$).

It has been proved that MITL is decidable in infinite words and point-wise semantics, which is the case considered here (see [33, 34] for details). The model checking and satisfiability problems are EXPSPACE-complete. It should be noted that in the context of timed systems, EXSPACE complexity is fairly low [35].

Example 2.1. Consider a set of atomic propositions $\Sigma = \{\sigma_1, \sigma_2\}$ and an MITL formla $\varphi = \Box_{[0,\infty]}(\sigma_1 \Rightarrow \Diamond_{[10,20]}\sigma_2)$ over 2^Σ. The formula means that all events σ_1 must be followed by a σ_2 event that occurs between 10 and 20 time units later. Consider also the timed word:

$$w^t = (\sigma_1, 1)(\sigma_1, 2)(\sigma_1, 3)(\sigma_2, 20)\ldots,$$

It can be observed that w^t satisfies φ ($w^t_1 \models \varphi$) since all events σ_1 at the corresponding time stamps $1, 2$ and 3 are followed by an event σ_2 after time that belongs to interval $[10, 20]$.

Example 2.2. Consider the WTS with $S = \{s_0, s_1, s_2\}$, $S_0 = \{s_0\}$, $Act = \emptyset$, $\longrightarrow = \{(s_0, \emptyset, s_1), (s_1, \emptyset, s_2), (s_1, \emptyset, s_0), (s_2, \emptyset, s_1)\}$, $d((s_0, \emptyset, s_1)) = 1.0$, $d((s_1, \emptyset,$

$s_2)) = 1.5$, $d((s_1, \emptyset, s_0)) = 2.0$, $d((s_2, \emptyset, s_1)) = 0.5$, $\Sigma = \{\text{green}\}$, $L(s_0) = \{\text{green}\}$, $L(s_1) = L(s_2) = \emptyset$ depicted in Figure 2.1. Let two timed runs of the system be:

$$r_1^t = (s_0, 0.0)(s_1, 1.0)(s_0, 3.0)(s_1, 4.0)(s_0, 6.0)\ldots,$$
$$r_2^t = (s_0, 0.0)(s_1, 1.0)(s_2, 2.5)(s_1, 3.0)(s_0, 5.0)\ldots,$$

and two MITL formulas $\varphi_1 = \Diamond_{[2,5]}\{\text{green}\}, \varphi_2 = \Box_{[0,5]}\{\text{green}\}$. According to the MITL semantics of Definition 2.13, it follows that the timed run r_1^t satisfies φ_1 ($r_1^t \models \varphi_1$), since at the time stamp $3.0 \in [2, 5]$ we have that $L(s_0) = \{green\}$ so the atomic proposition green occurs at least once in the given interval. On the other hand, the timed run r_2^t does not satisfy φ_2 ($r_2^t \not\models \varphi_2$) since the atomic proposition green does not hold at every time stamp of the run r_2^t (it holds only at the time stamp 0.0).

2.3 Timed Büchi Automata

Timed Büchi Automata (TBA) were originally introduced in [29]. In this work, we partially adopt the notation from [35, 36]. Let C be a finite set of *clocks* with $\mathcal{C} \triangleq |C|$. The set of *clock constraints* $\Phi(C)$ is defined by the grammar:

$$\phi := \top \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid c \bowtie \psi,$$

where $c \in C$ is a clock, $\psi \in \mathbb{T}$ is a clock constant and $\bowtie \in \{<, >, \geq, \leq, =\}$. A clock *valuation* is a function $\nu : C \to \mathbb{T}$ that assigns a value to each clock. A clock c_i has valuation ν_i for $i \in \{1, \ldots, \mathcal{C}\}$, and $\nu = (\nu_1, \ldots, \nu_{\mathcal{C}})$. We denote by $\nu \models \phi$ the fact that the valuation ν satisfies the clock constraint ϕ.

Definition 2.14. A *Timed Büchi Automaton* is a tuple (Q, Q_0, C, Inv, E, F, Σ, $\mathcal{L}$) where

- Q is a finite set of locations;
- $Q_0 \subseteq Q$ is the set of initial locations;
- C is a finite set of clocks;
- $Inv : Q \to \Phi(C)$ is an invariant function that labels each location $s \in S$ with a subset of clock constraints in $\Phi(C)$;
- $E \subseteq Q \times \Phi(C) \times 2^C \times Q$ gives the set of edges. An edge (s, γ, R, s') represents a transition from state s to state s'; γ is a clock constraint over C that specifies when the switch is enabled (guard). This can be a constraint either in the invariant or in the edge. The set $R \subseteq C$ represents the Reset set i.e. if a $c_i \in R$ then $\nu_i = 0, i \in \{1, \ldots, \mathcal{C}\}$ and if $c_i \notin R$, ν_i remains unchanged.
- $F \subseteq Q$ is a set of accepting locations; Σ is a finite set of atomic propositions; and

- $\mathcal{L} : Q \to 2^{\Sigma}$ labels every state with a subset of atomic propositions.

A state of a TBA is a pair (q, ν) where $q \in Q$ and ν satisfies the *invariant* $Inv(q)$, i.e., $\nu \models Inv(q)$. The initial state is $(q(0), (0, \ldots, 0))$, where $q(0) \in Q_0$. Given two states (q, ν) and (q', ν') and an edge $e = (q, \gamma, R, q')$, there exists a *discrete transition* $(q, \nu) \xrightarrow{e} (q', \nu')$ iff $\nu \models \gamma$, $\nu' \models Inv(q')$, and R is the *reset set*, i.e., $\nu'_i = 0$ for $c_i \in R$ and $\nu'_i = \nu_i$ for $c_i \notin R$. Given a $\delta \in \mathbb{T}$, there exists a *time transition* $(q, \nu) \xrightarrow{\delta} (q', \nu')$ iff $q = q', \nu' = \nu + \delta$ (δ is summed component-wise) and $\nu' \models Inv(q)$. We write $(q, \nu) \xrightarrow{\delta}\xrightarrow{e} (q', \nu')$ if there exists q'', ν'' such that $(q, \nu) \xrightarrow{\delta} (q'', \nu'')$ and $(q'', \nu'') \xrightarrow{e} (q', \nu')$ with $q'' = q$.

An infinite run of a TBA starting at state $(q(0), \nu)$ is an infinite sequence of time and discrete transitions

$$(q(0), \nu(0)) \xrightarrow{\delta_0} (q(0)', \nu(0)') \xrightarrow{e_0} (q(1), \nu(1)) \xrightarrow{\delta_1} (q(1)', \nu(1)') \ldots,$$

where $(q(0), \nu(0))$ is an initial state. This run produces the timed word

$$w = (\mathcal{L}(q(0)), \tau(0))(\mathcal{L}(q(1)), \tau(1)) \ldots,$$

with $\tau(0) = 0$ and $\tau(i+1) = \tau(i) + \delta_i, \forall\, i \geq 1$. The run is called *accepting* if $q(i) \in F$ for infinitely many times. A timed word is *accepted* if there exists an accepting run that produces it.

The problem of deciding the language emptiness of a given TBA is PSPACE-complete [29]. In other words, an accepting run of a given TBA can be synthesized, if one exists. Any MITL formula φ over Σ can be algorithmically translated into a TBA with the alphabet 2^{Σ}, such that the language of timed words that satisfy φ is the language of the accepting timed words of the TBA ([33, 37, 38]).

Example 2.3. A TBA with $Q = \{q_0, q_1, q_2\}, Q^{\text{init}} = \{q_0\}, C = \{c\}, Inv(q_0) = Inv(q_1) = Inv(q_2) = \emptyset$,

$$\begin{aligned} E = \{&(q_0, \{c \leq c_2\}, \emptyset, q_0), (q_0, \{c \leq c_1 \vee c > c_2\}, c, q_2), \\ &(q_0, \{c \geq c_1 \wedge c \leq c_2\}, c, q_1), (q_1, \top, c, q_1), (q_2, \top, c, q_2)\}, \end{aligned}$$

$F = \{q_1\}, \Sigma = \{green\}, \mathcal{L}(q_0) = \mathcal{L}(q_2) = \emptyset, \mathcal{L}(q_1) = \{green\}$, which accepts all the timed words that satisfy the formula $\varphi_3 = \Diamond_{[c_1, c_2]}\{green\}$ is depicted in Figure 2.2. An example of a timed run of this TBA is

$$(q_0, 0) \xrightarrow{\delta = \alpha_1} (q_0, \alpha_1) \xrightarrow{e = (q_0, \{c \geq c_1 \wedge c \leq c_2\}, c, q_1)} (q_1, 0) \ldots,$$

with $c_1 \leq \alpha_1 \leq c_2$, which generates the timed word

$$w^t = (\mathcal{L}(q_0), 0)(\mathcal{L}(q_0), \alpha_1)(\mathcal{L}(q_1), \alpha_1) \ldots = (\emptyset, 0)(\emptyset, \alpha_1)(\{green\}, \alpha_1) \ldots,$$

which satisfies the formula φ_3. The timed run

$$(q_0, 0) \xrightarrow{\delta = \alpha_2} (q_0, \alpha_2) \xrightarrow{e = (q_0, \{c \leq c_1 \vee c > c_2\}, c, q_2)} (q_2, 0) \ldots,$$

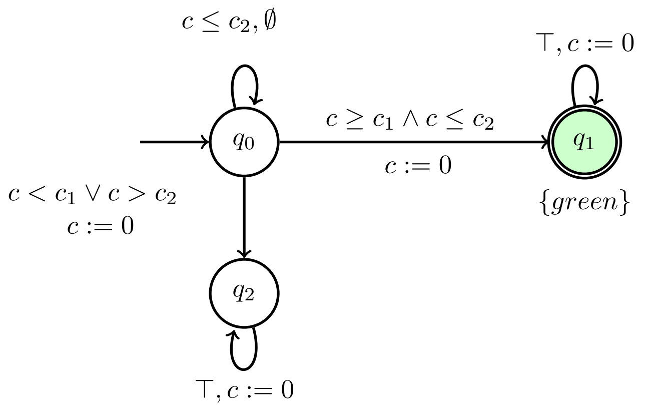

Figure 2.2: A TBA $\mathcal{A}$ that accepts the runs that satisfy the formula $\varphi = \Diamond_{[c_1,c_2]}\{green\}$.

with $\alpha_2 < c_1$, generates the timed word

$$w^t = (\mathcal{L}(q_0), 0)(\mathcal{L}(q_0), \alpha_2)(\mathcal{L}(q_2), \alpha_2) \ldots = (\emptyset, 0)(\emptyset, \alpha_2)(\emptyset, \alpha_2) \ldots,$$

which does not satisfy the formula φ_3.

Remark 2.1. Traditionally, the clock constraints and the TBAs are defined with $\mathbb{T} = \mathbb{N}$. However, they can be extended to accommodate $\mathbb{T} = \mathbb{Q}_+$, by multiplying all the rational numbers that are appearing in the state invariants and the edge constraints with their least common multiple.

2.4 Probabilistic Verification

Markov Decision Processes

Markov Decision Processes (MDPs) offer a mathematical framework for modeling systems with stochastic dynamics. These models provide an effective way for describing processes in which sequential decision making is required for a system.

Definition 2.15. A *probability distribution* over a countable set S is a function $\sigma : S \to [0, 1]$ satisfying $\sum_{s \in S} \sigma(s) = 1$. Define by $\Sigma(S)$ the set of all probability distributions over the set S.

Definition 2.16. A Discrete Time Markov Chain (DTMC) $\mathcal{D}$ is a tuple (S, s_0, P) where: S is a finite set of states; $s_0 \in S$ is the initial state; $P : S \times S \to [0, 1]$ is the transition probability matrix where for all $s \in S$ it holds that $\sum_{s' \in S} P(s, s') = 1$.

Definition 2.17. A Markov Decision Process (MDP) $\mathcal{M}$ is a tuple (S, s_0, Act, T) where:

- S is a finite set of states;
- $s_0 \in S$ is the initial state;
- Act is a finite set of actions (controls);
- $T : S \to 2^{Act \times \Sigma(S)}$ is the transition probability function.

Denote by $\mathcal{A}(s)$ the set of all actions that are available at the state $s \in S$ and let $\delta(s, \alpha, s') \in [0, 1]$ be the probability of transitioning from the state s to the state s' under the action $\alpha \in \mathcal{A}(s)$. For a state $s \in S$ and an action $\alpha \in \mathcal{A}(s)$, define the set $\text{Post}(s, \alpha) = \{s' \in S : \delta(s, \alpha, s') > 0\}$.

The transition probability function T can be represented as a matrix with $\sum_{i=0}^{|S|-1} |\mathcal{A}(s_i)|$ rows and $|S|$ columns.

An execution of an MPD is represented by a *path*. Formally, an *infinite path* r is a sequence of states of the form: $r = s_0 \xrightarrow{\alpha_0} s_1 \xrightarrow{\alpha_1} \dots \xrightarrow{\alpha_{k-1}} s_k \xrightarrow{\alpha_k} s_{k+1} \dots$, such that $s_k \in S_k, \alpha_k \in \mathcal{A}(s_k)$ and $\delta(s_k, \alpha_{k+1}, s_{k+1}) > 0, \forall k \geq 0$. A *finite path* $\rho = s_0 \xrightarrow{\alpha_0} s_1 \xrightarrow{\alpha_1} \dots \xrightarrow{\alpha_{n-1}} s_n$ is a prefix of an infinite path ending in a state. In case of the actions are not taken into consideration, the infinite and finite run can be written as $r = s_1 s_2 \dots s_n \dots$ and $\rho = s_1 s_2 \dots s_n$ respectively. Denote by $|\rho| = n$ the length of the finite path and by $r(k), \rho(k)$ the k-th element of the paths r, ρ respectively. The set of all finite and infinite paths are defined by $FPath$ and $IPath$, respectively.

A control policy at each state of an MDP and is formally defined as follows:

Definition 2.18. (Control Policy) A *control policy* $\mu : FPath \to Act$ of an MDP model $\mathcal{M}$ is a function mapping a finite path $\rho = s_0 \xrightarrow{\alpha_0} s_1 \xrightarrow{\alpha_1} \dots \xrightarrow{\alpha_{n-1}} s_n$, of $\mathcal{M}$ onto an action in $\mathcal{A}(s_n)$ and specifies for every finite path, the next action to be enabled. If a control policy depends only on the last state of the finite path ρ, then it is called a *stationary policy.*

Denote by M the set of all control policies. Under a control policy $\mu \in M$, an MDP becomes a DTMC $\mathcal{D}_\mu$ (see Definition 2.16). Let $IPath_\mu \subseteq IPath$ and $FPath_\mu \subseteq FPath$ denote the set of infinite and finite paths that can be produced under the control policy μ. For each policy $\mu \in M$, a probability measure $Prob_\mu$ over the set of all paths (under the control policy μ) $IPath_\mu$ is induced. A probability measure $Prob_\mu^{\text{fin}}$ over the set of paths $FPath_\mu$ for a finite path ρ, is defined as:

$$Prob_\mu^{\text{fin}}(\rho) = \begin{cases} 1 & \text{, if } |\rho| = 0, \\ P(s_0, s_1) P(s_1, s_2) \dots P(s_{n-1}, s_n) & \text{, otherwise,} \end{cases}$$

where $P(s_k, s_{k+1})$, $k \in \{0, \dots, n\}$ are the corresponding transition probabilities in $\mathcal{D}_\mu$.

Probabilistic Computational Tree Logic (PCTL)

Probabilistic Computational Tree Logic (PCTL) [39] is used to express properties of MDPs. PCTL formulas can be recursively defined as follows:

$$\begin{aligned} \varphi &:= \top \mid \chi \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \mathcal{P}_{\bowtie p}[\psi], && (\mathit{state\ formulas}) \\ \psi &:= \bigcirc\varphi \mid \varphi_1\, \mathcal{U}^{\leq k}\ \varphi_2, && (\mathit{path\ formulas}) \end{aligned}$$

where $\chi \in \mathit{Act}$ is an action, $\bowtie = \{<, >, \leq, \geq\}$, $p \in [0,1]$ and $k \in \mathbb{N} \cup \{\infty\}$. In the syntax above, we distinguish between state formulas φ and path formulas ψ, which are evaluated over states and paths, respectively. A property of a model will always be expressed as a state formula; path formulas only occur as the parameter of the probabilistic path operator $\mathcal{P}_{\bowtie p}[\psi]$. Intuitively, a state s satisfies $\mathcal{P}_{\bowtie p}[\psi]$ (we write $s \models \mathcal{P}_{\bowtie p}[\psi]$) if there exists a control policy μ under which the probability of all paths starting from s is in the range of the interval $\bowtie p$.

For path formulas, we allow the *next* ($\bigcirc$) operator which is true if the state formula φ is satisfied in the next state and the *until operator* ($\mathcal{U}^{\leq k}$) which is true if φ_2 is satisfied within k steps and φ_1 holds up until that point.

Definition 2.19. [39] (Semantics of PCTL) For any state $s \in S$, the satisfaction relation $\models$ is defined inductively as follows:

$$s \models \top, \text{ for every } \forall s \in S, \tag{2.2}$$

$$s \models \chi \Leftrightarrow \chi \in \mathcal{A}(s), \tag{2.3}$$

$$s \models \neg\varphi \Leftrightarrow s \not\models \varphi, \tag{2.4}$$

$$s \models \varphi_1 \wedge \varphi_2 \Leftrightarrow s \models \varphi_1 \text{ and } s \models \varphi_2, \tag{2.5}$$

$$s \models \mathcal{P}_{\bowtie p}[\psi] \Leftrightarrow \mathit{Prob}_\mu(s, \psi) \bowtie p. \tag{2.6}$$

Here $\mathit{Prob}_\mu(s, \psi)$ denotes the probability that a path starting from the state s satisfies the path formula ψ under the specific control policy μ. Moreover, for any path r we have that:

$$\begin{aligned} &r \models \bigcirc\varphi \Leftrightarrow r(1) \models \varphi, \\ &r \models \varphi_1\, \mathcal{U}^{\leq k}\ \varphi_2 \Leftrightarrow \exists\ i \leq k, r(i) \models \varphi_2 \wedge r(j) \models \varphi_1, \forall j < i. \end{aligned}$$

For the operators $\Box$ (always) and $\Diamond$ (eventually) it holds that:

$$\begin{aligned} \mathcal{P}_{\bowtie p}\left[\Diamond^{\leq k}\varphi\right] &= \mathcal{P}_{\bowtie p}\left[\top\, \mathcal{U}^{\leq k}\varphi\right], \\ \mathcal{P}_{\bowtie p}\left[\Box^{\leq k}\varphi\right] &= \mathcal{P}_{\overline{\bowtie} p}\left[\Diamond^{\leq k}\neg\varphi\right], \end{aligned}$$

where $\bowtie = \{<, >, \leq, \geq\}$ and $\overline{\bowtie} = \{>, <, \geq, \leq\}$.

Probabilistic Model Checking

There are three problems that have generally been considered in probabilistic model checking of stochastic systems:

- (Model Checking) Given an MDP $\mathcal{M}$ and a property φ, check which of the states of the MDP $\mathcal{M}$ satisfy φ.
- (Controller Synthesis): Given an MDP $\mathcal{M}$ and a property φ, find all the control policies under which the formula is satisfied.
- (Existence) Given an MDP $\mathcal{M}$ and a property φ, find, if it exists, a control policy μ such that the MDP $\mathcal{M}$ satisfies the property φ under μ.

We refer the reader to [40, 41] for more details about probabilistic model checking.

Chapter 3

Formation Control of Multiple Rigid Bodies

This chapter addresses the problem of position- and orientation-based formation control of a general class of 2nd nonlinear multi-agent systems in a 3D workspace with obstacles. More specifically, we design a decentralized control protocol such that each agent achieves a predefined geometric formation with its initial neighbors, while using local information based on a limited sensing radius. The latter implies that the proposed scheme should guarantee that the initially connected agents remain always connected. In addition, by introducing certain distance constraints, we guarantee inter-agent collision avoidance as well as collision avoidance with the obstacles and the boundary of the workspace. The proposed controllers employ a novel class of potential functions and do not require a priori knowledge of the dynamical model, except for gravity-related terms. Finally, simulation results verify the validity of the proposed framework.

3.1 Introduction

A particular multi-agent problem that has been considered in the literature is the formation control problem, where the agents represent robots that aim to form a prescribed geometrical shape, specified by a certain set of desired relative configurations. The main categories of formation control that have been studied in the related literature are ([7]) position-based control, displacement-based control, distance-based control and orientation-based control.

In position-based formation control, the agents control their positions to achieve the desired formation, as prescribed by some desired position offsets with respect to a global coordinate system. When orientation alignment is considered as a control design goal, the problem is known as orientation-based (or bearing-based) formation control. The desired formation is then defined by relative inter-agent orientations. The orientation-based control steers the agents to configurations that achieve desired relative orientation angles. In this work, we aim at designing decentralized control protocols such that both position- and orientation-based formation are achieved.

The literature on position-based formation control is rich, and is traditionally categorized in single or double integrator agent dynamics and directed or undirected communication topologies (see e.g. [6, 42–61]). Orientation-based formation control has been addressed in [62–65], whereas the authors in [65–67] have considered the combination of position- and orientation-based formation

The dominant case in the related literature of formation control is the two-dimensional one with simple dynamics and point-mass agents. In real applications, however, the engineering systems may have nonlinear 2nd order dynamics, for which due to imperfect modeling the exact model is not a priori known. Other objectives concern connectivity maintenance, collision avoidance between the agents as well as collision avoidance between the agents and potential obstacles of the workspace, which renders the formation control problem a particularly challenging task. According to the authors' best knowledge, the combination of the aforementioned specifications has not been addressed in the related literature.

Motivated by this, we aim to address here the position-based formation control problem with orientation alignment for a team of rigid bodies operating in 3D space, with 2nd order nonlinear dynamics. We propose a decentralized control protocol that guarantees a geometric prescribed position- and orientation-based formation between initially connected agents. The proposed methodology guarantees inter-agent collision avoidance and collision avoidance with the obstacles and the boundary of the workspace. In parallel, connectivity maintenance of the initially connected agents as well as representation singularity avoidance are ensured. In order to deal with the aforementioned specifications, we employ a novel class of potential functions. A special case of correct-by-construction potential functions, namely *navigation functions*, has been introduced in [68, 69] for the single-robot navigation, and has been employed in multi-agent formation control in [70–76]. A more general potential function framework has been employed in [46, 77]. The aforementioned works, however, have only addressed the single integrator case, with no straightforward extension to higher-order systems. The authors in [78] deal with the double integrator case, but the goal was only navigation of the agents to specific points. In addition, in many works that employ navigation functions for navigation/formation control, the designed gains and parameters (usually referred as k-the navigation function gain- and ε-the arbitrarily small distance to the obstacles) cannot be found trivially, since, they appear in both sides of the derived inequalities.

In our previous work [13], we treated a similar problem by utilizing a Prescribed Performance Control (PPC) scheme instead (for PPC controller design we refer to [79]), while only guaranteeing collision avoidance between neighboring agents forming a tree, with no obstacles or representation singularity avoidance. The main contribution of this chapter is a novel decentralized control protocol scheme that generalizes [13] and solves a wider class of problems of multiple rigid bodies under Lagrangian dynamics with guaranteed collision avoidance among the agents, collision avoidance between agents and obstacles as well as singularity avoidance by utilizing a novel class of decentralized potential functions.

The remainder of this chapter is structured as follows. Section 3.2 gives the necessary notation of this chapter. Section 3.3 provides the system dynamics and the formal problem statement. Section 3.4 discusses the technical details of the solution and Section 3.5 is devoted to a simulation example. Finally, conclusions of this chapter are discussed in Section 3.6.

Remark 3.1. This chapter is based on the author's ongoing research. Thus, the reported results are not definitive and will be updated accordingly in the future.

3.2 Notation

The vector connecting the origins of coordinate frames $\{A\}$ and $\{B\}$ expressed in frame $\{C\}$ coordinates in 3D space is denoted by $p^C_{B/A} \in \mathbb{R}^3$. We further denote by $q_{B/A} = [\phi_{B/A}, \theta_{B/A}, \psi_{B/A}]^\tau$ the Euler angles representing the orientation of frame $\{B\}$ with respect to frame $\{A\}$, with $-\pi \leq \phi_{A/B}, \psi_{A/B} \leq \pi$ and $\frac{\pi}{2} \leq \theta_{A/B} \leq \frac{\pi}{2}$, where $\mathbb{T} = (-\pi, \pi) \times (\frac{\pi}{2}, \frac{\pi}{2}) \times (\pi, \pi)$ is the 3D torus. The angular velocity of frame $\{B\}$ with respect to $\{A\}$, expressed in frame $\{C\}$ coordinates, is denoted by $\omega^C_{B/A} \in \mathbb{R}^3$. We also use the notation $\mathbb{M} = \mathbb{R}^3 \times \mathbb{T}$. For notational brevity, when a coordinate frame corresponds to an inertial frame of reference $\{0\}$, we will omit its explicit notation (e.g., $p_B = p^0_{B/0}, \omega_B = \omega^0_{B/0}$). All vector and matrix differentiations are derived with respect to the inertial frame $\{0\}$, unless otherwise stated.

3.3 Problem Formulation

3.3.1 System Model

Consider a set of N rigid bodies, with $\mathcal{V} = \{1, 2, \ldots, N\}$, $N \geq 2$, operating in $\mathbb{M}^N$, with coordinate frames $\{i\}, i \in \mathcal{V}$, attached to their centers of mass. Consider a workspace in $\mathbb{R}^3$ modeled by a bounded sphere $W = \mathcal{B}(p_w, \underline{r}_w)$ with center p_w and radius $\underline{r}_w$. Without loss of generality, we assume that $p_w = 0_{3\times 1}$, representing an inertial reference frame $\{0\}$. The subscript w stands for the workspace W. We consider that each agent occupies a sphere $\mathcal{B}(p_i, \underline{r}_i)$, where $p_i \in \mathbb{R}^3$ is the position of the agent's center of mass and $\underline{r}_i < \underline{r}_w$ is the agent's radius. We also denote by $q_i \in \mathbb{T}$, the Euler angles representing the agents' orientation with respect to $\{0\}$, with $q_i = [\phi_i, \theta_i, \psi_i]^\top$. By defining $x_i \in \mathbb{M}, v_i \in \mathbb{R}^6$, with $x_i = [x_{i_1}, \ldots, x_{i_6}]^\top = [p_i^\top, q_i^\top]^\top, v_i = [\dot{p}_i^\top, \omega_i^\top]^\top$, we model each agent's motion with the 2nd order dynamics:

$$\dot{x}_i = J(q_i)v_i, \tag{3.1a}$$

$$u_i = M_i(x_i)\dot{v}_i + C_i(x_i, \dot{x}_i)v_i + g_i(x_i), \tag{3.1b}$$

where $J : \mathbb{T} \to \mathbb{R}^{6\times 6}$ is a *Jacobian matrix* that maps the Euler angle rates to v_i, given by

$$J(q_i) = \begin{bmatrix} I_3 & 0_{3\times 3} \\ 0_{3\times 3} & J_{q_i}(q_i) \end{bmatrix},$$

$$J_q(q_i) = \begin{bmatrix} 1 & \sin(\phi_i)\tan(\theta_i) & \cos(\phi_i)\tan(\theta_i) \\ 0 & \cos(\phi_i) & -\sin(\phi_i) \\ 0 & \dfrac{\sin(\phi_i)}{\cos(\theta_i)} & \dfrac{\cos(\phi_i)}{\cos(\theta_i)} \end{bmatrix},$$

The matrix J is not defined when $\cos(\theta_i) = 0 \Leftrightarrow \theta_i = \pm\frac{\pi}{2}$, which we refer to as *representation singularity.* The proposed controller will guarantee, however, that this is always avoided and thus J is well-defined.

Furthermore, $M_i : \mathbb{M} \to \mathbb{R}^{6\times 6}$ is the positive definite *inertia matrix*, $C_i : \mathbb{M} \times \mathbb{R}^6 \to \mathbb{R}^{6\times 6}$ is the *Coriolis matrix*, and $g_i : \mathbb{M} \to \mathbb{R}^6$ is the *gravity vector*. We consider that the Coriolis and the inertia vector fields are *unknown* to the controller. Finally, $u_i \in \mathbb{R}^6$ is the control input representing the 6D generalized *actuation force* acting on agent $i \in \mathcal{V}$. Let us also define the stack vectors $x = [x_1^\top, \dots, x_N^\top]^\top \in \mathbb{M}^N$ and $v = [v_1^\top, \dots, v_N^\top]^\top \in \mathbb{R}^{6N}$.

Remark 3.2. According to [80], the following hold:

1. The matrices $M_i(x)$ are positive definite $\forall i \in \mathcal{V}$ and there exist positive and finite constants $\underline{m}_i, \overline{m}_i$ such that for all $i \in \mathcal{V}$:

$$\underline{m}_i \|y\|^2 \le y^\top M_i(x) y \le \overline{m}_i \|y\|^2, \forall x \in \mathbb{M}, y \in \mathbb{R}^6. \tag{3.2}$$

2. The matrices $M_i^d(x_i, \dot{x}_i) - 2C_i(x_i, \dot{x}_i)$ are skew-symmetric for every $(x_i, \dot{x}_i) \in \mathbb{M} \times \mathbb{R}^6$, $i \in \mathcal{V}$, with

$$M_i^d(x_i, \dot{x}_i) = \left[\frac{\partial M_i(x_{i_1})}{\partial x_{i_1}}, \dots, \frac{\partial M_i(x_{i_6})}{\partial x_{i_6}}\right] (\dot{x}_i \otimes \mathbb{1}_6).$$

From [81], we have that a quadratic form of a skew-symmetric matrix is always equal to 0. Hence, for the matrices $M_i^d - 2C_i$, the following holds:

$$y^\top \left[M_i^d(x,z) - 2C_i(x,z)\right] y = 0, \forall x, y, z \in \mathbb{R}^6, i \in \mathcal{V}. \tag{3.3}$$

We consider that in the given workspace there exist $Z \in \mathbb{N}$ *static obstacles*, with $\mathcal{Z} \triangleq \{1, \dots, Z\}$, modeled by the spheres $\mathcal{B}(p_{o_z}, \underline{r}_{o_z})$, with centers and radii $p_{o_z} \in \mathbb{R}^3, \underline{r}_{o_z} \in \mathbb{R}_{>0}, z \in \mathcal{Z}$, respectively. We also define the neighboring set as the set-valued function $\mathcal{N}_i : \mathbb{M}^N \rightrightarrows \mathbb{N}$, with $\mathcal{N}_i(x) = \{j \in \mathcal{V}\backslash\{i\} : \|p_i - p_j\| \le d_i\}$. Define also $N_i(x) \triangleq |\mathcal{N}_i(x)|$. For the state measurement of each agent, the following assumption is required.

Assumption 3.1. (Measurements Assumption) Each agent i can measure its own states x_i, v_i, $i \in \mathcal{V}$ and has a limited sensing range of

$$d_i > \max_{i,j \in \mathcal{V}, i \ne j, z \in \mathcal{Z}} \{\underline{r}_i + \underline{r}_j, \underline{r}_i + \underline{r}_{o_z}\}.$$

Moreover, at a configuration x, each agent $i \in \mathcal{V}$ knows $N_i(x)$, x_j, v_j, $\forall j \in \mathcal{N}_i(x)$ as well as $N_j(x), x_{j^*}, x_{jj^*,\text{des}}, \forall j^* \in \mathcal{N}_j(x)$. The terms $x_{ij,\text{des}}$ are the desired relative displacement that prescribe the desired formation configuration.

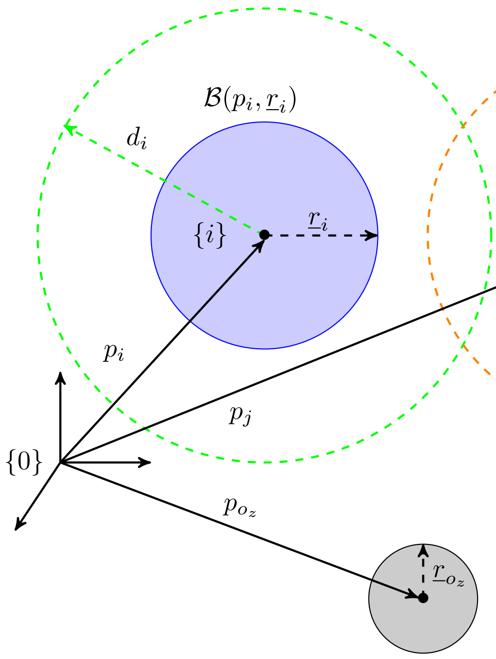

Figure 3.1: Illustration of two moving agents $i, j \in \mathcal{V}$ and a static obstacle o_z in the workspace; $\{0\}$ is the inertial frame, $\{i\}, \{j\}$ are the frames attached to the agents' center of mass, $p_i, p_j, p_{o_z} \in \mathbb{R}^3$ are the positions of the center of mass of the agents i, j and the obstacle o_z, respectively, with respect to $\{0\}$; $\underline{r}_i, \underline{r}_i, \underline{r}_{o_z}$ are the radii of the agents i, j and the obstacle o_z, respectively; d_i, d_j with $d_i > d_j$ are the agents' sensing ranges. Note that the agents are not neighbors since $p_j(t) \notin \mathcal{B}(p_i, d_i)$ and $p_i \notin \mathcal{B}(p_j, d_j)$.

Assumption 3.1 states that each agent $i \in \mathcal{V}$ has feedback of the states of the neighbors, as well as the poses of the neighbors' neighbors and their desired relative displacements, which can be continuously transmitted to agent $i \in \mathcal{N}$ by its neighbors $j \in \mathcal{N}_i(x)$. According to d_i from Assumption 3.1, an agent j can be in the neighboring set i without the agents colliding. The geometry of two agents i, j and an obstacle z in the workspace W is depicted in Figure 3.1.
Let us also define the distances $d_{ij,a} : \mathbb{R}^6 \to \mathbb{R}_{\geq 0}$, $d_{iz,o} : \mathbb{R}^3 \to \mathbb{R}_{\geq 0}$, with:

$$\begin{aligned} d_{ij,a}(p_i, p_j) &= \|p_i - p_j\|, \\ d_{iz,o}(p_i) &= \|p_i - p_{o_z}\|, \end{aligned}$$

as well as the constants

$$\begin{aligned} \underline{d}_{ij,a} &= \underline{r}_i + \underline{r}_j, \\ \underline{d}_{iz,o} &= \underline{r}_i + \underline{r}_{o_z}, \end{aligned}$$

$\forall i, j \in \mathcal{V}, i \neq j, z \in \mathcal{Z}$. The constants $\underline{d}_{ij,a}$, $\underline{d}_{ij,o}$ represent the minimum distance such that agents i and j and agent i and object z, do not collide, respectively. The subscripts a and o stand for *agent* and *obstacle*, respectively. The following assumption is required, for the feasibility of the problem:

Assumption 3.2. (Problem Feasibility) It is assumed that the following hold:

1. $\|p_{o_z} - p_{o_{z'}}\| \geq 2 \max\limits_{i \in \mathcal{V}}\{\underline{r}_i\} + \underline{r}_{o_z} + \underline{r}_{o_{z'}} + \varepsilon_r, \forall z, z' \in \mathcal{Z}$, with $z \neq z'$,
2. $\underline{r}_w - (\|p_{o_z}\| + \underline{r}_{o_z}) \geq 2 \max\limits_{i \in \mathcal{V}}\{\underline{r}_i\} + \varepsilon_r, \forall z \in \mathcal{Z}$,

where ε_r is an arbitrarily small positive constant.

The aforementioned assumption states that there is enough space between the obstacles and the workspace boundary as well as the obstacles themselves for the agents to navigate among them.

3.3.2 Problem Statement

Due to the fact that the agents are not dimensionless and their sensing capabilities are limited, the control protocol, except from achieving desired position formation (define it by $p_{ij,\text{des}} = -p_{ji,\text{des}}$) and desired formation angles (define it by $q_{ij,\text{des}} = -q_{ji,\text{des}}$) for all initially neighboring agents, it should also guarantee for all $t \in \mathbb{R}_{\geq 0}$ that (i) the agents avoid collision with each other; (ii) all agents avoid collision with obstacles; (iii) all agents avoid collision with the workspace boundary, (iv) all the initial edges are maintained, and (v) the singularity of the Jacobian matrices J is avoided.

Definition 3.1. (Feasible Formation) Given the initial neighboring sets $\mathcal{N}_i(x_0), i \in \mathcal{V}$, defined by the initial poses $x_0 = x(0)$, the *desired displacements* $x_{ij,\text{des}} = [p_{ij,\text{des}}^\top, q_{ij,\text{des}}^\top]^\top = -x_{ji,\text{des}}$, with $j \in \mathcal{N}_i(x_0)$, that characterize a formation configuration, are called *feasible* if the following holds:

$$\bigcap_{i \in \mathcal{V}} \big\{x_i \in \mathbb{M} : \|x_i - x_j - x_{ij,\text{des}}\| = 0, \\ d_{iz,o}(p_i) > 0, \|p_i\| + \underline{r}_i < \underline{r}_w, \forall z \in \mathcal{Z}, j \in \mathcal{N}_i(x_0)\big\} \neq \emptyset.$$

Formally, the control problem under the aforementioned constraints is formulated as follows:

Problem 3.1. Consider N agents governed by the dynamics (3.1), operating in $\mathbb{M}$, with Z static obstacles, and initial properties:

- $v_{i0} = 0_{6\times 1}, \forall i \in \mathcal{V}$;
- $-\frac{\pi}{2} < -\bar{\theta} \leq \theta_{i0} \leq \bar{\theta} < \frac{\pi}{2}, \forall i \in \mathcal{V}$;

- $\|p_{i0}\| + \underline{r}_i < \underline{r}_w, \forall i \in \mathcal{V}$;
- $\|p_{i0} - p_{j0}\| > \underline{d}_{ij,a}, \forall i, j \in \mathcal{V}, i \neq j$;
- $\|p_{i0} - p_{o_z}\| > \underline{d}_{iz,o}, \forall i \in \mathcal{V}, z \in \mathcal{Z}$;

The subscript 0 denotes the initial value at $t = 0$. The aforementioned properties concern singularity- and collision- free configurations at $t = 0$, where $\bar{\theta}$ is an arbitrary constant in the open set $(0, \frac{\pi}{2})$. Then, given non-empty initial neighboring sets $\mathcal{N}_i(x_0) \neq \emptyset$, *feasible* inter-agent displacements $p_{ij,\text{des}}$, $q_{ij,\text{des}}$, $\forall i \in \mathcal{V}$, $j \in \mathcal{N}_i(x_0)$, such that

$$\underline{d}_{ij,a} < d_{ij,a}(p_i, p_j) < d_i, \forall (p_i, p_j) \in \{(p_i, p_j) \in \mathbb{R}^6 : \|x_i - x_j - x_{ij,\text{des}}\| = 0\},$$

design decentralized control laws u_i, such that for every $i \in \mathcal{V}$ and $\forall t \in \mathbb{R}_{\geq 0}$ the following hold:

1. $\lim\limits_{t \to \infty} \|x_i(t) - x_j(t) - x_{ij,\text{des}}\| \leq \mu, \forall j \in \mathcal{N}_i(x_0)$;
2. $\|p_i(t) - p_j(t)\| > \underline{d}_{ij,a}, \forall j \in \mathcal{V} \backslash \{i\}, t \in \mathbb{R}_{\geq 0}$;
3. $\|p_i(t) - p_{o_z}\| > \underline{d}_{iz,o}, \forall z \in \mathcal{Z}, t \in \mathbb{R}_{\geq 0}$;
4. $\|p_i(t)\| + \underline{r}_i < \underline{r}_w, \forall t \in \mathbb{R}_{\geq 0}$;
5. $\|p_i(t) - p_j(t)\| < d_i, \forall j \in \mathcal{N}_i(x_0), t \in \mathbb{R}_{\geq 0}$;
6. $-\frac{\pi}{2} < -\bar{\theta} \leq \theta_i(t) \leq \bar{\theta} < \frac{\pi}{2}$, $\forall t \in \mathbb{R}_{\geq 0}$;

where μ is an arbitrarily small positive constant for every $i \in \mathcal{V}$.

The aforementioned specifications imply the following:
1) stands for formation control (both position and orientation);
2) stands for inter-agent collision avoidance;
3) stands for collision avoidance between the agents and the obstacles;
4) stands for collision avoidance between the agents and the boundary;
5) stands for connectivity maintenance of the initially connected agents and finally, and
6) stands for the representation of singularities avoidance.

3.4 Control Law Design

In this section, a systematic solution to Problem 3.1 is introduced. Our overall approach builds on designing a decentralized potential function for each agent that captures all the desired control specifications. This potential function will then be exploited by the decentralized controller of each agent. In particular, the following analysis is performed:

- The form of the proposed potential function along with its components is described in Section 3.4.1.
- The proposed decentralized controllers that guarantee the satisfaction of all the control specifications are provided in Section 3.4.2. The required stability analysis is presented subsequently.

3.4.1 Decentralized Potential Functions

In order to solve the formation control problem with the collision- and singularity-avoidance as well as connectivity maintenance, we construct a *decentralized potential function* for each agent $i \in \mathcal{V}$ of the form:

$$\varphi_i(x) = \gamma_i(x) + \frac{1}{\beta_i(x)}, \tag{3.4}$$

where:

- $\gamma_i(x) \in \mathbb{R}_{\geq 0}$ is the *goal function* that vanishes when agent i is at the desired position and orientation with its neighbors.
- $\beta_i(x) \in \mathbb{R}_{\geq 0}$ is an *obstacle function* that encodes collisions between agents and obstacles, collisions between agents and the obstacle boundary, connectivity losses between initially connected agents and singularities of the Jacobian matrix $J(q_i)$; $\beta_i(x)$ becomes zero when one or more of the above situations occurs.

The general form of the proposed potential function φ_i is motivated by the initial work of navigation functions [68]. The function φ_i proposed in this work, however, is not argued to be a navigation function. In the sequel, we describe the construction of a function of the form (3.4).

γ_i - Goal Function

The function γ_i encodes the control objective of agent i, which is to achieve position and orientation formation with its neighboring agents. In view of that, define the goal function by:

$$\gamma_i(x) = \frac{1}{2} \sum_{j \in \mathcal{N}_i(x_0)} \|x_i - x_j - x_{ij,\text{des}}\|^2. \tag{3.5}$$

The function γ_i reaches its unique global minimum when both $p_i - p_j = d_{ij,\text{des}}$ and $q_i - q_j = q_{ij,\text{des}}, \forall i \in \mathcal{V},\ j \in \mathcal{N}_i(x_0)$, i.e., when both formation and orientation alignment are achieved between all the initial neighboring agents. It should be noted that the gradient of γ_i with respect to x_i, computed as

$$\begin{aligned}\nabla_{x_i}\gamma_i(x) &= \sum_{j\in\mathcal{N}_i(x_0)} (x_i - x_j - x_{ij,\text{des}}) \\ &= N_i(x_0)x_i - \sum_{j\in\mathcal{N}_i(x_0)} (x_j + x_{ij,\text{des}}),\end{aligned} \tag{3.6}$$

can vanish in more configurations except for the desired one, at it is shown in the following example.

Example 3.1. Consider a system with three agents $\mathcal{V} = \{1, 2, 3\}$. Assume that 1 is neighbor with both agents 2 and 3. Then, the goal function of agent 1 is given by:

$$\gamma_1(x) = \frac{1}{2}\|x_1 - x_2 - [2,0,0,0,0,0]^\top\|^2 + \frac{1}{2}\|x_1 - x_3 - [3,0,0,0,0,0]^\top\|^2.$$

The latter results in $\nabla_{x_i}\gamma_i(x) = 2x_1 - \left(x_2 + x_3 + [5,0,0,0,0,0]^\top\right)$, which vanishes for any x_1, x_2, x_3 that satisfy $2x_1 - x_2 - x_3 = [5,0,0,0,0,0]^\top$. Clearly, this can happen in configurations other than the desired one. For example, consider the configuration:

$$x_1^\star = [1,0,0,0,0,0]^\top, x_2^\star = [0,0,0,0,0,0]^\top, x_3^\star = [-3,0,0,0,0,0]^\top.$$

It holds that

$$\gamma_1(x^\star) \neq 0, \nabla_{x_1}\gamma_1(x)\Big|_{x=x^\star} = 0. \tag{3.7}$$

β_i - Obstacle Function

The function β_i, encodes all inter-agent collisions, collisions between the agents and obstacles, collisions with the boundary of the workspace, connectivity between initially connected agents and singularities of the Jacobian matrix $J(q_i)$. First, for each agent i, we define the functions $\eta_{ij,a}$, $\eta_{ij,c} : \mathbb{R}^2 \to \mathbb{R}$, $\eta_{iz,o} : \mathbb{R} \to \mathbb{R}$ where:

$$\eta_{ij,a}(p_i, p_j) = d_{ij,a}^2(p_i, p_j) - \underline{d}_{ij,a}^2, \tag{3.8a}$$

$$\eta_{iz,o}(p_i) = d_{iz,o}^2(p_i) - \underline{d}_{iz,o}^2, \tag{3.8b}$$

$$\eta_{ij,c}(p_i, p_j) = d_i^2 - d_{ij,a}^2(p_i, p_j). \tag{3.8c}$$

The subscripts j and z correspond to agent $j \in \mathcal{V}\backslash\{i\}$ and obstacle $z \in \mathcal{Z}$, respectively, whereas the subscripts c stands for *connectivity.* Let us also define the functions $b_{ij,a}$, $b_{iz,o}$, $b_{ij,c} : \mathbb{R} \to [0, \kappa]$, $b_{iw} : \mathbb{R} \to \mathbb{R}_{\geq 0}$, $b_{J_i} : [-\frac{\pi}{2}, \frac{\pi}{2}] \to [0, 1]$, where:

$$b_{ij,a}(y) = \begin{cases} \phi_{i,a}(y), & 0 \leq y < d_i^2 - \underline{d}_{ij,a}^2, \\ \kappa, & d_i^2 - \underline{d}_{ij,a}^2 \leq y, \end{cases} \tag{3.9a}$$

$$b_{iz,o}(y) = \begin{cases} \phi_{i,o}(y), & 0 \leq y < d_i^2 - \underline{d}_{iz,o}^2, \\ \kappa, & d_i^2 - \underline{d}_{iz,o}^2 \leq y, \end{cases} \tag{3.9b}$$

$$b_{ij,c}(y) = \begin{cases} 0, & y < 0, \\ \phi_{i,c}(y), & 0 \leq y < d_i^2 - \underline{d}_{ij,a}^2, \\ \kappa, & d_i^2 - \underline{d}_{ij,a}^2 \leq y, \end{cases} \tag{3.9c}$$

$$b_{iw}(y) = \left[(r_w - r_i)^2 - y^2\right]^2, \tag{3.9d}$$

$$b_{J_i}(y) = \cos^2(y). \tag{3.9e}$$

The functions $b_{ij,a}$, $b_{iz,o}$, $b_{ij,c}$ correspond to inter-agent collision, collision with obstacles and connectivity maintenance, respectively, for agent $i \in \mathcal{V}$, while the functions b_{iw}, b_{J_i} correspond to collision with the workspace boundary and representation singularities. Each of these terms becomes zero when there is a collision, a connectivity break or a representation singularity. Note that all the aforementioned functions use only local information depending on the sensing range d_i of agent i. The functions $\phi_{i,a}$, $\phi_{i,o}$ and $\phi_{i,c}$ are *increasing functions*, appropriately selected to guarantee that the functions $b_{ij,a}$, $b_{iz,o}$ and $b_{ij,c}$ respectively, are twice continuously differentiable everywhere, with $\phi_{i,a}(0) = \phi_{i,o}(0) = \phi_{i,c}(0) = 0$, $\forall i \in \mathcal{V}$. Moreover, κ is a positive constant to be defined later. An example of a function $b_{ij,a}$ with $d_i^2 - \underline{d}_{ij,a}^2 = 5$ and $\phi_{i,a}(y) = 0.008y^3 - 0.12y^2 + 0.6y$, is depicted in Figure 3.2. We can now choose the function β_i as the following product for every $i \in \mathcal{V}$:

$$\beta_i(x) = b_{J_i}(\theta_i) b_{iw}(\|p_i\|^2) \left[\prod_{j \in \mathcal{V} \setminus \{i\}} b_{ij,a}(\eta_{ij,a}(p_i, p_j))\right] \left[\prod_{z \in \mathcal{Z}} b_{iz,o}(\eta_{iz,o}(p_i))\right] \left[\prod_{j \in \mathcal{N}_i(x_0)} b_{ij,c}(\eta_{ij,c}(p_i, p_j))\right]. \tag{3.10}$$

The aforementioned function becomes zero when one or more of the desired specifications is violated.

3.4.2 Control Design and Stability Analysis

In this section, we design controllers u_i such that all the specifications of Problem 3.1 are met. First, let us define the augmented vectors:

$$\zeta_i = [\gamma_i, \frac{1}{\beta_i}, v_i^\top]^\top \in \mathbb{R}^8, \tag{3.11a}$$

$$\zeta = [\zeta_1^\top, \ldots, \zeta_N^\top]^\top \in \mathbb{R}^{8N}. \tag{3.11b}$$

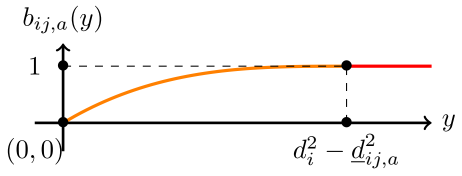

Figure 3.2: The function $b_{ij,a}(y)$, for $d_i^2 - \underline{d}_{ij,a}^2 = 5$ and $\phi_{i,a}(y) = 0.008y^3 - 0.12y^2 + 0.6y$.

Next, define the following open sets:

$$\begin{aligned}\mathcal{D}_i = \Big\{&(x,v) \in \mathbb{M}^N \times \mathbb{R}^{6N} : \|p_i - p_j\| > \underline{d}_{ij,a}, \forall j \in \mathcal{V},\\ &\|p_i - p_j\| < d_i, \forall j \in \mathcal{N}_i(x_0), \|p_i - p_{o_z}\| > \underline{d}_{iz,o}, \forall z \in \mathcal{Z},\\ &p_i \in W, \theta_i \in [-\bar{\theta}, \bar{\theta}]\Big\}, i \in \mathcal{V}, \end{aligned} \tag{3.12a}$$

$$\mathcal{D} = \bigcap_{i \in \mathcal{V}} \mathcal{D}_i, \tag{3.12b}$$

$$\mathcal{X} = \Big\{\zeta \in \mathbb{R}^{8N} : \beta_i > 0, \forall i \in \mathcal{V}\}, \tag{3.12c}$$

which, according to Assumption 3.1, are connected. Define also a positive constant:

$$r_0 = \max\{r \in \mathbb{R}_{>0} : \mathcal{B}(0_{8N\times 1}, r_0) \subseteq \mathcal{X}\}. \tag{3.13}$$

Note that the functions γ_i, β_i have the same form for all the agents, i.e., $\gamma_i(x)$ consists of the formation errors of agent i and $\beta_i(x)$ consists of terms corresponding to θ_i and distances associated with p_i and its neighbors. Therefore, since from Assumption 3.1 each agent $i \in \mathcal{V}$ knows $x_j, N_i(x), x_{j^*}, N_j(x), \forall j^* \in \mathcal{N}_j(x), j \in \mathcal{N}_i(x)$ at a configuration x, it can also compute $\varphi_j(x)$ and $\nabla_{x_i}\varphi_j(x), \forall j \in \mathcal{N}_i(x)$, by off-line transmission of some constant terms required for $\beta_j(x)$, like the radii r_j and d_j and the functions $\phi_{j,a}, \phi_{j,o}, \phi_{j,c}, \phi_{j,\gamma}$.
Design now the decentralized control laws $u_i : \mathcal{D}_i \to \mathbb{R}^6$, $i \in \mathcal{V}$ as:

$$u_i(x,v) = g_i(x_i) - [J(q_i)]^\top \lambda_i(x) + v_i\left(-\frac{\gamma_i(x)^2}{\sigma^2} - \frac{1}{\sigma^2 \beta_i(x)^2} - \frac{\|v_i\|^2}{\sigma^2} - \frac{1}{N} - 1\right), \tag{3.14}$$

where σ is a positive and finite controller gain to be defined later. The function λ_i is defined by:

$$\lambda_i(x) = \nabla_{x_i}\left(\varphi_i(x) + \sum_{j \in \mathcal{N}_i(x)} \varphi_j(x)\right)$$

$$
\begin{aligned}
&= \nabla_{x_i}\left[k\gamma_i(x) + k\sum_{j\in\mathcal{N}_i(x)}\gamma_j(x) + \frac{1}{\beta_i(x)} + \sum_{j\in\mathcal{N}_i(x)}\frac{1}{\beta_j(x)}\right] \\
&= kN_i(x_0)\nabla_{x_i}\gamma_i(x) + \nabla_{x_i}\left[\frac{1}{\beta_i(x)} + \sum_{j\in\mathcal{N}_i(x)}\frac{1}{\beta_j(x)}\right]. \qquad (3.15)
\end{aligned}
$$

Assumption 3.3. We assume here that any equilibrium points of the closed loop system (5.1) in the set

$$\mathcal{M} = \{\zeta \in \mathcal{X} : \|v\| = 0, \|\zeta\| \geq \sigma\}, \qquad (3.16)$$

are *unstable*, where σ as defined in (3.14), as well as the system does not start in these equilibrium points.

We now state the following theorem, which summarizes the main results of this chapter.

Theorem 3.1. *Suppose that Assumptions 3.1, 3.2 and 3.3 hold and let the initial states* (x_0, v_0) *such that* $\zeta_0 \in \mathcal{B}(0_{8N\times 1}, r_0)$*, as* r_0 *defined in* (3.13)*. Then, the decentralized control protocol* (3.14) *guarantees that* $\lim_{t\to\infty}\|x_i(t) - x_j(t) - x_{ij,des}\| \leq \mu$*,* $\forall j \in \mathcal{N}_i(x_0)$*,* $i \in \mathcal{V}$*, ensuring that* $\beta_i(x(t)) > 0$*,* $\forall t \in \mathbb{R}_{\geq 0}, i \in \mathcal{V}$ *and the boundedness of all closed loop signals, providing, thus, a solution to Problem 3.1.*

Proof. Consider the positive definite continuously differentiable function $L : \mathcal{D} \to \mathbb{R}_{\geq 0}$ for the system (5.1):

$$L(\zeta) = \sum_{i\in\mathcal{V}}\left\{\varphi_i + \frac{1}{2}v_i^\top M_i(x_i)v_i\right\}, \qquad (3.17)$$

which satisfies $\alpha_1(\|\zeta\|) \leq L(\zeta) \leq \alpha_2(\|\zeta\|)$, for some functions α_1, $\alpha_2 \in \mathcal{K}$. By defining $L^d : \mathcal{D} \to \mathbb{R}$, with $L^d \triangleq [\nabla_\zeta L(\zeta)]^\top\dot{\zeta}$, we obtain that:

$$
\begin{aligned}
L^d(\zeta) &= \sum_{i\in\mathcal{V}}\left\{(\nabla_{x_i}\varphi_i)^\top\dot{x}_i + \sum_{j\in\mathcal{N}_i(x)}\left(\nabla_{x_j}\varphi_i\right)^\top\dot{x}_j\right\} + \sum_{i\in\mathcal{V}}v_i^\top\left[M_i\dot{v}_i + \frac{1}{2}M_i^d v_i\right] \\
&= \sum_{i\in\mathcal{V}}\left\{(\nabla_{x_i}\varphi_i)^\top J(q_i)v_i + \sum_{j\in\mathcal{N}_i(x)}\left(\nabla_{x_j}\varphi_i\right)^\top J(q_j)v_j\right\} \\
&\qquad + \sum_{i\in\mathcal{V}}v_i^\top\left[-C_i v_i - g_i + u_i + \frac{1}{2}M_i^d v_i\right] \\
&= \sum_{i\in\mathcal{V}}\left\{v_i^\top[J(q_i)]^\top\nabla_{x_i}\varphi_i + \sum_{j\in\mathcal{N}_i(x)}v_j^\top[J(q_j)]^\top\nabla_{x_j}\varphi_i\right\} \\
&\qquad + \frac{1}{2}\sum_{i\in\mathcal{V}}v_i^\top\left[M_i^d - 2C_i\right]v_i + \sum_{i\in\mathcal{V}}v_i^\top[u_i - g_i].
\end{aligned}
$$

For the form of the derivations that are appearing in the later equality, we refer the reader to Appendix A. By employing the property (3.3) of Remark 1, one obtains:

$$\begin{aligned} L^d &= \sum_{i\in\mathcal{V}} \left\{ v_i^\top \left[[J(q_i)]^\top \left(\nabla_{x_i}\varphi_i + \nabla_{x_i} \sum_{j\in\mathcal{N}_i(x)} \varphi_j \right) \right] \right\} + \sum_{i\in\mathcal{V}} v_i^\top [u_i - g_i] \\ &= \sum_{i\in\mathcal{V}} \left\{ v_i^\top \left[u_i - g_i + [J(q_i)]^\top \lambda_i(x) \right] \right\}, \end{aligned} \tag{3.18}$$

where λ_i as defined in (3.15). By defining $\Lambda \triangleq \min\limits_{k\in\mathcal{V}} \|v_k\|^2 \leq \|v_i\|$, and substituting the controller u_i from (3.14), the latter becomes:

$$\begin{aligned} L^d(\zeta) &= \sum_{i\in\mathcal{V}} \|v_i\|^2 \left(-\frac{\gamma_i^2}{\sigma^2}\right) + \sum_{i\in\mathcal{V}} \|v_i\|^2 \left(-\frac{1}{\sigma^2\beta_i^2}\right) + \sum_{i\in\mathcal{V}} \|v_i\|^2 \left(-\frac{\|v_i\|^2}{\sigma^2}\right) \\ &\quad + \sum_{i\in\mathcal{V}} \|v_i\|^2 \left(-\frac{1}{N}\right) - \sum_{i\in\mathcal{V}} \|v_i\|^2 \\ &\leq \Lambda \sum_{i\in\mathcal{V}} \left(-\frac{\|\gamma_i\|^2}{\sigma^2}\right) + \Lambda \sum_{i\in\mathcal{V}} \left(-\frac{1}{\sigma^2\beta_i^2}\right) + \Lambda \sum_{i\in\mathcal{V}} \left(-\frac{\|v_i\|^2}{\sigma^2}\right) + \Lambda - \|v\|^2 \\ &\leq -\|v\|^2 + \Lambda \left[\sum_{i\in\mathcal{V}} \left(-\frac{\|\gamma_i\|^2}{\sigma^2} - \frac{1}{\sigma^2\beta_i^2} - \frac{\|v_i\|^2}{\sigma^2} \right) + 1 \right] \\ &= -\|v\|^2 + \sigma^2\Lambda \left[-\|\zeta\|^2 + \sigma^2 \right]. \end{aligned} \tag{3.19}$$

The latter implies that $L^d \leq 0$, when $\|\zeta\| \geq \sigma$. The control gain $\sigma > 0$ is designed sufficiently small such that

$$\sigma < \alpha_2^{-1}(\alpha_1(r_0)).$$

Since the equilibrium points in $\mathcal{M}$ are assumed to be unstable, we have the following. Since $\zeta_0 \in \mathcal{B}(0_{8N\times 1}, r_0)$, by invoking Theorem 2.2 from Chapter 2, we conclude that there exists a K_∞ function α_3 and a positive time T such that for every initial condition satisfying $\|\zeta_0\| \leq \alpha_2^{-1}(\alpha_1(r_0))$, the solution of the closed loop system satisfies the following:

$$\begin{aligned} &\|\zeta(t)\| \leq \alpha_3(\|\zeta_0\|), \forall\ 0 \leq t \leq T, \\ &\|\zeta(t)\| \leq \bar{\zeta} \triangleq \alpha_1^{-1}(\alpha_2(\sigma)), \forall t > T. \end{aligned} \tag{3.20}$$

Hence, we conclude that $\zeta(t)$ stays bounded for all $t \in \mathbb{R}_{\geq 0}$ and, thus the following holds:

$$\frac{1}{\beta_i(x(t))} \leq M \triangleq \max\left\{ [\alpha_3(\|\zeta_0\|)]^2, \left[\alpha_1^{-1}(\alpha_2(r_\sigma))\right]^2 \right\},$$

or equivalently,

$$\beta_i(x(t)) \geq \frac{1}{M} > 0,$$

i.e., $\forall t \in \mathbb{R}_{\geq 0}$, $i \in \mathcal{V}$. Hence, inter-agent collisions, collisions between the agents and the obstacles / workspace boundary as well as connectivity losses and singularities are avoided. Furthermore, by tuning the parameter σ, we can guarantee that the bound $\bar{\zeta}$ in (3.20) is small enough in order to hold that $\lim\limits_{t\to\infty} \|x_i(t) - x_j(t) - x_{ij,\text{des}}\| \leq \mu$, $\forall i \in \mathcal{V}, j \in \mathcal{N}_i(x_0)$. Note that this also implies that $\beta_i(x_i(t)) \leq \mu$, $\forall t \geq 0$. In order to guarantee that such configurations exist, we can set the value for κ in (3.9) sufficiently large, according to the desired $x_{ij,\text{des}}$.

Hence, by designing u_i as in (3.14), we guarantee that $\|\zeta(t)\| \leq \bar{\zeta}$. Furthermore, the obstacle functions β_i are proved to be always bounded, which guarantees inter-agent collision avoidance, collision avoidance between the agents and the obstacles/workspace boundary, connectivity maintenance between the initially connected agents, and representation singularity avoidance. Moreover, from (3.20) we conclude that all closed loop signals remain bounded, which leads to the conclusion of the proof. □

Remark 3.3. Assumption 3.3 forms an instability claim that concerns a subset of initial configurations and has not been proved yet, constituting an area of current research. A potential solution lies in the introduction of gain parameters in the potential field.

3.5 Simulation Results

To demonstrate the efficiency of the proposed control protocol, we consider a simulation example with $N = 4$, $\mathcal{V} = \{1, 2, 3, 4\}$ spherical agents of the form (3.1), with $\underline{r}_i = 0.25$m and $d_i = 5$m, $\forall i \in \{1, \ldots, 4\}$. The initial conditions are set to

$$\begin{aligned}
p_1(0) &= [-3, 0, 5]^\top \text{ m},\\
p_2(0) &= [-1, 4, 4]^\top \text{ m},\\
p_3(0) &= [-3, 4, 2]^\top \text{ m},\\
p_4(0) &= [-4, 3, 6]^\top \text{ m},\\
q_1(0) &= q_2(0) = q_3(0) = q_4(0) = [0, 0, 0]^\top \text{ rad}.
\end{aligned}$$

which imply the initial neighboring sets

$$\mathcal{N}_1(0) = \{2\}, \mathcal{N}_2(0) = \{1, 3, 4\}, \mathcal{N}_3(0) = \{2, 4\}, \mathcal{N}_4(0) = \{2, 3\}.$$

The desired formation is defined by the feasible displacements

$$\begin{aligned}
p_{12,\text{des}} &= -p_{21,\text{des}} = [-1,-1,-2]^\top \text{ m},\\
p_{23,\text{des}} &= -p_{32,\text{des}} = [-2,-3,0]^\top \text{ m},\\
p_{24,\text{des}} &= -p_{42,\text{des}} = [-1,-2,0]^\top \text{ m},\\
p_{34,\text{des}} &= -p_{43,\text{des}} = [1,1,0]^\top \text{ m},\\
q_{12,\text{des}} &= -q_{21,\text{des}} = \left[-\frac{\pi}{4},0,-\frac{\pi}{4}\right]^\top \text{ rad},\\
q_{23,\text{des}} &= -q_{32,\text{des}} = \left[-\frac{\pi}{12},0,0\right]^\top \text{ rad},\\
q_{24,\text{des}} &= -q_{42,\text{des}} = \left[-\frac{\pi}{8},0,0\right]^\top \text{ rad},\\
q_{34,\text{des}} &= -q_{43,\text{des}} = \left[\frac{5\pi}{24},0,0\right]^\top \text{ rad}.
\end{aligned}$$

We consider a workspace of radius $\underline{r}_w = 10$m containing two spherical static obstacles at $p_{o_1} = [-3,3,5]^\top$m, $p_{o_2} = [-1,1,3]^\top$m with radii $\underline{r}_{o_1} = \underline{r}_{o_2} = 0.75$m. An illustration of the workspace with the agents at $t = 0$ is given in Figure 3.3. The controller parameter were chosen as $\sigma = 0.3$. The simulation results are depicted in Figure 3.4-3.7 for $t \in [0,30]$sec. In particular, Figure 3.5 shows the evolution of the goal functions $\gamma_i, \forall i \in \mathcal{V}$, which are decreasing to zero, whereas Figure 3.6 depicts the obstacle functions β_i, $\forall i \in \mathcal{V}$ which stays always strictly positive. Furthermore, the control inputs are shown in Figure 3.4. Finally, the navigation of the agents in the workspace is pictured in Figure 3.7. As proven in the theoretical analysis, the formation is successfully achieved and all the specifications of Problem 3.1 are met.

3.6 Conclusions

In this chapter we proposed a potential function- based decentralized control protocol for multi-agent systems which guarantees formation control with inter-agent collision avoidance, collision avoidance between the agents and the obstacles/workspace boundary, connectivity maintenance as well as singularity avoidance of multiple rigid bodies. Simulation results have verified the validity of the proposed control design approach.

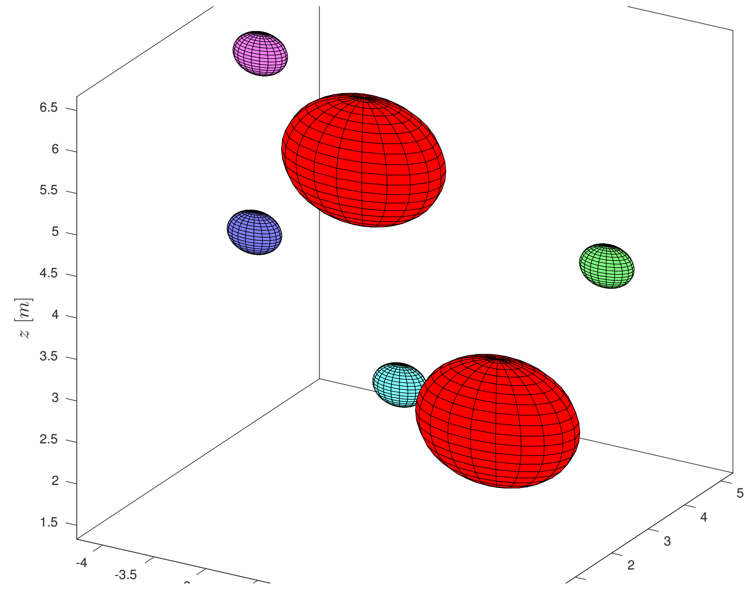

Figure 3.3: The initial workspace of the simulated scenario ($t = 0$). Agent 1 (with blue), agent 2 (with green), agent 3 (with cyan) and agent 4 (with purple) and two obstacles (with red).

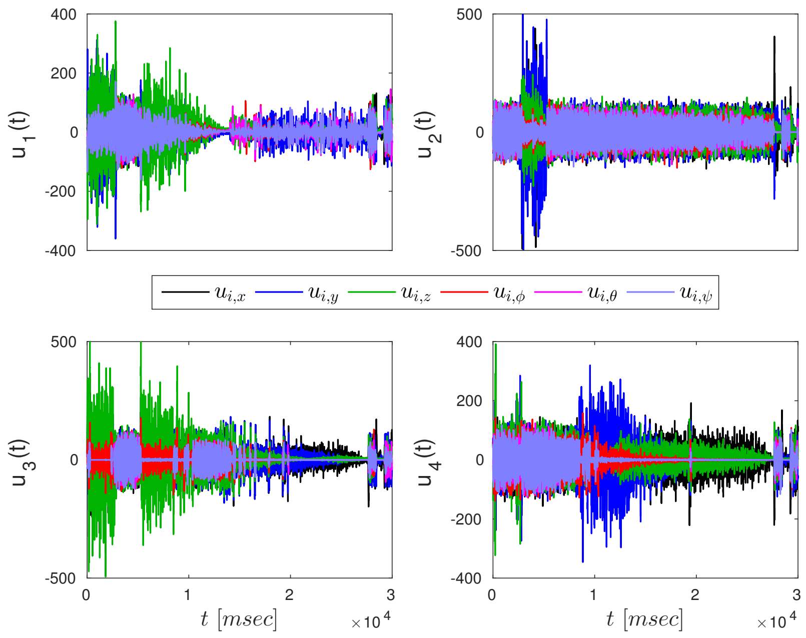

Figure 3.4: The resulting control inputs $u_i, \forall i \in \mathcal{V}$.

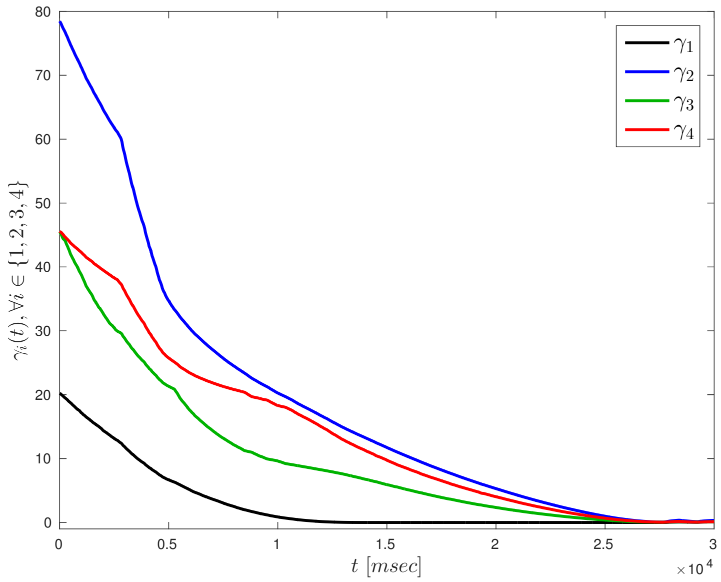

Figure 3.5: The evolution of the goal functions $\gamma_i, \forall i \in \mathcal{V}$, which are shown to converge to zero.

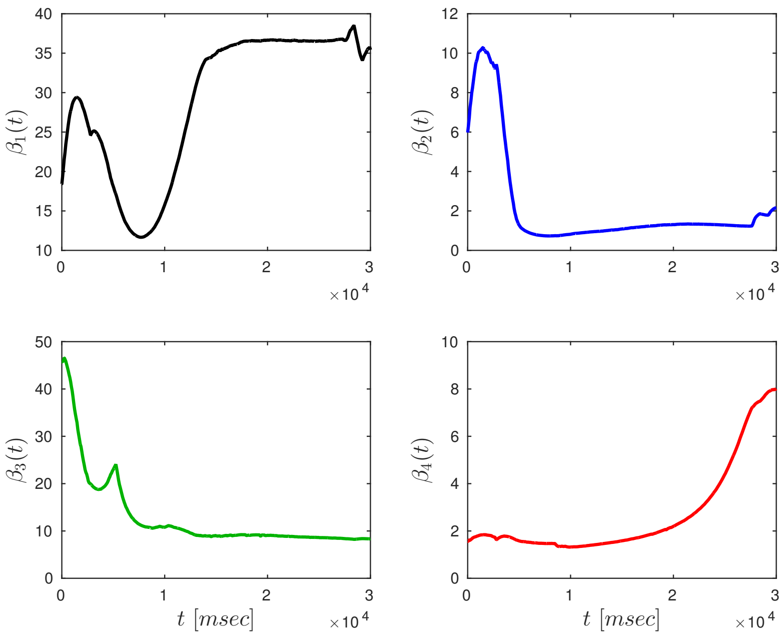

Figure 3.6: The evolution of the obstacle functions $\beta_i, \forall i \in \mathcal{V}$, which are shown to be always strictly positive, i.e., all the desired specifications are fulfilled.

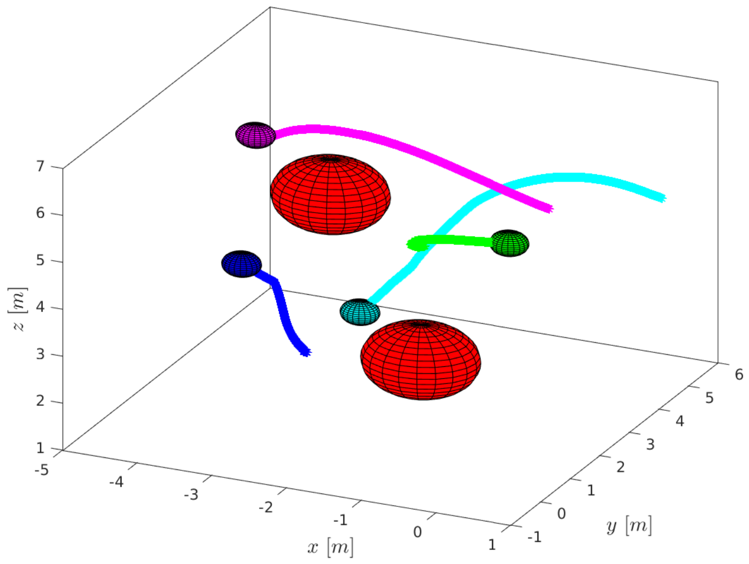

Figure 3.7: The motion of the agents in the workspace for $t \in [0, 30]$sec.

Chapter 4

Timed Constrained High-Level Planning for Multi-Agent Systems

In this chapter the problem of cooperative task planning of multi-agent systems when timed constraints are imposed to the system is investigated. We consider timed constraints given by Metric Interval Temporal Logic (MITL). We propose a method for automatic control synthesis in a two-stage systematic procedure. Under the proposed method, it is guaranteed that all the agents satisfy their own individual task specifications as well as the team satisfies a team task specification.

4.1 Introduction

The specification language that has extensively been used to express desired tasks for robots is Linear Temporal Logic (LTL) (see, e.g., [82–87]). LTL has proven a valuable tool for controller synthesis, because it provides a compact mathematical formalism for specifying desired behaviors of a system. There is a rich body of literature containing algorithms for verification and synthesis of multi-agent systems under temporal logic specifications [88–90]. A three-step hierarchical procedure to address the problem of multi-agent systems under LTL specifications is described as follows [91–97]: first the dynamics of each agent is abstracted into a discrete transition system using abstractions methods. Second, by invoking ideas from formal verification, a discrete plan that meets the high-level tasks is synthesized for each agent. Third, the discrete plan is translated into a sequence of continuous controllers for the original continuous dynamical system of each agent.

Explicit time constraints in the system modeling have been included e.g., in [84], where a method of automated planning of optimal paths of a group of agents satisfying a common high-level mission specification was proposed. The mission was given in LTL and the goal was the minimization of a cost function that captures the maximum time between successive satisfactions of the formula. Authors in [98, 99] used a different approach, representing the motion of each agent in the environment with a timed automaton. The composition of the team automaton was achieved through synchronization and the UPPAAL verification tool ([100]) was utilized for specifications given in Computational Tree Logic (CTL). In the same direction, authors in [101] modeled the multi-robot framework with timed automata and weighted transition systems considering LTL specifications and then, an optimal motion of the robots satisfying instances of the optimizing proposition was proposed.

Most of the previous works on multi-agent planning consider temporal properties which essentially treat time in a qualitative manner. For real applications, a multi-agent team might be required to perform a specific task within a certain time bound, rather than at some arbitrary time in the future (quantitative manner). Controller synthesis under timed specifications has been considered in [102–107]. In [102], the authors addressed the problem of designing high-level planners to achieve tasks for switching dynamical systems under Metric Temporal Logic (MTL) specification and in [103], the authors utilized a counterexample-guided synthesis for cyber-physical systems subject to Signal Temporal Logic (STL) specifications. In [106], the authors focused on motion planning based on the construction of an efficient timed automaton from a given MITL specifications. In [104], an optimal control problem for continuous-time stochastic systems subject to objectives specified in MITL was studied, whereas in [105], a framework that enables on-line planning for robotic systems in dynamic environments under MTL specifications is presented. In [107], the MTL formula for a single-agent was translated into linear constraints and a Mixed Integer Linear Programming (MILP) problem was solved. However, all these works are restricted to single-agent planning and cannot be extended to multi-agent systems in a straightforward way. The multi-agent case has been considered in [108], where the vehicle routing problem was addressed, under Metric Temporal Logic (MTL) specifications. The corresponding approach does not rely on automata-based verification, as it is based on a construction of linear inequalities and the solution of a Mixed-Integer Linear Programming (MILP) problem.

Motivated by the above, in this Chapter, we aim at designing an automated planning procedure for a team of agents. The agents are assigned with an individual, independent timed temporal specification each and, at the same time, a single global team specification. This constitutes the first step towards including time constraints to temporal logic-based multi-agent control synthesis. We consider a quantitative logic called Metric Interval Temporal Logic (MITL) [33] in order to specify explicit time constraints. The proposed solution is fully automated and completely desynchronized in the sense that a faster agent is not required to stay in a region and wait for the slower one. It is decentralized in handling the individual specifications and centralized only in handling the global team specification. To the best of the author's knowledge, a work that address the cooperative task planning for multi-agent systems under individual and global timed linear temporal logic specifications has not been proposed before.

The remainder of the chapter is structured as follows. Section 4.2 provides the model of the multi-agent system, the task specification, several motivation examples as well as the formal problem statement. Section 4.3 discusses the technical details of the solution. Section 4.4 is devoted to an illustrative example. Finally, the conclusions are discussed in Section 4.5.

4.2 Problem Formulation

4.2.1 System Model

Consider a multi-agent team composed of N agents operating in a bounded workspace $\mathcal{W}_0 \subseteq \mathbb{R}^n$. Let $\mathcal{V} = \{1, \ldots, N\}$ denote the index set of the agents. We assume that the workspace $\mathcal{W}_0$ is partitioned into a finite number (assume W) of regions of interest $\pi_1, \ldots, \pi_W$ where

$$\mathcal{W}_0 = \bigcup_{k \in \mathcal{W}} \pi_k \text{ and } \pi_k \cap \pi_{k'} \neq \emptyset, \forall\ k \neq k' \text{ with } k, k' \in \mathcal{W}, \tag{4.1}$$

for the index set $\mathcal{W} = \{1, \ldots, W\}$. We denote by π_k^i the agent i being at region π_k, where $i \in \mathcal{V}, k \in \mathcal{W}$. In this work, we focus on interaction and high-level control strategies rather than on nonlinear models, and we assume that the dynamics of each agent is given by a single integrator

$$\dot{x}_i = u_i, u_i \in \mathcal{U}, i \in \mathcal{V}, \tag{4.2}$$

where $\mathcal{U}$ is a set of input constraints. The partitioned environment (4.1) is a discretization that allows us to control the agents with dynamics (4.2) using finite models such as finite transition systems (e.g., [92, 109–111]). We define a weighted transition system (see Definition 4.1) so that:

- if there exists a controller $u_i,\ i \in \mathcal{V}$ such that agent i can be driven from any point within the region π^i to a neighboring region π^j, then we allow for a transition between the respective system states, and
- the weight of each transition estimates the time each agent needs in order to move from one region to another. In particular, the travel time is here determined as the worst-case shortest time needed to travel from an arbitrary point of the current region to the boundary of the following region. This estimate is indeed conservative, however, it is sufficient for specifications that we are generally interested in within multi-agent control. Namely, it is suitable for scenarios where tasks are given deadlines and upper rather than lower bound requirements are associated with events along the agents' runs.

Definition 4.1. The motion of each agent $i \in \mathcal{V}$ in the workspace is modeled by a WTS $\mathcal{T}_i = (\Pi_i, \Pi_i^{\text{init}}, \longrightarrow_i, d_i, \Sigma_i, L_i)$ where

- $\Pi_i = \left\{\pi_1^i, \pi_2^i, \ldots, \pi_W^i\right\}$ is the set of states of agent i. Any state of an agent i can be denoted as $\pi_k^i \in \Pi_i$ for $i \in \mathcal{V}, k \in \mathcal{W}$. The number of states for each agent is $|\Pi_i| = W$.

- $\Pi_i^{\text{init}} \subseteq \Pi_i$ is the initial states of agent i, i.e. the set of regions where agent i may start.
- $\longrightarrow_i \subseteq \Pi_i \times \Pi_i$ is the transition relation. For example, by $\pi_3^3 \longrightarrow_3 \pi_5^3$ we mean that agent 3 can move from region π_3 to region π_5.
- $d_i : \longrightarrow_i \to \mathbb{T}$ is a map that assigns a positive weight (duration) to each transition. For example, $d_2(\pi_2^2, \pi_5^2) = 0.7,$ where $\pi_2^2 \longrightarrow_2 \pi_5^2$, means that agent 2 needs at most 0.7 time units to move from any point of region π_2 to the boundary of the neighboring region π_5.
- Σ_i is a finite set of atomic propositions known to agent i. Without loss of generality, we assume that $\Sigma_i \cap \Sigma_{i'} = \emptyset$ for all $i \neq i' \in \mathcal{V}$.
- $L_i : \Pi_i \to 2^{\Sigma_i}$ is a labeling function that assigns to each state $\pi_k^i \in \Pi_i$, $k \in \mathcal{W}$, a subset of atomic propositions Σ_i that are satisfied when agent i is in region π_k.

Individual Timed Runs and Words

The behaviors of the individual agents can be captured through their timed runs and timed words. The timed run

$$r_i^t = (r_i(0), \tau_i(0))(r_i(1), \tau_i(1))(r_i(2), \tau_i(2)) \ldots,$$

of each WTS $\mathcal{T}_i,\ i \in \mathcal{V}$ and the corresponding timed word

$$w(r_i^t) = (L_i(r_i(0)), \tau_i(0))(L_i(r_i(1)), \tau_i(1))(L_i(r_i(2))), \tau_i(2)) \ldots,$$

are defined by using the terminology of Definition 2.11.

Remark 4.1. Note that in this chapter, we omit the set of actions *Act* from the definition of WTS, the definition of timed runs and timed words, as they are defined in 2.10, 2.11.

Collective Timed Run and Word

At the same time, the agents form a team and we are interested in their global, collective behaviors, which we formalize through the following definition.

Definition 4.2 (*Collective Run of the Agents*)**.** Let $r_1^t, \ldots, r_N^t$ be individual timed runs of the agents $1, \ldots, N$, respectively, as defined above. Then, the *collective timed run*

$$r_G = (r_G(0), \tau_G(0))(r_G(1), \tau_G(1))(r_G(2), \tau_G(2)) \ldots,$$

of the team of agents is defined inductively as follows:

1. $(r_G(0), \tau_G(0)) = ((r_1(0), \ldots, r_N(0)), \tau_G(0))$.

2. Let $(r_G(j), \tau_G(j)) = ((r_1(j_1), \dots, r_N(j_N)), \tau_G(j))$, where $j \geq 0$ is an index for the current state and time stamp of the collective timed run. Then, the next state and time stamp $(r_G(j+1), \tau_G(j+1)) = ((r_1(z_1), \dots, r_N(z_N)), \tau_G(j+1))$ are given by the following rules:

   - $\ell = \underset{i \in \mathcal{V}}{\operatorname{argmin}}\{\tau_i(j_i + 1)\}$.
   - $\tau_G(j+1) = \tau_\ell(j_\ell + 1)$.
   - $r_i(z_i) = \begin{cases} r_\ell(j_\ell + 1) & \text{if } i = \ell \\ r_i(j_\ell) & \text{if } i \neq \ell. \end{cases}$

Intuitively, given the current states $r_1(j_1), \dots, r_N(j_N)$ and the next states $r_1(j_1 + 1), \dots, r_N(j_N + 1)$ of the individual agents at time $\tau_G(j)$, ℓ is the index of the agent i who will finish its current transition from $r_\ell(j_\ell)$ to $r_\ell(j_\ell + 1)$ the soonest amongst of all the agents. The time of agent ℓ's arrival to its next state $r_\ell(j_\ell + 1)$ becomes the new time stamp $\tau_G(j + 1)$ of the collective timed run. The next state of the collective timed run reflects that each agent i which cannot complete its transition from $r_i(j_k)$ to $r_i(j_k + 1)$ before $\tau_G(j + 1)$ remains in $r_i(j_i)$.

In what follows, we write: $r^t_G = (r_G(0), \tau_G(0))(r_G(1), \tau_G(1)) \dots$, where

$$r_G(j) = (r_1(j_1), \dots, r_N(j_N)),\ j, j_i \geq 0, i \in \mathcal{V},$$

denotes the collective timed run.

Definition 4.3. We define the global set of atomic propositions $\Sigma_G = \bigcup_{i \in \mathcal{V}} \Sigma_i$ and for every state $r_G(j) = (r_1(j_1), \dots, r_N(j_N))$ of a collective timed run, where $j, j_i \geq 0$ and $i \in \mathcal{V}$, we define the labeling function $L_G : \Pi_1 \times \dots \times \Pi_N \to \Sigma_G$ as

$$L_G(r_G(j)) = \bigcup_{i \in \mathcal{V}}^{N} L_i(r_i(j_i)).$$

Therefore, a collective timed run r^t_G naturally produces a timed word

$$w^t_G = (L_G(r_G(0)), \tau_G(0))(L_G(r_G(1)), \tau_G(1)) \dots,$$

over the set Σ_G.

Example 4.1. Consider $N = 2$ robots operating in a workspace with $\mathcal{W} = \pi_1 \cup \pi_2 \cup \pi_3, W_0 = 3$ and $\mathcal{V} = \{1, 2\}$ modeled as the WTSs illustrated in Figure 4.1. Let $\Sigma_1 = \{green\}$, and $\Sigma_2 = \{red\}$. The labeling functions are $L_1(\pi^1_1) = \{green\}, L_1(\pi^1_2) = L_1(\pi^1_3) = \emptyset$, and $L_2(\pi^2_1) = L_2(\pi^2_2) = \emptyset, L_2(\pi^2_3) = \{red\}$.

$$\begin{tikzcd}
\text{WTS } \mathcal{T}_1 : \arrow[r] & \pi_1^1 \arrow[r, bend left, "1.0"] & \pi_2^1 \arrow[l, bend left, "2.0"] \arrow[r, bend left, "1.5"] & \pi_3^1 \arrow[l, bend left, "0.5"]
\end{tikzcd}$$

$$\begin{tikzcd}
\text{WTS } \mathcal{T}_2 : \arrow[r] & \pi_1^2 \arrow[r, bend left, "2.0"] & \pi_2^2 \arrow[l, bend left, "1.5"] \arrow[r, bend left, "0.5"] & \pi_3^2 \arrow[l, bend left, "2.0"]
\end{tikzcd}$$

Figure 4.1: Two WTSs $\mathcal{T}_1, \mathcal{T}_2$ representing two agents in $\mathcal{W}$ with $\Pi_1 = \{\pi_1^1, \pi_2^1, \pi_3^1\}$, $\Pi_1^{\text{init}} = \{\pi_1^1\}$, $\Pi_2 = \{\pi_1^2, \pi_2^2, \pi_3^2\}, \Pi_2^{\text{init}} = \{\pi_1^2\}$. The transitions are depicted with arrows which are annotated with the corresponding weights.

A timed run for each agent is given as follows:

$$\begin{aligned}
r_1^t =& (r_1(0) = \pi_1^1, \tau_1(0) = 0.0)(r_1(1) = \pi_2^1, \tau_1(1) = 1.0)\\
& (r_1(2) = \pi_3^1, \tau_1(2) = 2.5)(r_1(3) = \pi_2^1, \tau_1(3) = 3.0)\\
& (r_1(4) = \pi_1^1, \tau_1(4) = 5.0)\dots\\
r_2^t =& (r_2(0) = \pi_1^2, \tau_2(0) = 0.0)(r_2(1) = \pi_2^2, \tau_2(1) = 2.0)\\
& (r_2(2) = \pi_3^2, \tau_2(2) = 2.5)(r_2(3) = \pi_2^2, \tau_2(3) = 4.5)\\
& (r_2(4) = \pi_3^2, \tau_2(4) = 5.0)\dots
\end{aligned}$$

Given r_1^t and r_2^t the collective run r_G is given according to Definition 4.2 as follows:

$$\begin{aligned}
r_G^t =& (\underbrace{(\pi_1^1, \pi_1^2)}_{r_G(0)}, \tau_G(0) = 0.0)(\underbrace{(\pi_2^1, \pi_1^2)}_{r_G(1)}, \tau_G(1) = 1.0)\\
& (\underbrace{(\pi_2^1, \pi_2^2)}_{r_G(2)}, \tau_G(2) = 2.0)(\underbrace{(\pi_3^1, \pi_3^2)}_{r_G(3)}, \tau_G(3) = 2.5)\\
& (\underbrace{(\pi_2^1, \pi_3^2)}_{r_G(4)}, \tau_G(4) = 3.0)(\underbrace{(\pi_2^1, \pi_2^2)}_{r_G(5)}, \tau_G(5) = 4.5)\\
& (\underbrace{(\pi_1^1, \pi_3^2)}_{r_G(6)}, \tau_G(6) = 5.0)\dots
\end{aligned}$$

The produced collective timed word is

$$\begin{aligned}
w_G^t =& (\{green\}, 0.0)(\emptyset, 1.0)(\emptyset, 2.0)(\{red\}, 2.5)\\
& (\{red\}, 3.0)(\emptyset, 4.5)(\{green, red\}, 5.0)\dots.
\end{aligned}$$

4.2.2 Specification

Several temporal logics have been designed to express timed properties of real-time systems, such as MTL [112] that extends the until operator of LTL with a time interval. Here, we consider a fragment of MTL, called MITL (see Section 2.2) which has been proposed in [33]. Namely, we utilize its point-wise semantics and interpret its formulas over timed runs.

Local Agent's Specification

Each agent $i \in \mathcal{V}$ is given an individual, local, independent specification in the form of a MITL formula φ_i over the set of atomic propositions Σ_i. The satisfaction of φ_i is decided from the agent's own perspective, i.e., on the timed run r_i^t.

Global Team Specification

In addition, the team of agents is given a global team specification, which is a MITL formula φ_G over the set of atomic propositions Σ_G. The team specification satisfaction is decided on the collective timed run r_G^t.

Example 4.1 (Continued)**.** Recall the two agents from Example 4.1. Each of the agents is given a local, independent, specification and at the same time, the team is given an overall goal that may require collaboration or coordination. Examples of local specification formulas are $\varphi_1 = \Box\Diamond_{\leq 10}(\mathit{green})$ and $\varphi_2 = \Box(\mathit{red} \Rightarrow \bigcirc\Box_{\leq 5}(\neg \mathit{red}))$ stating that "The green region is periodically visited with at most 10 time units between two consecutive visits" and "Whenever a red region is visited, it will not be visited for the following 5 time units again", respectively. While φ_1 is satisfied on r_1^t, φ_2 is not satisfied on r_2^t. An example of the global specification is $\varphi_G = \Box\Diamond_{\leq 5}(\mathit{green} \wedge \mathit{red})$ that imposes requirement on the agents' collaboration; it states that agents 1 and 2 will periodically and simultaneously visit the green and the red region, respectively, with at most 5 time units between two consecutive visits.

4.2.3 Problem Statement

Problem 4.1 (Run Synthesis)**.** Given N agents governed by dynamics as in (4.2), a task specification MITL formula φ_G for the team of robots, over a set of atomic propositions Σ_G and N local task specifications φ_k over $\Sigma_i,\ i \in \mathcal{V}$, synthesize a sequence of individual timed runs $r_1^t, \ldots, r_N^t$ such that the following hold

$$\left(r_G^t \models \varphi_G\right) \wedge \left(r_1^t \models \varphi_1 \wedge \cdots \wedge r_N^t \models \varphi_N\right). \tag{4.3}$$

Though it might seem that the satisfaction of the individual specifications $\varphi_1, \ldots, \varphi_N$ can be treated as the satisfaction of the formula $\bigwedge\limits_{i \in \mathcal{V}} \varphi_i$ on the collective timed run r_G^t, this is generally not the case, as demonstrated through the following example:

Example 4.1 (Continued)**.** Recall the two agents from Example 4.1 and a local specification $\varphi_2 = \Box(\mathit{red} \Rightarrow \bigcirc\Box_{\leq 2}(\neg \mathit{red}))$. While this specification is satisfied on r_2^t

since $w(r_2^t) = (\emptyset, 0.0)(\emptyset, 2.0)(\{red\}, 2.5)(\emptyset, 4.5)(\{red\}, 5.0)\ldots$, it can be easily seen that it is not satisfied on r_G^t.

Formally, we have

$$r_G^t \models \bigwedge_{k \in \mathcal{V}} \varphi_k \not\Leftrightarrow r_1^t \models \varphi_1 \wedge \cdots \wedge r_N^t \models \varphi_N. \tag{4.4}$$

Hence, Problem 4.1 may not be treated in a straightforward, fully centralized way. We propose a two-stage solution that first pre-computes all timed runs of the individual agents in a decentralized way and stores them efficiently in weighted transition systems enhanced with a Büchi acceptance condition. Second, these are combined and inspected with respect to guaranteeing the satisfaction of the team specification by the collective timed run.

4.3 Proposed Solution

In this section, we introduce a systematic solution to Problem 4.1. Our overall approach builds on the following steps:

1. We construct TBAs $\mathcal{A}_i, \ \ i \in \mathcal{V}$ and $\mathcal{A}_G$ that accept all the timed words satisfying the specification formulas $\varphi_i, i \in \mathcal{V}$ and φ_G, respectively (Section 4.3.1).

2. We construct a *local Büchi WTS* $\widetilde{\mathcal{T}_i} = \mathcal{T}_i \otimes \mathcal{A}_i$, for all $i \in \mathcal{V}$. The accepting timed runs of $\widetilde{\mathcal{T}_i}$ are the timed runs of the $\mathcal{T}_i$ that satisfy the corresponding local specification formula $\varphi_i, i \in \mathcal{V}$ (Section 4.3.2).

3. We construct a *product Büchi WTS* $\mathcal{T}_G = \widetilde{\mathcal{T}_1} \otimes \cdots \otimes \widetilde{\mathcal{T}_N}$ such that its timed runs are collective timed runs of the team and their projections onto the agents' individual timed runs are admissible by the local Büchi WTSs $\widetilde{\mathcal{T}_1}, \ldots, \widetilde{\mathcal{T}_N}$ respectively (Section 4.3.3).

4. We construct a *global Büchi WTS* $\widetilde{\mathcal{T}_G} = \mathcal{T}_G \otimes \mathcal{A}_G$. The accepting timed runs of the $\widetilde{\mathcal{T}_G}$ are the timed runs of the $\mathcal{T}_G$ that satisfy the team formula φ_G (Section 4.3.4).

5. We find an accepting timed run $\widetilde{r}_G^t$ of the global Büchi WTS $\widetilde{\mathcal{T}_G}$ and project it onto timed runs of the product Büchi WTS $\mathcal{T}_G$, then onto timed runs of the local Büchi WTSs $\widetilde{\mathcal{T}_1}, \ldots, \widetilde{\mathcal{T}_N}$, and finally onto individual timed runs $r_1^t, \ldots, r_N^t$ of the original WTSs $\mathcal{T}_1, \ldots, \mathcal{T}_N$. By construction, $r_1^t, \ldots, r_N^t$ are guaranteed to satisfy $\varphi_1, \ldots, \varphi_N$, respectively, and furthermore r_G^t satisfies φ_G (Section 4.3.5).

4.3.1 Construction of TBAs

As stated in Section 2.3, every MITL formula φ can be translated into a language equivalent TBA. Several approaches are proposed for the translation e.g., [33, 37, 38, 113]. Here, we translate each local specification φ_i, where $i \in \mathcal{V}$ into a TBA $\mathcal{A}_i = (S_i, S_i^{\text{init}}, C_i, I_i, E_i, \mathcal{F}_i, \Sigma_i, \mathcal{L}_i)$, and the global specification φ_G into a TBA $\mathcal{A}_G = (S_G, S_G^{\text{init}}, C_G, I_G, E_G, \mathcal{F}_G, \Sigma_G, \mathcal{L}_G)$.

4.3.2 Construction of the local Büchi WTSs $\widetilde{\mathcal{T}}_1, \ldots, \widetilde{\mathcal{T}}_N$

Definition 4.4. Given a WTS $\mathcal{T}_i = (\Pi_i, \Pi_i^{\text{init}}, \longrightarrow_i, \Sigma_i, L_i, d_i)$, and a TBA $\mathcal{A}_i = (S_i, S_i^{\text{init}}, C_i, I_i, E_i, F_i, \Sigma_i, \mathcal{L}_i)$ with C_i clocks and $C_i^{\max}$ being the largest constant appearing in $\mathcal{A}_i$. Then, their *local Büchi WTS* $\widetilde{\mathcal{T}}_i = \mathcal{T}_i \otimes \mathcal{A}_i = (Q_i,\ Q_i^{\text{init}},\ \rightsquigarrow_i,\ \widetilde{d}_i,\ \widetilde{F}_i,\ \Sigma_i,\ \widetilde{L}_i)$ is defined as follows:

- $Q_i \subseteq \{(r_i, s_i) \in \Pi_i \times S_i : L_i(r_i) = \mathcal{L}_i(s_i)\} \times \mathbb{T}_\infty^{\mathcal{C}_i}$.
- $Q_i^{\text{init}} = \Pi_i^{\text{init}} \times S_i^{\text{init}} \times \{0\}^{\mathcal{C}_i}$.
- $q \rightsquigarrow_i q'$ iff
  - $q = (r, s, \nu_1, \ldots, \nu_{\mathcal{C}_i}) \in Q_i$,
    $q' = (r', s', \nu_1', \ldots, \nu_{\mathcal{C}_i}') \in Q_i$,
  - $r \longrightarrow_i r'$, and
  - there exists γ, R, such that $(s,\ \gamma,\ R,\ s') \in E_i$, $\nu_1,\ \ldots,\ \nu_{\mathcal{C}_i} \models \gamma$, $\nu_1',\ \ldots,\ \nu_{\mathcal{C}_i}' \models I_i(s')$, and for all $i \in \{1,\ \ldots,\ \mathcal{C}_i\}$

$$\nu_i' = \begin{cases} 0, & \text{if } c_i \in R \\ \nu_i + d_i(r, r'), & \text{if } c_i \notin R \text{ and} \\ & \nu_i + d_i(r, r') \le C_i^{\max} \\ \infty, & \text{otherwise.} \end{cases}$$

  Then $\widetilde{d}_i(q, q') = d_i(r, r')$.

- $\widetilde{F}_i = \{(r_i, s_i, \nu_1, \ldots, \nu_{\mathcal{C}_i}) \in Q_i : s_i \in F_i\}$.
- $\widetilde{L}_i(r_i, s_i, \nu_1, \ldots, \nu_{\mathcal{C}_i}) = L_i(r_i)$.

Each local Büchi WTS $\widetilde{\mathcal{T}}_i$, $i \in \mathcal{V}$, is in fact a WTS with a Büchi acceptance condition $\widetilde{F}_i$. A timed run of $\widetilde{\mathcal{T}}_i$ can be written as $\widetilde{r}_i^t = (q_i(0), \tau_i(0))(q_i(1), \tau_i(1)) \ldots$ using the terminology of Definition 2.11. It is *accepting* if $q_i(j) \in \widetilde{F}_i$ for infinitely many $j \ge 0$. An accepting timed run of $\widetilde{\mathcal{T}}_i$ projects onto a timed run of $\mathcal{T}_i$ that satisfies the local specification formula φ_i by construction. Formally, the following lemma, whose proof follows directly from the construction and the principles of automata-based LTL model checking (see, e.g., [30]), holds:

Lemma 4.1. *Consider an accepting timed run $\widetilde{r}_i^t = (q_i(0), \tau_i(0))(q_i(1), \tau_i(1))\dots$ of the local Büchi WTS $\widetilde{\mathcal{T}}_i$ defined above, where $q_i(j) = (r_i(j), s_i(j), \nu_{i,1}, \dots, \nu_{i,\mathcal{C}_i})$ denotes a state of $\widetilde{\mathcal{T}}_i$, for all $j \geq 1$. The timed run $\widetilde{r}_i^t$ projects onto the timed run $r_i^t = (r_i(0), \tau_i(0))(r_i(1), \tau_i(1))\dots$ of the WTS $\mathcal{T}_k$ that produces the timed word $w(r_i^t) = (L_i(r_i(0)), \tau_i(0))(L_i(r_i(1)), \tau_i(1))\dots$ accepted by the TBA $\mathcal{A}_i$ via its run $\rho_i = s_i(0)s_i(1)\dots$. Vice versa, if there exists a timed run $r_i^t = (r_i(0),\ \tau_i(0))(r_i(1),$ $\tau_i(1))\dots$ of the WTS $\mathcal{T}_i$ that produces a timed word*

$$w(r_i^t) = (L_i(r_i(0)), \tau_i(0))(L_i(r_i(1)), \tau_i(1))\dots,$$

accepted by the TBA $\mathcal{A}_i$ via its run $\rho_i = s_i(0)s_i(1)\dots$ then, there exists the accepting timed run $\widetilde{r}_i^t = (q_i(0),\ \tau_i(0))(q_i(1),\ \tau_i(1))\dots$ of $\widetilde{\mathcal{T}}_i$, where $q_i(j)$ denotes $(r_i(j), s_i(j), \nu_{i,1}(j), \dots, \nu_{i,\mathcal{C}_i}(j))$ in $\widetilde{\mathcal{T}}_i$.

4.3.3 Construction of the product Büchi WTS $\mathcal{T}_G$

In this section, we aim to construct a finite product WTS $\mathcal{T}_G$ whose timed runs represent the collective behaviors of the team and whose Büchi acceptance condition ensures that the accepting timed runs account for the local specifications. In other words, $\mathcal{T}_G$ is a product of all the local WTS $\widetilde{\mathcal{T}}_i$ built above. In the construction of $\mathcal{T}_G$, we need to specifically handle the cases when transitions of different agents are associated with different time durations, i.e, different transition weights. To this end, we introduce a vector $b = (b_1, \dots, b_N) \in \mathbb{T}^N$. Each element of the vector is a rational number $b_i \in \mathbb{T}, i \in \mathcal{V}$ which can be either 0, when the agent i has just completed its transition, or the time elapsed from the beginning of the agent's current transition, if this transition is not completed yet. Then, the state of the team of agents is written in the form $q_G = (q_1, \dots, q_N, b_1, \dots, b_N, \ell)$ where q_i is a state of $\widetilde{\mathcal{T}}_i$, for all $i \in \mathcal{V}$, and $\ell \in \mathcal{V}$ has a special meaning in relation to the acceptance condition of $\mathcal{T}_G$ that will become clear shortly. Taking the above into consideration we define the global model $\mathcal{T}_G$ as follows:

Definition 4.5. Given N local Büchi WTSs $\widetilde{\mathcal{T}}_1, \dots, \widetilde{\mathcal{T}}_N$ from Definition 4.4, their *product Büchi WTS* $\mathcal{T}_G = \widetilde{\mathcal{T}}_1 \otimes \dots \otimes \widetilde{\mathcal{T}}_N = (Q_G, Q_G^{\text{init}}, \longrightarrow_G, d_G, F_G, \Sigma_G, L_G)$ is defined as follows:

- $Q_G \subseteq Q_1 \times \cdots \times Q_N \times \mathbb{T}^N \times \{1, \dots, N\}$.
- $Q_G^{\text{init}} = Q_1^{\text{init}} \times \dots \times Q_N^{\text{init}} \times \{0\}^N \times \{1\}$.
- $q_G \longrightarrow_G q_G'$ iff
  - $q_G = (q_1, \dots, q_N, b_1, \dots, b_N, \ell) \in Q_G$,
    $q_G' = (q_1', \dots, q_N', b_1', \dots, b_N', \ell') \in Q_G$,
  - $\exists\ q_i'' \in Q_i : q_i \rightsquigarrow_i q_i''$, for some $i \in \mathcal{V}$,

- ○
$$b_i' = \begin{cases} 0, & \text{if } b_i + d_{\min} = \widetilde{d}_i(q_i, q_i'') \\ & \text{and } q_i' = q_i'' \\ b_i + d_{\min}, & \text{if } b_i + d_{\min} < \widetilde{d}_i(q_i, q_i'') \\ & \text{and } q_i' = q_i \end{cases}$$
where $d_{\min} = \min\limits_{k \in \{1,\ldots,N\}} (\widetilde{d}_i(q_i, q_i'') - b_i)$ is (loosely speaking) the smallest time step that can be applied, and

- ○
$$\ell' = \begin{cases} \ell, & \text{if } q_\ell \notin \widetilde{F}_\ell \\ ((\ell+1) \mod N), & \text{otherwise} \end{cases}$$

Then $d_G(q_G, q_G') = d_{\min}$.

- $F_G = \{(q_1, \ldots, q_N, b_1, \ldots, b_N, N) \in Q_G : q_N \in \widetilde{F}_N\}$.
- $\Sigma_G = \bigcup\limits_{i \in \mathcal{V}} \Sigma_i$.
- $L_G((q_1, \ldots, q_N, b_1, \ldots, b_N, \ell) = \bigcup\limits_{i \in \mathcal{V}} \widetilde{L}_i(q_i)$.

The product WTS $\mathcal{T}_G$ is again a WTS with a Büchi acceptance condition. The index ℓ in a state $q_G = (q_1, \ldots, q_N, b_1, \ldots, b_N, \ell) \in Q_G$ allows to project an accepting timed run of $\mathcal{T}_G$ onto an accepting run of every one of the local Büchi WTS. The construction is based on the standard definition of Büchi automata intersection (see, e.g., [30]).

The following lemma follows directly from the construction and the principles of automata-based LTL model checking (see, e.g., [30]):

Lemma 4.2. *For all $i \in \mathcal{V}$, an accepting timed run r_G^t of the product Büchi WTS $\mathcal{T}_G$ projects onto an accepting timed run r_i^t of the local Büchi WTS $\widetilde{\mathcal{T}}_i$ that produces a timed word $w(r_i^t)$ accepted by the corresponding TBA $\mathcal{A}_i$. Vice versa, if there exists a timed run r_i^t of the local Büchi WTS $\widetilde{\mathcal{T}}_i$ that produces a timed word $w(r_i^t)$ accepted by the TBA $\mathcal{A}_i$ for each $i \in \mathcal{V}$, then there exist an accepting timed run r_G^t of $\mathcal{T}_G$.*

4.3.4 Construction of the global Büchi WTS $\widetilde{\mathcal{T}}_G$

Definition 4.6. Given the product Büchi WTS $\mathcal{T}_G = (Q_G, Q_G^{\text{init}}, \longrightarrow_G, d_G, F_G, \Sigma_G, L_G)$, and a TBA $\mathcal{A}_G = (S_G, S_G^{\text{init}}, C_G, I_G, E_G, \mathcal{F}_G, \Sigma_G, \mathcal{L}_G)$ that corresponds to the team specification formula φ_G with $\mathcal{C}_G$ and $C_G^{\max}$ being the largest constant appearing in $\mathcal{A}_G$, we define their product WTS $\widetilde{\mathcal{T}}_G = \mathcal{T}_G \otimes \mathcal{A}_G = (\widetilde{Q}_G, \widetilde{Q}_G^{\text{init}}, \rightsquigarrow_G, \widetilde{d}_G, \widetilde{F}_G, \Sigma_G, \widetilde{L}_G)$ as follows:

- $\widetilde{Q}_G \subseteq \{(q,s) \in Q_G \times S_G : L_G(q) = \mathcal{L}_G(s)\} \times \mathbb{T}_\infty^{M_G}$.
- $\widetilde{Q}_G^{\text{init}} = Q_G^{\text{init}} \times S_G^{\text{init}} \times \{0\}^{\mathcal{C}_G} \times \{1,2\}$.
- $q \rightsquigarrow_G q'$ iff
  - $q = (r, s, \nu_1, \ldots, \nu_{\mathcal{C}_G}, \ell) \in Q_G$,
    $q' = (r', s', \nu_1', \ldots, \nu_{\mathcal{C}_G}', \ell') \in Q_G$,
  - $r \longrightarrow_G r'$, and
  - there exists γ, R, such that $(s,\ \gamma,\ R,\ s') \in E_G$, $\nu_1, \ldots, \nu_{\mathcal{C}_G} \models \gamma$, $\nu_1', \ldots, \nu_{\mathcal{C}_G}' \models I_G(s')$, and for all $i \in \{1, \ldots, \mathcal{C}_G\}$

$$\nu_i' = \begin{cases} 0, & \text{if } c_i \in R \\ \nu_i + d_G(r,r'), & \text{if } c_i \notin R \text{ and} \\ & \nu_i + d_G(r,r') \leq C_G^{\max} \\ \infty, & \text{otherwise} \end{cases}$$

  - $$\ell' = \begin{cases} 1 \text{ if } \ell = 1 \text{ and } r \notin F_G, \text{ or } \ell = 2 \text{ and } s \in \mathcal{F}_G \\ 2 \text{ otherwise} \end{cases}$$

  Then $\widetilde{d}_G(q,q') = d_G(r,r')$.

- $\widetilde{F}_G = \{(r, s, \nu_1, \ldots, \nu_{\mathcal{C}_G}, 1) \in Q_G : r \in F_G\}$.
- $\widetilde{L}_G(r_G, s_G, \nu_1, \ldots, \nu_{\mathcal{C}_G}) = L_G(r_G)$.

Similarly to the results stated above, the global Büchi WTS $\widetilde{\mathcal{T}}_G$ is a WTS with a Büchi acceptance condition. An accepting timed run of $\widetilde{\mathcal{T}}_G$ guarantees the satisfaction of the team specification formula φ_G by construction. Furthermore, the projected individual timed runs of the original $\mathcal{T}_1, \ldots, \mathcal{T}_N$ satisfy their respective local specifications. The following lemma follows directly from the construction and the principles of automata-based LTL model checking (see, e.g., [30]):

Lemma 4.3. *An accepting timed run $\widetilde{r}_G^t$ of the global Büchi WTS $\widetilde{\mathcal{T}}_G$ projects onto an accepting timed run r_G^t of the product Büchi WTS $\mathcal{T}_G$ that produces a timed word $w(r_G^t)$ accepted by the TBA $\mathcal{A}_G$. Vice versa, if there exists a timed run r_G^t of the product Büchi WTS $\mathcal{T}_G$ that produces a timed word $w(r_G^t)$ accepted by the TBA $\mathcal{A}_G$ then there exist an accepting timed run $\widetilde{r}_G^t$ of $\widetilde{\mathcal{T}}_G$.*

4.3.5 Projection to the desired timed runs of $\mathcal{T}_1, \dots, \mathcal{T}_N$

An accepting run $\widetilde{r}_G^t$ of the global Büchi WTS $\widetilde{\mathcal{T}}_G$ can be found efficiently leveraging ideas from automata-based LTL model checking [30]. Namely, $\widetilde{\mathcal{T}}_G$ is viewed as a graph that is searched for a so-called accepting lasso; a cycle containing an accepting state that is reachable from the initial state. Once $\widetilde{r}_G^t$ is obtained, Lemmas 4.3, 4.2, and 5.6 directly provide guidelines for projection of $\widetilde{r}_G^t$ onto the individual timed runs of $\mathcal{T}_1, \dots, \mathcal{T}_N$. In particular, $\widetilde{r}_G^t$ is projected onto a timed run r_G^t of $\mathcal{T}_G$, which is projected onto timed runs $\widetilde{r}_1^t, \dots, \widetilde{r}_N^t$ of $\widetilde{\mathcal{T}}_1, \dots, \mathcal{T}_N$, which are finally projected onto timed runs $r_1^t, \dots, r_N^t$ of $\mathcal{T}_1, \dots, \mathcal{T}_N$, respectively. Such a projection guarantees that $r_1^t, \dots, r_N^t$ are a solution to Problem 4.1.

4.4 Illustrative Example

For an illustrative example, consider 2 robots in the shared workspace of Figure 4.2. The workspace is partitioned into $W = 21$ cells and a robot's state is defined by the cell that the robot is currently present at. Agent 1 (R1) is depicted in green and it is two times faster than Agent 2 (R2) which is depicted in red. We assume that the environment imposes such moving constraints that the traveling right and up is faster than left and down. Let Agent 1 need 1 time unit for up and right moves and 2 time units for down and left moves. Let also Agent 2 need 2 time units for up and right moves and 4 time units for down and left moves.

We consider a scenario where the robots have to eventually meet at yellow regions (global team task), and at the same time, they have to recharge within a certain time interval in recharge locations (blue squares with the circles in the respective color). The individual specifications are $\varphi_1 = \Diamond_{\leq 6}(\mathit{recharge1})$ and $\varphi_2 = \Diamond_{\leq 12}(\mathit{recharge2})$ stating that agent 1 has to recharge within 6 time units and agent 2 within 12 units, respectively, and the team task is $\varphi_G = \Diamond_{\leq 30}\{(\mathit{meet}_1^A \wedge \mathit{meet}_2^A) \vee (\mathit{meet}_1^B \wedge \mathit{meet}_2^B)\}$ stating that the agents have to meet either in the yellow region A or B within 30 time units.
An accepting collective timed run is

$$\widetilde{r}_G^t = ((\pi_4^1, \pi_{18}^2), 0)((\pi_{11}^1, \pi_{18}^2), 2) \dots ((\pi_9^1, \pi_{10}^2), 6)((\pi_{16}^1, \pi_3^2), 8) \dots \\ ((\pi_{13}^1, \pi_5^2), 13)((\pi_6^1, \pi_6^2), 14) \dots,$$

with corresponding timed word

$$w(\widetilde{r}_G^t) = (\emptyset, 0)(\emptyset, \pi_{18}^2), 2) \dots (\{\mathit{recharge1}\}, 6)(\{\mathit{recharge2}\}, 8) \dots \\ (\emptyset, \pi_5^2), 13)((\{\mathit{meet}_1^A, \mathit{meet}_2^A\}, 14) \dots,$$

which satisfies the formula ϕ_G. The run $\widetilde{r}_G^t$ can be projected onto the individual

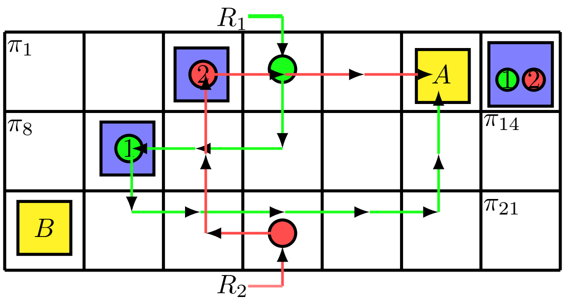

Figure 4.2: An illustrative example with 2 robots evolving in a common workspace. Let $\mathcal{W}_0 = \pi_1 \cup \ldots \cup \pi_{21}$. We enumerate the regions starting from the left region in every row and ending in the right. The initial positions of robots R_1, R_2 are depicted by a green and a red circle, respectively, the desired meeting points in yellow and the recharging spots by the agents' respective colors inside a blue box. The accepting runs for task specifications ϕ_1, ϕ_2, ϕ_G are depicted with green and red arrows for agent 1 and agent 2, respectively.

timed runs

$$\begin{aligned}\widetilde{r_1}^t &= (\pi_4^1, 0)(\pi_{11}^1, 2)(\pi_{10}^1, 4)(\pi_9^1, 6)(\pi_{16}^1, 8)(\pi_{17}^1, 9)(\pi_{18}^1, 10)\\ &\qquad (\pi_{19}^1, 11)(\pi_{20}^1, 12)(\pi_{13}^1, 13)(\pi_6^1, 14)\ldots,\\ \widetilde{r_2}^t &= (\pi_{18}^2, 0)(\pi_{17}^2, 4)(\pi_{10}^2, 6)(\pi_3^2, 8)(\pi_4^2, 10)(\pi_5^2, 12)(\pi_6^2, 14)\ldots\end{aligned}$$

(they are depicted in Figure 4.2 with green and red arrows, respectively) with corresponding timed words

$$\begin{aligned}w(\widetilde{r}_1^t) &= (\emptyset, 0)(\emptyset, 2)(\emptyset, 4)(\{\mathit{recharge1}\}, 6)(\emptyset, 8)\\ &\qquad (\emptyset, 9)(\emptyset, 10)(\emptyset, 11)(\emptyset, 12)(\emptyset, 13)(\{\mathit{meet}_1^A\}, 14)\ldots,\\ w(r_2^t) &= (\emptyset, 0)(\emptyset, 4)(\emptyset, 6)(\{\mathit{recharge2}\}, 8)(\emptyset, 10)(\emptyset, 12)(\{\mathit{meet}_2^A\}, 14)\ldots\end{aligned}$$

which satisfy formulas ϕ_1 and ϕ_2, respectively. All conditions from (4.3) are satisfied. The runs and the words of the illustrative example are depicted in Figure 4.3.

4.5 Conclusions

We have proposed a systematic method for multi-agent controller synthesis aiming cooperative planning under high-level specifications given in MITL formulas. The solution involves a sequence of algorithmic automata constructions such that not only team specifications but also individual specifications should be fulfilled.

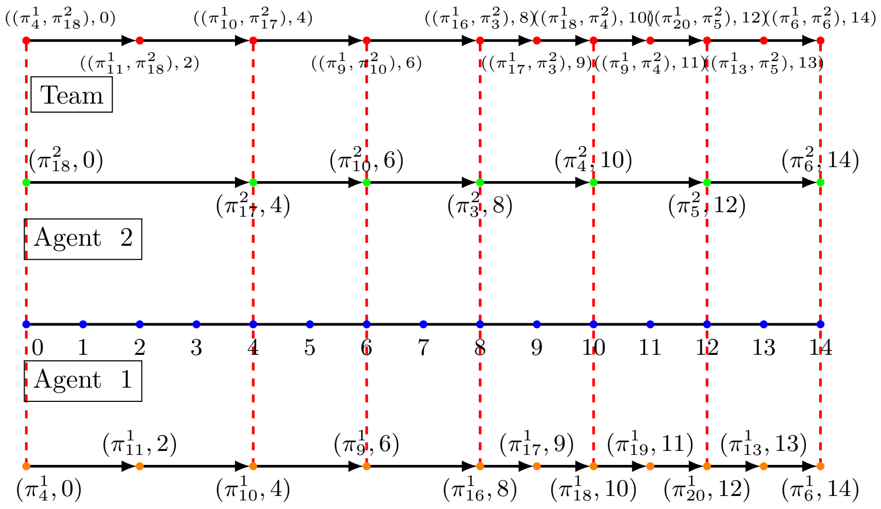

Figure 4.3: The accepting runs $\widetilde{r}^t_1, \widetilde{r}^t_2$, the collective run $\widetilde{r}^t_G$ and the corresponding timed stamps. We denote with red dashed lines the times that both agents have the same time stamps

Chapter 5

Decentralized Abstractions and Timed Constrained Planning

This chapter presents a fully automated procedure for controller synthesis for a general class of multi-agent systems under coupling constraints. Each agent is modeled with dynamics consisting of two terms: the first one models the coupling constraints and the other one is an additional bounded control input. We aim to design these inputs so that each agent meets an individual high-level specification given as a Metric Interval Temporal Logic (MITL). Furthermore, the connectivity of the initially connected agents, is required to be maintained. First, assuming a polyhedral partition of the workspace, a novel decentralized abstraction that provides controllers for each agent that guarantee the transition between different regions is designed. The controllers are the solution of a Robust Optimal Control Problem (ROCP) for each agent. Second, by utilizing techniques from formal verification, an algorithm that computes the individual runs which provably satisfy the high-level tasks is provided. Finally, simulation results conducted in MATLAB verify the performance of the proposed framework.

5.1 Introduction

In Chapter 4, an automata-based solution was proposed, where MITL formulas were introduced in order to synthesize controllers such that every agent fulfills an individual specification and the team of agents fulfills a global specification. Specifically, the abstraction of each agent's dynamics was considered to be given and an upper bound of the time that each agent needs to perform a transition from one region to another was assumed. Furthermore, potential coupled constraints between the agents were not taken into consideration. Motivated by this, in this chapter, we aim to address the aforementioned issues. We assume that the dynamics of each agent consists of two parts: the first part is a nonlinear function representing the coupling between the agent and its neighbors, and the second one is an additional control input which will be exploited for high-level planning. Hereafter, we call it a free input. A decentralized abstraction procedure is provided, which leads to an individual Weighted Transition System (WTS) for each agent and provides a basis for high-level planning.

Abstractions for both single and multi-agent systems have been provided e.g. in [111, 114–121]. In this chapter, we deal with the complete framework of both abstractions and controller synthesis of multi-agent systems. We start from the dynamics of each agent and we provide controllers that guarantee the transition between the regions of the workspace, while the initially connected agents remain connected for all times. The decentralized controllers are the solution of an ROCP. Then, each agent is assigned an individual task given as an MITL formulas. We aim to synthesize controllers, in discrete level, so that each agent performs the desired individual task within specific time bounds as imposed by the MITL formulas. In particular, we provide an automatic controller synthesis method of a general class of coupled multi-agent systems under high-level tasks with timed constraints. Compared to existing works on multi-agent planning under temporal logic specifications, the proposed approach considers dynamically coupled multi-agent systems under timed temporal specifications in a distributed way.

In our previous work [18], we treated a similar problem, but the under consideration dynamics were linear couplings and connectivity maintenance was not guaranteed by the proposed control scheme. Furthermore, the procedure was partially decentralized, due to the fact that a product Wighted Transition System (WTS) was required, which rendered the framework computationally intractable. To the best of the authors' knowledge, this is the first time that a fully automated framework for a general class of multi-agent systems consisting of both constructing purely decentralized abstractions and conducting timed temporal logic planning is considered.

This chapter is organized as follows. In Section 5.2 a description of the notations is given. Section 5.3 provides the modeling of the system and the formal problem statement. Section 5.4 discusses the technical details of the solution. Section 5.5 is devoted to a simulation example. Finally, conclusions and future work are discussed in Section 5.6.

5.2 Notation

In this chapter, the set-valued function $\mathcal{B} : \mathbb{R}^2 \times \mathbb{R}_{>0} \rightrightarrows \mathbb{R}^2$ given as $\mathcal{B}(c, \underline{r}) = \{x \in \mathbb{R}^2 : \|x - c\| \leq \underline{r}\}$ is the disk of center $c \in \mathbb{R}^2$ and $\underline{r} \in \mathbb{R}_{>0}$. The indexes i and j stand for agent i and its neighbors (see Section 5.3 for the definition of neighbors), respectively; $\mu, z \in \mathbb{N}$ are indexes used for sequences and sampling times, respectively. In the subsequent analysis a discrete partition of the workspace will be considered which is formalized through Definition 2.3. Hereafter, every member $S_\ell, \ell \in \mathbb{I}$ of a partition S, as is defined in 2.3 will be called *region*.

5.3 Problem Formulation

5.3.1 System Model

Consider a system of N agents, with $\mathcal{V} = \{1, \dots, N\}, N \geq 2$, operating in a workspace $W \subseteq \mathbb{R}^2$. The workspace is assumed to be closed and bounded. Let $x_i : \mathbb{R}_{\geq 0} \to W$ denotes the position of each agent in the workspace at time $t \in \mathbb{R}_{\geq 0}$. Each agent is equipped with a sensor device that can sense omni-directionally. Let the disk $\mathcal{B}(x_i(t), \underline{r})$ model the *sensing zone* of agent i at time $t \in \mathbb{R}_{\geq 0}$, where $\underline{r} \in \mathbb{R}_{\geq 0}$ is the sensing radius. The sensing radius is the same for all the agents. Let also $h > 0$ denote the *constant sampling period* of the system. We make the following assumption:

Assumption 5.1. (Measurements Assumption) It is assumed that each agent i, is able to measure its own position and all agents' positions that are located within agent's i sensing zone without any delays.

According to Assumption 5.1, the agent's i neighboring set at time t_0 is defined by $\mathcal{N}_i = \{j \in \mathcal{V} : x_j(t_0) \in \mathcal{B}(x_i(t_0), \underline{r})\}$. For the neighboring set $\mathcal{N}_i$ define also $N_i = |\mathcal{N}_i|$. Note that $i \in \mathcal{N}_j \Leftrightarrow j \in \mathcal{N}_i, \forall\, i, j \in \mathcal{V}, i \neq j$. The control design for every agent i should guarantee that it remains connected with all its neighbors $j \in \mathcal{N}_i$, for all times.

Consider the neighboring set $\mathcal{N}_i$. The coupled dynamics of each agent are given in the form:

$$\dot{x}_i = f(x_i, \bar{x}_i) + u_i,\ x_i \in W,\ i \in \mathcal{V}, \tag{5.1}$$

where $f : W \times W^{N_i} \to W$, is a nonlinear function representing the coupling between agent i and its neighbors $i_1, \dots, i_{N_i}$. The notation $\bar{x}_i = [x_{i_1}^\top, \dots, x_{i_{N_i}}^\top]^\top \in W^{N_i}$ is used for the vector of the neighbors of agent i, and $u_i : \mathbb{R}_{\geq 0} \to \mathbb{R}^2,\ i \in \mathcal{V}$ is the control input of each agent. The control inputs are assumed to be bounded by a positive constant $u_{\max}$. Hence,

$$u_i \in \mathcal{U}_i \triangleq \{u_i \in \mathbb{R}^2 : \|u_i\| \leq u_{\max}\}, i \in \mathcal{V}. \tag{5.2}$$

Assumption 5.2. The functions $f_i(x_i, \bar{x}_i), i \in \mathcal{V}$ are *Lipschitz continuous* in $W \times W^{N_i}$. Thus, there exists constants $L_i, \bar{L}_i > 0$ such that the following inequalities hold:

$$\|f_i(x_i, \bar{x}_i) - f_i(y_i, \bar{x}_i)\| \leq L_i \|x_i - y_i\|, \tag{5.3a}$$

$$\|f_i(x_i, \bar{x}_i) - f_i(x_i, \bar{y}_i)\| \leq \bar{L}_i \|\bar{x}_i - \bar{y}_i\|, \tag{5.3b}$$

for all $x_i, y_i \in W, \bar{x}_i, \bar{y}_i \in W^{N_i}, i \in \mathcal{V}$.

Remark 5.1. The coupling terms $f_i(x_i, \bar{x}_i), i \in \mathcal{V}$ are encountered in a large set of multi-agent protocols [58], including consensus, connectivity maintenance, collision avoidance and formation control. In addition, (5.1) may represent internal dynamics of the system as for instance in the case of smart buildings (see e.g., [122]) where the temperature $T_i, i \in \mathcal{V}$ of each room evolves according to the law

$$\dot{T}_i = \sum_{j \in \mathcal{N}_i} \alpha_{ij}(T_j - T_i) + u_i,$$

with α_{ij} representing the heat conductivity between rooms i and j and u_i the heating/cooling capabilities of the room.

5.3.2 Specification

Our goal is to control the multi-agent system (5.1) so that each agent obeys a given individual specification. In particular, it is required to drive each agent to a sequence of desired subsets of the *workspace* W within certain time limits and provide certain atomic tasks there. Atomic tasks are captured through a finite set of atomic propositions $\Sigma_i, i \in \mathcal{V}$, with $\Sigma_i \cap \Sigma_j = \emptyset$, for all $i, j \in \mathcal{V}, i \neq j$, which means that the agents do not share any atomic propositions. Each position x_i of each agent $i \in \mathcal{V}$ is labeled with atomic propositions that hold there. Initially, a labeling function

$$\Lambda_i : W \to 2^{\Sigma_i}, \tag{5.4}$$

is introduced for each agent $i \in \mathcal{V}$ which maps each state $x_i \in \mathbb{R}^2$ with the atomic propositions $\Lambda_i(x_i)$ which hold true at x_i i.e., the subset of atomic propositions that hold for agent i in position x_i. Define also by $\Lambda(x) = \bigcup\limits_{i \in \mathcal{V}} \Lambda_i(x)$ the union of all the labeling functions. Let us now introduce the following assumption which is important for defining the problem properly.

Assumption 5.3. There exists a partition $D = \{D_\ell\}_{\ell \in \mathbb{I}}$ of the workspace W which respects the labeling function Λ i.e., for all $D_\ell \in D$ it holds that $\Lambda(x) = \Lambda(x'), \forall\ x, x' \in D_\ell$. This assumption, intuitively, and without loss of generality, means that the same atomic propositions hold at all the points that belong to the same region of the partition.

Although the regions $D_\ell, \ell \in \mathbb{I}$ of the partition D may have different geometric shape, without loss of generality, we assume that they are hexagons with side length R. Define also for each agent i a labeling function:

$$L_i : D \to 2^{\Sigma_i}, \tag{5.5}$$

which maps every region of the partition D to the subset of the atomic propositions which hold true there. Furthermore, we assume that a time step $T > h > 0$ is given. This time step models the required time in which each agent should transit from a region to a neighboring region and is the same for all the agents.

The trajectory of each agent i is denoted by $x_i(t), t \geq 0, i \in \mathcal{V}$. The trajectory $x_i(t)$ is associated with a unique sequence:

$$r_{x_i}^t = (r_i(0), \tau_i(0))(r_i(1), \tau_i(1))(r_i(2), \tau_i(2)) \dots,$$

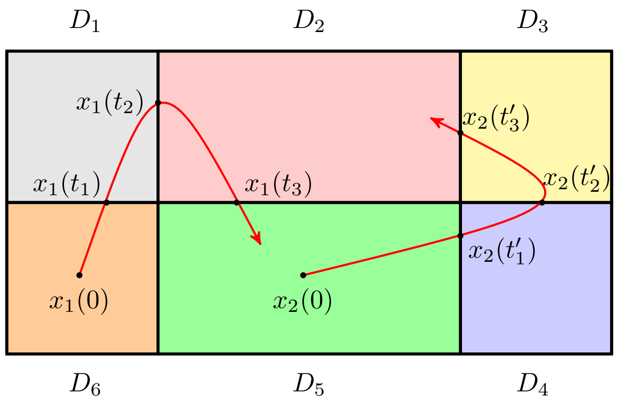

Figure 5.1: An example of two agents performing in a partitioned workspace.

of regions that the agent i crosses, where for all $\mu \geq 0$ it holds that: $x_i(\tau_i(\mu) \in r_i(\mu)$ and $\Lambda_i(x_i(t)) = L_i(r_i(\mu)), \forall\ t \in [\tau_i(\mu), \tau_i(\mu+1))$ for some $r_i(\mu) \in D$ and $r_i(\mu) \neq r_i(\mu+1)$. The timed word:

$$w^t_{x_i} = (L_i(r_i(0)), \tau_i(0))(L_i(r_i(1)), \tau_i(1))(L_i(r_i(2)), \tau_i(2)) \ldots,$$

where $w_i(\mu) = L_i(r_i(\mu)), \mu \geq 0, i \in \mathcal{V}$, is associated uniquely with the trajectory $x_i(t)$.

Definition 5.1. For each agent $i \in \mathcal{V}$ we define the *relaxed timed word* as:

$$\widetilde{w}^t_i = (w_i(0), \widetilde{\tau}_i(0))(w_i(1), \widetilde{\tau}_i(1))(w_i(2), \widetilde{\tau}_i(2)) \ldots, \tag{5.6}$$

where $w_i(\mu) = L_i(r_i(\mu)), \widetilde{\tau}_i(\mu) \in [\tau_i(\mu), \tau_i(\mu+1)), \forall\ \mu \geq 0$.

The time stamp $\tau_i(0) = \widetilde{\tau}_i(0) = t_0, i \in \mathcal{V}$ models the initial starting time of the agents. The time stamps $\tau_i(\mu), \mu \geq 1$ models the exact time in which the agent i crosses the boundary of the regions $r_i(\mu - 1)$ and $r_i(\mu)$. The time stamps $\widetilde{\tau}_i(\mu)$ model a time instant in which the agent i is in the region $r_i(\mu)$ of the workspace (see Example 5.1 below). The specification task φ_i given as an MITL formula over the set of atomic propositions Σ_i, represents desired tasks that are imposed to each agent $i \in \mathcal{V}$. We say that a trajectory $x_i(t)$ *satisfies* a formula φ_i given in MITL over the set Σ_i, and we formally write:

$$x_i(t) \models \varphi_i, \forall t \geq 0,$$

if and only if there exists a *relaxed timed word* $\widetilde{w}^t_i$ that complies with $x_i(t)$ and satisfies φ_i according to the semantics of MITL in 2.13.

Example 5.1. Consider $N = 2$ agents performing in the partitioned environment of Figure 5.1. Both agents have the ability to pick up, deliver and throw two different balls. Their sets of atomic propositions are $\Sigma_1 = \{pickUp1, deliver1, throw1\}$ and $\Sigma_2 = \{pickUp2, deliver2, throw2\}$, respectively, and satisfy $\Sigma_1 \cap \Sigma_2 = \emptyset$. Three points of the agents' trajectories that belong to different regions with different atomic propositions are captured. Assume that $t_1 < t'_1 < t_2 < t_2 < t'_2 < t_3 < t'_3$. The trajectories $x_1(t), x_2(t), t \geq 0$ are depicted with the red lines. According to Assumption 5.3, the partition $D = \{D_\ell\}_{\ell \in \mathbb{I}} = \{D_1, \ldots, D_6\}$ is given where $\mathbb{I} = \{1, \ldots, 6\}$ respects the labeling functions $\Lambda_i, L_i, i \in \{1, 2\}$. In particular, it holds that:

$$\begin{aligned}
&\Lambda_1(x_1(t)) = L_1(r_1(0)) = \{pickUp1\}, t \in [0, t_1),\\
&\Lambda_1(x_1(t)) = L_1(r_1(1)) = \{throw1\}, t \in [t_1, t_2),\\
&\Lambda_1(x_1(t)) = L_1(r_1(2)) = \{deliver1\}, t \in [t_2, t_3),\\
&\Lambda_1(x_1(t)) = L_1(r_1(3)) = \emptyset, t \geq t_3.\\
&\Lambda_2(x_2(t)) = L_2(r_2(0)) = \{pickUp2\}, t \in [0, t'_1),\\
&\Lambda_2(x_2(t)) = L_2(r_2(1)) = \{deliver2\}, t \in [t'_1, t'_2),\\
&\Lambda_2(x_2(t)) = L_2(r_2(2)) = \{throw2\}, t \in [t'_2, t'_3),\\
&\Lambda_2(x_2(t)) = L_2(r_2(3)) = \emptyset, t \geq t'_3.
\end{aligned}$$

By the fact that $w_i(\mu) = L(r_i(\mu)), \forall\ i \in \{1, 2\}, \mu \in \{1, 2, 3\}$, the corresponding individual timed words are given as:

$$\begin{aligned}
w^t_{x_1} &= (\{pickUp1\}, 0)(\{throw1\}, t_1)(\{deliver1\}, t_2)(\emptyset, t_3),\\
w^t_{x_2} &= (\{pickUp2\}, 0)(\{deliver2\}, t'_1)(\{throw2\}, t'_2)(\emptyset, t'_3).
\end{aligned}$$

According to (5.6), two relaxed timed words (depicted with red in Figure 5.1) are given as:

$$\begin{aligned}
w^t_1 &= (\{pickUp1\}, \widetilde{\tau}_1(0))(\{throw1\}, \widetilde{\tau}_1(1))(\{deliver1\}, \widetilde{\tau}_1(2))(\emptyset, \widetilde{\tau}_1(3)),\\
w^t_2 &= (\{pickUp2\}, \widetilde{\tau}_2(0))(\{deliver2\}, \widetilde{\tau}_2(1))(\{throw2\}, \widetilde{\tau}_2(2))(\emptyset, \widetilde{\tau}_2(3)).
\end{aligned}$$

The time stamps $\widetilde{\tau}_1(\mu), \widetilde{\tau}_2(\mu), \mu \in \{1, 2, 3\}$, should satisfy the following conditions:

$$\begin{aligned}
&\widetilde{\tau}_1(0) \in [\tau_1(0), \tau_1(1)) = [0, t_1),\\
&\widetilde{\tau}_1(1) \in [\tau_1(1), \tau_1(2)) = [t_1, t_2),\\
&\widetilde{\tau}_1(2) \in [\tau_1(2), \tau_1(3)) = [t_2, t_3),\\
&\widetilde{\tau}_1(3) \in [\tau_1(3), \cdot) = [t_3, \cdot),\\
&\widetilde{\tau}_2(0) \in [\tau_2(0), \tau_2(1)) = [0, t_1),\\
&\widetilde{\tau}_2(1) \in [\tau_2(1), \tau_2(2)) = [t_1, t_2),\\
&\widetilde{\tau}_2(2) \in [\tau_2(2), \tau_2(3)) = [t_2, t_3),\\
&\widetilde{\tau}_2(3) \in [\tau_2(3), \cdot) = [t_3, \cdot).
\end{aligned}$$

5.3.3 Problem Statement

We can now formulate the problem treated in this chapter as follows:

Problem 5.1. Given N agents operating in the bounded workspace $W \subseteq \mathbb{R}^2$, their initial positions $x_1(t_0), \dots, x_N(t_0)$, their dynamics as in (5.1), a time step $T > h > 0$, N task specification formulas $\varphi_1, \dots, \varphi_N$ expressed in MITL over the sets of services $\Sigma_1, \dots, \Sigma_N$, respectively, a partition of the workspace W into hexagonal regions $\{D_\ell\}_{\ell \in \mathbb{I}}$ with side length R, as given in Assumption 5.3 and the labeling functions $\Lambda_1, \dots, \Lambda_N, L_1, \dots, L_N$, as in (5.4), (5.5), assign control laws $u_1, \dots, u_N$ to each agent $1, \dots, N$, respectively, such that the connectivity between the agents that belong to the neighboring sets $\mathcal{N}_1, \dots, \mathcal{N}_N$ is maintained, as well as each agent fulfills its individual MITL specification $\varphi_1, \dots, \varphi_N$, respectively, i.e.,

$$x_1(t) \models \varphi_1, \dots, x_N(t) \models \varphi_N, \forall\, t \in \mathbb{R}_{\geq 0}.$$

Remark 5.2. The initial positions $x_1(t_0), \dots, x_N(t_0)$ should be such that the agents which are required to remain connected for all times need to satisfy the inequality $\|x_i(t_0) - x_{i'}(t_0)\| < \underline{r}, i, i' \in \mathcal{V}, i \neq i'$.

Remark 5.3. It should be noted that, in this work, the dependencies between the agents are induced through the coupled dynamics (5.1) and not in the discrete level, by allowing for couplings between the services (i.e., $\Sigma_i \cap \Sigma_j \neq \emptyset$, for some $i, j \in \mathcal{V}$). Hence, even though the agents do not share atomic propositions, the constraints on their motion due to the dynamic couplings and the connectivity maintenance specifications may restrict them to fulfill the desired high-level tasks. Treating additional couplings through individual atomic propositions in the discrete level is a topic of future work.

Remark 5.4. In Chapter 4, the multi-agent system was considered to have fully-actuated dynamics. The only constraints on the system were due to the presence of time constrained MITL formulas. In the current framework, we have two types of constraints: the constraints due to the coupling dynamics of the system (5.1), which constrain the motion of each agent, and, the timed constraints that are inherently imposed from the time bounds of the MITL formulas. Thus, there exist formulas that cannot be satisfied either due to the coupling constraints or the time constraints of the MITL formulas. These constraints, make the procedure of the controller synthesis in the discrete level substantially different and more elaborate than the corresponding multi-agent LTL frameworks in the literature ([89, 94, 95, 108]).

5.4 Main Results

In this section, a systematic solution to Problem 5.1 is introduced. Our overall approach builds on abstracting the system in (5.1) through a WTS for each agent and exploiting the fact that the timed runs in the i-th WTS project onto the

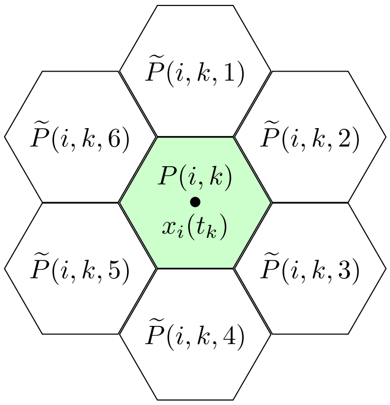

Figure 5.2: Illustration of agent i occupying region $P(i,k)$, depicted by green, at time $t_k = t_0 + kT$ with $\bar{P}(i,k) = \bigcup_{\widetilde{\ell} \in \mathbb{L}} \widetilde{P}(i,k,\widetilde{\ell})$ being the set of regions that the agent can transit at exactly time T.

trajectories of agent i while preserving the satisfaction of the individual MITL formulas $\varphi_i, i \in \mathcal{V}$. In particular, the following analysis is performed:

1. We propose a novel decentralized abstraction technique for the multi-agent system, i.e., discretization of the time into time steps T for the given partition $D = \{D_\ell\}_{\ell \in \mathbb{I}}$, such that the motion of each agent is modeled by a WTS $\mathcal{T}_i$, $i \in \mathcal{V}$ (Section 5.4.1). We adopt here the technique of designing Nonlinear Model Predictive Controllers (NMPC), for driving the agents between neighboring regions.

2. A three-step automated procedure for controller synthesis which serves as a solution to Problem 5.1 is provided in Section 5.4.2.

3. Finally, the computational complexity of the proposed approach is discussed in Section 5.4.3.

The next sections provide the proposed solution in detail.

5.4.1 Discrete System Abstraction

In this section we provide the abstraction technique that is designed in order to capture the dynamics of each agent into WTSs. Thereafter, we work completely at discrete level, which is necessary in order to solve Problem 5.1.

Workspace Geometry

Consider an enumeration $\mathbb{I}$ of the regions of the workspace, the index variable $\ell \in \mathbb{I}$ and the given time step T. The time step T models the time duration that each agent needs to transit between two neighboring regions of the workspace. Consider also a timed sequence:

$$\mathcal{S} = \{t_0, t_1 = t_0 + T, \dots, t_k = t_0 + kT, \dots\}, k \in \mathbb{N}. \tag{5.7}$$

S models the time stamps in which the agents are required to occupy different neighboring regions. For example, if at time t_k agent i occupies region D_ℓ, at the next time stamp $t_k + T$ is required to occupy a neighboring region of D_ℓ. The agents are always forced to change region for every different time stamp of the sequence $\mathcal{S}$. Let us define the mapping:

$$P : \mathcal{V} \times \mathbb{N} \to D,$$

which denotes the fact that the agent $i \in \mathcal{V}$, at time instant

$$t_k = t_0 + kT, k \in \mathbb{N},$$

occupies the region $D_{\ell_i} \in D$ for an index $\ell_i \in \mathbb{I}$. Define the mapping:

$$\widetilde{P} : \mathcal{V} \times \mathbb{N} \times \mathbb{L} \to D.$$

where $\mathbb{L} = \{1, \dots, 6\}$. By $\widetilde{P}(i, k, \widetilde{\ell}), \widetilde{\ell} \in \mathbb{L}$ we denote one and only one out of the six neighboring regions of region $P(i, k)$ that agent i occupies at time t_k. Define also by $\bar{P}(i, k)$ the union of all the six neighboring regions of region $P(i, k)$, i.e.,

$$\bar{P}(i, k) = \bigcup_{\widetilde{\ell} \in \mathbb{L}} \widetilde{P}(i, k, \widetilde{\ell}),$$

with $|\bar{P}(i, k)| = 6$. An example of agent i being at the region $P(i, k)$ along with its neighboring regions is depicted in Figure 5.2.

We start by giving a graphical example for the abstraction technique that will be adopted in this chapter. Consider an agent i occupying the green region $P(i, k) = D_{\ell_i}$ at time $t_k = t_0 + kT$ and let its neighbors j_1, j_2 occupying the red and blue regions $P(j_1, k) = D_{\ell_{j_1}}, P(j_2, k) = D_{\ell_{j_2}}$, respectively, as is depicted in Figure 5.3. The neighboring regions $\bar{P}(i, k)$, $\bar{P}(j_2, k)$ and $\widetilde{P}(j_1, k, \widetilde{\ell})$, $\widetilde{\ell} \in \{4, 5, 6\}$ for agent i, j_2, j_1, respectively, are also depicted. All the agents start their motion at time t_k simultaneously. Let $D_{\ell_{\text{des}}} \in \bar{P}(i, k)$ be a candidate neighboring region that agent i should transit to. The goal is to design a decentralized feedback control law $u_i(x_i, x_{j_1}, x_{j_2})$, that drives agent i in the neighboring region $D_{\ell_{\text{des}}}$ exactly at time T, *regardless of the transitions of its neighbors to their neighboring regions.* This procedure is repeated for all possible neighboring regions i.e., six times, and for all the agents. For the example of Figure 5.3, the procedure is performed 6^3 times (six times for each agent). By employing this procedure, we are able to: 1) synchronize the agents so that each of them knows at every time step T its position in the workspace as well as the region that occupies; 2) know which controller brings each agent in its desired region for any possible choice of controllers of its corresponding neighbors. We will hereafter present a formal approach of this procedure.

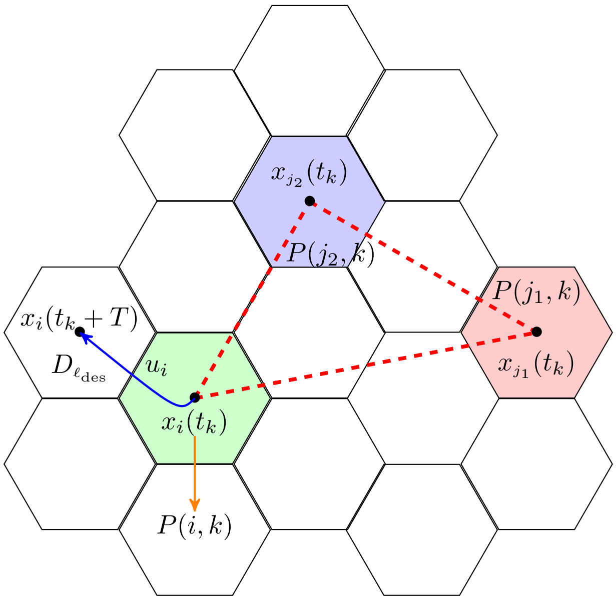

Figure 5.3: Illustration of three connected agents i, j_1, j_2. The agents are occupying the regions $P(i,k) = D_{\ell_i}$, $P(j_2,k) = D_{\ell_{j_1}}$ and $P(j_1,k) = D_{\ell_{j_2}}$ at time $t_k = t_0 + kT$, depicted by green, red and blue color, respectively. Their corresponding neighboring regions $\bar{P}(i,k), \bar{P}(j_1,k)$ and $\widetilde{P}(j_2,k,\widetilde{\ell}), \widetilde{\ell} \in \{4,5,6\}$, respectively, are also depicted; $\widetilde{P}(i,k,6) = D_{\ell_{\mathrm{des}}}$ is the desired region in which agent i needs to move at time T by applying a decentralized control law $u_i(x_i, x_{j_1}, x_{j_2})$.

Decentralized Controller Specification

Consider a time interval $[t_k, t_k + T]$. We state here the specifications that a decentralized feedback controller $u_i(x_i, \bar{x}_i)$ needs to guarantee so as agent i to have a *well-defined transition* between two neighboring regions within the time interval $[t_k, t_k + T]$.

(S1) The controller needs to take into consideration the dynamics (5.1) and the constraints that are imposed by (5.2).

(S2) Agent i should move from one region $P(i,k) \in D$ to a neighboring region $\widetilde{P}(i,k,\widetilde{\ell})$, without intersecting other regions, irrespectively of which region its neighbors are moving to. Thus, since the duration of the transition is T, it is required that $x_i(t_k) \in P(i,k)$, $x_i(t_k+T) \in \widetilde{P}(i,k,\widetilde{\ell})$ and $x_i(t) \in P(i,k) \cup \widetilde{P}(i,k,\widetilde{\ell})$, $\forall t \in (t_k, t_k + T)$. The neighbors of agent i will move also to exactly one of their corresponding neighboring regions.

Remark 5.5. The reason for imposing the aforementioned constraints is due to the need of imposing safety specifications to the agents. Thus, it is required that the agents will not cross more than one neighboring region within the duration of a transition T.

Error Dynamics

Let us define by $x_{i,k,\widetilde{\ell},\mathrm{des}} \in \widetilde{P}(i,k,\widetilde{\ell})$ the geometrical center of the desired region $\widetilde{P}(i,k,\ell)$ that agent i needs to occupy at time $t_k + T$. Define also by:

$$e_i(t) = x_i(t) - x_{i,k,\widetilde{\ell},\mathrm{des}}, t \in [t_k, t_k + T], \tag{5.8}$$

the error which the controller u_i needs to guarantee to become zero in the time interval $t \in [t_k, t_k + T]$. Then, the *nominal error dynamics* are given by:

$$\dot{e}_i(t) = g_i(e_i(t), \bar{x}_i(t), u_i(t)), t \in [t_k, t_k + T], \tag{5.9}$$

with initial condition $e_i(t_k) = x_i(t_k) - x_{i,k,\widetilde{\ell},\mathrm{des}}$, where:

$$g_i(e_i(t), \bar{x}_i(t), u_i(t)) \triangleq f_i(e_i(t) + x_{i,k,\widetilde{\ell},\mathrm{des}}, \bar{x}_i(t)) + u_i(t).$$

State Constraints

Before defining the ROCP we state here the state constraints that are imposed to the state of each agent. Define the set:

$$X_i = \\ \{x_i \in W, \bar{x}_i \in W^{N_i} : \|x_i - x_j\| < \underline{r}, \forall j \in \mathcal{N}_i(0), x_i \in P(i,k) \cup \widetilde{P}(i,k,\widetilde{\ell}), \widetilde{\ell} \in \mathbb{L}\},$$

as the set that captures the state constraints of agent i. The first constraint in the set X stands for the connectivity requirement of agent i with all its neighbors; the second one stands for the requirement each agent to transit from one region to exactly one desired neighboring region. In order to translate the constraints that are dictated for the state $x_i(t)$ into constraints regarding the error state $e_i(t)$ from (5.9), define the set $E_i = X_i \oplus (-x_{i,k,\widetilde{\ell},\mathrm{des}})$, where $\oplus$ stands for the Minkowski addition as defined in Definition 2.1. Then, the following implication holds: $x_i \in X_i \Rightarrow e_i \in E_i$.

Control Design

This subsection concerns the control design regarding the transition of agent i to one neighboring region $\widetilde{P}(i,k,\widetilde{\ell})$, for some $\widetilde{\ell} \in \mathbb{L}$. The abstraction design, however, concerns all the neighboring regions $\bar{P}(i,k)$, for which we will discuss in the next subsection.

The timed sequence $\mathcal{S}$ consists of intervals of duration T. Within every time interval $[t_k, t_k + T]$, each agent needs to be at time t_k in region $P(i,k)$ and at time $t_k + T$ in a neighboring region $\widetilde{P}(i,k,\widetilde{\ell}), \widetilde{\ell} \in \mathbb{L}$. We assume that T is related to the sampling time h according to: $T = mh, m \in \mathbb{N}$. Therefore, within the time interval $[t_k, t_k + T]$, there exists $m + 1$ sampling times. By introducing the notation $t_{k_z} \triangleq t_k + zh, \forall z \in \mathbb{M} \triangleq \{0, \dots, m\}$, we denote by $\{t_{k_z}\}_{z \in \mathbb{M}}$ the sampling sequence within the interval $[t_k, t_k + T]$. Note that $t_{k_0} = t_k$ and $t_{k_m} = t_k + T$. The indexes k, z stands for the interval and for the sampling times within this interval, respectively. As it will be presented hereafter, at every sampling time $t_{k_z}, z \in \mathbb{M}$, each agents solves a ROCP.

Our control design approach is based on Nonlinear Model Predictive Control (NMPC). NMPC has been proven to be efficient for systems with nonlinearities and state/input constraints. For details about NMPC we refer the reader to [123–132]. We propose here a sampled-data NMPC with decreasing horizon in order to design a controller that respects the desired specifications and guarantees the transition between regions at time T. In the proposed sampled-data NMPC, an open-loop Robust Optimal Control Problem (ROCP) is solved at every discrete sampling time instant $t_{k_z}, z \in \mathbb{M}$ based on the current error state information $e_i(t_{k_z})$. The solution is an optimal control signal $\hat{u}_i(t)$, for $t \in [t_{k_z}, t_{k_z} + T_z]$, where T_z is defined as follows.

Definition 5.2. A *decreasing horizon policy* is defined by:

$$T_z = T - zh, z \in \mathbb{M}. \tag{5.10}$$

This means that at every time sample t_{k_z} in which the ROCP is solved, the horizon is decreased by a sampling time h. The specific policy is adopted in order to enforce the controllers u_i to guarantee that agent i will reach the desired neighboring region at time T; (5.10) implies also that $t_{k_z} + T_z = t_k + T, \forall z \in \mathbb{M}$. A graphical illustration of the presented time sequences is given in Figure 5.4.

The *open-loop input signal* is applied in between the sampling instants and is given by the solution of the following Robust Optimal Control Problem (ROCP): $\mathcal{O}(k, x_i(t), \bar{x}_i(t), P(i,k), \widetilde{\ell}, x_{i,k,\widetilde{\ell},\text{des}}), t \in [t_{k_z}, t_{k_z} + T_z]$, which is defined as:

$$\min_{\hat{u}_i(\cdot)} J_i(e_i(t_{k_z})), \hat{u}_i(\cdot)) =$$

$$\min_{\hat{u}_i(\cdot)} \left\{ V_i(\hat{e}_i(t_{k_z} + T_z)) + \int_{t_{k_z}}^{t_{k_z}+T_z} \Big[F_i(\hat{e}_i(s), \hat{u}_i(s)) \Big] ds \right\} \tag{5.11a}$$

subject to:

$$\dot{\hat{e}}_i(s) = g_i(\hat{e}_i(s), \hat{\bar{x}}_i(s), \hat{u}_i(s)), \hat{e}_i(t_{k_z}) = e_i(t_{k_z}), \tag{5.11b}$$

$$\hat{e}_i(s) \in E^i_{s - t_{k_z}}, \hat{u}_i(s) \in \mathcal{U}_i, s \in [t_{k_z}, t_{k_z} + T_z], \tag{5.11c}$$

$$\hat{e}_i(t_{k_z} + T_z) \in \mathcal{E}_i. \tag{5.11d}$$

The ROCP has as inputs the terms $k, x_i(t)$, $\bar{x}_i(t)$, $P(i,k)$, $\widetilde{\ell}$, $x_{i,k,\widetilde{\ell},\text{des}}$, for time $t \in [t_{k_z}, t_{k_z} + T_z]$. We will explain hereafter all the terms appearing in the ROCP problem (5.11a)-(5.11d). By hat $\widehat{(\cdot)}$ we denote the predicted variables (internal to the controller), corresponding to the system (5.9) i.e., $\hat{e}_i(\cdot)$ is the solution of

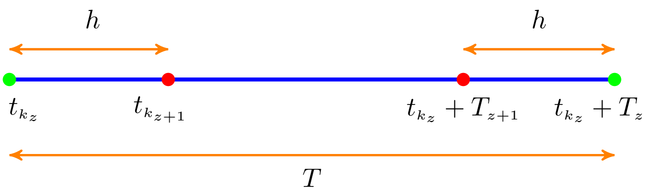

Figure 5.4: The prediction horizon of the ROCP along with the times $t_{k_z} < t_{k_{z+1}} < t_{k_z} + T_{z+1} < t_{k_z} + T_z$, with $t_{k_z} = t_k + zh$ and $T_z = T - zh, z \in \mathbb{M}$.

(5.11b) driven by the control input $\hat{u}_i(\cdot) : [t_{k_z}, t_{k_z} + T_z] \to \mathcal{U}_i$ with initial condition $\hat{e}_i(t_{k_z}) = e_i(t_{k_z})$. The set $E^i_{s-t_{k_z}}$ will be explicitly defined later.

Remark 5.6. In sampled-data NMPC bibliography an ROCP is defined over the time interval $s \in \{t_i, t_{i+1} = t_i + h, \ldots, t_i + T\}$, where T is the prediction horizon. Due to the fact that we have denoted by i the agents, and the fact that the ROCP is solved for every time interval, we use the notation $s \in \{t_{k_z} = t_k, t_{k_{z+1}} = t_k + h, \ldots, t_{k_z} + T_z = t_{k_z} + T\}$, instead. The indexes k, z stands for the interval and for the sampling time, respectively. A graphical illustration of the presented time sequence is given in Figure 5.4.

Remark 5.7. Note that the predicted values are not the same with the actual closed-loop values due to the fact that agent i, can not know the estimation of the trajectories of its neighbors $\hat{\tilde{x}}$, within a predicted horizon. Thus, the term $\hat{\tilde{x}}$ is treated as a disturbance to the *nominal system* (5.9).

The term $F_i : E_i \times \mathcal{U}_i \to \mathbb{R}_{\geq 0}$, stands for the *running cost*, and is chosen as:

$$F_i(e_i, u_i) = e_i^\top Q_i e_i + u_i^\top R_i u_i,$$

where $Q_i = \text{diag}\{q_{i_1}, q_{i_2}\}, R_i = \text{diag}\{\xi_{i_1}, \xi_{i_2}\}$, with $q_{i_\zeta} \in \mathbb{R}_{\geq 0}, \xi_{i_\zeta} \in \mathbb{R}_{>0}, \zeta \in \{1, 2\}$. For the running cost, it holds that $F_i(0, 0) = 0$, as well as:

$$\underline{m}_i \|e_i\|^2 \leq F_i(e_i, u_i) \leq \bar{m}_i \|e_i\|^2, \tag{5.12}$$

where $\underline{m}_i, \bar{m}_i$ will be defined later. Note that $\underline{m}_i \|e_i\|^2$ is $\mathcal{K}$ function, according to Definition 2.4.

Lemma 5.1. *The running cost function $F_i(e_i, u_i)$ is Lipschitz continuous in $E_i \times \mathcal{U}_i$, with Lipschitz constant:*

$$L_{F_i} = 2\bar{\varepsilon}_i \sigma_{\max}(Q_i),$$

where:

$$\bar{\varepsilon}_i = \sup_{e_i \in E_i} \{\|e_i\|\},$$

for all $e_i \in E_i, u_i \in \mathcal{U}_i$.

Proof. The proof can be found in Appendix B.1. □

The terms $V_i : E_i \to \mathbb{R}_{>0}$ and $\mathcal{E}_i \subseteq E_i$ are the *terminal penalty cost* and *terminal set*, respectively, and are used to enforce the stability of the system. The terminal cost is given by:

$$V_i(e_i) = e_i^\top P_i e_i.$$

where $P_i = \mathrm{diag}\{p_{i_1}, p_{i_2}\}$, with $p_{i_\zeta} \in \mathbb{R}_{>0}, \zeta \in \{1, 2\}$. We choose:

$$\begin{aligned} \underline{m}_i &= \min\{q_{i_1}, q_{i_2}, \xi_{i_1}, \xi_{i_2}\}, \\ \bar{m}_i &= \max\{q_{i_1}, q_{i_2}, \xi_{i_1}, \xi_{i_2}\}. \end{aligned} \tag{5.13}$$

The solution of the nominal model (5.9) at time $s \in [t_{k_z}, t_{k_z} + T_z]$, starting at time t_{k_z} from an initial condition $e_i(t_{k_z})$, applying a control input $u_i : [t_{k_z}, s] \to \mathcal{U}_i$ is denoted by:

$$e_i(s; u_i(\cdot), e_i(t_{k_z})), s \in [t_{k_z}, t_{k_z} + T_z].$$

The predicted state of the system (5.9) at time $s \in [t_{k_z}, t_{k_z} + T_z]$ is denoted by:

$$\hat{e}_i(s; u_i(\cdot), e_i(t_{k_z})), s \in [t_{k_z}, t_{k_z} + T_z],$$

and it is based on the measurement of the state $e_i(t_{k_z})$ at time t_{k_z}, when a control input $u_i(\cdot; e_i(t_{k_z}))$ is applied to the system (5.9) for the time period $[t_{k_z}, s]$. Thus, it holds that:

$$e_i(s) = \hat{e}_i(s; u_i(\cdot), e_i(s)), s \in [t_{k_z}, t_{k_z} + T_z]. \tag{5.14}$$

The state measurement enters the system via the initial condition of (5.11b) at the sampling instant, i.e. the system model used to predict the future system behavior is initialized by the actual system state. The solution of the ROCP (5.11a)-(5.11d) at time t_{k_z} provides an optimal control input denoted by $\hat{u}_i^\star(t; e(t_{k_z}))$, for $t \in [t_{k_z}, t_{k_z} + T_z]$. It defines the open-loop input that is applied to the system until the next sampling instant $t_{k_{z+1}}$:

$$u_i(t; e_i(t_i)) = \hat{u}_i^\star(t_{k_z}; e_i(t_{k_z})), t \in [t_{k_z}, t_{k_{z+1}}). \tag{5.15}$$

The corresponding *optimal value function* is given by:

$$J_i^\star(e_i(t_{k_z})) \triangleq J_i(e_i(t_{k_z}), \hat{u}_i^\star(\cdot; e_i(t_{k_z}))). \tag{5.16}$$

with $J_i(\cdot)$ as is given in (5.11a). The control input $u_i(t; e_i(t_{k_z}))$ is of the feedback form, since it is recalculated at each sampling instant using the new state information. Define an admissible control input as:

Definition 5.3. A control input $u_i : [t_{k_z}, t_{k_z} + T_z] \to \mathbb{R}^2$ for a state $e(t_{k_z})$ is called *admissible*, if the following hold:

1. $u_i(\cdot)$ is piecewise continuous;

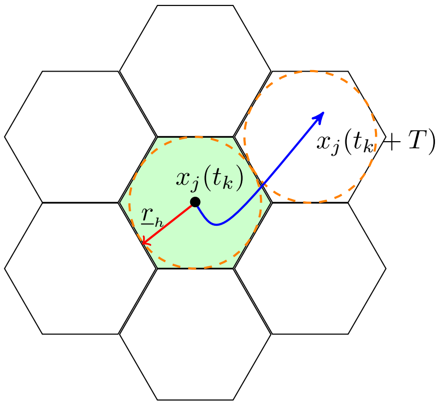

Figure 5.5: Illustration of agent j occupying region $P(j,k)$, depicted by green, at time $t_k = t_0 + kT$ along with the regions $\bar{P}(j,k)$. It is desired for agent j to move to region $\widetilde{P}(j,k,2)$ at exactly time T. The inscribed circle of regions $P(j,k), \widetilde{P}(j,k,2)$ are depicted with dashed orange color. The radius of the inscribed circle of the depicted hexagons is given by $\underline{r}_h = \frac{\sqrt{3}}{2}R$. By taking into consideration that each agent is moving at most to one neighboring region, according to the constraint set X_j, the following holds: $\sup\{\|x-y\| : x \in P(j,k), y \in \bar{P}(j,k)\} = 4\underline{r}_h = 2\sqrt{3}R$.

2. $u_i(s) \in \mathcal{U}_i, \forall\ s \in [t_{k_z}, t_{k_z} + T_z]$;

3. $e_i(s; u_i(\cdot), e(t_{k_z})) \in E_i, \forall\ s \in [t_{k_z}, t_{k_z} + T_z]$;

4. $e_i(T_z; u_i(\cdot), e(t_{k_z})) \in \mathcal{E}_i$;

Property 5.1. For the given hexagonal regions with side length R, the radius of the inscribed circle is given by $\underline{r}_h = \frac{\sqrt{3}}{2}R$ (two inscribed circles for the given regions are depicted with orange in Figure 5.5). Thus, according to Figure 5.5, an upper bound of the norm of differences between the actual position x_j and the estimated position $\hat{x}_j$ of the agent's i neighbors states, is given by:

$$\|x_j - \hat{x}_j\| \leq 2\underline{r}_h = \sqrt{3}R, j \in \mathcal{N}_i, \tag{5.17}$$

due to the fact that each agent can transit at most to a neighboring region, according to the constraint set X_i.

Lemma 5.2. *In view of Assumption 5.2, the difference between the actual measurement $e_i(s) = e_i(s; u_i(s; e_i(t_{k_z})), e_i(t_{k_z}))$ at time $s \in [t_{k_z}, t_{k_z} + T_z]$ and the predicted state $\hat{e}_i(s; u_i(s; e_i(t_{k_z})), e_i(t_{k_z}))$ at the same time under the same control law $u_i(s; e_i(t_{k_z}))$, starting at the same initial state $e_i(t_{k_z})$, is upper bounded by:*

$$\|e_i(s) - \hat{e}_i(s; u_i(s; e_i(t_{k_z})), e_i(t_{k_z}))\| \leq \rho_i(s - t_{k_z}), \tag{5.18}$$

where $\rho_i : \mathbb{R}_{\geq 0} \to \mathbb{R}$, *with:*

$$\rho_i(y) = \widetilde{\rho}_i \left[e^{L_i y} - 1 \right], \tag{5.19}$$

and

$$\widetilde{\rho}_i = \frac{R\sqrt{3N_i}\bar{L}_i}{L_i}. \tag{5.20}$$

Proof. The proof can be found in Appendix B.2. □

The satisfaction of the constraint on the state along the prediction horizon depends on the future evolution of the neighboring agents trajectories. Under Assumptions (5.2) of Lipschitz continuity and bounds of the nominal model, respectively, it is possible to compute a bound on the future effect of the disturbance on the system as is given by Lemma 5.2. Then, by considering this effect on the state constraint on the nominal prediction, it is possible to guarantee that the evolution of the real state of the system will be admissible all the time. In view of latter, the state constraint set E of the standard NMPC formulation, is being replaced by a restricted constrained set $E^i_{s-t_{k_z}} \subseteq E_i$ in (5.11c). This state constraint's tightening for the nominal system (5.9) is a key ingredient of the robust NMPC controller and guarantees that the evolution of the real system will be admissible. Authors in [132, 133] has considered such a Robust NMPC formulation. The restricted constrained set is then defined as $E^i_{s-t_{k_z}} = E_i \sim B^i_{s-t_{k_z}}$, where:

$$B^i_{s-t_{k_z}} = \left\{ e_i \in \mathbb{R}^2 : \|e_i(s)\| \leq \rho_i(s - t_{k_z}) \right\}, s \in [t_{k_z}, t_{k_z} + T_z],$$

as ρ_i given in 5.2.

Property 5.2. For every $s \in [t_{k_z}, t_{k_z} + T_z]$, we have that if

$$\hat{e}_i(s; u_i(s; e(t_{k_z})), e_i(t_{k_z})) \in E^i_{s-t_{k_z}} = E_i \sim B^i_{s-t_{k_z}} \subseteq E_i,$$

then the real state satisfies the constraint E_i, i.e., $e_i(s) \in E_i$.

Proof. The proof can be found in Appendix B.3. □

For the feasibility and convergence proofs of the ROCP the following assumptions are required.

Assumption 5.4. Assume that there exists a *local stabilizing controller* $u_{f,i} = \kappa_i(e_i) \in \mathcal{U}_i$ satisfying:

$$\frac{\partial V_i}{\partial e_i} \left[g_i(e_i, \bar{x}_i, \kappa_i(e_i)) \right] + F_i(e_i, \kappa_i(e_i)) \leq 0, \forall\ e_i \in \Phi_i, \tag{5.21}$$

where Φ_i is a set given by:

$$\Phi_i \triangleq \{ e_i \in \mathbb{R}^2 : V_i(e_i) \leq \alpha_{1,i} \}, \alpha_{1,i} > 0,$$

such than:

$$\Phi_i \subseteq \mathbb{E}_i \triangleq \{e_i \in E^i_{T_z} : \kappa_i(e_i) \in \mathcal{U}_i\},$$

where $E^i_{T_z} = E_i \sim B^i_{T_z}$.

Lemma 5.3. *The terminal penalty function $V_i(\cdot)$ is Lipschitz in Φ_i, with Lipschitz constant $L_{V,i} = 2\sigma_{\max}(P_i)\alpha_{1,i}$, for all $e_i(t) \in \Phi_i$.*

Proof. The proof can be found in Appendix B.4. □

Once the set Φ_i is computed, the terminal constraint set $\mathcal{E}_i$ is given by the following. Supposing that Assumption 5.4 holds. Then, by choosing:

$$\mathcal{E}_i = \{e_i \in \mathbb{R}^2 : V_i(e_i) \leq \alpha_{i,2}\}, \text{ with } \alpha_{2,i} \in (0, \alpha_{1,i}), \tag{5.22}$$

we guarantee the following: 1) $\mathcal{E}_i \subseteq \widetilde{P}(i,k,\widetilde{\ell})$, i.e. the terminal set is a subset of the desired neighboring region; 2) for all $e_i \in \Phi_i$ it holds that $g_i(e_i, \kappa_i(e_i)) \in \mathcal{E}_i$.

The following two lemmas are required in order to prove the basic Theorem or this paper.

Lemma 5.4. *Let $s \geq t_{k_{z+1}}$, $x \in E^i_{s-t_{k_z}}$ and $y \in \mathbb{R}^2$ such that: $\|x-y\| \leq \rho_i(t_{k_{z+1}} - t_{k_z}) = \rho_i(h)$, as ρ_i is given in Lemma 5.2. Then, it holds that $y \in E^i_{s-t_{k_{z+1}}}$.*

Proof. The proof can be found in Appendix B.5. □

Lemma 5.5. *Let $s \geq t_{k_z}$. The difference between two estimated trajectories $\hat{e}_i(s;$ $u_i(\cdot)$, $e_i(t_{k_{z+1}}))$, $\hat{e}_i(s;$ $u_i(\cdot)$, $e_i(t_{k_z}))$ at time s, starting from from initial points $t_{k_{z+1}}$, t_{k_z}, respectively, under the same control input $u_i(\cdot)$, is upper bounded by:*

$$\|\hat{e}_i(s; u_i(\cdot), e_i(t_{k_{z+1}})) - \hat{e}_i(s; u_i(\cdot), e_i(t_{k_z}))\| \leq \rho_i(t_{k_{z+1}} - t_{k_z}) = \rho_i(h). \tag{5.23}$$

Proof. The proof can be found in Appendix B.6. □

Theorem 5.1. *Suppose that Assumptions 5.1-5.4 hold. If the ROCP is feasible at time t_k, then, the closed loop system* (5.9) *of agent i, under the control input* (5.15)*, starting its motion at time $t_k = t_0 + kT$ from region $P(i,k)$, is Input to State Stable (ISS) (for ISS see [27]) and its trajectory converges to the admissible positively invariant terminal set $\mathcal{E}_i$ exactly at time $t_k + T$, if it holds that*

$$\rho_i(T_z) \leq \bar{\rho}_i \triangleq \frac{\alpha_{1,1} - \alpha_{2,i}}{L_{V_i}}. \tag{5.24}$$

Proof. The proof consists of two parts: in the first part it is established that initial feasibility implies feasibility afterwards. Based on this result it is then shown that the error $e_i(t)$ converges to the terminal set $\mathcal{E}_i$. The *feasibility analysis* as well as the *convergence analysis* can be found in Appendix B.7. □

Assumption 5.4 is common in the NMPC literature. Many methodologies on how to compute Φ_i and controllers $u_{f,i} = \kappa_i(e_i)$, if they exist, have been proposed. We refer the reader to [126, 134]. Regarding the initial feasibility, numerical tools (e.g. [128]) can be utilized in order to solve the ROCP and check if the problem is feasible at time t_k.

Remark 5.8. The term $\bar{\rho}_i, i \in \mathcal{V}$ gives an upper bound on the deviation of the trajectories of the neighboring agents of agent i from their real values. If this bound is satisfied, agent i can transit between the corresponding two neighboring regions, provided the ROCP is feasible at t_{k_z}.

Remark 5.9. It should be noted that, due to the nonlinear coupling terms $f_i(x_i, \bar{x}_i)$ and the desired connectivity specifications, some of the ROCPs for $k \in N$ might not have a feasible solution. Let $i' \in \mathcal{V}, k' \in \mathbb{N}, \widetilde{\ell}' \in \mathbb{L}$ represent an agent i' that at time step $t_{k'} = t_0 + k'T$ is desired to transit from region $P(i', k')$ to region $\widetilde{P}(i', k', \widetilde{\ell}')$. If the ROCP

$$\mathcal{O}(k', x_{i'}(t), \bar{x}_{i'}(t), P(i', k'), \widetilde{\ell}', x_{i',k',\widetilde{\ell}',\text{des}}), t \in [t_{k'_z}, t_{k'_z} + T_z],$$

has no solution, then there does not exist admissible controller that can drive agent i' from $P(i', k')$ to region $\widetilde{P}(i', k', \widetilde{\ell}')$. Our goal, through the proposed approach, is to seek all the possible solutions of the ROCP, which implies to seek for all possible transitions that will form later the individual WTS $\mathcal{T}_i$ of each agent. In this way, the resulting WTS $\mathcal{T}_i$ will capture the coupling dynamics (5.1) and the transition possibilities of agent i in the best possible way.

Generating the WTSs

Each agent $i \in \mathcal{V}$ solves the ROCP (5.11a)-(5.11d) for every time interval $[t_{k_z}, t_{k_z} + T_z], k \in \mathbb{N}$, for all the desired neighboring regions $\widetilde{P}(i, k, \widetilde{\ell}), \widetilde{\ell} \in \mathbb{L}$. This procedure is performed by off-line simulation, i.e., at each sampling time $t_{k_z}, z \in \mathbb{M}$, each agent exchanges information about its new state with its neighbors and simulates the dynamics (5.9). Between the sampling times the estimation $\hat{\bar{x}}_i$ is considered to be a disturbance, as discussed earlier.

Algorithm 1 provides the off-line procedure in order to generate the transition relation for each agent. At time t_0 each agent i calls the algorithm in order to compute all possible admissible controllers to all possible neighboring regions of the workspace. The term Transit, which is the output of the algorithm, is a matrix of control input sequences for all pairs of neighboring regions in the workspace, initialized at sequences of zeros. The function Point2Region$(\cdot)$ maps the point $x_i(t_k)$ to the corresponding region of the workspace. The function Sampling$(\cdot)$ takes as input the interval $[t_k, t_k + T]$ and returns the $m+1$ samples of this interval. The notation $(u_i^\star)_{k_z}$ stands for the z-th element of the vector $(u_i^\star)$. The function OptSolve$(k, x_i(t), \bar{x}_i(t), p, \widetilde{\ell})$ (i) solves the ROCP and the function UpdateStates$(x_i, \bar{x}_i)$ updates the states of agent i and its neighbors after every sampling time. If the OptSolve function does

Algorithm 1 CreateTransitionRelation($\cdot$)

```
 1: Input: $i, x_i(t_0), \bar{x}_i(t_0)$;
 2: Output: Transit;                          ▷ Matrix with regions\control inputs;
 3:
 4: Transit ← zeros($|\mathbb{I}|, 6$); k = 0; Flag = False;
 5: List ← {Point2Region($x_i(t_0)$)};                            ▷ Initialize;
 6: while List $\neq \emptyset$ do
 7:     for $p \in$ List do                         ▷ p is a region of the List;
 8:         for $\widetilde{\ell} \in \mathbb{L}$ do
 9:             $t \leftarrow$ Sampling($t_k, t_k + T$);
10:             for $t_{k_z} \in t, z \in \mathbb{M}$ do
11:                 $(u_i^\star)_{k_z} \leftarrow$ OptSolve($k, x_i(t), \bar{x}_i(t), p, \widetilde{\ell}$);
12:                 UpdateStates($x_i, \bar{x}_i$);
13:                 if $(u_i^\star)_{k_z} = \emptyset$ then                 ▷ $\nexists$ controller;
14:                     Flag = True;                      ▷ search next region;
15:                     break;
16:                 end if
17:             end for
18:             if Flag = False then
19:                 $u_i^\star \leftarrow \{(u_i^\star)_{k_z}\}_{z \in \mathbb{M}}$                        ▷ $u_i$ found;
20:                 Transit($p, \widetilde{\ell}$) ← $u_i^\star$;
21:                 List ← List $\cup \widetilde{P}(i, k, \widetilde{\ell})$
22:             else
23:                 Flag = False;
24:             end if
25:         end for
26:         List ← List$\backslash p$;
27:         $k = k + 1$;
28:     end for
29: end while
```

not return a solution, then there does not exist an admissible control input that can drive agent i to the desired neighboring region. After utilizing Algorithm 1, the WTS of each agent is defined as follows:

Definition 5.4. The motion of each agent $i \in \mathcal{V}$ in the workspace is modeled by the WTS $\mathcal{T}_i = (S_i, S_i^{\text{init}}, Act_i, \longrightarrow_i, d_i, \Sigma_i, L_i)$ where: $S_i = \{D_\ell\}_{\ell \in \mathbb{I}}$ is the set of states of each agent; $S_i^{\text{init}} = P(i, 0) \subseteq S_i$ is a set of initial states defined by the agents' initial positions $x_i(t_0) \in P(i, 0)$ in the workspace; Act_i is the set of actions containing the union of all the admissible control inputs $u_i \in \mathcal{U}_i$ that are a feasible solution to the ROCP and can drive agent i between neighboring regions; $\longrightarrow_i \subseteq S_i \times Act_i \times S_i$ is the transition relation. We say that $(P(i,k), u_i, \widetilde{P}(i,k,\widetilde{\ell})) \in \longrightarrow_i$, $k \in \mathbb{N}, \widetilde{\ell} \in \mathbb{L}$ if there exist an admissible controller $u_i \in Act_i$ which at step k drives the agent i from the region $P(i,k)$ to the desired region $\widetilde{P}(i,k,\widetilde{\ell})$. Algorithm 1 gives the steps how the transition relation can be constructed. $d_i : \longrightarrow_i \to \mathbb{R}_{\geq 0}$, is a map that assigns a positive weight (duration) to each transition. The duration of each transition is exactly equal to T; Σ_i, is the set of atomic propositions; $L_i : S_i \to 2^{\Sigma_i}$, is the labeling function.

The individual WTSs of the agents will allow us to work directly in the discrete level and design sequences of controllers that solve Problem 5.1. Every WTS $\mathcal{T}_i, i \in \mathcal{V}$ generates timed runs and timed words of the form $r_i^t = (r_i(0), \tau_i(0))\ (r_i(1), \tau_i(1)) \ldots$, $w_i^t = (w_i(0), \tau_i(0))\ (w_i(1), \tau_i(1)) \ldots$, respectively, over the set 2^{Σ_i} with $w_i(\mu) = L_i(r_i(\mu)), \tau_i(\mu) = \mu T, \forall\ \mu \geq 0$. The transition relation $\longrightarrow_i$ along with the output of the Algorithm 1, i.e, Transit$(\cdot)$, allows each agent to have all the necessary information in order to be able to make a decentralized plan in the discrete level that is presented hereafter. The relation between the timed words that are generated by the WTSs $\mathcal{T}_i, i \in \mathcal{V}$ with the timed service words produced by the trajectories $x_i(t), i \in \mathcal{V}, t \geq 0$ is provided through the following remark:

Remark 5.10. By construction, each timed word produced by the WTS $\mathcal{T}_i$ is a *relaxed timed word* associated with the trajectory $x_i(t)$ of the system (5.1). Hence, if we find a timed word of $\mathcal{T}_i$ satisfying a formula φ_i given in MITL, we also find for each agent i a desired timed word of the original system, and hence trajectories $x_i(t)$ that are a solution to the Problem 5.1. Therefore, the produced timed words of $\mathcal{T}_i$ are compliant with the relaxed timed words of the trajectories $x_i(t)$.

5.4.2 Controller Synthesis

The proposed controller synthesis procedure is described with the following steps:

1. N TBAs $\mathcal{A}_i,\ i \in \mathcal{V}$ that accept all the timed runs satisfying the corresponding specification formulas $\varphi_i, i \in \mathcal{V}$ are constructed.

2. A Büchi WTS $\widetilde{\mathcal{T}}_i = \mathcal{T}_i \otimes \mathcal{A}_i$ (see Definition 5.5 below) is constructed for every $i \in \mathcal{V}$. The accepting runs of $\widetilde{\mathcal{T}}_i$ are the individual runs of $\mathcal{T}_i$ that satisfy the corresponding MITL formula $\varphi_i,\ i \in \mathcal{V}$.

3. The abstraction procedure allows to find an explicit feedback law for each transition in $\mathcal{T}_i$. Therefore, an accepting run $\widetilde{r}_i^t$ in $\mathcal{T}_i$ that takes the form of a sequence of transitions is realized in the system in (5.1) via the corresponding sequence of feedback laws.

Definition 5.5. Given a WTS $\mathcal{T}_i = (S_i, S_i^{\text{init}}, Act_i, \longrightarrow_i, d_i, \Sigma_i, L_i)$, and a TBA $\mathcal{A}_i = (Q_i, Q_i^{\text{init}}, C_i, Inv_i, E_i, F_i, \Sigma_i, \mathcal{L}_i)$ with $|C_i|$ clocks and let $C_i^{\max}$ be the largest constant appearing in $\mathcal{A}_i$. Then, we define their *Büchi WTS* $\widetilde{\mathcal{T}}_i = \mathcal{T}_i \otimes \mathcal{A}_i = (\widetilde{S}_i, \widetilde{S}_i^{\text{init}}, \widetilde{Act}_i, \rightsquigarrow_i, \widetilde{d}_i, \widetilde{F}_i, \Sigma_i, \widetilde{L}_i)$ as follows:

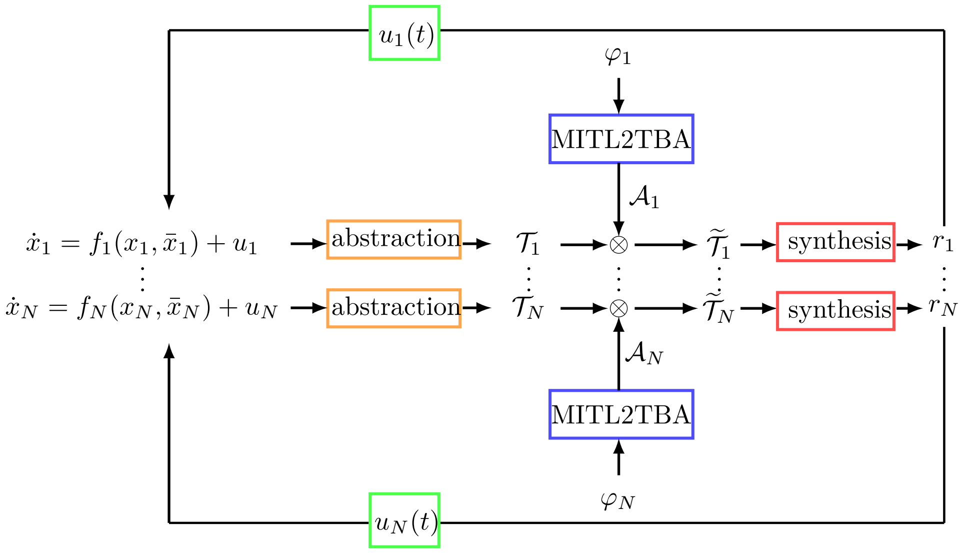

Figure 5.6: A graphical illustration of the proposed framework.

- $\widetilde{S}_i \subseteq \{(s_i, q_i) \in S_i \times Q_i : L_i(s_i) = \mathcal{L}_i(q_i)\} \times \mathbb{T}_{\infty}^{|C_i|}$.
- $\widetilde{S}_i^{\text{init}} = S_i^{\text{init}} \times Q_i^{\text{init}} \times \{0\}^{|C_i|}$.
- $\widetilde{Act}_i = Act_i$.
- $(\widetilde{q}, act_i, \widetilde{q}') \in \rightsquigarrow_i$ iff
  - $\widetilde{q} = (s, q, \nu_1, \ldots, \nu_{|C_i|}) \in \widetilde{S}_i$,
    $\widetilde{q}' = (s', q', \nu_1', \ldots, \nu_{|C_i|}') \in \widetilde{S}_i$,
  - $act_i \in Act_i$,
  - $(s, act_i, s') \in \longrightarrow_i$, and
  - there exists γ, R, such that $(q, \gamma, R, q') \in E_i$, $\nu_1,\ldots,\nu_{|C_i|} \models \gamma$, $\nu_1',\ldots,\nu_{|C_i|}' \models Inv_i(q')$, and for all $i \in \{1, \ldots, |C_i|\}$

$$\nu_i' = \begin{cases} 0, & \text{if } c_i \in R, \\ \nu_i + d_i(s, s'), & \text{if } c_i \notin R \text{ and} \\ & \nu_i + d_i(s, s') \leq C_i^{max}, \\ \infty, & \text{otherwise.} \end{cases}$$

  Then, $\widetilde{d}_i(\widetilde{q}, \widetilde{q}') = d_i(s, s')$.
- $\widetilde{F}_i = \{(s_i, q_i, \nu_1, \ldots, \nu_{|C_i|}) \in Q_i : q_i \in F_i\}$.

- $\widetilde{L}_i(s_i, q_i, \nu_1, \ldots, \nu_{|C_i|}) = L_i(s_i)$.

Each Büchi WTS $\widetilde{\mathcal{T}}_i, i \in \mathcal{V}$ is in fact a WTS with a Büchi acceptance condition $\widetilde{F}_i$. A timed run of $\widetilde{T}_i$ can be written as $\widetilde{r}_i^t = (q_i(0), \tau_i(0))(q_i(1), \tau_i(1)) \ldots$ using the terminology of Definition 2.11. It is *accepting* if $q_i(\mu) \in \widetilde{F}_i$ for infinitely many $j \geq 0$. An accepting timed run of $\widetilde{\mathcal{T}}_i$ projects onto a timed run of $\mathcal{T}_i$ that satisfies the local specification formula φ_i by construction.

Lemma 5.6. *Consider an accepting timed run $\widetilde{r}_i^t = (q_i(0), \tau_i(0))(q_i(1), \tau_i(1)) \ldots$ of the Büchi WTS $\widetilde{T}_i$ defined above, where $q_i(\mu) = (r_i(\mu), s_i(\mu), \nu_{i,1}, \ldots, \nu_{i,|C_i|})$ denotes a state of $\widetilde{\mathcal{T}}_i$, for all $\mu \geq 0$. The timed run $\widetilde{r}_i^t$ projects onto the timed run $r_i^t = (r_i(0), \tau_i(0))(r_i(1), \tau_i(1)) \ldots$ of the WTS $\mathcal{T}_i$ that produces the timed word $w(r_i^t) = (L_i(r_i(0)), \tau_i(0))(L_i(r_i(1)), \tau_i(1)) \ldots$ accepted by the TBA $\mathcal{A}_i$ via its run $\chi_i = s_i(0)s_i(1) \ldots$. Vice versa, if there exists a timed run $r_i^t = (r_i(0), \tau_i(0))(r_i(1), \tau_i(1)) \ldots$ of the WTS T_i that produces a timed word $w(r_i^t) = (L_i(r_i(0)), \tau_i(0))(L_i(r_i(1)), \tau_i(1)) \ldots$ accepted by the TBA A_i via its run $\chi_i = s_i(0)s_i(1) \ldots$ then there exist the accepting timed run $\widetilde{r}_i^t = (q_i(0), \tau_i(0))(q_i(1), \tau_i(1)) \ldots$ of $\widetilde{T}_i$, where $q_i(z)$ denotes $(r_i(z), s_i(z), \nu_{i,1}, \ldots, \nu_{i,|C_i|})$ in $\widetilde{T}_i$.*

The proposed framework is depicted in Figure 5.6. The dynamics (5.1) of each agent i is abstracted into a WTS $\mathcal{T}_i$ (orange rectangles). Then the product between each WTS $\mathcal{T}_i$ and the *TBA* $\mathcal{A}_i$ is computed according to Definition 5.5. The TBA $\mathcal{A}_i$ accepts all the words that satisfy the formula φ_i (blue rectangles). For every Büchi WTS $\widetilde{\mathcal{T}}_i$ the controller synthesis procedure that was described in this Section (red rectangles) is performed and a sequence of accepted runs $\{\widetilde{r}_1^t, \ldots, \widetilde{r}_N^t\}$ is designed. Every accepted run $\widetilde{r}_i^t$ maps into a decentralized controller $u_i(t)$ which is a solution to Problem 5.1.

Proposition 1. The solution that we obtain from Steps 1-5, if one found, gives a sequence of controllers $u_1, \ldots, u_N$ that guarantees the satisfaction of the formulas formulas $\varphi_1, \ldots, \varphi_N$ of the agents $1, \ldots, N$ respectively, governed by dynamics as in (5.1). Thus, we solved Problem 5.1.

5.4.3 Complexity

In the proposed abstraction technique 6^N MPC optimization problems are solved for every time interval $t \in [t_k, t_k + T]$. Assume that the desired horizon for the system to run is M steps i.e. the timed sequence $\mathcal{S}$ is written as: $\mathcal{S} = \{t_0, t_1 = t_0 + T, \ldots, t_M = t_0 + MT\}$. Then the complexity of the abstraction is $M6^N$. As for the controller synthesis framework now we have the following. Denote by $|\varphi|$ the length of an MITL formula φ. A TBA $\mathcal{A}_i, i \in \mathcal{V}$ can be constructed in space and time $2^{\mathbb{O}(|\varphi_i)|}, i \in \mathcal{V}$ ($\mathbb{O}$ stands for the "big O" from complexity theory). Let $\varphi_{\max} = \max\{|\varphi_i\}, i \in \mathcal{V}$ be the MITL formula with the longest length. Then, the complexity of Step 1 is $2^{\mathbb{O}(|\varphi_{\max})|}$. The model checking of Step 2 costs $\mathbb{O}(|\mathcal{T}_i|2^{|\varphi_i|}), i \in \mathcal{V}$ where $|\mathcal{T}_i|$ is the length of the

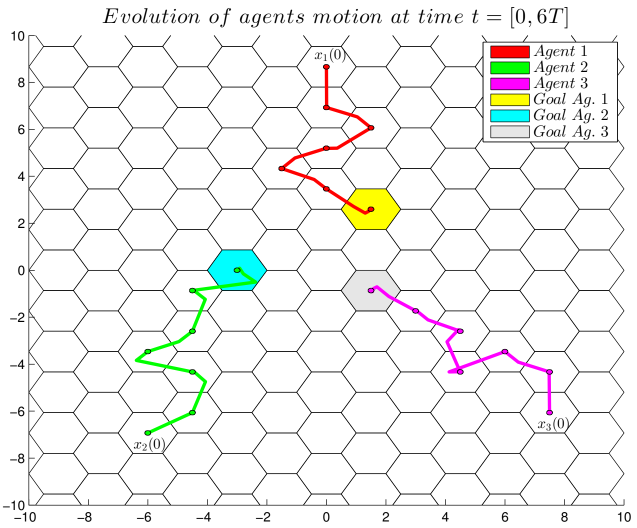

Figure 5.7: Evolution of the agents' trajectories up to time $6T$ in the workspace W. Each point-to-point transition has time duration $T = 3$. The depicted timed runs with red, green and magenta, of agents 1, 2 and 3, satisfy the formulas φ_1, φ_2 and φ_3, respectively, while the agents remain connected.

WTS $\mathcal{T}_i$ i.e., the number of its states. Thus, $\mathbb{O}(|\mathcal{T}_i|2^{|\varphi_i|}) = \mathbb{O}(|S_i|2^{|\varphi_i|}) = \mathbb{O}(|\mathbb{I}|2^{|\varphi_i|})$, where $|\mathbb{I}|$ is the number of cells of the cell decomposition D. The worst case of Step 2 costs $\mathbb{O}(|\mathbb{I}|2^{|\varphi_{\max}|})$ due to the fact that all WTSs $\mathcal{T}_i, i \in \mathcal{I}$ have the same number of states. Therefore, the complexity of the total framework is $\mathbb{O}(M|\mathbb{I}|6^N 2^{|\varphi_{\max}|})$.

5.5 Simulation Results

For a simulation example, a system of three agents with $x_i \in \mathbb{R}^2$, $i \in \mathcal{V} = \{1, 2, 3\}$, $\mathcal{N}_1 = \{2, 3\}$ $\mathcal{N}_2 = \{1, 3\}$, $\mathcal{N}_3 = \{1, 1\}$ is considered. The workspace $W = [-10, 10] \times [-10, 10] \subseteq \mathbb{R}^2$ is decomposed into hexagonal regions with $R = 1, r_h = \frac{\sqrt{3}}{2}$, which are depicted in Figure 5.7. The agents' initial positions are set to $x_1(0) = (0, 10r_h), x_2(0) = (-6, -8r_h)$ and $x_3(0) = (7.5, -7r_h)$. The sensing radius is $\underline{r} = 18$. The dynamics are set to: $\dot{x}_1 = -2x_1 + x_2 + x_3 - \sin^2(x_1 - x_2) + u_1$, $\dot{x}_2 = -2x_2 + x_1 + x_3 - \sin^2(x_2 - x_1) + u_2$ and $\dot{x}_3 = -2x_3 + x_1 + x_2 + u_3$. The time step is $T = 3$. The specification formulas are set to $\varphi_1 = \Diamond_{[15,27]}\{red\}, \varphi_2 = \Diamond_{[7.5,22]}\{green\}, \varphi_3 = \Diamond_{[0,19]}\{grey\}$ respectively. We set: $Q_i, P_i, R_i = I_2, \forall i \in \mathcal{V}$. Figure 5.7 shows a sequence of transitions for agents 1, 2 and 3 which form the accepting timed words $\widetilde{r}_1^t$, $\widetilde{r}_2^t$ and $\widetilde{r}_3^t$, respectively. Every timed word maps to a sequence of admissible control inputs for each agent, which is the outcome of solving the ROCPs. The agents remain connected for all $t \in [0, 6T]$. The simulations were carried out in MATLAB Environment by using the NMPC toolbox [128], on a desktop with 8 cores, 3.60GHz CPU and 16GB of RAM.

5.6 Conclusions

A systematic method of both decentralized abstractions and controller synthesis of a general class of coupled multi-agent systems has been proposed in which timed temporal specifications are imposed to the system. The solution involves a repetitive solving of an ROCP for every agent and for every desired region in order to build decentralized Transition Systems that are then used in the derivation of the controllers that satisfy the timed temporal formulas.

Chapter 6

Probabilistic Control Synthesis of Multi-Agent Systems

This chapter presents a fully automated procedure for controller synthesis for multi-agent systems under the presence of actuator and sensor uncertainties. We model the motion of each of the N agents in the environment as a Markov Decision Process (MDP) and we assign to each agent one individual high-level formula given in Probabilistic Computational Tree Logic (PCTL). Each agent may need to collaborate with other agents in order to achieve a task. The collaboration is imposed by sharing actions between the agents. We aim to design local control policies such that each agent satisfies its individual PCTL formula. The proposed algorithm builds on clustering the agents, MDP products construction and controller policies design. We show that our approach has reduced computational complexity than the centralized case, which traditionally suffers from very high computational demands.

6.1 Introduction

Most of the existing formal synthesis frameworks are based on the discretization of the agent's motion in a partitioned environment to a finite TS (via the abstraction process that was presented through the previous chapters) under the following assumptions: First, the measurements of the current state are accurate. Second, the transition system is either purely deterministic (namely, each control action enables a unique transition) or purely nondeterministic (namely, each control action enable multiple transitions). However, in realistic applications of robotic systems, noisy sensors and actuators can cause both of the aforementioned assumptions to be invalid. Motivated by this, we aim to model the multi-agent system in a probabilistic way such that the above two issues are taken into consideration.

Some recent works model the system in a probabilistic way and imposes high-level specifications, given in Linear Temporal Logic (see e.g., [135–139]). In [140], the authors modeled the system with an MDP and computed policies such that the satisfaction of a formula given in MITL, is maximized. Other works model the system in a probabilistic way with MDPs and introduce high-level tasks in PCTL (see e.g., [141–144]). However, all these works are restricted to single agent planning and they cannot be extended to multi-agent systems in a straightforward way since in the multi-agent case potential couplings may occur among the agents.

By extending these works to multi-agent systems, and at the same time handling an approach in which the system noise, model errors and external disturbances is explicitly considered, in this work we consider that each agent is modeled as a MDP and the task specifications are given in PCTL formulas. Motivated by the fact that in real applications, the agents (robots) are required to collaborate with each other to perform a task, we assume that there are agents in the system that are dependent to each other. They need to communicate, collaborate through sharing a common action in order to achieve a desired task.

The main contribution of the paper is to develop a strategy for controlling a general framework of N individual MDPs with respect to individual agents' specifications given in PCTL formulas. The proposed solution can handle the dependencies between the agents by considering agents clustering and MDP product construction and has provably better complexity than the centralized approach. When applied to robotic systems, our approach provides a framework for multi-robot control from temporal logic specifications with probabilistic guarantees. To the best of the authors' knowledge this is the first work that addresses the cooperative task planning for multi-agent systems under probabilistic temporal logic specifications in the presence of dependencies between the agents.

This chapter is divided into three parts. Section 6.2 provides the modeling of the system and the problem statement. Section 6.3 provides the technical details of the solution. Finally, conclusions are discussed in Section 6.4.

6.2 Problem Formulation

6.2.1 System Model and Abstraction

Consider a multi-agent team with $N \geq 2$ agents operating in the bounded workspace $\mathcal{W} \subseteq \mathbb{R}^n$. Let $\mathcal{V} = \{1, \ldots, N\}$ denote the index set of the agents. The workspace $\mathcal{W} = \bigcup\limits_{\ell \in \mathcal{W}} \gamma_\ell$ is partitioned using a finite number (assume W) of regions of interest $\gamma_1, \ldots, \gamma_W$. Denote by γ_ℓ^i an variable indicating that agent i is occupying the region γ_ℓ, where $i \in \mathcal{V}, \ell \in \mathcal{W}$.

Assumption 6.1. We assume here that an abstraction of the dynamics of each robot into a MDP is given, and that a low level continuous time controller that allows each robot to transit from one region γ_ℓ to an adjacent region γ_l with $\ell, l \in \mathcal{W}$, can be designed. It is also assumed that the probabilities of these transitions are known. This modeling has been also considered in [142].

Definition 6.1. The motion of each agent $i \in \mathcal{V}$ in the workspace can be described by a Markov Decision Process (MDP) $\mathcal{M}_i = (S_i, s_0^i, Act_i, T_i)$ where:

- $S_i = \{\gamma_1^i, \gamma_2^i, \ldots, \gamma_W^i\}$ is the set of states of agent i. The number of states for each agent is $|S_i| = W$, meaning that S_i includes all regions within $\mathcal{W}$.
- $s_0^i \in S_i$ is the initial state of agent i (the initial region where agent i may start). Note that the initial state is known and deterministic, i.e., we know exactly the region from which each agent starts its motion.
- Act_i is a finite set of actions (controls).
- $T_i : S_i \to 2^{Act_i \times \Sigma(S_i)}$ is the transition probability function.

Remark 6.1. We investigate here, under which conditions two or more agents are visiting simultaneously a specific region of the workspace. Consider a subset $\{i_1, \ldots, i_c\} \subseteq \mathcal{V}$, $c \geq 2$ of agents of the system under consideration. Let

$$r_{i_1} = s_0^{i_1} s_1^{i_1} s_2^{i_1} \ldots s_k^{i_1} \ldots s_n^{i_1},$$
$$\vdots$$
$$r_{i_c} = s_0^{i_c} s_1^{i_c} s_2^{i_c} \ldots s_k^{i_c} \ldots s_n^{i_c},$$

be the finite paths of length n that are executed by the corresponding MDPs $\mathcal{M}_1, \ldots, \mathcal{M}_c$, respectively, where $s_z^j \in S_z$, $\forall z \in \{0, \ldots, n\}$, $j \in \{i_1, \ldots, i_c\}$. Then, if there exists $k \geq 1$ such that for all the k-th elements of the above runs (in the same position at every path) it holds that $s_k^{i_1} = \cdots = s_k^{i_c} = s_k^{\text{meet}}$, then we say that the agents $\{i_1, \ldots, i_c\}$ are visiting simultaneously the same region s_k. If there does not exist such region s_k^{meet}, then the agents cannot meet simultaneously to one region.

6.2.2 Handshaking Actions

The motivation for introducing dependencies in the multi-agent system comes from real applications where more than one agents (robots) need to collaborate with each other in oder to perform a desired task. For example, imagine two aerial manipulators that are required to meet and grasp an object simultaneously and deliver it to a specific location in a warehouse.

In order to be able to introduce dependencies in the actions between the agents, we write the action set of each agent as: $Act_i = \{\Pi_i, \hat{\Pi}_i\}, i \in \mathcal{V}$, where Π_i is a finite set of actions that the agent i need to execute in collaboration with other agents (*handshaking* actions) and $\hat{\Pi}_i$ is a finite set of actions that the agent i executes independently of the other agents (*independent* actions). For the independent actions it holds that:

$$\hat{\Pi}_i \cap \hat{\Pi}_j = \emptyset, \forall i \neq j, i, j \in \mathcal{V}.$$

The independent actions can always be executed without any constraints. On the other hand, for the handshaking actions, we have the following requirements:

- First, the agents are required to meet and occupy the same region of the workspace (not necessarily a specific region).
- Once they meet, they need to execute simultaneously the same action.
- All the agents that share an action are required to execute it in order for the action to be completed properly.

Formally, the handshaking actions are defined as follows:

Definition 6.2. (Handshaking Actions) Let $\{i_1, \ldots, i_c\} \subseteq \mathcal{V}, c \geq 2$ be a set of agents that need to collaborate in order to execute simultaneously a task under the action α. The following two properties should hold in order for α to be well-posed handshaking action:

1. $\alpha \in \bigcap\limits_{i \in \{i_1, \ldots, i_c\}} \Pi_i$.
2. Let the following finite paths of length n:

$$r_{i_1} = s_0^{i_1} \xrightarrow{\alpha_0^{i_1}} \ldots \longrightarrow s_k^{i_1} \xrightarrow{\alpha} s_{k'}^{i_1} \ldots \xrightarrow{\alpha_{n-1}^{i_1}} s_n^{i_1},$$

$$\vdots$$

$$r_{i_c} = s_0^{i_c} \xrightarrow{\alpha_0^{i_c}} \ldots \longrightarrow s_k^{i_c} \xrightarrow{\alpha} s_{k'}^{i_c} \ldots \xrightarrow{\alpha_{n-1}^{i_c}} s_n^{i_c},$$

be executed by the MDPs $\mathcal{M}_{i_1}, \ldots, \mathcal{M}_{i_c}$ respectively. Here, $s_k^{i_1}, \ldots, s_k^{i_c}$ are the regions that the agents $i_1, \ldots, i_c$ should occupy, respectively, in order to execute the handshaking action α simultaneously. Then, there should exist at least one index $k \geq 0$ such that $s_k^{i_1} = \cdots = s_k^{i_c} = s_k^{\text{meet}}$ and $\delta(s_k^j, \alpha, s_{k'}^j) > 0$ for at least one $s_{k'}^j \in \text{Post}(s_k^j, \alpha)$ for every agent $j \in \{i_1, \ldots, i_c\}$.

Notice that the same condition for a state s_k^{meet} as in condition (2) was mentioned in Remark 6.1, but here the existence of a common action α is also required. It should be also noted that every region of the workspace in which the agents can potentially meet, can serve as a region that a handshaking action can be executed (if such an action exists).

6.2.3 Dependencies

Suppose that one agent i receives a cooperative task that involves other agent's $j \in \mathcal{V} \backslash \{i\}$ participation. This means that both agents need to execute the same action at the same region so as for the task to be performed. The dependencies are formally defined as:

Definition 6.3. The agents $i, j \in \mathcal{V}$ are called *dependent* if one the following statements holds:

1. Agent i depends on agent j if $\Pi_i \cap \Pi_j \neq \emptyset$, with $\Pi_i \in Act_i$ and $\Pi_j \in Act_j$.
2. Agent j depends on agent i if $\Pi_j \cap \Pi_i \neq \emptyset$, with $\Pi_j \in Act_j$ and $\Pi_i \in Act_i$.
3. There exist at least one region $s_k^{\text{meet}}, k \geq 0$ of the workspace such that the second condition of Definition 6.2 holds.

Conditions 1, 2 can be checked by comparing all the elements of the sets $\Pi_i, \Pi_j, \forall i, j \in \mathcal{V}$ one by one. Condition 3 can be checked by using graph search algorithms.

Remark 6.2. It should be noticed from the above definitions that all the agents that share an action, they are required to meet and execute it simultaneously.

Remark 6.3. Due to fact that the control policies are defined over finite paths, the handshaking actions are defined with respect to finite paths as well. Therefore, the graph search algorithm for condition (3) is searching into a finite graph.

Assumption 6.2. It is assumed that there exists at least two agents that are dependent. Otherwise, there exist no dependencies between the agents and the problem that is later defined can be straightforwardly solved by solving the controller synthesis methodology of Section 6.3 for each agent independently.

6.2.4 Problem Statement

We formally define here the problem that we aim to solve in this chapter:

Problem 6.1. Given N agents performing in a workspace $\mathcal{W}$, under the Assumptions 6.1, 6.2, individual task specification formulas $\varphi_1, \dots, \varphi_N$ over the actions $\Pi_i \cup \hat{\Pi}_i, i \in \mathcal{V}$ given in PCTL with semantics as in Section 2.4, synthesize individual control policies $\mu_1, \dots, \mu_N$ (if there exists one) which guarantee the satisfaction of the formulas $\varphi_1, \dots, \varphi_N$ respectively.

Remark 6.4. Traditionally, the PCTL semantics are defined over a set of atomic propositions. However, in this chapter, we aim to introduce dependencies over the actions among the agents. Therefore, the PCTL semantics are defined over a set of actions.

Remark 6.5. Among the three category of problems that are presented in Section 2.4, in this chapter, we are mainly interested in the Controller Synthesis problem i.e. Given an MDP $\mathcal{M}$ and a property φ, find all the control policies under which the formula is satisfied. The motivation for that is the following: if one control policy fails to guarantee the satisfaction of a formula, it may exists another policy under which the formula is satisfied.

6.3 Control Synthesis

6.3.1 Overview

An overview of the proposed solution is given as follows:

- Step 1: First, the dependencies among the agents are modeled as a dependency graph (see Section 6.3.2). The agents are split into clusters and each cluster contains the agents that are dependent according to Definition 6.3.
- Step 2: For each cluster of agents, the mutual specification φ_m and the product MDP $\widetilde{M}$ are defined (see Section 6.3.3).
- Step 3: By utilizing the controller synthesis algorithms of Section 6.3.6, we design a control policy $\widetilde{\mu}$ of each cluster that guarantees the satisfaction of φ_m (if such a control policy exists). We provide in Section 6.3.4 the definition of successful control policies, which project onto local control policies $\mu_1, \dots, \mu_N$ for each agent, which finally are a solution to Problem 1.

An algorithm describing all the steps of the proposed procedure is given in Section 6.3.5. Probabilistic model checking algorithms, which can compute all the control policies under which a PCTL formula φ_m is satisfied, are presented in Section 6.3.6. The computational complexity of the proposed framework is discussed in Section 6.3.7.

Problem 6.1 can be solved in a centralized way by computing the product of all individual MDPs $\mathcal{M}_i, i \in \mathcal{V}$ (see Definition 6.7) and performing the proposed methodology of this paper to the centralized system without any clustering among the agents. A comparison of the computational complexity of the proposed framework that exploits the potential sparsity of dependencies with the centralized approach is discussed in Section 6.3.7.

6.3.2 Modeling the Dependencies

Based on the dependency relation of the Definition 6.3, the dependency graph associated with the handshaking actions $\Pi_i, i \in \mathcal{V}$ is defined as follows:

Definition 6.4. The *dependency graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, is an undirected graph that consists of the set of vertices $\mathcal{V}$ in which each of the agents is a node of the graph and the edge set $\mathcal{E}$ which is defined as follows:

$$\mathcal{E} = \{\{i,j\} : i \text{ is dependent to } j \text{ and } i,j \in \mathcal{V}, i \neq j\}.$$

In order to proceed, the following definition is required:

Definition 6.5. [58] Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected graph. Then every graph $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$ with $\mathcal{V}' \subseteq \mathcal{V}$ and $\mathcal{E}' \subseteq \mathcal{E}$ if called a *subgraph* of the graph $\mathcal{G}$.

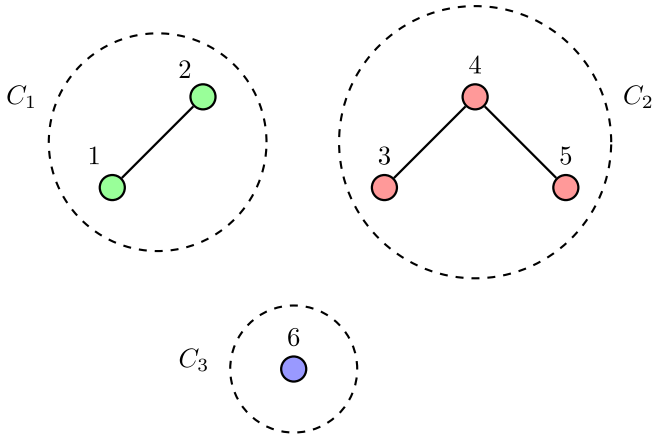

Figure 6.1: An example of a dependency graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and its subgraphs for $N = 6$ agents.

Definition 6.6. The set $\mathcal{C} = \{C_\ell : \ell \in \mathbb{M}\} \subseteq \mathcal{V}$, where $\mathbb{M} = \{1, \ldots, m\}$, forms a *dependency cluster* if and only if for every $i, j \in \mathcal{C}$ there is a path from node i to node j in the dependency graph $\mathcal{G}$; m denotes the number of the dependency clusters.

Define the function $f : \mathcal{V} \to \mathbb{M}$ which maps each agent to the index of the cluster that it belongs to. It can be observed that $\bigcup\limits_{\ell \in \mathbb{M}} C_\ell = \mathcal{V}$ and $\sum\limits_{\ell \in \mathbb{M}} |C_\ell| = |\mathcal{V}| = N$.

Each agent $i \in \mathcal{V}$ for which there does not exist $j \in \mathcal{V}$ such that $j \in C_{f(i)}$ will be called an *independent agent*. For an independent agent it holds that $|C_{f(i)}| = 1$. From Definition 6.6, it follows that every dependency cluster $C_\ell \in C, \ell \in \mathbb{M}$ is the vertex set of the subgraphs $G^{(\ell)} = (\mathcal{C}_\ell, \mathcal{E}_\ell), \mathcal{E}_\ell \subseteq \mathcal{E}, \ell \in \mathbb{M}$ of the system graph $\mathcal{G}$. Loosely speaking, two agents belong to the same cluster when they are directly dependent or transitively dependent by a dependency chain. An example of a dependency graph and dependency clusters is given as follows:

Example 6.1. Consider $N = 6$ agents with $\mathcal{V} = \{1, \ldots, 6\}$, $\mathcal{E} = \{\{1,2\}, \{3,4\}, \{4,5\}\}$. The $m = 3$ clusters are given as: $C_1 = \{1,2\}, C_2 = \{3,4,5\}$ and $C_3 = \{6\}$ and the corresponding subgraphs $\mathcal{G}^{(1)} = (C_1, \mathcal{E}_1 = \{1,2\}), \mathcal{G}^{(2)} = (C_2, \mathcal{E}_2 = \{\{3,4\}, \{4,5\}\})$ and $\mathcal{G}^{(3)} = (C_3, \mathcal{E}_3 = \emptyset)$. Moreover, $f(1) = f(2) = 1, f(3) = f(4) = f(5) = 2, f(6) = 3$. The dependency graph is depicted in Figure 6.1.

According to the mathematical derivation above, Assumption 6.2 is modified as follows:

Assumption 6.3. There exists at least one dependency cluster $C_\ell, \ell \in \mathbb{M}$ (as was defined in Definition 6.6) of the dependency graph $\mathcal{G}$ of the under consideration multi-agent system, which contains at least two dependent agents. Thus, there exists $\ell \in \mathbb{M}$ such that: $|C_\ell| = 2, \text{if } N = 2$ and $|C_\ell| \in [2, N-1], \text{if } N > 2$.

By employing the above computation, the initial multi-agent system is modeled as m subgraphs $\mathcal{G}^\ell, \ell \in \mathbb{M}$ which capture the dependencies between the agents, as defined in Definition 6.3. This forms a convenient modeling of the system's dependencies in order to compute the product MDP of every subsystem $\ell \in \mathbb{M}$ in the next Section.

6.3.3 Product Markov Decision Process

Define the *mutual specification* of a cluster of agents C_ℓ by:

$$\varphi_m^\ell = \bigwedge_{i \in C_\ell} \varphi_i, \ell \in \mathbb{M}, \tag{6.1}$$

over the set of actions $\bigcup_{i \in C_\ell} \left(\Pi_i \cup \hat{\Pi}_i\right)$. If the satisfaction of φ_m^ℓ for each cluster C_ℓ is guaranteed, it holds by definition that the satisfaction of all the individual formulas $\varphi_i, i \in \mathcal{V}$ is guaranteed as well. Thus, a method for finding a team control policy that guarantees the satisfaction of the formula $\varphi_m^\ell, \ell \in \mathbb{M}$ should be provided.

In the sequel, we construct a product MDP that captures the collaborative behavior of all the agents within a cluster. Having $\widetilde{M_\ell}$, allow us to synthesize a control policy $\widetilde{\mu}$ for C_ℓ, which guarantees the satisfaction of the collaborative formula φ_m^ℓ. Subsequently, the team control policy $\widetilde{\mu}_\ell$ can be projected onto the local agents' control policies $\mu_1, \ldots, \mu_N$ which are a solution to Problem 1.

Definition 6.7. (Product MDP) The *product MDP* $\widetilde{\mathcal{M}_\ell}$, $\ell \in \mathbb{M}$ for the cluster of agents C_ℓ is a tuple $(\widetilde{S}_\ell, \widetilde{s}_0^\ell, \widetilde{Act_\ell}, \widetilde{T}_\ell)$ where:

- $\widetilde{S}_\ell = \bigtimes_{i \in C_\ell} S_i$ is the set of states, where $\bigtimes$ stands for the Cartesian product operator, as defined in Chapter 2.
- $\widetilde{s}_0^\ell = \bigtimes_{i \in C_\ell} s_0^i$ is the initial state.
- $\widetilde{Act_\ell} = \bigcup_{i \in C_\ell} Act_i = \bigcup_{i \in C_\ell} \left\{\Pi_i, \hat{\Pi}_i\right\}$ is the set of actions.
- $\widetilde{T}_\ell : \widetilde{S}_\ell \to 2^{\widetilde{Act_\ell} \times \Sigma(\widetilde{S}_\ell)}$ is the transition probability function for the product system. Similar to δ of Definition 2.17, we define $\widetilde{\delta}(\widetilde{s}, \widetilde{\alpha}, \widetilde{s}') \in [0,1]$ the probability of transitioning from the state $\widetilde{s}$ to the state $\widetilde{s}'$ under the action $\widetilde{\alpha}$. Let $C_\ell = \{i_1, \ldots, i_{|C_\ell|}\}$ be an enumeration of the agents of the cluster C_ℓ. Then, $\widetilde{\delta}_\ell$ is defined as follows:

1. if $\alpha \in \bigcap_{j \in C_\ell} \mathcal{A}(s_j)$ then

$$\widetilde{\delta}((s_{i_1}, \ldots, s_{i_{|C_\ell|}}), \alpha, (s'_{i_1}, \ldots, s'_{i_{|C_\ell|}})) = \prod_{j \in C_\ell} \delta_j(s_j, \alpha, s'_j).$$

2. $\widetilde{\delta}((\widetilde{s}_{i_1}, \ldots, \widetilde{s}_{k_1}, \ldots, \widetilde{s}_{k_\nu}, \ldots, \widetilde{s}_{i_{|C_\ell|}}),\ \alpha, (\widetilde{s}_{i_1}, \ldots, \widetilde{s}'_{k_1}, \ldots, \widetilde{s}'_{k_\nu}, \ldots, \widetilde{s}_{i_{|C_\ell|}})) = \prod_{j=1}^{\nu} \delta_{k_j}(s_{k_j}, \alpha, s'_{k_j})$ if

$$\alpha \in \left[\bigcap_{j=1}^{\nu} \mathcal{A}(s_{k_j})\right] \Bigg\backslash \left[\bigcup_{z \in C_\ell \backslash \{k_1, \ldots, k_\nu\}} \mathcal{A}(s_z)\right],$$

for $k_j \in C_\ell, j \in \{1, \ldots, \nu\}$.

Intuitively, 1 denotes that all the agents $i_1, \ldots, i_{|C_\ell|}$ of the cluster $|C_\ell|$ are located in the states $s_{i_1}, \ldots, s_{i_{|C_\ell|}}$ respectively, and they are simultaneously transiting to the states $s'_{i_1}, \ldots, s'_{i_{|C_\ell|}}$ with action α; 2) denotes that among all the agents of the cluster C_ℓ, only the agent $\{k_1, \ldots, k_\nu\} \subsetneq C_\ell$ are transiting simultaneously to the states $s'_{k_1}, \ldots, s'_{k_\nu}$ respectively; 2) can not be handshaking action since for the handshaking action all the agents of the cluster should transit simultaneously to the next state. According to Definition 6.2, in order for 1 to be a handshaking transition, it is also require that $s_{i_1} = \ldots = s_{i_{|C_\ell|}}$.

Remark 6.6. In the case of a cluster $\ell \in \mathbb{M}$ that contains an independent agent $i \in \mathcal{V}$ with the property $|C_\ell| = |C_{f(i)}| = 1$, the product MDP $\widetilde{\mathcal{M}_\ell}$ coincides with the individual MDP $\mathcal{M}_i$ of Definition 6.1, i.e., $\widetilde{\mathcal{M}_\ell} \equiv \mathcal{M}_i$.

The infinite path $\widetilde{r}$, the finite path $\widetilde{\rho}$, the control policy $\widetilde{\mu}$ and the set of all infinite and finite paths $\widetilde{FPath}$ and $\widetilde{IPath}$, are defined similarly to Section 2.4.

6.3.4 Designing the Control Policies $\widetilde{\mu}$

The product MDP $\widetilde{M_\ell}, \ell \in \mathbb{M}$ of each cluster captures the paths and the control policies of the agents that belong to the same cluster and they are required to collaborate for achieving a task or acting independently.

By employing the controller synthesis algorithms (see Section 6.3.6), the control policies $\widetilde{\mu}_\ell$ for the team of agents in each cluster can be designed. The algorithms can compute all the control policies $\widetilde{\mu}_\ell$ that guarantees the satisfaction of formula φ_m^ℓ from (6.1).

What remains is to project these policies onto the individual control policies of the agents of each cluster in such a way that they serve as a solution to Problem 1.

Consider a cluster of agents $C_\ell = \{i_1, \dots, i_{|C_\ell|}\}$. A control policy $\widetilde{\mu}_\ell(\widetilde{\rho}) = \widetilde{\mu}_\ell(\widetilde{s}_0\widetilde{s}_1 \dots \widetilde{s}_n) \subseteq \widetilde{Act}$ for the finite path $\widetilde{\rho} = \widetilde{s}_0\widetilde{s}_1 \dots \widetilde{s}_n$ of length n, where $\widetilde{s}_k = (s^k_{i_1}, \dots, s^k_{i_{|C_\ell|}}), k \in \{1, \dots, n\}$, projects onto the local individual control policies $\mu_j(s^1_j, \dots, s^n_j), j \in \{i_1, \dots, i_{|C_\ell|}\}$, of the agents $\{i_1, \dots, i_{|C_\ell|}\}$ of the cluster $C_\ell, \ell \in \mathbb{M}$. Note that: $\mu_j \subseteq \widetilde{Act}\Big|_j \subseteq Act_j, j \in \{i_1, \dots, i_{|C_\ell|}\}$, and $\widetilde{Act}\Big|_j$ is the set of actions of the agent j that are appearing in the set $\widetilde{Act}$.

The set $\widetilde{\mu}(\widetilde{\rho})$ contains control policies that are either handshaking or independent. Let us also define the following set of handshaking actions: $\mathrm{Succ}(\alpha, \ell) = \{\alpha \in \Pi_{i_1} \cap \dots \cap \Pi_{i_{|C_\ell|}} : \alpha \in \widetilde{\mu}_\ell(\widetilde{\rho})\}$, which is the subset of $\widetilde{\mu}(\widetilde{\rho})$ that contains the handshaking actions. We need to search now if all the projections $\mu_{i_j}, \forall j \in \{1, \dots, |C_\ell|\}$ follow the handshaking rules of Definition 6.2.

Definition 6.8. (Successful Control Policy) Let $\widetilde{\mu}_\ell(\widetilde{\rho}) = \widetilde{\mu}_\ell(\widetilde{s}_0\widetilde{s}_1 \dots \widetilde{s}_n) \subseteq \widetilde{Act}$ be a control policy of a cluster C_ℓ. The control policy $\widetilde{\mu}_\ell(\widetilde{\rho})$ is called *successful* if for all $\alpha \in \mathrm{Succ}(\alpha, \ell)$ it holds that $s^n_{i_1} = \dots = s^n_{i_{|C_\ell|}}$ and $\delta(s^n_j, \alpha, (s^n_j)') > 0$ for at least one $(s^n_j)' \in \mathrm{Post}(s^n_j, \alpha), j \in \{i_i, \dots, i_{|C_\ell|}\}$.

Define by

$$SP(\ell) = \left\{\widetilde{\mu}_\ell(\widetilde{\rho}) \subseteq \widetilde{Act} : \widetilde{M}_\ell \models \varphi^\ell_m\right\}, \ell \in \mathbb{M},$$

the set of all the control policies that guarantee the satisfaction of the formula φ^ℓ_m. All the control policies of the set SP needs to be checked if they are successful. If $SP(\ell) = \emptyset$ for at least one $\ell \in \mathbb{M}$, then the Problem 6.1 has no solution. The set $SP(\ell)$ is computed by employing the algorithms of Section 6.3.6.

6.3.5 Proposed Algorithm

The proposed procedure of solving Problem 6.1 is shown in Algorithm 2. The function checkDepend() determines the dependent agents according to Definition 6.3. The product and projection that were introduced in Sections 6.3.3 and 6.3.4, can be computed by the functions product() and projection(), respectively. The algorithms of Section 6.3.6 are incorporated in the function controlSynthesis(). By employing Definition 6.8, the function succPolicy() determines if a sequence of control policies are successful.

Remark 6.7. Even though our proposed solution is centralized in each cluster, it is partially decentralized in terms of the whole multi-agent system.

6.3.6 Algorithms for Probabilistic Control Synthesis

In this section, we are investigating algorithms of computing all the control policies $\widetilde{\mu}_\ell \in SP(\ell)$. Once these control policies are found, then by following Algorithm 2,

Algorithm 2 - SolveProblem1

```
1: Input: MDPs: $\mathcal{M}_1, \dots, \mathcal{M}_N$;
2:         PCTL Formulas: $\varphi_1, \dots, \varphi_N$;
3: Output: $\mu_1, \dots, \mu_N$
4:
5: $\mathcal{C} = \{C_\ell, \ell \in \mathbb{M}\} = \text{checkDepend}(Act_1, \dots, Act_N)$;
6: $\varphi_m^\ell = \bigwedge_{i \in C_\ell} \varphi_i$;
7: for $z \in C_\ell = \{i_1, \dots, i_{|C_\ell|}\}$ do
8:     $\widetilde{M}_\ell = \text{product}(\{\mathcal{M}_j, j \in C_\ell\})$;
9:     $SP(\ell) = \text{controlSynthesis}(\widetilde{M}, \varphi_m^\ell)$;
10:    for $\widetilde{\mu}_\ell \in SP(\ell)$ do
11:        $\{\mu_1, \dots, \mu_N\} = \text{projection}(\widetilde{M}_\ell)$;
12:        if $\text{succPolicy}(\{\mu_1, \dots, \mu_N\}) = \top$ then
13:            solFound = 1;
14:            return $\{\mu_1, \dots, \mu_N\}$;                ▷ Solution found
15:        else
16:            Go to line 11;                      ▷ Search other control policies
17:        end if
18:    end for
19:    if solFound $\neq 1$ then
20:        Problem 1 has no solution;
21:    end if
22: end for
```

the individual policies $\mu_j, j \in \{i_1, \dots, i_{|C_\ell|}\}$ can be designed and the Problem 1 is solved (if there exists a solution). For more details about the algorithms we refer to [40, 41, 142, 145, 146].
First, define by:

$$\text{Sat}(\varphi_m^\ell) = \{s \in S : s \models \varphi_m^\ell\},$$

the set of states that satisfy φ_m^ℓ. Then, for two given PCTL formulas $\varphi_{m,1}^\ell$, $\varphi_{m,2}^\ell$ we have the following:

$$\begin{aligned}
&\text{Sat}(\top) = S,\\
&\text{Sat}(\pi) = \{s \in S : \pi \in \mathcal{A}(s)\},\\
&\text{Sat}(\neg\varphi_m^\ell) = S \backslash \text{Sat}(\varphi_m^\ell),\\
&\text{Sat}(\varphi_{m,1}^\ell \wedge \varphi_{m,2}^\ell) = \text{Sat}(\varphi_{m,1}^\ell) \cap \text{Sat}(\varphi_{m,2}^\ell)
\end{aligned}$$

Define also the minimum and the maximum probabilities of satisfying the formula under the control policy μ for a starting state s:

$$Prob_{\max}(s,\psi) = \sup_{\mu\in M}\{Prob_{\mu}(s,\psi)\}, \tag{6.2a}$$
$$Prob_{\min}(s,\psi) = \inf_{\mu\in M}\{Prob_{\mu}(s,\psi)\}. \tag{6.2b}$$

where M is set of all control policies. It has been proved in [41], that the model checking problem problem of the operator $\mathcal{P}_{\bowtie p}[\psi]$ can be reduced to the computation of (6.2a), (6.2b) according to the following:

- If $\bowtie = \{\geq, >\}$ then
$$s \models \mathcal{P}_{\bowtie p}[\psi] \Leftrightarrow Prob_{\min}(s,\psi) \bowtie p; \tag{6.3}$$
- If $\bowtie = \{\leq, <\}$ then
$$s \models \mathcal{P}_{\bowtie p}[\psi] \Leftrightarrow Prob_{\max}(s,\psi) \bowtie p; \tag{6.4}$$

For the controller synthesis (as was defined in Section 2.4) of the path operators $\mathcal{P}_{\bowtie p}[\bigcirc\varphi^{\ell}_{m}]$, $\mathcal{P}_{\bowtie p}[\varphi^{\ell}_{m,1}\mathcal{U}^{\leq k}\varphi^{\ell}_{m,2}]$, we utilize the following Algorithms 3 and 4.

Algorithm 3

If the collaborative formula has the form $\varphi^{\ell}_{m} = \mathcal{P}_{\bowtie p}[\bigcirc\varphi^{\ell}_{m,1}]$, initially the maximum probability of satisfying φ^{ℓ}_{m} at the state $s \in S$:

$$Prob_{\max}(s,\varphi^{\ell}_{m}) = \max_{\alpha\in\mathcal{A}(s)}\left(\sum_{s'\in Sat(\varphi^{\ell}_{m,1})}\delta(s,\alpha,s')\right), \tag{6.5}$$

is computed for every $s \in S$. By replacing max with min in (6.5), $Prob_{\min}(s,\bigcirc\varphi^{\ell}_{m,1})$ can be computed. Define the vector $\Phi(s) = 1$, if $s \in Sat(\varphi^{\ell}_{m,1})$ or $\Phi(s) = 0$, otherwise. Perform now the matrix multiplication $X = T\cdot\Phi$; X is a vector whose entries are the probabilities of satisfying $\bigcirc\varphi^{\ell}_{m,1}$, where each row corresponds to a state-action pair. After obtaining the vector X, eliminate the state-actions pairs whose probabilities are not in the range of $\bowtie p$ by taking into consideration the conditions (6.3), (6.4). This operation determines all the states $s \in S$ and all the actions $\mu \in M$ that satisfy the formula φ^{ℓ}_{m}.

Algorithm 4

For a collaborative formula of the form $\phi^{\ell}_{m} = \mathcal{P}_{\bowtie p}\left[\varphi^{\ell}_{m,1}\;\mathcal{U}^{\leq k}\;\varphi^{\ell}_{m,2}\right]$, define by:

$$\begin{aligned} S^{\text{yes}} &= \text{Sat}(\varphi^{\ell}_{m,2}),\\ S^{\text{no}} &= S\backslash\left[\text{Sat}(\varphi^{\ell}_{m,1}) \cup \text{Sat}(\varphi^{\ell}_{m,2})\right],\\ S^{\text{rem}} &= S\backslash(S^{\text{yes}} \cup S^{\text{no}}), \end{aligned}$$

the states that always satisfy the specification, the states that never satisfy the specification and the remaining states, respectively. Compute the maximum probability of satisfying φ_m^ℓ at the state $s \in S$ as: $Prob_{\max}(s, \varphi_m^\ell) = 1$ or $0,$ if $s \in S^{\text{yes}}$ or $s \in S^{\text{no}}$ respectively. For $s \in s \in S^{\text{rem}}$ and $k > 0$ compute recursively the following:

$$\begin{aligned}&Prob_{\max}(s, \varphi_m^\ell, k)\\ &= \max_{\alpha \in \mathcal{A}(s)} \left(\sum_{s' \in S^{\text{rem}}} \delta(s, \alpha, s') Prob_{\max}(s, \varphi_m^\ell, k-1) + \sum_{s' \in S^{\text{yes}}} \delta(s, \alpha, s') \right), \qquad (6.6)\end{aligned}$$

with $Prob_{\max}(s, \varphi_m^\ell, 0) = 0$. The computation can be carried out in k iterations, each similar to the process of Algorithm 3. By replacing max with min in (6.6), $Prob_{\min}(s, \varphi_m^\ell, k)$ can be computed.

Remark 6.8. The resulting control strategies of the aforementioned algorithms are stationary. Therefore, the control policies $\widetilde{\mu}_\ell(\widetilde{s}_1 \widetilde{s}_2 \ldots \widetilde{s}_n)$ depend only to the state $\widetilde{s}_n$.

6.3.7 Computational Complexity

According to [30], the time complexity for PCTL control synthesis is polynomial in the number of states of $\mathcal{M}$ and linear in the length of the formula φ. Denote by $|\varphi|$ the length of the formula φ in terms of the number of the operator it has e.g., $|\mathcal{P}_{\geq 0.5}[\bigcirc\{red\}]| = 2$. The complexity can be expressed mathematically as

$$\mathcal{O}(\text{poly}(|\mathcal{M}|)|\varphi|\kappa(\varphi)),$$

where poly() denotes the polynomial time,

$$\kappa(\varphi) = \max\{k : \phi_1 \mathcal{U}^{\leq k} \varphi_2\},$$

is the maximum step bound that appears in a sub-formula of the form $\varphi_1 \mathcal{U}^{\leq k} \varphi_2$. If φ does not contain any until operators, then $\kappa(\varphi) = 1$.

The number of states of the the product MDP in the centralized solution is $|\widetilde{S}| = \prod_{i \in \mathcal{V}} |S_i| = W^N$ and the corresponding complexity is in the class of

$$O = \mathcal{O}\left(\text{poly}(W^N)|\varphi_m^\ell|\kappa(\varphi_m^\ell)\right),$$

where φ_m^ℓ as it is defined in (6.1), for $|C_\ell| = N$.

The worst case complexity of the proposed framework occurs when one agent is independent and the other $N-1$ agents are dependent to each other. Then, there exists two clusters $\ell \in \{1, 2\}$: the first cluster contains the independent agent and the other one contains the remaining agents. The corresponding MDPs have $|\widetilde{S}_\ell| = |\bigtimes_{i \in C_\ell} S_i| = W^{|C_\ell|}, \ell \in \{1, 2\}$ states i.e., W, W^{N-1} states, respectively. Thus, the worst case complexity of our framework is:

$$\widetilde{O} = \mathcal{O}\bigg(\text{poly}(W)|\varphi_m^1|\kappa(\varphi_m^1) + \text{poly}(W^{N-1})|\varphi_m^2|\kappa(\varphi_m^2)\bigg).$$

The best case complexity of the proposed framework is when every agent is dependent to at most one other agent. Formally, if N is odd number then $|C_{\ell'}| = 1$ for only one $\ell' \in \mathbb{M}$ and $|C_\ell| = 2, \forall \ell \in \mathbb{M} \backslash \{\ell'\}$. In this case, the best case complexity is in the class:

$$\begin{aligned}\bar{O} &= \mathcal{O}\bigg(\sum_{\ell \in \mathbb{M} \backslash \{\ell'\}} \big[\text{poly}(W^2)|\varphi_m^\ell|\kappa(\varphi_m^\ell)\big] + \text{poly}(W)|\varphi_m^{\ell'}|\kappa(\varphi_m^{\ell'})\bigg) \\ &= \mathcal{O}\bigg(\text{poly}(W^2) \sum_{\ell \in \mathbb{M} \backslash \{\ell'\}} \big[|\varphi_m^\ell|\kappa(\varphi_m^\ell)\big] + \text{poly}(W)|\varphi_m^{\ell'}|\kappa(\varphi_m^{\ell'})\bigg).\end{aligned}$$

If N is even number, then previous summation in performed in all the elements $\ell \in \mathbb{M}$. In total, it holds that $\bar{O} < \widetilde{O} < O$, which verifies that our proposed framework achieves significantly better computational complexity than the centralized one.

6.4 Conclusions

We have proposed a systematic method for designing control policies for multi-agent systems under the presence of uncertainties. We assume that the under consideration system is under the presence of model uncertainties and actuation failures, thus the modeling is performed through MDPs. The agents are divided into dependency clusters which indicate the team of agents that they need to share an action in order to achieve a desired task. With the proposed framework, each agent is guaranteed to perform a task given in PCTL formulas. The computational complexity of the proposed framework is significantly better than the complexity of the centralized framework.

Chapter 7

Summary and Future Research Directions

This thesis was divided into three main parts, corresponding to Chapter 3 (Part 1), Chapter 4 and 5 (Part 2) and Chapter 6 (Part 3).

In Chapter 3, we proposed a potential-functions based decentralized control protocol for multi-agent systems which guarantees formation control with inter-agent collision avoidance, collision avoidance between the agents and the obstacles/workspace boundary, connectivity maintenance as well as singularity avoidance of multiple rigid bodies. Simulation results have verified the validity of the proposed approach. We are currently working towards resolving the issues with Assumption 3.3. The main disadvantage of the proposed control scheme is the requirement of calculations of derivatives which in practical applications, can be numerically unstable. A second problem is the fact that a lot of actuation force is required by the controller. A possible extension could be a framework towards NMPC such that an optimization problem is solved by each agent at every sampling time, in which control input constraints can be handled by the optimization problem. Other efforts will be devoted towards real-time experiments with a multi-agent team consisted of quad-rotors at the Smart Mobility Lab of KTH.

In Chapter 4, we proposed a systematic method for multi-agent controller synthesis aiming cooperative planning under high-level specifications given in MITL formulas. The solution involves a sequence of algorithmic automata constructions such that not only team specifications but also individual specifications should be fulfilled. For Chapter 4, future research directions include computational burden relaxation of the proposed scheme. Robust satisfaction criteria could be also a possible target framework, i.e., the introduction of metrics of how far away are the MITL formulas from their satisfaction. A framework that deals with dependencies in discrete level through atomic propositions, i.e., $\Sigma_i \cap \Sigma_j \neq \emptyset$, $i, j \in \mathcal{V}$ with $i \neq j$, could be also an interesting topic for further investigation.

In Chapter 5, a systematic method of both decentralized abstractions and controller synthesis of a general class of coupled multi-agent systems was proposed in which timed temporal specifications are imposed to the system. The solution involves a repetitive solving of an ROCP for every agent and for every desired region in order to build decentralized Transition Systems that are then used in the derivation of the controllers that satisfy the timed temporal formulas. For Chapter 5, future work includes further computational improvement of the proposed decentralized abstraction method towards finding more efficient methods of robust controllers in order to deal with the disturbances of the agents. Furthermore, each agent can be modeled as a rigid body such that the time T in which the agent is required for performing a transition between two neighboring polygon regions, can be computed easier. Decentralized event-based communication controllers between the agents, in order for the communication burden to be relaxed, could be also applied.

In Chapter 6, we proposed a systematic method for designing control policies for multi-agent systems under the presence of uncertainties. We assume that the system is under the presence of model uncertainties and actuation failures, thus the modeling is performed through MDPs. The agents are divided into dependency clusters which indicate the team of agents that they need to share an action in order to achieve a desired task. With the proposed framework, each agent is guaranteed to perform a task, given in PCTL formulas. The computational complexity of the proposed framework is significantly better than the complexity of the centralized framework. For Chapter 6, future efforts will be devoted towards investigating about stochastic abstractions, which in this chapter is considered be given according to Assumption 6.1, as well as a more decentralized control framework.

Finally, a multi-agent control design framework that combines time and probability i.e., a combination of Part 2 and Part 3 of this thesis, could be promising since it models more accurate the real multi-agent systems.

Appendix A

List of Derivatives for Chapter 3

By introducing the notations $x_i^+ = [p_i^\top, 0, 0, 0]^\top \in \mathbb{M}$, $i \in \mathcal{V}$ and $x_{o_z}^+ = [p_{o_z}^\top, 0, 0, 0]^\top \in \mathbb{M}$, $z \in \mathcal{Z}$, we obtain the following derivatives:

$$\nabla_{x_i}\varphi_i(x) = \nabla_{x_i}\gamma_i(x) - \frac{\nabla_{x_i}\beta_i(x)}{\beta_i(x)^2}, \tag{A.1a}$$

$$\nabla_{x_i}(\eta_{ij,a}) = \nabla_{x_i}\left(\|p_i - p_j\|^2\right) = 2(x_i^+ - x_j^+), \tag{A.1b}$$

$$\nabla_{x_i}(\eta_{iz,o}) = \nabla_{x_i}\left(\|p_i - p_{o_z}\|^2\right) = 2(x_i^+ + x_{o_z}^+), \tag{A.1c}$$

$$\nabla_{x_i}(\eta_{ij,c}) = \nabla_{x_i}\left(-\|p_i - p_j\|^2\right) = -2\left(x_i^+ - x_j^+\right), \tag{A.1d}$$

$$\frac{\partial b_{ij,a}}{\partial \eta_{ij,a}} = \begin{cases} \frac{\phi_{i,a}}{\partial \eta_{ij,a}}, & 0 \le \eta_{ij,a} < d_i^2 - \underline{d}_{ij,a}^2, \\ 0, & d_i^2 - \underline{d}_{ij,a}^2 \le \eta_{ij,a}, \end{cases}, \tag{A.1e}$$

$$\frac{\partial b_{iz,o}}{\partial \eta_{iz,o}} = \begin{cases} \frac{\phi_{i,o}}{\partial \eta_{iz,o}}, & 0 \le \eta_{iz,o} < d_i^2 - \underline{d}_{iz,o}^2, \\ 0, & d_i^2 - \underline{d}_{iz,o}^2 \le \eta_{iz,o}, \end{cases}, \tag{A.1f}$$

$$\frac{\partial b_{ij,c}}{\partial \eta_{ij,c}} = \begin{cases} 0, & \eta_{ij,c} < 0, \\ \frac{\phi_{i,c}}{\partial \eta_{ij,c}}, & 0 \le \eta_{ij,c} < d_i^2 - \underline{d}_{ij,a}^2, \\ 0, & d_i^2 - \underline{d}_{ij,a}^2 \le \eta_{ij,c}, \end{cases}, \tag{A.1g}$$

$$\nabla_{x_i}(b_{ij,a}) = 2\left(\frac{\partial b_{ij,a}}{\partial \eta_{ij,a}}\right)\left(x_i^+ - x_j^+\right), \tag{A.1h}$$

$$\nabla_{x_i}(b_{iz,o}) = 2\left(\frac{\partial b_{iz,o}}{\partial \eta_{iz,o}}\right)x_i^+, \tag{A.1i}$$

$$\nabla_{x_i}(b_{ij,c}) = -2\left(\frac{\partial b_{ij,c}}{\partial \eta_{ij,c}}\right)\left(x_i^+ - x_j^+\right), \tag{A.1j}$$

$$\nabla_{x_i}(b_{iw}) = -x_i^+, \tag{A.1k}$$

$$\nabla_{x_i}(b_{J_i}) = [0, 0, 0, 0, -2\sin(\theta_i), 0]^\top. \tag{A.1l}$$

Appendix B

Proofs of Chapter 5

B.1 Proof of Lemma 5.1

Proof. For every $e_1, e_2 \in E_i, u \in \mathcal{U}_i, i \in \mathcal{V}$, the following holds:

$$\begin{aligned}
|F_i(e_1,u) - F_i(e_2,u)| &= |e_1^\top Q_i e_1 + u^\top R_i u - e_2^\top Q_i e_2 - u^\top R_i u| \\
&= |e_1^\top Q_i e_1 - e_2^\top Q_i e_2| \\
&= |e_1^\top Q_i e_1 + e_1^\top Q_i e_2 - e_1^\top Q_i e_2 - e_2^\top Q_i e_2| \\
&= |e_1^\top Q_i (e_1 - e_2) - e_2^\top Q_i (e_1 - e_2)| \\
&\leq |e_1^\top Q_i (e_1 - e_2)| + |e_2^\top Q_i (e_1 - e_2)|. \qquad \text{(B.1)}
\end{aligned}$$

By employing the property that:

$$|x^\top A y| \leq \sigma_{\max}(A)\|x\|\|y\|, \forall\, x, y \in \mathbb{R}^n, A \in \mathbb{R}^{n \times n}, \qquad \text{(B.2)}$$

(B.1) is written as:

$$\begin{aligned}
|F_i(e_1,u) - F_i(e_2,u)| &\leq \sigma_{\max}(Q_i)\|e_1\|\|e_1 - e_2\| + \sigma_{\max}(Q_i)\|e_2\|\|e_1 - e_2\| \\
&= \sigma_{\max}(Q_i)(\|e_1\| + \|e_2\|)\|e_1 - e_2\| \\
&= \sigma_{\max}(Q_i)\left[\sup_{e_1,e_2 \in E_i}\{\|e_1\| + \|e_2\|\}\right]\|e_1 - e_2\| \\
&= 2\sigma_{\max}(Q_i)\left[\sup_{e_i \in E_i}\{\|e_i\|\}\right]\|e_1 - e_2\| \\
&= \left[2\bar{\varepsilon}_i \sigma_{\max}(Q_i)\right]\|e_1 - e_2\|,
\end{aligned}$$

which completes the proof. □

B.2 Proof of Lemma 5.2

Proof. Let us denote by:

$$
\begin{aligned}
u_i(\cdot) &\triangleq u_i(s; e(t_{k_z})),\\
e_i(s) &\triangleq e_i(s; u_i(\cdot), e_i(t_{k_z})).
\end{aligned}
$$

the control input and real trajectory of the system (5.1) for $s \in [t_{k_z}, t_{k_z} + T_z]$. Also, denote for sake of simplicity:

$$
\hat{e}_i(s) \triangleq \hat{e}_i(s; u_i(\cdot), e_i(t_{k_z})).
$$

the corresponding estimated trajectory. By integrating (5.1), (5.11b) for the time interval $[t_{k_z}, t_{k_z} + s]$ we have the following:

$$
\begin{aligned}
e_i(s) &= e_i(t_{k_z}) + \int_{t_{k_z}}^{s} \left[g_i(e_i(s'), \bar{x}_i(s'), u_i(\cdot))\right] ds',\\
\hat{e}_i(s) &= e_i(t_{k_z}) + \int_{t_{k_z}}^{s} \left[g_i(\hat{e}_i(s'), \hat{\bar{x}}_i(s'), u_i(\cdot))\right] ds',
\end{aligned}
$$

respectively. Then, we have that:

$$
\begin{aligned}
&\|e_i(s) - \hat{e}_i(s)\|\\
&= \left\| \int_{t_{k_z}}^{s} \left[g(e_i(s'), \bar{x}_i(s'), u_i(\cdot))\right] ds' - \int_{t_{k_z}}^{s} \left[g(\hat{e}_i(s'), \hat{\bar{x}}_i(s'), u_i(\cdot))\right] ds' \right\|\\
&= \left\| \int_{t_{k_z}}^{s} \left[f(e_i(s'), \bar{x}_i(s')) + u_i(s') - f(\hat{e}_i(s'), \hat{\bar{x}}_i(s')) - u_i(s')\right] ds' \right\|\\
&= \left\| \int_{t_{k_z}}^{s} \left[f(e_i(s'), \bar{x}_i(s')) - f(\hat{e}_i(s'), \hat{\bar{x}}_i(s'))\right] ds' \right\|\\
&\leq \int_{t_{k_z}}^{s} \left\| f(e_i(s'), \bar{x}_i(s')) - f(\hat{e}_i(s'), \hat{\bar{x}}_i(s')) \right\| ds'\\
&= \int_{t_{k_z}}^{s} \left\| f(e_i(s'), \bar{x}_i(s')) - f(\hat{e}_i(s'), \bar{x}_i(s')) + f(\hat{e}_i(s'), \bar{x}_i(s')) - f(\hat{e}_i(s'), \hat{\bar{x}}_i(s')) \right\| ds'\\
&\leq \int_{t_{k_z}}^{s} \left\| f(e_i(s'), \bar{x}_i(s')) - f(\hat{e}_i(s'), \bar{x}_i(s')) \right\| ds'\\
&\qquad\qquad + \int_{t_{k_z}}^{s} \left\| f(\hat{e}_i(s'), \bar{x}_i(s')) - f(\hat{e}_i(s'), \hat{\bar{x}}_i(s')) \right\| ds'.
\end{aligned}
$$

By using the bounds of (5.3a)-(5.3b) we obtain:

$$
\|e_i(s) - \hat{e}_i(s)\| \leq \int_{t_{k_z}}^{s} L_i \left\| e_i(s') - \hat{e}_i(s') \right\| ds' + \int_{t_{k_z}}^{s} \bar{L}_i \left\| \bar{x}_i(s') - \hat{\bar{x}}_i(s') \right\| ds'. \quad \text{(B.3)}
$$

The following property holds:

$$\|\bar{x}_i - \hat{\bar{x}}_i\| = \left(\sum_{j\in\mathcal{N}_i} \|x_j - \hat{x}_j\|^2\right)^{\frac{1}{2}}, \forall i \in \mathcal{V}, j \in \mathcal{N}_i.$$

Then, by combining the last inequality with (5.17) from Property 2, we have that:

$$\|\bar{x}_i - \hat{\bar{x}}_i\| = \left[\sum_{j\in\mathcal{N}_i} \left(\sqrt{3}R\right)^2\right]^{\frac{1}{2}} = \left[N_i\left(\sqrt{3}R\right)^2\right]^{\frac{1}{2}} = R\sqrt{3N_i}, \forall i \in \mathcal{V}, j \in \mathcal{N}_i.$$

By combining the last result with (B.3) we get:

$$\begin{aligned}\|e_i(s) - \hat{e}_i(s)\| &\le \int_{t_{k_z}}^{s} L_i\big\|e_i(s') - \hat{e}_i(s')\big\|ds' + \int_{t_{k_z}}^{s} \bar{L}_i\sqrt{3}RN_i ds' \\ &= \int_{t_{k_z}}^{s} L_i\big\|e_i(s') - \hat{e}_i(s')\big\|ds' + \sqrt{3}R\bar{L}_iN_i(s - t_{k_z}). \end{aligned} \tag{B.4}$$

By employing the Gronwall-Bellman from Lemma 2.1, (B.4) becomes:

$$\begin{aligned}&\|e_i(s) - \hat{e}_i(s)\| \\ &\le R\sqrt{3N_i}L_i\bar{L}_i\int_{t_{k_z}}^{s}(s' - t_{k_z})\exp\left[\int_{s'}^{s} L_i ds''\right]ds' + 2R\sqrt{3N_i}\bar{L}_i(s - t_{k_z}) \\ &= R\sqrt{3N_i}\bar{L}_i\int_{t_{k_z}}^{s}(s' - t_{k_z})e^{L_i(-s'+s)}ds' + \sqrt{3}R\bar{L}_iN_i(s - t_{k_z}) \\ &= -R\sqrt{3N_i}\bar{L}_i(s - t_{k_z}) + R\sqrt{3N_i}\bar{L}_i(s - t_{k_z}) + \sqrt{3}R\bar{L}_iN_i\int_{t_{k_z}}^{s} e^{L_i(-s'+s)}ds' \\ &= R\sqrt{3N_i}\bar{L}_i\int_{t_{k_z}}^{s} e^{L_i(-s'+s)}ds' \\ &= -\frac{R\sqrt{3N_i}\bar{L}_i}{L_i}\left[1 - e^{L_i(s-t_{k_z})}\right] \\ &= \frac{R\sqrt{3N_i}\bar{L}_i}{L_i}\left[e^{L_i(s-t_{k_z})} - 1\right]. \end{aligned} \tag{B.5}$$

which leads to the conclusion of the proof. □

B.3 Proof of Property 5.2

Proof. Let $s \in [t_{k_z}, t_{k_z} + T_z]$. Let us also define:

$$z_i(s) \triangleq e_i(s) - \hat{e}_i(s; u_i(s; e(t_{k_z})), e_i(t_{k_z})).$$

Then, according to Lemma 5.2, for $s \in [t_{k_z}, t_{k_z} + T_z]$, we get:

$$\|z_i(s)\| = \|e_i(s) - \hat{e}_i(s; u_i(s; e_i(t_{k_z})), e_i(t_{k_z}))\| \leq \rho_i(s - t_{k_z}).$$

Hence, $z_i \in B^i_{s-t_{k_z}}$, which implies that: $-z_i \in B^i_{s-t_{k_z}}$. The following implications hold:

$$\begin{aligned}
\hat{e}_i(s; u_i(s; e_i(t_{k_z})), e_i(t_{k_z})) &\in E_i \sim B^i_{s-t_{k_z}} \\
\Rightarrow e_i(s) - z_i &\in E \sim B^i_{s-t_{k_z}} \\
\Rightarrow e_i(s) + (-z_i) &\in E \sim B^i_{s-t_{k_z}} \\
\Rightarrow e_i(s) &\in E_i, \forall\ s \in [t_{k_z}, t_{k_z} + T_z],
\end{aligned}$$

which concludes the proof. □

B.4 Proof of Lemma 5.3

Proof. For every $e_1, e_2 \in \Phi_i$, $i \in \mathcal{V}$, the following holds:

$$\begin{aligned}
|V_i(e_1, u) - V_i(e_2, u)| &= |e_1^\top P_i e_1 + u^\top R_i u - e_2^\top P_i e_2 - u^\top P_i u| \\
&= |e_1^\top P_i e_1 - e_2^\top P_i e_2| \\
&= |e_1^\top P_i e_1 + e_1^\top P_i e_2 - e_1^\top P_i e_2 - e_2^\top P_i e_2| \\
&= |e_1^\top P_i (e_1 - e_2) - e_2^\top P_i (e_1 - e_2)| \\
&\leq |e_1^\top P_i (e_1 - e_2)| + |e_2^\top P_i (e_1 - e_2)|.
\end{aligned} \tag{B.6}$$

By employing property (B.6) is written as:

$$\begin{aligned}
|F_i(e_1, u) - F_i(e_2, u)| &\leq \sigma_{\max}(P_i)\|e_1\|\|e_1 - e_2\| + \sigma_{\max}(P_i)\|e_2\|\|e_1 - e_2\| \\
&= \sigma_{\max}(P_i)(\|e_1\| + \|e_2\|)\|e_1 - e_2\| \\
&= \sigma_{\max}(P_i)\left[\sup_{e_1, e_2 \in \Phi_i} \{\|e_1\| + \|e_2\|\}\right]\|e_1 - e_2\| \\
&= 2\sigma_{\max}(P_i)\left[\sup_{e_i \in \Phi_i} \{\|e_i\|\}\right]\|e_1 - e_2\| \\
&= [2\alpha_{i,1}\sigma_{\max}(P_i)]\,\|e_1 - e_2\|,
\end{aligned}$$

which completes the proof. □

B.5 Proof of Lemma 5.4

Proof. For every $s \geq t_{k_{z+1}}$, $L_i > 0$ the following inequality holds:

$$\left[e^{L_i(t_{k_{z+1}} - t_{k_z})} - 1\right] + \left[e^{L_i(s - t_{k_{z+1}})} - 1\right] \leq \left[e^{L_i(s - t_{k_z})} - 1\right],$$

which implies that:

$$\widetilde{\rho}_i\left[e^{L_i(t_{k_{z+1}}-t_{k_z})}-1\right]+\widetilde{\rho}_i\left[e^{L_i(s-t_{k_{z+1}})}-1\right]\leq\widetilde{\rho}_i\left[e^{L_i(s-t_{k_z})}-1\right]. \tag{B.7}$$

or equivalently

$$\rho_i(t_{k_{z+1}}-t_{k_z})+\rho_i(s-t_{k_{z+1}})\leq\rho_i(s-t_{k_z}). \tag{B.8}$$

Let us consider $\phi\in B^i_{s-t_{k_{z+1}}}$. Then, it holds $\|\phi\|\leq\rho_i(s-t_{k_{z+1}})$. Let us denote $z=x-y+\phi$. It is clear that:

$$\begin{aligned}\|z\|&\leq\|x-y+\phi\|\\&\leq\|x-y\|+\|\phi\|\\&\leq\rho_i(t_{k_{z+1}}-t_{k_z})+\rho_i(s-t_{k_{z+1}}).\end{aligned} \tag{B.9}$$

By employing (B.8), (B.9) becomes:

$$\|z\|\leq\rho_i(s-t_{k_z}),$$

which implies that $z\in B^i_{s-t_{k_z}}$. We have that:

$$\begin{aligned}x+(-z)&=y+(-\phi),\\x\in E_{s-t_{k_z}}&=E\sim B_{s-t_{k_z}},\\-z&\in B^i_{s-t_{k_z}},\\-\rho&\in B^i_{s-t_{k_{z+1}}},\end{aligned}$$

which implies that $y\in E_{s-t_{k_{z+1}}}=E\sim B_{s-t_{k_{z+1}}}$. □

B.6 Proof of Lemma 5.5

Proof. Let $s\geq t_{k_z}$. The following equalities hold:

$$\begin{aligned}&\|\hat{e}_i(s;u_i(\cdot),e_i(t_{k_{z+1}}))-\hat{e}_i(s;u_i(\cdot),e_i(t_{k_z}))\|\\&=\Bigg\|\hat{e}_i(s;u_i(\cdot),e_i(t_{k_{z+1}}))+\int_{t_{k_{z+1}}}^{s}g_i(\hat{e}_i(s'),\hat{\hat{x}}_i(s'),u_i(\cdot))ds'\\&-\hat{e}_i(t_{k_z};u_i(\cdot),e_i(t_{k_z}))-\int_{t_{k_z}}^{s}g_i(\hat{e}_i(s'),\hat{\hat{x}}_i(s'),u_i(\cdot)))ds\Bigg\|\end{aligned}$$

$$
\begin{aligned}
&= \Bigg\| e_i(t_{k_{z+1}}) - e_i(t_{k_z}) - \int_{t_{k_z}}^{s} g_i(\hat{e}_i(s'), \hat{\bar{x}}_i(s'), u_i(\cdot))ds' \\
&\qquad\qquad - \int_{s}^{t_{k_{z+1}}} g_i(\hat{e}_i(s'), \hat{\bar{x}}_i(s'), u_i(\cdot))ds' \Bigg\| \\
&= \left\| e_i(t_{k_{z+1}}) - e_i(t_{k_z}) - \int_{t_{k_z}}^{t_{k_{z+1}}} g_i(\hat{e}_i(s'), \hat{\bar{x}}_i(s'), u_i(\cdot))ds' \right\| \\
&= \left\| e_i(t_{k_{z+1}}) - e_i(t_{k_z}) - \int_{t_{k_z}}^{t_{k_{z+1}}} \frac{d}{dt} \left[\hat{e}_i(s'; u_i(\cdot), e_i(t_{k_z})\right] ds' \right\| \\
&= \left\| e_i(t_{k_{z+1}}) - e_i(t_{k_z}) - \hat{e}(t_{k_{z+1}}; u(\cdot), e_i(t_{k_z})) + \hat{e}_i(t_{k_z}; u_i(\cdot), e_i(t_{k_z})) \right\| \\
&= \left\| e_i(t_{k_{z+1}}) - e_i(t_{k_z}) - \hat{e}(t_{k_{z+1}}; u(\cdot), e_i(t_{k_z})) + e_i(t_{k_z}) \right\| \\
&= \left\| e_i(t_{k_{z+1}}) - \hat{e}_i(t_{k_{z+1}}; u_i(\cdot), e_i(t_{k_z})) \right\|,
\end{aligned}
$$

which, by employing Lemma 5.2 for $s = t_{k_{z+1}}$, becomes:

$$
\|\hat{e}_i(s; u(\cdot), e_i(t_{k_{z+1}})) - \hat{e}_i(s; u_i(\cdot), e_i(t_{k_z}))\| \le \rho_i(t_{k_{z+1}} - t_{k_z}) = \rho_i(h),
$$

since $t_{k_{z+1}} - t_{k_z} = h$, which concludes the proof. □

B.7 Proof of Theorem 5.1

Proof. The proof consists of two parts: in the first part it is established that initial feasibility implies feasibility afterwards. Based on this result it is then shown that the error $e_i(t)$ converges to the terminal set $\mathcal{E}_i$.

Feasibility Analysis: Consider any sampling time instant for which a solution exists, say t_{k_z}. In between t_{k_z} and $t_{k_{z+1}}$, the optimal control input $\hat{u}_i^\star(s; e_i(t_{k_z})), s \in [t_{k_z}, t_{k_{z+1}})$ is implemented. The remaining part of the optimal control input

$$
\hat{u}_i^\star(s; e_i(t_{k_z})), s \in [t_{k_{z+1}}, t_{k_z} + T_z],
$$

satisfies the state and input constraints E_i,$\mathcal{U}_i$, respectively. Furthermore, since the problem is feasible at time t_{k_z}, it holds that:

$$
\hat{e}_i(s; \hat{u}^\star(s; e_i(t_{k_z})), e_i(t_{k_z})) \in E^i_{s-t_{k_z}}, \tag{B.10a}
$$

$$
\hat{e}_i(t_{k_z} + T; \hat{u}_i^\star(s; e_i(t_{k_z})), e_i(t_{k_z})) \in \mathcal{E}_i, \tag{B.10b}
$$

for $s \in [t_{k_z}, t_{k_z} + T_z]$. By using Property 1, (B.10a) implies also that

$$
e_i(s; \hat{u}_i^\star(s; e_i(t_{k_z})), e_i(t_{k_z})) \in E_i.
$$

We know also from Assumption 5.4 that for all $e_i \in \mathcal{E}_i$, there exists at least one control input $u_{f,i}(\cdot)$ that renders the set $\mathcal{E}_i$ invariant over h. Picking any such input, a feasible control input $\bar{u}_i(\cdot; e_i(t_{k_{z+1}}))$, at time instant $t_{k_{z+1}}$, may be the following:

$$\bar{u}_i(s; e(t_{k_{z+1}})) = \begin{cases} \hat{u}_i^\star(s; e_i(t_{k_z})), & s \in [t_{k_{z+1}}, t_{k_z} + T_{z+1}], \\ u_{f,i}(t_{k_z} + T_{z+1}; \hat{u}^\star(\cdot), e(t_i))), & s \in [t_{k_z} + T_{z+1}, t_{k_z} + T_z]. \end{cases} \tag{B.11}$$

For the time intervals it holds that (see Fig. 5.4):

$$t_{k_z} + T_{z+1} = t_{k_z} + T_z - h = t_{k_z} + T - h.$$

For the feasibility of the ROCP, we have to prove the following three statements for every $s \in [t_{k_{z+1}}, t_{k_z} + T_z]$:

1. $\bar{u}_i(s; e(t_{k_{z+1}})) \in \mathcal{U}_i$.

2. $\hat{e}_i(t_{k_z} + T_z; \bar{u}(s; e(t_{k_{z+1}})), e(t_{k_{z+1}})) \in \mathcal{E}_i$.

3. $\hat{e}_i(s; \bar{u}_i(s; e(t_{k_{z+1}})), e(t_{k_{z+1}})) \in E^i_{s - t_{k_{z+1}}}$.

Statement 1: From the feasibility of $\hat{u}_i^\star(s, e(t_{k_z}))$ and the fact that $u_{f,i}(e_i(\cdot)) \in \mathcal{U}_i$, for all $e_i(\cdot) \in \Phi_i$, it follows that:

$$\bar{u}_i(s; e_i(t_{k_{z+1}})) \in \mathcal{U}_i, \forall\ s \in [t_{k_{z+1}}, t_{k_z} + T_z].$$

Statement 2: We need to prove in this step that for every $s \in [t_{k_{z+1}}, t_{k_z} + T_z]$ it holds that $\hat{e}_i(t_{k_z} + T_z; \bar{u}_i(s; e_i(t_{k_{z+1}}))), e_i(t_{k_{z+1}})) \in \mathcal{E}_i$. Since $V_i(\cdot)$ is Lipschitz continuous, we get:

$$\begin{aligned} V_i(\hat{e}_i(t_{k_z} + T_{z+1}; \bar{u}_i(\cdot), e_i(t_{k_{z+1}}))) - V_i(\hat{e}_i(t_{k_z} + T_{z+1}; \bar{u}_i(\cdot), e_i(t_{k_z}))) \leq \\ L_{V_i} \|\hat{e}_i(t_{k_z} + T_{z+1}; \bar{u}_i(\cdot), e_i(t_{k_{z+1}})) - \hat{e}(t_{k_z} + T_{z+1}; \bar{u}_i(\cdot), e(t_{k_z}))\|, \end{aligned} \tag{B.12}$$

for the same control input $\bar{u}_i(\cdot) = u_i^\star(s; e_i(t_{k_z}))$. By employing Lemma 5.5 for $\alpha = t_{k_z} + T_{z+1}$ and $u(\cdot) = \bar{u}_i(\cdot) = u_i^\star(s; e_i(t_{k_z}))$, we have that:

$$\begin{aligned} \|\hat{e}_i(t_{k_z} + T_z; \bar{u}_i(\cdot), e_i(t_{k_{z+1}})) - \hat{e}_i(t_{k_z} + T_{z+1}; \bar{u}_i(\cdot), e(t_{k_z}))\| \\ \leq \rho_i(t_{k_{z+1}} - t_{k_z}) = \rho_i(h). \end{aligned} \tag{B.13}$$

Note also that for the function $\rho_i(\cdot)$ the following implication holds:

$$h \leq T_z \Rightarrow \rho_i(h) \leq \rho_i(T_z).$$

By employing the latter result, (B.13) becomes:

$$\begin{aligned} &\|\hat{e}_i(t_{k_z} + T_z; \bar{u}_i(\cdot), e_i(t_{i+1})) \\ &\quad - \hat{e}_i(t_{k_z} + T_{z+1}; \bar{u}_i(\cdot), e(t_{k_z}))\| \leq \rho_i(h) \leq \rho_i(T_z). \end{aligned} \tag{B.14}$$

By combining (B.14) and (B.12) we get:

$$V_i(\hat{e}_i(t_{k_z} + T_{z+1}; \bar{u}_i(\cdot), e_i(t_{k_{z+1}}))) - \\ V_i(\hat{e}_i(t_{k_z} + T_{z+1}; \bar{u}_i(\cdot), e_i(t_{k_z}))) \leq L_{V_i} \rho_i(T_z),$$

or equivalently:

$$V_i(\hat{e}_i(t_{k_z} + T_{z+1}; \bar{u}_i(\cdot), e_i(t_{k_{z+1}}))) \leq \\ V_i(\hat{e}_i(t_{k_z} + T_{z+1}; \bar{u}_i(\cdot), e_i(t_{k_z}))) + L_{V_i} \rho_i(T_z). \tag{B.15}$$

By using (B.10b), we have that $\hat{e}_i(t_{k_z} + T_{z+1}; \bar{u}_i(\cdot), e_i(t_{k_z})) \in \mathcal{E}_i$. Then, (B.15) gives:

$$V_i(\hat{e}_i(t_{k_z} + T_{z+1}; \bar{u}_i(\cdot), e_i(t_{k_{z+1}}))) \leq \alpha_{2,i} + L_{V_i} \rho_i(T_z) \tag{B.16}$$

From (5.24) of the Theorem 1, we get equivalently:

$$\rho_i(T_z) \leq \frac{\alpha_{1,1} - \alpha_{2,i}}{L_{V_i}} \Leftrightarrow \alpha_{2,i} + L_{V_i} \rho_i(T_z) \leq \alpha_{1,i}. \tag{B.17}$$

By combining (B.16) and (B.17), we get:

$$V_i(\hat{e}_i(t_{k_z} + T_{z+1}; \bar{u}_i(\cdot), e_i(t_{k_{z+1}}))) \leq \alpha_{1,i},$$

which, from Assumption 5.4, implies that:

$$\hat{e}_i(t_{k_z} + T_{z+1}; \bar{u}_i(\cdot), e_i(t_{k_{z+1}})) \in \Phi_i. \tag{B.18}$$

But since $\bar{u}_i(\cdot)$ is chosen to be local admissible controller from Assumption 5.4, according to our choice of terminal set $\mathcal{E}_i$, (B.18) leads to:

$$\hat{e}_i(t_{k_z} + T_z; \bar{u}_i(\cdot), e_i(t_{k_{z+1}})) \in \mathcal{E}_i.$$

Thus, statement 2 holds.
Statement 3: By employing Lemma 5.5 for:

$$x = \hat{e}_i(s; \bar{u}_i(s; e(t_{k_z})), e(t_{k_z})) \in E^i_{s-t_i}, \\ y = \hat{e}_i(s; \bar{u}_i(s; e(t_{k_{z+1}})), e(t_{k_{z+1}})),$$

we get that:

$$\|y - x\| = \|\hat{e}(s; \bar{u}(s; e(t_{i+1})), e(t_{i+1})) \\ - \hat{e}(s; \bar{u}(s; e(t_i)), e(t_i)) \in E_{s-t_i}\| \leq \rho_i(h).$$

Furthermore, by employing Lemma 5.4 for $s \in [t_{k_{z+1}}, t_{k_z} + T_z]$ and the same x, y as previously we get that $y = \hat{e}_i(s; \bar{u}(s; e_i(t_{k_{z+1}})), e(t_{k_{z+1}})) \in E^i_{s-t_{k_{z+1}}}$, which according to Property 1, implies that $e_i(s; \bar{u}_i(s; e_i(t_{k_{z+1}})), e_i(t_{k_{z+1}})) \in E_i$. Thus, Statement 3 holds. Hence, the feasibility at time t_{k_z} implies feasibility at time $t_{k_{z+1}}$. Therefore, if the ROCP (5.11a) - (5.11d) is feasible at time t_{k_z}, i.e., it remains feasible for every $t \in [t_k, t_k + T]$.

Convergence Analysis: The second part involves proving convergence of the state e_i to the terminal set $\mathcal{E}_i$. In order to prove this, it must be shown that a proper value function is decreasing along the solution trajectories starting at a sampling time t_i. Consider the optimal value function $J_i^\star(e_i(t_{k_z}))$, as is given in (5.16), to be a Lyapunov-like function. Consider also the cost of the feasible control input, indicated by:

$$\bar{J}_i(e_i(t_{k_{z+1}})) \triangleq \bar{J}_i(e_i(t_{k_{z+1}}), \bar{u}_i(\cdot; e_i(t_{k_{z+1}}))), \tag{B.19}$$

where $t_{k_{z+1}} = t_{k_z} + h$. Define:

$$\bar{u}_1(s) \triangleq \bar{u}_i(s; e_i(t_{k_{z+1}})), \tag{B.20a}$$

$$\bar{e}_1(s) \triangleq \bar{e}_i(s; u_1(s), e_i(t_{k_{z+1}})), s \in [t_{k_{z+1}}, t_{k_z} + T], \tag{B.20b}$$

where $\bar{e}_1(s)$ stands for the predicted state e_i at time s, based on the measurement of the state e_i at time $t_{k_{z+1}}$, while using the feasible control input $\bar{u}_i(s; e(t_{k_{z+1}}))$ from (B.11). Let us also define the following terms:

$$\begin{aligned} \hat{u}_2(s) &\triangleq \hat{u}_i^\star(s; e_i(t_{k_z})), \\ \hat{e}_2(s) &\triangleq \hat{e}_i(s; \hat{u}_2(s), e_i(t_{k_z})), s \in [t_{k_z}, t_{k_z} + T - h]. \end{aligned} \tag{B.21}$$

where $\hat{e}_1(s)$ stands for the predicted state e_i at time s, based on the measurement of the state e_i at time t_{k_z}, while using the control input $\hat{u}_i(s; e(t_{k_z})), s \in [t_{k_z}, t_{k_z} + T - h]$ from (B.11). By employing (5.11a), (5.16) and (B.19), the difference between the optimal and feasible cost is given by:

$$\begin{aligned} \bar{J}(e_i(t_{k_{z+1}})) - J^\star(e_i(t_{k_z})) &= V_i(\bar{e}_1(t_{k_z} + T)) + \int_{t_{k_{z+1}}}^{t_{k_z}+T} \Big[F_i(\bar{e}_1(s), \bar{u}_1(s))\Big] ds \\ &\quad - V_i(\hat{e}_2(t_{k_z} + T - h)) - \int_{t_{k_z}}^{t_{k_z}+T-h} \Big[F_i(\hat{e}_2(s), \hat{u}_2(s))\Big] ds \\ &= V_i(\bar{e}_1(t_{k_z} + T)) + \int_{t_{k_{z+1}}}^{t_{k_{z+1}}+T-h} \Big[F_i(\bar{e}_1(s), \bar{u}_1(s))\Big] ds \\ &\quad + \int_{t_{k_z}+T-h}^{t_{k_z}+T} \Big[F_i(\bar{e}_1(s), \bar{u}_1(s))\Big] ds - V_i(\hat{e}_2(t_{k_z} + T - h)) \\ &\quad - \int_{t_{k_z}}^{t_{k_{z+1}}} \Big[F_i(\hat{e}_2(s), \hat{u}_2(s))\Big] ds - \int_{t_{k_{z+1}}}^{t_{k_z}+T-h} \Big[F_i(\hat{e}_2(s), \hat{u}_2(s))\Big] ds. \end{aligned} \tag{B.22}$$

Note that, from (B.11), the following holds:

$$\bar{u}_i(s; e_i(t_{k_{z+1}})) = \hat{u}_i^\star(s; e_i(t_{k_z})), \forall\ s \in [t_{k_{z+1}}, t_{k_z} + T - h]. \tag{B.23}$$

By combining (B.20a), (B.21) and (B.23), we have that:

$$\bar{u}_1(s) = \hat{u}_2(s) = \bar{u}_i(s; e_i(t_{k_{z+1}})) = \hat{u}_i^\star(s; e_i(t_{k_z})), \forall\ s \in [t_{k_{z+1}}, t_{k_z} + T - h], \quad \text{(B.24)}$$

By applying the last result and the fact that $F_i(e, u)$ is Lipschitz, the following holds:

$$\begin{aligned}
&\int_{t_{k_{z+1}}}^{t_{k_z}+T-h} \Big[F_i(\bar{e}_1(s), \bar{u}_1(s))\Big] ds - \int_{t_{k_{z+1}}}^{t_{k_z}+T-h} \Big[F_i(\hat{e}_2(s), \hat{u}_2(s))\Big] ds \\
&= \int_{t_{k_{z+1}}}^{t_{k_z}+T-h} \Big[F_i(\bar{e}_1(s), \bar{u}_1(s)) - F_i(\hat{e}_2(s), \hat{u}_2(s))\Big] ds \\
&= \int_{t_{k_{z+1}}}^{t_{k_z}+T-h} \Big[F_i(\bar{e}_1(s), \bar{u}_i(s; e_i(t_{k_{z+1}}))) - F_i(\hat{e}_2(s), \bar{u}_i(s; e_i(t_{k_{z+1}})))\Big] ds \\
&\le \left\| \int_{t_{k_{z+1}}}^{t_{k_z}+T-h} \Big[F_i(\bar{e}_1(s), \bar{u}_i(\cdot)) - F_i(\hat{e}_2(s), \bar{u}_i(\cdot))\Big] ds \right\| \\
&\le \int_{t_{k_{z+1}}}^{t_{k_z}+T-h} \Big\|F_i(\bar{e}_1(s), \bar{u}_i(\cdot)) - F_i(\hat{e}_2(s), \bar{u}_i(\cdot))\Big\| ds \\
&\le L_{F_i} \int_{t_{k_{z+1}}}^{t_{k_z}+T-h} \|\bar{e}_1(s) - \hat{e}_2(s)\| \, ds. \qquad \text{(B.25)}
\end{aligned}$$

By employing the fact that $\forall s \in [t_{k_{z+1}}, t_{k_z} + T - h]$ the following holds:

$$\bar{e}_i(s; \bar{u}_i(\cdot), e_i(t_{k_{z+1}})) = \hat{e}_i(s; \bar{u}_i(\cdot), e_i(t_{k_z})), \quad \text{(B.26)}$$

the term $\|\bar{e}_1(s) - \hat{e}_2(s)\|$ can be written as:

$$\begin{aligned}
\|\bar{e}_1(s) - \hat{e}_2(s)\| &= \|\bar{e}_i(s; \bar{u}_i(\cdot), e_i(t_{k_{z+1}})) - \hat{e}_i(s; \hat{u}_i(\cdot), e_i(t_{k_z}))\| \\
&= \Bigg\| \bar{e}_i(t_{k_{z+1}}; \bar{u}_i(\cdot), e_i(t_{k_{z+1}})) + \int_{t_{k_{z+1}}}^{s} g_i(\bar{e}_i(s'), \hat{\bar{x}}_i(s'), \bar{u}_i(\cdot)) ds' \\
&\quad - \hat{e}_i(t_{k_z}; \hat{u}_i(\cdot), e_i(t_{k_z})) - \int_{t_{k_z}}^{t_{k_{z+1}}} g_i(\hat{e}_i(s), \hat{\bar{x}}_i(s'), \hat{u}_i(\cdot)) ds \\
&\qquad - \int_{t_{k_{z+1}}}^{s} g_i(\hat{e}_i(s'), \hat{\bar{x}}_i(s'), \bar{u}_i(\cdot)) ds' \Bigg\|
\end{aligned}$$

$$
\begin{aligned}
&\leq \left\| e_i(t_{k_{z+1}}) - e_i(t_{k_z}) - \int_{t_{k_z}}^{t_{k_{z+1}}} g_i(\hat{e}_i(s'), \hat{\bar{x}}_i(s'), \hat{u}_i(\cdot)) ds' \right\| \\
&+ \left\| \int_{t_{k_{z+1}}}^{s} g_i(\bar{e}_i(s'), \hat{\bar{x}}_i(\cdot), \bar{u}_i(\cdot)) ds' - \int_{t_{k_{z+1}}}^{s} g_i(\hat{e}_i(s'), \hat{\bar{x}}_i(\cdot), \bar{u}_i(\cdot)) ds' \right\| \\
&= \left\| e_i(t_{k_{z+1}}) - e_i(t_{k_z}) - \int_{t_{k_z}}^{t_{k_{z+1}}} \frac{d}{dt} \left[\hat{e}_i(s; \hat{u}_i(\cdot), e_i(t_{k_z})) \right] ds \right\| \\
&+ \left\| \int_{t_{k_{z+1}}}^{s} \frac{d}{dt} \left[\bar{e}_i(s'; \bar{u}_i(\cdot), e_i(t_{k_{z+1}})) \right] ds' - \int_{t_{k_{z+1}}}^{s} \frac{d}{dt} \left[\hat{e}_i(s'; \bar{u}_i(\cdot), e_i(t_{k_z})) \right] ds' \right\| \\
&= \left\| e_i(t_{k_{z+1}}) - e_i(t_{k_z}) - \hat{e}_i(t_{k_{z+1}}; \hat{u}_i(\cdot), e_i(t_{k_z})) + \hat{e}_i(t_{k_z}; \hat{u}_i(\cdot), e_i(t_{k_z})) \right\| \\
&+ \left\| \bar{e}_i(s; \bar{u}_i(\cdot), e_i(t_{k_{z+1}})) - \bar{e}_i(t_{i+1}; \bar{u}_i(\cdot), e_i(t_{k_{z+1}})) \right. \\
&\left. - \hat{e}_i(s; \bar{u}_i(\cdot), e_i(t_{k_z})) + \hat{e}_i(t_{k_{z+1}}; \bar{u}_i(\cdot), e_i(t_{k_z})) \right\| \\
&= \left\| e_i(t_{k_{z+1}}) - e_i(t_{k_z}) - \hat{e}_i(t_{k_{z+1}}; \hat{u}_i(\cdot), e_i(t_{k_z})) + e_i(t_{k_z}) \right\| \\
&= \left\| e_i(t_{k_{z+1}}) - \hat{e}_i(t_{k_{z+1}}; \hat{u}_i(\cdot), e_i(t_{k_z})) \right\|,
\end{aligned}
$$

which, by employing Lemma 5.2, leads to:

$$
\|\bar{e}_1(s) - \hat{e}_2(s)\| \leq \rho_i(t_{k_{z+1}} - t_{k_z}) = \rho_i(h).
$$

By combining the last result with (B.25) we get:

$$
\begin{aligned}
&\int_{t_{k_{z+1}}}^{t_{k_z}+T-h} \Big[F_i(\bar{e}_1(s), \bar{u}_1(s)) \Big] ds - \int_{t_{k_{z+1}}}^{t_{k_z}+T-h} \Big[F_i(\hat{e}_2(s), \hat{u}_2(s)) \Big] ds \\
&\qquad \leq L_{F_i} \int_{t_{k_{z+1}}}^{t_{k_z}+T-h} \rho_i(h) ds = (T - 2h) \rho_i(h) L_{F_i}. \qquad \text{(B.27)}
\end{aligned}
$$

By combining the last result with (B.25), (B.22) becomes:

$$
\begin{aligned}
\bar{J}(e_i(t_{k_{z+1}})) - J^{\star}(e_i(t_{k_z})) &\leq (T - 2h) \rho_i(h) L_{F_i} \\
&+ V_i(\bar{e}_1(t_{k_z} + T)) - V_i(\hat{e}_2(t_{k_z} + T - h)) \\
&+ \int_{t_{k_z}+T-h}^{t_{k_z}+T} \Big[F_i(\bar{e}_1(s), \bar{u}_1(s)) \Big] ds - \int_{t_{k_z}}^{t_{k_{z+1}}} \Big[F_i(\hat{e}_2(s), \hat{u}_2(s)) \Big] ds. \qquad \text{(B.28)}
\end{aligned}
$$

By integrating inequality (5.21) from $t_{k_z} + T - h$ to $t_{k_z} + T$ and we get the following:

$$
\begin{aligned}
&\int_{t_{k_z}+T-h}^{t_{k_z}+T} \Big[\frac{\partial V}{\partial e} \cdot g_i(\bar{e}_1(s), \hat{\bar{x}}_i(s), \bar{u}_1(s)) + F_i(\bar{e}_1(s), \bar{u}_1(s)) \Big] ds \leq 0 \\
\Leftrightarrow & V_i(\bar{e}_1(t_{k_z} + T) - V_i(\bar{e}_1(t_{k_z} + T - h)) + \int_{t_{k_z}+T-h}^{t_{k_z}+T} \Big[F_i(\bar{e}_1(s), \bar{u}_1(s)) \Big] ds \leq 0,
\end{aligned}
$$

which by adding and subtracting the term $V_i(\hat{e}_2(t_{k_z}+T-h))$ becomes:

$$V_i(\bar{e}_1(t_{k_z}+T)-V_i(\hat{e}_2(t_{k_z}+T-h))+\int_{t_{k_z}+T-h}^{t_{k_z}+T}\Big[F_i(\bar{e}_1(s),\bar{u}_1(s))\Big]ds \\ \leq V_i(\bar{e}_1(t_{k_z}+T-h))-V_i(\hat{e}_2(t_{k_z}+T-h)).$$

By employing the property $y\leq|y|, \forall y\in\mathbb{R}$, we get:

$$V_i(\bar{e}_1(t_{k_z}+T)-V_i(\hat{e}_2(t_{k_z}+T-h))+\int_{t_{k_z}+T-h}^{t_{k_z}+T}\Big[F_i(\bar{e}_1(s),\bar{u}_1(s))\Big]ds \\ \leq \left|V_i(\bar{e}_1(t_{k_z}+T-h))-V_i(\hat{e}_2(t_{k_z}+T-h))\right|. \tag{B.29}$$

By employing Lemma 5.3, we have that:

$$|V_i(\bar{e}_1(t_{k_z}+T-h))-V_i(\hat{e}_2(t_{k_z}+T-h))|\leq \\ L_{V_i}\|\bar{e}_1(t_{k_z}+T-h)-\hat{e}_2(t_{k_z}+T-h)\|,$$

which by employing Lemma 5.5 and (B.24), becomes:

$$|V_i(\bar{e}_1(t_{k_z}+T-h))-V_i(\hat{e}_2(t_{k_z}+T-h))|\leq L_{V_i}\rho_i(t_{k_{z+1}}-t_{k_z})=\rho_i(h)L_{V_i}.$$

By combining the last result with (B.29), we get:

$$V_i(\bar{e}_1(t_{k_z}+T)-V_i(\hat{e}_2(t_{k_z}+T-h)) \\ +\int_{t_{k_z}+T-h}^{t_{k_z}+T}\Big[F_i(\bar{e}_1(s),\bar{u}_1(s))\Big]ds\leq\rho_i(h)L_{V_i}.$$

The last inequality along with (B.28) leads to:

$$\bar{J}(e(t_{k_{z+1}}))-J^\star(e(t_{k_z}))\leq(T-2h)\rho_i(h)L_{F_i}+\rho_i(h)L_{V_i} \\ -\int_{t_{k_z}}^{t_{k_{z+1}}}\Big[F_i(\hat{e}_2(s),\hat{u}_2(s))\Big]ds. \tag{B.30}$$

By substituting $e_i=\hat{e}_2(s), u_i=\hat{u}_2(s)$ in (5.12) we get $F_i(\hat{e}_2(s),\hat{u}_2(s))\geq m_i\|\hat{e}_2(s)\|^2$, or equivalently:

$$\int_{t_{k_z}}^{t_{k_{z+1}}}\Big[F_i(\hat{e}_2(s),\hat{u}_2(s))\Big]ds\geq\underline{m}_i\int_{t_{k_z}}^{t_{k_{z+1}}}\|\hat{e}_2(s)\|^2ds \\ \Leftrightarrow-\int_{t_{k_z}}^{t_{k_{z+1}}}\Big[F(\hat{e}_2(s),\hat{u}_2(s))\Big]ds\leq-\underline{m}_i\int_{t_{k_z}}^{t_{k_{z+1}}}\|\hat{e}_2(s)\|^2ds.$$

By combining the last result with (B.30), we get:

$$\bar{J}_i(e(t_{k_{z+1}}))-J_i^\star(e(t_{k_z}))\leq(T-2h)\rho_i(h)L_{F_i}+\rho_i(h)L_{V_i} \\ -\underline{m}_i\int_{t_{k_z}}^{t_{k_{z+1}}}\|\hat{e}_2(s)\|^2ds \tag{B.31}$$

It is clear that the optimal solution at time $t_{k_{z+1}}$ i.e., $J^\star(e_i(t_{k_{z+1}}))$ will not be worse than the feasible one at the same time i.e. $\bar{J}(e_i(t_{k_{z+1}}))$. Therefore, (B.31) implies:

$$\begin{aligned} J_i^\star(e_i(t_{k_{z+1}})) - J_i^\star(e_i(t_{k_z})) \leq (T-2h)\rho_i(h)L_{F_i} + \rho_i(h)L_{V_i} \\ - \underline{m}_i \int_{t_{k_z}}^{t_{k_{z+1}}} \|\hat{e}_2(s)\|^2 ds, \end{aligned}$$

which is equivalent to:

$$\begin{aligned} J_i^\star(e_i(t_{k_{z+1}})) - J_i^\star(e_i(t_{k_z})) \leq -m_i \int_{t_{k_z}}^{t_{k_{z+1}}} \|\hat{e}_i(s;\hat{u}_i^\star(s;e_i(t_{k_z})), e_i(t_{k_z}))\|^2 ds \\ + (T-2h)\rho_i(h)L_{F_i} + \rho_i(h)L_{V_i}. \end{aligned}$$

which, according to (2.1), is in the form:

$$J_i^\star(e_i(t_{k_{z+1}})) - J_i^\star(e_i(t_{k_z})) \leq -\alpha(\|e_i\|) + \sigma(\|\bar{x}_i\|). \tag{B.32}$$

Thus, the optimal cost J has been proven to be decreasing, and according to Definition 2.6 and Theorem 2.7, the closed loop system is ISS stable. Therefore, the closed loop trajectories converges to the closed set $\mathcal{E}_i$. □

Bibliography

[1] W. Ren and R. Beard. Consensus Seeking in Multi-agent Systems under Dynamically Changing Interaction Topologies. *IEEE Transactions on Automatic Control (TAC)*, 50(5):655–661, 2005.

[2] R. Olfati-Saber and R. Murray. Consensus Problems in Networks of Agents with Switching Topology and Time-Delays. *IEEE Transactions on Automatic Control (TAC)*, 49(9):1520–1533, 2004.

[3] A. Jadbabaie, J. Lin, and S. Morse. Coordination of Groups of Mobile Autonomous Agents Using Nearest Neighbor Rules. *IEEE Transactions on Automatic Control (TAC)*, 48(6):988–1001, 2003.

[4] H. Tanner, A. Jadbabaie, and G. Pappas. Flocking in Fixed and Switching Networks. *IEEE Transactions on Automatic Control (TAC)*, 52(5):863–868, 2007.

[5] D. Dimarogonas and K. Kyriakopoulos. On the Rendezvous Problem for Multiple Nonholonomic Agents. *IEEE Transactions on Automatic Control (TAC)*, 52(5):916–922, 2007.

[6] M. Egerstedt and X. Hu. Formation Constrained Multi-Agent Control. *IEEE Transactions on Robotics and Automation (TRA)*, 17(6):947–951, 2001.

[7] K. Oh, M. Park, and H. Ahn. A Survey of Multi-Agent Formation Control. *Automatica*, 53:424–440, 2015.

[8] M. Ji and M. Egerstedt. Distributed Coordination Control of Multi-Agent Systems While Preserving Connectedness. *IEEE Transactions on Robotics (TRO)*, 23(4):693–703, 2007.

[9] M. Zavlanos and G. J. Pappas. Potential Fields for Maintaining Connectivity of Mobile Networks. *IEEE Transactions on Robotics (TRO)*, 23(4):812–816, 2007.

[10] M. Zavlanos and G. Pappas. Distributed Connectivity Control of Mobile Networks. *IEEE Transactions on Robotics (TRO)*, 24(6):1416–1428, 2008.

[11] D. Dimarogonas, S. Loizou, K. Kyriakopoulos, and M. Zavlanos. A Feedback Stabilization and Collision Avoidance Scheme for Multiple Independent Non-Point Agents. *Automatica*, 42(2):229–243, 2006.

[12] Bottom-Up Hybrid Control and Planning Synthesis with Application to Multi-Robot Multi-Human Coordination (BUCOPHSYS). *http://bucophsys.eu/*, Accessed: February 2015.

[13] A. Nikou, C. K. Verginis, and D. V. Dimarogonas. Robust Distance-Based Formation Control of Multiple Rigid Bodies with Orientation Alignment. *20th World Congress of the International Federation of Automatic Control (IFAC WC), Toulouse, France*, 2017.

[14] A. Nikou, C. K. Verginis, and D. V. Dimarogonas. On the Position and Orientation Based Formation Control of Multiple Rigid Bodies with Collision Avoidance and Connectivity Maintenance. *56th IEEE Conference on Decision and Control (CDC)*, 2017, (Under Review).

[15] C. K. Verginis, A. Nikou, and D. V. Dimarogonas. Formation Control of Rigid Bodies with Collision Avoidance and Connectivity Maintenance. *IEEE Transactions on Control on Networked Systems (CONES)*, 2017, (Under Preparation).

[16] A. Nikou, J. Tumova, and D. V. Dimarogonas. Cooperative Task Planning Synthesis for Multi-Agent Systems Under Timed Temporal Specifications. *American Control Conference (ACC), Boston, MA, USA*, 2016.

[17] S. Andersson, A. Nikou, and D. V. Dimarogonas. Control Synthesis for Multi-Agent Systems under Metric Interval Temporal Logic Specifications. *20th World Congress of the International Federation of Automatic Control (IFAC WC), Toulouse, France*, 2017.

[18] A. Nikou, D. Boskos, J. Tumova, and D. V. Dimarogonas. Cooperative Planning Synthesis for Coupled Multi-Agent Systems Under Timed Temporal Specifications. *American Control Conference (ACC), Seattle, WA, USA*, 2017.

[19] A. Nikou, D. Boskos, J. Tumova, and D. V. Dimarogonas. On the Timed Temporal Logic Planning of Coupled Multi-Agent Systems. *Automatica*, 2017, (Under Review).

[20] A. Nikou, S. Heshmati-alamdari, C. K. Verginis, and D. V. Dimarogonas. Decentralized Abstractions and Timed Constrained Planning of a General Class of Coupled Multi-Agent Systems. *56th IEEE Conference on Decision and Control (CDC)*, 2017, (Under Review).

[21] A. Nikou, J. Tumova, and D. V. Dimarogonas. Probabilistic Plan Synthesis for Coupled Multi-Agent Systems. *20th World Congress of the International Federation of Automatic Control (IFAC WC), Toulouse, France*, 2017.

[22] A. Nikou, C. K. Verginis, S. Heshmati-alamdari, and D. V. Dimarogonas. A Nonlinear Model Predictive Control Scheme for Cooperative Manipulation with Singularity and Collision Avoidance. *25th IEEE Mediterranean Conference on Control and Automation (MED), Valletta, Malata*, 2017.

[23] S. Heshmati-alamdari, A. Nikou, and D. V. Dimarogonas. A Robust Control Approach for Underwater Vehicle Manipulator Systems in Interaction with Compliant Environments. *20th World Congress of the International Federation of Automatic Control (IFAC WC), Toulouse, France*, 2017.

[24] A. Nikou, G. Gavridis, and K. J. Kyriakopoulos. Mechanical Design, Modeling and Control of a Novel Aerial Manipulator. *IEEE International Conference on Robotics and Automation (ICRA), Washington State Convention Center, Seattle, USA.*, 2015.

[25] R. Horn and C. Johnson. Topics in Matrix Analysis. *Cambridge University Press*, 1994.

[26] H. K. Khalil. Nonlinear Systems. *Prentice Hall*, 2002.

[27] E. Sontag. Input to State Stability: Basic Concepts and Results. *Nonlinear and Optimal Control Theory*, pages 163–220, 2008.

[28] E. Sontag. On Characterizations of the Input-to-State Stability Property. *Systems and Control Letters*, 24(5):351–359, 1995.

[29] R. Alur and D.L. Dill. A Theory of Timed Automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

[30] C. Baier, J.P. Katoen, and K. G. Larsen. Principles of model checking. *MIT Press*, 2008.

[31] D. D. Souza and P. Prabhakar. On the Expressiveness of MTL in the Pointwise and Continuous Semantics. *International Journal on Software Tools for Technology Transfer*, 9(1):1–4, 2007.

[32] J. Ouaknine and J. Worrell. On the Decidability of Metric Temporal Logic. *20th Annual IEEE Symposium on Logic in Computer Science (LICS)*, pages 188–197, 2005.

[33] R. Alur, T. Feder, and T. A. Henzinger. The Benefits of Relaxing Punctuality. *Journal of the ACM (JACM)*, 43(1):116–146, 1996.

[34] M. Reynolds. Metric Temporal Logics and Deterministic Timed Automata. 2010.

[35] P. Bouyer. From Qualitative to Quantitative Analysis of Timed Systems. *Mémoire D ' habilitation, Université Paris*, 7:135–175, 2009.

[36] S. Tripakis. Checking Timed Buchi Automata Emptiness on Simulation Graphs. *ACM Transactions on Computational Logic (TOCL)*, 10(3), 2009.

[37] O. Maler, D. Nickovic, and A. Pnueli. From MITL to Timed Automata. *International Conference on Formal Modeling and Analysis of Timed Systems*, pages 274–289, 2006.

[38] D. Ničković and N. Piterman. From MTL to Deterministic Timed Automata. *Formal Modeling and Analysis of Timed Systems*, 2010.

[39] H. Hansson and B. Jonsson. A Logic for Reasoning about Time and Reliability. *Formal Aspects of Computing*, 1994.

[40] J. Rutten, M. Kwiatkowska, G. Norman, and D. Parker. Mathematical Techniques for Analyzing Concurrent and Probabilistic systems. *American Mathematical Society*, 2004.

[41] A. Bianco and L. Alfaro. Model Checking of Probabilistic and Nondeterministic Systems. *International Conference on Foundations of Software Technology and Theoretical Computer Science*, 1995.

[42] R. Beard, J. Lawton, and F. Hadaegh. A Coordination Architecture For Spacecraft Formation Control. *IEEE Transactions on Control Systems Technology (TCST)*, 9(6):777–790, 2001.

[43] J. A. Fax and R. Murray. Graph Laplacians and Stabilization of Vehicle Formations. *15th World Congress of the International Federation of Automatic Control (IFAC WC)*, 35(1):55–60, 2002.

[44] W. Ren and R. Beard. Formation Feedback Control For Multiple Spacecraft via Virtual Structures. *IEE Proceedings-Control Theory and Applications*, 151(3):357–368, 2004.

[45] L. A. Fax and R. Murray. Information Flow and Cooperative Control of Vehicle Formations. *IEEE Transactions on Automatic Control (TAC)*, 49(9):1465–1476, 2004.

[46] K. Do. Bounded Controllers For Formation Stabilization of Mobile Agents With Limited Sensing Ranges. *IEEE Transactions on Automatic Control (TAC)*, 52(3):569–576, 2007.

[47] W. Ren and E. Atkins. Distributed Multi-Vehicle Coordinated Control via Local Information Exchange. *International Journal of Robust and Nonlinear Control (IJRNC)*, 17(10-11):1002–1033, 2007.

[48] W. Dong and J. Farrell. Cooperative Control of Multiple Nonholonomic Mobile Agents. *IEEE Transactions on Automatic Control (TAC)*, 53(6):1434–1448, 2008.

[49] T. Van den Broek, N. van de Wouw, and H. Nijmeijer. Formation Control of Unicycle Mobile Robots: A Virtual Ctructure Approach. *48th IEEE Conference on Decision and Control (CDC)*, pages 8328–8333, 2009.

[50] B. Anderson, C. Yu, B. Fidan, and J. Hendrickx. Rigid Graph Control Architectures for Autonomous Formations. *IEEE Control Systems*, 28:48–63, 2008.

[51] J. Hendrickx, B. Fidan, C. Yu, B. Anderson, and V. Blondel. Formation Reorganization by Primitive Operations on Directed Graphs. *IEEE Transactions on Automatic Control (TAC)*, 53(4):968–979, 2008.

[52] J. Hendrickx. *Graphs and Networks for the Analysis of Autonomous Agent Systems*. PhD thesis, 2008.

[53] D. Paley, N. Leonard, and R. Sepulchre. Stabilization of Symmetric Formations to Motion Around Convex Loops. *Systems and Control Letters*, 57(3):209–215, 2008.

[54] L. Barnes, M. Fields, and K. Valavanis. Swarm formation Control Utilizing Elliptical Surfaces and Limiting Functions. *IEEE Transactions on Systems, Mechatronics and Cybernetics*, 39(6):1434–1445, 2009.

[55] L. Krick, M. Broucke, and B. Francis. Stabilisation of Infinitesimally Rigid Formations of Multi-Robot Networks. *International Journal of Control (IJC)*, 82(3):423–439, 2009.

[56] F. Dorfler and B. Francis. Formation Control of Autonomous Robots Based on Cooperative Behavior. *European Control Conference (ECC)*, pages 2432–2437, 2009.

[57] P. Lin and Y. Jia. Distributed Rotating Formation Control of Multi-Agent Systems. *Systems and Control Letters*, 59(10):587–595, 2010.

[58] M. Mesbahi and M. Egerstedt. Graph Theoretic Methods in Multiagent Networks. *Princeton University Press*, 2010.

[59] M. Cao, S. Morse, C. Yu, B. Anderson, and S. Dasgupta. Maintaining a Directed, Triangular Formation of Mobile Autonomous Agents. *Communications in Information and Systems*, 11(1):1, 2011.

[60] T. Summers, Changbin C. Yu, S. Dasgupta, and B. Anderson. Control of Minimally Persistent Leader-Remote-Follower and Coleader Formations in the Plane. *IEEE Transactions on Automatic Control (TAC)*, 56(12):2778–2792, 2011.

[61] A. Belabbas, S. Mou, S. Morse, and B. Anderson. Robustness Issues with Undirected Formations. *51st IEEE Conference on Decision and Control (CDC)*, pages 1445–1450, 2012.

[62] M. Basiri, A. Bishop, and P. Jensfelt. Distributed Control of Triangular Formations with Angle-Only Constraints. *Systems and Control Letters*, 59(2):147–154, 2010.

[63] Tolga Eren. Formation Shape Control Based on Bearing Rigidity. *International Journal of Control (IJC)*, 85(9):1361–1379, 2012.

[64] S. Zhao and D. Zelazo. Bearing Rigidity and Almost Global Bearing-Only Formation Stabilization. *IEEE Transactions on Automatic Control (TAC)*, 61(5):1255–1268, 2016.

[65] M. Trinh, K. Oh, and H. Ahn. Angle-Based Control of Directed Acyclic Formations with Three-Leaders. *2014 International Conference on Mechatronics and Control (ICMC)*, pages 2268–2271, 2014.

[66] A. Bishop, M. Deghat, B. Anderson, and Y. Hong. Distributed Formation Control with Relaxed Motion Requirements. *International Journal of Robust and Nonlinear Control (IJRNC)*, 25(17):3210–3230, 2015.

[67] K. Fathian, D. Rachinskii, M. Spong, and N. Gans. Globally Asymptotically Stable Distributed Control for Distance and Bearing Based Multi-Agent Formations. *American Control Conference (ACC)*, pages 4642–4648, 2016.

[68] D. Koditschek and E. Rimon. Robot Navigation Functions on Manifolds with Boundary. *Advances in Applied Mathematics*, 11(4):412–442, 1990.

[69] E. Rimon and D. Koditschek. Exact Robot Navigation Using Artificial Potential Functions. *IEEE Transactions on Robotics and Automation (TRA)*, 8(5):501–518, 1992.

[70] M. Gennaro and A. Jadbabaie. Formation Control for a Cooperative Multi-Agent System using Decentralized Navigation Functions. *American Control Conference (ACC)*, pages 6–pp, 2006.

[71] H. Tanner and A. Kumar. Formation Stabilization of Multiple Agents Using Decentralized Navigation Functions. *Robotics: Science and Systems*, 1:49–56, 2005.

[72] Z. Kan, A. Dani, J. M. Shea, and W. E. Dixon. Network Connectivity Preserving Formation Stabilization and Obstacle Avoidance via a Decentralized Controller. *IEEE Transactions on Automatic Control (TAC)*, 57(7):1827–1832, 2012.

[73] T. H. Cheng, Z. Kan, J. A. Rosenfeld, and W. E. Dixon. Decentralized Formation Control with Connectivity Maintenance and Collision Avoidance under Limited and Intermittent Sensing. *American Control Conference (ACC)*, pages 3201–3206, 2014.

[74] D. V. Dimarogonas and E. Frazzoli. Analysis of Decentralized Potential Field Based Multi-Agent Navigation via Primal-Dual Lyapunov Theory. *49th IEEE Conference on Decision and Control (CDC)*, pages 1215–1220, 2010.

[75] H. Tanner and A. Boddu. Multi-Agent Navigation Functions Revisited. *IEEE Transactions on Robotics (TRO)*, 28(6):1346–1359, 2012.

[76] D. Gennaro, M. Carmela, and A. Jadbabaie. Formation Control for a Cooperative Multi-Agent System Using Decentralized Navigation Functions. *American Control Conference (ACC)*, 2006.

[77] K. D. Do. Coordination Control of Multiple Ellipsoidal Agents with Collision Avoidance and Limited Sensing Ranges. *Systems and Control Letters*, 61(1):247–257, 2012.

[78] D. V. Dimarogonas and K. J. Kyriakopoulos. Decentralized Motion Control of Multiple Agents with Double Integrator Dynamics. *16th World Congress of the International Federation of Automatic Control (IFAC WC)*, 38(1):185–190, 2005.

[79] C. Bechlioulis and G. Rovithakis. Robust Adaptive Control of Feedback Linearizable MIMO Nonlinear Systems with Prescribed Performance. *IEEE Transactions on Automatic Control (TAC)*, 53(9):2090–2099, 2008.

[80] B. Siciliano, L. Sciavicco, and L. Villani. Robotics : Modelling, Planning and Control. *Springer*, 2009.

[81] R. Horn and C. Johnson. Matrix Analysis. *Cambridge university press*, 2012.

[82] S. Loizou and K. Kyriakopoulos. Automatic Synthesis of Multi-Agent Motion Tasks Based on LTL Specifications. *43rd IEEE Conference on Decision and Control (CDC)*, 1:153–158, 2004.

[83] T. Wongpiromsarn, U. Topcu, and R. Murray. Receding Horizon Control for Temporal Logic Specifications. *13th ACM International Conference on Hybrid Systems: Computation and Control (HSCC)*, pages 101–110, 2010.

[84] A. Ulusoy, S. Smith, X. Ding, C. Belta, and D. Rus. Optimality and Robustness in Multi-Robot Path Planning with Temporal Logic Constraints. *The International Journal of Robotics Research*, 32(8):889–911, 2013.

[85] R. Cowlagi and Z. Zhang. Route Guidance for Satisfying Temporal Logic Specifications on Aircraft Motion. *Journal of Guidance, Control, and Dynamics*, pages 1–12, 2016.

[86] I. Filippidis, D. V. Dimarogonas, and K. J. Kyriakopoulos. Decentralized Multi-Agent Control from Local LTL Specifications. *51st IEEE Conference on Decision and Control (CDC), Maui, Hawaii, USA*, pages 6235–6240, 2012.

[87] J. Liu, N. Ozay, U. Topcu, and R. Murray. Synthesis of reactive switching protocols from temporal logic specifications. *IEEE Transactions on Automatic Control (TAC)*, 58(7):1771–1785, 2013.

[88] M. Guo and D. Dimarogonas. Reconfiguration in Motion Planning of Single- and Multi-Agent Systems Under Infeasible Local LTL Specifications. *52nd IEEE Conference on Decision and Control (CDC)*, 2013.

[89] M. Guo and D. Dimarogonas. Multi-Agent Plan Reconfiguration Under Local LTL Specifications. *The International Journal of Robotics Research (IJRR)*, 34(2):218–235, 2015.

[90] Y. Kantaros and M. Zavlanos. A Distributed LTL-Based Approach for Intermittent Communication in Mobile Robot Networks. *ACC*, 2016.

[91] A. Bhatia, M. Maly, L. Kavraki, and M. Vardi. Motion Planning with Complex Goals. *IEEE Robotics and Automation Magazine*, 18(3):55–64, 2011.

[92] H. Kress-Gazit, G. Fainekos, and G. Pappas. Temporal-Logic-Based Reactive Mission and Motion Planning. *IEEE Transactions on Robotics (TRO)*, 25(6):1370–1381, 2009.

[93] G. Fainekos, A. Girard, K.G. Hadas, and G. Pappas. Temporal Logic Motion Planning for Dynamic Robots. *Automatica*, 45(2):343–352, 2009.

[94] M. Kloetzer and C. Belta. Automatic Deployment of Distributed Teams of Robots From Temporal Motion Specifications. *IEEE Transactions on Robotics (TRO)*, 26(1):48–61, 2010.

[95] M. Kloetzer, X. C. Ding, and C. Belta. Multi-Robot Deployment from LTL Specifications with Reduced Communication. *50th IEEE Conference on Decision and Control (CDC)*, pages 4867–4872, 2011.

[96] Y. Chen, X. Ding, A. Stefanescu, and C. Belta. A Formal Approach to Deployment of Robotic Teams in an Urban-Like Environment. *Distributed Autonomous Robotic Systems*, pages 313–327, 2013.

[97] Y. Chen, X. Ding, A. Stefanescu, and C. Belta. Formal Approach to the Deployment of Distributed Robotic Teams. *IEEE Transactions on Robotics*, 28(1):158–171, 2012.

[98] M. Quottrup, T. Bak, and R. Zamanabadi. Multi-Robot Planning: A Timed Automata Approach. *IEEE International Conference on Robotics and Automation (ICRA)*, 5:4417–4422, 2004.

[99] M. Andersen, R. Jensen T. Bak, and M. Quottrup. Motion Planning in Multi-Robot Systems Using Timed Automata. *Proc. of the 5th IFAC/EURON Symposium on Intelligent Autonomous Vehicles*, 2004.

[100] K. Larsen, P. Pettersson, and W. Yi. UPPAAL in a Nutshell. *International Journal on Software Tools for Technology Transfer*, 1(1):134–152, 1997.

[101] A. Ulusoy, S. Smith, X. Ding, C. Belta, and D. Rus. Optimal Multi-Robot Path Planning with Temporal Logic Constraints. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3087–3092, 2011.

[102] J. Liu and P. Prabhakar. Switching Control of Dynamical Systems from Metric Temporal Logic Specifications. *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5333–5338, 2014.

[103] V. Raman, A. Donzé, D. Sadigh, R. Murray, and S. Seshia. Reactive Synthesis from Signal Temporal Logic Specifications. *18th International Conference on Hybrid Systems: Computation and Control (HSCC)*, pages 239–248, 2015.

[104] J. Fu and U. Topcu. Computational Methods for Stochastic Control with Metric Interval Temporal Logic Specifications. *54th IEEE Conference on Decision and Control (CDC)*, pages 7440–7447, 2015.

[105] B. Hoxha and G. Fainekos. Planning in Dynamic Environments Through Temporal Logic Monitoring. 2016.

[106] Y. Zhou, D. Maity, and John S. Baras. Timed Automata Approach for Motion Planning Using Metric Interval Temporal Logic. *European Control Conference (ECC)*, 2016.

[107] Y. Zhou, D. Maity, and J. S. Baras. Optimal Mission Planner with Timed Temporal Logic Constraints. *European Control Conference (ECC)*, pages 759–764, 2015.

[108] S. Karaman and E. Frazzoli. Linear Temporal Logic Vehicle Routing with Applications to Multi-UAV Mission Planning. *International Journal of Robust and Nonlinear Control*, 21(12):1372–1395, 2011.

[109] P. Tabuada. Verification and Control of Hybrid Systems: a Symbolic Approach. *Springer Science and Business Media*, 2009.

[110] A. Girard and G. J. Pappas. Approximation Metrics for Discrete and Continuous Systems. *IEEE Transactions on Automatic Control (TAC)*, 52(5):782–798, 2007.

[111] R. Alur, T. Henzinger, G. Lafferriere, and G. Pappas. Discrete Abstractions of Hybrid Systems. *Proceedings of the IEEE*, 88(7):971–984, 2000.

[112] R. Koymans. Specifying Real-Time Properties with Metric Temporal Logic. *Real-time systems*, 2(4):255–299, 1990.

[113] T. Brihaye, M. Estiévenart, and G. Geeraerts. On MITL and Alternating Timed Automata. *FMATS*, 8053:47–61, 2013.

[114] C. Belta and V. Kumar. Abstraction and Control for Groups of Robots. *IEEE Transactions on Robotics (TRO)*, 20(5):865–875, 2004.

[115] M. Helwa and P. Caines. In-Block Controllability of Affine Systems on Polytopes. *53rd IEEE Conference on Decision and Control (CDC)*, pages 3936–3942, 2014.

[116] L. Habets, P. Collins, and V. Schuppen. Reachability and Control Synthesis for Piecewise-Affine Hybrid Systems on Simplices. *IEEE Transactions on Automatic Control (TAC)*, 51(6):938–948, 2006.

[117] M. Zamani, G. Pola, M. Mazo, and P. Tabuada. Symbolic Models for Nonlinear Control Systems without Stability Assumptions. *IEEE Transactions on Automatic Control (TAC)*, 57(7), 2012.

[118] J. Liu and N. Ozay. Finite Abstractions With Robustness Margins for Temporal Logic-Based Control Synthesis. *Nonlinear Analysis: Hybrid Systems*, 22:1–15, 2016.

[119] M. Zamani, M. Mazo, and A. Abate. Finite Abstractions of Networked Control Systems. *53rd IEEE Conference on Decision and Control (CDC)*, pages 95–100, 2014.

[120] G. Pola, P. Pepe, and M. D. Di Benedetto. Symbolic Models for Networks of Control Systems. *IEEE Transactions on Automatic Control (TAC)*, 2016.

[121] D. Boskos and D. Dimarogonas. Decentralized Abstractions For Multi-Agent Systems Under Coupled Constraints. *54th IEEE Conference on Decision and Control (CDC)*, pages 282–287, 2015.

[122] M. Andreasson, D. V. Dimarogonas, H. Sandberg, and K. H. Johansson. Distributed Control of Networked Dynamical Systems: Static Feedback, Integral Action and Consensus. *IEEE Transactions on Automatic Control*, 59(7):1750–1764, 2014.

[123] K. S. de Oliveira and M. Morari. Contractive Model Predictive Control for Constrained Nonlinear Systems. *IEEE Transactions on Automatic Control*, 45(6):1053–1071, 2000.

[124] B. Kouvaritakis and M. Cannon. Nonlinear Predictive Control: Theory and Practice. *IET*, 2001.

[125] Rolf Findeisen, Lars Imsland, Frank Allgower, and Bjarne A Foss. State and Output Feedback Nonlinear Model Predictive Control: An Overview. *European Journal of Control*, 9(2-3):190–206, 2003.

[126] H. Chen and F. Allgöwer. A Quasi-Infinite Horizon Nonlinear Model Predictive Control Scheme with Guaranteed Stability. *Automatica*, 34(10):1205–1217, 1998.

[127] R. Findeisen, L. Imsland, F. Allgöwer, and B. Foss. Towards a Sampled-Data Theory for Nonlinear Model Predictive Control. *New Trends in Nonlinear Dynamics and Control and their Applications*, pages 295–311, 2003.

[128] L. Grüne and J. Pannek. Nonlinear Model Predictive Control. *Springer London*, 2011.

[129] E. Camacho and C. Bordons. Nonlinear Model Predictive Control: An Introductory Review. *Assessment and Future Directions of Nonlinear Model Predictive Control*, pages 1–16, 2007.

[130] J. Frasch, A. Gray, M. Zanon, H. Ferreau, S. Sager, F. Borrelli, and M. Diehl. An Auto-Generated Nonlinear MPC Algorithm for Real-Time Obstacle Cvoidance of Ground Vehicles. *European Control Conference (ECC)*, 2013.

[131] F. Fontes. A General Framework to Design Stabilizing Nonlinear Model Predictive Controllers. *Systems and Control Letters*, 42(2):127–143, 2001.

[132] D. Marruedo, T. Alamo, and E. Camacho. Input-to-State Stable MPC for Constrained Discrete-Time Nonlinear Systems with Bounded Additive Uncertainties. *IEEE Conference on Decision and Control (CDC)*, 4:4619–4624, 2002.

[133] A. Eqtami, D. V. Dimarogonas, and K. Kyriakopoulos. Novel Event-Triggered Strategies for Model Predictive Controllers. *IEEE Conference on Decision and Control (CDC)*, pages 3392–3397, 2011.

[134] D. Mayne, J. Rawlings, C. Rao, and P. Scokaert. Constrained Model Predictive Control: Stability and Optimality. *Automatica*, 36(6):789–814, 2000.

[135] X. Ding, S. Smith, C. Belta, and D. Rus. LTL Control in Uncertain Environments with Probabilistic Satisfaction Guarantees. *18th World Congress of the International Federation of Automatic Control (IFAC WC)*, 2011.

[136] E. Wolff, U. Topcu, and R. Murray. Robust Control of Uncertain Markov Decision Processes with Temporal Logic Specifications. *51st IEEE Conference on Decision and Control (CDC)*, 2012.

[137] X. Ding, S. Smith, C. Belta, and D. Rus. Optimal Control of Markov Decision Processes With Linear Temporal Logic Constraints. *IEEE Transactions on Automatic Control (TAC)*, 2014.

[138] J. Fu, S. Han, and U. Topcu. Optimal control in markov decision processes via distributed optimization. *55th IEEE Conference on Decision and Control (CDC)*, 2015.

[139] J. Wang, X. Ding, M. Lahijanian, I. Paschalidis, and Calin A C. Belta. Temporal Logic Motion Control Using Actor–Critic Methods. *The International Journal of Robotics Research*, 2015.

[140] F. Montana, J. Liu, and T. Dodd. Sampling-Based Stochastic Optimal Control with metric interval temporal logic specifications. *IEEE Conference on Control Applications (CCA)*, 2016.

[141] A. Ayala, S. Andersson, and C. Belta. Temporal Logic Control in Dynamic Environments with Probabilistic Satisfaction Guarantees. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2011.

[142] M. Lahijanian, S. Andersson, and C. Belta. Temporal Logic Motion Planning and Control with Probabilistic Satisfaction Guarantees. *IEEE Transactions on Robotics (TRO)*, 2012.

[143] B. Wu and H. Lin. Counterexample-Guided Permissive Supervisor Synthesis for Probabilistic Systems Through Learning. *American Control Conference (ACC)*, 2015.

[144] B. Wu and H. Lin. Counterexample-Guided Distributed Permissive Supervisor Synthesis for Probabilistic Multi-Agent Systems Through Learning. *American Control Conference (ACC)*, 2016.

[145] M. Kwiatkowska, G. Norman, and D. Parker. Stochastic Model Checking. *International School on Formal Methods for the Design of Computer, Communication and Software Systems*, 2007.

[146] V. Forejt, M. Kwiatkowska, G. Norman, and D. Parker. Automated Verification Techniques for Probabilistic Systems. *Formal Methods for Eternal Networked Software Systems*, 2011.