Efficient Control for Cooperation

Communication, Learning and Robustness in Multi-Agent Systems

Jarne Ornia, D.

DOI
10.4233/uuid:97127a09-d53b-4969-a1e0-ae09b5e92a68

Publication date
2023

Document Version
Final published version

Citation (APA)
Jarne Ornia, D. (2023). *Efficient Control for Cooperation: Communication, Learning and Robustness in Multi-Agent Systems*. [Dissertation (TU Delft), Delft University of Technology].
https://doi.org/10.4233/uuid:97127a09-d53b-4969-a1e0-ae09b5e92a68

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Efficient Control for Cooperation: Communication, Learning and Robustness in Multi-Agent Systems

Efficient Control for Cooperation: Communication, Learning and Robustness in Multi-Agent Systems

Dissertation

for the purpose of obtaining the degree of doctor
at Delft University of Technology,
by the authority of the Rector Magnificus, Prof. dr. ir. T.H.J.J. van der Hagen,
chair of the Board of Doctorates,
to be defended publicly on
Monday 24 April at 10:00 o'clock

by

Daniel JARNE ORNIA

Master of Science, Aerospace Engineering, KTH Royal Institute of Technology,
Stockholm,
born in Seattle, United States of America.

This dissertation has been approved by the promotors.

Composition of the doctoral committee:

Rector Magnificus,	chairperson
Dr. M. Mazo Espinosa,	Delft University of Technology, promotor
Dr. J. Alonso Mora,	Delft University of Technology, copromotor
Independent members:	
Prof. dr. A. Abate,	University of Oxford, United Kingdom
Prof. dr. R. Babuška,	Delft University of Technology
Prof. dr. M. Cao,	University of Groningen
Prof. dr. R. Jungers,	Université Catholique de Louvain, Belgium
Prof. dr. ir. M. Wisse,	Delft University of Technology
Prof. dr. ir. B. De Schutter,	Delft University of Technology, reserve member

This research was partly funded by the European Research Council through the SENTIENT project, Grant No. 755953.

Keywords: Multi-Agent Systems, Formal Methods, Networked Control, Swarm Robotics, Event-Triggered Communication, Robust Reinforcement Learning, Cooperative Systems, Artificial Intelligence.

Printed by: Print Service Ede

Cover: Isabel Brenner

Style: TU Delft House Style, with small modifications

The author has set this thesis in LaTeX using the Libertinus and Inconsolata fonts.

ISBN: 978-94-6384-432-1
An electronic version of this dissertation is available at
`http://repository.tudelft.nl/`.

The revolution that takes place in your head, nobody will ever see that.
Gil Scott-Heron

Contents

Summary

Multi-Agent Systems have become a fascinating subject of study from the perspective of robotics and control, motivated by the parallel advances in computation and simulation power, and in hardware and microprocessor capacity. These have enabled the application of game-theoretical concepts to complex robotic and learning agents. Besides facing the same challenges as single-agent systems, the distributed nature of complex multi-agent systems sparks many questions and problems revolving around the constraints imposed by communication. The idea that multi-agent systems require communication to access information, to coordinate or simply to sense the environment they are acting on is sometimes overlooked when thinking of (and solving) emerging theoretical challenges. However, research problems related to communication in Cyber-Physical Systems have been a prevalent target for network control research for decades. In particular, we take inspiration on Event Triggered Control to study how communication affects performance, safety and robustness in multi-agent systems, through the following premise: If a system is robust enough, one does not need to update continuously a feedback controller, and it may be enough instead to update (*i.e.* communicate measurements, control actions, coordination signals...) the controller whenever a certain safety condition is triggered, allowing for *sparsity* in communication over control networks.

The work presented in this dissertation sits in the intersection between Network Control Systems, Robotics and (Multi-Agent) Reinforcement Learning. The general goal across the work presented in this dissertation is the following:

Understand how communication influences behaviour in multi-agent systems.

We begin by looking into a class of biologically inspired systems: Stigmergy-inspired robotic swarms. Stigmergy is a mechanism through which agents communicate with eachother via environment-based indirect signals. This is the case for ants, where they deploy pheromones as they move that others can then use to navigate unknown environments. Stigmergy methods have been used in recent years for swarm robotic control and coordination, and they present a very simple communication and coordination structure that constitutes a useful starting point for our work. We analyse how the agents converge to stationary (but unknown) distributions, and we study the evolution of the pheromone fields created by the agents. To be able to formally verify properties of stationary distributions, we propose mean field approximations of ant-inspired swarms. A mean field approximation considers the dynamics of a given multi-agent system when the number of agents is taken to infinity, generally allowing to de-couple agent trajectories, eliminate stochasticity and obtain closed form solutions for otherwise un-solvable dynamics.

We draw then a direct connection between ant-inspired swarms and multi-agent reinforcement learning systems. Through this abstraction, we propose first an event-triggered scheme to safely reduce communication needed in a distributed reinforcement learning

scenario which is directly inspired by the considered swarm problems. This allows us to retain value iteration convergence guarantees while allowing agents to use trajectory dependent decision functions to determine when to share information with a common environment. From this, we consider a more general multi-agent reinforcement learning problem: A setting where agents must share information with each other to be able to act according to learned controllers. This setting emerges whenever we aim to apply multi-agent reinforcement learning schemes to control real systems, and we want agents to use some degree of global information. We propose a solution that involves learning *robustness indicators*: Functions that tell agents, for a given joint state, how sensitive the problem at hand is to observation errors or delays. This allows agents to solve the converse problem: Use these functions to decide when is it necessary to send information to others.

In the last chapter of this dissertation we take a step back and look into robustness as an inherent problem to model free reinforcement learning systems. When trying to evaluate the impact of communicating (or not communicating) in complex multi-agent systems where controllers are learned from data, a recurrent problem is the limitations agents have to estimate the impact of acting under uncertain information. When we do not have dynamical models of a system (only learned *policies*), not only one cannot estimate future states or trajectories, but one does not even have tools to know if the observations that are being received by agents are being disturbed by some unknown signal. We propose then to approach this problem from an alternative perspective. If we need to learn controllers that are then deployed in possibly uncertain environments, we may want to make sure that "robustifying" the controller does not decrease (excessively) the capacity of the controller to successfully solve a given problem without uncertainty.

The work presented through this dissertation covers different problems and jumps between overlapping fields, but the methods and techniques proposed share a common principle: As complex multi-agent systems become more applicable to engineering problems, the need for understanding (and simplifying) communication rules is increasingly motivated by safety. Therefore, the problems and solutions considered aim to advance towards a formal understanding and design of communication logic in complex, model free multi-agent systems. Safety and explainability are relatively recent concerns in Artificial Intelligence, and sometimes overlooked in favour of empirical results. We argue here that this tendency will have to be inverted if we aim to solve human existential problems through these techniques.

Samenvatting

Multi-Agent Systemen zijn een fascinerend onderwerp van studie vanuit het perspectief van robotica en controle, gemotiveerd door de parallelle vooruitgang in berekening en simulatiekracht, en in hardware en microprocessor capaciteit. Hierdoor is het mogelijk om spel-theoretische concepten toe te passen op complexe robotica en leersystemen. Naast dezelfde uitdagingen als enkelvoudige agent systemen, roept de gedistribueerde aard van complexe multi-agent systemen veel vragen en problemen op die draaien om de beperkingen die worden opgelegd door communicatie. Het idee dat multi-agent systemen communicatie nodig hebben om informatie te verkrijgen, te coördineren of gewoon om de omgeving waarin ze actief zijn te kunnen waarnemen, wordt soms over het hoofd gezien bij het bedenken (en oplossen) van opkomende theoretische uitdagingen. Onderzoek naar communicatieproblemen in Cyber-Physical Systems is echter al tientallen jaren een belangrijk onderwerp voor netwerkcontroleonderzoek. In het bijzonder laten we ons inspireren door Event Triggered Control om te bestuderen hoe communicatie prestaties, veiligheid en robuustheid beïnvloedt in multi-agent systemen, via de volgende premisse: als een systeem robuust genoeg is, hoeft men niet voortdurend een feedbackcontroller bij te werken, en kan het voldoende zijn om in plaats daarvan de controller (*d.w.z.* communicatiemetingen, controlehandelingen, coördinatiesignalen...) bij te werken wanneer een bepaalde veiligheidsvoorwaarde wordt geactiveerd, wat zorgt voor *spaarzaamheid* in communicatie over controle netwerken.

Het werk gepresenteerd in deze dissertatie bevindt zich op het snijvlak van Network Control Systems, Robotica en (Multi-Agent) Reinforcement Learning. Het algemene doel van het werk gepresenteerd in deze dissertatie is als volgt:

Begrijpen hoe communicatie het gedrag beïnvloedt in multi-agent systemen.

We beginnen met het onderzoeken van een klasse van biologisch geïnspireerde systemen: Stigmergy-geïnspireerde robotzwermen. Stigmergy is een mechanisme waarmee agenten via omgevingsgebaseerde indirecte signalen met elkaar communiceren. Dit is het geval bij mieren, die feromonen verspreiden terwijl ze bewegen die anderen kunnen gebruiken om onbekende omgevingen te navigeren. Stigmergy-methoden zijn de afgelopen jaren gebruikt voor zwermrobotbesturing en -coördinatie, en ze presenteren een zeer eenvoudige communicatie- en coördinatiestructuur die een nuttig startpunt vormt voor ons werk. We analyseren hoe de agenten convergeren naar stationaire (maar onbekende) verdelingen, en we bestuderen de evolutie van de feromoonvelden gecreëerd door de agenten. Om eigenschappen van stationaire verdelingen formeel te kunnen verifiëren, stellen we mean field approximaties van mier-geïnspireerde zwermen voor. Een mean field approximatie overweegt de dynamiek van een gegeven multi-agent systeem wanneer het aantal agenten tot oneindig wordt genomen, waardoor agenttrajecten ontkoppeld kunnen worden, stochastiek geëlimineerd kan worden en gesloten vormoplossingen kunnen worden verkregen voor anders onoplosbare dynamiek.

We leggen dan een direct verband tussen mier-geïnspireerde zwermen en multi-agent reinforcement learning systemen. Door deze abstractie voor te stellen, stellen we eerst een event-gedreven schema voor om veilig de communicatie te verminderen die nodig is in een gedistribueerd versterkend leer scenario, dat rechtstreeks is geïnspireerd door de overwogen zwermproblemen. Dit stelt ons in staat om convergentiegaranties voor de waardeiteratie te behouden terwijl we agenten in staat stellen trajectafhankelijke beslissingsfuncties te gebruiken om te bepalen wanneer ze informatie moeten delen met een gemeenschappelijke omgeving. Van daaruit overwegen we een meer algemeen probleem van multi-agent reinforcement learning: een omgeving waarin agenten informatie met elkaar moeten delen om volgens geleerde controllers te kunnen handelen. Deze omgeving ontstaat wanneer we multi-agent reinforcement learning-schema's willen toepassen om echte systemen te controleren en we willen dat agenten enige mate van wereldwijde informatie gebruiken. We stellen een oplossing voor die inhoudt dat we "robuuste indicatoren" leren: functies die agenten vertellen, voor een gegeven gezamenlijke toestand, hoe gevoelig het probleem is voor observatiefouten of -vertragingen. Dit stelt agenten in staat om het omgekeerde probleem op te lossen: deze functies gebruiken om te beslissen wanneer het nodig is om informatie naar anderen te sturen.

In het laatste hoofdstuk van deze dissertatie nemen we een stap terug en kijken we naar robuustheid als een inherente probleem voor modelvrije reinforcement learning systemen. Bij het proberen de impact van communicatie (of het niet communiceren) te evalueren in complexe multi-agent systemen waar controllers worden geleerd uit gegevens, is een terugkerend probleem de beperkingen van agenten om de impact van handelen onder onzekerheid in te schatten. Als we geen dynamische modellen hebben van een systeem (alleen geleerde "beleidsregels"), kan men niet alleen toekomstige toestanden of trajecten schatten, maar men heeft ook geen tools om te weten of de observaties die door agenten worden ontvangen, worden verstoord door een onbekend signaal. We stellen dan voor om dit probleem vanuit een alternatieve perspectief aan te pakken. Als we controllers moeten leren die vervolgens in mogelijk onzekere omgevingen worden ingezet, willen we er misschien voor zorgen dat het "robuuster" maken van de controller de capaciteit van de controller om een gegeven probleem succesvol op te lossen zonder onzekerheid niet (teveel) vermindert.

Het werk dat in deze dissertatie wordt gepresenteerd behandelt verschillende problemen en springt tussen overlappende vakgebieden, maar de voorgestelde methoden en technieken delen een gemeenschappelijk principe: naarmate complexe multi-agent systemen meer toepasbaar worden op engineeringproblemen, wordt de behoefte aan begrip (en vereenvoudiging) van communicatieregels steeds meer gemotiveerd door veiligheid. Daarom richten de behandelde problemen en oplossingen zich op een formeel begrip en ontwerp van communicatielogica in complexe, modelvrije multi-agent systemen. Veiligheid en verklaringsmogelijkheden zijn relatief recente zorgen binnen Artificial Intelligence en worden soms over het hoofd gezien ten gunste van empirische resultaten. We betogen hier dat deze tendens moet worden omgekeerd als we menselijke existentiële problemen willen oplossen door middel van deze technieken.

Acknowledgements

I could never imagine, back in 2017 when I was a master student in Sweden, that I would end up obtaining a PhD degree in 2022 (after a multi-year global pandemic) and that *I would enjoy it*. At the time, that (*i.e.* the PhD, the pandemic goes without question...) seemed to me like the furthest option. I did not understand what research was or what it entailed, and I did not realise that it was actually a path I could choose. In hindsight, it was clear (as most things are once they pass) that my interests completely overlapped with the type of work I would be expected to produce as a researcher. But this was not obvious to me when following an engineering education; much focus was put on solving technical problems efficiently, and perhaps not enough on what it means to target *long term questions*, some of which will perhaps have an existential impact on society over the next decades. I did not begin to understand this until I came to Delft.

After this long (and truly enjoyable) four years in which I discovered my passion for science, I would like to thank and dedicate this dissertation to Isabel first, for literally being there every step of the way. To my family, for their support during (and through all the years leading to) the PhD. To Marc and Lluis, for (by chance) sharing the journey all this way. Finally, to Manuel for his guidance and good times spent meanwhile, and to Giannis, Gabriel and the others at DCSC for the endless discussions, talks and laughs.

Daniel
Delft, March 2023

1

Introduction

We can classify the challenges in the research and development of dynamical systems into just two classes: *perception* and *control*. Multi-agent systems are in essence no different, but the constrains that are naturally imposed in such systems give rise to a third class of problems spanning across the other two: *communication*. A multi-agent system can always be abstracted to a single agent with enough communication. If the problem is that of distributed perception, the multi-agent nature vanishes if we have (fast, regular, powerful enough) communication of information between the agents. Otherwise, if the problem is related to distributed control, communication allowing coordination can relax the multi-agent constraints.

In real-world systems these limit scenarios are not generally feasible. The challenge then lies in solving the communication-constrained problems, since these constraints can induce several types of complexities: Introduction of uncertainty, switched or hybrid dynamics, need for coordination strategies, *etc.* Additionally, in large multi-agent networked systems one may want to design the constraints themselves; under the premise that *more communication (data) is desirable*, one then needs to design systems that make the most efficient use of the communication resources. These resources are of different nature: Energy, network bandwidth or capacity, data storage, processing power.

The problems emerging from these constrains and requirements in communication have been the subject of study in control theory for years, in particular in the context of networked control [1–5], and for example in Event-Triggered Control [6] with the goal of making networked control systems more efficient. However, many of these problems are not well understood when considering multi-agent (or multi-robot) systems, and especially when these systems are designed *model free*. Without a model to estimate the impact of communication-related uncertainty in the dynamics of a system, one needs to devise different strategies to first investigate the robustness of complex multi-agent systems, and then to safely reduce the communication in such systems. Additionally, communication can take different forms: sampled data, coordination signals, sensed information, *etc.*, and all of these have different implications and generate different challenges when looked at from an event-triggered perspective. This sits at the core of the motivation for this doctoral dissertation.

1.1 Aim and Goals

Throughout this work we try to address the following sequence of problems.

Goals

1) To better understand the impact of communication in the dynamics of model-free multi-agent systems, and specifically in stability and optimality properties, starting with ant-inspired swarms and moving towards more general reinforcement learning systems.
2) To use this understanding to propose formal strategies one can use to reduce the communication required in such systems inspired by Event Triggered Control.
3) To abstract the reasoning and look into robustness problems in model free systems, and in particular robustness required to deal with communication sparsity, delays or general lack of noiseless information.

We expect these lines of work to have an impact in both theoretical and practical challenges. First, from a theoretical perspective, in advancing towards a deeper understanding of how communication patterns affect behaviour and dynamics of complex multi-agent systems, and allowing for the design of more efficient adaptive systems. Data-driven control, either model-based or model-free. Second, on the practical impact, we argue that as network systems grow more complex, there will be an increasing need to have efficient communication protocols to be able to maintain feasible infrastructure architectures. Reducing communication would first increase the overall efficiency of the system, and second allow for larger systems to be deployed. Additionally, with the recent progress on simulation-based learning for robotic control, we hope some of the work in this dissertation will help improve how well these learned models and controllers can adapt to real world settings.

1.2 Previous Work

We cover next the main lines of existing work that span across the topics covered in this dissertation.

1.2.1 Ant-Inspired Robotics

In recent years, Ant Colony (AC) algorithms have been used widely as inspiration for the design of swarm robotic control methods. AC Algorithms are biologically inspired algorithms that reproduce the pheromone-based communication that occurs in ant colonies. This communication principle is referred to as *stigmergy*: the agents communicate through indirect signals in the environment. in this case pheromones [7–9]. The agents modify the environment by adding pheromones, and make decisions on how to act as a function of the pheromones they sense. This sparked a whole branch of stochastic optimization algorithms: Ant Colony Optimization [7, 9]. Ant-inspired swarm coordination has also been applied to foraging problems in distributed robotic systems. Authors in [10] propose a stochastic ant-inspired approach to distribute swarm agents among different target regions. In [11] the authors present some early experiments on how robots can lay and

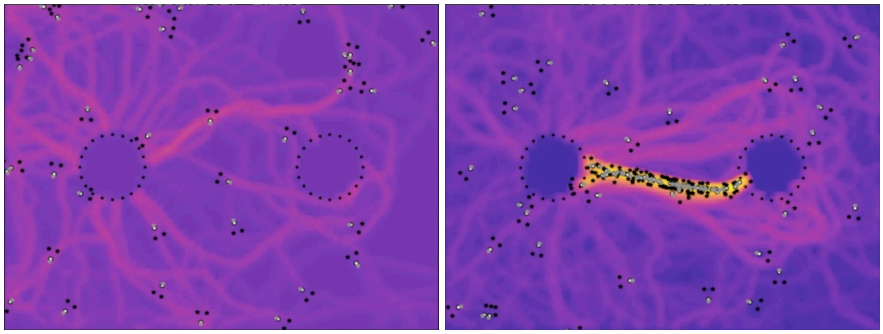


Figure 1.1: Pheromone-based swarm on a foraging problem.

follow pheromones to explore a space and collect targets, and [12–16] have presented similar robotic systems, either by using a *digital* pheromone field [14–16], using real chemicals [13], or fluorescent floors [12] (see also [17–19]). The authors in [20] use an ant-inspired swarm to solve a foraging problem on a 2D space by assuming a connected line-of-sight communication network, and having agents flood this network with their estimation of their relative position and angle at every time step. In [19, 21], authors use a combination of agents and beacon devices to guide navigation and store pheromone values. Authors treat pheromones as utility estimates for environmental states, and agents rely on line-of-sight communication and relative distances to the beacons to generate foraging trajectories.

In [22] the authors design a system where robots communicate with each other via LED signalling to indicate trails or vector fields pointing towards a given target, and provide empirical results on a wide number of scenarios. In [23, 24] the authors use a virtual-reality approach to implement the pheromone field, allowing the robots to have access to this virtual pheromone from a central controller, enabling effective foraging. It is worth noting that [17, 20, 22] assume agents in the swarm communicate directly with other agents, while [19, 21] de-couples this and proposes an environment-based interactions where agents only write and read data into locally reachable beacons.

1.2.2 Mean Field Approximations

To study the behaviour of stochastic systems of interacting agents and their asymptotic properties, a prolific approach that emerged from particle and fluid physics is based on so-called *mean field models*, where the number of interacting agents is taken to infinity to build continuous variable approximations of discrete stochastic systems. Mean field models have been extensively used in fluid mechanics and particle physics, and more recently in game theory and control [25–27], to derive mean-field based control approaches for networked systems [28].

The premise of mean field control is that, to find a control law for the state of an agent that has to interact with (a possibly large number of) other agents, one can do so by taking the limit scenario where the number of agents tends to infinity. This allows one to represent the networked system aggregatively, and consider the *mean* of the interactions.

In practice, this allows us to approximate stochastic differential (or difference) equations by a partial differential equation [29], usually easier to deal with and solve for feedback control. In recent years mean field formulations of large multi-agent systems or swarms have gained increased popularity [30–32] (see also an extensive survey in [33]) as these models abstract away the stochasticity in systems where the number of interacting agents becomes very large.

1.2.3 Event-Triggered Control and Learning

With the advances of computational power, processor size and communication infrastructure, much of the focus in control theory research has shifted in the last decades towards Network Control Systems [2, 3]. From the deployment of sensors and actuators over networks emerged the need to analyse the impact of more efficient control schemes. In particular, Event-Triggered Control (ETC) allows sensor and actuator to estimate, through trigger functions, when is it necessary to communicate state measurements or update control actions [6, 34, 35]. These ideas inspired many different lines of work addressing the general problem of efficient communication and control of systems. For example, for efficient distributed stochastic algorithms [36], or to learn parameters of linear [37, 38] and non-linear [39, 40]. In multi-agent settings, they have also been investigated for distributed control of linear [41] and non-linear [42] systems, to reduce the number of interactions between agents [43], or to speed up distributed policy gradient methods [44, 45].

1.2.4 Communication in Multi-Agent Systems

A number of Chapters in this doctoral dissertation address the problem of how to devise efficient communication strategies in complex multi-agent systems. There have been multiple examples of work studying different types of communication from a Multi-Agent Reinforcement Learning (MARL) and what problems arise from it [46–49]. In this line, in [50] actor coordination minimization is addressed and in [51, 52] authors allow agents to choose a communication action and receive a reward when this improves the policies of other agents. In [53] multi agent policy gradient methods are proposed with convergence guarantees where agents communicate gradients based on some trigger conditions, and in [44] agents are allowed to communicate a simplified form of the parameters that determine their value function.

Having non-reliable communication leads to severe disruptions in the robustness of the distributed policies' performance. The authors in [54] demonstrated experimentally how very small adversarial disruptions in state variable communications leads to a collapse of the performance of general collaborative MARL systems. In this regard, [55] proposes learning an "adviser" model to fall back on when agents have too much uncertainty in their state measurements, and more recently in [56] the authors enable agents to run simulated copies of the environment to compensate for a disruption in the communication of state variables, and in [57] agents are trained using adversarial algorithms to achieve more robust policies.

1.2.5 Robustness in Reinforcement Learning

This lack of robustness in communicative multi-agent learning presents difficulties when trying to design efficient systems where the goal is to communicate less often. On this

regard, the body of literature that deals with robustness problems in Reinforcement Learning [58] is extensive. Robustness in Reinforcement Learning (RL) can be looked at from different perspectives [59]: distributional shifts in the training data with respect to the deployment stage [60–63]; uncertainty in the model or observations [64, 65]; adversarial attacks against actions [66, 67]; and (4) sensitivity of neural networks (used as policy or value function approximators) towards input disturbances [68, 69].

In robustness against *model uncertainty*, the MDP may have noisy or uncertain reward signals or transition probabilities, as well as possible resulting *distributional shifts* in the training data [61, 63, 70–73], which connects to ideas on distributionally robust optimisation [74, 75]. One of the first examples is [61], where the author proposes using minimax approaches to learn Q functions that minimise the worst case total discounted cost in a general MDP setting. [76] propose a Bayesian approach to deal with uncertainty in the transitions. Another robustness sub-problem is studied in the form of *adversarial attacks or disturbances* by considering adversarial attacks on policies or action selection in RL agents [77–82]. Recently, [77] propose the idea that instead of modifying observations, one could attack RL agents by swapping the policy for an adversarial one at given times, and prove the existence of such policies in a zero-sum game framework. For a detailed review on Robust RL see [83].

At last, for robustness versus *observational disturbances*, agents observe a disturbed state measurement and use it as input for the policy [68, 69, 84–87]. In particular [85] consider both random and adversarial state perturbations, and introduce physically plausible generation of disturbances in the training of RL agents that make the resulting policy robust towards realistic disturbances. [86] propose a *state-adversarial* MDP framework, and utilise adversarial regularising terms that can be added to different deep RL algorithms to make the resulting policies more robust to observational disturbances, minimising the distance bound between disturbed and undisturbed policies through convex relaxations of neural networks to obtain robustness guarantees. In [87] the existence of optimal state-perturbing adversaries is studied, and how using LSTM increases robustness in such setting.

1.3 Main Contributions

We summarise the contributions presented in this doctoral dissertation in the following points.

1) We provide a set of theoretical results concerning the stationary properties of a class of ant-inspired random walks in terms of the probability distribution of agents and the graph weights.
2) By taking the mean field limit of the stochastic system, we analyse the closed form solutions of such probability distributions and are able to verify sub-optimality properties on these stationary distributions.
3) We abstract these systems to general Multi-Agent RL systems, and propose event-triggered schemes to reduce the communication for both the agent⟶environment and the agent⟶agent cases, and synthesise what type of formal guarantees still hold.
4) We propose a lexicographic optimisation based solution for obtaining robust policy gradient algorithms that preserve sub-optimality with respect to expected rewards, and maximise robustness against general state disturbances.

1

1.4 Content Structure

The work presented in this dissertation starts by looking into convergence properties of ant-inspired random walks, and evolves towards more general robustness problems in model-free systems. The structure is as follows.

- *Chapter 2*: We provide the mathematical notation used through the dissertation, as well as some useful concepts and auxiliary results.
- *Chapter 3*: We focus on ant-inspired swarms, modelled as a particular case of a biased random walk on weighted graphs. We provide formal guarantees on the convergence of agent distributions and graph weights, some preliminary results on communication problems, and numerical experiments demonstrating the predicted convergence properties.
- *Chapter 4*: We propose a mean field approximation of ant-inspired swarm. We obtain closed form solutions for the stationary distributions, which allows for formal verification of sub-optimality properties and to estimate the impact of hyper-parameters in a finite agent system.
- *Chapter 5*: We draw a parallelism between an ant inspired swarm and a distributed RL system. The agents build distributively a value function (pheromone field) that then they use to decide on their actions. Under this framework, we propose a class of trajectory dependent decision functions to allow agents to communicate experiences sparsely.
- *Chapter 6*: Looking past the distributed value function learning, we focus on MARL problems where agents need to communicate state measurements with each other to be able to execute their policies in real time. We propose learning *robustness surrogate* functions, that serve as trigger conditions for agents to decide when is it necessary to share information with others.
- *Chapter 7*: Following up on the robustness problem that emerges when observations are disturbed by additive noise in (Multi-Agent) RL problems, we propose an alternative framework to deal with state-observation robustness. By casting robustness as a quantifiable objective, we embed robustness in a Lexicographic approach to achieve robust RL policies through policy gradient algorithms in a *safe* way: Formally guaranteeing that the resulting policy will be quantifiably sub-optimal with respect to the original system.
- *Chapter 8*: We summarise the ideas presented through the dissertation, and provide possible directions for future work and open problems that emerge.

2

Mathematical Notation and Preliminaries

We provide in this chapter an introduction to the notation and concepts used recursively throughout this work.

2.1 Notation

We use $\mathbb{R}$ for the set of real numbers, $\mathbb{N}$ for the set of natural numbers. We say X is a set, $x \in X$ is an element of X and $\mathcal{X}$ is a collection of sets. We use $\Delta(X)$ as the space of probability measures over X. Subscripts are used to specify dependence, that is X_t is a set that depends on some variable t. For two elements of a vector space we use $\langle \cdot, \cdot \rangle$ as the inner product. We use $\mathbf{0}_n$ and $\mathbf{1}_n$ as a column-vector of size n that has all entries equal to 0 or 1 respectively. Unless otherwise stated, P, T, S are used for probability maps or stochastic matrices: We say that S is a $n \times n$ row-stochastic matrix if $S_{ij} \geq 0$ and each row of S sums to 1. We use $\mathbb{E}[\cdot]$ and $\mathrm{Var}[\cdot]$ for the expected value and the variance of a random variable, and in particular, $\mathbb{E}_{x\sim\mu}[f(x)]$ reads as "the expectation of $f(x)$ when x is sampled from some distribution μ".

The function $\mathrm{sgn}(\cdot)$ is the sign operator, with $\mathrm{sgn}(\mathbf{0}) = \mathbf{0}$. We say a function $f : \mathbb{R}_+ \longrightarrow \mathbb{R}_+$ is in the class of functions $\mathscr{K}$ if f is continuous, monotonically increasing and $f(0) = 0$. We say a function $f : \mathbb{R}_+ \longrightarrow \mathbb{R}_+$ is in class $\mathscr{K}_\infty$ if $f(\cdot) \in \mathscr{K}$ and $\lim_{a\to\infty} f(a) = \infty$.

2.2 Graph Theory

Through Chapters 3 and 4 we make use of several graph theoretic concepts, introduced in this section.

Definition 2.1. *We define a discrete time-varying weighted graph $G \coloneqq (X, E, W_t)$ as a tuple including a vertex (or* state*) set X, edge set E and weights $W : \mathbb{N}_0^+ \longrightarrow \mathbb{R}_+^{|X|\times|X|}$, where each value $W_t(i, j)$ is the weight assigned to edge $\{ij\} \in E$. Furthermore, the graph is connected if for every pair $i, j \in X$ there exists a set of edges*

$$\{\{iu_1\}, \{u_1u_2\}, \ldots, \{u_nj\}\} \subseteq E$$

that connects i and j.

We refer to an edge connecting i to j as $\{ij\} \equiv \{ji\}$ if the graph is undirected, and (ij) if the graph is directed. For simplicity, all concepts and definitions regarding weighted graphs will be define using undirected notation (edge from i to j as $\{ij\}$), but will apply to both directed and undirected graphs unless the opposite is stated.

The image of a function assigning values to edges in a graph can be written as a matrix, and the subscript will indicate both edges and entries in the image of the function. That is, let $f : \mathbb{N} \longrightarrow \mathbb{R}^{|X|\times|X|}$. Then, we use $f_k(i,j)$ as the ij-th entry in the image f_k, which corresponds to the edge $e \equiv \{ij\}$. We use this function class for the graph weights, and by definition

$$W_t(i,j) = 0 \quad \forall \{ij\} \notin E, \forall t.$$

The degree of vertex i is $\deg(i) \coloneqq |\{\{ij\} : \{ij\} \in E,\ j \in X\}|$, and weighted degree is

$$g_t(i) \coloneqq \sum_{k \in X} W_t(i,k).$$

Furthermore, when considering directed graphs the degree $\deg(i)$ refers to the out-degree unless the opposite is stated.

Definition 2.2 ([88]). *An $i-j$ path in G is a subgraph $X' \subseteq X$, $E' \subseteq E$*

$$X' = \{i,k,l,...,z,j\},\ E' = \big\{\{ik\},\{kl\},...,\{zj\}\big\}$$

where no vertex appears twice. An i-cycle is then a closed path $i-i$ starting and ending in the same vertex $i \in X$.

The diameter of the graph $\operatorname{diam}(G)$ or $|G|$ is the length of the longest path for any $i,j \in X$. The distance between two vertices i,j is $d(i,j)$, and is defined as the length of the shortest path between i and j.

2.3 Stochastic Matrices and Random Processes

2.3.1 Random Processes

In the context of stochastic systems we use Ω as the set of outcomes in a probability space, $\mathscr{F}$ as the measurable algebra (set) of events, and P as a probability measure $P : \mathscr{F} \longrightarrow [0,1]$. When two (or more) random variables follow the same probability distribution and are independent from each other we use *independent and identically distributed* (*i.i.d.*).

Definition 2.3 (Almost Sure Convergence[89]). *Let Ω be a probability sample space, with $\omega \in \Omega$ being any event. We say a sequence of random variables $x_0, x_1, ..., x_t$ defined on the set of events Ω converges almost surely (a.s.) to a random variable x_∞ as $t \longrightarrow \infty$ iff*

$$\Pr\{\omega : x_t(\omega) \longrightarrow x_\infty \text{ as } t \longrightarrow \infty\} = 1.$$

Theorem 2.1 (Strong Law of Large Numbers [90]). *Let $(\Omega, \mathscr{F}, p)$ be a probability space equipped with a σ-algebra of measurable subsets of Ω. Let x_n be a sequence of n i.i.d. random variables defined over the probability space, with expectation $\mathbb{E}[x_i]$. Let $s_n = x_1 + x_2 + ... + x_n$. Then,*

$$\lim_{n\to\infty} \frac{s_n}{n} = \mathbb{E}\big[x_i\big] \quad a.s.$$

Definition 2.4 ([89]). *A sequence of integrable random variables x_t measurable with respect to a sequence of increasing $\sigma-$algebras $\{\mathscr{F}_t\}$ is called a Martingale if*

$$\mathbb{E}\big[X_{t+1}|\mathscr{F}_t\big] = X_t \quad a.s. \quad \forall t \geqslant 0,$$

When considering a discrete time system, $\{\mathscr{F}_t\}$ includes all the information until time t.

Theorem 2.2 (Doob's Martingale Convergence [89]). *Let X_n be a Martingale such that*

$$\sup_n \mathbb{E}\big[X_n^+\big] < \infty.$$

Then, X_n converges a.s..

Theorem 2.3 (Stochastic Approximation with Non-Expansive Operator [91]). *Let $\{\xi_t\}$ be a random sequence with $\xi_t \in \mathbb{R}^n$ defined by the iteration:*

$$\xi_{t+1} = \xi_t + \alpha_t(F(\xi_t) - \xi_t + M_{t+1}),$$

where we assume all learning rates $\alpha_t(x,u) \in [0,1]$ satisfy the conditions $\sum_{t=1}^{\infty} \alpha_t(x,u) = \infty$ and $\sum_{t=1}^{\infty} \alpha_t(x,u)^2 < \infty$. Assume the following hold:

1) *$F : \mathbb{R}^n \mapsto \mathbb{R}^n$ is a $\|\cdot\|_\infty$ non-expansive map. That is, for any $\xi_1, \xi_2 \in \mathbb{R}^n$, $\|F(\xi_1) - F(\xi_2)\|_\infty \leq \|\xi_1 - \xi_2\|_\infty$.*
2) *$\{M_t\}$ is a martingale difference sequence with respect to the increasing family of $\sigma-$fields*

$$\mathscr{F}_t := \sigma(\xi_0, M_0, \xi_1, M_1, ..., \xi_t, M_t).$$

Then, the sequence $\xi_t \longrightarrow \xi^*$ almost surely where ξ^* is a fixed point such that $F(\xi^*) = \xi^*$.

2.3.2 Stochastic Matrices

We say that S is a $n \times n$ row (column) stochastic matrix if $S_{ij} \geq 0$ and each row (column) of S sums to 1. Their use to represent Markovian processes has been extensively studied, since the probability transition matrix of a Markovian discrete time process can be represented with such matrices.

Theorem 2.4 (Perron-Frobenius Theorem [92]). *Let $S \in \mathbb{R}_{\geq 0}^{n \times n}$ be a non-negative column stochastic irreducible matrix. Then,*

- *$\lambda_1(S) = 1$, all other eigenvalues are smaller in norm.*
- *The eigenvector $Sv = v$ defines a dimension 1 subspace with some basis vector having strictly positive entries.*

Let $\mathbb{M}_2$ be the class of all scrambling matrices (no two rows are orthogonal)[93].

Assumption 2.1 ([94]). *Let A_t be a discrete time dependent row stochastic matrix, with $\prod_{t=j}^{t=k} A_t$ its left product from k to j (i.e. $A_t A_{k-1} A_{k-2} ... A_j$). Suppose the process satisfies:*

1) *There exists integer $h > 0$ such that for all $k > 0$:*

$$\Pr\left[\prod_{t=k}^{h+k} A_t \in \mathbb{M}_2\right] > 0, \quad \sum_{i=1}^{\infty} \Pr\left[\prod_{t=k+(i-1)h}^{k+ih} A_t \in \mathbb{M}_2\right] = \infty.$$

2) There is a positive α such that any $A_t(i,j) > \alpha$ if $A_t(i,j) > 0$.

Theorem 2.5 ([94])**.** *Under Assumption 2.1, the product of the sequence of row stochastic matrices $\prod_{t=0}^{t=k} A_t$ converges to a random matrix of identical rows $L = \mathbf{1}\xi^\top$ a.s. as $k \to \infty$, where $\xi \in \mathbb{R}^n$ satisfies $\xi^\top \mathbf{1} = 1$.*

Note that the results in Theorem 2.5 do not imply that the stochastic matrix A_t converges, only its product.

2.4 Reinforcement Learning

Reinforcement Learning is (assumed here to be model free) a controller synthesis paradigm where we "learn" policies that map states to actions in some unknown stochastic environment, with the goal of maximising the discounted sum of rewards obtained by the action sequence executed. The general framework considered in RL problems is a Markov Decision Process.

Definition 2.5. *[Markov Decision Process] A Markov Decision Process (MDP) is a tuple (X, U, P, R) where X is a set of states, U is a set of actions, $P : U \times X \to \Delta(X)$ is the probability measure of the transitions between states and $R : X \times U \times X \to \mathbb{R}$ is the reward function. Finally, γ is a discount rate.*

In general, X, U are finite sets. We refer to x, u as the state-action pair at time t, and x, x' or x, y as two consecutive states. We write $P(x, u, y)$ as the probability of transitioning from x to y when taking action u. We denote in this work a *stochastic transition* MDP as the general MDP presented in Definition 2.5, and a *deterministic transition* MDP as the particular case where the transition probabilities additionally satisfy $P(x, u, y) \in \{0, 1\}$ (in other words, transitions are deterministic for a pair (x, u)). We say that an MDP is *ergodic* if for any policy the resulting Markov Chain (MC) is ergodic.

A (memoryless) policy for an agent taking actions on a MDP is a stochastic kernel $\pi : X \to \Delta(U)$. For simplicity, we overload notation on π, denoting by $\pi(x, u)$ as the probability of taking action u at state x under the stochastic policy π in the MDP, *i.e.*, $\pi(x, u) = \Pr\{u \mid x\}$. In a multi-agent case, we make use of the following extension of an MDP system.

Definition 2.6. *[Multi-Agent MDP] A Multi-Agent Markov Decision Process (MMDP) is a tuple $(N, X, U^n, P, \{R_i\}, \gamma)$ where N is a set of n agents, X is a cartesian product of state spaces $X = \prod_{i \in N} X_i$, $U^n = \prod_{i \in N} U_i$ is a joint set of actions, $P : X \times U^n \to \Delta(X)$ is a probability measure of the transitions between states and $R_i : X \times U^n \times X \to \mathbb{R}$ is a reward function for agent $i \in N$. If all reward functions are the same for all agents, we say the MMDP is* cooperative*. Finally, γ is a discount rate.*

We refer as *on-policy* algorithms to those RL algorithms that optimize a policy based on data collected from that same policy (*e.g.* A2C[58], PPO[95]...), and *off-policy* algorithms as those algorithms where experiences can be sampled from different exploration policies, and used to optimize a target policy (*e.g.* Q-Learning[96], DDPG[97]...).

Value Based Reinforcement Learning

The value function of a policy π, $V^\pi : X \longrightarrow \mathbb{R}$, and action-value function $Q : X \times U \longrightarrow \mathbb{R}$ are given by, respectively

$$V^\pi(x_0) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(x_t, \pi(x_t), x_{t+1})\right], \quad Q^\pi(x,u) = \mathbb{E}\left[R(x,u,y) + \sum_{t=1}^{\infty} \gamma^t R(x_t, \pi(x_t), x_{t+1})\right].$$

It is well known that, under mild conditions [58], the optimal value function can be obtained by means of the Bellman equation

$$V^*(x) := \max_u \sum_{y \in X} P(x,u,y)\big(R(x,u,y) + \gamma V^*(y)\big),$$

and an optimal policy $\pi^*(x) := \operatorname{argmax}_\pi V^\pi(x)\ \forall x \in X$ is guaranteed to exist. Similarly, the optimal Q function satisfies $Q^*(x,u) := \sum_{y \in X} P(x,u,y)(R(x,u,y) + \gamma V^*(y))$. We then define the objective function as

$$J(\pi) := \mathbb{E}_{x_0 \sim \mu_0}\big[V^\pi(x_0)\big]$$

with μ_0 being a distribution of initial states, and we use $J^* := \max_\pi J(\pi)$. If a policy is parameterised by $\theta \in \Theta$ we write π_θ and $J(\theta)$.

When considering RL learning rates, we assume they satisfy the following assumption.

Assumption 2.2 (Learning Rates)**.** *All learning rates $\alpha_t(x,u) \in [0,1]$ satisfy the conditions $\sum_{t=1}^{\infty} \alpha_t(x,u) = \infty$ and $\sum_{t=1}^{\infty} \alpha_t(x,u)^2 < \infty$.*

In Q-Learning [96], the Q values are initialised to some value $Q_0(x,u) \in \mathbb{R}\ \forall x,u$, and are updated after each transition observation $x \longrightarrow y$ with a decaying learning rate α_t as

$$Q_{t+1}(x,u) = Q_t(x,u) + \alpha_t\big(R(x,u,y) + \gamma \max_v Q_t(y,v) - Q_t(x,u)\big), \tag{2.1}$$

and $R(x,u,y) + \gamma \max_v Q_t(y,v) - Q_t(x,u)$ is the temporal difference (TD) error. The subscript t represents the number of iterations in (2.1). For ease of notation we may omit the explicit dependence of $\alpha_t(x,u)$ on (x,u), and write $\alpha_t \equiv \alpha_t(x,u)$. The iteration on (2.1) is known to converge to the optimal Q^* function under conditions on reward boundedness and sum convergence for the rates α_t [96].

Theorem 2.6 (Q-Learning [96])**.** *For an MDP with a bounded reward function, let the learning rate $\alpha_t \in [0,1)$ satisfy*

$$\sum_{t=1}^{\infty} \alpha_t(x,u) = \infty, \quad \sum_{t=1}^{\infty} \big(\alpha_t(x,u)\big)^2 < \infty.$$

Then, the iteration 2.1 converges $Q_t \longrightarrow Q^*$ almost surely for $t \longrightarrow \infty$.

Policy Based Reinforcement Learning
In policy-based reinforcement learning methods, one aims to construct a parametrised policy that directly maximises the RL objective $J(\pi)$. In general, this is formulated through the gradient ascent scheme:

$$\pi_{t+1} = \pi_t + \alpha_t \nabla J(\pi_t),$$

2

or if π is parametrised by some $\theta \in \Theta$, $\theta_{t+1} = \theta_t + \alpha_t \nabla_\theta J(\pi_t)$. In practise, the objective $J(\pi)$ is constructed as an estimate from Monte Carlo trajectory samples, and the policy iteration is then a stochastic gradient ascent scheme. When these estimators are learned in the form of a value function, we refer to them as Actor-Critic Algorithms [98]. There are plenty of PG algorithms that incorporate different heuristics to *e.g.* reduce variance in the objective estimator, stabilise the iteration steps or allow for off-policy learning [95, 99, 100].

Lexicographic Reinforcement Learning
We introduce now some concepts related to multi-objective RL [101] necessary for Chapter 7. A Multi-Objective MDP is an MDP with a set of reward functions $\{R_i\}$ that define a set of discounted reward objectives $\{J_i\}$. In this setting, Lexicographic Reinforcement Learning [102] provides an approach to find policies that optimise each objective prioritised lexicographically, such that it optimizes the first J_1 up to a certain ϵ, and from those solutions optimizes J_2, etc. In particular, we introduce now Policy-Based Lexicographic RL (PB-LRL) for an example with two objective functions. Consider a parametrised policy π_θ with $\theta \in \Theta$, and two objective functions K_1 and K_2. PB-LRL uses a multi-timescale optimisation scheme to optimise θ faster for higher-priority objectives, iteratively updating the constraints induced by these priorities and encoding them via Lagrangian relaxation techniques [103]. Let $\theta' \in \operatorname{argmax}_\theta K_1(\theta)$. Then, PB-LRL can be used to find parameters:

$$\theta'' = \operatorname{argmax}_\theta K_2(\theta) \quad \text{such that} \quad K_1(\theta) \geq K_1(\theta') - \epsilon.$$

This is done through the estimated gradient ascent update:

$$\begin{aligned} \theta &\leftarrow \operatorname{proj}_\Theta \left[\theta + \nabla_\theta \hat{K}(\theta)\right], \\ \lambda &\leftarrow \operatorname{proj}_{\mathbb{R}_{\geq 0}} \left[\lambda + \eta_t(\hat{k}_1 - \epsilon_t - K_1(\theta))\right], \end{aligned} \tag{2.2}$$

where $\hat{K}(\theta) := (\beta_t^1 + \lambda\beta_t^2) \cdot K_1(\theta) + \beta_t^2 \cdot K_2(\theta)$, λ is a Langrange multiplier, $\beta_t^1, \beta_t^2, \eta_t$ are learning rates, and $\hat{k}_1$ is an estimate of $K_1(\theta')$. Typically, we set $\epsilon_t \longrightarrow 0$, though we can use other tolerances too, *e.g.*, $\epsilon_t = 0.9 \cdot \hat{k}_1$. For more details on the convergence proofs and technicalities of PB-LRL we refer the reader to [102]. The next result is the main Theorem in [102], it establishes convergence of general policy-based RL algorithms under lexicographic constraints, and it is used in Chapter 7 of this work.

Theorem 2.7 (PB-LRL Convergence[102])**.** *Let $\mathscr{M}$ be a multi-objective MDP with objectives J_i, $i \in \{1, ..., m\}$ of the same form. Assume a policy π is twice differentiable in parameters θ, and if using a critic V_i assume it is continuously differentiable on θ_i. Suppose that if PB-LRL is run for T steps, there exists some limit point $w_i^*(\theta)$ when θ is held fixed under conditions $\mathscr{C}$ on $\mathscr{M}$, π and V_i. If $\lim_{T\to\infty} \mathbb{E}_t[\theta] \in \Theta_1^\epsilon$ for $m = 1$, then for any $m \in \mathbb{N}$ we have $\lim_{T\to\infty} \mathbb{E}_t[\theta] \in \Theta_m^\epsilon$ where ϵ depends on the representational power of the parametrisations of π, V_i.*

3

Convergence and Properties of Ant Inspired Swarm Dynamics

This chapter establishes a set of asymptotic convergence results for ant-inspired random walks, while providing conditions in graph structure and parameter choice. We give estimates on convergence rates, and how they relate to problem parameters and graph topology. For this, we first provide a formal model the agent environment as a weighted graph, where agents add weight to the edges as they traverse them. This defines a coupling in the stochastic dynamics where the graph weights are modified by the agent movements, and introduces a dependency in the system between the agent trajectories and weight sequences that give rise to non-homogeneous Markovian processes. To derive assymptotic guarantees, we solve this coupling by combining Martingale theory with stochastic matrix products and ergodic coefficients and obtain distributional limits for both the graph weights and agent positions.

This chapter is based on 🗎 *D Jarne Ornia and M. Mazo Jr, "Convergence of Ant Colony Multi-Agent Swarms"* HSCC '20: Proceedings of the 23rd International Conference on Hybrid Systems: Computation and Control*, Apr. 2020.* [104]. All proofs for technical statements can be found in Appendix A.

3.1 Introduction

Swarms occur naturally in many insect and animal species, therefore attempting to model these biological swarming behaviours is a big part of biomimicry research [105, 106]. In this framework, Ant Colony (AC) algorithms are a subset of biologically inspired stochastic algorithms based on the behavioural traits of ants, used commonly for optimization problems. Their main characteristic is the use of stigmergy: the environment is the main communication medium and information storage tool [7–9]. The agents mark the environment and make stochastic decisions based on the marks they encounter. These algorithms can be used as a control strategy for robotic swarms, either as a path planning system [107–110] or to directly stablish coordination in a robotic swarm [15, 111–113].

With these applications in mind, we are interested in studying the convergence properties of AC algorithms when applied to swarm coordination over graphs. These processes result in *ant inspired biased random walks*: Random walks where the edge weights are time dependent, and modified by the agents as they move. Random walks have been largely studied [114–116], and edge-reinforced random walks on weighted graphs and its asymptotic behaviour has been studied for continuous and discrete time [117–119]. Alternatively, convergence has been proven for certain kinds of AC optimization algorithms [120–122], but the results do not apply directly to the proposed ant inspired random walk. In most cases they require a central entity to analyse all paths the agents are generating, and add more or less weight depending on a cost function. Furthermore, when applying these algorithms to cyber-physical swarms, interacting with the environment translates into some kind of data transmission. In such networks there can be communication restrictions (desired or undesired), under which the existing convergence proofs would not hold. We are interested in applying these techniques to control and route real swarms, hence the motivation to find more general convergence conditions.

3.1.1 Main Contribution

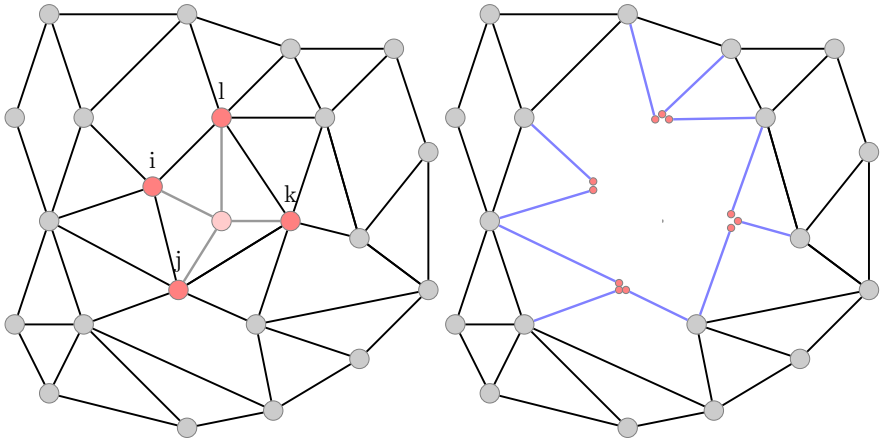
The main contributions of this chapter are two-fold.

Convergence of agent probability distributions We first show, leveraging results on random sequences of stochastic matrices how, with mild assumptions on the ant-inspired dynamics and underlying graphs, the probability distribution of agents converges (exponentially) to a limiting distribution for fixed initial conditions.

Convergence of transition functions Then, we show that for directed graphs, the sequence of transition functions is in fact a *martingale* in an ant-inspired random walk, allowing us to verify convergence properties of the sequence of transition functions.

3.2 Ant-Inspired Biased Random Walks

We aim now to formulate a general description of Ant-Inspired swarm dynamics on graphs, which can be generalised as a biologically inspired biased random walk. These agents follow transition probability distributions that are a function of the graph weights, and the graph weights are then modified by the movement of the agents. We describe first the



(a) Planar graph.

(b) Expanded graph, modified edges (blue).

Figure 3.1: Graph Expansion with target set (red)

agent dynamics, then the graph weight dynamics, and at last how these are combined into the dynamical problem to be studied in this Chapter.

3.2.1 Agent Dynamics

Let G be a weighted connected graph as in Definition 2.1[1]. Let $N = \{1, 2, ..., n\}$ be a set of agents walking from vertex to vertex. The position of agent a at time t is $x_t^a = v$, $v \in X$, and we group them in a vector $x_t \coloneqq \{x_t^a : a \in N\}$. The position of the agents will evolve depending on some probability transition matrix $P_t : X \times X \longrightarrow [0,1]$. We are interested in getting our agents to converge to trajectories connecting a starting vertex x^0 and a target vertex x^g infinitely often. First, consider all the vertices in our graph that are not connected to x^0 nor x^g. In this case, the agents move by selecting adjacent vertices based on the weight dependent probability distribution

$$P_t(j,i) = \Pr\{x_{t+1}^a = j \,|\, x_t^a = i\} = \frac{W_t(i,j)}{g_t(i)}, \; a \in A, \; i, j \neq x^g, x^0. \tag{3.1}$$

This is analogous to a biased random walk in a graph. When agents are adjacent to the target vertex x^g, they should prioritise this over any other vertex choice (mimicking ants moving towards food sources),

$$P_t(j,i) = \begin{cases} 1 & \text{if } i = x^g, x^0, \; \{ij\} \in E \\ \frac{W_t(i,j)}{g_t(i)} & \text{else.} \end{cases} \tag{3.2}$$

Remark 3.1. *In general, for foraging or path planning problems we require the agents to turn around when finding x^g (x^0). This can be incorporated in the graph structure without*

[1]We consider in this Chapter *edge-weighted graphs.*

breaking Markovian properties or considering different transition matrices for different agents. This is done as in Figure 3.1b.

The agent transition dynamics translate into the following dynamics for the agent probability distribution.

Definition 3.1. *The probability distribution $y_t \in \Delta(X)$ is the probability of having an agent in any vertex $i \in X$ at time t. The distribution evolves according to*

$$y_{t+1} = P_t y_t.$$

That is, given a distribution y_t, the product $P_t y_t$ gives us the distribution at the next time step. Note that $P_t(j,i)$ represents then the probability of moving from i to j, and $\Pr\{x^a_{t+1} = j\}$ is the j-th entry of y_{t+1}. The distribution is initialised to some initial distribution y_0.

Remark 3.2. *See that the agent distribution y_t follows Markovian dynamics; the probabilities at time $t+1$ are fully determined by the state at t. However, the system is only fully Markovian if we consider the joint set of random variables $\{x_t, M_t, W_t\}$. Additionally, the Markov process is* non-homogeneous *and the evolution of the transition probabilities is itself stochastic, which prevents us from computing stationary distributions. This is the main problem addressed in this chapter, and the proposed solution will become clear in further Sections.*

3.2.2 Graph Dynamics

Let us first define the following agent movement matrix.

Definition 3.2. *The matrix of agent movements at time t, $M_t :\in \mathbb{N}^{|X|\times|X|}$, has entries*

$$M_{t+1}(i,j) \coloneqq \left|\{a \in N \,:\, x^a_{t+1} = j,\, x^a_t = i\}\right|, \tag{3.3}$$

that is, the entry i,j of the matrix M_{t+1} is the amount of agents that were at vertex i at time t, and move to vertex j at time $t+1$.

Observe that $M_{t+1}(i,j)$ is a random variable, since it depends on the agent state at $t+1$ and this follows a stochastic process described in Definition 3.1. With this we can write the weight dynamics in the graph.

Definition 3.3. *Let M be an agent movement matrix. If G is a directed graph, each time step the graph weight matrix is updated following the dynamics*

$$W_{t+1} = (1-\alpha)W_t + \alpha\frac{M_{t+1}}{n},$$

where $\alpha \in (0,1)$ is a chosen evaporation factor. If G is undirected, M_{ij} and M_{ji} act over the same edge, and the dynamics are

$$W_{t+1} = (1-\alpha)W_t + \frac{\alpha}{n}\left(M_{t+1} + M^\top_{t+1}\right).$$

All weights are initialised to a uniform weight distribution, $W_0 = \omega_0 A$, where A is the adjacency matrix of G.

The value of n may be limited to the practical application, but in principle the amount of weight added per agent is a design parameter and we are free to choose any value. The choice of $\frac{\alpha}{n}$ is motivated by the fact that it ensures the total amount of weight will be constant if the initial weight amount adds to 1, i.e. $\sum_i \sum_j W_t(i,j) = 1 \; \forall t > 0$ if $\omega_0 |E| = 1$, both for directed and undirected graphs, and it is consistent with seminal work on Ant Colony Optimisation [7].

3.2.3 Problem Definition

We consider now the graph and agent dynamics together to define the complete AC Swarm system in a graph.

Definition 3.4. *We define an Ant System (AS) as a tuple* $(G, \{x_t\}, \Lambda)$ *where* G *is a planar connected weighted graph with at least one odd length cycle. The weights* W_t *follow the dynamics in Definition 3.3. The agent positions* $\{x_t\}$ *follow the agent probability distribution dynamics in Definition 3.1. Finally,* $\Lambda = (x^g, x^0, P_t)$ *is the tuple of restrictions to the agent movements, with* P_t *defined as* (3.2)*. The vertices* x^g, x^0 *are the initial and goal vertices.*

Remark 3.3. *Observe the requirement of* G *being connected and having at least one odd length cycle. This implies that for long enough times, any vertex* $i \in X$ *is reachable from any other* $j \in X$*. This is a common concept when studying random walks, and it is shown in the next section. The necessity of this will become clear in further sections.*

We are ready now to formulate the convergence problem that concerns this Chapter.

Problem 3.1. *Let an AC Graph System AS as defined in Definition 3.4. Can we ensure the distribution of agents around the graph* G *converges to a stationary distribution* y_∞*? and, what are the conditions for the graph topology and parameters that need to be satisfied?*

3.3 Results

As pointed out in Definition 3.3, the weight dynamics are different if we consider a directed graph since the weights $W_t(i,j)$ are affected by the symmetric agent movements $M_t(j,i)$. This motivates to approach the problem in slightly different ways for directed or undirected graphs. We first present general convergence results that hold for any connected graph. After that, we present stronger convergence results in the case the graph is directed. Before these, let us state a result relating existing work on stochastic matrices for consensus problems with this Chapter.

Corollary 3.1. *The results in Theorem 2.5 apply similarly to a sequence of column stochastic matrices. In particular, for a sequence* $\{B_t\}$ *where* $B_i \in \mathbb{R}^{n \times n}$ *and all* $B_i^\top$ *satisfy Assumption 2.1:*

$$\lim_{k \to \infty} \prod_{t=0}^{t=k} B_t = \left(\mathbf{1} \xi^\top \right)^\top$$

3.3.1 Dynamics of probability distributions

Recall the agent distribution dynamics in Definition 3.1. With any connected graph, we can write the distribution at any time $t > 0$ as

$$y_{t+1} = P_t y_t = P_t P_{t-1} y_{t-1} = \ldots = \prod_{k=0}^{k=t} P_k y_0.$$

Therefore, if the limit $L_\infty \coloneqq \lim_{t\to\infty} \prod_{k=0}^{k=t} P_k$ exists,

$$\lim_{t\to\infty} y_{t+1} = \lim_{t\to\infty} \prod_{k=0}^{k=t} P_k y_0 = L_\infty y_0 \coloneqq y_\infty. \tag{3.4}$$

That is, if we can show the product of our sequence of stochastic matrices P_t converges to a stochastic matrix, the agent distribution will converge to a stationary distribution. For this, let us defined a restricted weight matrix.

Assumption 3.1. *Any graph G is connected and has at least one odd length cycle.*

Assumption 3.2. *We assume the weights W_t to be lower bounded for any edge in the graph by some non-zero constant $\varepsilon > 0$.*

To show the agent distribution convergence properties, we first present the property introduced in Remark 3.3.

Proposition 3.1. *Let l_c be the maximum length of any odd length cycle in G. Let* $\operatorname{diam}(G)$ *be the diameter of the graph. Then,*

$$t \geqslant 2\operatorname{diam}(G) + l_c \Longrightarrow \Pr\{y_t(j) \mid y_0(i) = 1\} > 0 \quad \forall i, j \in X.$$

Remark 3.4. *We consider graphs that represent discretisations of space in navigation problems. Since we can always add a self loop in a vertex with weight ε, we consider that effectively the bound in Proposition 3.1 can be tightened to $t \leqslant 2\operatorname{diam}(G) + 1$.*

Proposition 3.2. *The sequence $\{P_t^\top\}$ satisfies the conditions in Assumption 2.1.*

Now, we present the main result for any connected graph regarding agent distribution convergence.

Theorem 3.1 (Agent Distribution Convergence)**.** *Let AS be an AC graph system from Definition 3.4. The product $\prod_{t=0}^{t=k} P_t$ converges to a column matrix a.s. as $t \longrightarrow \infty$,*

$$\lim_{k\to\infty} \prod_{t=0}^{t=k} P_t = \xi \mathbf{1}^\top, \tag{3.5}$$

where $\xi \in \Delta(X)$ is a probability distribution.

Corollary 3.2. *For an AS, let every agent $a \in N$ use a different weight matrix W_t^a such that*

$$W_t^a(i,j) = \begin{cases} 0 & \text{if } x_t^a = i \text{ and } x_{t-1}^a = j, \\ W_t(i,j) & \text{else.} \end{cases}$$

Then, each agent will converge to a different stationary distribution $y_t^a \xrightarrow{a.s.} y_\infty^a$ as $t \longrightarrow \infty$.

Other conclusions can be now extracted from the presented results. In particular, how the decision of agents to add or not add weight can be made without disrupting convergence properties of the system.

Corollary 3.3. *Let AS be an AC graph system. Let $\chi(a) \in \{0,1\}$ be a random variable taking value 1 if a communication event from agent $a \in N$ takes place, and value 0 otherwise. If $\chi(a)$ is independent of M_t, then it does not affect convergence properties of the system.*

3.3.2 Directed Graphs: Convergence of Transition Matrices

In a directed graph, the weights of an AC graph system *AS*, and edges (ij) are not affected by the changes in edge (ji). Considering this, to prove the main result for directed graphs we present first a set of necessary concepts.

Proposition 3.3. *Let $AS = (G, \{x_t\}, \Lambda)$ be an AC system. Let its state be fully defined at time t by σ-algebra*

$$\mathscr{F}_t = \sigma(M_0, M_1 ..., M_t).$$

At last, let $n_t(i) \coloneqq |\{a \in N \,|\, x_t^a = i\}|$ be the total amount of agents in vertex i at time t. Then, the position of an agent $x_{t+1}(a_0)$ is a random variable independent of other agent positions $x_{t+1}(a_k)$, $a_k \in N \setminus \{a_0\}$, and the conditional expected value of M_{t+1} is

$$\mathbb{E}\left[M_{t+1}(i,j) \mid \mathscr{F}_t\right] = P_t(j,i) n_t(i).$$

The sum over the rows in M_{t+1} depends on the state of our system at time t. More specifically,

$$\sum_{j \in X} M_{t+1}(i,j) = |\{a \in N \,|\, x_t^a = i\}| = n_i(t).$$

Similarly, the weighted degree $g_{t+1}(i)$ is also determined if we know the values of M_0, M_1, ..., M_t. By definition

$$g_{t+1}(i) = \sum_{k \in X} (1-\alpha) W_t(i,k) + \frac{\alpha}{n} M_{t+1}(i,k) = (1-\alpha) g_t(i) + \frac{\alpha}{n} \sum_{k \in X} M_{t+1}(i,k).$$

Then,

$$g_{t+1}(i) = (1-\alpha) g_t(i) + \frac{\alpha}{n} n_i(t).$$

With this, we can show a strong stochastic property of the evolution of P_t when the underlying graph is directed.

Proposition 3.4. *Let AS be an AC graph system with G being a directed graph. Let the increasing σ algebra $\mathscr{F}_t = \sigma(M_0, M_1 ..., M_t)$, and the temporal difference of the probability transition matrix P_t be defined $\partial P_{ji}(t) \coloneqq P_{t+1}(j,i) - P_t(j,i)$. For any $\alpha \in (0,1)$,*

$$\mathbb{E}\left[\partial P_t(j,i) \,|\, \mathscr{F}_t\right] = 0.$$

At last, we present the main Theorem of this section.

Theorem 3.2. *[Transition Probability Convergence for directed graphs] Let AS with G being a directed graph. Then, the probability transition matrix of the agent movement converges a.s. to a stationary P_∞. That is, $P_t \xrightarrow{a.s.} P_\infty$ as $t \longrightarrow \infty$.*

Remark 3.5. *In an undirected graph, the probabilities $P_t(j,i)$ can be affected by flow of agents moving inwards to i. Theorem 3.2 relies on the fact that this does not happen to directed graphs. Nevertheless, the authors believe an analogous proof can be established for undirected graphs, using the fact that the edges of the graph are modified by a set of agents that do converge to a fixed distribution.*

Corollary 3.4. *Let AS with G to be a directed connected planar graph. Let $\chi(a) \in \{0,1\}$ be a random variable taking value 1 with probability p_χ if a communication event from agent $a \in N$ takes place, and value 0 otherwise. If $\chi(a)$ is independent of M_t, then $P_t \xrightarrow{a.s.} P_\infty$ as $t \longrightarrow \infty$.*

3.3.3 Convergence Speed

Consider the results of Theorem 3.1. By establishing a minimum weight ε we ensure convergence of the agent distribution as $t \longrightarrow \infty$. Let us recall concepts from Qin et. al. [94].

Theorem 3.3 (Qin et. al. [94])**.** *In addition to Assumption 2.1, if there exists a number $q \in (0,1)$ such that for any $k \in \mathbb{N}_0$ we have $\Pr\left\{\prod_{i=k}^{h} W_i \in \mathscr{M}_2\right\} \geq p > 0$, then the almost sure convergence of the product to a random matrix L is exponential, and the rate is no slower than $(1 - q\kappa^h)^{1/h}$.*

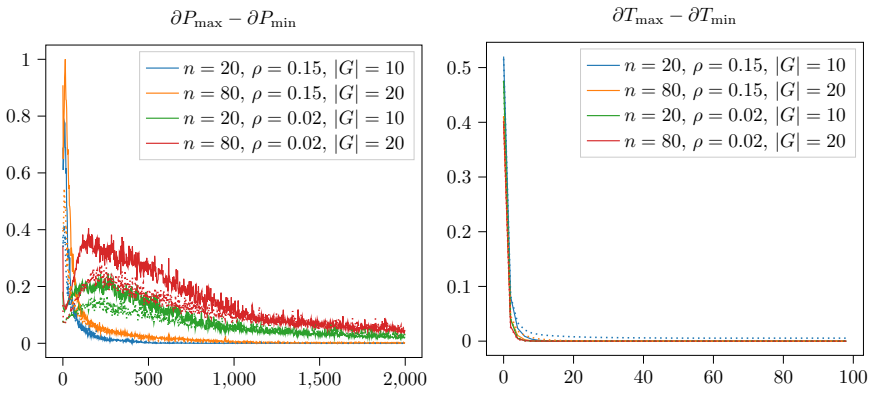
Recall Proposition 3.2. By adding a minimum weight ε, the graph is connected for all t and since there exists at least an odd length cycle,

$$\Pr\left\{\left(\prod_{t=t_0}^{t_0+2\,\mathrm{diam}(G)+1} P_t\right)^{\top} \in \mathscr{M}_2\right\} = 1 \quad \forall t_0.$$

Therefore, with $q = 1$ and $\kappa = \frac{\varepsilon}{1+(\deg^*(G)-1)\varepsilon}$, the convergence rate for an *AS* system with minimum weight ε is no slower than $\left(1 - \alpha^{1+2\,\mathrm{diam}(G)}\right)^{\frac{1}{1+2\,\mathrm{diam}(G)}}$.

3.4 Experiments

To show the convergence results in simulated examples, we restrict our cases to the following baseline scenarios. First, all edge weights are initialised to a uniform value $W(0) = \omega_0 A$, where A is the adjacency matrix and $\omega_0 = 1/|E|$.



(a) P matrix convergence.

(b) $\prod_{t=0}^{k} P_t$ convergence.

Figure 3.2: Directed and Undirected Graphs with $\varepsilon = 0$.

- Directed and undirected triangular planar lattices.
- $|x^0| = |x^g| = 1$. Sets placed randomly in the graph.
- $\text{diam}(G) \in \{10, 20\}$, $|N| \in \{20, 80\}$.
- $\varepsilon \in \{0, \frac{\omega_0}{5}\}$, $\alpha \in \{2 \cdot 10^{-2}, 1.5 \cdot 10^{-1}\}$

We consider $\varepsilon = 0$ for both directed and undirected graphs. This is since, although we only showed P_∞ convergence for directed graphs, by Remark 3.5 there is enough reason to believe it will also converge for directed graphs. For simplicity, we consider only triangular planar lattice graphs. Therefore, there is no need to add a self loop in the graph, and G satisfies the necessary conditions. The choice of low α values is motivated by the size of the graphs. The parameter α influences how fast weights go to zero (or ε). A value of $\alpha = 0.05$ yields a half life time of $t_{1/2} \approx 13$ time steps, and we consider graphs of diameters between 10 and 20.

To show the convergence in the case of P_∞ we plot the values $\partial P_{\max} - \partial P_{\min}$,where $\partial P_{\max} = \max_{i,j}\{P_{t+1} - P_t\}$, and the converse for the minimum. To show convergence of the matrix product to an identical column matrix, let first $\partial T = \left[\prod_{t=0}^{k} P_t\right]_i - \left[\prod_{t=0}^{k} P_t\right]_j$, where $\left[\prod_{t=0}^{k} P_t\right]_i$ is the i-th column of the matrix product, and i and j are chosen at random among all columns. Therefore, to show convergence we plot $\partial T_{\max} - \partial T_{\min}$.

3.4.1 Convergence Results

Figures 3.2a and 3.2b show the convergence results both for the matrix P_t and the product of matrices with $\varepsilon = 0$, and Figure 3.3 shows the convergence of the product for $\varepsilon = \omega_0/5$. Each line represents the average of 50 simulations done with the same parameter set. The colors correspond to a fixed set of parameter in the legend, dotted lines are undirected graphs and full lines directed graphs. Note from Figure 3.2b how the convergence of the matrix product is indeed exponential, and has a very fast convergence rate. However, from Theorem 3.2, we require the minimum weight to be set to zero to ensure the convergence of P_t, but $\varepsilon > 0$ to have convergence in the matrix product. From Figures 3.2a and 3.2b we

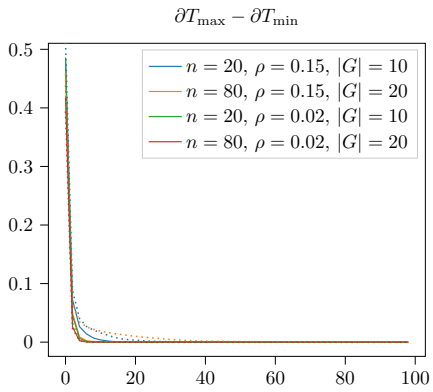


Figure 3.3: Directed and Undirected Graphs with $\varepsilon = \omega_0/5$.

can see that both the matrix product and P_t converge. This is consistent with the results in Theorems 3.2 and 3.1; for convergence to P_∞ we need to set $\varepsilon = 0$. By setting $\varepsilon = 0$ we allow the graph to become virtually disconnected, therefore in some cases the matrix product may not converge to an identical column matrix. Figure 3.3 shows the convergence of the matrix product for $\varepsilon = \omega_0/5$. Note that there does not seem to be much difference in the convergence for $\varepsilon = \omega_0/5$ or $\varepsilon = 0$.

At last, observe that the convergence in P_∞ seems to be much slower and noisy than for y_∞. This is consistent with the fact that y_∞ converges exponentially fast, while for P_∞ we do not have that guarantee, and thus may converge only as $t \longrightarrow \infty$. Observe that the convergence results are extremely similar for both directed and undirected graphs. This confirms the idea pointed out in Remark 3.5. Furthermore, The convergence to a P_∞ transition matrix seems to be heavily influenced by the evaporation rate.

3.5 Discussion

Convergence of ant-inspired biased random walks The results in Section 3.3 show different kinds of convergence for an ant-inspired random walk and what conditions the system needs to satisfy. Convergence of the probability transition matrix to P_∞ is guaranteed for $\varepsilon = 0$. However, to ensure convergence in agent distribution to y_∞, the graph cannot become disconnected (although, as seen in Figures 3.2b and 3.3, convergence in agent distribution seems to occur for most simulations even when $\varepsilon = 0$). This has a straight-forward solution, to be explored in the next Chapter: One can, instead of forcing a minimum weight in the graph, add randomness in the agent decision making. This would effectively allow for the simultaneous guaranteeing of convergence properties for the probability distribution *and* the transition matrix.

Properties of stationary distributions The main question that arises from these results is: How can we know more (and maybe control) the final distribution y_∞, and how do the swarm parameters affect this stationary distribution? This is addressed in the following Chapter: one can consider formulate the *mean field limit* of the multi-agent system,

and take y_t as the agent density. This opens a venue to further study the target distributions y_∞.

Extension to continuous space dynamics We analysed in this Chapter the dynamics of an ant-inspired stochastic system by taking the state space to be a graph, allowing us to deal with discrete time, discrete space stochastic difference equations. A natural question that follows is how would this work apply to continuous space (and continuous time) dynamics. When extending the system to both continuous state and continuous time, one needs to consider the stochastic differential equation (SDE) that emerges as the continuous form of a (biased) random walk. Given the nature of the ant-inspired random walk, the SDE would depend on a time-dependent weight field, affecting both the drift and noise terms. This formulation is not (as the problem stands) trivial to consider but it would have interesting consequences for continuous time learning systems, with connections to optimal control and Hamilton-Jacobi-Bellman equations.

4

Mean Field Dynamics of a Cooperative Foraging Swarm

We have seen in the previous Chapter how, for ant inspired swarm processes, the simultaneous modification of the environment with the adaptive agent behaviour results in undesired dynamical couplings that complicate the analysis and experiments when solving a specific problem or task. This collides with the idea that biologically-inspired robotics rely on simplifying agents and increasing their number to obtain more efficient solutions to such problems, drawing similarities with natural processes. We now zoom in on the problem of a biologically-inspired multi-agent system solving cooperative foraging. We show how mean field techniques can be used to re-formulate such a stochastic multi-agent problem into a deterministic autonomous system, de-coupling agent dynamics and enabling the computation of limit behaviours and the analysis of optimality guarantees. Furthermore, we analyse how having finite number of agents affects the performance when compared to the mean field limit and we discuss the implications of such limit approximations in this multi-agent system, which have impact on more general collaborative stochastic problems.

This chapter is based on 🗎 *D Jarne Ornia, P. J. Zufiria and M. Mazo Jr*, "Mean Field Behavior of Collaborative Multi-Agent Foragers", *IEEE Transactions on Robotics*, vol. 38, no. 4, pp. 2151-2165, Aug. 2022. [123]. All proofs for technical statements can be found in Appendix A.

4.1 Introduction

Biological inspiration has had a significant impact in cooperative robotic problems [124–127]. We now focus our attention on a particular subclass of bio-inspired multi-agent stochastic coordination problems: foraging. Foraging is the problem of locating an unknown target, e.g. a source of food, in an unknown environment, and exploiting the shortest path to such target from a given initial location, e.g. a nest, with the goal of depleting the food source as fast as possible. For an extensive set of stochastic multi-agent methods the foraging problem has served as both a benchmark but also a study subject on itself, given the combined nature of exploration plus optimization that the problem presents [128]. Naturally, many of the biological systems capable of solving foraging problems present some (degree of) de-centralised behaviour. Ants, for example, communicate with each-other only by depositing pheromones on the environment, and achieve global coordination by combining the individual contributions of all members of the swarm without the need of centralised instructions. The mechanism of communicating indirectly through enviromental marking is known as *stigmergy*.

Ant-inspired heuristics have been widely used to solve foraging problems in a distributed fashion [7, 9, 105, 129–131]. However, little is known about the convergence guarantees of such systems, or the influence of hyper-parameters on the agent behaviour patterns. In this Chapter we propose how using so called *mean field approximations* can help us understand, design and analyse stigmergy-based robotic systems. Mean field models have been extensively used in fluid mechanics and particle physics, and more recently in game theory and control [25, 26]. In recent years mean field formulations of large multi-agent systems or swarms have gained increased popularity [30–32] (see also an extensive survey in [132]) as these models abstract away the stochasticity in systems where the number of interacting agents becomes very large.

4.1.1 Main Contribution

The main contributions of this Chapter are then twofold:

Mean Field Foraging System We approximate a multi agent foraging system for a foraging problem as a mean-field non-stochastic process. This helps us provide intuition on the role of the different parameters in a large multi-agent stigmergy swarm.

Distribution of Agents, Rewards and Convergence We derive convergence guarantees that can be attributed to the mean field model of a stigmergy swarm, and provide insight on the resulting shape of the stationary solutions, both for the agent trajectories and the pheromone field. Additionally, we argue that the given mean field approximation yields a qualitative mechanism for *reward shaping*: An explicit relation between rewards and agent distributions for designing reward functions that achieve desired agent behaviours.

4.2 Foraging Swarm Model

We state in this section the statement of a foraging problem over a graph, and present the dynamics of a proposed finite multi-agent system trying to solve the foraging problem.

Consider a swarm of n agents moving over an undirected weighted graph $G = (V, E, W)$ trying to solve a *foraging problem*: the graph has a source vertex $x^0 \in X$ where the agents are initialised, and a goal vertex $x^g \in X$ they are supposed to find, converging to trajectories following the shortest path between x^g and x^0. *Foraging* concerns, in general, both finding the shortest path between two points and depleting a food source as fast as possible. Given the discretised form of the problem, we consider in this Chapter the *foraging problem* to be solved if agents reach a state of steadily following the shortest path between x^0 and x^g, back and forth, since, for real agents that move at constant speed and are able to carry a limited amount of food per trip, this would be the desired scenario for maximal depletion of the food source.

The swarm does not have accurate individual position information (GPS-like data). They can only receive measurements of a weight field from the vertices immediately next to them. Additionally, assume the agents are not able to communicate with any other member of the swarm. The agents are only able to send information to the vertex they are located at, and to receive information only from the neighbouring vertices.

4.2.1 Agent Dynamics

We are interested in solving the foraging problem using only indirect communication through the environment (the graph). It is convenient now to introduce the assumptions that are used throughout this Chapter.

Assumption 4.1. *Any undirected graph G is strongly connected and has at least one odd length cycle.*

Assumption 4.2. *We assume there is only one $x^0 \in X$ and $x^g \in X$, and the distance between them is larger than one.*

Remark 4.1. *Since we use graphs to discretise physical space, we are free to choose a particular discretisation and enforce Assumption 4.1 to be always satisfied (*e.g. *through triangular grids).*

Now, let G be a vertex weighted undirected graph. Let $A \in \{0,1\}^{|X|\times|X|}$ be its adjacency matrix. We define $N := \{1, 2, ..., n\}$ as a set of agents walking from vertex to vertex. The position of agent a at time t is $x_t^a = x$, $x \in X$, and we group them as $x_t := \{x_t^a : a \in N\}$. We define the *empirical distribution* of n agents as $\hat{y}_t^n \in \Delta(X)$ such that for any vertex $x \in X$, $\hat{y}_t^n(i) = \frac{1}{n}|\{a \in N : x_t^a = i\}|$. Since the weights are vertex-based (as opposed to edge-based) we can write $w_t \in \mathbb{R}^{|X|}$ to be the weight vector (and the weight matrix W_t is fully determined by A and w_t).

The agents have a state-dependent policy, such that we write the joint time-dependent transition probability measure of the agent state evolution as $P_t : X \times X \longrightarrow [0,1]$, which depends on some parameters (to be defined). That is, for $i, j \in X$,

$$\Pr\left\{x_{t+1}^a = j \,|\, x_t^a = i\right\} = P_t(j,i), \quad \forall\, a \in A. \tag{4.1}$$

Remark 4.2. *The probability measures P_t are column stochastic matrices. Therefore, when we consider transitions $i \longrightarrow j$, the corresponding probability is $P_t(j,i)$ to avoid using the transposed matrix.*

In stigmergy algorithms, the transition probabilities are usually defined as the normalised weights around a vertex. In our case, drawing inspiration from experimental examples in literature, we model the probabilities of transitioning between vertices with an ε–greedy approach. Define the gradient matrix:

Definition 4.1. *Let $m_i^{out} := |\operatorname{argmax}_k W(i,k)|$. The gradient matrix P_w^{∇} is a stochastic matrix such that*

$$P_w^{\nabla}(j,i) = \begin{cases} \frac{1}{m_i^{out}} & \text{if } W(i,j) = \max_k\{W(i,k)\}, \\ 0 & \text{else.} \end{cases} \tag{4.2}$$

Definition 4.2. *For a minimum probability $\varepsilon > 0$ and graph G we define the ε–greedy matrix as*

$$P_t^{\varepsilon} := \varepsilon(D(A)^{-1}A)^{\top} + (1-\varepsilon)P_{w_t}^{\nabla}.$$

Then, the *foraging agent dynamics* are as follows.

Definition 4.3. *The distribution of agents $y_t \in \Delta(X)$ is the probability of any agent a being on vertex $i \in X$ at time t. This probability evolves as a random walk,*

$$y_{t+1}^n(i) = \Pr\{x_{t+1}^a = i\} = \left(P_t^{\varepsilon} y_t^n\right)_i \quad \forall a \in N, \tag{4.3}$$

with $y_0^n(x^0) = 1$.

From (4.3) we define the indicator vectors

$$\zeta_t^a(i) = \begin{cases} 1 & \text{if } x_t^a = i, \\ 0 & \text{else,} \end{cases}, \quad \hat{y}_t^n = \frac{1}{n}\sum_{a=1}^{n} \zeta_t^a. \tag{4.4}$$

Since P_t^{ε} is the same for all agents, all $\zeta_t^a(i)$ share the same probability distribution for all $t \geq 0$ if $\zeta_0^a = \hat{y}_0^n \ \forall a$. Then, $\hat{y}_t^n$ is a sum of identically distributed random variables with probability distribution y_t.

Remark 4.3. *Equation* (4.3) *can be read as "The probability of having an agent in some vertex i at time $t+1$ is equal to the probability of being in a neighbourhood of i at time t times the probability of moving to i". However, this raises some complications. In our case P_t^{ε} is a stochastic sequence with respect to t and is a function of the state evolution until time t. Therefore, the transition probabilities depend on the entire event history. A way of dealing with this challenge is proposed in Section 4.3.*

4.2.2 Weight Dynamics

The agents also modify the weights in the graph, similar to ants laying pheromones on the ground. Let $R_t \in \mathbb{R}^{|X|\times|X|}$ be the amount of weight added to each vertex at time t (to be properly defined below). Then, the weights in G evolve as

$$w_{t+1} = (1-\alpha)w_t + \alpha R_t \hat{y}_t^n,$$

where $\alpha \in (0,1)$ is a chosen discount factor. The weights are initialised such that $w_0 = \mathbf{1}\omega_0$ with $\omega_0 \geq 0$.

Remark 4.4. *Keeping in mind that these systems are defined over a continuous space in reality, and to avoid over-accumulation of communication (or marking) events in one single vertex, it is useful to consider a saturated form of reinforcement, where we write* (4.6) *as*

$$w_{t+1} = (1-\alpha)w_t + \alpha R_t \operatorname{sgn}\left(\hat{y}_t^n\right).$$

Effectively, this saturates the agent vector such that at every vertex there can be only one "reinforcement" event at a given time. From a real implementation point of view, this is logical since the reinforcement needs to be processed as some form of aggregated signal by an interacting environment or infrastructure, and otherwise such environment would need to process arbitrarily large amount of signals in finite time. Additionally, unbounded accumulation of weights may be undesirable. From this point on, we will retain this formulation.

In order for the swarm to solve the foraging problem, we draw similarities with reinforcement learning approaches to design our reward function R. Let $r \in \mathbb{R}_{\geq 0}$ be some positive constant, and let the goal-based reward $\Sigma_r \in \{0, r\}^{|X|\times|X|}$ such that $\Sigma_r(x^g, x^g) = \Sigma_r(x^0, x^0) = r$, zero elsewhere. With $\gamma \in (0,1)$ being a diffusion parameter, we can write the reward function in diagonal matrix form as

$$R_t := I + \Sigma_r + \gamma \operatorname{diag}_i\left(\max_j W_t(i,j)\right), \tag{4.5}$$

and the weight dynamics are simply

$$w_{t+1} = (1-\alpha)w_t + \alpha R_t \operatorname{sgn}\left(\hat{y}_t^n\right). \tag{4.6}$$

The intuition about this is as follows. The reward diagonal matrix has three explicit terms in each component. First, a constant reward 1 to all vertices to replicate the behaviour of ants: ants add pheromones to every position they are located at, with at least a minimum amount (1 in our case), reflecting that vertex has been visited before. Second, the term Σ_r where the agents reward with an additional amount r the specific goals of our problem: finding x^g and returning to x^0. This is also inspired in entomology; ants may mark the ground with different intensities if they have found food [133, 134]. Finally, the third term is a diffusivity term (pheromones diffuse through the air to their immediate surroundings), and this term makes ants reinforce more or less based on neighbouring weights. Additionally, diffusivity is a commonly used strategy in value function learning problems. When using Q-values, diffusivity represents the maximum utility to be obtained at the next (or previous) step.

With (4.3), (4.4) and (4.6) the stochastic dynamics of the agents and weights are fully defined. We can now present how to use such a model to obtain a foraging swarm.

4.2.3 Foraging Swarm

For the swarm to produce emerging behaviour solving the foraging problem, some additional conditions must be added into the agent behaviour and weight update rules. As proposed in several examples in the literature [17, 135, 136], one way to achieve this is to make use of two different pheromones (or weights w_t^1, w_t^2). In such situation, agents looking for x^g follow w_t^1 (move according to $P^{G^1}(t,\varepsilon) \equiv P_t^{\varepsilon,1}$) and modify w_t^2, while agents

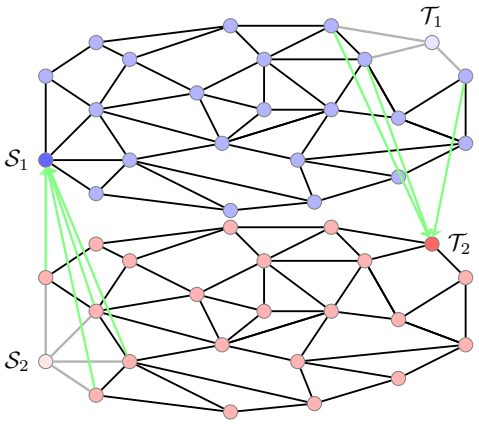


Figure 4.1: Doubled interconnected Graph resulting from constructing P_t^ε.

looking for x^0 follow w_t^2 (move according to $P^{G^2}(t,\varepsilon) \equiv P_t^{\varepsilon,2}$) and modify w_t^1. This implies a certain "memory" condition in the agent behaviour that may look non-Markovian: the agents follow a set of pheromones and reward another depending on their trajectory. But in fact, we can modify the system by duplicating the graph size so it remains "memoryless". The effects of this can be seen in Figure 4.1. In blue we have the weights $w^1(t)$ and in red the weights $w^2(t)$. The green edges represent the new directed edges added to the graph as a result of the interconnection of the two *original* sub-graphs. The intuition behind this "duplication" of the graph is to translate into the size of the state-space the fact that there are 2 simultaneous goals in the foraging problem (finding x^0 and finding x^g). We can retain the memoryless condition of the swarm by duplicating the size of the state-space and interconnecting sub-graphs. For details on how to construct this interconnected graph, see Appendix C.

Definition 4.4. *Given two (original) undirected graphs $G^1 = G^2 = G$, we define a* Foraging Swarm *(FS) as the tuple $(G, x^0, x^g, \hat{y}_t^n, n, \varepsilon, \gamma)$ with $x^0 \in X, x^g \in X, \hat{y} \in \Delta(X), n \in \mathbb{N}, \varepsilon \in \mathbb{R}_{\geq 0}, \gamma \in (0,1)$ such that*

$$\hat{y}_t^n := \begin{pmatrix} \hat{y}_t^{n,1} \\ \hat{y}_t^{n,2} \end{pmatrix}, y_t^n := \begin{pmatrix} y_t^{n,1} \\ y_t^{n,2} \end{pmatrix},$$
$$w_t := \begin{pmatrix} w_t^1 \\ w_t^2 \end{pmatrix}, W(t) := \begin{pmatrix} W^1(t) & 0) \\ 0 & W^2(t) \end{pmatrix},$$
$$P_t^\varepsilon := \begin{pmatrix} (I-T)P_t^{\varepsilon,2} & SP_t^{\varepsilon,1} \\ TP_t^{\varepsilon,2} & (I-S)P_t^{\varepsilon,1} \end{pmatrix},$$

initialised as $w(0) = \mathbf{0}^{2\times|X|}$, $y_0^n(x^0) = 1$, $\hat{y}_0^n = y_0^n$ which follows the dynamics

$$\begin{aligned} y_{t+1}^n &= P_t^\varepsilon y_t^n, \\ w_{t+1} &= (1-\alpha)w_t + \alpha R_t \operatorname{sgn}\left(\hat{y}_t^n\right), \\ R_t &:= I + \Sigma_r + \gamma \operatorname{diag}_i\left(\max_j W_t(i,j)\right) \end{aligned} \tag{4.7}$$

Remark 4.5. *The resulting connected graph in a Foraging Swarm has some edges removed with respect to the original graph G. It effectively disconnects x_1^g and x_2^0 from the rest of the graph. Nevertheless, the density of agents initialised in these vertices is* 0*, and since this is a virtual duplication of the graph, we can simply consider G to have edges $E = \{(ij) \in E_1 \cup E_2 : i, j \neq x_1^g \cup x_2^0\}$ and vertices $X = \{i \in X_1 \cup X_2 : i \neq x_1^g \cup x_2^0\}$. This does not affect the dynamics, and results again in a strongly connected graph. We will refer to $x_2^g \equiv x^g$ and $x_1^0 \equiv x^0$ as the resulting target and starting vertices.*

Observe the weight dynamics in (4.7) present coupled terms between the weights and the agents position, which are both random variables. Furthermore, the support of w_t *depends on n*, which implies that the time evolution of the probability distribution of agents in the graph depends on n and on the sequence of weights w_t (which are, themselves, dependent on the empirical distribution of agents). Therefore, y_{t+1} is a function of the sequence of empirical realisations $\{\hat{y}_t\}$. For this reason, it becomes extremely challenging to analyse the limiting behaviour of the agent distributions. One way to solve this is to study what happens when we consider very large number of agents. With the presented framework of stochastic foraging swarm, we can specify the first problem to solve in further sections.

Problem 4.1. *Consider a foraging swarm communicating based on a double pheromone stigmergy method. Construct a non-stochastic mean field model of the system as $n \longrightarrow \infty$.*

4.3 Mean Field Swarm

In mean field models for Swarm Robotics, the number of agents is assumed to be large enough ($n \longrightarrow \infty$) so that random variables can effectively be replaced by a mean valued deterministic variable. We show here how to do this in the foraging swarm presented in System 4.4. Recall that the state of our system is fully defined by the σ–algebra generated by the proportion of agents in each vertex,

$$\mathscr{F}_t = \sigma(\hat{y}_0^n, \dots, \hat{y}_t^n).$$

Let us define the sequence $\hat{\mathscr{Y}}_t(n) := \{\hat{y}_0^n, ..., \hat{y}_t^n\}$. In this case, an event $\mathscr{Y}_t(n) \in \mathscr{F}_t$ is a sequence of agent proportion vectors until time t resulting in the generator sequence of random variables $\hat{y}_0^n, \dots \hat{y}_t^n$. Now, observe that the conditional expected value of ζ_t^a is

$$\mathbb{E}\left[\zeta^a(t+1) = 1 \mid \mathscr{F}_t\right] = P_t^\varepsilon \zeta_t^a. \tag{4.8}$$

Recall that $\zeta_0^a = y_0^n \ \forall a$, and note that while all ζ_t^a follow the same probability distribution for all $t \geq 0$ they are not independent from each other. The evolution of the probability distribution of every ζ^a follows a product of probability matrices that resembles the dynamics of a Markov process. From (4.3),

$$\Pr\{\zeta_t^a(i) = 1\} = y_t^n(i) = \left(\prod_{t_k=0}^{t} P_{t_k}^\varepsilon y_0^n\right)_i.$$

But in this case, $P(t_k, \varepsilon) = f(\mathscr{Y}_k)$. That is, the sequence of probability transition matrices is a function of the agent positions for all previous times. This means that, in general, for

two different events $\mathscr{Y}_t^m(n), \mathscr{Y}_t^l(n) \in \mathscr{F}_t$,

$$\Pr\{\zeta_{t+1}^a(i) = 1 | \mathscr{Y}_t^m(n)\} \neq \Pr\{\zeta_{t+1}^a(i) = 1 | \mathscr{Y}_t^l(n)\}.$$

Furthermore, observe that the dependence is on the entire sequence until time t. Therefore, in general, the probability of finding agents in each vertex will depend as well on the position of other agents (making their indicator random vectors dependent). Despite this complexity, we can show convergence of the agent proportion vector to its distribution when $n \longrightarrow \infty$.

Theorem 4.1. *Consider a Foraging Swarm from Definition 4.4, and let* $\hat{\mathscr{Y}}_t(n) := \{\hat{y}_0^n, ..., \hat{y}_t^n\}$ *and* $\mathscr{Y}_t(n) = \{y_0^n, y_1^n, ..., y_t^n\}$. *Then,* $\exists \mathscr{Y}_t := \{y_0, y_1, ..., y_t\}$ *such that*

$$\lim_{n\longrightarrow\infty} \hat{\mathscr{Y}}_t(n) = \lim_{n\longrightarrow\infty} \mathscr{Y}_t(n) = \mathscr{Y}_t \text{ a.s.} \quad \forall t \geq 0,$$

and $\mathscr{Y}_t = f(y_0, w_0, t)$.

When the number of agents is taken to infinity, the sequence of empirical distributions converges to its mean, and they both converge to a deterministic sequence $\mathscr{Y}_t$ which is a function of t, the graph and the initial conditions. By making use of Theorem 4.1, one can take the mean field limit and approximate the behaviour of $\hat{y}_t^n$ with y_t as $n \longrightarrow \infty$. Additionally, the indicator variables ζ_t^a become *i.i.d.* When $n \longrightarrow \infty$, there is only one possible sequence $\mathscr{Y}_t$ occurring with probability one. In other words, the sequence $\mathscr{Y}_t$ happens for a set of outcomes of measure 1, and the evolution of the agent density becomes deterministic. This means that the sequence of matrices P_t^ε is also deterministic and independent of every single ζ_t^a. Therefore, the probability distribution y_t of all ζ_t^a becomes independent from individual agent trajectories. This translates into the indicator vectors being *i.i.d.* with respect to each other.

Remark 4.6. *Observe the difference between* y_t^n *(probability distribution of agent positions for a finite number n) and* y_t *(probability distribution of agent positions when* $n \longrightarrow \infty$*). In all cases,* $\hat{y}_0^n = y_0$*, but they can be different from each other for* $t > 0$ *since they evolve according to* P_t^ε*, which implicitly depends on n.*

We can now define our *mean field swarm system.*

Definition 4.5. *Let* $G^1 = G^2 = G$ *be two identical connected weighted graphs. A* mean field foraging swarm *system (mf-FS) is defined as the tuple* $(G, x^0, x^g, y_t, \varepsilon, \gamma)$ *with* $x^0 \in X, x^g \in X, y_t \in \Delta(X), \varepsilon \in [0,1], \gamma \in [0,1]$. *The state variables are initialised as* $w(0) = \mathbf{0}^{|X|}$, $y_0(x^0) = 1$, *and:*

$$\begin{aligned} y_{t+1} &= P_t^\varepsilon y_t, \\ w_{t+1} &= (1-\alpha) w_t + \alpha R_t \operatorname{sgn}\left(y_t\right), \end{aligned} \tag{4.9}$$

These concepts lead us to the second goal of this Chapter.

Problem 4.2. *Consider a mf-FS. Do the mean field dynamics converge to a (sub-optimal) fixed point? Additionally, what can we say (experimentally) about the deviation from the mean field case when choosing a finite number n?*

4.4 Convergence Guarantees

We study next the convergence properties of the mf-FS of System 4.5.

Proposition 4.1. *Let Assumption 4.1 hold. Let* $1 \geq \varepsilon > 0$. *Then, the agent density* y_t *converges exponentially to a stationary density* $y_\infty \in \Delta(X)$, *unique for given initial conditions* y_0 *and* w_0, *that satisfies* $y_\infty(i) > 0 \; \forall i \in X$.

Additionally, we can show the following result. Let $\mathrm{diam}(G)$ be the diameter of the underlying graph.

Lemma 4.1. *For a mf-FS, with* $t_\delta = 2\,\mathrm{diam}(G)$, *it holds that* $\mathrm{sgn}(y_t) = \mathbf{1} \quad \forall t > t_\delta$.

Since there is a minimum probability of accessing any vertex in the graph (and the graph has odd cycles), eventually there is a non-zero amount of agents in every vertex, regardless of the foraging dynamics. This is equivalent to saying that agents have a non-zero probability of accessing every vertex of the graph for all times greater than t_δ.

Remark 4.7. *In fact,* $2\,\mathrm{diam}(G)$ *is an upper bound for the required time* t_δ. *It represents the case where G is one edge away from being bipartite, and to reach some even vertex in odd time it takes* $\mathrm{diam}(G)$ *time steps to reach the (only) odd length cycle plus* $\mathrm{diam}(G)$ *time steps to reach the target vertex. In practice,* $t_\delta \in [\mathrm{diam}(G), 2\,\mathrm{diam}(G)]$.

With these preliminary results, we can present the main contribution of this section.

Proposition 4.2. *There is a unique weight vector* w_∞ *and corresponding matrix* W_∞ *satisfying* $w_\infty := (I + \Sigma_r + \gamma\,\mathrm{diag}_i\big(\max_j W_\infty(i,j)\big))\mathbf{1}$ *for a fixed reward matrix* Σ_r *and* $\gamma \in [0,1)$.

Theorem 4.2. *The weight dynamics in a mf-FS have a fixed point* w_∞, *and*

$$\lim_{t\longrightarrow\infty} w_t = w_\infty := \Big(I + \Sigma_r + \gamma\,\mathrm{diag}_i\big(\max_j W_\infty(i,j)\big)\Big)\mathbf{1}.$$

Corollary 4.1. *The probability transition matrix converges to a unique matrix* $\lim_{t\longrightarrow\infty} P_t^\varepsilon = P_\infty^\varepsilon$, *and the stationary distribution of agents* $\lim_{t\longrightarrow\infty} y_t = y_\infty$ *is the eigenvector corresponding to the eigenvalue 1. That is,*

$$P_\infty^\varepsilon := \lim_{t\longrightarrow\infty} P_t^\varepsilon, \;\; y_\infty = P_\infty^\varepsilon y_\infty. \tag{4.10}$$

4.4.1 On the optimality of solutions

Let us examine now what do the agent distributions look like in a mf-FS system. To this end, we define first a few useful concepts to characterize the state variables.

Definition 4.6. *Let w be the weight vector in a mf-FS. We define a maximum (weight) gradient set of paths* $\mathscr{P}_w^\nabla(i,j)$ *as the set of all paths between vertices i, j satisfying*

$$\begin{aligned} &p \in \mathscr{P}_w^\nabla(i,j) \iff p := \{i, i_2, i_3, ..., i_k, j\}, \\ &i_2 = \mathrm{argmax}_v(W_t(i,v)), \; i_3 = \mathrm{argmax}_v(W_t(i_2,v)), ..., \\ &j = \mathrm{argmax}_v(W_t(i_k,v)). \end{aligned}$$

In other words, let the weight vector be w_t. Then, $\mathscr{P}_{ij}^{\nabla}(w_t)$ is the set of all paths obtained from following the maximum neighbouring weights at each step when going from i to j. Note that, for any two $i, j \in X$, it can be that $\mathscr{P}_{ij}^{\nabla}(w_t) = \emptyset$ if picking the maximum weight neighbour at every step does never connect i with j.

Definition 4.7. *We define the set of optimal weight vectors $\mathscr{W}^* \subset \mathbb{R}_{\geq 0}^{|X|}$ for a mf-FS as*

$$\mathscr{W}^* := \{w^* : \mathscr{P}_{w^*}^{\nabla}(x^0, x^g) \equiv \mathscr{P}(x^0, x^g) \wedge \mathscr{P}_{w^*}^{\nabla}(x^0, x^g) \equiv \mathscr{P}(x^g, x^0)\}.$$

That is, for every weight vector $w^* \in \mathscr{W}^*$, the set of paths resulting from starting at x^0 (x^g) and following the maximum weight lead to x^g (x^0) and is equal to $\mathscr{P}(x^0, x^g)$ ($\mathscr{P}(x^g, x^0)$).

This interpretation of an optimal set of weights is entirely pragmatic. We call a weight distribution optimal if, when starting at x^0 and following the maximum weight vertex at every step, we end up at x^g and we obtain a minimum length path between the two (and vice-versa from x^g to x^0). Additionally, observe that the optimal weight set $\mathscr{W}^*$ is defined for the *doubled* graph in Figure 4.1. Nevertheless, given the symmetry of the graph (the sub-graphs satisfy $G^1 = G^2$), any optimal weight vector $w^* \in \mathscr{W}^*$ generates optimal paths on the original (unweighed) graph too, but it does so separately for paths $x^0 \longrightarrow x^g$ and for paths $x^g \longrightarrow x^0$. Intuitively, constructing a weight vector w^* means the swarm has solved the foraging problem by building a weight function whose gradient always leads towards an optimal path.

Proposition 4.3. *Consider a mf-FS. Then,* $\lim_{t \longrightarrow \infty} w_t = w_\infty \in \mathscr{W}^*$.

From Definition 4.2, abusing the notation for the variable ε we can decompose P_∞^ε in two matrices such that

$$P_\infty^\varepsilon = (1 - \varepsilon)P_\infty^0 + \varepsilon P_\infty^1, \tag{4.11}$$

where P_∞^0 is the transition matrix corresponding to moving according to the gradient of the weights w_∞. Observe as well that P_∞^1 depends only on the adjacency matrix A, which guarantees the decomposition (4.11) to be unique. We define then the following sets.

Definition 4.8. *We define $X_v^{out} = \{j \in X : P_\infty^0(j, v) > 0\}$, and $X_v^{in} = \{j \in X : P_\infty^\infty(v, j) > 0\}$ as the out and in neighbour vertices connected to v by following P_∞^0.*

Observe that in Definition 4.1 we use m to count the number of (out) neighbours that have maximum weight around a vertex, and therefore $m_v^{out} \equiv |X_v^{out}|$. Now let $k = d(x^0, x^g)$ and let us use $m_v^{in} \equiv |X_v^{in}|$. Define $\overline{y} \in \Delta(X)$ to be a probability vector taking values

$$\overline{y}_i := \begin{cases} \frac{1}{2k} & \text{if } i = x^0, x^g, \\ \frac{1}{2k} \sum_{p \in \mathscr{P}(x^0, i)} \prod_{u \in p \setminus i} \frac{1}{m_u^{out}} & \text{if } i \in \cup \mathscr{P}_{w^*}^{\nabla}(x^0, x^g) \setminus \{x^0, x^g\}, \\ \frac{1}{2k} \sum_{p \in \mathscr{P}(x^g, i)} \prod_{u \in p \setminus i} \frac{1}{m_u^{out}} & \text{if } i \in \cup \mathscr{P}_{w^*}^{\nabla}(x^g, x^0) \setminus \{x^0, x^g\}, \\ 0 & \text{else.} \end{cases}$$

The term $\frac{1}{m_u^{out}}$ can be interpreted as the probability of moving out of u towards a specific neighbour, therefore the product $\Pr\{p\} := \prod_{u \in p \setminus i} \frac{1}{m_u^{out}}$ can be interpreted as the probability of following a path p until vertex i, starting at x^0, x^g. Then, we obtain the following result.

Proposition 4.4. *Consider a mf-FS as $t \longrightarrow \infty$ for a fixed $1 > \varepsilon > 0$ and let P^0_∞ defined in* (4.11)*. Then, $P^0_\infty \overline{y} = \overline{y}$.*

Proposition 4.4 indicates that the vector $\overline{y}$ is the first eigenvector of the "gradient" matrix P^0_∞. That is, as the weights converge, when the agents move by selecting the maximum weight vertex around them, the only stationary distribution is the one that spreads all agents equally across the optimal paths between x^0 and x^g.

Theorem 4.3. *Consider a mf-FS. Let $f : [0,\infty) \longrightarrow [0,\infty)$ be $f \in \mathscr{K}_\infty$. Then, it holds that*

$$\|y_\infty - \overline{y}\|_1 \leq f(\varepsilon).$$

That is, the stationary agent distribution of the mf-FS gets arbitrarily close to the optimal distribution as $\varepsilon \longrightarrow 0$.

We can now reflect on the similarities between a Q-Learning (or other value function iteration) strategy for finding optimal policies on a MDP and our weight-based foraging problem, as discussed in the introduction. There is a parallelism between the Q values associated to state-action pairs and the weight values (pheromones) associated to vertices on our graph: In both cases they represent the "utility" of a state. However, in our case, by taking the mean field limit we can study the limit distribution of agents interacting with this utility field, as well as the utility values themselves. Additionally, in the mean field limit we can derive deterministic guarantees about both the *distribution* of agents around the graph (i.e. the distance $\|y_\infty - \overline{y}\|_1$) and the *trajectories* of the agents given by P^ε_∞.

4.5 Experiments

Some experiments are here presented to verify the results presented in Section 4.4. We consider a 20×20 triangular lattice graph, which has $\min_i \deg^{out}(i) = 2$, $\max_i \deg^{out}(i) = 6$ and $\operatorname{diam}(G) = 31$. It is worth mentioning that the amount of "parameter tuning" applied is minimal. The guarantees from Section 4.4 ensure the mean field process converges to (a neighbourhood of) a set of vertices along the shortest path, and we choose parameters to obtain representative results when having a finite amount of agents in the graph. We picked $\gamma = 0.9$ to have high diffusion, $r = 5$ to be significantly higher than the unitary reinforcement in R_t, $\varepsilon = 0.5$ to have an average exploration rate and $\alpha = 0.005$ since this yields an evaporation of $(1-\alpha)^{4\operatorname{diam}(G)} \approx 1/2$.

4.5.1 Mean Field Process

In Figures 4.3 and 4.4 we present the results for two different scenarios of simulations for a mf-FS:

1) One without obstacles where the shortest path is a perfect line between nest (red triangle) and food source (upside red triangle).
2) One with a sample (non-convex) obstacle where the shortest path (or collection of paths) has to go around it on the right side.

At every vertex we plot the value $w^1_t(i) - w^2_t(i)$, with the color bar representing the values of the last plot. In this way we can see the vertices that have a higher overall weight corresponding to each goal. The number of agents is then proportional to the size of the red

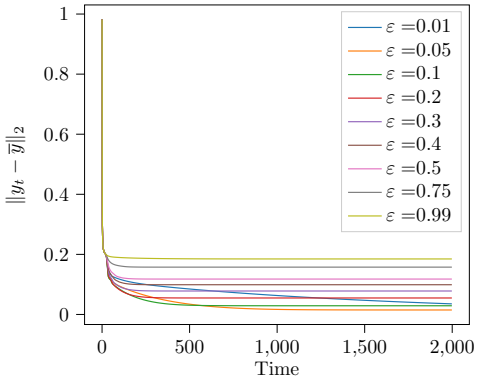


Figure 4.2: Mean Field trajectories compared to $\overline{y}$

markers on the vertices. The behaviour of the system is specially interesting in the case of the obstacles in Figure 4.4. In the first few time-steps there is a random exploration taking place, and very early (around $t = 40$) the agent density starts accumulating in the diagonal line outwards from the nest, indicating that the shortest path is starting to be exploited. Soon after (around $t = 120$) the shortest connecting path can already be observed, but the agent distribution presents oscillations. After enough time-steps the oscillations dampen (since the graph has enough odd length cycles) and the distribution converges to y_∞. It is important to remark the convergence speed of the mean field dynamics; G has 452 vertices, and the agent density has converged to the shortest path in about 300 time-steps. In Figure 4.2 we present the temporal trajectories for the mean field system compared to the optimal vector $\overline{y}$, for different values of ε. Plots for different α and γ values are not included since these parameters did not have an impact on the mean field results. To better isolate the effect of every parameter in the system, the influence of ε is only studied on the mean field system, and later a fixed value of ε is applied to the rest of the experiments.

4.5.2 Finite Agents vs. Mean Field

We compare now the results obtained from a mean field approximation system to the ones obtained when using a finite number of agents. Figures 4.5 and 4.6 show a similar scenario from the mean field case, but in this case for a finite number of agents. As it can be seen, both cases take longer to achieve convergence to the shortest path. Additionally the agents concentrate over wider regions, and some are "trapped" in irrelevant parts of the graph. Other examples in literature [136] solve this by re-setting the agent position if they have not found the goal vertices over a too long period of time.

We now study the impact of having a reduced number of agents compared to the optimal solutions obtained in the mean field case. Let us use the finite sample expectation

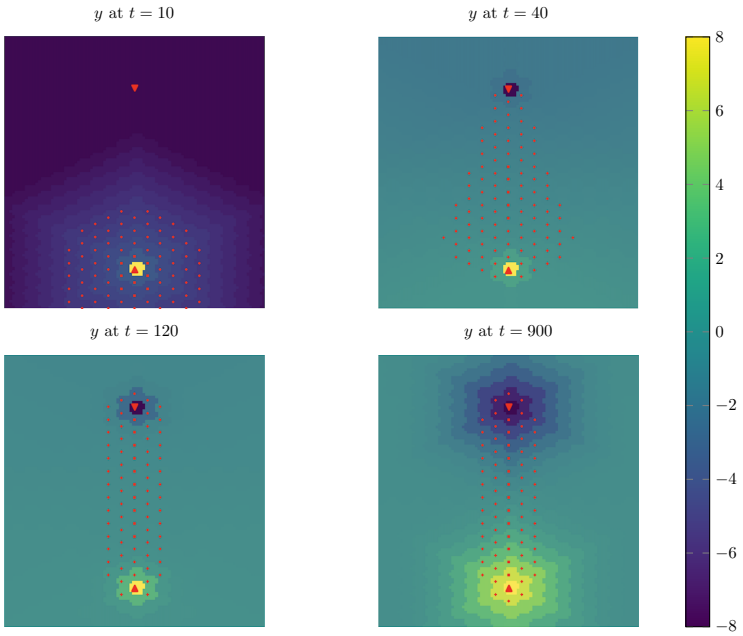


Figure 4.3: Mean Field results without obstables

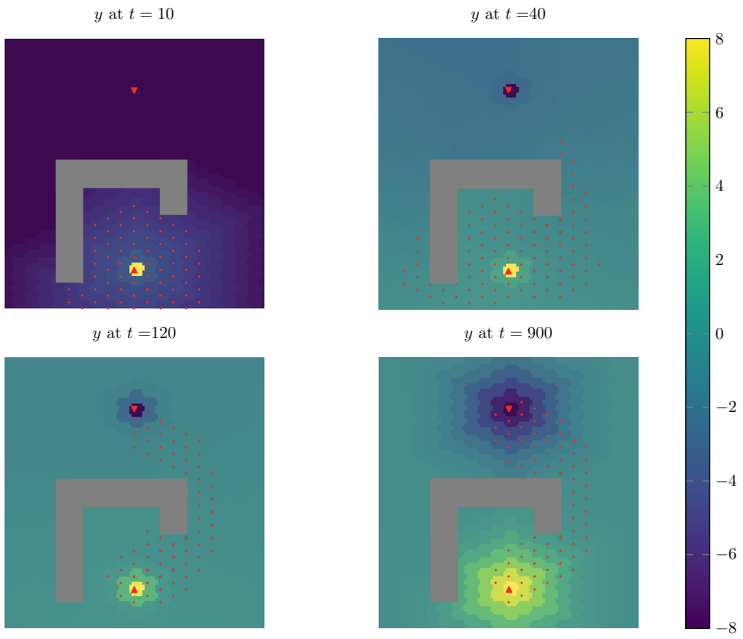


Figure 4.4: Mean Field results with obstacles

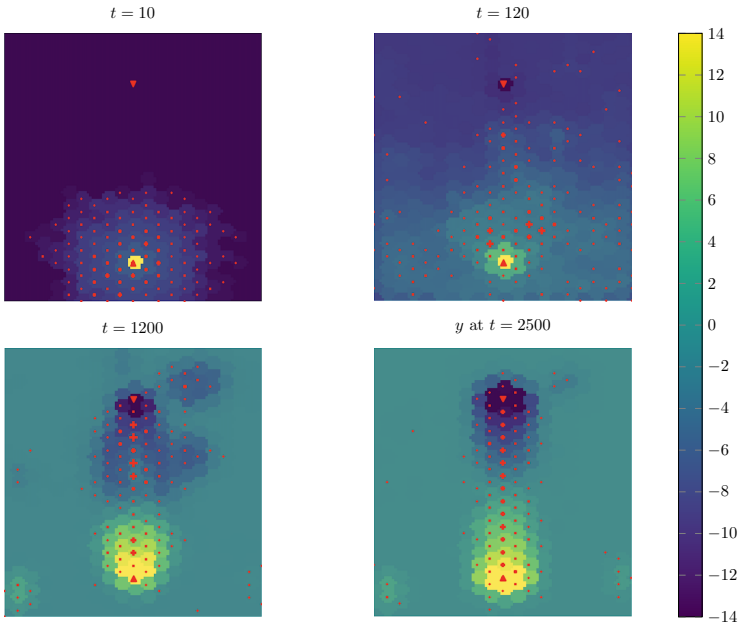


Figure 4.5: Discrete agent swarm with $n = 600$

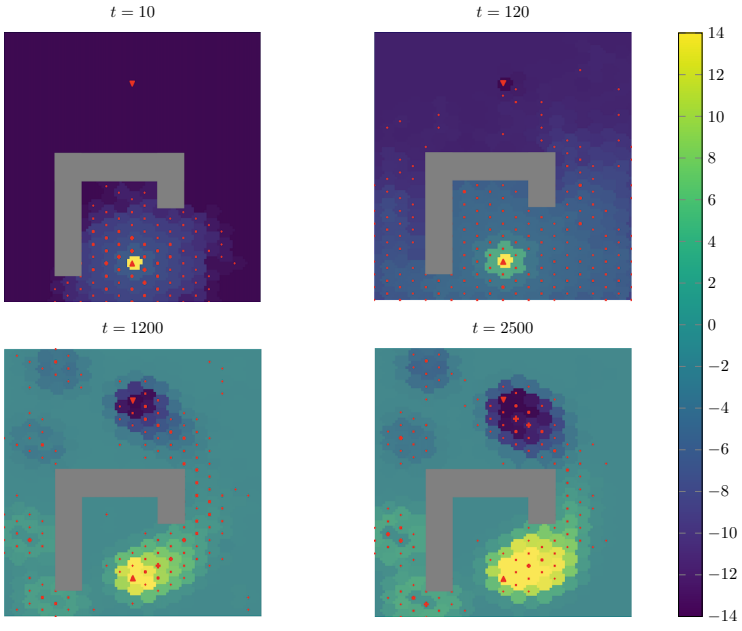


Figure 4.6: Discrete agent swarm with $n = 600$ and obstacles

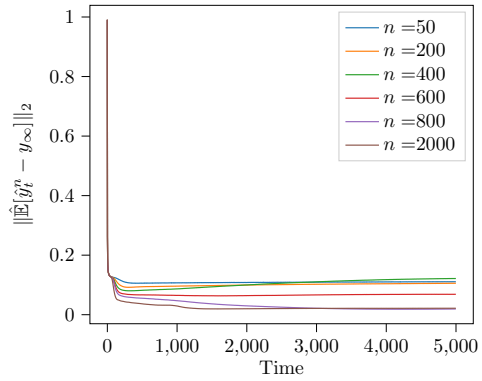


Figure 4.7: Sample expectation of $\hat{y}_t^n - y_\infty$ for different n

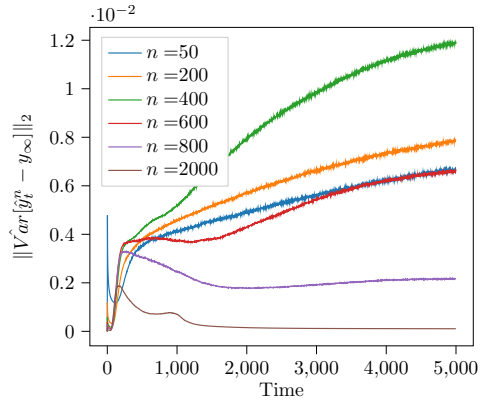


Figure 4.8: Sample variance of $\hat{y}_t^n - y_\infty$ for different n

and variance as

$$\hat{\mathbb{E}}\big[\hat{y}_t^n - y_\infty\big] := \frac{1}{K}\sum_{k=1}^{K} \hat{y}_t^n - y_\infty,$$

$$\hat{\mathrm{Var}}\big[\hat{y}_t^n - y_\infty\big] := \frac{1}{K}\sum_{k=1}^{K} (\hat{y}_t^n - y_\infty - \hat{\mathbb{E}}[\hat{y}_t^n - y_\infty])^2.$$

In Figures 4.7 and 4.8 show the results over $K = 5000$ runs. As expected from Theorem 4.1, both the mean and the variance approach zero for large times as n increases. Interestingly, they both exhibit a peak value after a few time-steps into the runs. This is likely due to the fact that when agents find x^g the weights change quite fast since reward is added to G^2 suddenly, and the stochastic system runs may be prone to differ more from each other.

4.5.3 Interpretation of Variance results

Figures 4.7 and 4.8 show the norms of the finite sample (error) expectation and the finite sample variance. As expected, for large numbers of agents the plots go to zero relatively quickly. However, it is interesting to note that the variance and expectation of error increase with n until $n \approx 600$. A possible justification for this is that there is a threshold under which more agents cause more disorder, but not necessarily better solutions. Looking at the variance values at $t = 5000s$, the first curve for $n = 50$ settles around $0.6 \cdot 10^{-2}$, and the following curves for $n = 200$, $n = 400$ go up until around 10^{-2}. This indicates that the variance increases for a range of n values, until a certain threshold where it decreases until approaching 0 for $n > 1000$.

r	α	n	$\|\hat{\mathrm{E}}[\hat{y}_{\bar{t}}^n - y_\infty]\|_2$	$\|\hat{\mathrm{V}}\mathrm{ar}[\hat{y}_{\bar{t}}^n - y_\infty]\|_2$
20	0.005	200	0.112	0.0097
5	0.05	200	0.121	0.0140
5	0.005	200	0.105	0.0078
20	0.005	800	0.028	0.0006
5	0.05	800	0.022	0.0003
5	0.005	800	0.018	0.0021

Table 4.1: Simulation results

Table 4.1 displays the end results of different combination of parameters over 5000 runs, for $\bar{t} = 5000$ and fixed $\varepsilon = 0.5$, $\gamma = 0.99$. In general, lower α values and larger agent numbers seem to cause smaller variances and smaller $v(t, n)$ values. However, for large swarms ($n = 800$) decreasing the evaporation results in an increase in variance. This effect seems to be caused by the fact that for large enough swarms, higher evaporation actually pulls agents towards the optimal solutions faster, therefore decreasing the variance (or diversity) in trajectories. Interestingly, the impact of r in $\hat{y}_t^n - y_\infty$ seems to be small for the tested cases. Further study of this issue is left for future work, since it may have implications on other multi-agent stochastic systems where stochastic processes exhibit couplings that vanish for large number of agents.

4.6 Discussion

Convergence of mf-FS The mean field foraging system converges to a unique stationary solution, and does so exponentially fast, under the proposed conditions. In fact, the distance between the mean field agent distribution y_∞ and the optimal distribution $\overline{y}$ seems to only depend on the exploration rate ε (see Figure 4.2, Theorem 4.3). That is, the evaporation (or learning) rate α and the discount factor γ do not have an effect in the stationary solutions, nor in the convergence speed of the mean field system. This can be explained by the fact that α and γ act as scaling parameters that do not change the shape of the weight gradients, thus not having an impact on the matrices P_t^ε. Note as well from the results in Figure 4.2 that the distance between y_t and $\overline{y}$ shows some linearity with ε as obtained

in the bounds of Theorem 4.3. This indicates that, for collaborative multi-agent systems in stochastic settings, learning rates and discount factors cease to have an impact when considering large numbers of agents. Therefore, the study of mean field limits on such a multi-agent system allows us to de-couple the influence of some parameters, that may only come into play when considering small agent numbers.

Oscillations and Damping Lower exploration rates cause a much slower spread of agents along the optimal path, resulting in a slowly damped "wave-like" behaviour. These waves are caused by the initial conditions of agents, since all agents start at x^0 on the first sub-graph, but they are more quickly damped (agents spreading out faster) for higher values of ε. This may have an impact when considering finite agent numbers; if we observe fast oscillations for a set of parameters as $n \longrightarrow \infty$, there may be reasons to believe that these can result in non-convergent behaviour for finite agents.

Interpretation of the mean field limit By considering $y_\infty := \lim_{t\to\infty}\left(\lim_{n\to\infty} \hat{y}_t^n\right)$ we are computing the (limit) behaviour in time of an infinitely large system of agents. Our results do not guarantee, however, that the alternate limit $\lim_{n\to\infty}\left(\lim_{t\to\infty} \hat{y}_t^n\right)$ exists as well. The problem of studying this second limit corresponds to the limitations of a mean field approximation, and the study of stochastic trajectories of the finite agent system. Such study would shine some more light on how agents affect the limit distributions in these systems. It is worth mentioning that the impact of mean field solutions on discrete time MDPs is in itself a whole subject of study (see [137–139]), and the interest on such mean field solutions applied to reinforcement learning problems seems to be growing fast in the last years. Knowing more about the relation between the distributions of finite agent systems and their mean field limits will give us tools to design multi-agent systems with guarantees concerning the number of agents needed to solve a specific problem.

5

Event-Based Communication in Shared Value Function Learning

We have seen until now how the problem of navigation and foraging in an ant-inspired multi-agent system can be analysed from a mean-field perspective, allowing us to verify properties of the limiting distributions and behaviour of the agents. We move in this Chapter to study how the collective construction of the pheromone (weight) field can be done through an event-triggered approach, reducing the amount of unnecessary communication. We make a parallelism with a Distributed Q-Learning system, and show how we can used trigger rules inspired by Event Triggered Control techniques to build the shared value function more efficiently. Agents sharing a value function explore an environment (MDP) and compute a triggering signal which they use distributively to decide when to communicate information to a central learner in charge of computing updates on the action-value function. We then apply the proposed algorithm to a cooperative path planning problem, and show how the agents are able to learn optimal trajectories communicating a fraction of the information. Additionally, we discuss what effects (desired and undesired) these event-based approaches have on the learning processes studied, and how they can be applied to more complex multi-agent systems.

This chapter is based on *D. Jarne Ornia and M. Mazo Jr, "Event-Based Communication in Distributed Q-Learning"* 61st IEEE Conference on Decision and Control, *2022* [140]. All proofs for technical statements can be found in Appendix A.

5.1 Introduction

A foraging swarm building a pheromone field to collectively learn how to navigate an unknown environment can be seen as a particular form of a distributed value function learning problem. In particular, since the pheromone field encodes direction information (when agents measure neighbouring weights, they are implicitly querying state-action value functions) one can look into Distributed Q-Learning [141–144] as a general formulation of the problem. These techniques have been applied to many forms of cooperative and non-cooperative problems [145–149]. In the latter, agents are allowed to collaborate to reach higher reward solutions compared to a selfish approach [145, 150]. In model free multi-agent systems, collaboration is often defined as either sharing experiences, value functions or policies, or some form of communication including current state variables of the agents. However such cooperative learning systems often include aggressive assumptions about communication between agents (as pointed out by [49, 151]). Approaches to reduce this communication are, in the framework of federated learning [152], in RL using efficient policy gradient methods [53], limiting the amount of agents (or information) that communicate [49] or allowing agents to learn how to communicate [51, 153]. These approaches focus on defining orthogonal communication actions (similar to auxiliary random variables for finding correlated equilibria in games) or learning graph topologies for the communication network.

5.1.1 Main Contribution

Drawing a parallelism with a networked system, in this Chapter we take inspiration from ETC techniques and turn the communication of a distributed Q-Learning problem [142] event-based, with the goal of reducing communication events, data storage and learning iterations. The difficulty of such problem is that allowing agents to independently decide when to transmit samples may bias the resulting probability distribution of the collected data, and *iid* assumptions do not generally hold. The main contribution of this chapter is then split twofold.

Distributed Trigger-Functions for Communication We propose fully distributed trigger functions that incorporate trajectory memory which agents use to decide when to communicate samples in a distributed Q learning system.

Formal Convergence Guarantees We provide convergence guarantees of the resulting learning algorithm to the optimal Q function fixed point, and show how these guarantees depend (and do not depend) on the design parameters of the system. To the best of our knowledge, such ideas have not been applied before to distributed Q-Learning with convergence guarantees and formal bounds on optimality.

To verify the theoretical results, we analyse experimentally how such event-based techniques result in more efficient learning in a distributed robotic path planning problem.

5.2 Distributed Q-Learning

Let us now consider the case where n agents (actors) perform exploration on the same MDP with a central learner entity, generalized as a *distributed* MDP with (X^n, U^n, P, R).

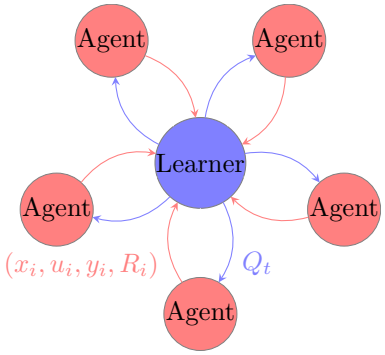


Figure 5.1: Distributed Q-Learning System

Agents gather experiences distributively, possibly following different policies, and send these experiences to the central learner to build a (centralised) Q function. The goal of the distributed nature is to speed up exploration, and ultimately find the optimal policy π^* faster. Such a system may have different architectures regarding the amount of learner entities, parameter sharing between them, *etc.*

Assumption 5.1. *We assume in this Chapter the reward function to be independent of the transitions,* $R : X \times U \longrightarrow \mathbb{R}$.

We consider here a simple architecture where n actors gather samples of the form $s_i = (x, u, R(x, u), y)_i$ following (possibly different) policies π_i. These actors send the experiences to a single central learner, where these are sampled in batches to perform gradient descent steps on a single Q function, and updates each agent's policy π_i if needed. This is a typical architecture on distributed Q-Learning problems where exploring is much less computationally expensive than learning [142].

Definition 5.1. *A* distributed Q-learning *system for a distributed MDP* (X^n, U^n, P, R) *is a set of actor agents* $N = \{1, 2, ..., n\}$ *exploring transitions and initialised at the same* $x_0 \in X$*, together with a single central learner agent storing a* $Q_t : X \times U \longrightarrow \mathbb{R}$ *estimator function. Let the subsets* $N_x = \{i \in N : x_i = x\}$ *have cardinality* n_x*. The estimator function is updated with samples* $s_i \, \forall i \in N$ *as:*

$$Q_{t+1}(x, u) = Q_t(x, u) + \alpha_t \frac{1}{n_x} \sum_{i \in N_x} \left(R(x, u) + \gamma \max_v Q_t(y_i, v) - Q_t(x, u) \right). \tag{5.1}$$

We consider the following Assumption to ensure persistent exploration.

Assumption 5.2. *Any policy* $\pi_i \, \forall i \in N$ *induces an irreducible Markov Chain.*

It is straight-forward to show that the distributed form of the Q-Learning algorithm in (5.1) converges to the optimal Q^* with probability one under the same assumptions as in Theorem 2.6. An example architecture of a distributed Q-Learning system is shown in Figure 5.1.

5.2.1 Problem Definition

In practice, the distributed system in (5.1) implicitly assumes that actors provide their experiences at every step to the central learner that performs the iterations on the estimator Q. When using large amounts of actors and exploring MDP's with large state-spaces, this can result in memory and data transmission rate requirements that scale badly with the number of actors, and can become unmanageable in size. Additionally, when the MDP has a unique initial state (or a small set thereof), the memory may become saturated with data samples that over-represent the regions of the state-space close to x_0. Let us define $\chi_t(i) \in \{0,1\}$ to be an indicator variable of whether agent i communicates information at time t, and let $\chi_t := \sum_i \chi_t(i)$ be the communication rate, and $\chi := \sum_{t=0}^{\infty}$ be the total amount of communication events. From this framework, we present the problem addressed in this Chapter.

Problem 5.1. *For a distributed Q-learning system, design logic rules for the agents to decide when to communicate information to a central learner (and when not to) that maintain convergence and sub-optimality guarantees, and reduce both χ_t and χ.*

We consider the communication to happen from explorers to learner and from learner to explorers (star topology), and a communication event is an agent sending a sample s_a to the central learner. The goal formulated as *reduce* instead of minimise: It is not obvious how to define a minimum χ_t, χ further than the bounds $\chi_t \in [0, N]$ and $\chi = [0, \infty)$, and it will depend on the acceptable trade-off with the optimality of the value functions obtained. Therefore, we attempt here to propose an heuristic that *significantly reduces* both χ_t and χ when compared to the baseline algorithm, while satisfying sub-optimality constraints in terms of the value functions obtained.

5.3 Efficient Distributed Q-Learning

From the convergence proofs of Q-Learning, we know $\lim_{t\to\infty} Q_t(x,u) - Q^*(x,u) = 0$ *a.s.* Each explorer agent obtains samples $s_a = (x, u, R(x,u), y)_a$, and has a function Q_t to use as a policy. For every sample, the agent can compute the estimated loss with respect to the estimator Q_t, which is an indication of how far the estimator is from the optimal Q^*. This suggests that, for $\beta \in (0,1)$, we can define the surrogate function for convergence certification to be a temporal difference (TD) error tracker:

$$L_a(t+1) := (1-\beta)L_a(t) + \beta\left|R(x_a,u_a) + \gamma \max_v Q_t(y_a,v) - Q_t(x_a,u_a)\right|, \forall a \in N, \tag{5.2}$$

with $L_a(0) = 0$. We could now use L_a to trigger communications analogously to the role of Lyapunov functions in ETC. The parameter β serves as a temporal discount factor, that helps the agent track the TD error smoothly. Observe that $L_a(t) \geq 0 \; \forall t, a$, and $L_a(t) \longrightarrow 0 \Longrightarrow Q_t - Q^* \longrightarrow 0$. This last property is an equivalence only in the case that the MDP has deterministic transitions. The intuition about this surrogate function is as follows. Agents compute the TD error term as they move through a trajectory, which gives an indication of how close their Q_t estimator is to the optimal Q^*. Then, they accumulate these losses in a temporal discounted sum $L_a(t)$, such that by storing only one scalar value they can estimate the cumulative loss in the recent past.

Our convergence surrogate function in (5.2) computes the norm of the TD error at each time step, therefore it may not go to zero for stochastic transitions, but to a neighbourhood of zero.

Proposition 5.1. *Consider a distributed Q-Learning problem from Definition 5.1. For a deterministic MDP, $Q_t \longrightarrow Q^*$ a.s. $\Longleftrightarrow L_a(t) \longrightarrow 0$ a.s. For a stochastic MDP, $Q_t \longrightarrow Q^*$ a.s. $\Longrightarrow$ $L_a(t) \longrightarrow \mathscr{L}_0$ a.s., where $\mathscr{L}_0 = [0, \bar{l}]$, and $\bar{l} = \gamma \max_{x,u,y} \mathbb{E}[\max_v Q^*(y,v)|x,u] - \max_v Q^*(y,v)$.*

5.3.1 Event Based Communication

Just as in decentralised ETC [34] a surrogate for stability (Lyapunov function) can be employed to guide the design of communication triggers, we propose here using the distributed signals $L_a(t)$ based on the TD error of the Q estimators. In our problem's context, the actors can be considered to be the sensors/actuators, and the central controller computes the iterations on Q_t based on the samples sent by the actors. This central controller updates everyone's control action (policy π_a). The state variable to stabilize is the difference $Q_t - Q^*$. The control action analogy is the TD step, and applying the control action results in $\|E[Q_{t+1}|\mathscr{F}_t] - Q^*\|_\infty < \|Q_t - Q^*\|_\infty$, where $\mathscr{F}_t$ is a σ–algebra of outcomes over the trajectories (see [154]). To decide when to transmit, we propose to use triggering rules of the form

$$\chi_t(i) := \begin{cases} 1 & \text{if } \left|R(x_a,u_a) + \gamma \max_u Q_t(y_a,u_a) - Q_t(x_a,u_a)\right| \geq \max\{\beta L_a(t), \varepsilon_\chi\} \\ 0 & \text{else,} \end{cases} \tag{5.3}$$

with $\beta \in [0,1]$ andi $\chi_t(i) = 1$ means that agent i sends the sample s_a at time t to the central learner. The event triggered rule in (5.3) has an intuitive interpretation in the following way. Agents accumulate the value of $L_a(t)$ through their own trajectories in time. If the trajectories sample states that are already well represented in the current Q_t function, there is no need to transmit a new sample to the central learner. This can happen for a variety of reasons; some regions of the state-space may be well represented by a randomized initialisation of Q, or some explorers may have already sampled the current trajectory often enough for the learner to approximate it. The resulting system is then an event-based distributed Q-Learning (EBd-Q) system. We divide now the results in stochastic and determinstic MDPs.

5.3.2 Deterministic MDP

Let us first define H to be the operator:

$$H_P\big(Q_t(x,u)\big) := \sum_y P(x,u,y)\big(R(x,u) + \gamma \max_v Q_t(y,v)\big). \tag{5.4}$$

The mapping H_P is a contraction operator on the ∞–norm, with $H_P(Q^*) = Q^*$ being the only fixed point (see [154] for the proof). Now observe, for a deterministic MDP, that the transition $(x,u) \longrightarrow y$ happens for a single y, and $\mathbb{E}_{y\sim P(x,u,\cdot)}[R(x,u) + \gamma \max_v Q_t(y,v) - Q_t(x,u) \mid \mathscr{F}_t] = H_P(Q_t(x,u)) - Q_t(x,u)$.

Theorem 5.1. *Let (X^n, U^n, P, R) be a deterministic MDP with $\|R\|_\infty < \infty$. Let the event triggering condition determining communication events be* (5.3)*. Then, the resulting* EBd-Q *system*

learning on the samples $\{s_a : \chi_t(i) = 1\}$ *converges* a.s. *to* Q^{ε_χ} *satisfying* $\|Q^{\varepsilon_\chi} - Q^*\|_\infty \leq f(\varepsilon_\chi)$ *with* $f(\varepsilon_\chi) \in \mathscr{K}_\infty$.

Remark 5.1. *Observe that for* $\varepsilon_\chi = 0$ *the triggering rule in 5.3 may result in regular (almost periodic) communications as* $t \longrightarrow \infty$. *Setting* $\varepsilon_\chi > 0$ *implies the number of expected communication events goes to* 0 *as* $t \longrightarrow \infty$, *and* $\chi < \infty$ *at the expense of* Q_t *converging to a neighbourhood of* Q^*.

One can show that, in the case of a deterministic MDP, convergence is also guaranteed for the case where the learning rate $\alpha_t = \alpha$ is fixed. In practice, we can consider this to be the case when applying ETC rules on a deterministic MDP.

5.3.3 Stochastic MDP

We now present similar results to Theorem 5.1 for general stochastic transition MDPs. Consider a distributed MDP as in Definition 5.1. Let $\mathscr{P}$ be the set of all possible transition functions for a given set of actions and states, *i.e.* $\mathscr{P} := \{P : X \times U \longrightarrow \Delta(X)\}$. Define $\mathscr{T} := \{(x, u, y) : x, y \in X, u \in U\}$ as the set of transitions for the given state and action set X, U. Let $\psi : 2^{\mathscr{T}} \times \mathscr{P} \longrightarrow \mathscr{P}$ be an operator such that given a subset of transitions $\mathscr{T}_k \subset \mathscr{T}$ and a transition function P sets the probability of all transitions $\mathscr{T}_k$ to zero and normalizes the resulting transition function, and we use $\psi(\mathscr{T}_k, P) \in \mathscr{P}$.

Given the MDP probability measure P, we define the set $\mathscr{P}_P \subset \mathscr{P}$ as the set containing all transition functions P resulting from "deleting" any combination of transitions in P. That is, $\mathscr{P}_P := \{\psi(\mathscr{T}_k, P) : \mathscr{T}_k \in 2^{\mathscr{T}}\}$. Consider now the case where we apply an event triggered rule to transmit samples on a stochastic MDP. For any pair (x, u), agent i and time t, the samples are transmitted (and learned) if $\chi_t(i) = 1$. This means that, in general, it may happen that for the set of resulting states for a pair (x, u), $\{y \in X : P_{xy}(u) > 0\}$, some transitions will not be transmitted. This biases the resulting stochastic approximation process, and in practice this is equivalent to applying different transition functions $P^{\mathscr{T}_k} \in \mathscr{P}_P$ at every step t. This leads to the next assumption.

Assumption 5.3. *There exists probability measure* $\mu_P : \Delta(\mathscr{P}_P)$ *that is only a function of the MDP, the initial conditions* x_0, Q_0 *and the parameters* $\gamma, \varepsilon_\chi, \beta$, *such that* $\mu_P(\psi(\mathscr{T}_k, P))$ *is the probability of applying function* $\psi(\mathscr{T}_k, P)$ *at any time step.*

Remark 5.2. *In fact it follows from the Definition of* $\mathscr{P}_P$ *that the dependence on* β, ε_χ *must exist, given that* $\beta, \varepsilon_\chi = 0 \Longrightarrow \mu_P = 1$*: in this case all samples are always transmitted. In a similar way, it also holds that* $\lim_{\varepsilon_\chi \to \infty} \mu_P(\psi(\mathscr{T}_k, P)) = 0$ $\forall \mu_P(\psi(\mathscr{T}_k, P)) \in \mathscr{P}_P$ *since in such case no samples are ever transmitted.*

Let us reflect on the implications of Assumption 5.3. When applying an event triggered rule in (5.3) to transmit samples, it may result on experiences not being transmitted if the trigger condition is not met. This introduces a bias in the communication of samples, but it can be modelled by considering different transition functions applied at every step (which have some values $\psi(\mathscr{T}_k, P)_{xy}(u) = 0$ compared to the original function P) by every agent. Assumption 5.3 implies that, even though every agent uses different functions at every time-step, the probability of using each $P^{\mathscr{T}_k} \in \mathscr{P}_P$ is measurable for fixed initial conditions.

Remark 5.3. *Assumption 5.3 is necessary to obtain the convergence guarantees presented in the following results. Based on the experimental results obtained, and on the fact that agents use different policies for exploration when generating trajectories, the Assumption seems to hold in the cases explored when large amounts of agents are used. One could interpret this as, for a large enough sample of agents using different policies, the resulting sequence of transition measures observed is "random enough" to be generated by a fixed probability measure. However, we leave this as a conjecture, with the possibility that the assumption could be relaxed to a time-varying distribution over $\mathscr{P}_P$.*

Let us now define the operator $\tilde{H}$ as the expectation

$$\tilde{H}(Q_t(x,u)) := \sum_{P' \in \mathscr{P}_P} \mu_P(P') H_{P'}(Q_t(x,u)) = \mathbb{E}_{P' \sim \mu_P}\left[H_{P'}(Q_t(x,u))\right].$$

For a transition function P, set $\mathscr{P}_P$, and density μ_P, define $\tilde{P} := \sum_{P'} \mu_P(P')P'$. We derive the following results.

Lemma 5.1. *For a given EBd-Q system, $\tilde{P}$ satisfies $\tilde{H}(Q_t(x,u)) = H_{\tilde{P}}\big(Q_t(x,u)\big)$, and the operator has a fixed point $\tilde{H}(\tilde{Q}) = \tilde{Q}$ satisfying $\tilde{Q}(x,u) := \sum_y \tilde{P}(x,u,y)\big(R(x,u) + \gamma \max_v \tilde{Q}(y,v)\big)$.*

Therefore, applying $\tilde{H}(\tilde{Q})$ is equivalent to applying the contractive operator in (5.4) with transition function $\tilde{P}$.

Theorem 5.2. *Consider a distributed Q-Learning system as in Definition 5.1, and let the underlying MDP have stochastic transitions and bounded reward function. Let the event triggering condition determining communication events be* (5.3)*. Then, the resulting* EBd-Q *system learning on the samples $\{s_a : \chi_t(i) = 1\}$ converges* a.s. *to a fixed point $\tilde{Q}$.*

From Remark 5.2 we know that $\beta, \varepsilon_\chi = 0 \implies \mu_P(P) = 1 \implies \tilde{Q} = Q^*$. Additionally, for P_t being the probability transition function applied at time t, it holds that $E[P_t] = \tilde{P}$. But we can say something more about how the difference $P - \tilde{P}$ influences the distance between the fixed points $\|Q^* - \tilde{Q}\|_\infty$.

Corollary 5.1. *Consider a distributed stochastic MDP with an event triggered condition as defined in* (5.3)*. For a given transition function P, a set of functions $\mathscr{P}_P$ and density μ_p, $\exists f \in \mathscr{K}_\infty$ such that $\|Q^* - \tilde{Q}\|_\infty \leq f\left(\|P - \tilde{P}\|_\infty\right)$.*

In fact, the distance $\|P - \tilde{P}\|_\infty$ is explicitly related to the probability measure μ_P, since it determines how far $\tilde{P}$ is from the original P based on the influence of every function in the set $\mathscr{P}_P$. One can show that $\|P - \tilde{P}\|_\infty \leq (1 - \mu_P)|\mathscr{P}_P|$, and $1 - \mu_P$ is a measure of how often we use transition functions different to P, which depends on the aggressiveness of the parameters β, ε_χ. We continue now to study experimentally the behaviour of the Event Based d-Q systems in Theorem 5.1 and 5.2 regarding the communication rates and performance of the policies obtained for a given path planning MDP problem.

5.4 Experiments

To demonstrate the effectiveness of the different triggering functions and how they affect the learning of Q-values over a MDP, we use a benchmark problem consisting of a

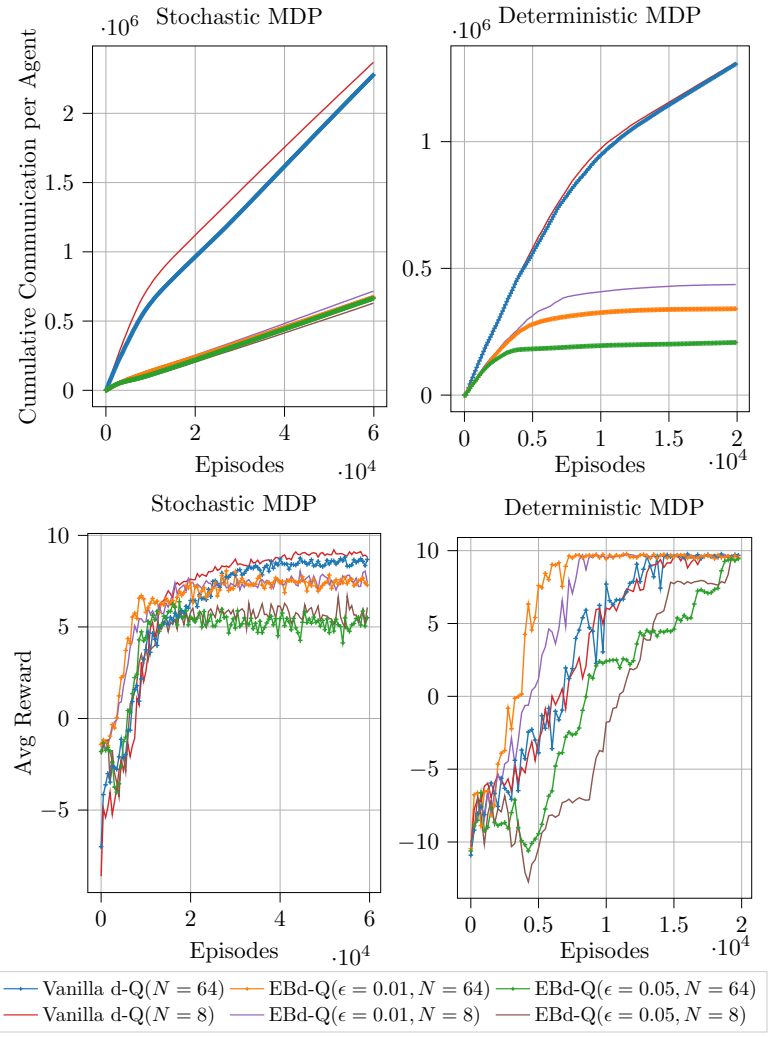


Figure 5.2: Path Planning Learning, Vanilla Distributed Q-Learning vs. EBd-Q

path planning problem. The average reward and communication results for a stochastic and deterministic MDP are presented in Figure 5.2. We use as a benchmark a "vanilla" Distributed Q-Learning algorithm where all agents are communicating samples continuously, and we compare with different combinations of parameters for the presented *EBd-Q* systems. Comparing with other available research is not straight-forward, since it would require interpreting similar methods designed for other problems (in the case of distributed stochastic gradient descent works [36], or policy gradient examples [44, 45]), or comparing with other methods designed for learning speed (e.g. [144]), where the goal is not to save bandwidth or storage capacity.

Analysing the experimental results, in both the stochastic and deterministic MDP scenarios, the systems reach an optimal policy quicker by following an event triggered sample communication strategy, but only for $\varepsilon_\chi = 0.01$. This can be explained by the same principle as in *prioritized sampling* [143, 155]: samples of un-explored regions of the environment are transmitted (and learned) earlier and more often. However, in our case this emerges as a consequence of the trigger functions $\chi_a(t)$, and it is the result of a fully distributed decision process where agents decide independently of each-other when to share information, and does not require to accumulate and sort the experiences in the first place.

When increasing the triggering threshold to $\varepsilon_\chi = 0.05$, the learning gets compromised and the reward decreases for both $n = 64$ and $n = 8$. Additionally, we observe in both scenarios how the total number of communications increase much slower in the event based case compared to the vanilla distributed Q-Learning example, and even stabilize in the case of the deterministic MDP, indicating the number of events is approaching zero. This is due to the *EBd-Q* systems sending a much lower amount of samples through the network per time step. As anticipated by the theoretical results in Theorems 5.1 and 5.2, higher ε_χ results in a larger reduction of communication rates, at the expense of obtaining less optimal Q functions.

Finally, let us comment on the influence of the number of agents in the experiments. First, $n = 8$ has lower communication requirements observable in the case with $\varepsilon_\chi = 0.01$: the total communication number plateaus earlier and at a significantly lower value for $n = 64$. Second, in the deterministic MDP case we see how larger number of agents result in faster reaching of a maximum reward.

5.5 Discussion

We have presented a design of ETC inspired trigger functions for d-Q systems with the goal to allow agents in a such systems to make distributed decisions on which particular experiences may be valuable and which ones not, reducing the amount of communication events (and data transmission and storage).

Convergence Results Regarding the convergence guarantees, we have shown how applying such triggering functions on the communication events results in the centralised learner converging to a Q-Function that may slightly deviate from the optimal Q^*. However, we were able to provide an indication on how far the resulting Q functions can be from Q^* based on the triggering parameters ε_χ, β, explicitly for a deterministic MDP and implicitly (via the distribution μ_P) for a stochastic MDP.

Reduction of Communication on Path Planning problems Event based rules reduce significantly the amount of communication required in the explored path planning problem considered in previous chapters. The weight based graph (pheromone) of previous chapter has been abstracted here to a general MDP scenario with distributed value function learning. In such problems, as seen in the experiments, event-based communication techniques allow the agents to reduce the communication requirements while keeping a reasonable learning speed, even though they intrinsically modify the probability distributions of the data. In fact, it was observed in the experiments how the proposed *EBd-Q*

systems resulted collaterally in a faster learning rate than for the constant communication case (an effect similar to that in *prioritized learning*).

Impact on other MARL problems This Chapter is a first step towards *safely* reducing communication in model-free multi-agent systems, but it still considers a simple scenario with a star topology. It would be valuable to explore the effect of such event based communication on general multi-agent RL systems where agents could be sharing more than experiences (Q–values, policies, state measurements...). A next step in this direction is to consider general MARL problems that require communication in policy roll-out phases. Then, the effect of event-based communication methods is not only on the learning process, but on the controller execution.

6

Robust Event-Based Interactions in Multi-Agent Reinforcement Learning

This chapter presents an approach to reduce the communication required between agents in a Multi-Agent Reinforcement Learning system by exploiting the inherent robustness of the underlying Markov Decision Process. We compute so-called robustness surrogate functions (off-line), that give agents a conservative indication of how far their state measurements can deviate before they need to update other agents in the system with new measurements. This results in fully distributed decision functions, enabling agents to decide when it is necessary to communicate state variables. We derive bounds on the optimality of the resulting systems in terms of the discounted sum of rewards obtained, and show these bounds are a function of the design parameters. Additionally, we extend the results for the case where the robustness surrogate functions are learned from data.

This chapter is based on *D Jarne Ornia and M. Mazo Jr*, "Robust Event-Driven Interactions in Cooperative Multi-Agent Learning", *Formal Modeling and Analysis of Timed Systems. FORMATS 2022.* Lecture Notes in Computer Science, vol 13465. Springer, Cham. [156]. All proofs for technical statements can be found in Appendix A

6.1 Introduction

In the previous chapter we took a step towards distributed event triggered value function learning. However, questions remained regarding how these ideas can be applied to general MARL systems that require communication to solve a problem. We consider now a *centralised training - decentralised execution*, where agents must communicate state measurements to other agents in order to execute the distributed policies. Such a problem represents most real applications of MARL systems: It is convenient to train such systems in a simulator, in order to centrally learn all agents' value functions and policies. But if the policies are to be executed in a live (real) setting, agents will have access to different sets of state variables that need to be communicated with each-other. In this case, having non-reliable communication leads to severe disruptions in the robustness of the distributed policies' performance. The authors in [54] demonstrated experimentally how very small adversarial disruptions in state variable communications leads to a collapse of the performance of general Cooperative MARL systems. In this regard, [55] proposes learning an "adviser" model to fall back on when agents have too much uncertainty in their state measurements, and more recently in [56] the authors enable agents to run simulated copies of the environment to compensate for a disruption in the communication of state variables, and in [57] agents are trained using adversarial algorithms to achieve more robust policies. This lack of robustness in communicative multi-agent learning presents difficulties when trying to design efficient systems where the goal is to communicate less often.

6.1.1 Main Contributions

We consider in this Chapter a general cooperative MARL scenario where agents have learned distributed policies that must be executed in an on-line scenario, and that depend on other agent's measurements.

Reducing Communication in MARL Policy Execution We propose a constructive approach to synthesise communication strategies that minimise the amount of communication required and guarantee a minimum performance of the MARL system in terms of cumulative reward when compared to an optimal policy. We construct so-called *robustness surrogate* functions (robustness certificates), which quantify the robustness of the system to disturbances in agent state variable measurements, allowing for less communication in more robust state regions.

Data Driven Robustness Surrogates Computing robustness certificates is computationally expensive, so we consider the case where these surrogate functions are learned through the *scenario approach* [157, 158], and provide formal guarantees for the reward sum bounds in the case where the functions are approximations learned from data.

6.2 Information sharing between cooperative agents

We consider in this Chapter the following problem. Take a cooperative MMDP with set of agents N, where each agent has learned a distributed policy $\pi_i : X \to U_i$. We are interested in the scenario where the state variable $x_t \in X$ at time t is composed by a set

of joint observations from all agents, and these observations need to be communicated to other agents for them to compute their policies.

Assumption 6.1. *We assume that each agent a has access to a set $X_i \subset X$, such that the observed state for agent a is $x^a \in X_a$. That is, the global state at time t is $x_t = (x_t^1\ x_t^2\ ...\ x_t^n)^T$. Furthermore, we assume that the space X accepts a sup-norm $\|\cdot\|_\infty$.*

We can use $u \in U^n$ to represent a specific joint action $u := \{u^1, u^2, ..., u^n\}$, and $\pi := \{\pi_1, \pi_2, ..., \pi_n\}$ to represent the joint policies of all agents such that $\pi : X \longrightarrow U^n$. The optimal joint (or *centralised*) policy in a cooperative MMDP is the joint policy π^* that maximises the discounted reward sum in the "centrally controlled" MDP, and this policy can be decomposed in a set of agent-specific optimal policies $\pi^* = \{\pi_1^*, \pi_2^*, ..., \pi_n^*\}$. We can similarly consider a value function under a joint policy π, $V^\pi : X \longrightarrow \mathbb{R}$ be $V^\pi(x) = \sum_y P(x, \pi(x), y)(R(x, \pi(x), y) + \gamma V^\pi(y))$.

Assumption 6.2. *Agents have access to optimal policies π^* and a global optimal function Q^* (learned as a result of e.g. a multi-agent actor critic algorithm [149]).*

Remark 6.1. *Assumption 6.1 is satisfied in most MARL problems where the underlying MDP represents some real physical system (*e.g. *robots interacting in a space, autonomous vehicles sharing roads, dynamical systems where the state variables are metric...). In the case where X is an abstract discrete set, we can still assign a trivial bijective map $I : X \longrightarrow \mathbb{N}$ and compute distances on the mapped states $\|x_1 - x_2\|_\infty \equiv \|I(x_1) - I(x_2)\|_\infty$. However, we may expect the methods proposed in this Chapter to have worse results when the states are artificially numbered, since the map a may have no relation with the transition probabilities (we come back to this further in the work).*

Consider the case where at a given time t, a subset of agents $\hat{N}_t \subseteq N$ does not share their state measurements with other agents. Let t_i be the last time agent a transmitted its measurement. We define $\hat{x}_t \in X$ as

$$\hat{x}_t := \left(x_{t_1}^1, x_{t_2}^2, ..., x_{t_n}^n\right). \tag{6.1}$$

That is, $\hat{x}_t$ is the last known state corresponding to the collection of agent states last transmitted, at time t. Then, the problem considered in this Chapter is as follows.

Problem 6.1. *Consider a c-MMDP with a set of optimal shared state policies π^*. Synthesise strategies that minimise the communication events between agents and construct distributed policies $\hat{\pi}$ that keep the expected reward within some bounds of the optimal rewards, these bounds being a function of design parameters.*

6.3 Efficient Communication Strategies

To solve the problem of minimizing communication, we can first consider a scenario where agents can request state measurements from other agents. Consider a c-MMDP where agents have optimal policies Π^*. If agents are allowed to request state observations from other agents at their discretion, a possible algorithm to reduce the communication when agents execute their optimal policies is to use sets of neighbouring states $\mathscr{D} : X \longrightarrow 2^X$ such

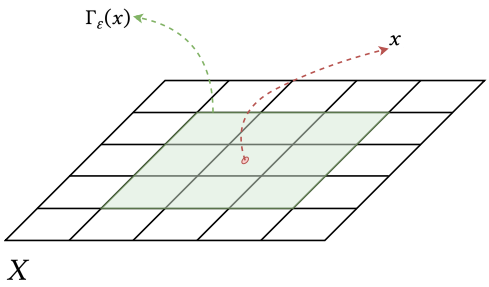


Figure 6.1: Robustness surrogate representation.

that $\mathscr{D}(x) = \{y : \|x - y\| \leq d\}$ for some maximum distance d. Agents could compute such sets for each point in space, and request information from others only if the optimal action changes for any state $y \in \mathscr{D}(x)$. This approach, however, is not practical on a large scale multi agent system. First, it requires agents to request information, which could already be considered a communication event and therefore not always desirable. Additionally, computing the sets "on the fly" has a complexity of $O(|X|^2)$ in the worst case, and it has to be executed at every time-step by all agents. We therefore propose an approach to reduce communication in a MARL system where agents do not need to request information, but instead send messages (or not) to other agents based on some triggering rule.

6.3.1 Event-Driven Interactions

To construct an efficient communication strategy based on a distributed triggering approach, let us first define a few useful concepts. In order to allow agents to decide when is it necessary to transmit their own state measurements, we define the *robustness indicator* $\Gamma : X \longrightarrow \mathbb{R}_{\geq 0}$ as follows.

Definition 6.1. *For a cooperative MMDP with optimal global $Q^* : X \times U^n \longrightarrow \mathbb{R}$, we define the* robustness surrogate $\Gamma_\varepsilon : X \longrightarrow \mathbb{R}_{\geq 0}$ *with sensitivity parameter $\varepsilon \geq 0$ as:*

$$\Gamma_\varepsilon(x) := \max\{d \,|\, \forall y : \|y - x\|_\infty \leq d \Longrightarrow \\ \Longrightarrow Q^*(y, \pi^*(x)) \geq V^*(y) - \varepsilon\}.$$

The function Γ_ε gives a maximum distance (in the sup-norm) such that for any state y which is Γ_ε close to x guarantees the action $\pi^*(x)$ has a Q value which is ε close to the optimal value in y. A representation can be seen in Figure 6.1. Computing the function Γ_ε in practice may be challenging, and we cover this in detail in following sections. In Algorithm 1 we present a self-triggered state sharing approach for agents to send state measurements to each other. Essentially, by having agents check the values of $\Gamma_\varepsilon(\hat{x}_t)$, they can make distributed decisions on when to update others based on how much the expected reward will deviate from the optimal. Using this Algorithm for communication allows us to synthesise the following results.

Proposition 6.1. *Consider a c-MMDP communicating and acting according to Algorithm 1. Let $\hat{x}_t^a$ be the last known joint state stored by agent i at time t, and x_t be the true state at*

Algorithm 1 Self-Triggered State Sharing

Initialise N agents at x_0;
Initialise last-known state vector $\hat{x}_0 = x_0,\ a \in N$
$t = 0$,
while $t < t_{max}$ **do**
  for $a \in N$ **do**
    if $\|x_t(i) - \hat{x}_{t-1}(i)\|_\infty > \Gamma_\varepsilon(\hat{x}_{t-1})$ **then**
      $\hat{x}_t(i) \longleftarrow x_t(i)$
      Send updated $\hat{x}_t(i)$ to all N_{-i};
    end if
    Execute action $\hat{U}_i^* = \pi_i^*(\hat{x})$;
  end for
  $t++$;
end while

time t. Then, it holds:

$$\hat{x}_t^a = \hat{x}_t \ \forall a \in N, \tag{6.2}$$

$$\|\hat{x}_t - x_t\|_\infty \leq \Gamma_\varepsilon(\hat{x}_t)\ \forall t. \tag{6.3}$$

Now let us use $\hat{R}_t = R(x_t, \pi^*(\hat{x}_t), x_{t+1})$ as the reward obtained when using the delayed state $\hat{x}_t$ as input for the optimal policies. We then present the following result.

Theorem 6.1. *Consider a c-MMDP and let agents apply Algorithm 1 to update the delayed state vector $\hat{x}_t$. Then it holds $\forall x_0 \in X$:*

$$\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R(x_t, \pi^*(\hat{x}_t), x_{t+1}) \mid x_0] \geq V^*(x_0) - \varepsilon \frac{\gamma}{1-\gamma}.$$

In other words, Theorem 6.1 indicates that when using Algorithm 1 for self-triggered communication, the expected discounted reward sum will deviate at most $\varepsilon \frac{\gamma}{1-\gamma}$ from the original utility, which is linearly dependent with the design sensitivity ε.

6.4 Robustness Surrogate and its Computation

The computation of the robustness surrogate Γ_ε may not be straight-forward. When the state-space of the c-MMDP is metric, we can construct sets of neighbouring states for a given x. Algorithm 2 produces an exact computation of the robustness surrogate Γ_ε for a given c-MMDP and point x. Observe that in the worst case, Algorithm 2 has a complexity of $O(|X|)$ to compute the function $\Gamma_\varepsilon(x)$ for a single point x. If this needs to be computed across the entire state-space, it explodes to an operation of worst case complexity $O(|X|^2)$, which clearly suffers of the curse of dimensionality. In order to compute such functions more efficiently while retaining probabilistic guarantees, we can make use of the Scenario Approach for regression problems [157].

Algorithm 2 Computation of Robustness Indicator

Initialise x.
Initialise $d = 1$.
Done = *False*
while Not Done **do**
    Compute Set $X^d := \{y : \|x - y\| = d\}$;
    if $\exists y \in X^d : Q^*(y, \pi^*(x)) \leq V^*(y) - \varepsilon$ **then**
        Done = *True*
    else $d++$
    end if
end while
$\Gamma_\varepsilon(x) = d - 1$

6.4.1 Learning the Robustness Surrogate with the Scenario Approach

The data driven computation of the function Γ_ε can be proposed in the terms of the following optimization program. Assume we only have access to a uniformly sampled set $X_s \subset X$ of size $|X_s| = s$. Let $\hat{\Gamma}_\varepsilon^\theta$ be an approximation of the real robustness surrogate parametrised by θ. To apply the scenario approach optimization, we need $\hat{\Gamma}_\varepsilon^\theta$ to be convex with respect to θ. For this we can use a Support Vector Regression (SVR) model, and embed the state vector in a higher dimensional space trough a feature non-linear map $\phi(\cdot)$ such that $\phi(x)$ is a feature vector, and we use the kernel $k(x_1, x_2) = \langle \phi(x_1), \phi(x_2) \rangle$. Let us consider sampled pairs $\{(x_j, y_j)\}_s$, with $y_j = \Gamma_\varepsilon(x_j)$ computed through Algorithm 2. Then, we propose solving the following optimization problem with parameters $\tau, \varrho > 0$:

$$\begin{aligned} \underset{\substack{\theta \in X, \kappa \geq 0, b \in \mathbb{R}, \\ \xi_i \geq 0, j=1,2,\dots,S}}{\text{minimise}} & \left(\kappa + \tau \|\theta\|^2\right) + \varrho \sum_{i=1}^{S} \xi_i, \\ s.t. \quad & \left| y_j - k(\theta, X_j) - b \right| - \kappa \leq \xi_j, \quad j = 1, \dots, s. \end{aligned} \tag{6.4}$$

The solution to the optimization problem (6.4) yields a trade-off between how many points are outside the *prediction tube* $\left| y - k(\theta^*, X_j) - b^* \right| < \kappa^*$ and how large the tube is (the value of κ^*). Additionally, the parameter ϱ enables us to tune how much we want to penalise sample points being out of the prediction tube. Now take $(\theta^*, \kappa^*, b^*, \xi_j^*)$ as the solution to the optimization problem (6.4). Then, the learned robustness surrogate function will be:

$$\hat{\Gamma}_\varepsilon^{\theta^*} := k(\theta^*, X_j) + b^*.$$

From Theorem 3 [157], it then holds for a sample of points X_s and a number of outliers $s^* := |\{(x, y) \in X_S : \left| y - k(\theta^*, x) - b^* \right| > \kappa^*\}|$:

$$\overset{S}{\Pr}\left\{ \underline{\epsilon}(s^*) \leq \Pr\left\{ x : \left| \Gamma_\varepsilon(x) - \hat{\Gamma}_\varepsilon^{\theta^*}(x) \right| > \kappa^* \right\} \leq \overline{\epsilon}(s^*) \right\} \geq 1 - \beta \tag{6.5}$$

where $\underline{\epsilon}(s^*) := \max\{0, 1 - \overline{t}(s^*)\}$, $\overline{\epsilon}(s^*) := 1 - \underline{t}(s^*)$, and $\overline{t}(s^*), \underline{t}(s^*)$ are the solutions to the polynomial

$$\binom{s}{s^*} t^{s-s^*} - \frac{\beta}{2s} \sum_{i=k}^{s-1} \binom{i}{s^*} t^{i-k} - \frac{\beta}{6s} \sum_{i=s+1}^{4S} \binom{i}{s^*} t^{i-s^*} = 0.$$

Now observe, in our case we would like $\Gamma_\varepsilon(x) \geq \hat{\Gamma}_\varepsilon^{\theta^*}(x)$ to make sure we are never over-estimating the robustness values. Then, with probability larger than $1 - \beta$:

$$\begin{aligned} \overline{\epsilon}(s^*) \geq \Pr\left\{x : \left|\Gamma_\varepsilon(x) - \hat{\Gamma}_\varepsilon^{\theta^*}(x)\right| > \kappa^*\right\} \geq \Pr\left\{x : \Gamma_\varepsilon(x) - \hat{\Gamma}_\varepsilon^{\theta^*}(x) < -\kappa^*\right\} = \\ = \Pr\left\{x : \Gamma_\varepsilon(x) < \hat{\Gamma}_\varepsilon^{\theta^*}(x) - \kappa^*\right\}. \end{aligned} \tag{6.6}$$

Therefore, taking $\|x_t(i) - \hat{x}_{t-1}(i)\|_\infty > \hat{\Gamma}_\varepsilon^{\theta^*}(\hat{x}_{t-1}) - \kappa^*$ as the condition to transmit state measurements for each agent, we know that the probability of using an over-estimation of the true value $\Gamma_\varepsilon(x_t)$ is at most $\overline{\epsilon}(s^*)$ with confidence $1 - \beta$.

Then, let $\{\hat{U}_t\}$ be the sequence of joint actions taken by the system. The probability of U_t violating the condition $Q^*(x_t, U_t) \geq V^*(x_t) - \varepsilon$ for any $x_t \in X$ is at most $\overline{\epsilon}(s^*)$. Then, we can extend the results from Theorem 6.1 for the case where we use a SVR approximation as a robustness surrogate. Define the worst possible suboptimality gap $\iota := \max_{x,U} |V^*(x) - Q^*(x, U)|$.

Corollary 6.1. *Let $\hat{\Gamma}_\varepsilon^{\theta^*}$ obtained from* (6.4) *from samples X_S. Then, a c-MMDP communicating as in Algorithm 1 and using trigger condition $\|x_t(i) - \hat{x}_{t-1}(i)\|_\infty > \hat{\Gamma}_\varepsilon^{\theta^*}(\hat{x}_{t-1}) - \kappa^*$ yields, with probability higher than $1 - \beta$:*

$$\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t \hat{r}_t \mid x_0] \geq V^*(x_0) - \delta,$$

with $\delta := (\varepsilon + \overline{\epsilon}(s^*)(\iota - \varepsilon))\frac{\gamma}{1-\gamma}$.

We can interpret the results of Corollary 6.1 in the following way. When using the exact function Γ_ε, the sequence of actions produced ensures that, at all times, an action is picked such that the expected sum of rewards is always larger than some bound close to the optimal. When using the approximated $\hat{\Gamma}_\varepsilon^{\theta^*}$, however, we obtain from the scenario approach a maximum probability of a real point not satisfying the design condition: $\|x - y\| \leq \hat{\Gamma}_\varepsilon^{\theta^*} - \kappa^* \wedge Q^*(y, \Pi^*(x)) < V^*(y) - \varepsilon$. When this happens during the execution of the c-MMDP policies it means that the agents are using delayed state information for which they do not have guarantees of performance, and the one-step-ahead value function can deviate by the worst sub-optimality gap ι.

6.5 Experiments

We set out now to quantify experimentally the impact of Algorithm 1 on the performance and communication requirements of a benchmark c-MMDP system. First of all, it is worth mentioning that the comparison of the proposed algorithm with existing work is not possible since, to the best of our knowledge, no previous work has dealt with the problem

of reducing communication when executing learned policies in a c-MMDP system. For this reason, the results are presented such that the performance of different scenarios (in terms of different Γ_ε functions) is compared with the performance of an optimal policy with continuous communication.

6.5.1 Benchmark: Collaborative Particle-Tag

We evaluate the proposed solution in a typical particle tag problem (or predator-prey) [149]. We consider a simple form of the problem with 2 predators and 1 prey. The environment is a 10×10 arena with discrete states, and the predators have the actions $U_i = \{\text{up}, \text{down}, \text{left}, \text{right}, \text{wait}\}$ available at each time step and can only move one position at a time. The environment has no obstacles, and the prey can move to any of the 8 adjacent states after each time step. The predators get a reward of 1 when, being in adjacent states to the prey, *both* choose to move into the prey's position (tagging the prey). They get a reward of -1 when they move into the same position (colliding), and a reward of 0 in all other situations. A representation of the environment is presented in Figure 6.2. The global state is then a vector $x_t \in \{0, 1, 2, ..., 9\}^6$, concatenating the x, y position of both predators and prey. For the communication problem, we assume each agent is only able to measure its own position and the prey's. Therefore, in order to use a joint state based policy $\pi_i : \{0, 1, 2, ..., 9\}^6 \longrightarrow U$, at each time-step predators are required to send its own position measurement to each other.

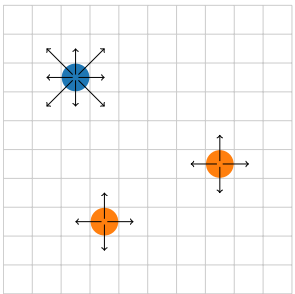


Figure 6.2: Particle Tag, Predators in orange, Prey in blue.

6.5.2 Computation of Robustness Surrogates

With the described framework, we first compute the optimal Q^* function using a fully cooperative vanilla Q-learning algorithm [148], by considering the joint state and action space, such that $Q : \{0, 1, 2, ..., 9\}^6 \times U^2 \longrightarrow \mathbb{R}$. The function was computed using $\gamma = 0.97$. We then take the joint optimal policy as $\pi^*(x) = \text{argmax}_U Q^*(x, U)$, and load in each predator the corresponding projection π_i^*. To evaluate the trade-off between expected rewards and communication events, we compute the function $\hat{\Gamma}_\varepsilon$ by solving an SVR problem as described in (6.4) for different values of sensitivity ε. Then, the triggering condition for agents to communicate their measurements is $\|x_t(i) - \hat{x}_{t-1}(i)\|_\infty > \hat{\Gamma}_\varepsilon^{\theta^*}(x) - \kappa^*$.

The hyper-parameters for the learning of the SVR models are picked through heuristics, using a sample of size $s = 10^4$ to obtain reasonable values of mis-predicted samples s^* and regression mean-squared error scores. Note that $s = \frac{1}{100}|X|$. To estimate the values $\overline{\epsilon}(s^*)$, a coefficient of $\beta = 10^{-3}$ was taken, and the values were computed using the code in [159]. For more details on the computation of ϱ-SVR models (or μ–SVR)[160] see the project code[1].

Figure 6.3 shows a representation of the obtained SVR models for different values of ε, plotted over a 2D embedding of a subset of state points using a t-SNE [161] embedding. It can be seen how for larger ε values, more "robust" regions appear, and with higher values

[1] https://github.com/danieljarne/Event-Driven-MARL

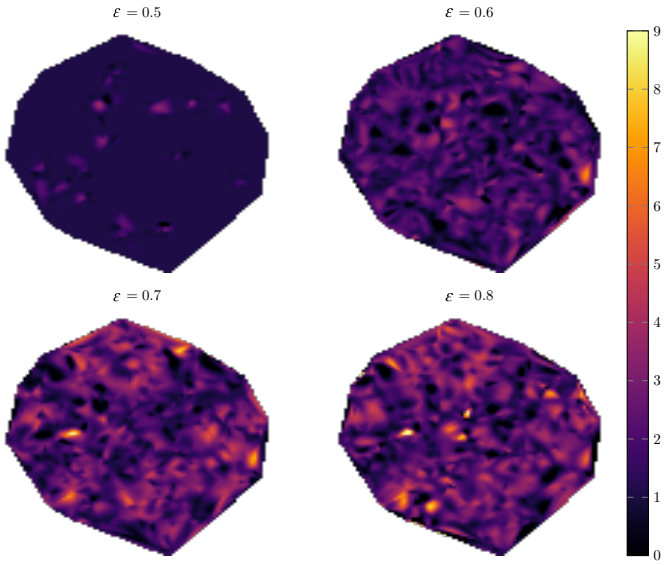


Figure 6.3: Obtained $\hat{\Gamma}_{\varepsilon}^{\theta^*}$ models on a 2D embedding.

of Γ_{ε}. This illustrates how, when increasing the sensitivity, the obtained approximated $\hat{\Gamma}_{\varepsilon}^{\theta^*}$ take higher values almost everywhere in the state space, and form clusters of highly robust points.

6.5.3 Results

The results are presented in Table 6.1. We simulated in each case 1000 independent runs of 100 particle tag games, and computed the cumulative reward, number of communication events and average length of games. For the experiments, $\hat{\mathbb{E}}[\cdot]$ is the expected value approximation (mean) of the cumulative reward over the 1000 trajectories, and in every entry we indicate the standard deviation of the samples (after ±). We use τ_{ε} as the generated trajectories (games) for a corresponding parameter ε, $h(\tau_{\varepsilon})$ as the total sum of communication events per game for a collection of games, and $\overline{g} := \sum_{g \in \tau_{\varepsilon}} \frac{|g|}{|\tau_{\varepsilon}|}$ as the average length of a game measured over the collected τ_{ε}. For the obtained Q^* function, the worst optimality gap is computed to be $\iota = 1.57$.

Let us remark the difference between the 4th and 5th column in Table 6.1. The metric $h(\tau_{\varepsilon})$ is a direct measure of amount of messages for a given value of ε. However, note that we are simulating a fixed number of games, and the average number of steps per game increases with ε: the lack of communication causes the agents to take longer to solve the game. For this reason we add the values $h(\tau_{\varepsilon})/\overline{g}$, which are a measure of total amount of messages sent versus amount of simulation steps for a fixed ε (i.e. total amount of steps where a message *could* be sent). Broadly speaking, $h(\tau_{\varepsilon})$ compares raw amount of information shared to solve a fixed amount of games, and $h(\tau_{\varepsilon})/\overline{g}$ compares amount of messages per time-step (information transmission rate). Note at last that there are two

ε	$\hat{\mathbb{E}}[\sum_{t=0}^{\infty}\gamma^t R_t]$	$\overline{g}$	$h(\tau_\varepsilon)$	$\frac{h(\tau_\varepsilon)}{\overline{g}}$	$\overline{\epsilon}(s^*)$	δ
0	2.72 ± 0.50	10.72 ± 0.44	21.49 ± 0.89	2.00	-	-
0.4	2.72 ± 0.53	10.77 ± 0.44	19.13 ± 0.87	1.78	0.079	16.33
0.5	1.62 ± 0.92	12.45 ± 0.58	16.75 ± 0.85	1.35	0.148	22.39
0.6	0.99 ± 1.09	13.71 ± 0.71	14.49 ± 0.87	1.06	0.205	26.61
0.7	0.93 ± 1.08	13.74 ± 0.69	14.09 ± 0.85	1.03	0.117	26.85
0.8	0.74 ± 1.10	14.65 ± 0.80	14.11 ± 0.87	0.96	0.075	28.10
0.9	0.64 ± 1.08	14.82 ± 0.83	14.33 ± 0.88	0.96	0.097	31.69

Table 6.1: Simulation Results for Event-Triggered Communication in MARL

collaborative players in the game, therefore a continuous communication scheme would yield $h(\tau_\varepsilon)/\overline{g} = 2$.

6.6 Discussion

Summary We have presented an approach to reduce the communication required in a collaborative reinforcement learning system when executing optimal policies in real time, while guaranteeing the discounted sum of rewards to stay within some bounds that can be adjusted through the parameters ε and $\overline{\epsilon}(s^*)$ (this last one indirectly controlled by the learning of data driven approximations $\hat{\Gamma}_\varepsilon^{\theta^*}$). The guarantees were first derived for the case where we have access to *exact* robustness surrogates Γ_ε, and extended to allow for surrogate functions learned through a *scenario approach* based SVR optimization. In the proposed experiments for a 2-player particle tag game the total communication was reduced between $10\% - 44\%$ and the communication rate by $12\% - 52\%$, while keeping the expected reward sum $\hat{\mathbb{E}}[\sum_{t=0}^{\infty}\gamma^t R_t] \in [0.68, 2.76]$.

Experiments From the experimental results we can get a qualitative image of the trade-off between communication and performance. Larger ε values yield a decrease in expected cumulative reward, and a decrease in state measurements shared between agents. Note finally that in the given c-MMDP problem, the minimum reward every time step is $\min R(x_t, U, x_{t+1}) = -1$, therefore a lower bound for the returns is $\mathbb{E}[\sum_{t=0}^{\infty}\gamma^t R_t] \geq -1\frac{\gamma}{1-\gamma} = -32.333$. Then, the performance (even for the case with $\varepsilon = 0.9$ remains relatively close to the optimum computed with continuous communication.

Robustness Surrogates and Conservativeness The computation of the values $\Gamma_\varepsilon(x)$ and the learning of the SVR models for $\hat{\Gamma}_\varepsilon^{\theta^*}(x)$ introduced significant conservativeness with respect to the theoretical bounds. Recall the bound obtained in Corollary 6.1, and observe that for $\varepsilon = 0.5 \implies \delta = 22.39$. On average, $\hat{\mathbb{E}}[V^*(x_0)] \approx 2.72$ when initialising x_0 at random (as seen on Table 6.1). This yields a quite conservative bound of $\hat{\mathbb{E}}[V^*(x_0)] - \delta = -19.67$ on the expected sum of rewards, while the communication events are reduced by around 22% due to the conservative computation of Γ_ε. One first source of conservativeness is in Algorithm 1. When computing the exact value $\Gamma_\varepsilon(x) = d$, it requires every point $y : \|x - y\|_\infty \leq d$

to satisfy the condition in Definition 6.1. The number of states to be checked grows exponentially with d, and many of those states may not even be reachable from x by following the MDP transitions. Therefore we are effectively introducing conservativeness in cases where probably, for many points x, we could obtain much larger values $\Gamma_\varepsilon(x)$ if we could check the transitions in the MDP. Another source of conservativeness comes from the SVR learning process and in particular, the values of κ^*. Since the states are discretised, $\|x_t(i) - \hat{x}_{t-1}(i)\|_\infty \in \{0, 1, 2, 3, ..., 10\}$. Therefore, the triggering condition is effectively constrained to $\|x_t(i) - \hat{x}_{t-1}(i)\|_\infty > \lfloor \hat{\Gamma}_\varepsilon^{\theta^*}(x) - \kappa^* \rfloor$, which makes it very prone to under-estimate even further the true values of $\Gamma_\varepsilon(x)$. Additionally, for most SVR models we obtained predictions $\hat{\Gamma}_\varepsilon^{\theta^*}(x) - \kappa^*$ that are extremely close to the real value, so small deviations in κ^* can have a significant impact in the number of communications that are triggered "unnecessarily". A possible improvement for future work could be to compute the true values $\Gamma_\varepsilon(x)$ through a Monte-Carlo based approach by sampling MDP trajectories. This would yield a much more accurate representation of how "far" agents can deviate without communicating, and the guarantees could be modified to include the possibility that the values $\Gamma_\varepsilon(x)$ are correct up to a certain probability. Another option would be to compute $\Gamma_\varepsilon(x)$ using a different topology through embedding x in some higher dimensional space. At last, we can come back now to the statements in Remark 6.1. It is now evident how having a certain physical structure in the MDP (*i.e.* transition probabilities being larger for states closer in space) would help mitigate the conservativeness. An MDP with large transition jumps with respect to the sup-norm will result in more conservative and less meaningful robustness surrogates.

Complexity versus Robustness The qualitative results of Figure 6.3 have a relevant interpretation when considering the motivation for this Chapter. Generalising conclusions from the studied case, one can see how for any MMDP that requires state variable communication, the structure of the MMDP has a deep impact on how this communication relates to robustness in terms of utility loss due to delayed information. In Figure 6.3 we can see how the state space (even for a relatively simple MMDP) presents a complex structure in terms of regions where communication is required. This hints at the following conclusion: the inherent structure of MMDPs (or MDPs in general) in terms of state-action complexity has a profound impact on robustness properties of the agent policies, and this complexity is not straight-forward to reproduce even for very simple MDPs. These ideas are further explored in the following Chapter.

7

Robust Reinforcement Learning Through Lexicographic Objectives

We have seen until now how the lack of robustness in Reinforcement Learning policies is an organically emerging problem when reducing communication in model free multi-agent systems. However, robustness-inducing algorithms inevitably disrupt the learning dynamics. Robustness may not be desirable at any price; the alterations caused by robustness requirements from otherwise optimal policies should be explainable and quantifiable. In particular, policy gradient algorithms that have strong convergence guarantees are often modified to induce robustness in ways that do not preserve algorithm guarantees, which defeats the purpose of formal robustness requirements. In this Chapter we propose and study a notion of robustness in partially observable MDPs where state observations are perturbed by a noise-induced stochastic kernel, which is a generalisation of the delayed state problem in Chapter 6. We use these notions to propose a robustness-inducing scheme, applicable to any policy gradient algorithm, to formally trade off the reward achieved by a policy with its robustness level through lexicographic optimisation*, which preserves convergence properties of the original algorithm.*

This chapter is based on *D. Jarne Ornia, L. Ribeiro, L. Hammond, M. Mazo Jr and A. Abate,* "Observational Robustness and Invariances in Reinforcement Learning via Lexicographic Objectives", arXiv Pre-Print [162]. All proofs for technical statements can be found in Appendix A.

7.1 Introduction

In Robust RL [83], there is often a trade-off between how robust a policy is and how close it is to the set of optimal policies in its training environment. To address this trade-off in the context of robustness versus observational noise, we define a regret-based robustness notion characterised by a stochastic map to quantify robustness and investigate what makes a policy *maximally robust*. We then propose using a *relaxed lexicographic trade-off*, in which we allow policies to trade performance (discounted reward sum) to robustness. This is done through the use of Lexicographic RL (LRL) [102], which allows policies to deviate from a prioritised objective in favour of obtaining better results for the next objectives priority-wise.

This chapter tackles the study of robustness versus *observational disturbances*, where agents observe a disturbed state measurement and use it as input for the policy. We argue that robustness, being interpreted from a formal perspective, should be quantified and induced in a policy (controller) by modifying the underlying learning process in a verifiable way.

7.1.1 Main Contributions

Most existing work on RL with observational disturbances proposes modifying RL algorithms (learning to deal with perturbations through linear combinations of regularising loss terms or adversarial terms) that come at the cost of *explainability* (in terms of sub-optimality bounds) and *verifiability*, since the induced changes in the new policies result in a loss of convergence guarantees. Our main contributions are summarised in the following points.

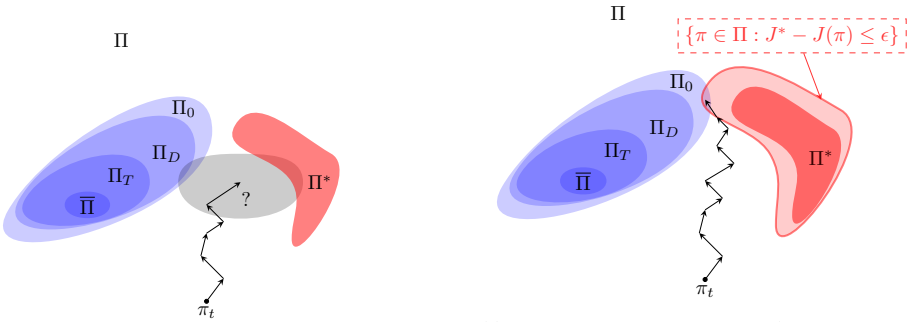
Structure of Robust Policy Sets We consider general unknown stochastic disturbances and formulate a quantitative definition of observational robustness that allows us to characterise the sets of robust policies for any MDP in the form of operator-invariant sets. We analyse how the structure of these sets depends on the MDP and noise kernel, and obtain an inclusion relation (cf. the Inclusion Theorem, Section 7.3) providing intuition into how we can search for robust policies more effectively.[1]

Verifiable Robustness through LRL The proposed characterisation and analysis allows us to cast robustness as a lexicographic optimisation objective and propose a meta-algorithm that can be applied to any existing policy gradient algorithm: Lexicographically Robust Policy Gradient (LRPG). Compared to existing approaches for observational robustness, LRPG allows us to:

1) Retain policy sub-optimality up to a specified tolerance while maximising robustness.
2) Formally control the utility-robustness trade-off through this design tolerance.
3) Preserve formal guarantees of the PG algorithm.

We provide numerical examples on how this approach is applied to existing policy gradient algorithms, comparing them to previous work and verifying how the previously

[1] We claim novelty on the application of such concepts to the understanding and improvement of robustness in disturbed observation RL. Although we have not found our results in previous work, there are strong connections between Sections 7.2-7.3 in this paper and the literature on planning for POMDPs [163, 164] and MDP invariances [165–167].



(a) PG algorithms when robustness terms are added to the cost function *indiscriminately*.

(b) In LRPG, the policy is guaranteed (up to the original algorithm used) to converge to an ϵ ball of Π^*, and from those, the most robust ones.

Figure 7.1: Qualitative representation of the proposed LRPG algorithm, compared to usual robustness-inducing algorithms. The sets in blue are the maximally robust policies to be defined in the coming sections. Through LRPG we guarantee that the policies will only deviate a bounded distance from the original objective, and induce a search for robustness in the resulting valid policy set.

mentioned Inclusion Theorem helps to induce more robust policies while retaining algorithm optimality. Figure 7.1 represents a qualitative interpretation of the results in this work (the structure of the robust sets will become clear in following sections).

7.2 Observationally Robust Reinforcement Learning

Robustness-inducing methods in model-free RL must address the following dilemma: How do we deal with uncertainty without an explicit mechanism to estimate such uncertainty during policy execution? Consider an example of an MDP where, at policy roll-out phase, there is a non-zero probability of measuring a "wrong" state. In such a scenario (even without adversarial uncertainty) optimal policies can be almost useless: measuring the wrong state can lead to executing unboundedly bad actions. This problem is represented by the following version of a noise-induced partially observable Markov Decision Process [164].

Definition 7.1. *An observationally-disturbed MDP (DOMDP) is (a POMDP) defined by the tuple* (X, U, P, R, T, γ) *where* X *is a finite set of states,* U *is a set of actions,* $P : U \times X \mapsto \Delta(X)$ *is a probability measure of the transitions between states and* $R : X \times U \times X \mapsto \mathbb{R}$ *is a reward function. The map* $T : X \mapsto \Delta(X)$ *is a stochastic kernel induced by some unknown noise signal, such that* $T(y \mid x)$ *is the probability of measuring* y *while the true state is* x*, and acts only on the state observations. At last* $\gamma \in [0, 1]$ *is a reward discount.*

In a DOMDP[2] agents can measure the full state, but the measurement will be disturbed by some unknown random signal *in the policy roll-out phase.* Unlike the POMDP setting the agent has access to the true state x during learning of the policies (the simulator is noise-free), and no information about the noise kernel T or a way to estimate it. The difficulty of acting in such DOMDP is that the transitions are actually undisturbed and a

[2]Definition 7.1 is a generalised form of the State-Adversarial MDP used by [86]: the adversarial case is a particular form of DOMDP where T is a probability measure that assigns probability 1 to one state.

function of the true state x, but agents will have to act based on disturbed states $\tilde{x} \sim T(\cdot \mid x)$. We then need to construct policies that will be as robust as possible against noise without being able to construct noise estimates. This is a setting that reflects many robotic problems; we can design a policy for ideal noise-less conditions, and we know that at deployment there will likely be noise, data corruption, adversarial perturbations, *etc.*, but we do not have certainty on the disturbance structure. A (memoryless) policy for the agent is a stochastic kernel $\pi : X \mapsto \Delta(U)$. For simplicity, we overload notation on π, denoting by $\pi(x,u)$ as the probability of taking action u at state x under the stochastic policy π in the MDP, *i.e.*, $\pi(x,u) = \Pr\{u \mid x\}$. The value function of a policy π, $V^\pi : X \mapsto \mathbb{R}$, is given by $V^\pi(x_0) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R(x_t, \pi(x_t), x_{t+1})]$. The action-value function of π (Q-function) is given by $Q^\pi(x,u) = \sum_{y \in X} P(x,u,y)(R(x,u,y) + \gamma V^\pi(y))$. It is well known that, under mild conditions [58], the optimal value function can be obtained by means of the Bellman equation $V^*(x) := \max_u \sum_{y \in X} P(x,u,y)(R(x,u,y) + \gamma V^*(y))$, and an optimal policy is guaranteed to exist such that $\pi^*(x) := \operatorname{argmax}_\pi V^\pi(x) \forall x \in X$. We then define the objective function as $J(\pi) := \mathbb{E}_{x_0 \sim \mu_0}[V^\pi(x_0)]$ with μ_0 being a distribution of initial states, and we use $J^* := \max_\pi J(\pi)$. If a policy is parameterised by $\theta \in \Theta$ we write π_θ and $J(\theta)$.

Assumption 7.1. *For any DOMDP and policy π, the resulting MC is irreducible and aperiodic.*

We now formalise a notion of *observational robustness*. Firstly, due to the presence of the stochastic kernel T, the policy we are applying is altered as we are applying a collection of actions in a possibly wrong state. This behaviour can be formally captured by:

$$\Pr\{u \mid x, \pi, T\} = \langle \pi, T \rangle(x,u) := \sum_{y \in X} T(y \mid x)\pi(y,u), \tag{7.1}$$

where $\langle \pi, T \rangle : X \mapsto \Delta(U)$ is the *disturbed* policy, which averages the current policy given the error induced by the presence of the stochastic kernel. Notice that $\langle \cdot, T \rangle(x) : \Pi \mapsto \Delta(U)$ is an averaging operator yielding the alteration of the policy due to noise. We can then define the *robustness regret*[3]:

$$\rho(\pi, T) := J(\pi) - J(\langle \pi, T \rangle). \tag{7.2}$$

Definition 7.2 (Policy Robustness)**.** *We say that a policy π is* κ-robust *against a stochastic kernel T if $\rho(\pi, T) \leq \kappa$. If π is* 0-robust *we say it is maximally robust. We define the sets of κ-robust policies, $\Pi_\kappa := \{\pi \in \Pi : \rho(\pi, T) \leq \kappa\}$, with Π_0 being the set of maximally robust policies.*

One can motivate the characterisation and models above from a control perspective, where policies use as input discretised state measurements with possible sensor measurement errors. Formally ensuring robustness properties when learning RL policies will, in general, force the resulting policies to deviate from optimality in the undisturbed MDP. With this motivation, we solve the following problem.

Problem 7.1. *For a DOMDP and a given tolerance level ϵ, derive a policy π^ϵ that satisfies $J^* - J(\pi^\epsilon) \leq \epsilon$ as a prioritised objective and is as robust as possible according to Definition 7.2.*

[3]The robustness regret satisfies $\rho(\pi^*, T) \geq 0 \; \forall \, T$, and it allows us to directly compare the robustness regret with the utility regret of the policy.

7.3 Characterisation of Robust Policies

An important question to be addressed, before trying to synthesise robust policies through LRL, is what these robust policies look like, and how they are related to DOMDP properties. The robustness notion in Definition 7.2 is intuitive and it allows us to classify policies. We begin by exploring what are the types of policies that are maximally robust, starting with the set of constant policies and set of fix point of the operator $\langle \cdot, T \rangle$, whose formal descriptions are now provided.

Definition 7.3. *A policy $\pi : X \mapsto \Delta(U)$ is said to be constant if $\pi(x) = \pi(y)$ for all $x, y \in X$, and the collection of all constant policies is denoted by $\bar{\Pi}$. A policy $\pi : X \mapsto \Delta(U)$ is called a fixed point of the operator $\langle \cdot, T \rangle$ if $\pi(x) = \langle \pi, T \rangle(x)$ for all $x \in X$. The collection of all fixed points will be denoted by Π_T.*

In other words, a constant policy is any policy that yields the same action distribution for any state, and a fixed point policy is any policy whose action distributions are unaltered by the noise kernel. Observe furthermore that Π_T *only depends on the kernel* T and the set[4] X. We now present a proposition that links the two sets of policies in Definition 7.3 with our notion of robustness.

Proposition 7.1. *Consider a DOMDP as in Definition 7.1, the robustness notion given in Definition 7.2 and the concepts in Definition 7.3, then we have that*

$$\bar{\Pi} \subseteq \Pi_T \subseteq \Pi_0.$$

The importance of Proposition 7.1 is that it allows us to produce (approximately) maximally robust policies by computing the distance of a policy to either the set of constant policies or to the fix point of the operator $\langle \cdot, T \rangle$, and this is at the core of the construction in Section 7.4. However, before this, let us introduce another set that is sandwiched between Π_0 and Π_T. Let us assume we have a policy iteration algorithm that employs an action-value function Q^π and policy π. The advantage function for π is defined as $A^\pi(x, u) \coloneqq Q^\pi(x, u) - V^\pi(x)$ and can be used as a maximisation objective to learn optimal policies (as in, *e.g.*, A2C [99], A3C [168]). We can similarly define the *noise disadvantage* (a form of negative advantage) of policy π as:

$$D^\pi(x, T) := V^\pi(x) - \mathbb{E}_{u \sim \langle \pi, T \rangle(x)}[Q^\pi(x, u)], \tag{7.3}$$

which measures the difference of applying at state x an action according to the policy π with that of playing an action according to $\langle \pi, T \rangle$ and then continuing playing an action according to π. Our intuition says that if it happens to be the case that $D^\pi(x, T) = 0$ for all states in the DOMDP, then such a policy is maximally robust. And this is indeed the case, as shown in the next proposition.

Proposition 7.2. *Consider a DOMDP as in Definition 7.1 and the robustness notion as in Definition 7.2. If a policy π is such that $D^\pi(x, T) = 0$ for all $x \in X$, then π is maximally robust, i.e., let*

$$\Pi_D := \{\pi \in \Pi : \mu_\pi(x) D^\pi(x, T) = 0 \, \forall \, x \in X\}.$$

[4] There is a (natural) bijection between the set of constant policies and the space $\Delta(U)$. The set of fixed points of the operator $\langle \cdot, T \rangle$ also has an algebraic characterisation in terms of the null space of the operator $\mathrm{Id}(\cdot) - \langle \cdot, T \rangle$. We are not exploiting the later characterisation in this paper.

then we have that $\Pi_D \subseteq \Pi_0$.

So far we have shown that both the set of fixed points $\overline{\Pi}$ and the set of policies for which the disadvantage function is equal to zero Π_D are contained in the set of maximally robust policies. More interesting is the fact that the inclusion established in Proposition 7.1 and the one in Proposition 7.2 can be linked in a natural way through the following Inclusion Theorem.

Theorem 7.1 (Inclusion Theorem)**.** *For a DOMDP with noise kernel* T*, consider the sets* $\overline{\Pi}, \Pi_T, \Pi_D$ *and* Π_0*. Then, the following inclusion relation holds:*

$$\overline{\Pi} \subseteq \Pi_T \subseteq \Pi_D \subseteq \Pi_0.$$

Additionally, the sets $\overline{\Pi}, \Pi_T$ *are* convex *for all MDPs and kernels* T*, but* Π_D, Π_0 *may not be.*

Let us reflect on the inclusion relations of Theorem 7.1. The inclusions are in general not strict, and in fact the geometry of the sets (as well as whether some of the relations are in fact equalities) is highly dependent on the reward function, and in particular on the complexity (from an information-theoretic perspective) of the reward function. As an intuition, less complex reward functions (more uniform) will make the inclusions above expand to the entire policy set, and more complex reward functions will make the relations collapse to equalities. The following Corollary illustrates this.

Corollary 7.1. *For any* ergodic *DOMDP there exist reward functions* $\overline{R}$ *and* $\underline{R}$ *such that the resulting DOMDP satisfies:*

(i) $\Pi_D = \Pi_0 = \Pi$ *(any policy is max. robust) if* $R = \overline{R}$*,*

(ii) $\Pi_T = \Pi_D = \Pi_0$ *(only fixed point policies are maximally robust) if* $R = \underline{R}$*.*

We can now summarise the insights from Theorem 7.1 and Corollary 7.1 in the following conclusions: (1) The set $\overline{\Pi}$ is maximally robust, convex and *independent of the DOMDP*, (2) The set Π_T is maximally robust, convex, includes $\overline{\Pi}$, and its properties *only depend* on T, (3) The set Π_D includes Π_T and is maximally robust, but its properties *depend on the DOMDP*.

7.4 Robustness through Lexicographic Objectives

We have now characterised robustness in a DOMDP and explored the relation between the sets of policies that are robust according to the definition proposed. We have seen in the Inclusion Theorem that several classes of policies are maximally robust, and our goal now is to connect these results with lexicographic optimisation. To be able to apply LRL results to our robustness problem we need to first cast robustness as a valid objective to be maximised, and then show that a stochastic gradient descent approach would indeed find a global maximum of the objective, therefore yielding a maximally robust policy. Then, this robustness objective can be combined with a primary reward-maximising objective $K_1(\theta) = \mathbb{E}_{x_0 \sim \mu_0}[V^{\pi_\theta}(x_0)]$ and any algorithm with certified convergence to solve Problem 7.1. Policy-based LRL (PB-LRL) allows us to encode the idea that, when learning how to solve an RL task, robustness is important but *not at any price*, *i.e.*, we would like to solve

the original objective reasonably well[5], and from those policies efficiently find the most robust one.

7.4.1 Robustness Objectives

We propose now a valid lexicographic objective for which a minimising solution yields a maximally robust policy. For this, we will perturb the policy during training according to the following logic. In the introduction, we emphasised that the motivation for this work comes partially from the fact that we may not know T in reality, or have a way to estimate it. However, the theoretical results until now depend on T. Our proposed solution to this lies in the results of Theorem 7.1. We can use a *design* generator $\tilde{T}$ to perturb the policy during training such that $\tilde{T}$ has the smallest possible fixed point set (i.e. the constant policy set), and any algorithm that drives the policy towards the set of fixed points of $\tilde{T}$ *will also drive the policy towards fixed points of* T: from Theorem 7.1, $\Pi_{\tilde{T}} \subseteq \Pi_T$.

Assumption 7.2. *The design kernel* $\tilde{T}$ *satisfies* $\Pi_{\tilde{T}} = \overline{\Pi}$

We discuss further the choice and implications of using a design kernel $\tilde{T}$ in Section 7.5. One of the messages of the Inclusion Theorem is the fact that fixed point policies are maximally robust. Consider the objective to be minimised:

$$K_{\tilde{T}}(\theta) = \sum_{x \in X} \mu_{\pi_\theta}(x) \frac{1}{2} \| \pi_\theta(x) - \langle \pi_\theta, \tilde{T} \rangle(x) \|_2^2, \tag{7.4}$$

Notice that optimising (7.4) projects the current policy onto the set of fixed points of the operator $\langle \cdot, \tilde{T} \rangle$, and due to Assumption 7.1, which requires $\mu_{\pi_\theta}(x) > 0$ for all $x \in X$, the optimal solution is equal to zero if and only if there exists a value of the parameter θ for which the corresponding π_θ is a fixed point of $\langle \cdot, \tilde{T} \rangle$. In practice, the objectives are computed for a batch of trajectory sampled states $X_s \subset X$, and averaged over $\frac{1}{|X_s|}$; we denote these approximations with a hat. By applying standard stochastic approximation arguments, we can prove that convergence is guaranteed for a SGD iteration using $\nabla_\theta \hat{K}_{\tilde{T}}(\theta)(x) = (\pi_\theta(x) - \pi_\theta(y)) \nabla_\theta \pi_\theta(x)$, $y \sim \tilde{T}(\cdot \mid x)$ to the optimal solution of problem 7.4.

Lemma 7.1. *Let* π_θ *be a fully-parameterised policy in a DOMDP, and* α_t *a learning rate. Consider the following* approximated *gradient for objective* $K_{\tilde{T}}(\pi)$ *and sampled point* $x \in X$*:*

$$\nabla_\theta \hat{K}_{\tilde{T}}(\theta)(x) = (\pi_\theta(x) - \pi_\theta(y)) \nabla_\theta \pi_\theta(x), \quad y \sim \tilde{T}(\cdot \mid x). \tag{7.5}$$

Then, the following iteration with $x \in X$ *and some initial* θ_0*,*

$$\theta_{t+1} = \theta_t - \alpha_t \nabla_\theta \hat{K}_{\tilde{T}}(\theta_t) \tag{7.6}$$

yields $\theta \longrightarrow \tilde{\theta}$ almost surely *where* $\tilde{\theta}$ *satisfies* $K_{\tilde{T}}(\tilde{\theta}) = 0$.

[5] The advantage of using LRL is that we need not know in advance how to define "reasonably well" for each new task. Additionally, we obtain a hyper-parameter that directly controls the trade-off between *robustness and optimality*: the tolerance ϵ. Through ϵ we determine how far we allow our resulting policy to be from an optimal policy in favour of it being more robust.

7.4.2 Lexicographically Robust Policy Gradient

We present now the proposed LRPG meta-algorithm to achieve lexicographic robustness for any policy gradient algorithm at choice.

Algorithm 3 LRPG

input MDP, $\tilde{T}$, ϵ
intialise θ, critic (if using), λ, $\{\beta^1, \beta^2, \eta\}$
set $t = 0$, $x_t \sim \mu_0$
while $t <$ max_iterations **do**
    perform $u_t \sim \pi_\theta(x_t)$
    observe r_t, x_{t+1}
    if $\hat{K}_1(\theta)$ not converged **then** $\hat{k}_1 \longleftarrow \hat{K}_1(\theta)$
    end if
    update critic (if using)
    update θ and λ using (2.2)
end while
output θ

From [102], the convergence of PB-LRL algorithms is guaranteed as long as the original policy gradient algorithm (such as PPO [169] or A2C [170, 171]) for each single objective converges. We can then combine Lemma 7.1 with these results to guarantee that Lexicographically Robust Policy Gradient (LRPG), Algorithm 3, converges to a policy that maximise robustness while remaining (approximately) optimal with respect to R.

7

Theorem 7.2. *Consider a DOMDP as in Definition 7.1 and let π_θ be a parameterised policy. Take $K_1(\theta) = \mathbb{E}_{x_0 \sim \mu_0}[V^{\pi_\theta}(x_0)]$ to be computed through a chosen algorithm (*e.g.*, A2C, PPO) that optimises $K_1(\theta)$, and let $K_2(\theta) = -K_{\tilde{T}}(\theta)$. Given an $\epsilon > 0$, if the iteration $\theta \longleftarrow \text{proj}_\Theta\left[\theta + \nabla_\theta \hat{K}_1\right]$ is guaranteed to converge to a parameter set θ^* that maximises K_1, and hence J (locally or globally), then LRPG converges* a.s. *under PB-LRL conditions to parameters θ^ϵ that satisfy:*

$$\theta^\epsilon \in \text{argmin}_{\theta \in \Theta'} K_{\tilde{T}}(\theta) \quad \textit{such that} \quad K_1^* \geq K_1(\theta^\epsilon) - \epsilon, \tag{7.7}$$

where $\Theta' = \Theta$ if θ^* is globally optimal and a compact local neighbourhood of θ^* otherwise.

We reflect again on Figure 7.1. The main idea behind LRPG is that by formally expanding the set of acceptable policies with respect to K_1, we may find robust policies more effectively while guaranteeing a minimum performance in terms of expected rewards. This addresses directly the premise behind Problem 7.1. In LRPG the first objective is still to minimise the distance $J^* - J(\pi)$ up to some tolerance. Then, from the policies that satisfy this constraint, we want to steer the learning algorithm towards a maximally robust policy, and we can do so without knowing T as long as Assumption 7.2 is satisfied.

7.5 Assumptions on T

A natural question following Section 7.4.1 and the theoretical results in Section 7.4 is how to choose $\tilde{T}$, and how the choice influences the resulting policy robustness towards any

other true T. In general, for any arbitrary policy utility landscape in a given MDP, there is no way of bounding the distance of the resulting policies for two different noise kernels T_1, T_2. As a counter-example, consider an MDP where there are 2 possible optimal policies π_1^*, π_2^*, and take these two policies to be maximally different, *i.e.* $D_{TV}(\pi_1^* \| \pi_2^*) = 1 \; \forall x \in X$. Then, when using LRPG to obtain a robust policy, a slight deviation in the choice of $\tilde{T}$ can cause the gradient descent scheme to deviate from converging to π_1^* to converging to π_2^*, yielding in principle a completely different policy. However, *the optimality of the policy* remains bounded: Through LRPG guarantees we know that, for both cases, the utility of the resulting policy will be at most ϵ far from the optimal. We can, thus, state the following.

Corollary 7.2. *Take T to be any arbitrary noise kernel, and $\tilde{T}$ to satisfy Assumption 7.2. Let π be a policy resulting from a LRPG algorithm. Assume that $\min_{\pi' \in \Pi_{\tilde{T}}} D_{TV}(\pi \| \pi') = a$ for some $a < 1$. Then, it holds for any T that $\min_{\pi' \in \Pi_T} D_{TV}(\pi \| \pi') \leq a$.*

That is, when using LRPG to obtain a robust policy π, the resulting policy is at most a far from the set of fixed points (and therefore a maximally robust policy) with respect to the true T. This is the key argument behind our choices for $\tilde{T}$: A priori, the most sensible choice is a kernel that has no other fixed point than the set of constant policies.

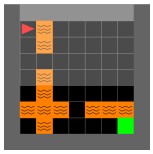
Remark 7.1. *This fixed point condition is satisfied in the discrete state case for any $\tilde{T}$ that induces an irreducible Markov Chain, and in continuous state for any $\tilde{T}$ that satisfies a reachability condition (*i.e. *for any $x_0 \in X$, there exists a finite time for which the probability of reaching any ball $B \subset X$ of radius $r > 0$ through a sequence $x_{t+1} = T(x_t)$ is measurable). This holds for (additive) uniform or Gaussian disturbances.*

7.6 Experiments

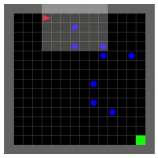
We verify the theoretical results of LRPG in a series of experiments on discrete state/action safety-related environments [172]. *Minigrid-LavaGap*, *Minigrid-LavaCrossing* are safe exploration tasks where the agent needs to navigate an environment with cliff-like regions and receives a reward of 1 when it finds a target. *Minigrid-DynamicObstacles* is a dynamic obstacle-avoidance environment where the agent is penalised for hitting an obstacle, and gets a positive reward when finding a target. *Minigrid-LavaGap* is small enough to be fully observable, and the other two environments are partially observable. In all cases observations consist of a 7×7 field of view in front of the agent, with 3 channels encoding the color and state of objects in the environment. We use A2C [58] and PPO [95] for our implementations of LRPG which we denote by LR-PPO and LR-A2C, respectively. In all cases, the lexicographic tolerance was set to $\epsilon = 0.99\hat{k}_1$ to deviate as little as possible from the primary objective.

Sampling $\tilde{\mathbf{T}}$ To simulate $\tilde{T}$ we disturb x as $\tilde{x} = x + \xi$ for (1) a uniform bounded noise signal $\xi \sim \mathcal{U}_{[-b,b]}$ ($\tilde{T}^u$) with $b = 2$ (1.5 for LavaCrossing) and (2) and a Gaussian noise ($\tilde{T}^g$) such that $\xi \sim \mathcal{N}(0, 0.5)$. We test the resulting policies against a noiseless environment ($\emptyset$), a kernel $T_1 = \tilde{T}^u$ and a kernel $T_2 = \tilde{T}^g$. The main point of these combinations is to also test the policies when the true noise T is similar to $\tilde{T}$.

(a) *MiniGrid-LavaGap*



(b) *MiniGrid-LavaCrossing*



(c) *MiniGrid-DynamicObstacles*

Figure 7.2: Screenshots of the environments used.

General Robustness Results. Firstly, we investigate the robustness of four algorithms where we do not have a Q function. If we do not have an estimator for the critic Q^π, Proposition 7.1 suggests that minimising the distance between π and $\langle \pi, T \rangle$ can serve as a proxy to minimise the robustness regret. We consider the algorithms:

1) Vanilla PPO (noiseless).
2) LR-PPO with a uniform noise kernel (K_T^u).
3) LR-PPO with a Gaussian noise kernel (K_T^g).
4) SA-PPO from [86].

In these experiments, we use PPO with a neural policies and value functions; the architectures and hyper-parameters used in each case can be found in Appendix B. The results are summarised in the left-hand side of Table 7.1. Each entry is the median of 10 independent training processes, with reward values measured as the mean of 100 independent trajectories.

Robustness through Disadvantage Objectives. If we have an estimator for the critic Q^π we can obtain robustness without inducing regularity in the policy using D^π, yielding a larger policy subspace to steer towards, and hopefully achieving policies closer to optimal. With the goal of diving deeper into the results of Theorem 7.1, we consider the objective:

$$K_D(\theta) := \sum_{x \in X} \mu_{\pi_\theta}(x) \frac{1}{2} \| D^{\pi_\theta}(x, T) \|_2^2.$$

We aim to test the hypothesis introduced through this work: by setting $K_2 = K_D$ and thus aiming to minimise the disadvantage D, we may obtain policies that yield better robustness with similar expected rewards. Observe that $\pi_D \in \Pi_D \implies K_D(\pi_D) = 0$. To test this, we compare the following algorithms on the same environments:

1. Vanilla A2C (noiseless).
2. LR-A2C with K_T^u.
3. LR-A2C with K_T^g.
4. LR-A2C with K_D.

We use A2C in this case since the structure of the original cost functions are simpler than PPO, and hence easier to compare between the scenarios above, and we modified A2C to retain a Q function as a critic. With each objective function resulting in gradient descent steps that pull the policy towards different maximally robust sets ($K_T \to \Pi_T$ and $K_D \to \Pi_D$ respectively), we would expect to obtain increasing robustness for K_D. The results are presented in the right-hand side of Table 7.1.

	PPO on MiniGrid Environments				*A2C on MiniGrid Environments*			
Noise	Vanilla	LR_{PPO} (K_T^u)	LR_{PPO} (K_T^g)	SA-PPO	Vanilla	LR_{A2C} (K_T^u)	LR_{A2C} (K_T^g)	LR_{A2C} (K_D)
LavaGap								
$\emptyset$	**0.95±0.003**	**0.95±0.075**	**0.95±0.101**	0.94±0.068	**0.94±0.004**	**0.94±0.005**	**0.94±0.003**	**0.94±0.006**
T_1	0.80±0.041	**0.95±0.078**	0.93±0.124	0.88±0.064	0.83±0.061	**0.93±0.019**	0.89±0.032	0.91±0.088
T_2	0.92±0.015	**0.95±0.052**	**0.95±0.094**	0.93±0.050	0.89±0.029	**0.94±0.008**	0.93±0.011	0.93±0.021
LavaCrossing								
$\emptyset$	**0.95±0.023**	0.93±0.050	0.93±0.018	0.88±0.091	0.91±0.024	0.91±0.063	0.90±0.017	**0.92±0.034**
T_1	0.50±0.110	**0.92±0.053**	0.89±0.029	0.64±0.109	0.66±0.071	**0.78±0.111**	0.72±0.073	0.76±0.098
T_2	0.84±0.061	**0.92±0.050**	**0.92±0.021**	0.85±0.094	0.78±0.054	0.83±0.105	0.86±0.029	**0.87±0.063**
DynamicObstacles								
$\emptyset$	**0.91±0.002**	**0.91±0.008**	**0.91±0.007**	**0.91±0.131**	**0.91±0.011**	0.88±0.020	0.89±0.009	**0.91±0.013**
T_1	0.23±0.201	**0.77±0.102**	0.61±0.119	0.45±0.188	0.27±0.104	0.43±0.108	0.45±0.162	**0.56±0.270**
T_2	0.50±0.117	**0.75±0.075**	0.70±0.072	0.68±0.490	0.45±0.086	0.53±0.109	0.52±0.161	**0.67±0.203**

Table 7.1: Reward values gained by LRPG and baselines.

7.7 Discussion

Experiments We applied LRPG on PPO and A2C algorithms, for a set of discrete action, discrete state grid environments. These environments are particularly sensitive to robustness problems; the rewards are sparse, and applying a sub-optimal action at any step of the trajectory often leads to terminal states with zero (or negative) reward. LRPG successfully induces lower robustness regrets in the tested scenarios, and the use of K_D as an objective (even though we did not prove the convergence of a gradient based method with such objective) yields a better compromise between robustness and rewards. When compared to recent observational robustness methods, LRPG obtains similar robustness results while preserving the original guarantees of the chosen algorithm (it even outperforms in some cases, although this is probably highly problem dependent, so we do not claim an improvement for every DOMDP).

Further Considerations on LRPG The advantages of LRPG with respect to, for example, using POMDP results to construct optimal policies for the disturbed state case (taking the observation map to be the noise kernel) lie mainly in the situations where we are not able to construct approximations of such noise kernels. For example, if the system presents some kind of non-stationarity (in terms of noise), it may be un-feasible to try to estimate the observation map. Through LRPG, however, we can always steer the policy towards a set that is *always* maximally robust, and this may lead to improvements even in the non-stationary setting. Exploiting this consequence can be an interesting research direction.

Robustness, Complexity and Invariances Sections 7.2 and 7.3 discuss at large the structure, shape and dependence of the maximally robust policy sets. These insights help derive optimisation objectives to use in LRPG, but there is more to be said about how policy robustness is affected by the underlying MDP properties. We hint at this in the proof of Corollary 7.1. More regular (*less complex* in entropy terms, or more *symmetric*) reward functions (*e.g.*, reward functions with smaller variance across the actions $R(x, \cdot, y)$) seem to induce larger robust policy sets. In other words, for a fixed policy, a *more complex* re-

ward function yields larger robustness regrets as soon as any noise is introduced in the system. This raises questions on how to use these principles to derive more robust policies in a comprehensive way, but we leave these questions for future work. Additionally, one could quickly extend these ideas and use LRL to induce other kinds of invariances to policies. For example, one could use LRL to obtain policies that generalise to a subclass of reward functions (connecting to previous thoughts on complexity). One advantage of this approach would be that the resulting policy gradient algorithm would retain the original algorithm guarantees.

8

Conclusion

The constrains introduced in model-free multi-agent systems by considering communication practicalities span across many different problems. Through this dissertation, we have started by looking into how ant-inspired multi-robot systems are affected by these constraints, and evolved towards more general state-observation robustness problems that emerge in imperfect information Reinforcement Learning scenarios. We now summarise some observations and statements that can be distilled from this work, and we point towards future possibilities and problems that follow from these observations.

8.1 Key Observations

Interpretation of Ant-Inspired Swarm Results In Chapter 3 we analyse how, by interpreting the class of ant-inspired swarms considered as a particular biased random walk one can extract conclusions about the convergence of both the pheromone fields and the probability distributions of the agents. First, randomness in the decision making is necessary for convergence of distributions, which resonates with other convergence results for RL algorithms on MDPs: persistent exploration is required in swarm systems. Additionally, it is not possible to a priori determine the stationary distribution of the agents. This will depend on the sequence of stochastic matrices that dictate the evolution of the probabilities, and this sequence is itself a random process determined by the agent movements. Interestingly, this sequence is itself a Martingale when the swarm walks on a directed graph (which is the scenario that better represents a connection with a value-based RL system).

Mean Field Models for Verification of Systems One important take from Chapter 4 is the utility of studying mean field processes in large multi-agent model-free systems. Through mild assumptions in the reinforcement scheme, one can obtain deterministic discrete time difference equations that yield the stationary agent distribution in (almost) closed form for fixed initial conditions. This allows us to, first, study which parameters have an influence in the stationary behaviour of the system and which do not, and second, verify properties of such distributions and establish bounds on expected behaviours when having a finite amount of agents.

Learning of Robustness Certificates Chapter 6 is built on the following premise: If we gather data to learn policies and value functions, perhaps we can use it to know more about the system. In our case, we use it to learn certificate functions that measure how much delay the MARL can tolerate in agent-to-agent communication depending on the state-space region. Interestingly, the amount of computation required for this is much smaller in comparison with the complexity of RL algorithms, and it provides a substantially powerful tool when rolling out policies (or controllers) in a multi-agent MDP. Additionally, since the motivation for this is reducing communication *safely*, through the use of the scenario approach we make sure that this is done without breaking possible inherited optimality guarantees.

Verifiable Reinforcement Learning versus Regularisation In Chapter 7 we propose an alternative interpretation to robustness in RL. Most of the robust RL literature (specifically for observational robustness) hinges on the idea that, in model free RL, the best we can do is to propose open-loop robustness: Introducing carefully designed regularisation terms in the learning of the policies *flattens* the policy approximator in sensitive regions of the input space, therefore obtaining (almost as a collateral effect) robustness versus disturbances. We suggest a different logic: instead of modifying thoroughly crafted RL algorithms in un-predictable ways in hopes of obtaining robustness, one can use LRPG to *carefully* guide the evolution of the policies towards more robust regions, while making sure the policy will still work reasonably well in the original problem. This is done by interpreting robustness as an invariance towards disturbance operators, which although seemingly straight-forward, was not proposed in existing robust RL work.

8.2 Shortcomings and Improvements

An interpretation of Ant Colony Algorithms (or in particular, ant-inspired multi-robot systems) that is partly exploited in this dissertation is the following: *Ant-inspired systems are a subclass of MARL systems.* In particular, the pheromone fields are value functions, and the decision making algorithms combined with the directional sensing of the pheromones constitute a value-based policy (*e.g.* Q value ϵ-greedy). Even Ant Colony Optimization can be thought of as a form of reinforcement-based stochastic optimization, and although developed almost independently of seminal RL work, one should ask the question of whether these methods have been over-powered by more general (MA)RL algorithms. Not only this, but given that (as presented in this dissertation) the most interesting theoretical results on ant-inspired multi-robot systems come through the explicit use of RL related concepts (reward signals, exploration rates, transition probabilities...), a critical question to be asked is if we are not simply solving a particular form of a general problem that has been addressed extensively in other forms in the literature (*re-inventing the wheel*).

A motivation for synthesising closed form solutions of the mean field swarm systems is to be able to use the reward function as a *control input.* Ideally, one could take the results of Chapter 4 to design reward functions that achieve desired agent distributions, and one could do it perhaps through a feedback control scheme. To achieve this, however, one needs to improve on the results in Chapter 4 to devise an iterative approach that allows for *reward shaping in foraging swarms*: the closed form solutions of the agent distributions

are a non-linear function of the weights, which may not be straight-forward to turn into a solvable feedback iteration.

In Chapter 6 the proposed method proves to be very useful at safely reducing communication in the studied cases, but they yield very conservative results both in terms of the theoretical bounds and the experimental sensitivity to variations in ε. A large source of conservativeness is the computation of the robustness surrogate functions. First, the functions are computed using the sup-norm on the state-space, and bounding the norm by the first state that violates the robustness condition. This means that, for large dimensional state-spaces, the condition violation can happen for only one state in an n-dimensional box, and this is enough to restrict the robustness surrogate. This induces *significant conservativeness* in the systems. It would be much more useful to compute these bounds based on the MDP trajectories: if a state y is never visited from x, even though it may be close in norm, it should not play a role in the robustness of x. Alternatively, one could lift the state space to a higher dimensional space where using the sup-norm produces tighter bounds.

At last, the main shortcoming of the LRPG (meta) algorithm proposed in Chapter 7 is the fact that *one needs to choose a noise map* $\tilde{T}$ to induce robustness properties in the policy gradient algorithms. This raises questions on, first, what noise generator to pick. We argue that a uniform distribution may be a sensible choice if no information is available, but one could similarly argue that an adversarial disturbance would be more effective at producing robust policies. Second, it is not clear how this choice would affect the obtained robustness regrets, and how this can be problem dependent. One may want to find alternative formulations to avoid this choice completely without breaking LRPG assumptions.

8.3 Future Research Directions

Event-Based Cooperation In Chapter 6 we propose schemes for *event-based information sharing* in model free multi-agent systems. However, we consider fully cooperative agents. A very interesting research direction would be to investigate the case of mixed cooperative-competitive agents. In such systems, one could try to learn action classifiers that would tell agents if a given action is more *selfish* or more *altruistic*. In that case, one must choose when to execute altruistic actions, and when to execute selfish actions. This could be framed from an event-based perspective, where this choice is triggered by some bounded estimation on the cooperative-competitive objectives.

Learning more than Policies A prevalent idea through Chapter 6 is that one should perhaps take advantage of the data collection requirements of model-free RL algorithms to learn something else than a value function and a policy. In our case, we learn robustness certificates that help with reducing communication in MARL. One could devise similar problems where we would want to learn other types of certificates or indicators. For example, one could learn *uncertainty indicators*, that would give agents an estimation on whether the current state is specially sensitive to any form of uncertainty in the system. Then agents could use this to trade-off safety for rewards

Putting a Price on Uncertainty Consider a game where, when playing it live, getting full state information is increasingly costly. That is, if the game is represented by some

form of (multi-agent) MDP, one can always access the full state x at a high cost (through a cost model not necessarily known), but one can instead use increasingly uncertain states at a decreasing cost. That is, reducing uncertainty is a choice that the agent has access to, but in exchange for some resource. This may be the case when e.g. measurements are acquired through costly mechanisms (communication) that one may not desire to use at all time steps. For diverse reasons, e.g. establishing a separation of concerns between (control) strategy design and uncertainty reduction cost (communication requirements), it may not be desirable to include the selection of a specific into the learning of the strategy. This sort of problem is connected to the motivation behind Chapter 7, and can open very interesting research directions on how to reduce uncertainty when necessary, and how would this affect the dynamics of the game.

Model-Based Robustness in Reinforcement Learning A natural conclusion from Chapter 7 is that model free approaches can only go so far when trying to assert robustness in learned policies. If the agent does not have any mechanism to estimate how much noise is being introduced in the systems, or whether there are disturbances at all, acting safely under uncertainty becomes an (almost) educated guess. Learning specific models to try to estimate how much uncertainty there is in the system would produce much more reasonable results when trying to solve robust RL problems. This could be done through model based RL techniques, but also through other less restrictive approaches. For example, one could try to learn uncertainty estimators based on sequences of states; a single state is not enough to estimate uncertainty, but sequences of states are.

A

Technical Proofs

Chapter 3

Corollary 3.1. We can show this by contradiction. The results presented in Theorem 2.5 are formulated for row-stochastic matrices. Let $\mathscr{M}$ be a class of row stochastic matrices which sequences satisfy Assumption 2.1. Consider the sequence of column stochastic matrices $\{B_k\}$, with any sequence formed by transposed elements $\{B_i^\top\} \in \mathscr{M}$. Let $\mathscr{M}_B$ be the set of all possible matrices B_i, and $\mathscr{M}_{B^\top}$ the set of all $B_i^\top$, such that $B_i \in \mathscr{M}_B$ and $B_i^\top \in \mathscr{M}_{B^\top}$ for any i. Consider the left product of the original sequence. Observe that we can take the transposed of the product:

$$\left[\prod_{t=0}^{t=k} B_t\right]^\top = \prod_{t=0}^{t=k} A_t, \tag{A.1}$$

where $A_t \in \mathscr{M}_{B^\top}$ for all t. If the limit as $k \longrightarrow \infty$ of (A.1) does not exist, there exists a sequence $\{A_t\}$ for which its product does not converge. But by definition, the sequence $\{A_t\}$ satisfies Assumption 2.1 since $A_i \in \mathscr{M}_{B^\top}$, and any sequence $\{B_i^\top\} \in \mathscr{M}$. Therefore, the limit in (A.1) must satisfy (*a.s.*):

$$\lim_{k\to\infty} \prod_{t=0}^{t=k} B_t = \lim_{k\to\infty} \left[\prod_{t=0}^{t=k} A_t\right]^\top = \left(\mathbf{1}\xi^\top\right)^\top,$$

where $\xi \in \Delta(n)$ and all its entries sum to 1. □

Proposition 3.1. If there are no odd length cycles in G, then we can split the graph in odd and even vertices. Starting from an odd vertex it is only possible to reach any other odd vertex in even times, and the converse. Let there now be one odd cycle $\mathscr{C}$. Let i be a starting node and j any other vertex, with the shortest $i-j$ path being of even length l_{ij}. Then, $\Pr\{x_t^a(j) \mid x_t^a(i)\} > 0$ if $t > 2k + l_{ij}\ \forall k \in \mathbb{N}_0^+$. The only way of reaching j in odd time is by completing then the odd length cycle. Let l_{iv} be the minimum path length between i and any vertex $v \in \mathscr{C}$, and let $l_v j$ be the minimum $v-j$ path length. Then, $\Pr\{x_t^a(j) \mid x_t^a(i)\} > 0$

if $t > 2k + l_{iv} + l_{vj} + l_c \; \forall k \in \mathbb{N}_0^+$. Since $\mathscr{C}$ is the only odd length cycle, $t_{\text{odd}} = 2k + l_{iv} + l_{vj} + l_c$ is an odd number. And particularly,

$$t_{\text{odd}} \leq 2\,\text{diam}(G) + l_c.$$

□

Proposition 3.2. First of all, $W_t(i,j) \in [\varepsilon, 1]$ for all edges $\{ij\}$ satisfying $W_0(i,j) \neq 0$. Let $\deg^* \coloneqq \max\{d_i : i \in X\}$. Then,

$$\alpha = \frac{\varepsilon}{1 + (\deg^* - 1)\varepsilon} \Longrightarrow P_t(j,i) > \alpha \quad \forall P_t(j,i) > 0, \tag{A.2}$$

which satisfies the condition (2) of Assumption 2.1. For condition (1) in Assumption 2.1, see that the associated digraph to P_t is a connected planar graph. From Proposition 3.1, the matrix product

$$\Big[\prod_{k=0}^{2\,\text{diam}+1} P_{t_0+k} \Big]^\top$$

has all entries > 0 for any pair k, l and any t_0. This follows from connected graphs properties. From (A.2) we make sure that the graph can never become disconnected, therefore P_t is irreducible for all t. Furthermore, since no edges are being deleted for any t, the probability

$$\Pr\Big\{ \Big[\prod_{k=0}^{2\,\text{diam}+1} P_{t_0+k} \Big]^\top \in \mathscr{M}_2 \Big\} = 1 \quad \forall t_0 > 0.$$

Hence, $P_t^\top$ satisfies Assumption 2.1. □

Theorem 3.1. Let P_t be constructed with $\varepsilon > 0$ being a minimum weight at choice. From Proposition 3.2, we know that the sequence $\{P_t^\top\}$ satisfies Assumption 1, and recalling Corollary 3.1, the left product

$$\lim_{k \longrightarrow \infty} \prod_{t=0}^{k} P_t = \xi \mathbf{1}^\top.$$

Then, the agent distribution as $t \longrightarrow \infty$ is

$$\lim_{t \longrightarrow \infty} y_t = \xi \mathbf{1}^\top y_0 = \xi,$$

since $\xi \mathbf{1}^\top$ is a matrix of identical columns ξ and the vector y_0 sums 1 over all its entries. The agent probability distribution converges *a.s.* to the vector ξ regardless of y_0. □

Proof: Corollary 3.2. The proof follows identical steps to Theorem 3.1. Now we have $|N|$ different sequences $\{P_t^a\}$, depending on the movement of each agent. However, each sequence satisfies Assumption 2.1 (it can be easily checked by the logic in Proposition 3.2). Therefore, each agent converges to a distribution $y_t^a \xrightarrow{\text{a.s.}} y_\infty^a$ as $t \longrightarrow \infty$. □

Proof: Corollary 3.3. Proof is analogous to Theorem 3.1. In fact, to have $y_t \xrightarrow{\text{a.s.}} y_\infty$ as $t \longrightarrow \infty$ we do not need to impose γ to be independent from M_t, but this requirement does affect a second corollary in the next section. □

Proof: Proposition 3.3. First see that if W_t is known, so is the transition probability matrix P_t. Now recall that $P_t(j,i)$ determines the probability of any agent moving from vertex i to vertex j at time t. Therefore, for any agent $a \in N$,

$$P\{x_{t+1}(a) = j \mid x_t(a) = i\} = \begin{cases} 1, & i \in \{x^g, x^0\}, \\ \frac{W_t(i,j)}{g_t(i)} & \text{else.} \end{cases}$$

The weights in the graph are only updated after all agents have moved. Then, the choice of one agent at time t does not affect the choices of other agents at t. Denote $N_i = \{a \in N : x_t(a) = i\}$ and observe that $n_t(i) \equiv |N_i|$. Then,

$$\mathbb{E}[M_{t+1}(i,j) \mid \mathscr{F}_t] = \sum_{a \in N_i} P_t(j,i) = P_t(j,i) n_i(t).$$

□

Proposition 3.4. First, it is trivial from (3.2) that for any $i \in \{x^g, x^0\}$:

$$\partial P_t(j,i) = 0 \Longrightarrow \mathbb{E}[\partial P_t(j,i) \mid \mathscr{F}_t] = 0 \quad \forall t > 0.$$

Consider now the rest of the edges ($i \notin \{x^g, x^0\}$). From (3.2) and substituting the weight dynamics in Definition 3.3:

$$P_{t+1}(j,i) = \frac{W_{t+1}(i,j)}{g_{t+1}(i)} = \frac{(1-\alpha) W_t(i,j) + \frac{\alpha}{n} M_{t+1}(i,j)}{g_{t+1}(i)}. \tag{A.3}$$

Recall that

$$g_{t+1}(i) = (1-\alpha) g_t(i) + \frac{\alpha}{n} n_t(i). \tag{A.4}$$

Now we can compute the probability increment $\partial P_t(j,i) = P_{t+1}(j,i) - P_t(j,i)$ from (A.3) as

$$\partial P_t(j,i) = \frac{\left((1-\alpha) W_t(i,j) + M_{t+1}(i,j) \frac{\alpha}{n}\right) g_t(i) - W_t(i,j) g_{t+1}(i)}{g_t(i) g_{t+1}(i)} \tag{A.5}$$

and substituting (A.4) in the numerator in (A.5),

$$\partial P_t(j,i) = \frac{\frac{\alpha}{n} \left(M_{t+1}(i,j) g_t(i) - W_t(i,j) n_t(i)\right)}{g_t(i) g_{t+1}(i)}. \tag{A.6}$$

Observe that, by using the result in Proposition 3.3

$$\frac{W_t(i,j) n_t(i)}{g_t(i)} = P_t(j,i) n_t(i) = \mathbb{E}[M_{t+1}(i,j) \mid \mathscr{F}_t]. \tag{A.7}$$

A

Finally, substituting (A.7) in (A.6):

$$\partial P_t(j,i) = \frac{\alpha}{n}\frac{M_{t+1}(i,j) - \mathbb{E}[M_{t+1}(i,j) \mid \mathscr{F}_t]}{g_{t+1}(i)}. \tag{A.8}$$

Let us now take the conditional expected value of (A.8). The denominator is fully determined by $\mathscr{F}_t$. Furthermore,

$$\begin{aligned}&\mathbb{E}[\mathbb{E}[M_{t+1}(i,j) \mid \mathscr{F}_t] \mid \mathscr{F}_t] = \mathbb{E}[M_{t+1}(i,j) \mid \mathscr{F}_t] \Longrightarrow \\ \Longrightarrow\ &\mathbb{E}[\partial P_t(j,i) \mid \mathscr{F}_t] = \alpha\frac{\mathbb{E}[M_{t+1}(i,j) - \mathbb{E}[M_{t+1}(i,j) \mid \mathscr{F}_t] \mid \mathscr{F}_t]}{n g_{t+1}(i)} = 0.\end{aligned}$$

□

Theorem 3.2. Take the probability transition matrix increment $\partial P_t(j,i)$. See that it is a random variable that takes values $\partial P_t(j,i) \in [-1,1]$ (therefore, $\sup_t \mathbb{E}[\partial P_t(j,i)^+] < \infty$). Now, from Proposition 3.4

$$\mathbb{E}[\partial P_t(j,i) \mid \mathscr{F}_t] = 0 \Longrightarrow \mathbb{E}[P_{t+1}(j,i) - P_t(j,i) \mid \mathscr{F}_t] = 0.$$

See that $P_t(j,i)$ is fully determined by the the information in σ-algebra $\mathscr{F}_t$. Then,

$$\begin{aligned}&E[P_{t+1}(j,i) - P_t(j,i) \mid \mathscr{F}_t] = \mathbb{E}[P_{t+1}(j,i) \mid \mathscr{F}_t] - P_t(j,i) = 0 \iff \\ \iff\ &\mathbb{E}[P_{t+1}(j,i) \mid \mathscr{F}_t] = P_t(j,i).\end{aligned} \tag{A.9}$$

From Definition 2.4 it is clear that the entries of the probability transition matrix are all Martingales, and by Theorem 2.2 the matrix will converge to a P_∞ *a.s.* □

Corollary 3.4. Take eq. (A.6). If $\chi \in \{0,1\}$ is a random variable determining if weight is being added or not, we can write

$$\partial P_t(j,i) = \frac{\frac{\alpha}{n}\left(\sum_{k=1}^{M_{t+1}(i,j)} \gamma_k - P_t(j,i)\sum_{k=1}^{n_t(i)} \gamma_k\right)}{g_{t+1}(i)}, \tag{A.10}$$

with $g_{t+1}(i) = (1-\alpha)w_i + \frac{\alpha}{n}\sum_{k=1}^{n_t(i)} \gamma_k$. But if the variables M_t and χ are independent, $\mathbb{E}[XY \mid \mathscr{F}_t] = \mathbb{E}[X \mid \mathscr{F}_t]\mathbb{E}[Y \mid \mathscr{F}_t]$. Furthermore, let $Z = \sum_{k=1}^{M_{t+1}(i,j)} \chi_k$, and observe that by the law of total expectation

$$\begin{aligned}&\mathbb{E}[Z \mid \mathscr{F}_t] = E[E[Z \mid M_{t+1}(i,j)] \mid \mathscr{F}_t] = \\ &= \mathbb{E}\left[M_{t+1}(i,j)p_\chi \mid \mathscr{F}_t\right] = p_\chi \mathbb{E}[M_{t+1}(i,j) \mid \mathscr{F}_t].\end{aligned} \tag{A.11}$$

Then, taking the expected value of the numerator in (A.10):

$$\begin{aligned}&\mathbb{E}\left[\sum_{k=1}^{M_{t+1}(i,j)} \gamma_k - P_t(j,i)\sum_{k=1}^{n_t(i)} \chi_k \mid \mathscr{F}_t\right] = \\ &= p_\chi\left(\mathbb{E}\left[M_{ij}(t+1) \mid \mathscr{F}_t\right] - \mathbb{E}\left[M_{ij}(t+1) \mid \mathscr{F}_t\right]\right) = 0.\end{aligned} \tag{A.12}$$

Therefore, $P_t \xrightarrow{\text{a.s.}} P_\infty$ as $t \longrightarrow \infty$ regardless of χ. □

Chapter 4

Theorem 4.1. We show this by induction. Let us look first at $t = 0$. For a fixed set of initial conditions $\hat{y}_0^n$, w_0, observe that $\hat{y}_0^n = \hat{y}_0$, and we have $\forall a \in N$

$$\Pr\{\zeta_1^a(i) = 1 \mid \hat{y}_0^n, w_0\} = \hat{y}_1(i) = \left(P_0^\varepsilon \hat{y}_0^n\right)_i.$$

Observe that in this case, the initial conditions are fixed, therefore we can consider the total probability

$$\Pr[\{\zeta_1^a(i) = 1\}] = \left(P_0^\varepsilon \hat{y}_0^n\right)_i = \left(P_0^\varepsilon \hat{y}_0\right)_i. \tag{A.13}$$

Observe that (A.13) does not depend on a. Therefore, all agents have the same marginal probability distribution for $t = 1$. Additionally, the transition probabilities at $t = 0$ have not been affected by agent trajectories, therefore for the first time step $\zeta_1^a(i)$ are *i.i.d.* $\forall a$. We can then specify the joint distribution of having k agents in vertex i at time $t = 1$: this is the joint probability of events resulting in k agents moving to i, and $n - k$ agents moving elsewhere. Recall $\hat{y}_1^n = \frac{1}{n}\sum_{a=1}^n \zeta^a(1)$. Since ζ_i^a are indicator variables,

$$\mathbb{E}[\zeta_1^a] = P_0^\varepsilon \hat{y}_0 = \hat{y}_1.$$

Let us now consider the case where $n \longrightarrow \infty$, and define $y_1 := \lim_{n\to\infty} \hat{y}_1$. Since at $t = 0$ the initial conditions are fixed and all agents are initialised in the same vertex, it also holds that $\hat{y}_0 = y_0$. Additionally, P_0^ε is not affected by the limit $n \longrightarrow \infty$, and $\hat{y}_1 = P_0^\varepsilon \hat{y}_0 = P_0^\varepsilon y_0 = y_1$. Therefore, by Theorem 2.1 we have

$$\lim_{n\to\infty} \hat{y}_1^n = \mathbb{E}[\zeta_1^a] = y_1 \quad \text{a.s.} \tag{A.14}$$

That is, with probability 1, the agent proportion converges to the marginal probability distribution as $n \longrightarrow \infty$ for $t = 1$. From (A.14) it holds that any event $\mathscr{Y}_1 \in \mathscr{F}_1$ (*i.e.* any possible combination of agent positions until time $t = 1$) satisfies $\mathscr{Y}_1 \in \mathscr{F}_1 \Rightarrow \mathscr{Y}_1 = \{y_0, y_1\}$ *a.s.* That is, $\Pr\{\mathscr{Y}_1 \in \mathscr{F}_1 : q(1) = y_1\} = 1$ (the union of events has measure 1). Then, the update of P_1^ε depends on w_1, and in the limit $\lim_{n\to\infty} w_1 = f(\lim_{n\to\infty} \hat{y}_1^n, w_0) = f(y_0, w_0)$. Now for $t = 2$,

$$\mathbb{E}[\zeta_2^a] = \mathbb{E}[\mathbb{E}[\zeta_2^a \mid \mathscr{F}_1]] = \mathbb{E}[P_1^\varepsilon \mathbb{E}[\zeta_1^a \mid \mathscr{F}_1]] = P_1^\varepsilon \mathbb{E}[\zeta_1^a] = P_1^\varepsilon y_1 = y_2. \tag{A.15}$$

Therefore, with probability 1, the marginal probability distributions ζ_2^a are determined by y_1 (since they depend on $\mathscr{Y}_1$, and this occurs *a.s.*). Therefore, the variables are *i.i.d.* in the limit $n \longrightarrow \infty$, and by the law of large numbers,

$$\lim_{n\to\infty} \hat{y}_2^n = \mathbb{E}[\zeta_2^a] = y_2 \quad \text{a.s.}$$

By induction, it holds that there is only one possible sequence $\mathscr{Y}_t = \{y_0, y_1, ..., y_t\}$ where $\Pr\{\lim_{n\to\infty} \mathscr{Y}_t(n) = \mathscr{Y}_t\} = 1\ \forall t \geq 0$. Therefore $\mathbb{E}[\zeta_{t+1}^a] = P_t^\varepsilon y_t = y_{t+1}$, thus

$$\lim_{n\to\infty} \hat{y}_t^n = y_t \quad \text{a.s.} \quad \forall t \geq 0.$$

□

Proposition 4.1. Given the bounded probability matrix P_t^ε, for any edge $(ij) \in E$, we have $P_t^\varepsilon(j,i) \geq \varepsilon$. Furthermore, since by Assumption 4.1 there is at least one odd length cycle, the graph is aperiodic and we can directly invoke results from Chapter 3 on convergence of stigmergy swarm probability distributions. In particular, from Theorem 3.1

$$\exists y_\infty \,:\, \lim_{t\longrightarrow\infty}\left(\prod_{t_k=0}^{t} P_{t_k}^\varepsilon\right) y_0 = y_\infty.$$

Since all positive terms in matrices P_t^ε are lower bounded, the product matrix $P_\infty^\varepsilon := \lim_{t\longrightarrow\infty}\prod_{t_k=0}^{t} P_{t_k}^\varepsilon$ is irreducible, and from Theorem 2.4 the eigenvector y_∞ is unique and has strictly positive entries. Additionally, from Theorem 3.1 we know that the convergence is exponential, with a rate bounded by $\alpha = \left(1 - \frac{\varepsilon}{1+(g^*-1)\varepsilon}^{1+2\,\mathrm{diam}(G)}\right)^{\frac{1}{1+2\,\mathrm{diam}(G)}}$. □

Lemma 4.1. First, since all P_t^ε have the positive entries lower bounded by ε and the graph is connected, they are all irreducible and we can infer

$$\left(\prod_{t_k=t_0}^{t_0+2\,\mathrm{diam}(G)} P_{t_k}^\varepsilon\right)_{ji} = \left(P_{t_0+2\,\mathrm{diam}(G)}^\varepsilon \cdots P_{t_0}^\varepsilon\right)_{ji} \geq \varepsilon^{2\,\mathrm{diam}(G)} \;\forall i,j \in X. \tag{A.16}$$

In other words, all vertices are reachable from any other for times larger than $2\,\mathrm{diam}(G)$. Now, making use of (A.16), and $l_1^\top, l_2^\top, ..., l_{|X|}^\top$ being the rows:

$$\prod_{t_k=t_0}^{t_0+2\,\mathrm{diam}(G)} P_{t_k}^\varepsilon = \begin{pmatrix} l_1^\top \\ \cdots \\ l_{|X|}^\top \end{pmatrix} \Rightarrow \prod_{t_k=t_0}^{t_0+2\,\mathrm{diam}(G)} P_{t_k}^\varepsilon y_{t_0} = \begin{pmatrix} l_1^\top y_{t_0} \\ \cdots \\ l_{|X|}^\top y_{t_0} \end{pmatrix} \geq \varepsilon^{2\,\mathrm{diam}(G)}\mathbf{1}.$$

Therefore, for $t_0 = 0$ and $t > 2\,\mathrm{diam}(G)$ we have $y_t = \prod_{t_k=0}^{t} P_{t_k}^\varepsilon y_0 \geq \varepsilon^t \mathbf{1}$. Last, from Proposition 4.1, $\lim_{t\longrightarrow\infty} y_t = y_\infty > \mathbf{0}$, therefore

$$t > 2\,\mathrm{diam}(G) \Rightarrow y_t > \mathbf{0} \iff \mathrm{sgn}(y_t) = \mathbf{1}.$$

□

Proposition 4.2. Let $B \in \{0,1\}^{|X|\times|X|}$ be the selector matrix satisfying

$$Bw_\infty = \mathrm{diag}_i\left(\max_j W_\infty(i,j)\right)\mathbf{1}$$

. Since B is a row stochastic matrix by Theorem 2.4 it has all its eigenvalues in the unit disc, and $(I - \gamma B)$ has all its eigenvalues in a disc of radius γ centred at 1. Therefore, its inverse is properly defined and $w_\infty = (I - \gamma B)^{-1}(I + \Sigma_r)$ has a unique solution if $\gamma \in [0,1)$. □

Theorem 4.2. Recall the weight dynamics:

$$w_{t+1} = (1-\alpha)w_t + \alpha\left(I + \Sigma_r + \gamma\,\mathrm{diag}_i\left(\max_j W_t(i,j)\right)\right)\mathrm{sgn}(y_t). \tag{A.17}$$

Let us use for brevity $\phi(W) := \mathrm{diag}_i\big(\max_j W(i,j)\big)$, assume $w_\infty = (I + \Sigma_r + \gamma\phi(W_\infty))\mathbf{1}$ and define $z_t := w_t - w_\infty$. Subtract w_∞ at each side of (A.17):

$$z_{t+1} = (1-\alpha)z_t + \alpha\big(R_t\,\mathrm{sgn}(y_t) - w_\infty\big). \tag{A.18}$$

Define $e_t := \mathrm{sgn}(y_t) - \mathbf{1}$ to obtain:

$$\begin{aligned}&\big(I + \Sigma_r + \gamma\phi(W_t)\big)\,\mathrm{sgn}(y_t) - w_\infty = \\ =&(I + \Sigma_r + \gamma\phi(W_t))e_t + \gamma\big(\phi(W_t) - \phi(W_\infty)\big)\,\mathbf{1}.\end{aligned}$$

Taking the ∞-norm at each side of (A.18):

$$\begin{aligned}\|z_{t+1}\|_\infty &= \|(1-\alpha)z_t + \alpha(R_t e_t + \gamma\big(\phi(W_t) - \phi(W_\infty)\big)\,\mathbf{1}\|_\infty \le \\ &\le (1-\alpha)\|z_t\|_\infty + \alpha\|R_t\|_\infty\|e_t\|_\infty + \alpha\gamma\|\phi(W_t) - \phi(W_\infty)\|_\infty\|\mathbf{1}\|_\infty.\end{aligned} \tag{A.19}$$

Recall the induced ∞-norm of a matrix is its maximum absolute row sum. Then,

$$\begin{aligned}&\|\phi(W_t) - \phi(W_\infty)\|_\infty\|\mathbf{1}\|_\infty = \max_i|\max_j W_t(i,j) - \max_j W_\infty(i,j)| \le \\ &\le \max_i|\max_j|W_t(i,j) - W_\infty(i,j)|| = \max_i|z_t(i)| = \|z_t\|_\infty.\end{aligned} \tag{A.20}$$

Now from Lemma 4.1, $\|e_t\|_\infty = 0\ \forall t > 2\,\mathrm{diam}(G)$, therefore substituting (A.20) in (A.19):

$$\begin{aligned}&\|z_{t+1}(i)\|_\infty \le (1-\alpha)\|z_t\|_\infty + \alpha\gamma\|z_t\|_\infty = \\ &= (1-\alpha(1-\gamma))\|z_t\|_\infty \le (1-\alpha(1-\gamma))^2\|z_{t-1}\|_\infty \le \\ &\le (1-\alpha(1-\gamma))^{t-2\,\mathrm{diam}(G)}\|z(2\,\mathrm{diam}(G))\|_\infty \Longrightarrow \lim_{t\to\infty}\|z_t(i)\|_\infty = 0.\end{aligned} \tag{A.21}$$

Finally, $\lim_{t\to\infty}\|z_t\|_\infty = 0 \Longrightarrow \lim_{t\to\infty} w_t = w_\infty$, and the proof is complete. □

Corollary 4.1. From Theorem 3.1 we know that the limit $\lim_{t\to\infty} y_{t+1} = \lim_{t\to\infty} y_t = y_\infty$ exists. Additionally, from Theorem 4.2 we know that the limit $\lim_{t\to\infty} P_t^\varepsilon = P_\infty^\varepsilon$ also exists. Therefore, using the limit product rule:

$$\lim_{t\to\infty} y_{t+1} = \lim_{t\to\infty} P_t^\varepsilon y_t = P_\infty^\varepsilon y_\infty = y_\infty.$$

□

Proposition 4.3. From Theorem 4.2, the fixed point is

$$w_\infty = (I + \Sigma_r + \gamma\,\mathrm{diag}_i\big(\max_j W_\infty(i,j)\big))\mathbf{1},$$

and recall from Proposition 4.2 that it is unique. Additionally, $\Sigma_r(i,i) = r$ for $i \in \{x^0, x^g\}$ and is 0 for all other vertices, and it can be shown by contradiction (not added here for brevity) that $\mathrm{argmax}_i(w_\infty(i)) = x^0, x^g$. Now, to prove the proposition we assume the following structure for w_∞, and later show it is indeed a solution (and therefore the only one, since it is unique). Let us assume for w_∞:

$$v, u \in X^1 \,:\, d(x^0, v) > d(x^0, u) \Longrightarrow w_\infty(v) < w_\infty(u), \tag{A.22}$$

A

and the same holds for the converse $v, u \in X^2$ with the distance to x^g. That is, if v is one step further away from x^0 than u, then it has a smaller weight value. Now recall

$$w_\infty(i) = (1 + \Sigma_r(i,i) + \gamma \max_{j \in X} W_\infty(i,j)), \tag{A.23}$$

and $\Sigma_r(i,i) = 0 \; \forall \; i \neq x^0, x^g$. Then, $\forall j \in X \; : \; d(x^0, j) = 1$:

$$\begin{aligned} w_\infty(j) =& (1 + \gamma \max_{k \in X} W_\infty(j,k)) = (1 + \gamma w_\infty(x^0)), \\ w_\infty(x^0) =& (1 + r + \gamma \max_{k \in X} w_\infty(i,k) = (1 + r + \gamma w_\infty(j)). \end{aligned} \tag{A.24}$$

Solving (A.24) for both weights we obtain

$$w_\infty(x^0) = \frac{1 + r + \gamma}{1 - \gamma^2}, \; w_\infty(j) = \frac{1 + \gamma(1 + r)}{1 - \gamma^2}. \tag{A.25}$$

Therefore, $r > 0 \Longrightarrow w_\infty(x^0) > w_\infty(j) \forall j \; : \; d(i,j) = 1$. Then, for any $k \in X^1$, $k \neq x^0$,

$$\begin{aligned} & w_\infty(k) = 1 + \gamma \max_{l \in X} W_\infty(k,l) = 1 + \gamma + \gamma^2 \max_{m \in X} W_\infty(l,m) \\ =& \ldots = \sum_{a=1}^{d(x^0,k)} \gamma^{a-1} + \gamma^{d(x^0,k)} w_\infty(i) = \\ =& \sum_{a=1}^{d(x^0,k)} \gamma^{a-1} + \frac{\gamma^{d(x^0,k)}\left(1 + r + \gamma\right)}{1 - \gamma^2} = \frac{1 + \gamma + \gamma^{d(x^0,k)} r}{1 - \gamma^2}, \end{aligned} \tag{A.26}$$

and the same holds for any $k \in X^2$ with the distance $d(x^g, k)$. Observe (A.26) yields an explicit solution to the fixed point w_∞ that satisfies the assumption in (A.22). From Proposition 4.2, this is the only solution, thus (A.22) indeed holds for the fixed point and graphs considered. Finally, by construction (A.26) guarantees that picking the neighbouring maximum weight w_∞ from any $v \in X$ leads to x^0 (or x^g) through the minimum distance path, i.e. $w_\infty \in \mathscr{W}^*$. □

Proposition 4.4. From Definition 4.2, if $\overline{y}$ is the eigenvector of P^0_∞ corresponding to the eigenvalue 1,

$$P^0_\infty \overline{y} = \overline{y} \Longleftrightarrow \begin{cases} (I - T) P^\nabla_{w^2_\infty} \overline{y}^1 + S P^\nabla_{w^1_\infty} \overline{y}^2 = \overline{y}^1 \\ T P^\nabla_{w^2_\infty}) \overline{y}^1 + (I - S) P^\nabla_{w^1_\infty} \overline{y}^2 = \overline{y}^2. \end{cases} \tag{A.27}$$

Recall Remark 4.5. Since we are considering the full doubled graph with $|X| = 2|X_1| = 2|X_2|$ (that is, with all ${x^g}_1, {x^0}_1, {x^g}_2, {x^0}_2 \in X$), there are two vertices in the graph that are effectively disconnected from the rest, namely ${x^g}_1$ and ${x^0}_2$. Therefore, $y_t(x^g_1) = y_t(x^0_2) = 0 \quad \forall t$. Similarly,

$${x^g}_1, {x^0}_2 \notin \left(\cup \mathscr{P}(x^0, x^g)\right) \cup \left(\cup \mathscr{P}(x^g, x^0)\right) \Longrightarrow \overline{y}_{{x^g}_1} = \overline{y}_{{x^0}_2} = 0. \tag{A.28}$$

Let us focus on the first equality in (A.27). Recall $\overline{y}^1_{x^0} = \overline{y}^2_{x^g} = \frac{1}{2k}$ and $\overline{y}^1_{x^g} = \overline{y}^2_{x^0} = 0$. Let us now verify that $P^{\nabla}_{w^1_\infty}\overline{y}^2 = \overline{y}^2$ for all vertices $v \neq x^0, x^g$. Recall from Definition 4.1 that $P^{\nabla}_{w^1_\infty}(j,i) = \frac{1}{|X_i^{out}|}\ \forall\, j \in X_i^{out}$. Then, for any $v \neq x^0, x^g$,

$$\left(P^{\nabla}_{w^1_\infty}\overline{y}^2\right)_v = \sum_{j\in X_v^{in}} \frac{\overline{y}^2_j}{m_j^{out}}. \tag{A.29}$$

Substituting now $\overline{y}^2_j = \frac{1}{2k}\sum_{p\in\mathscr{P}(x^g,j)}\prod_{u\in p\setminus j}\frac{1}{m_u^{out}}$ in (A.29):

$$\begin{aligned}\sum_{j\in X_v^{in}} \frac{\overline{y}^2_j}{m_j^{out}} &= \sum_{j\in X_v^{in}} \frac{1}{2k}\left(\sum_{p\in\mathscr{P}(x^g,j)}\prod_{u\in p\setminus j}\frac{1}{m_u^{out}}\right)\frac{1}{m_j^{out}} = \\ &= \frac{1}{2k}\sum_{j\in X_v^{in}}\sum_{p\in\mathscr{P}(x^g,j)}\prod_{u\in p}\frac{1}{m_u^{out}}.\end{aligned} \tag{A.30}$$

Since all $j \in X_v^{in}$ lead to v, (A.30) is simply

$$\frac{1}{2k}\sum_{j\in X_v^{in}}\sum_{p\in\mathscr{P}(x^g,j)}\prod_{u\in p}\frac{1}{m_u^{out}} = \frac{1}{2k}\sum_{p\in\mathscr{P}(x^g,v)}\prod_{u\in p\setminus v}\frac{1}{m_u^{out}} = \overline{y}^2_v, \tag{A.31}$$

and $\left(P^{\nabla}_{w^1_\infty}\overline{y}^2\right)_v = \overline{y}^2_v$. Similarly,

$$\left(P^{\nabla}_{w^2_\infty}\overline{y}^1\right)_v = \overline{y}^1_v \quad \forall v \neq x^0, x^g, \tag{A.32}$$

and $\overline{y}^1_{x^g} = 0$. Now observe

$$\left(SP^{\nabla}_{w^1_\infty}\overline{y}^2\right)_i = \begin{cases}\left(P^{\nabla}_{w^1_\infty}\overline{y}^2\right)_i & \text{if } i = x^0,\\ 0 \quad \text{else.}\end{cases} \tag{A.33}$$

From Proposition 4.3, we know that $w^1_\infty(x^0) = \max_j w_\infty(j) \Longrightarrow P^{\nabla}_{w^1_\infty}(x^0,i) = 1\ \forall\,(ix^0) \in \mathscr{E}$. Since all paths $p \in \mathscr{P}(x^g, x^0)$ start and end at the same vertices and have the same length, recall $\frac{1}{m_u^{out}}$ can be interpreted as the probability of moving out of u, therefore the product $\Pr\{p\} := \prod_{u\in p\setminus x^0}\frac{1}{m_u^{out}}$ is the probability of following the entire path p, and it holds that

$$\sum_{p\in\mathscr{P}(x^g,x^0)}\prod_{u\in p\setminus x^0}\frac{1}{m_u^{out}} = \sum_{p\in\mathscr{P}(x^g,x^0)}\Pr\{p\} = 1, \tag{A.34}$$

Therefore, by making use of (A.34), we can compute (A.33):

$$\begin{aligned}\left(SP^{\nabla}_{w^1_\infty}\overline{y}^2\right)_{x^0} &= \sum_{j\in N^{in}_{x^0}} \frac{\overline{y}^2_j}{m^{out}_j} = \sum_{j\in N^{in}_{x^0}} \frac{1}{2k}\left(\sum_{p\in\mathscr{P}(x^g,j)}\prod_{u\in p\setminus j}\frac{1}{m^{out}_u}\right)\frac{1}{m^{out}_j} = \\ &= \frac{1}{2k}\sum_{j\in N^{in}_{x^0}}\sum_{p\in\mathscr{P}(x^g,j)}\prod_{u\in p}\frac{1}{m^{out}_u} = \frac{1}{2k}\sum_{p\in\mathscr{P}(x^g,x^0)}\prod_{u\in p\setminus x^0}\frac{1}{m^{out}_u} = \frac{1}{2k} = \overline{y}^1_{x^0}.\end{aligned} \tag{A.35}$$

Finally, combining (A.32) and (A.35) we have

$$(I-T)P^{\nabla}_{w^2_\infty}\overline{y}^1 + SP^{\nabla}_{w^1_\infty}\overline{y}^2 = \overline{y}^1, \tag{A.36}$$

and analogously one can show that the same holds for the second equation in (A.27). Therefore, $P^0_\infty\overline{y} = \overline{y}$. □

Theorem 4.3. Recall $P(\infty,\varepsilon) = (1-\varepsilon)P^0_\infty + \varepsilon P^1_\infty$. Additionally, from Corollary 4.1 and Proposition 4.4,

$$P(\infty,\varepsilon)y_\infty = y_\infty, \quad P^0_\infty\overline{y} = \overline{y}.$$

Now let $L := \left(I - P^0_\infty\right)$. Then, we can expand

$$\begin{aligned}y_\infty - \overline{y} &= P(\infty,\varepsilon)y_\infty - P^0_\infty\overline{y} = (1-\varepsilon)P^0_\infty y_\infty + \varepsilon(P^1_\infty y_\infty - P^0_\infty)\overline{y} = \\ &= P^0_\infty(y_\infty - \overline{y}) + \varepsilon(P^1_\infty - P^0_\infty)y_\infty \Longrightarrow L(y_\infty - \overline{y}) = \varepsilon(P^1_\infty - P^0_\infty)y_\infty.\end{aligned} \tag{A.37}$$

The null space of L is given by $Lb = 0 \iff P^0_\infty b = b$, and by Theorem 2.4 we know b is unique, therefore $\operatorname{rank}(L) = |X| - 1$. But to solve the system of equations $L(y_\infty - \overline{y}) = \varepsilon(P^1_\infty - P^0_\infty)y_\infty$, L needs to be invertible. For this we can add the following additional equation: We know it must hold that $\mathbf{1}^\top(y_\infty - \overline{y}) = 0$, and this equation is linearly independent from all rows in L if and only if $\nexists c \in \mathbb{R}^{|X|}$ that satisfies $Lb = \mathbf{1}$. Let us show that there does not exist such a b by contradiction. Assume $\exists b : Lb = \mathbf{1}$. Recall $\mathscr{P}(x^0,x^g), \mathscr{P}(x^g,x^0)$ are the sets of optimal paths between x^0, x^g and x^g, x^0, with $\mathscr{P}(x^0,x^g) \in \mathscr{P}(x^0,x^g)$. Then, $\forall i_1 \in X^{out}_{x^0}, L_{x^0x^0} = L_{i_1i_1} = 1, \quad L_{i_1x^0} = -\frac{1}{m^{out}_{x^0}}$. Adding the rows of L $\forall i_1$:

$$\left(\sum_{i_1\in X^{out}_{x^0}} L_{i_1}\right)_j = \begin{cases} 1 & \text{if } j \in X^{out}_{x^0}, \\ -1 & \text{if } j = x^0, \\ 0 & \text{else.}\end{cases} \tag{A.38}$$

Now let $\cup_{i_1} X^{out}_{i_1} := \{k : k \in X^{out}_{i_1} \forall\, i_1 \in X^{out}_{x^0}\}$ be the set of all vertices at distance 2 from x^0 when following optimal paths, and $i_2 \in \cup_{i_1} X^{out}_{i_1}$. Adding the rows of L $\forall i_2$:

$$\left(\sum_{i_2\in\cup_{i_1}X^{out}_{i_1}} L_{i_2}\right)_j = \begin{cases} 1 & \text{if } j \in \cup_{i_1} X^{out}_{i_1}, \\ -1 & \text{if } j \in X^{out}_{x^0}, \\ 0 & \text{else.}\end{cases} \tag{A.39}$$

Now it is clear that adding (A.38) and (A.39):

$$\left(\sum_{i\in X_{x^0}^{out}} L_{i_1} + \sum_{i_2\in\cup_{i_1} X_{i_1}^{out}} L_{i_2}\right)_j = \begin{cases} 1 & \text{if } j\in\cup_{i_1} X_{i_1}^{out}, \\ -1 & \text{if } j = x^0, \\ 0 & \text{else.} \end{cases} \tag{A.40}$$

Extending the sum until vertex x^g, we add rows $\forall i \in \cup\mathscr{P}(x^0, x^g)$:

$$\left(\sum_{i\in\cup\mathscr{P}(x^0,x^g)\setminus x^0} L_i\right)_j = \begin{cases} 1 & \text{if } j = x^g, \\ -1 & \text{if } k = x^0, \\ 0 & \text{else.} \end{cases} \tag{A.41}$$

Analogously, considering the reverse paths $\mathscr{P}(x^g, x^0)$ one obtains

$$\left(\sum_{i\in\cup\mathscr{P}(x^g,x^0)\setminus x^g} L_i\right)_j = \begin{cases} 1 & \text{if } j = x^0, \\ -1 & \text{if } j = x^g, \\ 0 & \text{else.} \end{cases} \tag{A.42}$$

Define $\overline{m} = |\cup\mathscr{P}(x^0, x^g)| = |\cup\mathscr{P}(x^g, x^0)|$ as the number of vertices in all optimal paths, and from (A.41) and (A.42) one obtains

$$\sum_{i\in\cup\mathscr{P}(x^0,x^g)\setminus x^0} L_i c = \overline{m} - 1, \quad \sum_{j\in\cup\mathscr{P}(x^g,x^0)\setminus x^g} L_j c = \overline{m} - 1 \Longrightarrow -\overline{m} = \overline{m},$$

which is a contradiction. Then, $\nexists b : Lb = \mathbf{1}$, and there is a row in L, $P_\infty^1 - P_\infty^0$ such that replacing it (assuming it is the last row, without loss of generality) we obtain

$$\tilde{L} := \begin{pmatrix} L_1 \\ \dots \\ \mathbf{1}^\top \end{pmatrix}, My_\infty := \begin{pmatrix} \left(P_\infty^1 - P_\infty^0\right)_1 y_\infty \\ \dots \\ \mathbf{0}^\top \end{pmatrix},$$

where $\text{rank}(\tilde{L}) = |X|$. Now, observe

$$\tilde{L}(y_\infty - \overline{y}) = \varepsilon My_\infty \Longrightarrow y_\infty - \overline{y} = \varepsilon\tilde{L}^{-1}My_\infty.$$

Finally, since $\|\tilde{L}^{-1}\|_1$ is bounded and does not depend on ε and $\|My_\infty\|_1 \le 2$, $\exists c \in \mathbb{R}_{\ge 0}$ and $f(\varepsilon) \in \mathscr{K}_\infty$ such that

$$\|y_\infty - \overline{y}\|_1 \le \varepsilon\|\tilde{L}^{-1}My_\infty\|_1 \le \varepsilon c =: f(\varepsilon). \tag{A.43}$$

□

Chapter 5

Proposition 5.1. Consider first a deterministic MDP. In this case, $P_{xy}(u) \in \{0, 1\}$, and

$$Q^*(x,u) = R(x,u) + \gamma\mathbb{E}_{y\sim P(x,u,\cdot)}\left[\max_v Q^*(y,v)\right] = R(x,u) + \gamma\max_v Q^*(x',v)$$

A

where x' satisfies $P(x,u,x')=1$. Then,

$$Q_t \to Q^* \ a.s. \iff Q_t(x,u) - Q^*(x,u) \to 0 \ a.s. \ \forall x \in X, u \in U.$$

Recalling the Q-learning iteration, $\forall x \in X, u \in U$ it holds *a.s.*:

$$\begin{aligned}
&\lim_{t\to\infty} Q_t(x,u) = Q^*(x,u) \iff \lim_{t\to\infty} Q_{t+1}(x,u) - Q_t(x,u) = \\
&= 0 \iff \lim_{t\to\infty} Q_t(x,u) + \alpha_t \frac{1}{n_x}\Big(\sum_{i\in n_x} R(x,u) + \gamma \max_v Q^*(x',v) - Q_t(x,u)\Big) - Q_t(x,u) = 0 \iff \\
&\iff \lim_{t\to\infty} \frac{1}{n_x}\Big(\sum_{i\in n_x} R(x,u) + \gamma \max_v Q^*(x',v) - Q_t(x,u)\Big) = 0.
\end{aligned} \tag{A.44}$$

In a deterministic MDP for a fixed (x,u) there is a single state x' in the possible transitions. Therefore, $\frac{1}{n_x}\sum_{i\in n_x} R(x,u) + \gamma \max_v Q^*(x',v) - Q_t(x,u) \to 0 \iff |R(x,u) + \gamma \max_v Q^*(x',v) - Q_t(x,u)| \to 0$, and *a.s.*

$$\lim_{t\to\infty} |R(x,u) + \gamma \max_v Q^*(x',v) - Q_t(x,u)| = 0 \iff \lim_{t\to\infty} L(t) = 0,$$

which happens almost surely.

For the stochastic transition MDP, $\lim_{t\to\infty} \hat{Q}_t(x,u) = Q^*(x,u)$ and for any $i \in N_x$ with observed transition (x_i,u_i,y_i):

$$\lim_{t\to\infty} R(x_i,u_i) + \gamma \max_v Q^*(y_i,v) - Q_t(x_i,u_i) = \gamma\big(\mathbb{E}_{y\sim P(x_i,u_i,\cdot)}\big[\max_v Q^*(y,v)\big] - \max_v Q^*(y_i,v)\big).$$

Finally, from Assumption 5.2, all (x,u) are visited infinitely often. Therefore,

$$\Big\|\big(\max_v Q^*(y_i,v) - \mathbb{E}_{y\sim P(x_i,u_i,\cdot)} \max_v Q^*(y,v)\big)\Big\|_\infty =: l^*,$$

and therefore $\lim_{t\to\infty} L(t) \le l^*$ *a.s.* □

Theorem 5.1. We show this by contradiction. Assume first that $\exists t_0$ such that a communication event is never triggered for $t > t_0$. Then, for any agent $i \in N$ it holds that

$$\Big|R(x_i,u_i) + \gamma \max_v Q_t(y_i,v) - Q_t(x_i,u_i)\Big| < \max\{\rho L_i(t), \epsilon\}.$$

Take first $\max\{\beta L_i(t), \epsilon_\chi\} = \epsilon_\chi$. This implies the desired result, $|R(x_i,u_i) + \gamma \max_v Q_t(y_i,v) - Q_t(x_i,u_i)| < \epsilon_\chi$.

Now take $\max\{\beta L_i(t), \epsilon_\chi\} = \beta L_i(t) \implies |R(x_i,u_i) + \gamma \max_v Q_t(y_i,v) - Q_t(x_i,u_i)| < \beta L_i(t)$, and observe $L_i(t+1) \le (1-\beta(1-\beta))L_i(t)$. Therefore, $\exists t_\epsilon \ge t_0 \ : \ L_i(t_\epsilon) < \epsilon_\chi \implies \Big|R(x_i,u_i) +$

$\gamma \max_v Q_t(y_i, v) - Q_t(x_i, u_i)\big| < \epsilon_\chi$, and $\forall (x,u), t > t_\epsilon$:

$$\begin{aligned}
&\Big|R(x,u) + \gamma \max_v Q_t(y,v) - Q_t(x,u)\Big| \le \epsilon_\chi \Longrightarrow \\
\Longrightarrow &\Big|Q^*(x,u) - Q_t(x,u) + \gamma\Big(\max_v Q_t(y,v) - \max_v Q^*(y,v)\Big)\Big| \le \epsilon_\chi \Longrightarrow \\
\Longrightarrow &\Big|Q^*(x,u) - Q_t(x,u)\Big| \le \epsilon_\chi + \gamma \max_v \Big|Q_t(y,v) - Q^*(y,v)\Big| \le \epsilon_\chi + \gamma \|Q_t - Q^*\|_\infty \Longrightarrow \\
\Longrightarrow &\|Q^* - Q_t\|_\infty \le \epsilon_\chi + \gamma \|Q_t - Q^*\|_\infty \le \frac{\epsilon_\chi}{1-\gamma}.
\end{aligned} \tag{A.45}$$

Furthermore, it follows from (5.3) that no samples are transmitted for $t > t_\epsilon$, therefore Q_t has converged for $t > t_\epsilon$ to some Q_ϵ. Therefore, $\lim_{t\to\infty} \|Q^* - Q_t\|_\infty = \|Q^* - Q_\epsilon\|_\infty \le \frac{\epsilon_\chi}{1-\gamma}$.

Now assume that communication events happen infinitely often after some t_0. Since all pairs (x,u) are visited infinitely often and the MDP is deterministic, we have

$$\begin{aligned}
&\|H(Q_{t+1})(x,u) - Q_{t+1}(x,u)\|_\infty = \\
&= \Big\|R(x,u) + \gamma \max_v Q_{t+1}(y,v) - Q_t(x,u) - \alpha_t\big(R(x,u) + \gamma \max_v Q_t(y,v) - Q_t(x,u)\big)\Big\|_\infty \le \\
&\le \|(1-\alpha_t)\big(R(x,u) - Q_t(x,u)\big) + \gamma \max_v \Big(Q_t(y,v) + \alpha_t\big(R(y,v) + \gamma \max_{v'} Q_t(y',v') - Q_t(y,v)\big)\Big) - \\
&- \alpha_t \gamma \max_v Q_t(y,v)\|_\infty = \|(1-\alpha_t)\big(R(x,u) - Q_t(x,u) + \gamma \max_v Q_t(y,v)\big) + \\
&+ \gamma\alpha_t \max_v \big(R(y,v) + \gamma \max_{v'} Q_t(y',v') - Q_t(y,v)\big)\|_\infty \le \\
&\le (1-\alpha_t)\|R(x,u) - Q_t(x,u) + \gamma \max_v Q_t(y,v)\|_\infty + \gamma\alpha_t \|R(y,v) + \gamma \max_{v'} Q_t(y',v') - Q_t(y,v)\|_\infty \le \\
&\le (1-\alpha_t(1-\gamma))\|R(x,u) - Q_t(x,u) + \gamma \max_v Q_t(y,v)\|_\infty = \\
&= (1-\alpha_t(1-\gamma))\|H(Q_t)(x,u) - Q_t(x,u)\|_\infty.
\end{aligned}$$

Therefore, $\lim_{t\to\infty} \|\|H(Q_t)(x,u) - Q_t(x,u)\|_\infty = 0$, which implies no samples are transmitted as $t \longrightarrow \infty$ and contradicts the *infinitely often* assumption. From (A.45), $\lim_{t\to\infty} \|Q^* - Q_t\|_\infty \le \frac{\epsilon_\chi}{1-\gamma}$. □

Lemma 5.1. Take any $\hat{P} \in \mathscr{P}_P$. By the law of total expectation and making use of $\Pr[\hat{P}] = \mu_P(\hat{P})$, it follows that $\mathbb{E}[H_{\hat{P}}(Q_t) \mid \mathscr{F}_t, \hat{P}] = \sum_{\hat{P}\in\mathscr{P}_P} \mu_P(\hat{P}) H_{\hat{P}}(Q_t)(x,u) \equiv H_{\tilde{P}}(Q_t)$. To show that $\tilde{Q}$ is a fixed point, observe we can write

$$\begin{aligned}
&\sum_{\hat{P}\in\mathscr{P}_P} \mu_P(\hat{P}) \sum_y \hat{P}(x,u,y)\Big(R(x,u) + \gamma \max_v \tilde{Q}(y,v)\Big) = \\
&= \sum_y \Big(\sum_{\hat{P}} \mu_P(\hat{P})\hat{P}(x,u,y)\Big)\Big(R(x,u) + \gamma \max_v \tilde{Q}(y,v)\Big) = \\
&= \sum_y \tilde{P}(x,u,y)\Big(R(x,u) + \gamma \max_v \tilde{Q}(y,v)\Big).
\end{aligned}$$

□

Theorem 5.2. Define $\xi_t(x,u) := Q_t(x,u) - \tilde{Q}(x,u)$. Then, the iteration (2.1) applied at every time step is $\xi_{t+1}(x,u) = (1-\alpha_t)\xi_t(x,u) + \alpha_t(R(x,u) + \gamma \max_v Q_t(y,v) - \tilde{Q}(x,u))$. Now, from Lemma 5.1,

$$\begin{aligned}
&\|\mathbb{E}\left[R(x,u) + \gamma \max_v Q_t(y,v) - \tilde{Q}(x,u) \mid \mathcal{F}_t\right]\|_\infty = \\
&= \|\tilde{H}(Q_{t+1})(x,u) - \tilde{H}(\tilde{Q})(x,u)\|_\infty = \\
&= \gamma \|\tilde{P}(x,u,y)(\max_v Q_t(y,v) - \max_v \tilde{Q}(y,v))\|_\infty \leq \\
&\leq \gamma \|\tilde{P}(x,u,y)\|_\infty \|Q_t - \tilde{Q}\|_\infty \leq \gamma \|\xi_t(x,u)\|_\infty.
\end{aligned}$$

Therefore, the expected value of the operator $\tilde{H}$ is a γ-contraction in the sup-norm, with fixed point $\tilde{Q}$, and it follows that $\|\xi_t(x,u)\|_\infty \longrightarrow 0$ *a.s.* □

Corollary 5.1. Recall $H_{\tilde{P}}(\tilde{Q}) = \tilde{Q}$ and $H_P(Q^*) = Q^*$. Then,

$$\begin{aligned}
&\|Q^* - \tilde{Q}\|_\infty = \|H_P(Q^*) - H_{\tilde{P}}(\tilde{Q})\|_\infty = \\
&= \|\sum_y P(x,u,y)\left(R(x,u) + \gamma \max_v Q^*(y,v)\right) - \tilde{P}(x,u,y)\left(R(x,u) + \gamma \max_v \tilde{Q}(y,v)\right)\|_\infty = \\
&= \gamma \left\| \sum_y P(x,u,y) \max_v Q^*(y,v) - \tilde{P}(x,u,y) \max_v \tilde{Q}(y,v) \right\|_\infty .
\end{aligned} \tag{A.46}$$

Define $\delta P := \tilde{P} - P$ and substitute in (A.46):

$$\begin{aligned}
&\|Q^* - \tilde{Q}\|_\infty = \gamma \|\sum_y P(x,u,y)\left(\max_v Q^*(y,v) - \max_v \tilde{Q}(y,v)\right) - \delta P(x,u,y) \max_v \tilde{Q}(y,v)\|_\infty \leq \\
&\leq \gamma \|\sum_y P(x,u,y) \max_v |Q^*(y,v) - \tilde{Q}(y,v)|\|_\infty + \gamma \|\delta P(x,u,y) \max_v \tilde{Q}(y,v)\|_\infty .
\end{aligned} \tag{A.47}$$

Finally, observe $\|\sum_y P(x,u,y) \max_v |Q^*(y,v) - \tilde{Q}(y,v)|\|_\infty \leq \gamma \|Q^* - \tilde{Q}\|_\infty$. Additionally, since the reward functions are bounded, for a discount rate $\gamma \in (0,1)$ the values of $\tilde{Q}(x,u) \leq c$ are also bounded for some constant $c \in \mathbb{R}_{\geq 0}$. Therefore,

$$\begin{aligned}
\|Q^* - \tilde{Q}\|_\infty &\leq \gamma \|Q^* - \tilde{Q}\|_\infty + \gamma \|\delta P(x,u,y)\|_\infty \|\tilde{Q}\|_\infty \leq \\
&\leq \gamma \|Q^* - \tilde{Q}\|_\infty + c\gamma \|\delta P(x,u,y)\|_\infty \Longrightarrow \|Q^* - \tilde{Q}\|_\infty \leq \quad f\left(\|P - \tilde{P}\|_\infty\right),
\end{aligned} \tag{A.48}$$

with $f\left(\|P - \tilde{P}\|_\infty\right) := c\frac{\gamma}{1-\gamma}\|P - \tilde{P}\|_\infty$. □

Chapter 6

Proposition 6.1. Properties (6.2) and (6.3) hold by construction. First, all agents update their own $\hat{x}_t(i)$ based on the received communication, therefore all have the same last-known state. Second, whenever the condition $\|x_t(i) - \hat{x}_{t-1}(i)\|_\infty > \Gamma_\varepsilon(\hat{x})$ is violated, agent i transmits the new state measurement to others, and $\hat{x}_t$ is updated. Therefore $\|x_t - \hat{x}_t\|_\infty > \Gamma_\varepsilon(\hat{x}_t)$ holds for all times. □

Theorem 6.1. From Proposition 6.1, $\|x_t - \hat{x}_t\|_\infty \leq \Gamma_\varepsilon(\hat{x}_t) \forall t$, and recalling the expression for the optimal Q values:

$$Q^*(x_t, \pi^*(\hat{x}_t)) = \mathbb{E}[\hat{R}_t + \gamma V^*(x_{t+1}) \mid x_t] \geq V^*(x_t) - \varepsilon. \tag{A.49}$$

Now let $\hat{V}(x_0) := \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t \hat{r}_t \mid x_0]$ be the value of the policy obtained from executing the actions $\pi^*(\hat{x}_t)$. Then:

$$\begin{aligned}
&\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t \hat{R}_t \mid x_0] = \mathbb{E}[\hat{r}_0 + \gamma V^{\hat{\pi}}(x_1) \mid x_0] = \\
&= \mathbb{E}[\hat{R}_0 + \gamma \hat{V}(x_1) + \gamma V^*(x_1) - \gamma V^*(x_1) \mid x_0] = \\
&= \mathbb{E}[\hat{R}_0 + \gamma V^*(x_1) \mid x_0] + \gamma \mathbb{E}[\hat{V}(x_1) - V^*(x_1 \mid x_0].
\end{aligned} \tag{A.50}$$

Then, substituting (A.49) in (A.50):

$$\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t \hat{R}_t \mid x_0] \geq V^*(x_0) - \varepsilon + \gamma \mathbb{E}[\hat{V}(x_1) - V^*(x_1) \mid x_0]. \tag{A.51}$$

Now, observe we can apply the same principle as in (A.50) for the last term in (A.51),

$$\begin{aligned}
&\hat{V}(x_1) - V^*(x_1) = \mathbb{E}[\hat{R}_1 + \gamma \hat{V}(x_2) \mid x_1] - V^*(x_1) = \\
&= Q^*(x_1, \pi^*(\hat{x}_1)) + \gamma \mathbb{E}[\hat{V}(x_2) - V^*(x_2) \mid x_1] - V^*(x_1) \geq \\
&\geq V^*(x_1) - \varepsilon - V^*(x_1) + \gamma \mathbb{E}[\hat{V}(x_2) - V^*(x_2) \mid x_1] = \\
&= -\varepsilon + \gamma \mathbb{E}[\hat{V}(x_2) - V^*(x_2) \mid x_1].
\end{aligned} \tag{A.52}$$

Substituting (A.52) in (A.51):

$$\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t \hat{R}_t \mid x_0] \geq V^*(x_0) - \varepsilon - \gamma\varepsilon + \gamma^2 \mathbb{E}[\mathbb{E}[\hat{V}(x_2) - V^*(x_2) \mid x_1] \mid x_0]. \tag{A.53}$$

Now it is clear that, applying (A.52) recursively:

$$\begin{aligned}
&\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t \hat{R}_t \mid x_0] \geq V^*(x_0) - \varepsilon - \gamma\varepsilon + \gamma^2 \mathbb{E}[\mathbb{E}[\hat{V}(x_2) - V^*(x_2) \mid x_1] \mid x_0] \geq \\
&\geq V^*(x_0) - \varepsilon \sum_{k=0}^{\infty} \gamma^k.
\end{aligned} \tag{A.54}$$

Substituting $\sum_{k=0}^{\infty} \gamma^k = \frac{\gamma}{1-\gamma}$ in (A.54):

$$\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t \hat{R}_t \mid x_0] \geq V^*(x_0) - \varepsilon \frac{\gamma}{1-\gamma}.$$

□

A

Corollary 6.1. Take expression (6.6), and consider the action sequence executed by the agents to be $\{\hat{U}_t\}_{t=0}^{\infty}$. We can bound the total expectation of the sum of rewards by considering $\{\hat{U}_t\}_{t=0}^{\infty}$ to be a sequence of random variables that produce $Q^*(x_t, \hat{U}_t) \geq V^*(x_t) - \varepsilon$ with probability $1 - \overline{\epsilon}(s^*)$, and $Q^*(x_t, \hat{U}_t) \geq V^*(x_t) - \iota$ with probability $\overline{\epsilon}(s^*)$. Then,

$$\begin{aligned}\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t \hat{R}_t \mid x_0] = \mathbb{E}[\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t \hat{R}_t | \{\hat{U}_t\}, x_0]] = \mathbb{E}[\mathbb{E}[\hat{R}_0 + \gamma \hat{V}(x_1) | \{\hat{U}_t\}, x_0]] = \\ = \mathbb{E}[\mathbb{E}[\hat{R}_0 + \gamma V^*(x_1) | \{\hat{U}_t\} \mid x_0] + \gamma \mathbb{E}[\hat{V}(x_1) - V^*(x_1) | \{\hat{U}_t\}, x_0]].\end{aligned} \tag{A.55}$$

Observe now, for the first term in (A.55):

$$\begin{aligned}\mathbb{E}[\mathbb{E}[\hat{R}_0 + \gamma V^*(x_1) | \{\hat{U}_t\}, x_0]] \geq (1 - \overline{\epsilon}(s^*))(V^*(x_0) - \varepsilon) + \\ + \overline{\epsilon}(s^*)(V^*(x_0) - \iota) = V^*(x_0) - \varepsilon - \bar{\epsilon}(s^*)(\iota - \varepsilon).\end{aligned} \tag{A.56}$$

Take the second term in (A.55), and $\forall x_1 \in X$ given actions $\{\hat{U}_t\}$ it holds:

$$\begin{aligned}\mathbb{E}[\hat{V}(x_1) - V^*(x_1) \mid \{\hat{U}_t\}] \geq \mathbb{E}[\hat{R}_1 + \gamma V^*(x_2) + \gamma(\hat{V}(x_2)) - V^*(x_2)) - \\ - V^*(x_1) \mid \{\hat{U}_t\}] \geq -\varepsilon - \overline{\epsilon}(s^*)(\iota - \alpha) + \gamma \mathbb{E}[\hat{V}(x_2) - V^*(x_2)].\end{aligned}$$

Therefore, we can write

$$\begin{aligned}\gamma \mathbb{E}[\mathbb{E}_{x_0}[\hat{V}(x_1) - V^*(x_1) \mid \{\hat{U}_t\}]] = \gamma \mathbb{E}[\mathbb{E}[\hat{V}(x_1) - V^*(x_1) \mid \{\hat{U}_t\}] \mid x_0] \geq \\ \geq \gamma \left(-\varepsilon - \overline{\epsilon}(s^*)(\iota - \varepsilon) + \gamma \mathbb{E}[\mathbb{E}[\hat{V}(x_2) - V^*(x_2)] \mid x_0]\right)\end{aligned} \tag{A.57}$$

Finally, substituting (A.56) and (A.57) in (A.55):

$$\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t \hat{R}_t \mid x_0] \geq V^*(x_0) - (\varepsilon + \overline{\epsilon}(s^*)(\iota - \varepsilon)) \frac{\gamma}{1 - \gamma}. \tag{A.58}$$

□

Chapter 7

Proposition 7.1. If a policy $\pi \in \Pi$ is a fixed point of the operator $\langle \cdot, T \rangle$, then it holds that $\langle \pi, T \rangle = \pi$. Therefore, one can compute the robustness of the policy π to obtain $\rho(\pi, T) = J(\pi) - J(\langle \pi, T \rangle) = J(\pi) - J(\pi) = 0 \implies \pi \in \Pi_0$. Therefore, $\Pi_T \subseteq \Pi_0$.

For a discrete state and action spaces, the space of stochastic kernels $\mathscr{K} : X \mapsto \Delta(X)$ is equivalent to the space of row-stochastic $|X| \times |X|$ matrices, therefore one can write $T(y \mid x) \equiv T_{xy}$ as the xy–th entry of the matrix T. Then, the representation of a constant policy as an $X \times U$ matrix can be written as $\overline{\pi} = \mathbf{1}_{|X|} \nu^\top$, where $\nu \in \Delta(U)$ is any probability distribution over the action space. Observe that, applying the operator $\langle \pi, T \rangle$ to a constant policy yields:

$$\langle \overline{\pi}, T \rangle = T \mathbf{1}_{|X|} \nu^\top. \tag{A.59}$$

Matrix T is row-stochastic, and by the Perron-Frobenius Theorem [173] it has at least one eigenvalue $\text{eig}(T) = 1$, and this admits a (strictly positive) eigenvector $T\mathbf{1}_{|X|} = \mathbf{1}_{|X|}$. Therefore, substituting this in (A.59):

$$\langle \overline{\pi}, T \rangle = T \mathbf{1}_{|X|} \nu^\top = \mathbf{1}_{|X|} \nu^\top = \overline{\pi} \implies \overline{\Pi} \subseteq \Pi_T.$$

□

Proposition 7.2. Recall the definition in (7.1) and that the noise disadvantage function of a policy π is given by (7.3). We want to show that $D^{\pi}(x,T)=0 \implies \rho(\pi,T)=0$. Taking $D^{\pi}(x,T)=0$ one has a policy that produces an disadvantage of zero when noise kernel T is applied. Then,

$$D^{\pi}(x,T)=0 \implies \mathbb{E}_{u\sim\langle\pi,T\rangle(x)}[Q^{\pi}(x,u)]=V^{\pi}(x)\;\forall\, x\in X. \tag{A.60}$$

Now define the value of the disturbed policy

$$V^{\langle\pi,T\rangle}(x_0) := \mathbb{E}_{\substack{u_k\sim\langle\pi,T\rangle(x_k),\\ x_{k+1}\sim P(\cdot|x_k,u_k)}}\left[\sum_{k=0}^{\infty}\gamma^k R(x_k,u_k)\right],$$

and take:

$$V^{\langle\pi,T\rangle}(x)=\mathbb{E}_{\substack{u\sim\langle\pi,T\rangle(x),\\ y\sim P(\cdot|x,u)}}\left[R(x,u,y)+\gamma V^{\langle\pi,T\rangle}(y)\right].$$

We will now show that $V^{\pi}(x)=V^{\langle\pi,T\rangle}(x)$, for all $x\in X$. Observe, from (A.60) using $V^{\pi}(x)=\mathbb{E}_{u\sim\langle\pi,T\rangle(x)}[Q^{\pi}(x,u)]$, we have $\forall x\in X$:

$$\begin{aligned}
&V^{\pi}(x)-V^{\langle\pi,T\rangle}(x)=\\
=&\mathbb{E}_{u\sim\langle\pi,T\rangle(x)}[Q^{\pi}(x,u)]-\mathbb{E}_{\substack{u\sim\langle\pi,T\rangle(x)\\ y\sim P(\cdot|x,u)}}\left[R(x,u,y)+\gamma V^{\langle\pi,T\rangle}(y)\right]\\
=&\mathbb{E}_{\substack{u\sim\langle\pi,T\rangle(x)\\ y\sim P(\cdot|x,u)}}\left[R(x,u,y)+\gamma V^{\pi}(y)-R(x,u,y)-\gamma V^{\langle\pi,T\rangle}(y)\right]\\
=&\gamma\mathbb{E}_{y\sim P(\cdot|x,u)}\left[V^{\pi}(y)-V^{\langle\pi,T\rangle}(y)\right].
\end{aligned} \tag{A.61}$$

Now, taking the sup norm at both sides of (A.61) we get

$$\|V^{\pi}(x)-V^{\langle\pi,T\rangle}(x)\|_{\infty}=\gamma\left\|\mathbb{E}_{y\sim P(\cdot|x,u)}\left[V^{\pi}(y)-V^{\langle\pi,T\rangle}(y)\right]\right\|_{\infty}. \tag{A.62}$$

Observe that for the right hand side of (A.62), we have $\left\|\mathbb{E}_{y\sim P(\cdot|x,u)}\left[V^{\pi}(y)-V^{\langle\pi,T\rangle}(y)\right]\right\|_{\infty}\leq \|V^{\pi}(x)-V^{\langle\pi,T\rangle}(x)\|_{\infty}$. Therefore, since $\gamma<1$,

$$\begin{aligned}
&\|V^{\pi}(x)-V^{\langle\pi,T\rangle}(x)\|_{\infty}\leq\gamma\|V^{\pi}(x)-V^{\langle\pi,T\rangle}(x)\|_{\infty}\implies\\
\implies&\|V^{\pi}(x)-V^{\langle\pi,T\rangle}(x)\|_{\infty}=0.
\end{aligned}$$

Finally, $\|V^{\pi}(x)-V^{\langle\pi,T\rangle}(x)\|_{\infty}=0 \implies V^{\pi}(x)-V^{\langle\pi,T\rangle}(x)=0\;\forall x\in X$, and $V^{\pi}(x)-V^{\langle\pi,T\rangle}(x)=0\,\forall\, x\in X \implies J(\pi)=J(\langle\pi,T\rangle)\implies \rho(\pi,T)=0$. □

Inclusion Theorem 7.1. Combining Proposition 7.1 and Proposition 7.2, we simply need to show that $\Pi_T\subset\Pi_D$. Take π to be a fixed point of $\langle\pi,T\rangle$. Then $\langle\pi,T\rangle=\pi$, and from the definition in (7.3):

$$\begin{aligned}
D^{\pi}(x,T)=&V^{\pi}(x)-\mathbb{E}_{u\sim\langle\pi,T\rangle(x,\cdot)}[Q^{\pi}(x,u)]=V^{\pi}(x)-\mathbb{E}_{u\sim\pi(x,\cdot)}[Q^{\pi}(x,u)]=\\
=&V^{\pi}(x)-V^{\pi}(x)=0.
\end{aligned}$$

Therefore, $\pi \in \Pi_D$, which completes the inclusions.

To show convexity of $\overline{\Pi}, \Pi_T$, first for a constant policy $\overline{\pi} \in \overline{\Pi}$, recall that we can write $\overline{\pi} = \mathbf{1}\nu^\top$, where $\nu \in \Delta(U)$ is any probability distribution over the action space. Now take $\overline{\pi}_1, \overline{\pi}_2 \in \overline{\Pi}$. For any $\alpha \in [0,1]$, $\alpha\overline{\pi}_1 + (1-\alpha)\overline{\pi}_2 = \alpha\mathbf{1}\nu_1^\top + (1-\alpha)\mathbf{1}\nu_2^\top = \mathbf{1}(\alpha\nu_1 + (1-\alpha)\nu_2)^\top \in \overline{\Pi}$.

Finally, for the set Π_T, assume there exist two different policies π_1, π_2 both fixed points of $\langle \cdot, T\rangle$. Then, for any $\alpha \in [0,1]$, $\langle(\alpha\pi_1 + (1-\alpha)\pi_2), T\rangle = \alpha T\pi_1 + (1-\alpha)T\pi_2 = \alpha\pi_1 + (1-\alpha)\pi_2$. Therefore, any affine combination of fixed points is also a fixed point. □

Corollary 7.1. For statement (i), let $\overline{R}(\cdot,\cdot,\cdot) = c$ for some constant $c \in \mathbb{R}$. Then, $J(\pi) = \mathbb{E}_{x_0\sim\mu_0}[\sum_t \gamma^t \overline{R}_t \mid \pi] = \frac{c\gamma}{1-\gamma}$, which does not depend on the policy π. For any noise kernel T and policy π, $J(\pi) - J\langle\pi, T\rangle = 0 \implies \pi \in \Pi_0$.

For statement (ii) assume $\exists \pi \in \Pi_0 : \pi \notin \Pi_T$. Then, $\exists x^* \in X$ and $u^* \in U$ such that $\pi(x^*, u^*) \neq \langle \pi, T\rangle(x^*, u^*)$. Let:

$$\underline{R}(x, u, x') := \begin{cases} c & \text{if } x = x^* \text{ and } u = u^* \\ 0 & \text{otherwise} \end{cases}.$$

Then, $\mathbb{E}[R(x, \pi(x), x'] < \mathbb{E}[R(x, \langle\pi, T\rangle(x), x']$ and since the MDP is ergodic x is visited infinitely often and

$$J(\pi) - J(\langle\pi, T\rangle) > 0 \implies \pi \notin \Pi_0,$$

which contradicts the assumption. Therefore, $\Pi_0 \setminus \Pi_T = \emptyset \implies \Pi_0 = \Pi_T$. □

Lemma 7.1. We use standard results on stochastic approximation with non-expansive operators, specifically Theorem 2.3 [91]. First, observe that for a fully parameterised policy, one can assume to have a tabular representation such that $\pi_\theta(x, u) = \theta_{xu}$, and $\nabla_\theta \pi_\theta(x) \equiv \mathrm{Id}$. We can then write the stochastic gradient descent problem in terms of the policy. Let $y \sim \tilde{T}(\cdot \mid x)$. Then:

$$\begin{aligned}\pi_{t+1}(x) =& \pi_t(x) - \alpha_t\big(\pi_t(x) - \pi_t(y)\big) = \\ =& \pi_t(x) - \alpha_t\Big(\pi_t(x) - \langle\pi_t, \tilde{T}\rangle(x) - \big(\pi_t(y) - \langle\pi_t, \tilde{T}\rangle(x)\big)\Big).\end{aligned}$$

We now need to verify that the necessary conditions for applying Theorem 2.3 hold. First, α_t satisfies Assumption 2.2. Second, making use of the property $\|\tilde{T}\|_\infty = 1$ for any row-stochastic matrix $\tilde{T}$, for any two policies $\pi_1, \pi_2 \in \Pi$:

$$\begin{aligned}\|\langle\pi_1, \tilde{T}\rangle - \langle\pi_2, \tilde{T}\rangle\|_\infty =& \|\tilde{T}\pi_1 - \tilde{T}\pi_2\|_\infty = \|\tilde{T}(\pi_1 - \pi_2)\|_\infty \leq \\ \leq& \|\tilde{T}\|_\infty \|\pi_1 - \pi_2\|_\infty = \|\pi_1 - \pi_2\|_\infty.\end{aligned}$$

Therefore, the operator $\langle\cdot, \tilde{T}\rangle$ is non-expansive with respect to the sup-norm. For the final condition, we have

$$\mathbb{E}_{y\sim\tilde{T}(\cdot|x)}\left[\pi_t(y) - \langle\pi_t, \tilde{T}\rangle(x) \mid \pi_t, \tilde{T}\right] = \sum_{y\in X} \tilde{T}(y \mid x)\pi_t(y) - \langle\pi_t, \tilde{T}\rangle(x) = 0.$$

Therefore, the difference $\pi_t(y) - \langle\pi_t, \tilde{T}\rangle(x)$ is a martingale difference for all x. One can then apply Theorem 2.3 with $\xi_t(x) \equiv \pi_t(x)$, $F(\cdot) \equiv \langle\cdot, \tilde{T}\rangle$ and $M_{t+1} \equiv \pi_t(y) - \langle\pi_t, \tilde{T}\rangle(x)$ to

conclude that $\pi_t(x) \longrightarrow \tilde{\pi}(x)$ almost surely. Finally from assumption 7.1, for any policy all states $x \in X$ are visited infinitely often, therefore $\pi_t(x) \longrightarrow \tilde{\pi}(x) \forall x \in X \implies \pi_t \longrightarrow \tilde{\pi}$ and $\tilde{\pi}$ satisfies $\langle \tilde{\pi}, \tilde{T} \rangle = \tilde{\pi}$, and $K_{\tilde{T}}(\tilde{\pi}) = 0$. □

Theorem 7.2. We apply the results from [102] in Theorem 2.7. Essentially, authors in [102] prove that for a policy gradient algorithm to lexicographically optimise a policy for multiple objectives, it is a sufficient condition that the stochastic gradient descent algorithm finds optimal parameters for each of the objectives independently. From Lemma 7.1 we know that a policy gradient algorithm using the gradient estimate in (7.5) converges to a maximally robust policy, *i.e.* a set of parameters $\theta' = \text{argmax}_\theta K_{\tilde{T}}$. Additionally, by assumption, the chosen algorithm for K_1 converges to an optimal point θ^*. While the two objective functions are not of the same form – as in [102] – the fact they are both invex [174] either locally or globally depending on the form of K_1, implies that $\hat{K}$ is also invex and hence that the stationary point θ^ϵ computed by LRPG satisfies equation (7.7). □

B

Experimental Frameworks

B.1 Cooperative Path Planning Experiments

We modified the Frozen Lake environment in OpenAI GYM [175]. We edited the environment to have a bigger state-space (1296 (x, u) pairs), the agents get a reward $R = -1$ when choosing an action that makes them fall in a hole, and $R = 10$ when they find the goal state. Additionally, the agents get a constant reward of -0.01 every time they take an action, to reflect the fact that shorter paths are preferred. The action set is $\mathcal{U} = \{\text{up, down, left, right}\}$. For the stochastic transition case, the agents get a reward based on the pair (x, u) regardless of the end state x'. The resulting Frozen Lake environment can be seen in Figure B.1. We consider a population of $N \in \{8, 64\}$ agents, all using ε-greedy policies with different exploration rates (as proposed in [168]). The number of agents is chosen to be multiple of 8 (to facilitate running on parallel cores of the computer), to represent both a "large" and a "small" agent number scenario. The agents are initialised with a value $\varepsilon_i \in \{0.01, 0.2, 0.4, 0.6, 0.8.0.99\}$ chosen at random. For all the simulations we use $\alpha = 0.01$, $\gamma = 0.97$, $\beta = 0.05$. We plot results for $\varepsilon_\chi \in \{0.01, 0.05\}$. The Q–function is initialised randomly $Q_0(x, u) \in [-1, 1] \forall (x, u)$. The results are computed for 25 independent runs and averaged for each scenario. We present results for a stochastic and a deterministic MDP. In the stochastic case, for a given pair (x, u) there is a probability $p = 0.7$ of ending up at the corresponding state x' (e.g. moving down if the action chosen is *down*) and $\bar{p} = 0.3$ of ending at any other adjacent state.

To compare between the different scenarios, we use an experience replay buffer of size $N \times 1000$ for the central learner's memory, where at every episode we sample mini-batches of 32 samples. The policies are evaluated with a fixed $\varepsilon_0 = 0.01$, computing the rewards for 10 independent runs. The learning rate α and "diffusion" γ were picked based on similar size Q-learning examples in the literature. In the case of the ET related parameters β, ε_χ, these were picked after a very quick parameter scan. First, $\beta = 0.05$ yields a half-life time of ≈ 15 time steps, which is on the same order as the diameter of the path planning arena. The value function magnitude is related to the maximum reward in the MDP. Take a pair (x, u) being 1 step away from the path planning goal has an associated reward on the order of $\gamma \|R(x, u)\|_\infty \approx 9.7$. However, a pair (x, u) being 2 steps away has $\gamma^2 \|R(x, u)\|_\infty \approx 9.4$. Therefore, when being really close to the goal, the error associated with taking one extra

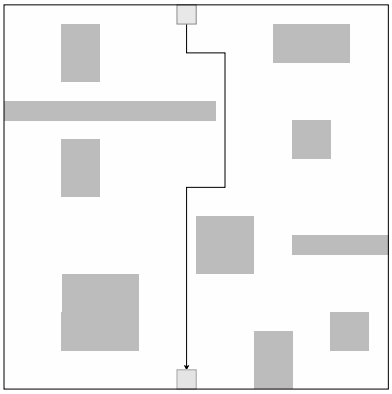


Figure B.1: Path Planning map used.

step is on the order of ≈ 0.03. By choosing $\varepsilon_\chi = 0.01$, we ensure the threshold is low enough to capture one-step errors. Then, $\varepsilon_\chi = 0.05$ is larger than this gap, so it ensures a significant enough difference for comparison.

B.2 Observational Robustness Experiments

We use in the experiments well-tested implementations of A2C and PPO adapted from [176] to include the computation of the lexicographic parameters in (2.2). Since all the environments use a pixel representation of the observation, we use a shared representation for the value function and policy, where the first component is a convolutional network, implemented as in [176]. The hyperparameters of the neural representations are presented in Table B.1.

Layer	Output	Func.
Conv1	16	ReLu
Conv2	32	ReLu
Conv3	64	ReLu
Fc4	256	ReLu

Table B.1: Shared Observation Layers

The actor and critic layers, for both algorithms, are a fully connected layer with 256 features as input and the corresponding output. We used in all cases an Adam optimiser. We optimised the parameters for each (vanilla) algorithm through a quick parameter search, and apply the same parameters for the Lexicographically Robust versions.

	LavaGap	LavaCrossing	DynObs
Steps	10^6	10^6	8×10^5
γ	0.99	0.999	0.99
α	0.001	0.001	0.001
ϵ(Adam)	10^{-8}	10^{-8}	10^{-8}
Grad. Clip	0.5	0.5	0.5
Gae	0.95	0.95	0.95
Rollout	256	512	256

Table B.2: A2C Parameters

	LavaGap	LavaCrossing	DynObs
Parallel Envs	8	8	8
Steps	10^6	10^6	8×10^5
γ	0.99	0.99	0.99
α	0.001	0.001	0.001
ϵ(Adam)	10^{-8}	10^{-8}	10^{-8}
Grad. Clip	0.5	0.5	0.5
Ratio Clip	0.2	0.2	0.2
Gae	0.95	0.95	0.95
Rollout	256	512	256
Epochs	10	10	10
Entr. Weight	0	0	0

Table B.3: PPO Parameters

For the implementation of the LRPG versions of the algorithms, in all cases we allow the algorithm to iterate for 1/3 of the total steps before starting to compute the robustness objectives. In other words, we use $\hat{K}(\theta) = K_1(\theta)$ until $t = \frac{1}{3}$ max_steps, and from this point we resume the lexicographic robustness computation as described in Algorithm 3. This is due to the structure of the environments simulated. The rewards (and in particular the positive rewards) are very sparse in the environments considered. Therefore, when computing the policy gradient steps, the loss for the primary objective is practically zero until the environment is successfully solved at least once. If we implement the combined lexicographic loss from the first time step, many times the algorithm would converge to a (constant) policy without exploring for enough steps, leading to convergence towards a maximally robust policy that does not solve the environment.

Noise Kernels We consider two types of noise; a normal distributed noise $\tilde{T}^g$ and a uniform distributed noise $\tilde{T}^u$. For the environments LavaGap and DynamicObstacles, the kernel $\tilde{T}^u$ produces a disturbed state $\tilde{x} = x + \xi$ where $\|\xi\|_\infty \leq 2$, and for LavaCrossing $\|\xi\|_\infty \leq 1.5$. The normal distributed noise is in all cases $\mathcal{N}(0, 0.5)$. The maximum norm of the noise is quite large, but this is due to the structure of the observations in these environments.

The pixel values are encoded as integers $0-9$, where each integer represents a different feature in the environment (empty space, doors, lava, obstacle, goal...). Therefore, any noise $\|\xi\|_\infty \leq 0.5$ would most likely not be enough to *confuse* the agent. On the other hand, too large noise signals are unrealistic and produce pathological environments. All the policies are then tested against two "true" noise kernels, $T_1 = \tilde{T}^u$ and $T_2 = \tilde{T}^g$. The main reason for this is to test both the scenarios where we assume a *wrong* noise kernel, and the case where we are training the agents with the correct kernel.

LRPG Parameters The LRL parameters are initialised in all cases as $\beta^1 = 2$, $\beta^2 = 1$, $\lambda = 0$ and $\eta = 0.001$. The LRL tolerance is set to $\epsilon_t = 0.99\hat{k}_1$ to ensure we never deviate too much from the original objective, since the environments have very sparse rewards. We use a first order approximation to compute the LRL weights from the original LMORL implementation.

Comparison with SA-PPO One of the baselines included is the State-Adversarial PPO algorithm proposed in [86]. We altered our implementation of PPO to incorporate the adversarial optimisation for the disturbances as described by [86]. The implementation includes an extra parameter that multiplies the regularisation objective, k_{ppo}. Since we were not able to find indications on the best parameter for discrete action environments, we implemented $k_{ppo} \in \{0.1, 1, 2\}$ and picked the best result for each entry in Table 7.1. Larger values seemed to de-stabilise the learning in some cases. The rest of the parameters are kept as in the vanilla PPO implementation.

C

Additional Results

C.1 Incorporating multiple target vertices in graphs

Consider two identical graphs G^1, G^2 at $t = 0$. Let $\hat{y}_t^{n,1}$ be the agent proportion in G^1 with probability distribution $y_t^{n,1}$, and $\hat{y}_t^{n,2}$ be the agent proportion in G^2 such that $y_0^{n,2} = \mathbf{0}$. Since the agents follow opposite weight fields (agents in graph 1 follow weights of graph 2, and vice-versa), let us write the following matrices by considering the union of both systems, $G^1 \cup G^2$:

$$w_t := \begin{pmatrix} w_t^1 \\ w_t^2 \end{pmatrix}, P^{\cup}(t,\varepsilon) := \begin{pmatrix} P^2(t,\varepsilon) & 0 \\ 0 & P^1(t,\varepsilon) \end{pmatrix},$$

and $\hat{y}_t^n := \begin{pmatrix} y_t^{n,1} \\ y_t^{n,2} \end{pmatrix}$. Note the reordering of the blocks in $P^{\cup}(t,\varepsilon)$, reflecting the fact that the agents follow opposite weights. The weight dynamics as written in (4.6) are then

$$\begin{aligned} w_{t+1}^1(i) &= (1-\rho) w_t^1(i) + \rho \hat{y}_t^{n,1}(i) R_t, \\ w_{t+1}^2(i) &= (1-\rho) w_t^2(i) + \rho \hat{y}_t^{n,2}(i) R_t, \end{aligned} \tag{C.1}$$

with $\mathbf{w}(0) = \mathbf{w}_0 \mathbf{1}$. Agents enter graph G^2 when they find the vertex $\mathscr{T}^1$, and go back to graph G^1 when they find vertex $\mathscr{S}^2$, resulting in two interconnected systems having each an inflow $(\mathbf{u}^1(t), \mathbf{u}^2(t) \in \mathbb{P}^{|X|})$ and outflow $(\mathbf{v}^1(t), \mathbf{v}^2(t) \in \mathbb{P}^{|X|})$ of agents exiting and entering the graphs. The dynamics for the agent distribution can be written as

$$\begin{aligned} \hat{\mathbf{y}}^1(t+1) &= P^2(t,\varepsilon) y_t^{n,1} + \mathbf{u}^1(t) + \mathbf{v}^1(t) \\ \hat{\mathbf{y}}^2(t+1) &= P^1(t,\varepsilon) y_t^{n,2} + \mathbf{u}^2(t) + \mathbf{v}^2(t). \end{aligned} \tag{C.2}$$

Define now the selector matrices $S \in \mathbb{R}^{|V| \times |V|}$ and $T \in \mathbb{R}^{|V| \times |V|}$ as diagonal matrices with $S_{ii}, T_{jj} = 1$ for $i = \mathscr{S}, j = \mathscr{T}$, zero otherwise. If $\mathbf{u}^1(t)$ is the distribution of agents entering graph G^1 from graph G^2 at time t and $\mathbf{v}^1(t)$ is the density of agents leaving G^1, both graphs

are interconnected and closed to external inputs, and then:

$$\begin{aligned} \mathbf{u}^1(t) \equiv -\mathbf{v}^2(t) &= SP^1(t,\varepsilon)y_t^{n,2} \\ \mathbf{v}^1(t) \equiv -\mathbf{u}^2(t) &= TP^2(t,\varepsilon)y_t^{n,1}. \end{aligned} \tag{C.3}$$

Therefore, substituting (C.3) in (C.2), the agent probability distribution dynamics are given by

$$\begin{aligned} \hat{y}_{t+1}^n &= \begin{pmatrix} (I-T)P^2(t,\varepsilon) & SP^1(t,\varepsilon) \\ TP^2(t,\varepsilon) & (I-S)P^1(t,\varepsilon) \end{pmatrix} \hat{y}_t^n = \\ &=: P(t,\varepsilon)\hat{y}_t^n. \end{aligned} \tag{C.4}$$

Furthermore, observe that the matrix $P(t,\varepsilon)$ in (C.4) is also column stochastic. Effectively, we have interconnected the two graphs by the vertices $\mathcal{S}$ and $\mathcal{T}$, and made the agents move according to the opposite pheromones.

Bibliography

References

[1] D. Hristu and K. Morgansen, "Limited communication control," *Systems & Control Letters*, vol. 37, no. 4, pp. 193–205, 1999. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167691199000201

[2] F.-L. Lian, J. Moyne, and D. Tilbury, "Network design consideration for distributed control systems," *IEEE Transactions on Control Systems Technology*, vol. 10, no. 2, pp. 297–307, 2002.

[3] F. Bullo, J. Cortés, and S. Martinez, *Distributed control of robotic networks: a mathematical approach to motion coordination algorithms*. Princeton University Press, 2009, vol. 27.

[4] P. Tabuada, "Event-triggered real-time scheduling of stabilizing control tasks," *IEEE Transactions on Automatic Control*, vol. 52, no. 9, pp. 1680–1685, 2007.

[5] B. Sinopoli, C. Sharp, L. Schenato, S. Schaffert, and S. Sastry, "Distributed control applications within sensor networks," *Proceedings of the IEEE*, vol. 91, no. 8, pp. 1235–1246, 2003.

[6] P. Tabuada, "Event-triggered real-time scheduling of stabilizing control tasks," *IEEE Transactions on Automatic Control*, vol. 52, no. 9, pp. 1680–1685, 2007.

[7] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 26, no. 1, pp. 29–41, 1996.

[8] M. Dorigo and G. Di Caro, "Ant colony optimization: a new meta-heuristic," in *Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406)*, vol. 2. IEEE, 1999, pp. 1470–1477.

[9] M. Dorigo and C. Blum, "Ant colony optimization theory: A survey," *Theoretical Computer Science*, vol. 344, no. 2, pp. 243 – 278, 2005. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0304397505003798

[10] M. A. Hsieh, Á. Halász, S. Berman, and V. Kumar, "Biologically inspired redistribution of a swarm of robots among multiple sites," *Swarm Intelligence*, vol. 2, no. 2, pp. 121–141, 2008.

[11] A. Drogoul and J. Ferber, "Some experiments with foraging robots," in *From Animals to Animats 2: Proceedings of the Second International Conference on Simulation of Adaptive Behavior*, vol. 2. MIT Press, 1993, p. 451.

[12] K. Sugawara, T. Kazama, and T. Watanabe, "Foraging behavior of interacting robots with virtual pheromone," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 3. IEEE, 2004, pp. 3074–3079.

[13] R. Fujisawa, S. Dobata, K. Sugawara, and F. Matsuno, "Designing pheromone communication in swarm robotics: Group foraging behavior mediated by chemical substance," *Swarm Intelligence*, vol. 8, no. 3, pp. 227–246, 2014. [Online]. Available: https://doi.org/10.1007/s11721-014-0097-z

[14] A. Campo, Á. Gutiérrez, S. Nouyan, C. Pinciroli, V. Longchamp, S. Garnier, and M. Dorigo, "Artificial pheromone for path selection by a foraging swarm of robots," *Biological cybernetics*, vol. 103, no. 5, pp. 339–352, 2010.

[15] S. Alers, K. Tuyls, B. Ranjbar-Sahraei, D. Claes, and G. Weiss, "Insect-inspired robot coordination: foraging and coverage," *Artificial life*, vol. 14, pp. 761–768, 2014.

[16] A. Font Llenas, M. S. Talamali, X. Xu, J. A. R. Marshall, and A. Reina, "Quality-sensitive foraging by a robot swarm through virtual pheromone trails," in *Swarm Intelligence*, M. Dorigo, M. Birattari, C. Blum, A. L. Christensen, A. Reina, and V. Trianni, Eds. Cham: Springer International Publishing, 2018, pp. 135–149.

[17] D. Payton, M. Daily, R. Estowski, M. Howard, and C. Lee, "Pheromone robotics," *Autonomous Robots*, vol. 11, no. 3, pp. 319–324, 2001.

[18] S. Garnier, F. Tache, M. Combe, A. Grimal, and G. Theraulaz, "Alice in pheromone land: An experimental setup for the study of ant-like robots," in *2007 IEEE swarm intelligence symposium*. IEEE, 2007, pp. 37–44.

[19] B. Hrolenok, S. Luke, K. Sullivan, and C. Vo, "Collaborative foraging using beacons." in *AAMAS*, vol. 10, 2010, pp. 1197–1204.

[20] F. Ducatelle, G. A. Di Caro, C. Pinciroli, F. Mondada, and L. Gambardella, "Communication assisted navigation in robotic swarms: self-organization and cooperation," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2011, pp. 4981–4988.

[21] K. Russell, M. Schader, K. Andrea, and S. Luke, "Swarm robot foraging with wireless sensor motes," in *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*. Citeseer, 2015, pp. 287–295.

[22] S. Nouyan, A. Campo, and M. Dorigo, "Path formation in a robot swarm," *Swarm Intelligence*, vol. 2, no. 1, pp. 1–23, 2008.

[23] A. Reina, A. J. Cope, E. Nikolaidis, J. A. Marshall, and C. Sabo, "Ark: Augmented reality for kilobots," *IEEE Robotics and Automation letters*, vol. 2, no. 3, pp. 1755–1761, 2017.

[24] M. S. Talamali, T. Bose, M. Haire, X. Xu, J. A. Marshall, and A. Reina, "Sophisticated collective foraging with minimalist agents: a swarm robotics test," *Swarm Intelligence*, vol. 14, no. 1, pp. 25–56, 2020.

[25] J.-M. Lasry and P.-L. Lions, "Mean field games," *Japanese journal of mathematics*, vol. 2, no. 1, pp. 229–260, 2007.

[26] D. A. Gomes *et al.*, "Mean field games models—a brief survey," *Dynamic Games and Applications*, vol. 4, no. 2, pp. 110–154, 2014.

[27] S. Grammatico, F. Parise, M. Colombino, and J. Lygeros, "Decentralized convergence to nash equilibria in constrained deterministic mean field control," *IEEE Transactions on Automatic Control*, vol. 61, no. 11, pp. 3315–3329, 2015.

[28] A. Bensoussan, J. Frehse, P. Yam *et al.*, *Mean field games and mean field type control theory*. Springer, 2013, vol. 101.

[29] R. Buckdahn, J. Li, and S. Peng, "Mean-field backward stochastic differential equations and related partial differential equations," *Stochastic processes and their Applications*, vol. 119, no. 10, pp. 3133–3154, 2009.

[30] K. Lerman, A. Martinoli, and A. Galstyan, "A review of probabilistic macroscopic models for swarm robotic systems," in *International workshop on swarm robotics*. Springer, 2004, pp. 143–152.

[31] K. Elamvazhuthi, S. Biswal, and S. Berman, "Mean-field stabilization of robotic swarms to probability distributions with disconnected supports," in *2018 Annual American Control Conference (ACC)*. IEEE, 2018, pp. 885–892.

[32] L. Stella and D. Bauso, "Mean-field games for bio-inspired collective decision-making in dynamical networks," *arXiv preprint arXiv:1802.03435*, 2018.

[33] K. Elamvazhuthi and S. Berman, "Mean-field models in swarm robotics: A survey," *Bioinspiration & Biomimetics*, vol. 15, no. 1, p. 015001, 2019.

[34] M. Mazo and P. Tabuada, "Decentralized event-triggered control over wireless sensor/actuator networks," *IEEE Transactions on Automatic Control*, vol. 56, no. 10, pp. 2456–2461, 2011.

[35] ——, "On event-triggered and self-triggered control over sensor/actuator networks," in *2008 47th IEEE Conference on Decision and Control*. IEEE, 2008, pp. 435–440.

[36] J. George and P. Gurram, "Distributed stochastic gradient descent with event-triggered communication," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 05, 2020, pp. 7169–7178.

[37] F. Solowjow and S. Trimpe, "Event-triggered learning," *Automatica*, vol. 117, p. 109009, 2020.

[38] K. G. Vamvoudakis and H. Ferraz, "Model-free event-triggered control algorithm for continuous-time linear systems with optimal performance," *Automatica*, vol. 87, pp. 412–420, 2018.

[39] X. Zhong, Z. Ni, H. He, X. Xu, and D. Zhao, "Event-triggered reinforcement learning approach for unknown nonlinear continuous-time system," in *2014 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2014, pp. 3677–3684.

[40] N. Funk, D. Baumann, V. Berenz, and S. Trimpe, "Learning event-triggered control from data through joint optimization," *IFAC Journal of Systems and Control*, vol. 16, p. 100144, 2021.

[41] D. V. Dimarogonas, E. Frazzoli, and K. H. Johansson, "Distributed event-triggered control for multi-agent systems," *IEEE Transactions on Automatic Control*, vol. 57, no. 5, pp. 1291–1297, 2011.

[42] E. Tsang, K. Fai, and K. H. Johansson, "Distributed event-triggered learning-based control for nonlinear multi-agent systems," in *2021 60th IEEE Conference on Decision and Control (CDC)*, 2021, pp. 3399–3405.

[43] R. Becker, S. Zilberstein, and V. Lesser, "Decentralized markov decision processes with event-driven interactions," in *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 1*. Citeseer, 2004, pp. 302–309.

[44] Y. Lin, K. Zhang, Z. Yang, Z. Wang, T. Başar, R. Sandhu, and J. Liu, "A communication-efficient multi-agent actor-critic algorithm for distributed reinforcement learning," in *2019 IEEE 58th Conference on Decision and Control (CDC)*. IEEE, 2019, pp. 5562–5567.

[45] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[46] S. Sen, M. Sekaran, J. Hale *et al.*, "Learning to coordinate without sharing information," in *AAAI*, vol. 94, 1994, pp. 426–431.

[47] D. H. Ackley and M. L. Littman, "Altruism in the evolution of communication," in *Artificial life IV*. Cambridge, MA, 1994, pp. 40–48.

[48] D. Szer and F. Charpillet, "Improving coordination with communication in multi-agent reinforcement learning," in *16th IEEE International Conference on Tools with Artificial Intelligence*. IEEE, 2004, pp. 436–440.

[49] J. R. Kok and N. Vlassis, "Sparse cooperative q-learning," in *Proceedings of the twenty-first international conference on Machine learning*, 2004, p. 61.

[50] C. Zhang and V. Lesser, "Coordinating multi-agent reinforcement learning with limited communication," in *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, 2013, pp. 1101–1108.

[51] J. N. Foerster, Y. M. Assael, N. De Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," *arXiv preprint arXiv:1605.06676*, 2016.

[52] D. Kim, S. Moon, D. Hostallero, W. J. Kang, T. Lee, K. Son, and Y. Yi, "Learning to schedule communication in multi-agent reinforcement learning," *arXiv preprint arXiv:1902.01554*, 2019.

[53] T. Chen, K. Zhang, G. B. Giannakis, and T. Basar, "Communication-efficient distributed reinforcement learning," *arXiv preprint arXiv:1812.03239*, 2018.

[54] J. Lin, K. Dzeparoska, S. Q. Zhang, A. Leon-Garcia, and N. Papernot, "On the robustness of cooperative multi-agent reinforcement learning," in *2020 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2020, pp. 62–68.

[55] F. L. Da Silva, P. Hernandez-Leal, B. Kartal, and M. E. Taylor, "Uncertainty-aware action advising for deep reinforcement learning agents," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 5792–5799.

[56] M. O. Karabag, C. Neary, and U. Topcu, "Planning not to talk: Multiagent systems that are robust to communication loss," 2022.

[57] W. Xue, W. Qiu, B. An, Z. Rabinovich, S. Obraztsova, and C. K. Yeo, "Mis-spoke or mis-lead: Achieving robustness in multi-agent communicative reinforcement learning," *arXiv preprint arXiv:2108.03803*, 2021.

[58] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[59] J. Morimoto and K. Doya, "Robust reinforcement learning," *Neural computation*, vol. 17, no. 2, pp. 335–359, 2005.

[60] J. K. Satia and R. E. Lave Jr, "Markovian decision processes with uncertain transition probabilities," *Operations Research*, vol. 21, no. 3, pp. 728–740, 1973.

[61] M. Heger, "Consideration of risk in reinforcement learning," in *Machine Learning Proceedings 1994*. Elsevier, 1994, pp. 105–111.

[62] A. Nilim and L. El Ghaoui, "Robust control of markov decision processes with uncertain transition matrices," *Operations Research*, vol. 53, no. 5, pp. 780–798, 2005.

[63] H. Xu and S. Mannor, "The robustness-performance tradeoff in markov decision processes," *Advances in Neural Information Processing Systems*, vol. 19, 2006.

[64] L. Pinto, J. Davidson, R. Sukthankar, and A. Gupta, "Robust adversarial reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2017, pp. 2817–2826.

[65] M. Everett, B. Lütjens, and J. P. How, "Certifiable robustness to adversarial state uncertainty in deep reinforcement learning," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[66] A. Pattanaik, Z. Tang, S. Liu, G. Bommannan, and G. Chowdhary, "Robust deep reinforcement learning with adversarial attacks," *arXiv preprint arXiv:1712.03632*, 2017.

[67] M. Fischer, M. Mirman, S. Stalder, and M. Vechev, "Online robustness training for deep reinforcement learning," *arXiv preprint arXiv:1911.00887*, 2019.

[68] J. Kos and D. Song, "Delving into adversarial attacks on deep policies," *arXiv preprint arXiv:1705.06452*, 2017.

[69] S. Huang, N. Papernot, I. Goodfellow, Y. Duan, and P. Abbeel, "Adversarial attacks on neural network policies," *arXiv preprint arXiv:1702.02284*, 2017.

[70] J. Fu, K. Luo, and S. Levine, "Learning robust rewards with adverserial inverse reinforcement learning," in *International Conference on Learning Representations*, 2018.

[71] A. Pattanaik, Z. Tang, S. Liu, G. Bommannan, and G. Chowdhary, "Robust deep reinforcement learning with adversarial attacks," in *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, ser. AAMAS '18. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2018, p. 2040–2042.

[72] M. Pirotta, M. Restelli, A. Pecorino, and D. Calandriello, "Safe policy iteration," in *International Conference on Machine Learning*. PMLR, 2013, pp. 307–315.

[73] M. A. Abdullah, H. Ren, H. B. Ammar, V. Milenkovic, R. Luo, M. Zhang, and J. Wang, "Wasserstein robust reinforcement learning," *arXiv preprint arXiv:1907.13196*, 2019.

[74] W. Wiesemann, D. Kuhn, and M. Sim, "Distributionally robust convex optimization," *Operations Research*, vol. 62, no. 6, pp. 1358–1376, 2014.

[75] B. P. Van Parys, D. Kuhn, P. J. Goulart, and M. Morari, "Distributionally robust control of constrained stochastic systems," *IEEE Transactions on Automatic Control*, vol. 61, no. 2, pp. 430–442, 2015.

[76] E. Derman, D. Mankowitz, T. Mann, and S. Mannor, "A bayesian approach to robust reinforcement learning," in *Uncertainty in Artificial Intelligence*. PMLR, 2020, pp. 648–658.

[77] A. Gleave, M. Dennis, C. Wild, N. Kant, S. Levine, and S. Russell, "Adversarial policies: Attacking deep reinforcement learning," in *International Conference on Learning Representations*, 2020.

[78] Y.-C. Lin, Z.-W. Hong, Y.-H. Liao, M.-L. Shih, M.-Y. Liu, and M. Sun, "Tactics of adversarial attack on deep reinforcement learning agents," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, 2017, pp. 3756–3762.

[79] C. Tessler, Y. Efroni, and S. Mannor, "Action robust reinforcement learning and applications in continuous control," in *International Conference on Machine Learning*. PMLR, 2019, pp. 6215–6224.

[80] X. Pan, D. Seita, Y. Gao, and J. Canny, "Risk averse robust adversarial reinforcement learning," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 8522–8528.

[81] K. L. Tan, Y. Esfandiari, X. Y. Lee, S. Sarkar *et al.*, "Robustifying reinforcement learning agents via action space adversarial training," in *2020 American control conference (ACC)*. IEEE, 2020, pp. 3959–3964.

[82] R. Klima, D. Bloembergen, M. Kaisers, and K. Tuyls, "Robust temporal difference learning for critical domains," in *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, ser. AAMAS '19. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2019, p. 350–358.

[83] J. Moos, K. Hansel, H. Abdulsamad, S. Stark, D. Clever, and J. Peters, "Robust reinforcement learning: A review of foundations and recent advances," *Machine Learning and Knowledge Extraction*, vol. 4, no. 1, pp. 276–315, 2022.

[84] V. Behzadan and A. Munir, "Vulnerability of deep reinforcement learning to policy induction attacks," in *International Conference on Machine Learning and Data Mining in Pattern Recognition*. Springer, 2017, pp. 262–275.

[85] A. Mandlekar, Y. Zhu, A. Garg, L. Fei-Fei, and S. Savarese, "Adversarially robust policy learning: Active construction of physically-plausible perturbations," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 3932–3939.

[86] H. Zhang, H. Chen, C. Xiao, B. Li, M. Liu, D. Boning, and C.-J. Hsieh, "Robust deep reinforcement learning against adversarial perturbations on state observations," *Advances in Neural Information Processing Systems*, vol. 33, pp. 21 024–21 037, 2020.

[87] H. Zhang, H. Chen, D. Boning, and C.-J. Hsieh, "Robust reinforcement learning on state observations with learned optimal adversary," in *International Conference on Learning Representation (ICLR)*, 2021.

[88] R. Diestel, "Graph theory, volume 173 of," *Graduate texts in mathematics*, p. 7, 2012.

[89] A. Gut, *Probability: a graduate course*. Springer Science & Business Media, 2013, vol. 75.

[90] T. C. Gard, *Introduction to Stochastic Differential Equations. Monographs and Textbooks in pure and applied mathematics*. Dekker, Inc, 1988.

[91] V. Borkar and K. Soumyanatha, "An analog scheme for fixed point computation. i. theory," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 44, no. 4, pp. 351–355, 1997.

[92] P. Brémaud, *Markov chains: Gibbs fields, Monte Carlo simulation, and queues*. Springer Science & Business Media, 2013, vol. 31.

[93] E. Seneta, *Non-negative matrices and Markov chains*. Springer Science & Business Media, 2006.

[94] Y. Qin, M. Cao, and B. D. O. Anderson, "Lyapunov criterion for stochastic systems and its applications in distributed computation," *IEEE Transactions on Automatic Control*, pp. 1–1, 2019.

[95] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[96] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[97] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[98] V. Konda and J. Tsitsiklis, "Actor-critic algorithms," *Advances in neural information processing systems*, vol. 12, 1999.

[99] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Proceedings of the 12th International Conference on Neural Information Processing Systems*, ser. NIPS'99. Cambridge, MA, USA: MIT Press, 1999, pp. 1057–1063.

[100] S. Bhatnagar, M. Ghavamzadeh, M. Lee, and R. S. Sutton, "Incremental natural actor-critic algorithms," *Advances in neural information processing systems*, vol. 20, 2007.

[101] K. Van Moffaert and A. Nowé, "Multi-objective reinforcement learning using sets of pareto dominating policies," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 3483–3512, 2014.

[102] J. Skalse, L. Hammond, C. Griffin, and A. Abate, "Lexicographic multi-objective reinforcement learning," in *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*, jul 2022, pp. 3430–3436.

[103] D. P. Bertsekas, "Nonlinear programming," *Journal of the Operational Research Society*, vol. 48, no. 3, pp. 334–334, 1997.

[104] D. J. Ornia and M. Mazo, "Convergence of ant colony multi-agent swarms," in *Proceedings of the 23rd International Conference on Hybrid Systems: Computation and Control*, ser. HSCC '20. New York, NY, USA: Association for Computing Machinery, 2020. [Online]. Available: https://doi.org/10.1145/3365365.3382199

[105] P.-P. Grassé, "La reconstruction du nid et les coordinations interindividuelles chezbellicositermes natalensis etcubitermes sp. la théorie de la stigmergie: Essai d'interprétation du comportement des termites constructeurs," *Insectes sociaux*, vol. 6, no. 1, pp. 41–80, 1959.

[106] J.-L. Deneubourg, S. Aron, S. Goss, and J. M. Pasteels, "The self-organizing exploratory pattern of the argentine ant," *Journal of insect behavior*, vol. 3, no. 2, pp. 159–168, 1990.

[107] X. Fan, X. Luo, S. Yi, S. Yang, and H. Zhang, "Optimal path planning for mobile robots based on intensified ant colony optimization algorithm," in *IEEE International Conference on Robotics, Intelligent Systems and Signal Processing, 2003. Proceedings. 2003*, vol. 1. IEEE, 2003, pp. 131–136.

[108] M. P. Garcia, O. Montiel, O. Castillo, R. Sepúlveda, and P. Melin, "Path planning for autonomous mobile robot navigation with ant colony optimization and fuzzy cost function evaluation," *Applied Soft Computing*, vol. 9, no. 3, pp. 1102–1110, 2009.

[109] G.-Z. TAN, H. HE, and A. SLOMAN, "Ant colony system algorithm for real-time globally optimal path planning of mobile robots," *Acta Automatica Sinica*, vol. 33, no. 3, pp. 279 – 285, 2007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1874102907600123

[110] M. Brand, M. Masuda, N. Wehner, and X.-H. Yu, "Ant colony optimization algorithm for robot path planning," in *2010 International Conference On Computer Design and Applications*, vol. 3. IEEE, 2010, pp. V3–436.

[111] R. Fujisawa, H. Imamura, T. Hashimoto, and F. Matsuno, "Communication using pheromone field for multiple robots," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2008, pp. 1391–1396.

[112] R. A. Russell, "Heat trails as short-lived navigational markers for mobile robots," in *Proceedings of International Conference on Robotics and Automation*, vol. 4. IEEE, 1997, pp. 3534–3539.

[113] R. Johansson and A. Saffiotti, "Navigating by stigmergy: A realization on an rfid floor for minimalistic robots," in *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 245–252.

[114] E. W. Montroll and G. H. Weiss, "Random walks on lattices. ii," *Journal of Mathematical Physics*, vol. 6, no. 2, pp. 167–181, 1965.

[115] L. Lovász *et al.*, "Random walks on graphs: A survey," *Combinatorics, Paul erdos is eighty*, vol. 2, no. 1, pp. 1–46, 1993.

[116] Z. Zhang, T. Shan, and G. Chen, "Random walks on weighted networks," *Physical Review E*, vol. 87, no. 1, p. 012112, 2013.

[117] M. F. Shlesinger, "Asymptotic solutions of continuous-time random walks," *Journal of Statistical Physics*, vol. 10, no. 5, pp. 421–434, 1974.

[118] F. Merkl and S. W. W. Rolles, "Edge-reinforced random walk on a ladder," *Ann. Probab.*, vol. 33, no. 6, pp. 2051–2093, 11 2005. [Online]. Available: https://doi.org/10.1214/009117905000000396

[119] ——, "Asymptotic behavior of edge-reinforced random walks," *Ann. Probab.*, vol. 35, no. 1, pp. 115–140, 01 2007. [Online]. Available: https://doi.org/10.1214/009117906000000674

[120] T. Stutzle and M. Dorigo, "A short convergence proof for a class of ant colony optimization algorithms," *IEEE Transactions on evolutionary computation*, vol. 6, no. 4, pp. 358–365, 2002.

[121] W. J. Gutjahr, "A graph-based ant system and its convergence," *Future generation computer systems*, vol. 16, no. 8, pp. 873–888, 2000.

[122] ——, "Aco algorithms with guaranteed convergence to the optimal solution," *Information processing letters*, vol. 82, no. 3, pp. 145–153, 2002.

[123] D. J. Ornia, P. J. Zufiria, and M. M. Jr, "Mean field behavior of collaborative multiagent foragers," *IEEE Transactions on Robotics*, vol. 38, no. 4, pp. 2151–2165, 2022.

[124] G. Beni and J. Wang, "Swarm intelligence in cellular robotic systems," in *Robots and biological systems: towards a new bionics?* Springer, 1993, pp. 703–712.

[125] M. Dorigo, M. Birattari *et al.*, "Swarm intelligence." *Scholarpedia*, vol. 2, no. 9, p. 1462, 2007.

[126] C. Blum and D. Merkle, "Swarm intelligence," *Swarm Intelligence in Optimization; Blum, C., Merkle, D., Eds*, pp. 43–85, 2008.

[127] J. Kennedy, "Swarm intelligence," in *Handbook of nature-inspired and innovative computing*. Springer, 2006, pp. 187–219.

[128] O. Zedadra, N. Jouandeau, H. Seridi, and G. Fortino, "Multi-agent foraging: state-of-the-art and research challenges," *Complex Adaptive Systems Modeling*, vol. 5, no. 1, pp. 1–24, 2017.

[129] G. Bernasconi and J. E. Strassmann, "Cooperation among unrelated individuals: the ant foundress case," *Trends in Ecology & Evolution*, vol. 14, no. 12, pp. 477–482, 1999.

[130] C. R. Carroll and D. H. Janzen, "Ecology of foraging by ants," *Annual Review of Ecology and systematics*, vol. 4, no. 1, pp. 231–257, 1973.

[131] J. F. Traniello, "Foraging strategies of ants," *Annual review of entomology*, vol. 34, no. 1, pp. 191–210, 1989.

[132] K. Elamvazhuthi and S. Berman, "Mean-field models in swarm robotics: a survey," *Bioinspiration & Biomimetics*, vol. 15, no. 1, p. 015001, nov 2019. [Online]. Available: https://dx.doi.org/10.1088/1748-3190/ab49a4

[133] A. Dussutour, S. C. Nicolis, G. Shephard, M. Beekman, and D. J. Sumpter, "The role of multiple pheromones in food recruitment by ants," *Journal of Experimental Biology*, vol. 212, no. 15, pp. 2337–2348, 2009.

[134] T. J. Czaczkes, C. Grüter, L. Ellis, E. Wood, and F. L. Ratnieks, "Ant foraging on complex trails: route learning and the role of trail pheromones in lasius niger," *Journal of Experimental Biology*, vol. 216, no. 2, pp. 188–197, 2013.

[135] B. Meyer, "A tale of two wells: noise-induced adaptiveness in self-organized systems," in *2008 Second IEEE International Conference on Self-Adaptive and Self-Organizing Systems*. IEEE, 2008, pp. 435–444.

[136] L. Panait and S. Luke, "A pheromone-based utility model for collaborative foraging," in *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems, 2004. AAMAS 2004.* IEEE, 2004, pp. 36–43.

[137] R. Carmona, M. Laurière, and Z. Tan, "Model-free mean-field reinforcement learning: mean-field mdp and mean-field q-learning," *arXiv preprint arXiv:1910.12802*, 2019.

[138] J. A. Carrillo, Y.-P. Choi, and M. Hauray, "The derivation of swarming models: mean-field limit and wasserstein distances," in *Collective dynamics from bacteria to crowds*. Springer, 2014, pp. 1–46.

[139] N. Gast, B. Gaujal, and J.-Y. Le Boudec, "Mean field for markov decision processes: from discrete to continuous optimization," *IEEE Transactions on Automatic Control*, vol. 57, no. 9, pp. 2266–2280, 2012.

[140] D. Jarne Ornia and M. Mazo Jr, "Event-based communication in multi-agent distributed q-learning," *CoRR*, vol. abs/2109.01417, 2021. [Online]. Available: https://arxiv.org/abs/2109.01417

[141] G. Weiß, "Distributed reinforcement learning," in *The Biology and technology of intelligent autonomous agents*. Springer, 1995, pp. 415–428.

[142] A. Nair, P. Srinivasan, S. Blackwell, C. Alcicek, R. Fearon, A. De Maria, V. Panneershelvam, M. Suleyman, C. Beattie, S. Petersen *et al.*, "Massively parallel methods for deep reinforcement learning," *arXiv preprint arXiv:1507.04296*, 2015.

[143] D. Horgan, J. Quan, D. Budden, G. Barth-Maron, M. Hessel, H. Van Hasselt, and D. Silver, "Distributed prioritized experience replay," *arXiv preprint arXiv:1803.00933*, 2018.

[144] S. Kapturowski, G. Ostrovski, J. Quan, R. Munos, and W. Dabney, "Recurrent experience replay in distributed reinforcement learning," in *International conference on learning representations*, 2018.

[145] M. Tan, "Multi-agent reinforcement learning: Independent vs. cooperative agents," in *Proceedings of the tenth international conference on machine learning*, 1993, pp. 330–337.

[146] E. Yang and D. Gu, "Multiagent reinforcement learning for multi-robot systems: A survey," tech. rep, Tech. Rep., 2004.

[147] A. Nowé, P. Vrancx, and Y.-M. De Hauwere, "Game theory and multi-agent reinforcement learning," in *Reinforcement Learning*. Springer, 2012, pp. 441–470.

[148] L. Busoniu, R. Babuska, and B. De Schutter, "A comprehensive survey of multiagent reinforcement learning," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 38, no. 2, pp. 156–172, 2008.

[149] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," *arXiv preprint arXiv:1706.02275*, 2017.

[150] M. Lauer and M. Riedmiller, "An algorithm for distributed reinforcement learning in cooperative multi-agent systems," in *In Proceedings of the Seventeenth International Conference on Machine Learning*. Citeseer, 2000.

[151] L. Panait and S. Luke, "Cooperative multi-agent learning: The state of the art," *Autonomous agents and multi-agent systems*, vol. 11, no. 3, pp. 387–434, 2005.

[152] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[153] Q. Li, F. Gama, A. Ribeiro, and A. Prorok, "Graph neural networks for decentralized multi-robot path planning," *arXiv preprint arXiv:1912.06095*, 2019.

[154] F. S. Melo, "Convergence of q-learning: A simple proof," *Institute Of Systems and Robotics, Tech. Rep*, pp. 1–4, 2001.

[155] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *arXiv preprint arXiv:1511.05952*, 2015.

[156] D. Jarne Ornia and M. Mazo, "Robust event-driven interactions in cooperative multi-agent learning", booktitle=Formal Modeling and Analysis of Timed Systems, year=2022, publisher=Springer International Publishing, address=Cham, pages=281–297, isbn=978-3-031-15839-1," S. Bogomolov and D. Parker, Eds.

[157] M. C. Campi and S. Garatti, "Scenario optimization with relaxation: a new tool for design and application to machine learning problems," in *2020 59th IEEE Conference on Decision and Control (CDC)*. IEEE, 2020, pp. 2463–2468.

[158] G. C. Calafiore and M. C. Campi, "The scenario approach to robust control design," *IEEE Transactions on automatic control*, vol. 51, no. 5, pp. 742–753, 2006.

[159] S. Garatti and M. C. Campi, "Risk and complexity in scenario optimization," *Mathematical Programming*, pp. 1–37, 2019.

[160] B. Schölkopf, P. Bartlett, A. Smola, and R. C. Williamson, "Shrinking the tube: a new support vector regression algorithm," *Advances in neural information processing systems*, vol. 11, 1998.

[161] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *Journal of machine learning research*, vol. 9, no. 11, 2008.

[162] D. J. Ornia, L. Romao, L. Hammond, M. Mazo, and A. Abate, "Observational robustness and invariances in reinforcement learning via lexicographic objectives," 2022. [Online]. Available: https://arxiv.org/abs/2209.15320

[163] M. T. Spaan and N. Vlassis, "A point-based pomdp algorithm for robot planning," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, vol. 3. IEEE, 2004, pp. 2399–2404.

[164] M. T. Spaan, "Partially observable markov decision processes," in *Reinforcement Learning*. Springer, 2012, pp. 387–414.

[165] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *Proc. of the Sixteenth International Conference on Machine Learning, 1999*, 1999.

[166] E. van der Pol, T. Kipf, F. A. Oliehoek, and M. Welling, "Plannable approximations to mdp homomorphisms: Equivariance under actions," in *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, ser. AAMAS '20. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2020, p. 1431–1439.

[167] J. Skalse, M. Farrugia-Roberts, S. Russell, A. Abate, and A. Gleave, "Invariance in policy optimisation and partial identifiability in reward learning," *arXiv preprint arXiv:2203.07475*, 2022.

[168] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*. PMLR, 2016, pp. 1928–1937.

[169] B. Liu, Q. Cai, Z. Yang, and Z. Wang, "Neural trust region/proximal policy optimization attains globally optimal policy," in *Advances in Neural Information Processing Systems*, vol. 32. Curran Associates, Inc., 2019.

[170] V. R. Konda and J. N. Tsitsiklis, "Actor-critic algorithms," in *Advances in Neural Information Processing Systems 12*, S. A. Solla, T. K. Leen, and K. Müller, Eds. MIT Press, 2000, pp. 1008–1014.

[171] S. Bhatnagar, R. S. Sutton, M. Ghavamzadeh, and M. Lee, "Natural actor–critic algorithms," *Automatica*, vol. 45, no. 11, pp. 2471–2482, Nov. 2009.

[172] M. Chevalier-Boisvert, L. Willems, and S. Pal, "Minimalistic gridworld environment for openai gym," https://github.com/maximecb/gym-minigrid, 2018.

[173] R. A. Horn and C. R. Johnson, *Matrix analysis*. Cambridge university press, 2012.

[174] A. Ben-Israel and B. Mond, "What is invexity?" *The ANZIAM Journal*, vol. 28, no. 1, pp. 1–9, 1986.

[175] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[176] S. Zhang, "Modularized implementation of deep rl algorithms in pytorch," https://github.com/ShangtongZhang/DeepRL, 2018.

Glossary

ETC Event-triggered control.

MC Markov Chain.

MDP Markov Decision Process.

RL Reinforcement Learning.

MARL Multi-Agent Reinforcement Learning.

PG Policy Gradient (algorithms).

LRPG Lexicographically Robust Reinforcement Learning.

LMORL Lexicographic Multi-Objective Reinforcement Learning.

Curriculum Vitæ

Daniel Jarne Ornia

28-12-1993 Born in Bellevue, WA, United States of America.

Education

2011–2015 Aerospace Engineering
Universitat Politecnica de Catalunya
Barcelona, Spain

2015–2017 M.Sc. Aerospace Engineering — Space Systems and Physics
KTH Royal Institute of Technology
Stockholm, Sweden

2018–2022 Ph.D. Control and Learning for Multi-Agent Systems
Delft University of Technology
Delft, The Netherlands
Thesis: Efficient Control for Cooperation: Communication, Learning and Robustness in Multi-Agent Systems
Promotor: dr. Manuel Mazo Jr.
Co-Promotor: dr. Javier Alonso Mora

List of Publications

1. **D. Jarne Ornia** and M. Mazo Jr., "Convergence of Ant Colony Multi-Agent Swarms" in *Proceedings of the 23rd International Conference on Hybrid Systems: Computation and Control*, 2020.

2. S. Adams, **D. Jarne Ornia**, and M. Mazo Jr., "A Self-Guided Approach for Navigation in a Minimalistic Foraging Robotic Swarm", *Autonomous Robots*, 2023.

3. **D. Jarne Ornia**, P. J. Zufiria and M. Mazo Jr., "Mean Field Behavior of Collaborative Multi-Agent Foragers" in *IEEE Transactions on Robotics*, vol. 38, no. 4, pp. 2151-2165, 2022,

4. **D. Jarne Ornia** and M. Mazo Jr., "Event-Based Communication in Distributed Q-Learning," *61st IEEE Conference on Decision and Control*, 2022.

5. **D. Jarne Ornia** and M. Mazo Jr., "Robust Event-Driven Interactions in Cooperative Multi-Agent Learning" in *Formal Modeling and Analysis of Timed Systems (FORMATS)*, 2022.

6. **D. Jarne Ornia**, L. Romao, L. Hammond, M. Mazo Jr. and A. Abate "Observational Robustness and Invariances in Reinforcement Learning via Lexicographic Objectives", *arXiv Pre-Print*, 2022.

Included in this thesis.