Exploring the effects of conditioning Independent Q-Learners on the sufficient plan-time statistic for Dec-POMDPs

by

Alex Mandersloot

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Monday August 17, 2020 at 11:00 AM.

Student number:	4399986	
Faculty:	EEMCS	
Master Programme:	Computer Science	
Thesis committee:	Dr. F. A. Oliehoek,	TU Delft, Thesis Supervisor
	Prof. C. M. Jonker,	TU Delft
	Dr. M. M. de Weerdt,	TU Delft
	Dr. A. Czechowski,	TU Delft

Delft University of Technology

Acknowledgements

First and foremost, I would like to thank dr. Frans Oliehoek, my thesis supervisor. You have always challenged me to critically analyze related works, and to see the bigger picture. Moreover, you have continuously helped me to stay focused on what is important. The opportunity to work with someone as experienced in the field of Dec-POMDPs as you has been invaluable. Your ability to instantly come up with a potential solution to a problem I had been facing for days (and, admittedly, sometimes weeks) is nothing short of remarkable.

To dr. Aleksander Czechowski, I hope you realize how valuable you have been to me throughout this project as well. Our initial idea on Bridge really helped to get the ball rolling. Even though ultimately we ended up steering in a different direction, I feel that the ideas and papers we discussed were still highly relevant to this work. Moroever, you have continuously provided me with useful feedback for my intermediate results. Thank you.

Finally, I would like to thank dr. Alex Cowan for being a great friend. You have always shown interest in my work, and I really enjoyed casually chatting with you about my thesis and academic life in general. You have also helped me keep my expectations in check.

Abstract

The Decentralized Partially Observable Markov Decision Process is a commonly used framework to formally model scenarios in which multiple agents must collaborate using local information. A key difficulty in a Dec-POMDP is that in order to coordinate successfully, an agent must decide on actions not only using its own information, but also by reasoning about the information available to the other agents. Nevertheless, existing value-based Reinforcement Learning techniques for Dec-POMDPs typically take the individual perspective, under which each agent optimizes its own actions using solely its local information, thus essentially neglecting the presence of others. As a result, the concatenation of individual policies learned in this way has a tendency to result in a sub-optimal joint policy. In this work, we propose to additionally condition such Independent Q-Learners on the plan-time sufficient statistic for Dec-POMDPs, which contains a distribution over the joint action-observation history. Using this, the agents can accurately reason about the resulting actions the other agents will take, and adjust their own behavior accordingly. Our main contributions are threefold. (1) We thoroughly investigate the effects of conditioning Independent Q-Learners on the sufficient statistic for Dec-POMDPs. (2) We identify novel exploration strategies that the agents can follow by conditioning on the sufficient statistic, as well as their implications on the decision rules, the sufficient statistic and the learning process. (3) We substantiate and demonstrate that by conceptually sequencing the decision-making, and additionally conditioning the agents on the current decision rules of the earlier agents, such learners are able to consistently escape sub-optimal equilibria and learn the optimal policy in our test environment, Dec-Tiger.

Contents

Chapter 1

Introduction

In many real-world scenarios, such as the control of autonomous drones and cars, as well as the coordination of traffic lights, multiple entities must learn to cooperate successfully. For example, traffic lights must coordinate with each other in order to optimize the flow of traffic.

In such scenarios, often the entities, commonly referred to as agents, are not allowed to freely share the information that is available to them. For example, suppose a self-driving car would share its point of view to a coordinating system, which subsequently relays all information it receives from other cars in this way this back to it. This would require (1) a costly communication channel, (2) latency- and noise-free broadcasts, and (3) protective measures against the single point of failure. All of these combined prevent such a method from being practical in many cases.

Instead, in such scenarios agents base their decision-making on their local information. Moreover, in order to coordinate successfully, an agent must not only reason about its own information, but also about the information that is available to the other agents. For example, if we return to the example of a self-driving car, it would be wise for such a car not to perform a sudden manoeuvre while it is in the blind spot of the car in front of it.

The Decentralized Partially Observable Markov Decision Process was established to formally model such environments. In a Dec-POMDP, multiple agents cooperate in an environment under uncertainty, as they only have access to local information. Solving Dec-POMDPs using planning methods has a rich history, e.g. [1][2][3][4].

More recently, there have been many breakthroughs in the field of Deep Reinforcement Learning [5][6][7]. Such methods use a complex neural network function approximator to learn policies by repeatedly interacting with the environment. Existing applications of Deep Reinforcement Learning to Dec-POMDPs often take the perspective of Independent Learners that optimize each agent's individual policy based on its local information. While this improves the scalability of such an approach, it also has a tendency to result in a sub-optimal joint policy, as agents are essentially oblivious to the presence of other agents.

In this Thesis we propose to augment such Independent Learners with knowledge about the information available to the other agents. We make use of a key advancement that identifies a sufficient statistic of a Dec-POMDP. This sufficient statistic contains a probability distribution over the joint action-observation history, i.e. each agent's local information. Since optimal decision-making of an agent in a Dec-POMDP is tightly interwoven with the information available to the other agents (and their resulting actions), such augmented learners are equipped with an invaluable learning signal.

Our main contributions are threefold. (1) We thoroughly investigate the effects of conditioning Independent Q-Learners on the sufficient statistic for Dec-POMDPs. (2) We identify novel exploration strategies that the agents can follow by conditioning on the sufficient statistic, as well as their implications on the decision rules, the sufficient statistic and the learning process. (3) We substantiate and demonstrate that by conceptually sequencing the decision-making, and additionally conditioning the agents on the current decision rules of the earlier agents, such learners are able to consistently escape sub-optimal equilibria and learn the optimal policy.

This Thesis is structured as follows. In Chapter 2, we recite necessary background knowledge. In Chapter 3 we recite a key related work on the sufficient statistic for Dec-POMDPs. We furthermore re-introduce the reformulation of a Dec-POMDP as a Non-Observable Markov Decision Process, whose perspective is highly relevant to our work. In Chapter 4 we delineate the specifics of our approach. In Chapter 5 we formally state our research questions, as well as present an overview of our experimental setup. In our experimental Chapters 6, 7 and 8 we investigate various exploration strategies for our proposed learners. In Chapter 9, we delineate how decision-making in Dec-POMDPs can be conceptually sequenced and subsequently we perform experiments following this perspective. In Chapter 10 we describe related work to this Thesis. Finally, in Chapter 11 we conclude our Thesis.

.

Chapter 2

Background

In this Chapter we provide the necessary background knowledge on single and multi-agent decision-theoretic planning and learning for this Thesis. In this Thesis, we focus on the Decentralized Partially Observable Markov Decision Process, in which multiple agents have to collaborate using local information. We introduce this framework by starting with the simpler Markov Decision Process. We then gradually introduce partial observability as well as multiple agents.

Furthermore, we discuss a widely used baseline for multi-agent Reinforcement Learning, namely Independent Q-Learning. We also recite the Deep Q-Network, which uses a neural network as function approximator when tabular Q-Learning is infeasible. We conclude by briefly discussing how learning can be centralized in Dec-POMDPs, while still yielding decentralized policies.

2.1 Single Agent Decision-Theoretic Planning

2.1.1 Markov Decision Process

A Markov Decision Process (MDP) is a stochastic process, in which a single agent interacts with an environment in discrete time steps [8][9][10].

Definition 2.1.1. An MDP is defined by a tuple $\langle \mathcal{S}, \mathcal{A}, T, R \rangle$, where

- $\mathcal{S}$ is a finite set of states in the world
- $\mathcal{A}$ is a finite set of actions available to the agent.
- T: $\mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ is the transition probability function, which specifies $P(s_{t+1}|s_t, a_t)$.
- R: $\mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the reward function.

In this Section, we largely follow the exposition by Sutton et al. [10]. At each time step $t = 0, 1, 2, \ldots$, the agent is in a certain observable state $s_t \in \mathcal{S}$. Based on the state, it takes an action $a_t \in \mathcal{A}$ and the environment transitions to state $s_{t+1} \in \mathcal{S}$, as specified by the transition function. The agent subsequently receives a numerical reward $r_t \in \mathcal{R}$ specified by the reward function. The current timestep then concludes, and this process is repeated. The dynamics between an agent and the environment thus give rise to a sequence,

$$s_0, a_0, r_0, s_1, a_1, r_1, s_2, a_2, r_2, \dots, \tag{2.1.1}$$

which is commonly referred to as the trajectory.

Trajectories either end at a a specific horizon h, or continue endlessly ($h = \infty$). The goal of the agent to maximize the expected sum of rewards it receives during a trajectory:

$$\mathbb{E}\left[\sum_{t=0}^{h} \gamma^t \cdot r_t\right] \tag{2.1.2}$$

Here, $0 \leq \gamma \leq 1$ is a discount factor which usually incentivises the agent to obtain rewards sooner rather than later.

Within this framework, the reward and transition function only depend on the current state and action, which is known as the Markov Property. Consequently, since the agent can directly observe its current state in an MDP, the current state is a sufficient statistic. The agent can behave optimally if it remembers only the current state, as opposed to its entire history of states and actions. Intuitively, for example, if you can directly observe your position in a maze, your best action does not depend on *how* you got to this position.

In an MDP, a stochastic policy $\pi : \mathcal{S} \to \Delta\mathcal{A}$ maps states to a distribution over actions. A deterministic policy $\pi : \mathcal{S} \to \mathcal{A}$ simply maps each state to a single action. The goal of the agent is to find a policy which, when executed, maximizes Eq. 2.1.2.

We will introduce both the value function and the action value function here, as they will be used ubiquitously throughout this thesis. The value of a state s under a fixed policy π is the expectation of the rewards the agent will receive, given that it starts in s and follows π. It is defined as

$$\begin{aligned} V^\pi(s) &= \mathbb{E}^\pi\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| s_t = s\right] \\ &= R(s, \pi(s)) + \gamma \sum_{s' \in \mathcal{S}} T(s, \pi(s), s') V^\pi(s'), \end{aligned} \tag{2.1.3}$$

where $\pi(s)$ denotes the action we take in state s according to the policy π. Consequently, the optimal value of a state follows easily:

$$V^*(s) = \max_\pi V^\pi(s) \tag{2.1.4}$$

Besides the value function, the action value function under policy π, denoted by $Q^\pi(s, a)$, is also used frequently. It defines the value of taking action a from state s and consequently following π:

$$Q^{\pi}(s,a) = R(s,a) + \gamma \sum_{s' \in \mathcal{S}} T(s,a,s') Q^{\pi}(s', \pi(s')) \tag{2.1.5}$$

which differs from the value function in that we are now free to choose an immediate action and only follow the policy thereafter. For the optimal Q-function, instead of following π, we assume to behave optimally in the future:

$$Q^{*}(s,a) = R(s,a) + \gamma \sum_{s' \in \mathcal{S}} T(s,a,s') \max_{a'} Q^{*}(s', a'), \tag{2.1.6}$$

and the relation between the value function and the Q-function becomes apparent:

$$V^{*}(s) = \max_{a} Q^{*}(s,a). \tag{2.1.7}$$

2.1.2 Partially Observable Markov Decision Process

An MDP is used to model environments in which an agent directly observes the state it is in at any given moment. More commonly, however, the state is not directly observable. Instead, an agent receives noisy observations. For example, multiple states could result in the same observation, or the same state can result in different observations. To account for this partial observability, the Partially Observable Markov Decision Process [11][12][13][14] was established.

Definition 2.1.2. A POMDP is a tuple $\langle \mathcal{S}, \mathcal{A}, T, R, \Omega, O, b_0 \rangle$, where

- $\mathcal{S}$, $\mathcal{A}$, T, R are as in an MDP.
- Ω is the set of possible observations.
- O: $\mathcal{S} \times \mathcal{A} \times \Omega \to [0,1]$ is the observation probability function, which specifies $P(o_t|a_t, s_{t+1})$.
- b_0 is the initial state distribution.

Like before, at each timestep t, the agent is in state s_t, takes an action a_t, and the environment transitions to state s_{t+1}. However, contrary to an MDP, the agent can not directly observe s_t and s_{t+1}. Instead, it only receives an observation $o_t \in \Omega$ after taking action a_t and landing in state s_{t+1}, whereby the probability of a specific observation o_t is given by the observation function $O(s_{t+1}, a_t, o_t)$.

A simple example of a POMDP is described by Spaan [15]. Consider an environment with two states $\langle s, \dot{s} \rangle$ and two possible observations $\langle o, \dot{o} \rangle$. There is only one action, which is omitted for clarity. Therefore, the observation function $O(s,a,o)$ depends solely on $O(s,o)$. Suppose we have the following observation probabilities:

$$O(s,o) = 0.8 \qquad O(\dot{s},o) = 0.2$$
$$O(s,\dot{o}) = 0.2 \qquad O(\dot{s},\dot{o}) = 0.8$$

Now, when the agent receives observation o or $\dot{o}$, it can not tell with full certainty which state it is in.

In an MDP we directly observed the current state, which was a sufficient statistic. In a POMDP the agent receives an observation instead of directly observing the state. One can then wonder if the current observation is also a sufficient statistic for a POMDP: can the agent act optimally by only remembering its last observation, or does it have to memorize its entire history of observations?

It turns out that it cannot. See e.g. Section 3 in [16] for a counterexample. Instead of solely remembering its last observation, an agent should maintain a belief over states it can be in. This (state) belief summarizes all information about the agent's history of actions and observations in a vector b. Initially, the agent has a belief b_0 as defined by the environment, which contains, for each state, the probability that the agent starts in that state. After taking an action a from state s and receiving observation o, the belief of being in state s' is updated according to Bayes' Rule:

$$b_{t+1}^{ao}(s') = \frac{O(s',a,o)\sum_{s\in\mathcal{S}} T(s,a,s')b_t(s)}{\sum_{s'\in\mathcal{S}} O(s',a,o)\sum_{s\in\mathcal{S}} T(s,a,s')b_t(s)} \tag{2.1.8}$$

and a new belief vector b_1 is formed by updating $b_0(s')$ for all new states s'.

This belief vector is a sufficient statistic for a POMDP [17]. A policy in a POMDP thus maps beliefs to actions. As such, we can model a POMDP as a continuous, belief-state MDP, where the beliefs form the states. To do so, we must specify the transition and reward function for beliefs (instead of states, as in an MDP), which we can do as follows:

$$\begin{aligned} T(b,a,b') &= \sum_{s'\in\mathcal{S}} O(s',a,o)\sum_{s\in\mathcal{S}} T(s,a,s')b(s) \\ R(b,a) &= \sum_{s\in\mathcal{S}} b(s)R(s,a) \end{aligned} \tag{2.1.9}$$

2.2 Multi-Agent Decision-Theoretic Planning

Both an MDP and a POMDP can be used to model environments with a single agent. Naturally, similar frameworks have been established for environments with multiple agents. In these, an important distinction must be made between the information available to each agent. This Section is largely based on Oliehoek et al. [18].

2.2.1 Multi-Agent Markov Decision Process

First, let us suppose the environment is fully observable for each agent. That is, each agent can directly observe the state, as well as the actions of other agents. Such an environment is closely related to an MDP and is called a Multiagent Markov Decision Process (MMDP) [19].

Definition 2.2.1. An MMDP is a tuple $\langle \mathcal{D}, \mathcal{S}, \mathcal{A}, T, R \rangle$, where

- $\mathcal{D} = \{1, \ldots, n\}$ is the set of agents.
- $\mathcal{S}$ is as in an MDP.
- $\mathcal{A}$ is the set of joint actions.
- T and R are the transition and reward function resp., which are now dependent on joint actions.

In an MMDP, at each timestep t, each agent i fully knows the state of the environment and chooses an action a_t^i from the actions available to it, denoted by $\mathcal{A}^i$. The joint action $a_t = \langle a_t^1, \ldots, a_t^n \rangle$ is formed and executed, and the state transitions according to the transition function $T(s_t, a_t, s_{t+1})$. A reward $R(s_t, a_t)$ is obtained, and the process repeats.

Since the state is directly observable to each agent, they make decisions based on the same information. Therefore, instead of seeing this as a multi-agent environment, one can think of it as a single agent MDP in which an imaginary agent, commonly referred to as a puppeteer agent, employs a policy that maps states to joint actions. This joint policy specifies a joint action for each of the states the environment can be in, and can then be decomposed into an individual policy for each agent. In other words, although an MMDP is used to model an MDP with multiple agents, conceptually we can still see it as a single-agent MDP. The main limitation is that the joint action space scales exponentially in the number of agents.

2.2.2 Multi-Agent Partially Observable Markov Decision Process

Similarly to the extension of an MDP to a Multi-Agent MDP, the Multi-Agent POMDP extends the single agent POMDP, when the state is not directly observable to the agents. Importantly, in an MPOMDP the agents are allowed to freely share their individual actions and observations.

Definition 2.2.2. An MPOMDP is a tuple $\langle \mathcal{D}, \mathcal{S}, \mathcal{A}, T, R, \Omega, O, b_0 \rangle$, where

- $\mathcal{D}$, $\mathcal{S}$, $\mathcal{A}$, T, and R are as in an MMDP.
- b_0 is as in a POMDP.
- Ω is the set of joint observations.

- O is the observation function, dependent on joint actions.

Again, since each agent makes decisions based on the same information, we can make use of the puppeteer agent to approach the underlying POMDP. Similarly to how we computed a belief in a POMDP, we now compute a joint belief over the states. Initially, this is specified by b_0, the initial state distribution. After taking a joint action a from state s and receiving a joint observation o, we update the belief using Bayes' Rule as follows:

$$b_{t+1}^{ao}(s') = \frac{O(s', a, o) \sum_{s \in \mathcal{S}} T(s, a, s') b_t(s)}{\sum_{s' \in \mathcal{S}} O(s', a, o) \sum_{s \in \mathcal{S}} T(s, a, s') b_t(s)} \tag{2.2.1}$$

Note the similarity between this update and the corresponding update in a POMDP (Eq. 2.1.8), the only difference being that we are now dealing with joint actions and observations. A policy in an MPOMDP then maps joint beliefs to joint actions, and can be decomposed into individual policies for each agent, which map joint beliefs to individual actions. Since each agent has access to the joint actions and observations in an MPOMDP, they can compute this joint belief consistently.

2.2.3 Decentralized Partially Observable Markov Decision Process

Thus far we have considered multi-agent environments which can be conceptually reformulated as single agent environments. However, such a reduction is not possible when the agents are not allowed to share their local information, as the puppeteer no longer has access to all information. Instead, agents must then truly act in a decentralized fashion, based on their own, local information.

Framework. To model such environments in which multiple agents collaborate based on local information, the Decentralized-Partially Observable Markov Decision Process (Dec-POMDP) framework, first described by Bernstein et al. [20], was established.

Definition 2.2.3. A Dec-POMDP is a tuple $\langle \mathcal{D}, \mathcal{S}, \mathcal{A}, T, R, \Omega, O, b_0 \rangle$, where

- $\mathcal{D}$, $\mathcal{S}$, T, R, Ω, O and b_0 are as in an MPOMDP.

The underlying environment dynamics in a Dec-POMDP are the same as in an MPOMDP. Rather, it differs in its agent model. To be more specific, in a Dec-POMDP, each agent only observes its own action a_t^i and does not observe the actions of other agents. Similarly, an agent only receives its own component of the joint observation, o_t^i, and does not receive the observations of other agents. As a result, each agent has access to some local information only, contrary to the sharing of information discussed previously.

We have previously introduced a sufficient statistic for both an MDP and a POMDP (which can be extended to an MMDP and MPOMDP), namely the current state and the belief respectively. Using these, an agent does not have to memorize its entire history of observations and actions. Does a Dec-POMDP have a similar sufficient statistic available to the agents during execution? One can easily see how both an MDP's and a POMDP's sufficient statistic are not applicable to Dec-POMDPs: the state is not directly observable, and an agent can not compute a belief over states as in an MPOMDP. To see why, recall that the state and observation functions used in the MPOMDP belief update (Eq. 2.2.1), depend on joint actions, which the agent does not know during execution. It only knows its own action and observation. As a result, to date, there is no known sufficient statistic of an agent's individual history in a Dec-POMDP during execution. In order to act optimally, an agent thus needs to remember its entire history of observations (and possibly actions[1]) .

General Definitions. Here, we shall also take the opportunity to formally introduce general definitions which will be used frequently throughout this Thesis.

Definition 2.2.4. The action-observation history (AOH) of agent i up until timestep t is defined as

$$\overline{\theta}^i_t = \langle a^i_0, o^i_1, a^i_1, o^i_2 \ldots, a^i_{t-1}, o^i_t \rangle \in \overline{\Theta}^i_t. \tag{2.2.2}$$

Definition 2.2.5. A stochastic decision rule for agent i at timestep t maps every possible action-observation history up until timestep t to a distribution over individual actions, i.e.

$$\delta^i_t : \overline{\Theta}^i_t \rightarrow \Delta\mathcal{A}^i. \tag{2.2.3}$$

A decision rule thus contains precisely the information an agent needs during execution: for each possible action-observation history it can experience, the decision rule prescribes the correct (distribution over) actions. An agent's policy can then be represented as a sequence of decision rules, one for each timestep.

Definition 2.2.6. A horizon-h policy is a chronologically ordered sequence of decision rules

$$\pi^i = \langle \delta^i_0, \delta^i_1, \ldots, \delta^i_{h-1} \rangle. \tag{2.2.4}$$

In other words, at each timestep the agent looks up its current AOH in the respective decision rule, and takes the action prescribed by it.

Since we are dealing with multiple agents in a Dec-POMDP, it will be useful to introduce joint definitions as well.

Definition 2.2.7. The joint action-observation history up until timetep t is defined as

$$\overline{\theta}_t = \langle a_0, o_1, a_1, o_2, \ldots, a_{t-1}, o_t \rangle. \tag{2.2.5}$$

Note that there is no longer an agent superscript.

[1]We specify what we mean with 'possibly' shortly.

Definition 2.2.8. A joint decision rule is the concatenation of each agent's individual decision rule.

$$\delta_t = \langle \delta_t^1, \delta_t^2, \ldots \delta_t^n \rangle. \tag{2.2.6}$$

Given the joint action-observation history, the joint decision rule prescribes the joint action.

Definition 2.2.9. The history of joint decision rules at timestep t is defined as

$$\varphi_t = \langle \delta_0, \delta_1, \ldots, \delta_{t-1} \rangle. \tag{2.2.7}$$

.

Another way to think of φ_t is as a partly specified joint policy, up until timestep $t-1$.

Deterministic decision rules. In the general case when agents employ stochastic decision rules (and thus stochastic policies), indeed there is no known summary of the individual action-observation history. However, if we restrict ourselves to deterministic decision rules, it is sufficient to remember only the observation history $\overline{o}_t^i$.

Definition 2.2.10. The individual observation history of agent i at timestep t is defined as

$$\overline{o}_t^i = \langle o_1^i, o_2^i, \ldots, o_t^i \rangle. \tag{2.2.8}$$

The joint observation history follows as before.

To see why the observation history is sufficient, suppose deterministic decision rules would map from action-observation histories to actions (as before). Consider the policy depicted in Fig. 2.1. Initially, assume the agent always takes action a_0 (colored green). Therefore, the bottom subtree corresponding to action $\dot{a}_0$ becomes unreachable. As a result, in the next timestep the agent can infer in which action-observation history it is in based solely on the observation history. If it observed o_1, it must be in AOH $\langle a_0, o_1 \rangle$, and similarly it must be in AOH $\langle a_0, \dot{o}_1 \rangle$ after observing $\dot{o}_1$. Finally, in the leaf nodes, again the agent can infer which one it is currently in based solely on the observation history. For example, if an agent remembers its observations $\overline{o}_t^i = \langle o_1, \dot{o}_2 \rangle$, it must be in AOH $\overline{\theta}_t^i = \langle a_0, o_1, \dot{a}_1, \dot{o}_2 \rangle$.

For deterministic decision rules, the action-observation history can thus be reconstructed from the observation history, for a fixed policy. The OH thus replaces the AOH in the formulations above. It has already been shown that a finite-horizon Dec-POMDP has at least one optimal deterministic joint policy [21].

Policies. For ease of exposition, we restrict ourselves to deterministic decision rules here. At each timestep t, there are $\mathcal{O}\left(|\Omega|^t\right)$ individual observation histories. An individual decision rule maps each of these to a possible action, hence

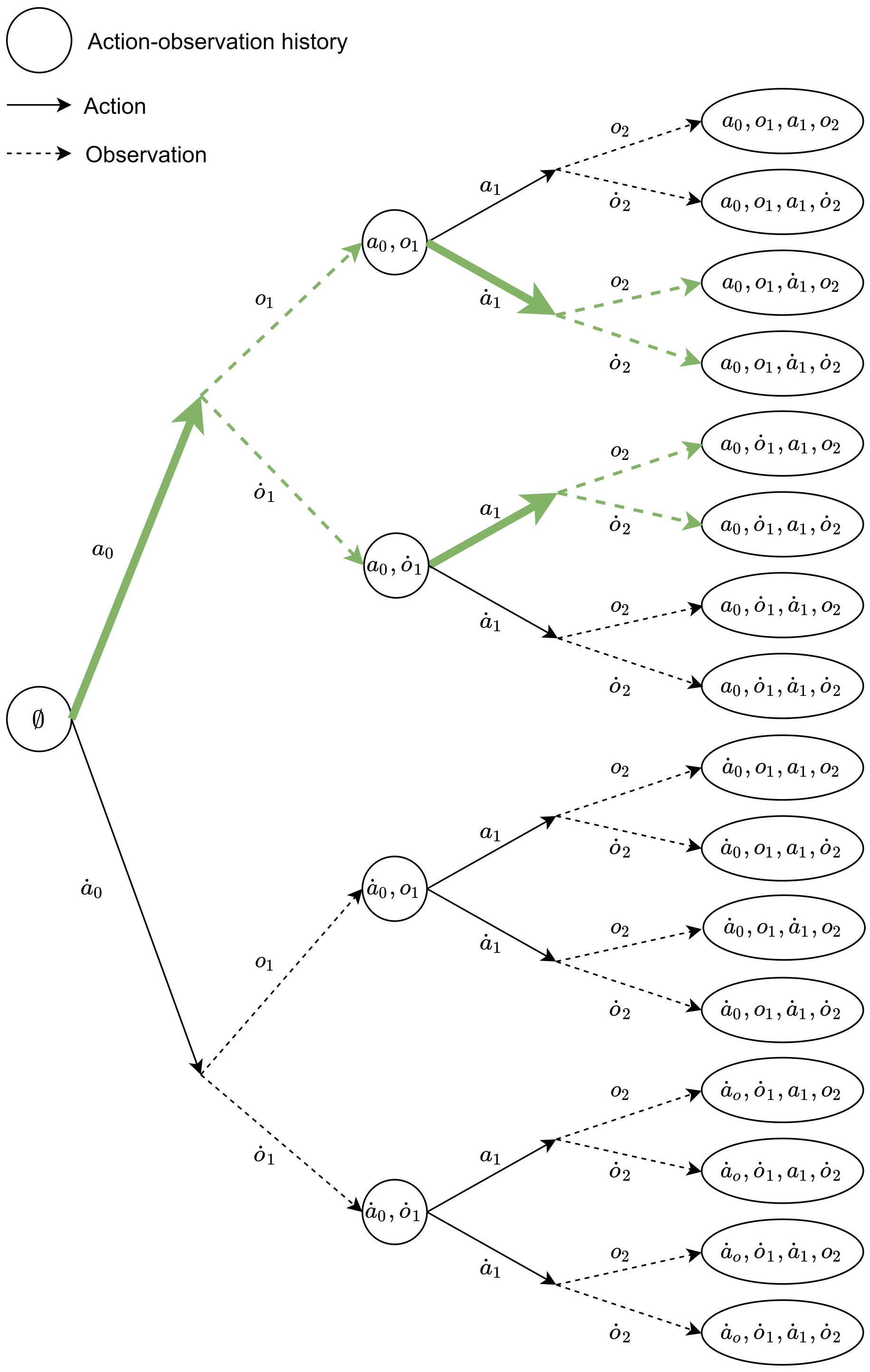

Action-observation history
Action
Observation
$\emptyset$
a_0
$\dot{a}_0$
o_1
$\dot{o}_1$
a_0, o_1
$a_0, \dot{o}_1$
$\dot{a}_0, o_1$
$\dot{a}_0, \dot{o}_1$
a_1
$\dot{a}_1$
o_2
$\dot{o}_2$
a_0, o_1, a_1, o_2
$a_0, o_1, a_1, \dot{o}_2$
$a_0, o_1, \dot{a}_1, o_2$
$a_0, o_1, \dot{a}_1, \dot{o}_2$
$a_0, \dot{o}_1, a_1, o_2$
$a_0, \dot{o}_1, a_1, \dot{o}_2$
$a_0, \dot{o}_1, \dot{a}_1, o_2$
$a_0, \dot{o}_1, \dot{a}_1, \dot{o}_2$
$\dot{a}_0, o_1, a_1, o_2$
$\dot{a}_0, o_1, a_1, \dot{o}_2$
$\dot{a}_0, o_1, \dot{a}_1, o_2$
$\dot{a}_0, o_1, \dot{a}_1, \dot{o}_2$
$\dot{a}_o, \dot{o}_1, a_1, o_2$
$\dot{a}_o, \dot{o}_1, a_1, \dot{o}_2$
$\dot{a}_o, \dot{o}_1, \dot{a}_1, o_2$
$\dot{a}_o, \dot{o}_1, \dot{a}_1, \dot{o}_2$

Figure 2.1: A graphical example of a deterministic policy, which demonstrates that solely remembering one's observation history is sufficient. Arrows colored green represent the reachable subtrees, as prescribed by the policy.

there are $\mathcal{O}\left(\mathcal{A}^{|\Omega|^t}\right)$ distinct individual decision rules, and $\mathcal{O}\left(\mathcal{A}^{|\Omega|^{nt}}\right)$ joint decision rules. Recall that a policy is a chronological sequence of such decision rules. In order to compute the value of such a policy, we must compute the value of each (state, joint OH)-pair, under this policy. We know that for the last timestep, the value is the immediate reward:

$$V^\pi(s_{h-1}, \overline{o}_{h-1}) = R(s_{h-1}, \pi(\overline{o}_{h-1})). \tag{2.2.9}$$

For all other timesteps, the expected value is given by

$$V^\pi(s_t, \overline{o}_t) = R(s_t, \pi(\overline{o}_t)) + \sum_{s_{t+1}\in\mathcal{S}} \sum_{o_{t+1}\in\mathcal{O}} P(s_{t+1}, o_{t+1}|s_t, \pi(\overline{o}_t)) V^\pi(s_{t+1}, \overline{o}_{t+1}), \tag{2.2.10}$$

where $\overline{o}_{t+1} = \langle \overline{o}_t, o_{t+1}\rangle$. Finally, the value of the policy is found by weighting these pairs by the initial state distribution b_o:

$$V(\pi) = \sum_{s_o\in\mathcal{S}} b_0(s_0) V^\pi(s_0, \overline{o}_0). \tag{2.2.11}$$

2.3 Multi-Agent Decision-Theoretic Learning

When we do not have access to a model of the environment, we must instead learn from our interactions with the environment. For example, if the state, observation and reward function are unknown, one can not plan a joint policy in advance. In our Thesis, we focus on value-based learning algorithms, specifically those which learn the Q-values of actions.

2.3.1 Game Theoretic Definitions

Two definitions from the field of Game Theory are of particular interest to us.

Definition 2.3.1. (In cooperative settings) An individual policy π^i is called a best response to the policies of other agents, π^{-i}, if

$$V(\langle \pi^i, \pi^{-i}\rangle) \geq V(\langle \pi'^i, \pi^{-i}\rangle) \qquad \forall \pi'^i \in \Pi^i. \tag{2.3.1}$$

Definition 2.3.2. A joint policy π is in a (Nash) equilibrium if π^i is a best response to π^{-i} for all i.

In other words, if the expected reward can not be increased when any agent changes its policy *while the other agents keep theirs fixed*, the joint policy is in an equilibrium.

2.3.2 Independent Q-Learning

In a multi-agent MDP, one could apply conventional Q-Learning to the joint action space of the puppeteer described in 2.2.1. However, the scalability of this approach is limited because the size of the joint action space increases exponentially with the number of agents. Alternatively, one can retain the individual perspective, and have each agent optimize its individual Q-function instead.

Independent Q-Learning (IQL) [22] uses the Q-Learning algorithm [23][24] to learn their individual Q-values in a Multi-Agent MDP. More specifically, each agent i maintains an estimate of $Q^i(s_t, a_t^i)$ as

$$Q^{i,\text{new}}(s_t, a_t^i) = (1-\lambda)\ Q^{i,\text{old}}(s_t, a_t^i) + \lambda \left[r_t + \gamma \max_{a_{t+1}^i \in \mathcal{A}^i} Q^i(s_{t+1}, a_{t+1}^i) \right], \quad (2.3.2)$$

where $0 \leq \lambda \leq 1$, commonly referred to as the learning rate, controls the trade-off between old information and newly acquired information. After learning has concluded, each agent chooses actions based on its individual Q-function, i.e. by acting greedily:

$$\pi^i(s) = \arg\max_a Q^i(s, a),\ \forall s \in \mathcal{S}. \quad (2.3.3)$$

IQL thus gets around the exponential scale-up in the number of agents by considering individual actions only. This does, however, come with numerous drawbacks. Since each agent only learns its individual Q-function, they are essentially completely oblivious to the presence of other agents, which are learning simultaneously. Since the state transition and reward function are dependent on joint actions, individual Q-values $Q^i(s, a^i)$ are insufficient to capture this dependency on other agents.

Convergence. In the cooperative environments that are of interest to us, Independent Q-Learners are not guaranteed to converge to the optimal joint policy. To see why, consider a very simple one-shot environment with a single state and two actions, a and $\dot{a}$, whose payoff are as follows.

	a^2	$\dot{a}^2$
a^1	10	0
$\dot{a}^1$	0	10

Suppose agents take actions uniformly random during learning. In theory, both agents will learn the value of either action to be 5 on average. In practice, due to the inherent stochasticity of selecting actions during learning, each agent might slightly prefer one action over the other. However, there is no guarantee that both agents will prefer a coordinated action.

Claus et al. [25] therefore suggest agents to follow an exploitative exploration strategy instead, in which an agent selects its best action with a higher probability than random actions. In doing so, they show that IQL almost surely converges to equilibria, under the conditions that

- The learning rate decreases over time, such that $\sum_{\lambda=0}^{t} \lambda = \infty$ and $\sum_{\lambda=0}^{t} \lambda^2 < \infty$.
- Each agent samples each of its actions infinitely often. (Both of these are required for single agent Q-Learning to converge as well.)
- Each agent's exploration strategy is exploitative, with the probability of exploration decreasing over time: $\lim_{t \to \infty} P(a_t^i) = 0$, if $a_t^i \neq \arg\max_a Q^i(s, a)$.

Intuitively, if agents are in an equilibrium, they are likely to remain there due to the self-confirming nature of equilibria. Agents can only escape it when both agents explore (favorably) at the same time. The probability of joint exploration, and thereby escaping equilibria, diminishes over time, as by the third condition. Furthermore, when agents are in a non-equilibrium, when one agent explores it will find another action to be more favorable, *unless* the other agent is exploring simultaneously. Again, this probability diminishes over time. Although IQL is thus likely to converge to equilibria, there is, however, no guarantee about the *quality* of such an equilibrium. It does not have to be the optimal equilibrium. We refer to [25] for one such example.

2.3.3 Deep Q Network

Tabular Q-Learning learns $Q(s, a)$ for each (state, action)-pair. When the state space becomes too large, typically a function approximator is used to generalize across the state space. One particularly succesful function approximator is the Deep Q-Network (DQN) [5], which uses a neural network as a function approximator to generalize Q-values across the input space. It aims to learn $Q(s, a; \boldsymbol{\theta}_t)$, where $\boldsymbol{\theta}_t$ are the network parameters at training iteration t [2]. Instead of directly updating Q-values, as in tabular Q-Learning, updates are now made to the parameters of the network to minimize a differentiable loss function. The loss function for iteration t is given by

$$\begin{aligned} L_t(\boldsymbol{\theta}_t) &= \mathbb{E}_{s,a,r,s'} \left[(y_t - Q(s, a; \boldsymbol{\theta}_t))^2 \right] \\ \text{where } y_t &= r + \gamma \max_{a'} \hat{Q}(s', a'; \boldsymbol{\theta}). \end{aligned} \tag{2.3.4}$$

A major contributor to the stability of the training process is the Experience Replay [26]. In supervised Deep Learning, training samples are commonly assumed to be independent and identically distributed. In Deep Reinforcement Learning however, consecutive states we experience throughout a trajectory are highly correlated. Moreover, supervised Deep Learning assumes the underlying data distribution to be stationary, whereas in Deep Reinforcement Learning the samples we experience are highly correlated with the current network parameters. To overcome the instability this introduces to the learning process, a replay memory D is used. A newly observed transition tuple $\langle s_t, a_t, r_t, s_{t+1} \rangle$ is stored

[2] In the remainder of this Thesis, we drop the notation for this dependency on $\boldsymbol{\theta}_t$ for clarity; It is clear from the context of DQNs, and we do not want to overload the symbol θ.

in this memory (up to a maximum in FIFO fashion). Instead of updating the network's parameters $\boldsymbol{\theta}$ sequentially with each newly observed transition tuple, at each iteration we randomly sample a minibatch of transitions from D and update by performing stochastic gradient descent on this minibatch.

A second contributor to the stability of the method is the use of a separate network $\hat{Q}$ for generating the targets y_t. Initially, the agent network and the target network share their parameters. The target network's parameters are then held fixed for C iterations, and copied from the agent network afterwards. Temporarily fixing the target network thus prevents the agent network from prematurely overfitting to current experiences.

For completeness, we present the full algorithm for DQN in Alg. 1. As a final remark, we want to emphasize that the Deep Q-Network by itself is not a way to achieve multi-agent learning. Instead, Indepedent Q-Learners can use a Deep Q-Network to learn their individual Q-values, when tabular Q-Learning falls short.

Algorithm 1: Deep Q-Learning, adapted from [5]

```
1  Initialize replay memory $D$ to capacity $N$
2  Initialize action-value function $Q(s, a; \boldsymbol{\theta})$ with random weights $\boldsymbol{\theta}$
3  Initialize target action-value function $\hat{Q}(s, a; \overline{\boldsymbol{\theta}})$ with weights $\overline{\boldsymbol{\theta}} = \boldsymbol{\theta}$
4  for episode $= 0$ to $M$ do
5      for $t = 0$ to $h$ do
6          With probability $\varepsilon$ select random action $a_t$
7          otherwise select $a_t = \arg\max_a Q(s_t, a; \boldsymbol{\theta})$
8          Execute action $a_t$ and observe reward $r_t$ and state $s_{t+1}$
9          Store transition $\langle s_t, a_t, r_t, s_{t+1} \rangle$ in $D$
                                                  // Experience Replay
10         Sample random minibatch of transitions $\langle s_j, a_j, r_j, s_{j+1} \rangle \in D$
11         if episode terminates at step $j + 1$ then
12             $y_j = r_j$
13         end
14         else
15             $y_j = r_j + \gamma \max_{a'} \hat{Q}(s_{j+1}, a'; \overline{\boldsymbol{\theta}})$
16         end
17         Perform gradient descent step on $(y_j - Q(s_j, a_j, \boldsymbol{\theta}))^2$ with
             respect to $\boldsymbol{\theta}$
18         Every C steps update $\hat{Q} = Q$            // Target Network
19     end
20 end
```

2.3.4 Centralized Learning in Dec-POMDPs

In Dec-POMDPs, during execution the agents must act based on local information, in a decentralized fashion. A key insight, however, is that learning does not need to be decentralized, but can instead be centralized. This paradigm is formalized as Centralized Learning with Decentralized Execution [21][27], under which agents can freely share observations and actions, and they can freely access the true (hidden) state during learning. Kraemer et al. [27] give a helpful intuition behind this approach: during rehearsal (learning), actors can read the script, discuss with others and receive feedback from the director. During the show (execution), they are on their own.

Depending on the amount of hidden information we reveal to the agents, **during learning** a Dec-POMDP can accordingly be reduced to either a Multi-Agent MDP or Multi-Agent POMDP. In the case that we reveal both the hidden state as well as the actions of other agents to each agent, for example, the Dec-POMDP is reduced to a Multi-Agent MDP. The optimal Q-values, $Q^{MMDP}(s,a)$ of joint actions in the underlying MMDP can then be learned. The main challenge is how decentralized policies can be extracted from Q^{MMDP}, as during execution the agents do not have access to the state and actions of other agents. Instead, they only know their own action-observation history $\overline{\theta}_t^i$.

One way to make progress is for agents to maintain an individual belief over precisely those hidden features. Kraemer et al. [27], for example, employ a frequency analysis during learning to compute $\hat{P}^i(s|\overline{\theta}_t^i)$ and $\hat{P}^i(a^{-i}|s,\overline{\theta}_t^i)$, and subsequently extract individual Q-functions from the centralized Q-function as

$$Q^i(\overline{\theta}_t^i, a^i) = \sum_{s,a^{-i}} \hat{P}^i(a^{-i}|s,\overline{\theta}_t^i)\hat{P}^i(s|\overline{\theta}_t^i)Q^{MMDP}(s,\langle a^i, a^{-i}\rangle). \tag{2.3.5}$$

Chapter 3

Dec-POMDPs as Non-Observable MDPs

In this Chapter, we recite the a key concept underpinning our research, namely a sufficient statistic of the past joint policy. It was introduced by Oliehoek [28], whose exposition we shall adhere to. Furthermore, we recite work on a reformulation of a Dec-POMDP as a Non-Observable MDP, which introduces a perspective on Dec-POMDPs highly relevant to our work. Finally, we discuss how Q-Learning could be applied to Non-Observable MDPs. Even though Sections 3.1 and 3.2 can be seen as Background, we want to explicitly discuss them separately due to their significance.

3.1 Sufficient plan-time statistics for Dec-POMDPs

Optimal value function of a Dec-POMDP. Recall that the goal of solving a Dec-POMDP is to find a sequence of joint decision rules $\langle \delta_0, \delta_1, \ldots, \delta_{h-1} \rangle$ which, when executed, maximizes the expected return. But a key question we have delayed in answering thus far is how we should compute the expected value of a joint decision rule, such that we can select the maximizing one. It should be clear that its value depends on the history of joint decision rules φ_t, as these greatly impact the underlying state and joint action-observation history distribution at the current timestep. With this in the back of our mind, we now present the optimal value function for a Dec-POMDP [21], which explicitly conditions on the initial state distribution b_0.

$$Q_t(b_0, \varphi_t, \overline{\theta}_t, \delta_t) = R(\overline{\theta}_t, \delta_t) + \sum_{a_t} \sum_{o_{t+1}} P(\overline{\theta}_{t+1} | \overline{\theta}_t, \delta_t)\, Q_{t+1}(b_0, \varphi_{t+1}, \overline{\theta}_{t+1}, \delta^*_{t+1}). \tag{3.1.1}$$

There are some terms on the right-hand side that warrant additional explanation. The first, the immediate expected reward $R(\overline{\theta}_t, \delta_t)$ of taking the joint decision rule, is fairly straightforward:

$$R(\overline{\theta}_t, \delta_t) = \sum_{s_t} P(s_t|b_0, \overline{\theta}_t) \sum_{a_t} R(s_t, a_t)\, \delta_t(a_t|\overline{\theta}_t), \tag{3.1.2}$$

where $\delta_t(a_t|\overline{\theta}_t)$ denotes the probability with which joint action a_t is taken under the decision rule. As for the second term, $P(\overline{\theta}_{t+1}|\overline{\theta}_t, \delta_t)$, we let $\overline{\theta}_{t+1} = \langle \overline{\theta}_t, a_t, o_{t+1} \rangle$ (i.e. the addition of the current joint action and joint observation to the joint action-observation history). We get

$$P(\overline{\theta}_{t+1}|\overline{\theta}_t, \delta_t) = \sum_{s_t} P(s_t|b_0, \overline{\theta}_t) \sum_{s_{t+1}} P(o_{t+1}|a_t, s_{t+1}) P(s_{t+1}|s_t, a_t) \delta_t(a_t|\overline{\theta}_t). \tag{3.1.3}$$

In both Eq. 3.1.2 and 3.1.3 we require the state belief $P(s_t|b_0, \overline{\theta}_t)$, which we can compute in similar recursive fashion to the Bayesian belief in a POMDP, except that we are now dealing with joint actions and joint observations:

$$P(s_t|b_0, \overline{\theta}_t) = \frac{P(o_t|a_{t-1}, s_t) \sum_{s_{t-1}} P(s_t|s_{t-1}, a_{t-1}) P(s_{t-1}|b_0, \overline{\theta}_{t-1})}{\sum_{s_t} P(o_t|a_{t-1}, s_t) \sum_{s_{t-1}} P(s_t|s_{t-1}, a_{t-1}) P(s_{t-1}|b_0, \overline{\theta}_{t-1})}. \tag{3.1.4}$$

In these, again a_{t-1} and o_t stem from $\overline{\theta}_t$.

The optimal decision rule δ^* can then be found as:

$$Q_t(b_0, \varphi_t, \delta_t) = \sum_{\overline{\theta}_t} P(\overline{\theta}_t|b_0, \varphi_t) Q_t(b_0, \varphi_t, \overline{\theta}_t, \delta_t) \tag{3.1.5}$$

$$\delta_t^* = \arg\max_{\delta_t} Q_t(b_0, \varphi_t, \delta_t). \tag{3.1.6}$$

It is important to note that in order to perform the maximization over $Q_t(b_0, \varphi_t, \delta_t)$ in Eq. 3.1.6, we require $P(\overline{\theta}_t|b_0, \varphi_t)$, the probability of a particular joint action-observation history occurring given the partly specified policy thus far. Since each φ_t might induce a different distribution over $\overline{\theta}_t$, as a result we must re-evaluate these equations for each such φ_t.

Sufficient statistic for general policies. It turns out that the above equations can be reformulated to no longer depend on the history of joint decision rules φ_t, but on a sufficient statistic σ_t that summarizes it instead. The main motivation is that many φ_t might correspond to the same statistic, thus allowing for a more compact representation of the optimal value function.

Definition 3.1.1. The sufficient statistic of the past joint policy for general policies is a distribution over joint action-observation histories

$$\sigma_t(\overline{\theta}_t) = P(\overline{\theta}_t|b_0, \varphi_t). \tag{3.1.7}$$

In other words, it precisely contains the information for which φ_t was used previously. The sufficient statistic thus summarizes the entire past joint policy. We can update σ_t at each timestep as follows:

$$\sigma_{t+1}(\overline{\theta}_{t+1}) = P(o_{t+1}|\overline{\theta}_t, a_t)\, \delta_t(a_t|\overline{\theta}_t)\, \sigma_t(\overline{\theta}_t), \tag{3.1.8}$$

where $\overline{\theta}_{t+1} = \langle \overline{\theta}_t, a_t, o_{t+1} \rangle$, and

$$P(o_{t+1}|\overline{\theta}_t, a_t) = \sum_{s_t} P(s_t|b_0, \overline{\theta}_t) \sum_{s_{t+1}} P(o_{t+1}|a_t, s_{t+1}) P(s_{t+1}|s_t, a_t). \tag{3.1.9}$$

For completeness, we recite the reformulation of the optimal value function (Eq. 3.1.1) in terms of this sufficient statistic now. The optimal value function which conditions on σ_t is given as

$$Q_t(b_0, \sigma_t, \overline{\theta}_t, \delta_t) = R(\overline{\theta}_t, \delta_t) + \sum_{a_t} \sum_{o_{t+1}} P(\overline{\theta}_{t+1}|\overline{\theta}_t, \delta_t) Q_{t+1}(b_0, \sigma_{t+1}, \overline{\theta}_{t+1}, \delta^*_{t+1}). \tag{3.1.10}$$

Continuing the reformulation, optimal decision rules (Eq. 3.1.5, 3.1.6) can be formed as

$$Q_t(b_0, \sigma_t, \delta_t) = \sum_{\overline{\theta}_t} \sigma_t(\overline{\theta}_t) Q_t(b_0, \sigma_t, \overline{\theta}_t, \delta_t) \tag{3.1.11}$$

$$\delta^*_t = \arg\max_{\delta_t} Q_t(b_0, \sigma_t, \delta_t). \tag{3.1.12}$$

For a formal proof of the equivalence of both formulations, we refer to [28]. Intuitively, φ_t was used previously only to compute the probability distribution over joint action-observation histories $P(\overline{\theta_t}|b_0, \varphi_t)$ in Eq. 3.1.5. In these new formulations, σ_t represents exactly this probability distribution, and thus replaces the dependency on φ_t.

Including the state distribution in the sufficient statistic. Finally, note that in order to update σ_t, implicitly a Bayesian state-belief $P(s_t|b_0, \overline{\theta}_t)$ must still be maintained (Eq. 3.1.9). Instead, the state-belief can also be explicitly maintained in σ_t.

Definition 3.1.2. The sufficient statistic of a tuple (b_0, φ_t) with φ_t stochastic is a distribution over (state, joint action-observation histories) tuples

$$\sigma_t(s_t, \overline{\theta}_t) = P(s_t, \overline{\theta}_t|b_0, \varphi_t). \tag{3.1.13}$$

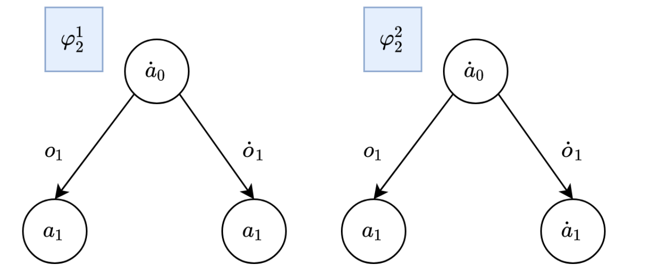
Initially,

$$\sigma_0(s_0, \overline{\theta}_0) = P(s_0, \emptyset) \sim b_0 \qquad \forall s_0. \tag{3.1.14}$$

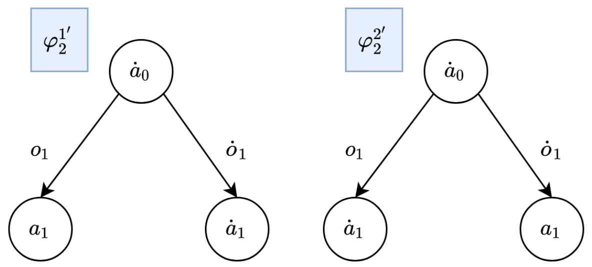
Subsequently, σ_t is updated as

$$\sigma_{t+1}(s_{t+1}, \overline{\theta}_{t+1}) = \sum_{s_t} P(s_{t+1}, o_{t+1}|s_t, a_t)\, \delta_t(a_t|\overline{\theta}_t)\, \sigma_t(s_t, \overline{\theta}_t). \tag{3.1.15}$$

Sufficient statistic for deterministic policies. Again, if we restrict ourselves to deterministic policies, we only need to consider (joint) observation histories instead of action-observation histories.



(a) Both φ_2 differ for $\overline{o}_1 = \dot{o}_1$, but result in the same σ_2 when o_1 is received with probability 1 after $\dot{a}_0$.



(b) Both φ_2 differ for any $\overline{o}_1$, but result in the same σ_2 when the observations are uniformly random.

Figure 3.1: Two examples where different φ_2 correspond to the same statistic $\sigma_2(s_2, \overline{o}_2)$ (depending on the environment).

Definition 3.1.3. The sufficient statistic of a tuple (b_0, φ_t) with φ_t deterministic is a distribution over (state, joint observation histories) tuples

$$\sigma_t(s_t, \overline{o}_t) = P(s_t, \overline{o}_t | b_0, \varphi_t). \tag{3.1.16}$$

As before, $\sigma_0 \sim b_0$. At each timestep, σ_t is updated as

$$\sigma_{t+1}(s_{t+1}, \overline{o}_{t+1}) = \sum_{s_t} P(s_{t+1}, o_{t+1} | s_t, \delta_t(\overline{o}_t))\, \sigma_t(s_t, \overline{o}_t). \tag{3.1.17}$$

Different φ_t correspond to the same σ_t. Let us consider deterministic decision rules here for ease of exposition. As was argued before, different φ_t might correspond to the same σ_t, and this new representation can thus be more compact. In other words, different histories of joint decision rules might induce the same probability distribution over (state, joint observation history) tuples. To see why, consider a simple environment with two joint observations $(o, \dot{o})$ and two joint actions $(a, \dot{a})$. Fig. 3.1 depicts two different partly specified policies. They differ in the joint action for joint observation history $\overline{o}_1 = \langle \dot{o}_1 \rangle$. φ^1 performs action a_1, whereas φ^2 performs action $\dot{a}_1$. However, suppose that action $\dot{a}_0$ deterministically leads to observation o_1. The right path will never be realized, and as a result both φ_2 induce the same distribution over joint observation histories $\sigma_2(s_2, \overline{o}_2)$. Similarly, if the observation is uniformly random, both $\varphi^{1\prime}$ and $\varphi^{2\prime}$ in Fig. 5.2b result in the same σ_2 as well.

3.2 Dec-POMDPs as Non-Observable MDPs

Oliehoek et al. [29] further show that using the sufficient statistic, we can reformulate a Dec-POMDP as a Non-Observable POMDP. We follow their exposition here.

One way to conceptually think about a Dec-POMDP is that a **single** decision maker needs to choose the joint decision rule at each timestep, based on the joint policy φ_t that was followed thus far. The sufficient statistic $\sigma_t(s_t, \overline{o}_t)$ summarizes precisely this past joint policy, and thus replaces the dependency on it. As such, we can reduce a Dec-POMDP to a single agent environment, a POMDP to be specific. In this POMDP, the states $\check{s}$ are formed by $\check{s}_t = \langle s_t, \overline{o}_t \rangle$. Moreover, the actions in this POMDP correspond to the joint decision rules in the underlying Dec-POMDP. Note that the true state is hidden. The decision maker does not know the true state and joint observation history, as agents are not allowed to share their information. Instead, the decision maker maintains a typical state-belief, which corresponds precisely to $\sigma_t(s_t, \overline{o}_t)$.

In this POMDP, the transition from one state $\check{s}_t$ to another $\check{s}_{t+1}$ is deterministic and can be expressed as

$$\check{T}(\langle s_t, \overline{o}_t \rangle, \delta_t, \langle s_{t+1}, \overline{o}_{t+1} \rangle) = \begin{cases} P(s_{t+1}, o_{t+1} | s_t, \delta_t(\overline{o}_t)) & \text{if } \overline{o}_{t+1} = \langle \overline{o}_t, o_{t+1} \rangle \\ 0 & \text{otherwise.} \end{cases} \tag{3.2.1}$$

Furthermore, we can define the reward function quite easily as

$$\check{R}(\langle s_t, \overline{o}_t \rangle, \delta_t) = R(s_t, \delta_t(\overline{o}_t)). \tag{3.2.2}$$

We are now ready to present the reformulation formally.

Definition 3.2.1. The plan-time Non-Observable MDP for a Dec-POMDP is a tuple $\langle \check{\mathcal{S}}, \check{\mathcal{A}}, \check{T}, \check{R}, \check{\Omega}, \check{O}, \check{b}_0 \rangle$ [29], where

- $\check{\mathcal{S}}$ is the set of augmented states, each $\check{s}_t = \langle s_t, \overline{o}_t \rangle$.
- $\check{\mathcal{A}}$ is the set of actions, whereby each action corresponds to a joint decision rule in the Dec-POMDP.
- $\check{T}$ is the transition function, defined in Eq. 3.2.1.
- $\check{R}$ is the reward function, defined in Eq. 3.2.2.
- $\check{\Omega} = \{NULL\}$ is the observation set which contains only the NULL observation.
- $\check{O}$ is the observation function, which specifies that the NULL observation is received with probability 1.
- $\check{b}_0$ is the initial state distribution in the Dec-POMDP, as follows from the definition of σ_0.

It might seem confusing that the observation space solely contains the NULL observation. However, recall that we are now taking the perspective of a (single) coordinator that decides on joint decision rules. Since individual agents can not share their information in a Dec-POMDP, such a coordinator can therefore not receive any observations. It also does not *need* any observations. Since the state transitions are deterministic, the coordinator knows precisely the belief σ_{t+1} after taking joint decision rule δ_t from belief σ_t. Therefore, given such a sequence of decision rules, the state-belief σ_t can be completely determined in advance for all timesteps.

3.3 Q-Learning in NOMDPs

In this Section, we show how Q-Learning can be applied to NOMDPs, and how, in theory, it converges to the optimal Q-values.

Naively, in order for the decision maker to select the right joint decision rule at teach timestep, one could employ Q-Learning to learn the value of each $Q(\check{s}_t, \delta_t)$ pair. However, these values are not directly useful. The underlying states $\check{s} = \langle s_t, \overline{o}_t \rangle$ are hidden; we are dealing with a POMDP, after all. Alternatively, like any POMDP, a NOMDP can be seen as a continuous-state MDP, in which the belief over $\check{s}$ forms the states. In other words, in this MDP, precisely $\sigma_t(s_t, \overline{o}_t)$ form the states. Q-learning could then be used to learn $Q(\sigma_t, \delta_t)$ values.

The problem, of course, is that the states σ_t lie in a continuum. The value $Q(\sigma_t, \delta_t)$ would, therefore, have to be learned for infinite amount of σ_t. However, one insight we can use is that in case of deterministic decision rules, only a subset of all σ_t are reachable. Recall that each partly specified policy φ_t implies a certain σ_t in a many-to-one fashion. The total number of partly specified deterministic policies is certainly finite at all timesteps. As a result, the subset of reachable σ_t must necessarily be finite at all timesteps as well.

One could construct the finite subset of reachable σ_t for all timesteps and subsequently, using Q-Learning, attempt to learn $Q(\sigma_t, \delta_t)$ for all such pairs as

$$Q^{\text{new}}(\sigma_t, \delta_t) = (1-\lambda)\; Q^{\text{old}}(\sigma_t, \delta_t) + \lambda \left[r_t + \gamma \max_{\delta_{t+1} \in \check{\mathcal{A}}} Q(\sigma_{t+1}, \delta_{t+1}) \right], \quad (3.3.1)$$

where σ_{t+1} is the result of the deterministic transition defined by Eq. 3.1.17 for all $\langle s_{t+1}, \overline{o}_{t+1} \rangle$ pairs. Since we are now dealing with an MDP (albeit, with an enormous state and action space), in theory these learned Q-values will converge to the optimal Q^*-values

$$Q^*(\sigma_t, \delta_t) = \begin{cases} R(\sigma_t, \delta_t) & \text{for the last stage } t = h-1 \\ R(\sigma_t, \delta_t) + \gamma \max_{\delta_{t+1}} Q^*(\sigma_{t+1}, \delta_{t+1}) & \text{otherwise,} \end{cases} \quad (3.3.2)$$

where

$$R(\sigma_t, \delta_t) = \sum_{s_t} \sum_{\overline{o}_t} \sigma_t(s_t, \overline{o}_t) R(s_t, \delta(\overline{o}_t)). \quad (3.3.3)$$

Of course, the necessary conditions for convergence in an MDP [30] must still apply.

Theorem 1. Q-Learning in a NOMDP over the subset of reachable σ_t converges to $Q^*(\sigma_t, \delta_t)$, under the same conditions that guarantee convergence of Q-Learning in an MDP.

In practice, however, this is not directly useful. The state and action space are simply too large. Even learning the value of joint decision rules for a single σ_t quickly becomes intractible, as there are $|\mathcal{A}|^{|\Omega|^{nt}}$ such joint decision rules.

Chapter 4

Our approach

In this Chapter, we present our approach to tackling Dec-POMDPs, which uses the sufficient plan-time statistic and is inspired by the NOMDP reformulation.

4.1 Towards feasible Q-Learning for NOMDPs

As discussed in Chapter 3, exact tabular Q-Learning for a NOMDP, which attempts to learn $Q(\sigma_t, \delta_t)$ for each such (reachable) tuple, is intractible due to both its enormous state and action space. In our approach, we therefore make an approximation with respect to both the state and the action space.

Addressing the state space. When dealing with a large state space, a common way to make progress is to use a function approximator. In our work, we employ the Deep Q-Network, which uses a neural network function approximator. As discussed in Section 2.3.3, such a neural network does not need to explicitly learn Q-values for each σ_t. Instead, it uses its function approximator to generalize Q-values across the state space.

Moreover, neural networks are well-equipped to deal with continuous state spaces, for precisely the reason described above. Since they no longer explicitly learn Q-values for each σ_t, but rather optimize their network weights in such a way that generalizes across the state space, states that lie in a continuum can thus be dealt with. As a result, we no longer have to compute the subset of reachable σ_t in advance.

Nevertheless, solely addressing the state space is insufficient, as optimizing joint decision rules still quickly becomes intractible.

Addressing the action space. Although the reformulation of a Dec-POMDP as a NOMDP in principle allows the application of single agent solution techniques, its main limitation is an explosion in the action space. The coordinator now decides on joint decision rules, which not only scale exponentially in the horizon, but also exponentially in the number of agents. A common way to address the exponential scale-up in the number of agents, as already discussed in Section 2.3.2, is to take the individual perspective. Instead of a coordinator learning the value of joint decision rules, in this individual perspective each agent concurrently learns the value of individual decision rules, i.e. $Q(\sigma_t, \delta_t^i)$.

There is one additional insight we can use to further reduce the size of the action space [31][32]. By taking the individual perspective, we can additionally condition the agents on their local information $\overline{\theta}_t^i$. In doing so, the DQN can now *implicitly* output an individual decision rule based on the sufficient statistic, i.e. $Q(\sigma_t, \delta_t^i)$ as before. Subsequently, it then *explicitly* outputs a single entry from this decision rule: precisely the one that maps its indiviudal action-observation history to an action. Each DQN thus learns $Q(\overline{\theta}_t^i, \sigma_t, a_t^i)$. In this way, we reduce the size of the action space by increasing the size of the state space. This is desirable, as neural networks have powerful abilities to generalize across the input space, but are limited in their ability to deal with a large output space.

Of course, the main drawback of such an individual Q-Learning approach is that agents now optimize their individual actions without taking the other agents into account. The value of an individual action clearly depends on the actions the other agents take. The individual $Q(\overline{\theta}_t^i, \sigma_t, a_t^i)$-values are therefore insufficient to capture this inter-agent dependency. However, contrary to *true* Independent Q-Learners (which would learn $Q(\overline{\theta}_t^i, a_t^i)$), our agents are aware of the presence of other agents by conditioning on the sufficient statistic. This sufficient statistic contains information about the decision rules that the other agents are following, as well as the resulting state distribution. Our main area of research therefore lies in investigating the effects of conditioning Independent Q-Learners on the sufficient statistic. We will discuss this more concretely when posing our research questions in Chapter 5.

A hybrid approach. Our approach can thus be seen as a hybrid between joint planning and individual learning. On the one hand, we let each agent optimize its own actions based on its local information. On the other hand, we aid the learning process by conditioning the agents on the sufficient statistic, which contains information about the joint state of the agents and the environment. Moreover, in order to compute σ_t, the agents require full knowledge of the transition and observation function, as opposed to *true* learning in which the environment dynamics are assumed to be wholly unknown.

Where joint planners optimize joint (deterministic) decision rules, of which at each timestep there are $\mathcal{O}\left(|\mathcal{A}|^{|\Omega|^{nt}}\right)$, we optimize individual actions, of which there are only $\mathcal{O}\left(|\mathcal{A}^i|\right)$. However, in order to do so, we condition on $\sigma_t(s_t, \overline{\theta}_t)$, whose size is $\mathcal{O}\left(|\mathcal{S}| \cdot (|\Omega| \cdot |\mathcal{A}|)^{nt}\right)$. Moreover, it lies in a continuum. This, in turn, is considerably worse than purely individual learners, which optimize individual actions by solely conditioning on their own action-observation history.

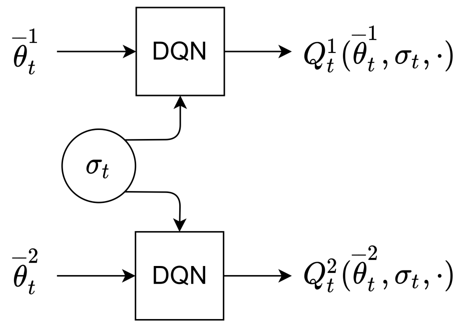

$\overline{\theta}_t^1$
DQN
$Q_t^1(\overline{\theta}_t^1, \sigma_t, \cdot)$
σ_t
$\overline{\theta}_t^2$
DQN
$Q_t^2(\overline{\theta}_t^2, \sigma_t, \cdot)$

Figure 4.1: A graphical representation of our approach. (Two) Individual learners take their own action-observation history as well as the sufficient statistic as input, and output the Q-values of each action. Separate DQNs are trained for each timestep, to deal with the increasing size of $\overline{\theta}_t$ over time.

4.2 Deep Q-Network specifics

We use a Deep Q-Network that learns $Q_t^i(\overline{\theta}_t^i, \sigma_t, a_t^i)$. Our learning setup, for two agents, is depicted in Fig, 4.1. There is one subtlety we have to address. The size of $\overline{\theta}_t^i$ grows every timestep, yet neural networks expect input size to remain constant. To get around this, we train a separate DQN $Q_t^i(\cdot)$ for each timestep (and for each agent). Normally, a DQN uses its own target network to predict the value of the next state (line 15 in Alg. 1). In our setup, the target network of the DQN at the next timestep, i.e. $\hat{Q}_{t+1}^i(\cdot)$, is used for this instead.

We thus define the loss function of agent i's DQN at timestep t to be

$$\begin{aligned} L &= \mathbb{E}\left[(y_t - Q_t^i(\overline{\theta}_t^i, \sigma_t, a_t^i))^2\right] \\ \text{where } y_t &= r + \gamma \max_{a'_{t+1}} \hat{Q}_{t+1}^i(\overline{\theta}_{t+1}^i, \sigma_{t+1}, a'_{t+1}), \end{aligned} \tag{4.2.1}$$

where the gradient of the loss is taken with respect to the weights of the respective DQN, as usual.

4.3 Computing σ_t: methodology and assumptions

How should the agents compute σ_t? Recall from Eq. 3.1.15 that if the joint decision rules δ_t, the state function and observation function are known, σ_t can be computed consistently and independently by each agent. However, there is a subtle problem here. In our setup, instead of joint decision rules, each agent outputs Q-values for individual actions. How, then, should the joint decision rules be recovered?

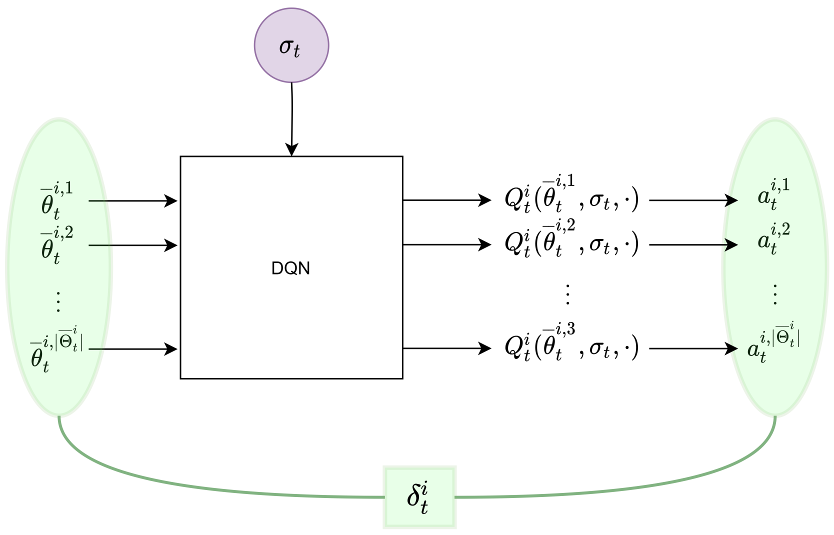

σ_t
DQN
$\overline{\theta}_t^{i,1}$
$\overline{\theta}_t^{i,2}$
$\overline{\theta}_t^{i,|\overline{\Theta}_t^i|}$
$Q_t^i(\overline{\theta}_t^{i,1}, \sigma_t, \cdot)$
$Q_t^i(\overline{\theta}_t^{i,2}, \sigma_t, \cdot)$
$Q_t^i(\overline{\theta}_t^{i,3}, \sigma_t, \cdot)$
$a_t^{i,1}$
$a_t^{i,2}$
$a_t^{i,|\overline{\Theta}_t^i|}$
δ_t^i

Figure 4.2: Recovering an explicit representation of δ_t^i from the learning algorithm, by quering every action-observation history. For ease of exposition, we consider a deterministic decision rule here. Stochastic decision rules would map every $Q(\cdot)$ to a probability distribution over actions instead. The first superscript index refers to the agent, the second to the AOH index.

The procedure to retrieve individual decision rules from a DQN is as follows. Given a specific individual action-observation history $\overline{\theta}_t^i$ and sufficient statistic σ_t, the network outputs Q-values for each action. Let us suppose the action corresponding to the highest Q-value is then chosen, i.e. we act greedily. We have then recovered one entry from the individual decision rule, namely $\delta_t^i(\overline{\theta}_t^i) = \arg\max_a Q_t^i(\overline{\theta}_t^i, \sigma_t, a)$. To recover an explicit representation of δ_t^i from the neural network, we thus need to present every possible AOH to it, and observe the resulting action. This procedure is depicted in Fig. 4.2. By repeating this process to recover the decision rule δ_t^i of each agent, the joint decision rule $\delta_t = \langle \delta_t^1, \ldots, \delta_t^n \rangle$ can then be recovered.

Assumptions. In order for each agent to be able to compute σ_t, we therefore make three main assumptions in this Thesis:

1. **The agents have full knowledge about the environment's state transition and observation function.**
2. **During learning, agents can freely access the neural networks of other agents.**
3. **During execution, the sequence of joint decision rules $\langle \delta_0, \delta_1, \ldots, \delta_{h-1} \rangle$ is known to each agent.**

We thus follow the Centralized Learning with Decentralized Execution paradigm. In particular, we do not reveal the state nor the actions of other agents during learning. Instead, we reveal the decision rules of the other agents during learning. If we indeed assume the agents are allowed to share their decision rules before execution (Ass. 3), then the Q-values optimized during learning can be directly used during execution, as there is no additional hidden information that Q conditions on. We therefore do not require to factorize or bootstrap from the learned Q-function.

Furthermore, note that the third assumption still allows for truly decentralized execution, as no information needs to be shared **during execution**. In essence, the agents simply agree on how they will act in any given situation beforehand (i.e. the decision rules), without actually sharing information while acting.

4.4 Algorithmic Properties

We present pseudocode for reconstructing the joint decision rule δ_t, as well as updating σ_t here. Furthermore, we analyze their respective complexity.

Reconstructing the joint decision rule. We have already delineated the procedure for reconstructing δ_t from individual DQNs. Pseudocode for this is shown in Alg. 2. For each agent, we need to present each possible individual action-observation history, alongside the current sufficient statistic, to it. Given these, we subsequently assign a probability to each individual action based on the exploration strategy of the agent. Since there are $\mathcal{O}\left((|\mathcal{A}^i| \cdot |\Omega^i|)^t\right)$ possible AOHs for each agent at timestep t, the complexity of reconstructing δ_t can thus be formulated as

$$\mathcal{O}\left(n(|\mathcal{A}^i| \cdot |\Omega^i|)^t \cdot |\mathcal{A}^i|\right). \tag{4.4.1}$$

Algorithm 2: `reconstruct_`$\delta_t(\sigma_t)$

```
1  for $i \in \{1, \dots, n\}$                                   // For each agent
2  do
3      foreach $\overline{\theta}^i_t \in \overline{\Theta}^i_t$          // For each possible indivdual
        action-observation history thus far.
4      do
5          foreach $a^i_t \in \mathcal{A}^i$                    // For each individual action
6          do
7              $\delta^i_t(a^i_t|\overline{\theta}^i_t) \leftarrow$ action_probability($Q^i_t(\overline{\theta}^i_t, \sigma_t, \cdot)$)
8          end
9      end
10 end
11 $\delta_t \leftarrow \langle \delta^1_t, \dots, \delta^n_t \rangle$
```

Updating the sufficient statistic. In order to construct σ_{t+1} from σ_t, first we have to reconstruct the joint decision rule, as explained above. Next, we must evaluate Eq. 3.1.15 for each new (state, joint AOH) pair. Given that we know $\delta_t(\cdot)$ and the state and observation function, evaluating each term in the right-hand side in Eq. 3.1.15 is a simple lookup in $\mathcal{O}(1)$. Since we have to evaluate the right hand side for each old state s_t, its complexity is $\mathcal{O}(|\mathcal{S}|)$. Combining the complexity of reconstructing δ_t, and subsequently evaluating Eq. 3.1.15 for each (state, joint AOH) pair, we can formulate the complexity of the update of σ_t as follows

$$\mathcal{O}\left(n(|\mathcal{A}^i| \cdot |\Omega^i|)^t \cdot |\mathcal{A}^i|\right) + \mathcal{O}\left(|\mathcal{S}| \cdot (|\mathcal{A}^*|) \cdot |\Omega^*|)^{n(t+1)} \cdot |\mathcal{S}|\right) = \mathcal{O}\left((|\mathcal{A}^*|) \cdot |\Omega^*|)^{n(t+1)} \cdot |\mathcal{S}|^2\right), \tag{4.4.2}$$

where the complexity of reconstructing the joint decision rule is outweighed for any problem with $|\mathcal{A}|$, $|\Omega|$, $n >= 2$. $\mathcal{A}^*$ resp. $\mathcal{O}^*$ refer to the largest individual action, observation space. Pseudocode for the procedure is shown in Alg. 3.

Algorithm 3: `update_sigma`(σ_t)

```
1 $\delta_t \leftarrow$ reconstruct_$\delta_t(\sigma_t)$
2 foreach $s_{t+1} \in \mathcal{S}$                          // For each new state
3 do
4     foreach $\overline{\theta}_{t+1} = \langle \overline{\theta}_t, a_t, o_{t+1} \rangle \in \Theta_{t+1}$      // For each possible
       next joint action-observation history
5     do
6         $\sigma_{t+1}(s_{t+1}, \overline{\theta}_{t+1}) \leftarrow \sum_{s_t} P(s_{t+1}, o_{t+1} | s_t, a_t)\, \delta_t(a_t | \overline{\theta}_t)\, \sigma_t(s_t, \overline{\theta}_t)$
           // Implements Eq.  3.1.15
7         $\sigma_{t+1}(s_{t+1}, \overline{\theta}_{t+1}) =$
8     end
9 end
```

4.5 Discussion on the action-observation history, potentially stochastic decision rules and σ

Let us finally address the elephant in the room: why do we bother with the action-observation history and possibly stochastic decision rules, when we know at least one optimal deterministic policy exists (which would require only the observation history)?

Action-Observation history. Recall that if the policy is fixed and deterministic, one can summarize the action-observation history with the observation history, as the AOH can be reconstructed from the OH. However, during *learning*, the policy is certainly not fixed. The reason we use the action-observation history is coupled to our use of Deep Q-Networks, which have difficulty with implicitly reconstructing the AOH from the OH. This is in part due to the fact that, while learning, such networks constantly change their policies. Consequently, at some point a specific OH could imply a different AOH than at a later point. This problem is brought to light when the DQN samples both old and new trajectories from its memory in a minibatch.

Moreover, exploratory actions, which are key to the learning process, are wholly uncaptured by the observation history. In other words, the network would have no way of differentiating between a greedy trajectory and a partly exploratory trajectory, given that it receives the same observations.

Deterministic or Stochastic decision rules? The use of deterministic or stochastic decision rules is tightly coupled to the exploration strategy, as exploration strategies precisely describe the probability with which a specific action is taken. Consider naive individual epsilon-greedy exploration, for example. Given a specific input, we can take each action with probability at least $\frac{\varepsilon}{|\mathcal{A}^i|}$. Therefore, it would require stochastic decision rules. In our experiments, we shall investigate various exploration strategies. We therefore defer their specifics, as well as their implications on the the decision rules, for our experimental Chapters.

On the sufficient statistic. Out of the presented sufficient plan-time sufficient statistics in Chapter 3, we use the statistic for general policies that includes the state distribution, i.e. $\sigma_t(s_t, \overline{\theta}^i_t)$. The reason we maintain a probability distribution over the joint action-observation history is logical given what we have just discussed. We explicitly include the state distribution in σ_t, as we hypothesize that this will contain relevant information for the learning process of our agents.

Chapter 5

Experimental Overview

In this Chapter, we begin by exactly defining our research questions and formally stating a hypothesis. Next, we describe our test environment, namely Dec-Tiger. Finally, we describe the experimental setup for all experiments that we will conduct.

5.1 Research Questions

In order to make Q-Learning in NOMDPs feasible, we make an approximation regarding the state space by using a neural network, and we make an approximation regarding the action space by taking the perspective of optimizing individual actions. In other words, instead of a coordinator learning $Q(\sigma_t, \delta_t)$ in a tabular fashion, we focus on Independent Q-Learners that condition on the sufficient statistic, and use a Deep Q-Network to learn $Q^i_t(\overline{\theta}^i_t, \sigma_t, a^i_t)$. Our primary interest in conducting our experiments is therefore to investigate whether such approximations are viable, or whether they completely hamper the learning process.

The main question we aim to investigate in this Thesis is as follows.

What are the effects of conditioning Independent Q-Learners in a Dec-POMDP on the Sufficient Statistic?

In particular, the following 'effects' will be analyzed in detail.

1. Can independent learners (more consistently) find the optimal joint policy by conditioning on the sufficient statistic?
2. (Related) Does conditioning on the sufficient statistic prevent premature convergence of independent learners to sub-optimal equilibria?
3. How do various exploration strategies relate to the decision rules, the sufficient statistic and the learning process?

4. How does the complexity of the Deep Q-Network influence the effects of conditioning the learners on the sufficient statistic?

The first and second question aim to address the shortcoming of conventional Independent Q-Learning, which is that they are prone to get stuck in sub-optimal equilibria. As for the third, we have already discussed how the decision rules are tightly interwoven with the exploration strategy. We will thoroughly investigate a number of exploration strategies and their respective effects. Finally, the fourth question covers the consequences of the high dimensionality of the sufficient statistic for the learning algorithm.

Baseline: True Indepedent Q-Learners. We will compare our learners that condition on the sufficient statistic to truly independent learners. These truly independent learners use a DQN to learn $Q_t^i(\overline{\theta}_t^i, a_t^i)$. That is, they take solely their own action-observation history as input, and learn the values of individual actions.

Hypothesis. In the remainder of this Thesis, we will refer to the learners that use the sufficient statistic as σ-DQN. We will refer to the truly independent learners as i-DQN.

We know that optimal decision making in a Dec-POMDP not only depends on your own actions, but also on the actions of all the other agents. Using the sufficient statistic, σ-DQN has knowledge about the distribution over joint action-observation histories, and they can therefore make better informed predictions about the actions other agents will take. In general, we therefore expect σ-DQN to learn better policies than i-DQN. Moreover, σ-DQNs can adjust their own actions according to the likely actions the other agents will take. Consequently, we expect σ-DQN to be able to escape sub-optimal equilibria more easily compared to i-DQN. Furthermore, we expect σ-DQN to require a more complex network architecture compared to i-DQN, as the size σ_t is orders of magnitude higher than a sole encoding of the action-observation history. Finally, we expect σ-DQN to require more training time until convergence, again because of the increased input space.

Regarding the exploration strategies, we initially expected naive epsilon greedy to suffice. However, as we can hopefully convince you, investigating various exploration strategies turned out to be one of the most interesting parts of our research. Without properly introducing these exploration strategies first, formally stating a hypothesis regarding them now is not particularly useful. We shall discuss exploration in great depth in our experimental Chapters.

5.2 Test Domain: Dec-Tiger

We shall now present the Dec-Tiger environment in which we will conduct our experiments.

The Dec-Tiger [1] environment is a Dec-POMDP in which the agents must learn to cooperate to find a treasure. The treasure is hidden behind one of two doors. Behind the other, a tiger awaits. At each timestep, the agents can choose to either Listen (**Li**), to Open the left door (**OL**) or to Open the right door (**OR**). If any agent opens the door to the tiger, both are punished (heavily). Similarly, if any agent opens the door to the treasure, the agents are positively rewarded, unless the other agent opened the door to the tiger. Additionally, agents are rewarded higher and punished lighter when jointly opening a door, thus stimulating coordination. If an agent chooses to listen, it can hear where the tiger is located with high probability.

To be specific, Dec-Tiger has properties $n = 2$, $|\mathcal{S}| = 2$, $|\mathcal{A}^i| = 3$ and $|\Omega^i| = 2$. The state transition function is quite simple, and is shown in Table 5.1, where s_l denotes the *tiger* being behind the left door. Initially, the state is uniformly random. When both agents choose to listen, the state remains the same. For any other joint action (that is, if any agent opens a door), the state is reset. Since execution is decentralized, if one agent listens and the other opens a door, the state will be reset unbeknownst to the first agent.

	$s_l \rightarrow s_l$	$s_l \rightarrow s_r$	$s_r \rightarrow s_l$	$s_r \rightarrow s_r$
$\langle a_{Li}, a_{Li} \rangle$	1	0	0	1
otherwise	0.5	0.5	0.5	0.5

Table 5.1: Transition probabilities. For any joint action that is not (Li, Li), the state resets without notifying the agents.

The individual observation function is quite simple as well (Tab. 5.2). If both agents listen, each agent can correctly hear the position of the tiger with probability 0.85. For any other joint action, the observation is uniformly random, since the state will be reset.

	s_l		s_r	
	o_{HL}	o_{HR}	o_{HL}	o_{HR}
$\langle a_{Li}, a_{Li} \rangle$	0.85	0.15	0.15	0.85
otherwise	0.5	0.5	0.5	0.5

Table 5.2: Individual observation probabilities. Only when both agents listen is the observation reliable.

Finally, the reward function is shown in Tab. 5.3. Indeed, the punishment for opening the wrong door is harsh. Note that if the agents open both doors, they are always punished for opening the door to the tiger. Moreover, the payoffs for jointly opening a door (as opposed to individually) stimulate coordination.

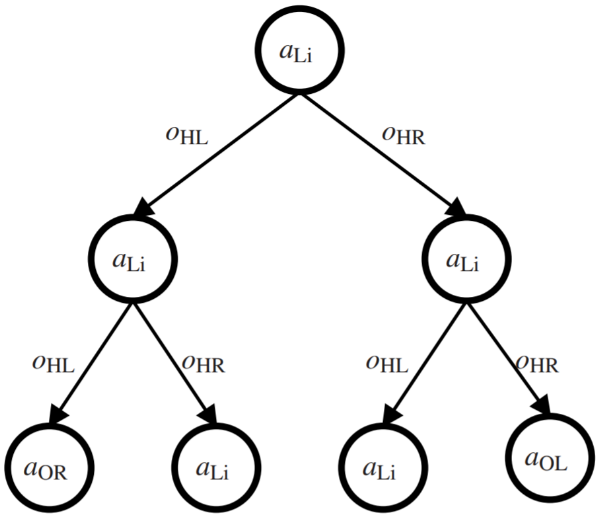


Figure 5.1: The optimal policy that both agents should follow in Dec-Tiger. Taken from [18].

	s_l	s_r
$\langle a_{Li}, a_{Li}\rangle$	-2	-2
$\langle a_{Li}, a_{OL}\rangle$	-101	+9
$\langle a_{Li}, a_{OR}\rangle$	+9	-101
$\langle a_{OL}, a_{Li}\rangle$	-101	+9
$\langle a_{OL}, a_{OL}\rangle$	-50	+20
$\langle a_{OL}, a_{OR}\rangle$	-100	-100
$\langle a_{OR}, a_{Li}\rangle$	+9	-101
$\langle a_{OR}, a_{OL}\rangle$	-100	-100
$\langle a_{OR}, a_{OR}\rangle$	+20	-50

Table 5.3: Reward function. Agents are rewarded higher / punished less severely when acting in accordance.

Optimal Policy. In our experiments, we let the agents act for three timesteps in total. Since the expected value of jointly opening a door without any prior information is -15, the optimal joint policy can be described as follows. During the first and second timestep, each agent should always listen in order to gain information about the position of the tiger. Finally, in the third timestep, an agent should open a door only if it heard the tiger behind the other door twice. Otherwise, it should simply listen again. The optimal individual policy is depicted in Fig. 5.1.

Sub-optimal equilibria. This environment is deceptively simple. The reason why it is quite difficult is that (potentially sub-optimal) equilibria exist at each

Learning Rate	0.001
γ	1
ε	0.1
Batch Size	50
Minimum Memory Size	5000
Maximum Memory Size	20000
Update Target Network every	10000
Optimizer	RMSprop

Table 5.4: Default DQN parameter settings.

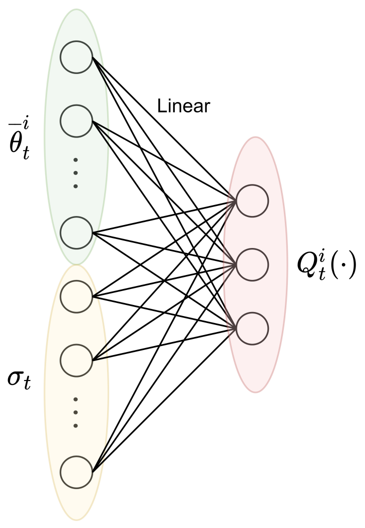
timestep. If both agents prefer to open the same door at any timestep, the expected reward can not be improved if a single agent deviates from the current plan. For example, if one agent decides to listen, the reward will always be lower, and the observation will be useless as the state will be reset by the other agent. The same holds for opening the other door.

This environment is therefore perfect to test our approach. Since the agents can get stuck in an equilibrium at any timestep, they will have ample opportunity to escape sub-optimal ones as well.

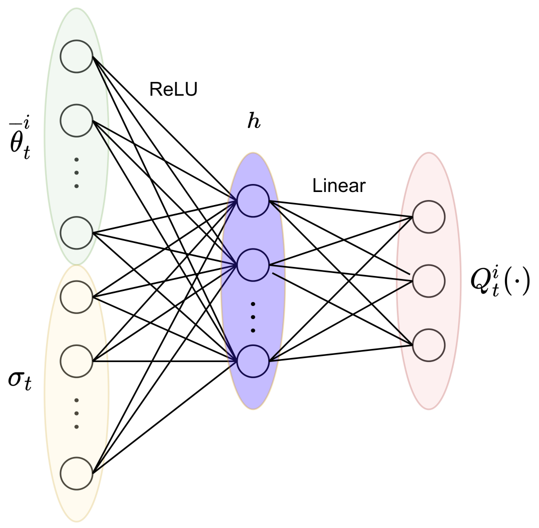
5.3 Experimental Setup

In all of our experiments, we set the horizon to 3. We train a separate DQN $Q_t^i(\cdot)$ for each agent and each timestep $t \in \{0, 1, 2\}$ in the undiscounted Dec-Tiger domain. We let the DQNs train for 1000000 iterations, whereby we repeat each experiment 50 times to obtain more robust results. We evaluate the DQNs' intermediate performance by completing 100 simulations once every 1000 iterations. After training is finished, we evaluate the final performance of the networks by simulating another 10000 trajectories. When evaluating the network's performance, we choose the action corresponding to the maximum Q-value, i.e. we act greedily.

Network Architecture. The default parameters for the DQNs are listed in Table 5.4. Furthermore, we investigate two network architectures of the DQNs. The first network has the input layer directly connected to the output layer, using a linear activation function. It can thus be seen as a linear function approximator. The second architecture has an intermediate hidden layer of size h_t with a rectified linear unit as activation function. The size of h_t is determined by the respective timestep of the DQN: $h_t \in \{0, 10, 100\}$. Since the size of σ_t increases each timestep, so does the hidden layer. Because of the presence of reLU activations, this network represents a non-linear function approximator. See Fig. 5.2 for a graphical overview of the architectures. We shall refer to these architectures as the *small* and *large* network henceforth.



(a) **Small**: A fully connected linear layer connects the input layer directly to the output layer.



(b) **Large**: Contains an intermediary hidden layer of size h_t with ReLU activation.

Figure 5.2: Graphical overview of both **small** and **large** σ-DQN architectures.

Main simulation pseudocode. The main simulation pseudocode for σ-DQN is presented in Alg. 4. i-DQN uses largely the same pseudocode, except that it excludes any functionality based on σ_t.

5.4 Experimental Runtime

Although the runtime of our algorithms are not the focus of our experiments, it is nevertheless important to state it. We present it in here, as to not distract from our actual experiments in the following Chapters. Table 5.5 shows the average time in seconds it takes to complete 1000 training iterations in the Dec-Tiger domain, for various horizons. Although we solely train agents using $h = 3$, varying the horizon does bring to light the scale-up for σ-DQN. Where i-DQN's runtime scales linearly with the horizon, σ-DQN clearly scales exponentially, as expected. Moreover, although n is fixed at 2 in Dec-Tiger (and thus not shown in Table 5.5), σ-DQN also scales exponentially in n, whereas i-DQN is indifferent to n.

Algorithm 4: Main simulation loop

```
$Q_0^1, Q_1^1, Q_2^1, Q_0^2, Q_1^2, Q_2^2 \leftarrow$ Deep Q-Networks for agents 1,2 and
  timesteps 0,1,2
$\hat{Q}_0^1, \hat{Q}_1^1, \hat{Q}_2^1, \hat{Q}_0^2, \hat{Q}_1^2, \hat{Q}_2^2 \leftarrow$ Target Networks, with equal parameters to
  the Deep Q Networks $Q_{0:2}^{1,2}$ initially.
$D_0^1, D_1^1, D_2^1, D_0^2, D_1^2, D_2^2 \leftarrow$ Replay Memory for each DQN.
for $e = 0 \ldots 1000000$ do
    $\overline{\theta}_0^1 \leftarrow \emptyset; \overline{\theta}_0^2 \leftarrow \emptyset$                    // Initialize empty histories
    $\sigma_0(s_l, \emptyset) \leftarrow 0.5$            // Initialise $\sigma$ to initial belief
    $\sigma_0(s_r, \emptyset) \leftarrow 0.5$
    for $t = 0 \ldots 2$ do
        for $i = 1 \ldots 2$                              // For both agents
         do
            $a_t^i \leftarrow$ exploration_strategy($Q_t^i(\overline{\theta}_t^i, \sigma_t, \cdot)$
        end
        $o_{t+1}, r_t \leftarrow$ env.step($a_t^1, a_t^2$)    // Environment emits joint
         observation and reward
        $\overline{\theta}_{t+1}^1 \leftarrow \langle \overline{\theta}_t^1, a_t^1, o_{t+1}^1 \rangle$  // $o_{t+1}^1$ is individual component of
         joint observation.
        $\overline{\theta}_{t+1}^2 \leftarrow \langle \overline{\theta}_t^2, a_t^2, o_{t+1}^2 \rangle$
        $\sigma_{t+1} \leftarrow$ update_sigma($\sigma_t$)
        for $i = 1 \ldots 2$                              // Handle training
         do
            if $size(D_t^i) = max\_mem\_size$ then
                $D_t^i$.pop()
            end
            $D_t^i \leftarrow \left\langle D_t^i, \left\langle \langle \overline{\theta}_t^i, \sigma_t \rangle, a_t^i, r_t, \langle \overline{\theta}_{t+1}^i, \sigma_{t+1} \rangle \right\rangle \right\rangle$
            Sample minibatch $B$ experiences
             $\left\langle \langle \overline{\theta}_j^i, \sigma_j \rangle, a_j^i, r_j, \langle \overline{\theta}_{j+1}^i, \sigma_{j+1} \rangle \right\rangle$ from $D_t^i$.
            if $j + 1 = horizon$ then
                $y_j \leftarrow r_j$
            end
            else
                $y_j \leftarrow r_j + \max_{a'} \hat{Q}_{j+1}^i(\overline{\theta}_{j+1}^i, \sigma_{j+1}, a')$          // Target
                 network of the next timestep
            end
            Perform gradient descent on $(y_j - Q_j^i(\overline{\theta}_j^i, \sigma_j, a_j^i))^2$ w.r.t.
             $Q_j^i$ network parameters.
            if $e \mod C = 0$    // Update Target Network every $C$
             then
                $\hat{Q}_t^i \leftarrow Q_t^i$
            end
        end
    end
end
```

	$h = 1$	$h = 2$	$h = 3$	$h = 4$
i-DQN	5.97 (0.01)	11.87 (0.10)	19.95 (0.26)	28.67 (0.35)
σ-DQN	5.86 (1.08)	21.83 (2.40)	105.81 (0.96)	4686.02 (46.16)

Table 5.5: Seconds (standard deviation) to complete 1000 training iterations in the Dec-Tiger environment for various horizons.

Chapter 6

Exploration in the space of individual actions

6.1 Setup

In this Chapter, we investigate the natural exploration strategy for independent learners, namely exploration in the space of individual actions. We follow the epsilon-greedy strategy ($\varepsilon = 0.1$). With probability $1 - \varepsilon$, an agent chooses the individual action corresponding to the highest Q-value. With probability ε, it chooses a random action instead. Both σ-DQN and i-DQN follow this exploration strategy. Recall that σ-DQN conditions on both the individual action-observation history and the sufficient statistic, whereas baseline i-DQN solely conditions on the former.

6.2 Decision Rules

The use of naive epsilon-greedy exploration requires stochastic decision rules, as given a certain action-observation history, we can take any action with probability at least $\frac{\varepsilon}{|\mathcal{A}^i|}$. In Alg. 5 we show pseudocode specifying `action_probability`, which is used to form the individual decision rules in Alg. 2.

Algorithm 5: `action_probability`$(\overline{\theta}^i_t, \sigma_t, a^i_t)$

```
1 if $a^i_t = \arg\max_a Q^i_t(\overline{\theta}^t_i, \sigma_t, a)$ then
2     $\delta^i_t(a^i_t|\overline{\theta}^i_t) \leftarrow (1-\varepsilon) + \frac{\varepsilon}{|\mathcal{A}^i|}$
3 else
4     $\delta^i_t(a^i_t|\overline{\theta}^i_t) \leftarrow \frac{\varepsilon}{|\mathcal{A}^i|}$
5 end
```

	small i-DQN	**small σ-DQN**	**large i-DQN**	**large σ-DQN**
Average Reward (std)	2.77 (7.18)	-10.29 (12.66)	-5.63 (16.48)	-8.21 (12.93)
no. Optimal Policies / 50	44	20	22	22

Table 6.1: Average reward and number of optimal policies found of the 50 runs, using exploration in the space of individual actions.

6.3 Results

For all four tested variants, small i-DQN, small σ-DQN, large i-DQN and large σ-DQN, we show the average reward of the final learned policies in Table 6.1. It also shows the number of optimal policies found, out of the total 50 runs. Moreover, Fig. 6.1, 6.2, 6.3 and 6.4 show the learning curves over the 1000000 training iterations for each DQN. An average reward of about 5 indicates an optimal policy has been found.

6.4 Analysis

Learned Policies. Let us discuss Table 6.1 first. We see that small i-DQN vastly outperforms all three other combinations. Of its 50 respective runs, 44 manage to find the optimal policy, at least twice as much as the others. Furthermore, we see that when increasing the size of i-DQN from small to large, performance deteriorates. This could be explained by the fact that i-DQN's input, a one-hot encoding of the action-observation history, never grows beyond 10 bits. Therefore, an intermediate hidden layer with 100 units is uncalled for, and we observe it hampers the learning process.

As for σ-DQN, we see that it manages to find the optimal policy less than half the time. We do, however, see a slight improvement from large σ-DQN over small σ-DQN, as we hypothesized, but not convincingly so.

Learning Curves. However, the final policies do not tell the whole story. When we inspect the learning curves, we see a clear pattern for each tested DQN: as soon as they settle for an equilibrium in which both agents jointly open any door at any timestep, they are completely stuck [1]. The oscillations that are visible are simply the result of the stochastisity in the environment. We fully expected i-DQN to get stuck in sub-optimal equilibria, but not σ-DQN.

Why do σ-DQNs not manage to escape sub-optimal equilibria, as we hypothesized? The reason is tied to the use of exploration in the space of individual actions, which requires stochastic decision rules. Recall that we recover the individual decision rule from the neural network by presenting each possible

[1]The rather slow improvement from -6 to about 5 visible in small i-DQN and small σ-DQN does not resemble agents getting unstuck from an equilibrium. Rather, it is simply the improvement from (1) always listening to (2) one agent opening a door at the final timestep, to (3) the optimal policy of both agents opening the same door at the final timestep (in case of hearing the tiger behind the other door twice). Neither (1) nor (2) are equilibria.

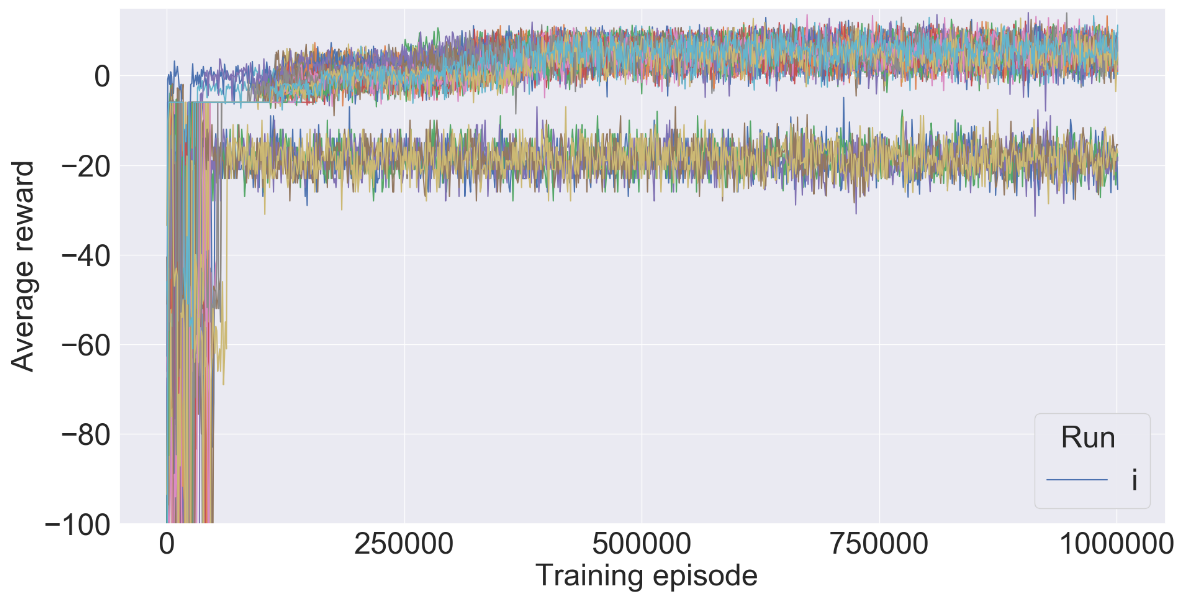


Figure 6.1: Small i-DQN learning curve, with exploration in the space of individual actions. Each line represents a single run. An average reward of about 5 indicates an optimal policy has been found.

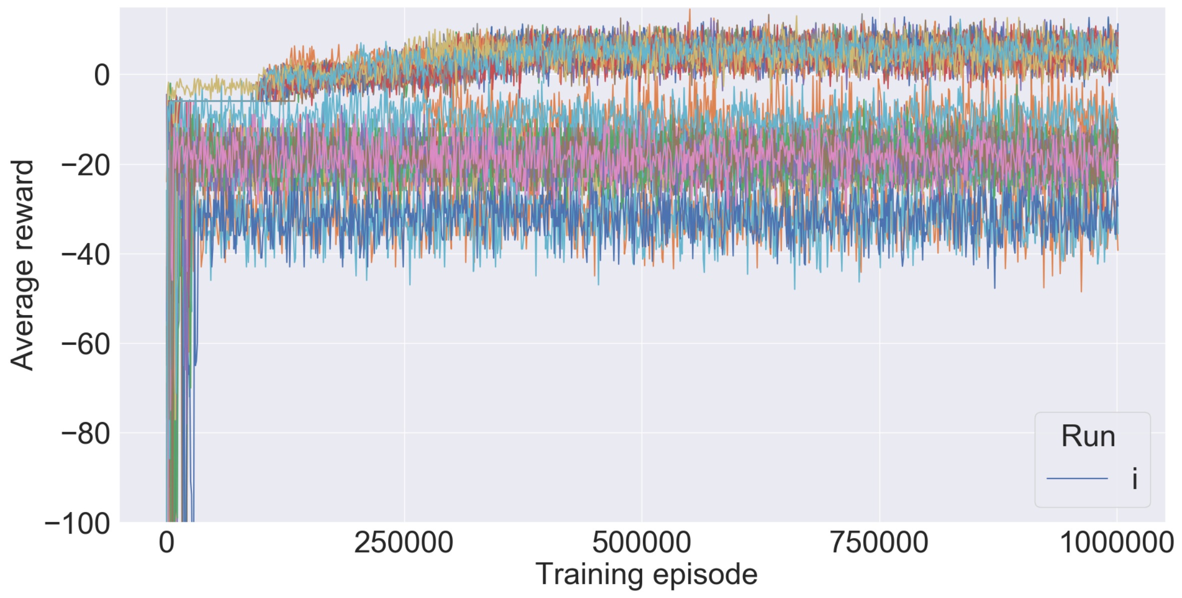


Figure 6.2: Small σ-DQN learning curve, with exploration in the space of individual actions.

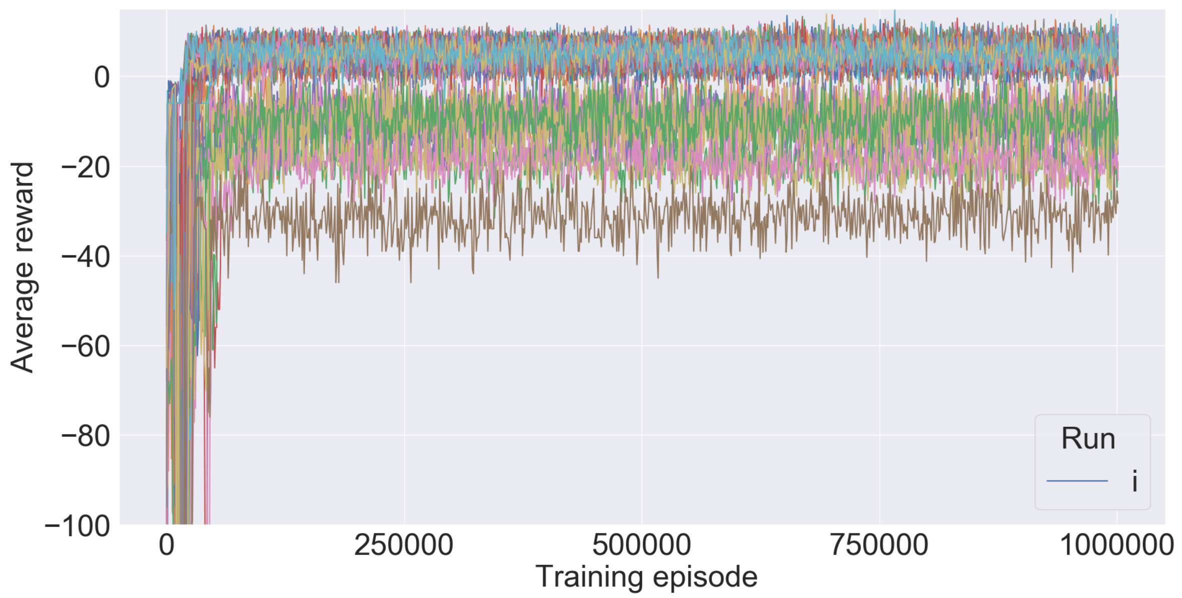


Figure 6.3: Large i-DQN learning curve, with exploration in the space of individual actions.

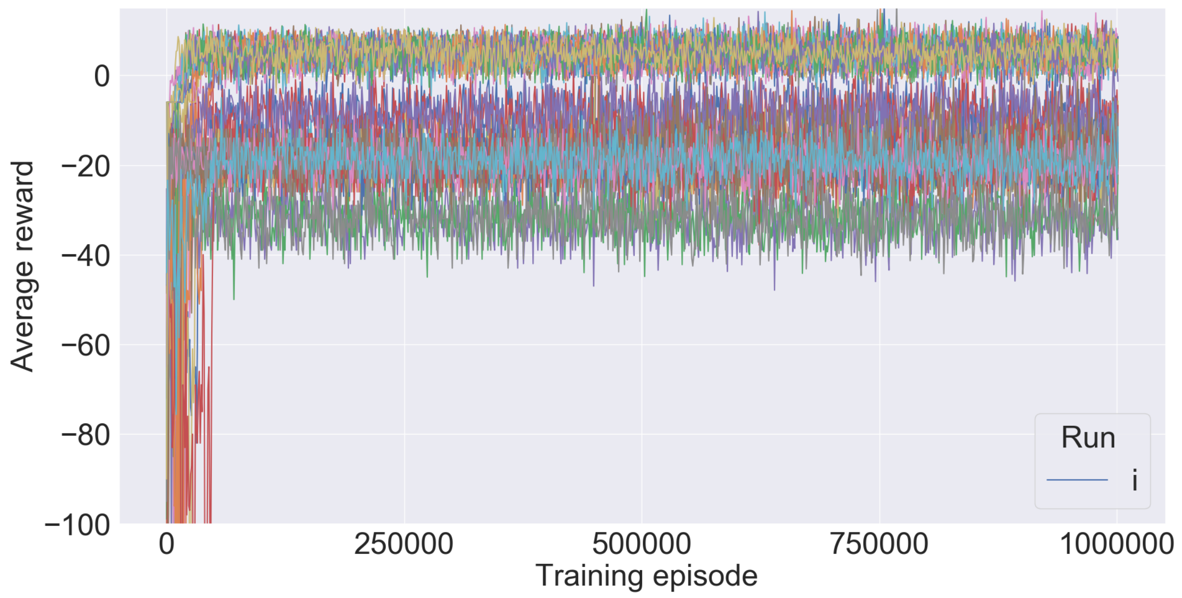


Figure 6.4: Large σ-DQN learning curve, with exploration in the space of individual actions.

action-observation history (alongside σ_t) to it. For each such $\overline{\theta}_t^i$, we observe the resulting Q-values $Q(\overline{\theta}_t^i, \sigma_t, \cdot)$, and based on those we assign probabilities to each action:

$$\delta_t^i(a_t^i|\overline{\theta}_t^i) = \begin{cases} 1 - \varepsilon + \frac{\varepsilon}{|\mathcal{A}^i|} & \text{if } a_t^i = \arg\max_a Q_t^i(\overline{\theta}_t^i, \sigma_t, a). \\ \frac{\varepsilon}{|\mathcal{A}^i|} & \text{otherwise.} \end{cases} \tag{6.4.1}$$

The actual action that was taken is not relevant to this procedure. In other words, the individual stochastic decision rule is the same regardless of whether the agent took a greedy or an exploratory action. When the agents are stuck in an equilibrium, their greedy actions $\arg\max_a Q_t^i(\overline{\theta}_t^i, \sigma_t, a)$ remain the same for all action-observation histories. As a consequence, their entire decision rules are stuck as well.

How does this affect the sufficient statistic? For ease of readability, we restate the update rule for σ_t.

$$\sigma_{t+1}(s_{t+1}, \overline{\theta}_{t+1}) = \sum_{s_t} P(s_{t+1}, o_{t+1}|s_t, a_t)\, \delta_t(a_t|\overline{\theta}_t)\, \sigma_t(s_t, \overline{\theta}_t) \tag{6.4.2}$$

Since the environment dynamics $P(s_{t+1}, o_{t+1}|s_t, a_t)$ are constant, the only factor that can bring about change in the sufficient statistic σ_{t+1} is the joint decision rule. When the individual decision rules, and thereby the joint decision rule are stuck, σ_t is completely stuck as well. That is, when agents are stuck in an equilibrium, they will repeatedly experience the same σ_t. Even when an agent takes an exploratory action, this is not reflected in its decision rule, and thus it is also not captured by the sufficient statistic. Therefore, σ_t has no added value for escaping sub-optimal equilibria.

6.5 Conclusion

To conclude, the reason why i-DQN outperforms σ-DQN quite substantially using exploration in the space of individual actions, is that agents settle for an equilibrium quite quickly. Since i-DQN's input space is much smaller, the agents can quickly focus on what is important: the individual action observation history. On the other hand, by the time σ-DQN agents have finally had enough time to decipher the meaning of the sufficient statistic, they are highly likely to already be stuck in an equilibrium. When this happens, σ_t is completely stuck as well, and thus it provides no useful information to escape sub-optimal equilibria. The sufficient statistic therefore only obfuscates the important input.

Chapter 7

Exploration in the space of individual decision rules

7.1 Setup

The problem with exploration in the space of individual actions is that an exploratory action of one agent is not observable to the other agent. The reason for this is that an actual exploratory action is not captured by the decision rule, and consequently it is not captured by the sufficient statistic either. Since *both* agents need to take a correct exploratory action *simultaneously* to escape an equilibrium, it would be desirable for exploratory actions of one agent to be visible to the other agent.

To overcome the aforementioned problem, we can have agents explore in the space of individual decision rules instead. Specifically, with probability $1 - \varepsilon$, the greedy decision rule described by the neural network is followed. With probability ε, a random, deterministic decision rule is formed by assigning a random action to every possible individual action-observation history. Subsequently, an agent simply looks up the prescribed action for its individual AOH in this newly formed decision rule.

Conceptually, taking the action prescribed by a random decision rule is equivalent to simply taking a random action. However, the key difference is that the decision rule is now capable of capturing an exploratory action. If the greedy, neural network decision rule is followed, it will be completely different from when an exploratory, random decision rule is followed. As a consequence, through the sufficient statistic, an agent will now know *when* the other agent has taken a greedy or exploratory action. Not only that, but the sufficient statistic also contains information about *how* the other agent explored, as this is precisely described by the random decision rule.

7.2 Decision Rules and Sufficient Statistic

Decision Rules. Exploration in the space of individual decision rules permits deterministic decision rules, as exploration is no longer on the action-level. To be specific, an agent either follows the arg max decision rule described by its neural network when acting greedily, or follows the random, deterministic rule when exploring instead. We present pseudocode, again specifying `action_probability`, in Alg. 6.

Algorithm 6: `action_probability`$(\overline{\theta}_t^i, \sigma_t, a_t^i, \delta_t^{i,rnd})$

```
1  if $random\_float > \varepsilon$    // Follow the deterministic decision
     rule from the neural network.
2   then
3      if $a_t^i = \arg\max_a Q_t^i(\overline{\theta}_t^i, \sigma_t, a)$ then
4          $\delta_t^i(a_t^i|\overline{\theta}_t^i) \leftarrow 1$
5      else
6          $\delta_t^i(a_t^i|\overline{\theta}_t^i) \leftarrow 0$
7      end
8  else if $random\_float \leq \varepsilon$   // Follow the random deterministic
     decision rule instead
9   then
10     if $a_t^i = \delta_t^{i,rnd}(\overline{\theta}_t^i)$ then
11         $\delta_t^i(a_t^i|\overline{\theta}_t^i) \leftarrow 1$
12     else
13         $\delta_t^i(a_t^i|\overline{\theta}_t^i) \leftarrow 0$
14     end
15 end
```

Sufficient Statistic. Since we are now working in the space of deterministic decision rules, one might suggest to use the sufficient statistic for deterministic policies, i.e. $\sigma_t(s_t, \overline{o}_t)$, instead of the sufficient statistic for stochastic policies $\sigma_t(s_t, \overline{\theta}_t^i)$. However, we still require the action-observation history as input to the DQN, because when the policy changes during learning, a certain observation history at one point might imply a different action-observation history at a later point. Hence, there is a mismatch between, on one hand, using the individual action-observation history as input (as is typical for stochastic decision rules), and, on the other hand, working in the space of deterministic decision rules. Our decision rules thus deterministically map action-observation histories to actions. We therefore also stick to our use of the general sufficient statistic $\sigma_t(s_t, \overline{\theta}_t^i)$, which contains a distribution over precisely those joint action-observation histories.

	small σ-DQN	**large σ-DQN**
Average Reward (std)	-6.72 (12.53)	-8.42 (13.35)
no. Optimal Policies / 50	26	23

Table 7.1: Average reward and number of optimal policies found of the 50 runs, using exploration in the space of individual decision rules.

7.3 Results

Since exploration in the space of individual decision rules reduces to exploration in the space of individual actions when an agent does not explicitly condition on the sufficient statistic, we do not repeat experiments for i-DQNs.

Again, we show the evaluation of the final learned policies of both small and large σ-DQNs in Tab. 7.1, as well as their respective learning curves in Fig. 7.1 and 7.2.

7.4 Analysis

Learned Policies. We observe that exploration in the space of individual decision rules still does not come close to the performance of small i-DQN, which managed to find 44 optimal policies earlier. With regard to the σ-DQNs, we observe that the number of optimal policies found is slightly higher now compared to exploration in the space of individual actions: small σ-DQN improved from 20 to 26, whereas large σ-DQN improved from 22 to 23. The average reward for small σ-DQN improved, whereas the average reward for large σ-DQN deteriorated slightly. However, due to the high standard deviations, none of these results are conclusive.

Learning Curves. Again, the more interesting aspect of our results are the learning curves. To our surprise, we still observe that σ-DQN gets completely stuck in sub-optimal equilibria as soon as one is found. Even though the exploration of one agent is now visible to the other agent through the sufficient statistic, the other agent is still not able to adapt its behavior accordingly.

This time, the reason can be explained as follows. In order to escape an equilibrium, both agents need to explore simultaneously. We hypothesized that the sufficient statistic will facilitate this, by capturing the exploratory decision rule. However, a key limitation we have overlooked is tied to the very definition of the sufficient statistic: σ_t is a summary of the **history** of joint decision rules φ_t at timestep t. Therefore, if one agent takes an exploratory action and thereby samples a random decision rule at timestep t, this will be captured only in σ_{t+1}. In other words, if one agent explores at timestep t, this will become visible to the other agent at timestep $t + 1$. Agents can therefore still not coordinate their (joint) exploration through the sufficient statistic. As a result, sub-optimal equilibria can still not be escaped.

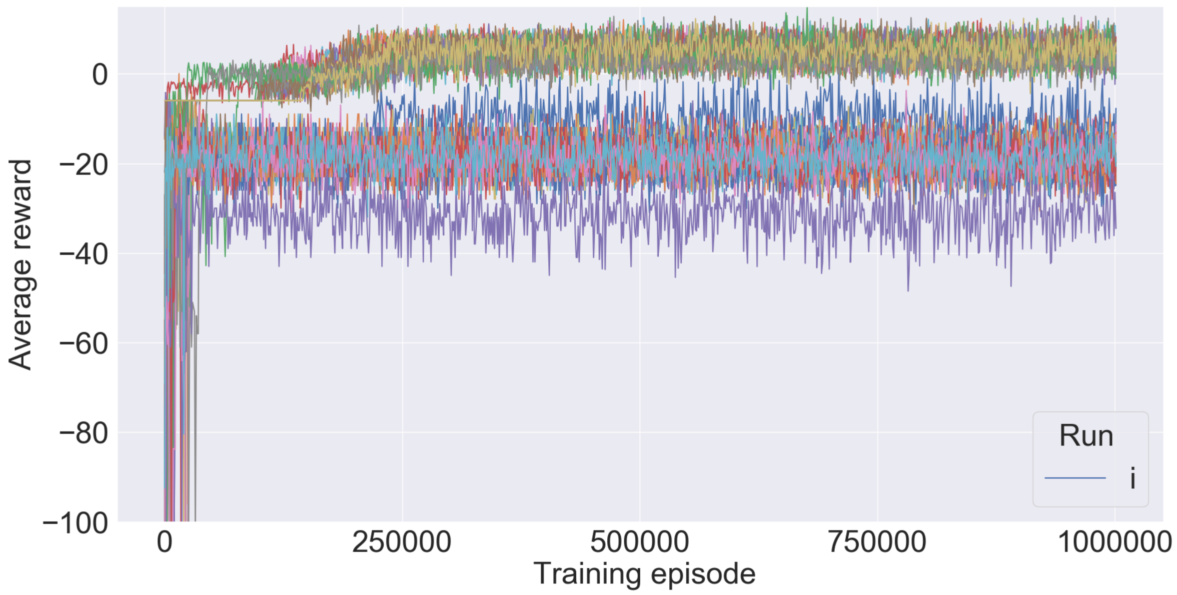


Figure 7.1: Small σ-DQN learning curve, with exploration in the space of individual decision rules.

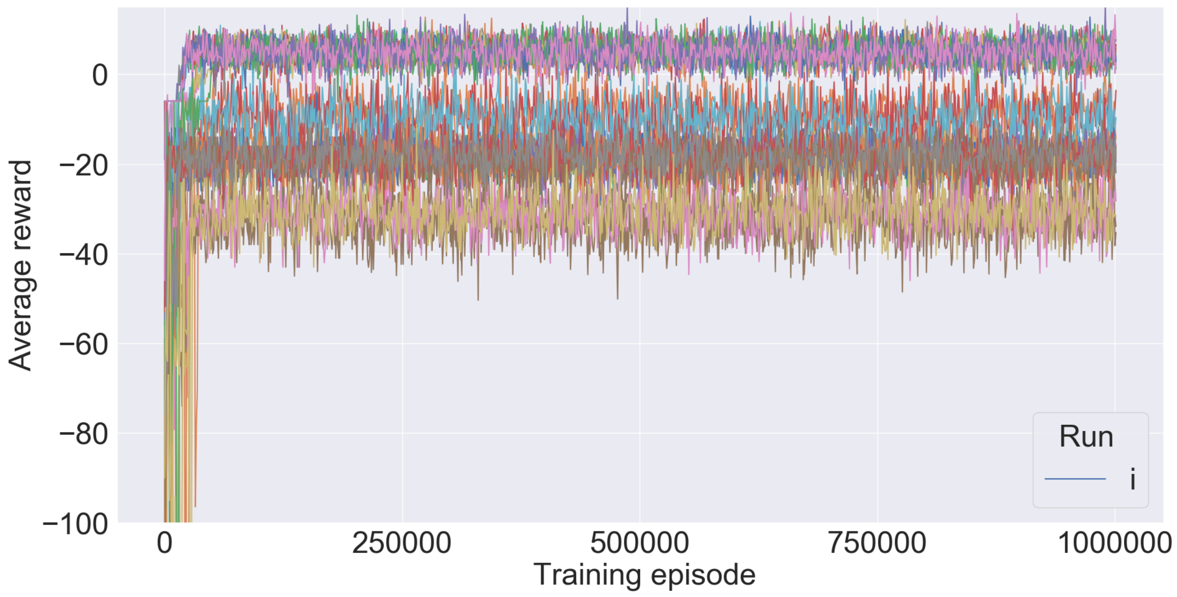


Figure 7.2: Large σ-DQN learning curve, with exploration in the space of individual decision rules.

7.5 Conclusion

In the Dec-Tiger domain, we observe that exploration in the space of individual actions or decision rules makes little difference. However, we still believe that exploration in the space of decision rules is the superior way to approach individual exploration when conditioning on the sufficient statistic. Essentially, the agents are still exploring to random actions. However, the key difference is that by exploring in the space of individual decision rules, an agent can communicate *when* and *how* it explored. Moreover, it allows for deterministic decision rules, which significantly reduces the size of σ_t's entries with non-zero probability.

Nevertheless, it still provides no way to escape sub-optimal equilibria, as an exploratory action by one agent only becomes apparent to the other agent when it is already too late.

Chapter 8

Exploration in the space of joint decision rules

8.1 Setup

So far we have observed that the agents quickly settle for an equilibrium, and that individual exploration is insufficient to escape them. One idea would then be to perform some form of joint exploration instead, which is precisely what we investigate in this Chapter.

Specifically, with probability $1-\varepsilon$, **both** agents follow their greedy, deterministic decision rule described by their respective DQNs. Alternatively, with probability ε, a random, deterministic joint decision rule is formed, and each agent takes the action prescribed by it. Note that exploration in the space of joint decision rules is precisely how Q-Learning for NOMDPs (i.e. $Q(\sigma_t, \delta_t)$) would explore as well. Furthermore, for i-DQNs, this exploration strategy simply reduces to exploration in the space of joint actions, which is what we will use as baseline here.

Since agents will now always explore simultaneously, a necessary condition to escape an equilibrium, we shall investigate how this affects the learning process.

8.2 Results

Like before, final evaluation of the policies is shown in Table 8.1. As for the learning curves, we notice that a single graph depicting all 50 runs is too cluttered now. We therefore split up each tested setting into four subgraphs, each containing 10 runs. These are depicted in Fig. 8.1, 8.2, 8.3 and 8.4.

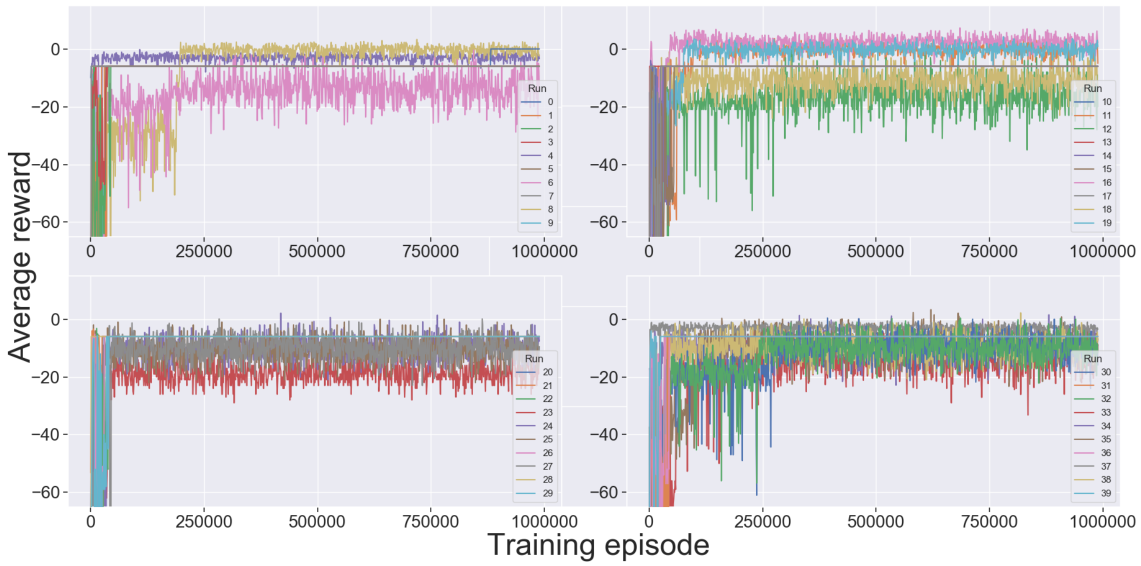


Figure 8.1: Small i-DQN learning curve, with exploration in the space of joint decision rules (which reduces to joint actions for i-DQN). Each subplot shows 10 runs.

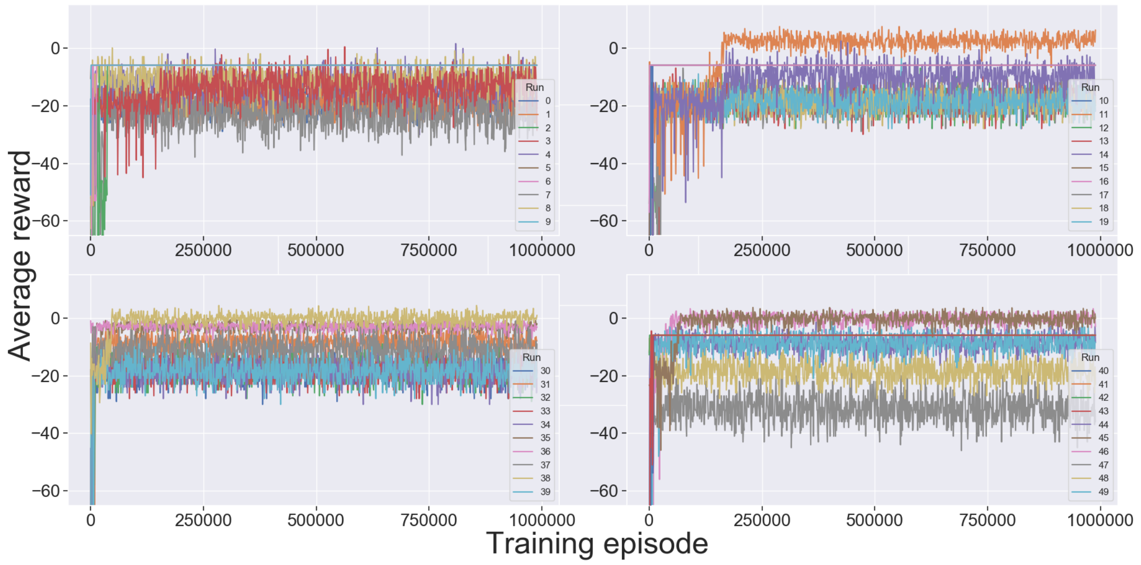


Figure 8.2: Small σ-DQN learning curve, with exploration in the space of joint decision rules.

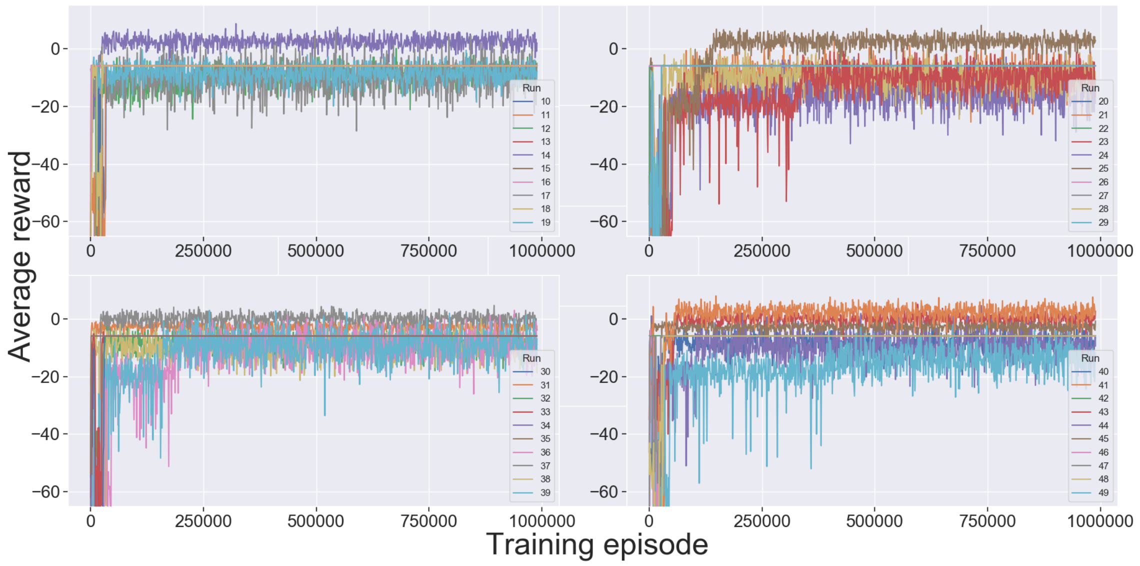


Figure 8.3: Large i-DQN learning curve, with exploration in the space of joint decision rules.

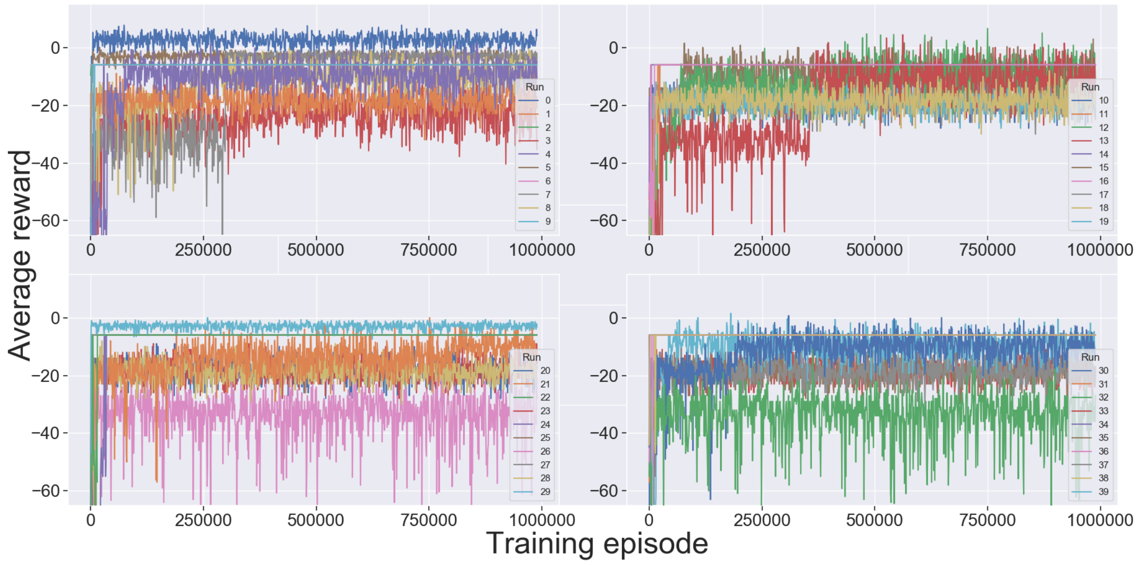


Figure 8.4: Large σ-DQN learning curve, with exploration in the space of joint decision rules.

	small i-DQN	**small σ-DQN**	**large i-DQN**	**large σ-DQN**
Average Reward (std)	-7.02 (3.88)	-11.82 (8.05)	-6.38 (3.53)	-10.15 (7.10)
no. Optimal Policies / 50	0	0	0	2

Table 8.1: Average reward and number of optimal policies found of the 50 runs, using exploration in the space of joint decision rule.

8.3 Analysis

Learned Policies. For any tested DQN, the final reward was observed to deteriorate using joint exploration compared to individual exploration. Moreover, large σ-DQN was the only DQN that managed to find two optimal policies. The others were not able to find a single one. The reason for these deteriorations can be explained by the fact that joint exploration is not even likely to converge to an equilibrium in the first place. To see why, again consider a one-shot toy environment with a single state, two agents and two actions, with payoff described as in Table 8.2.

	a^2	$\dot{a}^2$
a^1	10	-20
$\dot{a}^1$	0	10

Table 8.2: Toy environment demonstrating that agents are not able to escape non-equilibrium $(\dot{a}^1, a^2)$ with joint exploration.

Let us set $\varepsilon = 0.9$. Suppose the agents currently prefer joint action $(\dot{a}^1, a^2)$, yielding a reward of 0. Using individual exploration, when agent two explores towards the equilibrium $(\dot{a}^1, \dot{a}^2)$ by selecting action $\dot{a}^2$, with probability 0.95 the first agent chooses $\dot{a}^1$, and a reward of 10 is obtained. Similarly, a reward of -20 is received with probability 0.05. Therefore, on average the value of $\dot{a}^2$ will be 8.5, which is better than the value of action a^2. The agents will therefore likely converge to equilibrium (a^1, a^2) or $(\dot{a}^1, \dot{a}^2)$.

However, this is not the case when the agents use joint exploration. Again, suppose agent two explores towards the equilibrium $(\dot{a}^1, \dot{a}^2)$ by selecting action $\dot{a}^2$. This time, agent one is always acting randomly when agent two explores. As a result, the expected reward of $\dot{a}^2$ is -5. Moreover, any penalties that might occur when agent two explores to selecting its currently preferred action a^2, will be outweighed by greedily taking that same action with probability 0.9. As a result, the agent will not find be able to escape this non-equilibrium.

The reason exploration in the space of joint decision rules *does* work for the NOMDP setting, is that Q-Learning for NOMDPs reinforces $Q(\sigma_t, \delta_t)$, i.e. the joint decision rules. It therefore circumvents the ambiguity that comes with the action selection of the other agent in reinforcing individual actions, as this ambiguity is resolved when considering joint decision rules.

We observe this effect in the final policies as well. For example, some of the runs converged to the optimal policy, except that, at the final timestep, one agent did not open the left door when it heard the tiger behind the other door twice, whereas the other agent did. This joint policy is clearly sub-optimal, as it is always better for both agents to open a door together.

Learning Curves. When we inspect the learning curves, we firstly notice that many of the runs get stuck in a reward of -6, corresponding to the policy of always listen. The reason is precisely given by what we have just described: *on average*, the reward for exploring to opening the correct door at the final timestep (and another agent possibly exploring to opening the wrong door), is lower than safely choosing to listen. For example, even if an agent knows with certainty that it should open the left door, exploring towards this action will yield a reward of $\frac{1}{3}(9 + 20 - 100) = -23.67$ on average.

We do, however, see that some of the runs (for all tested DQN settings) manage to improve upon their policies during learning. This mostly happens when the current policy is quite bad, with an average reward of about -20 or worse. On average, the reward for a possible improvement then outweighs the current penalty. 'Less bad' policies, e.g. the ones that always listen with average reward of -6, could not be improved.

In theory, by conditioning on the sufficient statistic, it should be easier for agents to improve their policies. The reason is as follows. Suppose agents are stuck in opening a door at the second timestep. After taking a joint exploratory action, their best action at the third timestep depends on how they explored earlier. For example, if both listened, it would be wise to open a door. However, if one agent listened, but the other did not, the agents should not open a door. Without conditioning on σ_t, the agent that listened in both cases is unable to distinguish between these two scenarios, and consequently takes the same action in both. By conditioning on σ_t, however, the agent *can* differentiate between these two scenarios, and consequently adjust its behavior accordingly.

Although this theory is not directly visible in the learning curves, it does become apparent if we train the DQNs for another 1000000 iterations, as depicted in Fig. 8.5. We see that runs that are stuck in jointly opening a door in both the first and second timestep, manage to learn the optimal policy. As described, this is precisely where sufficient statistic would be useful. Conversely, training the i-DQNs for an additional 1000000 iterations did not change anything.

8.4 Conclusion

Contrary to individual exploration, joint exploration is not likely to converge to an equilibrium. Our experiments confirm this. Using joint exploration, the performance of the final policies deteriorates. None of the tested DQNs manage to find an optimal policy, except for large σ-DQN, which found two. However, we do observe (for all DQNs) that some runs manage to improve on their policy during learning, which we did not observe with individual exploration. This

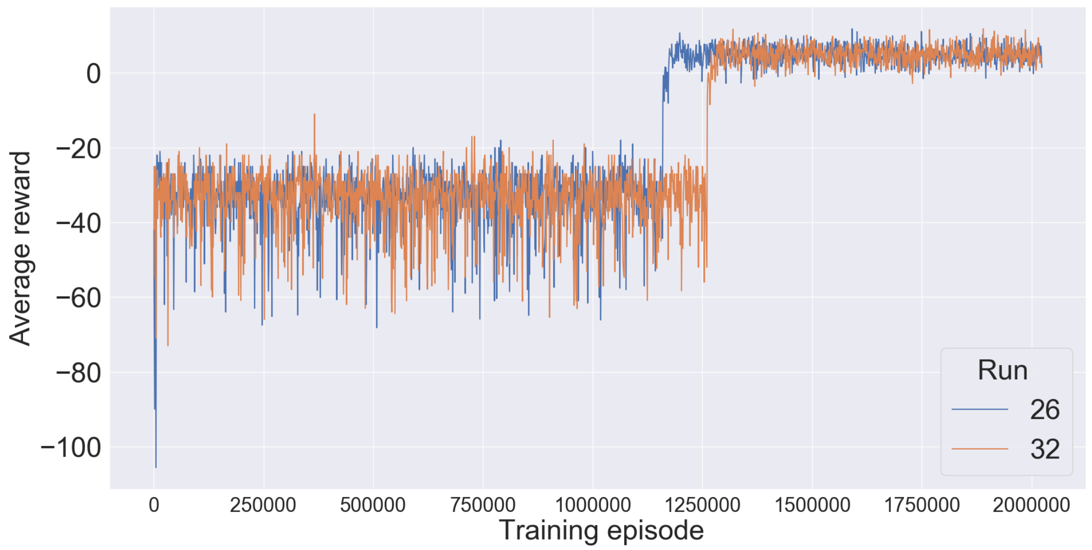


Figure 8.5: Two runs of large σ-DQN trained for an additional 1000000 iterations.

improvement is limited to bad policies, for which joint exploration tends to yield higher rewards on average than the current penalty.

Moreover, the effect of conditioning on the sufficient statistic only becomes apparent when we train the σ-DQNs for an additional 1000000 iterations, thereby confirming our hypothesis that such σ-DQNs require much longer training time than their i-DQN counterparts. Given more time, the large σ-DQN was able to perform a rather difficult improvement from a joint policy that opened a door at both the first and second timestep, to the optimal policy.

Chapter 9

Conceptually Sequencing the Decision-Making

9.1 Setup

So far, we have observed that agents can not escape equilibria using individual exploration. A potential solution is to use joint exploration, but this came with numerous other drawbacks. We therefore return to exploration in the space of individual decision rules.

Using the sufficient statistic, the problem with exploration in the space of individual decision rules is that an exploratory action of one agent only becomes apparent to the other agent at the next timestep. To solve this problem, we could additionally share the **current** decision rules during learning, and consequently let each agent condition on (1) the individual AOH, (2) the sufficient statistic, and (3) the current decision rule of the other agent(s). An exploratory action by one agent will then become apparent to the other agent at the current timestep, and it can adjust its behavior accordingly.

However, sharing the decision rule of both agents is impossible for the following reason. Normally, to extract δ_t^1 from agent 1's DQN, we take the current sufficient statistic and, alongside it, present each AOH $\overline{\theta}_t^1$ to its DQN, and observe the resulting action. When agent 1 additionally conditions on the decision rule of the other agent, we then require δ_t^2 as well for this procedure. The same, however, holds for forming δ_t^2, which would in turn require δ_t^1. In order to form an agent's input, we require its output. The problem of infinite recursion becomes apparent.

We can get around this problem by sequencing the decision making during learning, whereby we share the decision rules forward in time. Specifically, we pretend as if agent 1 acts first, followed by agent 2. In this way, agent two can use the decision rule of the first agent (but not the other way around), without the problem of infinite recursion. More generally, for N agents, agent i conditions on $\delta_t^{1:i-1}$, and its own decision rule d_t^i is subsequently used by agents

$\{i+1, \ldots, N\}$. As before, after we have collected each agent's resulting action, the joint action is formed and executed in the environment. Since we already require each agent's individual decision rule to update the sufficient statistic, this approach requires little additional computation.

Does conceptually sequencing the decision-making still allow for decentralized execution? The answer is it does. During execution, we no longer have to sequence the decision making, as the joint decision rules are completely determined and shared beforehand (by our third assumption). Therefore, each agent can compute the sufficient statistic and the decision rules of the earlier agents in advance as well, and no information needs to be shared during execution.

How does conditioning on the decision rules of earlier agents help to escape equilibria? Contrary to our previous approaches, if the first agent explores, this now becomes immediately apparent to the second agent. The second agent is thus able to differentiate between an exploratory and a greedy action of the other agent. It can therefore (learn to) act in accordance.

For example, suppose the agents are stuck in jointly opening a door at the first timestep. When the first agent samples a random decision rule that listens at the first timestep, the second agent will be notified of this. Initially, the second agent will probably choose a poor action. However, as long as the first agent keeps exploring to such a decision rule, at some point agent two will learn to listen as well (assuming the agents act optimally in the future). Once it does, listening in such cases will become the greedy action for the second agent. As a result, whenever the first agent now explores to listening at the first timestep, it will receive a reward higher than the current penalty for opening a door. It will then learn to listen as well, and the sub-optimal equilibrium is escaped.

Using this approach, the n'th essentially knows everything: it knows its own AOH $\overline{\theta}_t^n$, a summary of the joint decison rules followed thus far in σ_t, and the decision rules followed by the other agents at the current timestep in δ_t^{-n}. In a way, the n'th agent thus learns the values of joint decision rules $Q(\sigma_t, \delta_t^{-n}, \overline{\theta}_t^n, a_t^n)$. The key difference, however, is that its action space is still A^i (as opposed to the space of joint decision rules). Instead, it takes the other decision rules **as input**, and using its neural network, it can therefore potentially generalize across δ_t^{-n}, similarly to how it can generalize across various sufficient statistics. Once the n'th agent has learned the value of joint decision rules, the $n-1$'th agent can then correctly learn the value of reduced joint decision rules δ_t^{-n}, knowing that agent n will act optimally after it. And so on.

There is one final subtlety we have to address. During training, the DQN requires its prediction at the next timestep as target (line 28 in Alg. 4). At timestep t, the second agent therefore requires the decision rule of the first agent at timestep $t+1$, which could potentially be a random decision rule. We therefore make a fourth assumption:

4. **During learning, the exploratory timesteps are determined at the start of each trajectory.**

Under this assumption, both agents can compute σ_t and (possibly exploratory) δ_t, for all t, at the start of any trajectory.

	small $\sigma\delta$-DQN	**large $\sigma\delta$-DQN**
Average Reward (std)	4.57 (2.59)	5.00 (0.77)
no. Optimal Policies / 50	46	46

Table 9.1: Average reward and number of optimal policies found of the 50 runs, using exploration in the space of individual decision rules.

9.2 Results

We show the evaluation of final learned policies for these $\sigma\delta$-DQNs in Tab. 9.1. Moreover, their learning curves are depicted in Fig. 9.1 and 9.2.

9.3 Analysis

Learned Policies. Our numerical results validate our hypothesis that sharing the **current** decision rules is essential. We observe that both small and large $\sigma\delta$-DQNs manage achieve a better reward than the previously best result of small i-DQN (2.77). Moreover, both manage to find the optimal policy more consistently.

Learning Curves. When we inspect small $\sigma\delta$-DQN's learning curve, we see that many of the runs immediate settle for the optimal policy. Of the four runs that did not converge to the optimal policy, one (light blue) gets stuck in a sub-optimal equilibrium. The other three follow the optimal policy, except that **the second agent** does not open either the left door after hearing the tiger right twice, or the right door after hearing the tiger left twice. The first agent does follow the optimal policy. The resulting joint policy is thus in a non-equilibrium.

For large $\sigma\delta$-DQN, contrary to its small counterpart, we observe that **all** runs immediately settle for the optimal policy. However, we see that every now and then, they momentarily unlearn this policy and follow a significantly worse one. Afterwards, they quickly re-learn the optimal policy. This behavior was not observed for the small DQN.

The tendency for small $\sigma\delta$-DQN to rarely get stuck in a sub-optimal policy could be explained by the fact that its neural network simply is not powerful enough to deal with its large input, which would be as we hypothesized. This reason is further supported by the fact that large $\sigma\delta$-DQN never gets stuck in a sub-optimal policy. In turn, these large networks appear to sporadically unlearn the optimal policy, a behavior we have not observed thus far. Once the optimal policy is found, the decision rules of the first agent and the sufficient statistic should be completely fixed. Both agents should receive the same input over and over, and it therefore seems unlikely that the agents should ever decide to take an action that deviates from the optimal policy.

One possible explanation might be that when the first agent performs an update to its neural network for the Q-value of its optimal action, it inadvertently changes the greedy action for another input, through the generalization

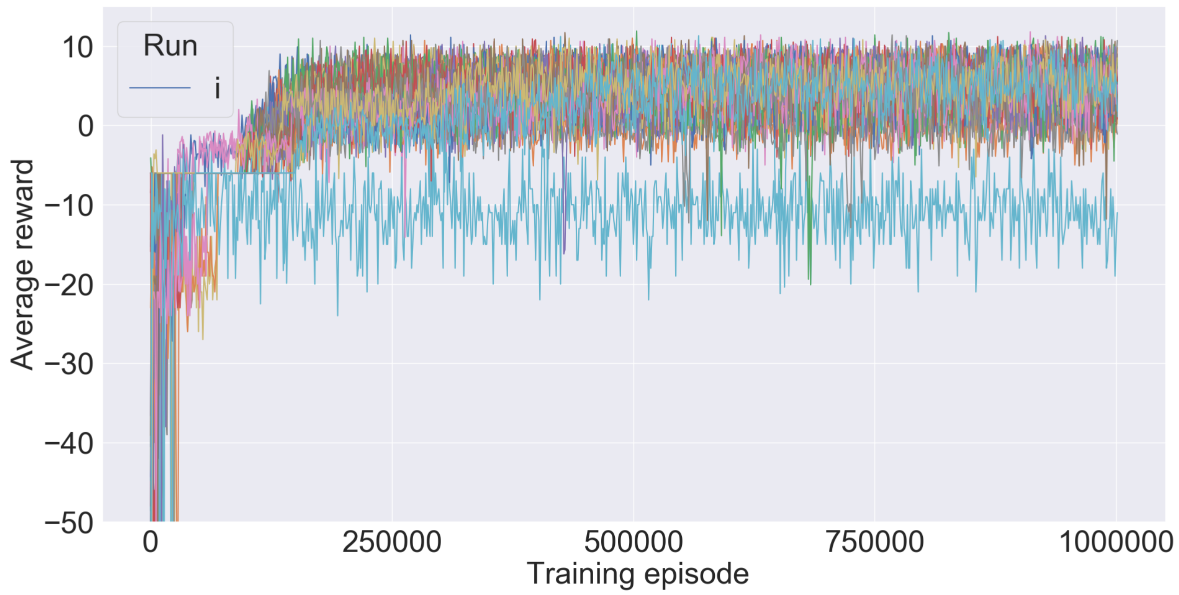


Figure 9.1: Small $\sigma\delta$-DQN learning curve, with conceptually sequencing the decision-making.

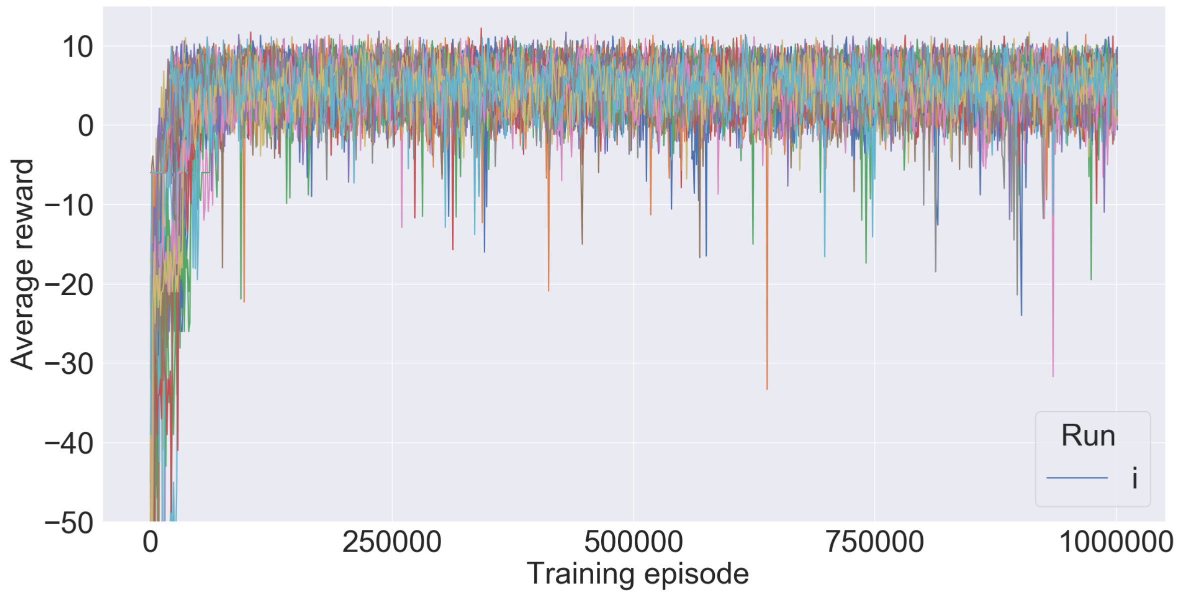


Figure 9.2: Large $\sigma\delta$-DQN learning curve, with conceptually sequencing the decision-making.

Policy for both agents	$\langle OL, Li, Li\rangle$	$\langle Li, OR, Li\rangle$	$\langle OL, Li, OR\rangle$	$\langle OR, OL, Li\rangle$	$\langle Li, OL, OR\rangle$
Number of runs	10	10	10	10	10

Table 9.2: Sub-optimal policies the agents are forced to follow before starting the learning process.

	small $\sigma\delta$-DQN	**large $\sigma\delta$-DQN**
Average Reward (std)	4.36 (3.15)	4.87 (1.03)
no. Optimal Policies / 50	45	43

Table 9.3: Average reward and number of optimal policies found of the 50 runs, using exploration in the space of individual decision rules.

of the DQN. As a result, the decision rule of agent one will momentarily differ, which is observed by the second agent, who consequently chooses the wrong action. However, this theory should then also apply to the σ-DQNs, as this change in decision rule is visible through the sufficient statistic as well. We did not observe such behavior earlier. Perhaps it is brought to light here because of the importance the second agent assigns to the relative compactness of the decision rule's dimensionality, compared to the size of the sufficient statistic.

9.4 Forcing sub-optimal equilibria initially

Although the performance of the $\sigma\delta$-DQN agents significantly improved on anything we have tried so far, we can not state with certainty that they are able to escape sub-optimal equilibria, as we have hypothesized. The reason is that almost all of the runs immediately settle for the optimal policy. We therefore perform one last experiment, in which we force the agents into sub-optimal equilibria beforehand, by temporarily adjusting the reward function. See Tab. 9.2 for the specifics of such policies.

See Tab. 9.3 for the final performance, and Fig. 9.3 and 9.4 for the learning curves. Although we observe a slight deterioration in final performance, the learning curves certainly validate our hypothesis that such $\sigma\delta$-DQNs are able to escape sub-optimal equilibria. Like before, the effect is more pronounced for the large DQN, which again confirms our hypothesis that a sufficiently complex network is mandated.

9.5 Conclusion

Our experiments indicate that augmenting σ-DQNs with the **current** decision rule of the previous agents, by conceptually sequencing the decision-making, greatly improves their performance. The agent that is last to act essentially knows everything: its local information, a summary of the past joint information, and the (reduced) current joint decision rule. Using this approach, agents

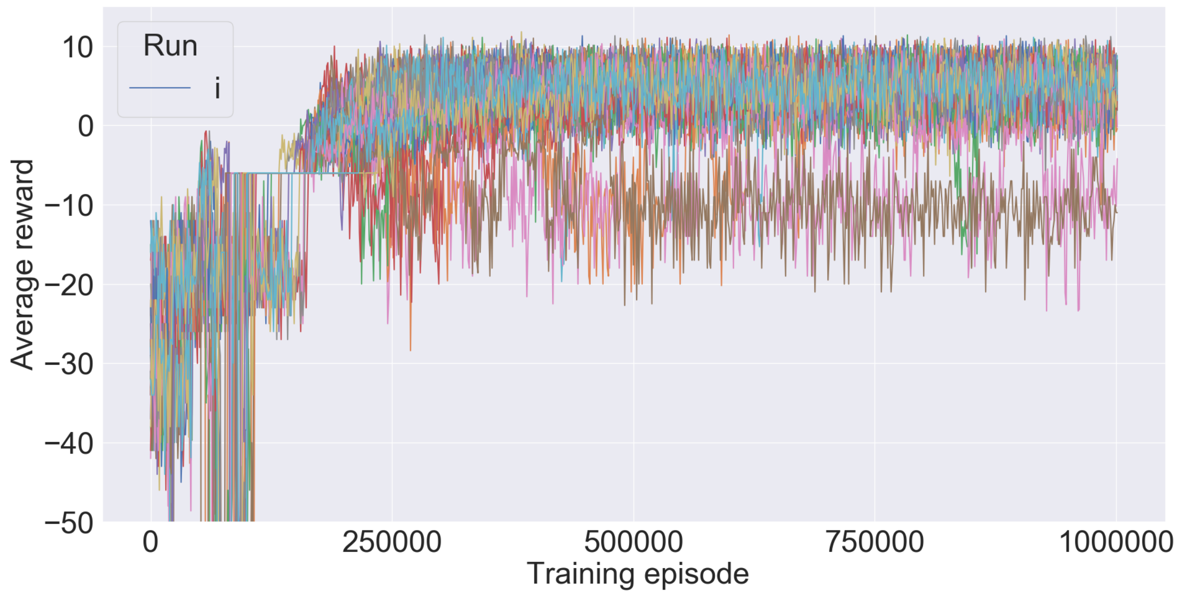


Figure 9.3: Small $\sigma\delta$-DQN learning curve, with conceptually sequencing the decision-making, and initially forcing the agents into a sub-optimal equilibrium.

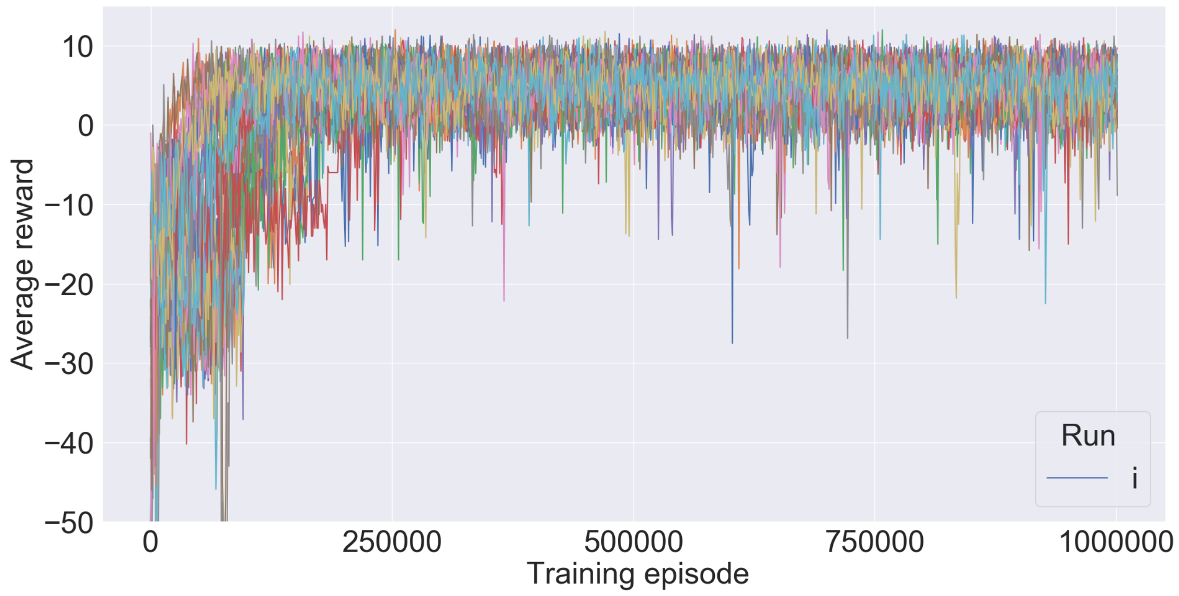


Figure 9.4: Large $\sigma\delta$-DQN learning curve, with conceptually sequencing the decision-making, and initially forcing the agents into a sub-optimal equilibrium.

manage to learn the optimal policy consistently. Moreover, they are consistently able to escape sub-optimal equilibria, even when they are forced upon them. These effects are more pronounced for sufficiently complex networks. However, when using such networks, we do observe that the agents sporadically unlearn the optimal policy before quickly re-learning it.

Chapter 10

Related Work

10.1 Addressing the convergence issues of Independent Q-Learning

When Tan [22] formalized Independent Q-Learners, he already identified the convergence issues accompanying them. Consequently, there have been more than a few works attempting to address these. However, almost all of these focus on the tabular, fully observable Multi-Agent MDP setting. We refer to [33] for an extensive Survey.

10.1.1 Tabular, Multi-Agent MDPs

Lauer et al. [34] introduced Distributed Q-Learning, in which agents update individual Q-function only when its value would increase.

$$Q^i(s, a^i) = \begin{cases} Q^i(s, a^i) & \text{if } y < Q^i(s, a^i) \\ (1 - \lambda)\, Q^i(s, a^i) + \lambda\, y & \text{otherwise.} \end{cases} \tag{10.1.1}$$

$$\text{where } y = r + \gamma \max_{a^{i\prime} \in \mathcal{A}^i} Q^i(s', a^{i\prime}) \tag{10.1.2}$$

In deterministic MMDPs, taking action a from state s leads to state s' with probability 1. Therefore, penalties to the Q-function can only be the result of other agents selecting a sub-optimal action, which are then ignored. In such deterministic MMDPs (and a tabular setting), Distributed Q-Learning is guaranteed to converge to the optimal equilibrium when it is unique. However, convergence guarantees are lost in stochastic MMDPs, in which penalties are not solely the result of uncoordinated actions, but can also be the result of stochasticity in the environment.

Hysteretic Q-Learning [35] aims to generalize Distributed Q-Learning to stochastic MMDPs by updating the Q-values for penalties with a separate, smaller learning rate λ', instead of simply neglecting them.

$$Q^i(s, a^i) = \begin{cases} (1-\lambda')\ Q^i(s, a^i) + \lambda'\, y & \text{if } y < Q^i(s, a^i) \\ (1-\lambda)\ Q^i(s, a^i) + \lambda\, y & \text{otherwise.} \end{cases} \tag{10.1.3}$$

$$\text{where } y = r + \gamma \max_{a^{i\prime} \in \mathcal{A}^i} Q^i(s', a^{i\prime}) \tag{10.1.4}$$

It thus addresses overestimations of the Q-values of Distributed Q-Learning in stochastic environments. However, it is not guaranteed to converge.

Related to hysteretic Q-Learning is lenient learning [36], which uses a temperature for the degree of leniency towards penalties. The temperature is inversely proportionate to the visitation count of each (state, action)-pair. Initially, it thereby largely ignores penalties due to miscoordination and exploration, but ultimately, as the temperature decreases, such penalties are accepted and the Q-values average out.

10.1.2 Dec-POMDPs

Much less progress has been made in addressing theoretical convergence guarantees for IQL in Dec-POMDPs, with results mostly being empirical. A leading cause for this is that we are not only dealing with the non-stationarity resulting from the presence of multiple agents, but the state is also no longer directly observable.

Banerjee et al. [37] propose distributed Q-Learning for Dec-POMDPs, in which each agent learns $Q^i(\overline{\theta}^i_t, a^i_t)$ in a tabular fashion. In their work, the agents take turns in updating their Q-function, as opposed to concurrent Independent Q-Learners. When one agent is optimizing its Q-function, the policies of the other agents are kept fixed, and the agent thus learns a best-response policy. This approach therefore by definition converges to an equilibrium, with no guarantees about its quality.

More commonly, a (Recurrent) neural network is used to deal with the large size of the action-observation history. However, even in the fully observable setting, such function approximators have no convergence guarantees. Many of these deep multi-agent reinforcement learning approaches tend to favor policy-based methods, which make use of a centralized critic [38][39][40]. There have, however, been some works using value-based methods like IQL.

Foerster et al. [41] empirically demonstrate how IQL using a Deep Recurrent Q-Network [42] can succesfully learn to act in partilaly observable, cooperative environments. They use inter-agent weight sharing to train a single DRQN which learns $Q(\langle i, a^i_{t-1}, o^i_t, h^i_{t-1}\rangle, a^i_t)$, where h^i_{t-1} is the agent's previous hidden state of its RNN, i.e. the summary of the full action-observation history. In other words, besides the inter-agent weight sharing, this can be seen as a direct application of IQL to Dec-POMDPs using a recurrent neural network as function approximator, without any additional techniques to address convergence issues. As a result, the problem of sub-optimal equilibria is largely neglected, and exploring local minima is explicitly stated as future work.

In similar vein, Omidshafiei et al. [43] learn to act in a multi-agent, multi-task setting with partial observability. However, again, each individual task is solved with Independent Q-Learning in combination with a Deep Recurrent Q-Network, without any further modifications to address the problem of sub-optimal equilibria.

10.2 Sufficient Statistic

10.2.1 Direct work on NOMDPs

The few works that directly build on reformulating a Dec-POMDP as a NOMDP (also coined an Occupancy-State MDP) are done by Dibangoye et al. For example, [44] introduces the NOMDP planning algorithm FB-HSVI. A key property it exploits is that the value function of a NOMDP is piece-wise linear and convex [29], like that of a POMDP. Since a NOMDP is a POMDP, the Heuristic Search Value Iteration planning algorithm for POMDPs was successfully applied to NOMDPs, albeit with numerous modifications.

Furthermore, in oSARSA [41], Dibangoye et al. focus on learning $Q(\sigma_t, \delta_t)$. Like us, they address the large state space with a linear or neural network function approximator. Conversely, where we address the large action space by taking the individual perspective $Q^i(\overline{\theta}_t^i, \sigma_t, a_t^i)$, they address the large action space by representing the optimal Q-function of joint decision rules as a set of α-vectors, thereby again exploiting the piece-wise linearity and convexity of the value function. Furthermore, to address the computational difficulty of identifying the current best joint decision rule, a mixed integer linear program is used. Finally, they do not assume any knowledge about the environment (whereas we do), and instead introduce a way of estimating σ_t from interactions with it.

10.2.2 Relating the Sufficient Statistic to the Public Belief

Some Dec-POMDPs allow for partial information sharing during execution [31]. Foerster et al. [32] exploit such a structure in the game of Hanabi. Like us, they take the individual perspective in which each agent takes as input its local information ($\overline{\theta}_t^i$ in our case), as well as the public belief (σ_t in our case). The key difference is that in Hanabi, the actions of agents are visible to everyone. The public belief thus only needs to maintain a probability distribution over observation histories (and possibly states). Moreover, if the policies of the agents are known, when agents observe an action by another agent, they can *retrospectively* [45] prune observation histories that are inconsistent with the actions that agent has taken thus far. For example, if an agent fails to play a diamond-suited card at the fifth timestep, and it has not drawn any cards between the first and fifth timestep, this also implies it did not have a diamond-suited card at the first timestep. As a result, agents can reduce the size of the set of elements in the public belief with non-zero probability at an earlier timestep using the newly acquired information.

Our approach can be seen as a generalization of the aforementioned one to Dec-POMDPs *without* partial information sharing. When the actions of others are not visible, such retrospective belief updates are impossible. This is precisely why the observation set in a NOMDP only contains the NULL observation: there is no additional information to be gained regarding the sufficient statistic during execution. Consequently, σ_t reasons about joint decision rules, i.e. what the agents would have done for **any** situation, rather than the action that was actually taken as in the public belief.

Chapter 11

Conclusion, Discussion and Future Work

11.1 Conclusion

In this Thesis, we have approached Decentralized Partially Observable Markov Decision Processes from the perspective of a Non-Observable Markov Decision Process. In this NOMDP, a coordinator decides on joint decision rules δ_t given the plan-time sufficient statistic σ_t. We have investigated how Q-Learning for such a NOMDP, which would learn $Q(\sigma_t, \delta_t)$, can be made tractible. We have attempted this by (1) making an approximation regarding the state space by using a Deep Q-Network function approximator, and (2) making an approximation regarding the action space by considering individual actions instead of joint decision rules.

We have thus followed the perspective of Independent Q-Learners, which attempt to learn $Q_t^i(\overline{\theta}_t^i, \sigma_t, a_t^i)$. Our main contribution is an investigation of the effects of conditioning such Independent Q-Learners on the sufficient statistic σ_t, as opposed to solely their individual action-observation history $\overline{\theta}_t^i$, as is commonly done elsewhere. Additionally, we have delineated how conditioning on σ_t allows for several novel exploration strategies, and subsequently we have investigated their effects on the learning process.

Let us concretely answer our Research Questions now.

1. **Can independent learners (more consistently) find the optimal joint policy by conditioning on the sufficient** statistic?
2. **(Related) Does conditioning on the sufficient statistic prevent premature convergence of independent learners to sub-optimal equilibria?**

σ-DQN is unable to escape sub-optimal equilibria using exploration in the space of both individual actions as individual decision rules. To escape such an equilibrium, both agents must explore favorably simultaneously. When the agents explore in the space of individual actions, an exploratory action by one agent is unobservable to the other agent, because an agent's stochastic decision rule is the same regardless of the *actual* action it took. When the agents explore in the space of individual decision rules, an agent now samples a random, deterministic decision rule when it explores. Consequently, through the sufficient statistic, the other agent can now observe this exploratory action. However, because the sufficient statistic is a summary of the *past* joint policy, the other agent can only observe this exploratory action at the next timestep. They are therefore still unable to successfully coordinate their exploration. During learning, the agents settle for an equilibrium quite quickly. Since the sufficient statistic has no added value for escaping such an equilibrium, such σ-DQNs fail to find the optimal policy consistently.

Conversely, when agents explore in the space of joint decision rules, they *do* manage to escape sub-optimal equilibria, but only when they are particularly bad. The main drawback of joint exploration is that, when one agent explores, the other agent is always taking a random action as well. Joint exploration is therefore not likely to settle for an equilibrium in the first place. Since Dec-Tiger is quite a punishing environment, *on average* joint exploration results in largely negative rewards. As a result, the agents were able to find the optimal policy only twice out of 100 runs.

Finally, we address the shortcoming of exploration in the space of individual decision rules by conceptually sequencing the decision-making. Specifically, the second agent additionally conditions on the current decision rule of the first agent. As a result, an exploratory action by the first agent is immediately observable to the second agent, and it can thus act accordingly. This method finds the optimal policy consistently, even when we explicitly force the learners into sub-optimal equilibria initially.

3. **How do various exploration strategies relate to the decision rules, the sufficient statistic and the learning process?**

We have already delineated how exploration strategies affect the learning process, and shall focus on the decision rules and the sufficient statistic now. Exploration in the space of individual actions requires stochastic decision rules, as given a specific action-observation history, we can take any action with probability at least $\frac{\varepsilon}{|\mathcal{A}^i|}$. This naturally warrants the sufficient statistic for general policies, i.e. $\sigma_t(s_t, \overline{\theta}_t^i)$.

Conversely, exploration in the space of individual and joint decision rules permits deterministic decision rules. When acting greedily, the agents follow the decision rule as prescribed by the neural network, i.e. by argmax'ing the DQN's Q-values. Alternatively, when the agents explore, they form a deterministic decision rule by assigning a random action to each action-observation history. In principle, exploration in the space of decision rules also permits the sufficient statistic for deterministic policies, i.e. $\sigma_t(s_t, \overline{o}_t)$. However, we have stuck with the general sufficient statistic by arguing that the DQN requires the action-observation history as input (instead of solely the observation history, as is common for deterministic decision rules). The general sufficient statistic maintains a probability distribution over precisely those action-observation histories.

4. **How does the complexity of the Deep Q-Network influence the effects of conditioning the learners on the sufficient statistic?**

When the agents explore individually without explicitly conditioning on the decision rule of the previous agent, they quickly settle for an equilibrium. It appears arbitrary which of the equilibria they find. As a result, we cannot draw significant conclusions relating the complexity of the DQN to the effects of conditioning the learners on the sufficient statistic for these experiments.

When the agents explore in the space of joint decision rules, we observe a slight increase in iterations that manage to improve a sub-optimal policy for large σ-DQN compared to small σ-DQN. However, the final learned policies are terrible for both, due to the aforementioned drawbacks of joint exploration.

Finally, when we conceptually sequence the decision-making, we can observe a significant effect of increasing the complexity of the DQN. Where small $\sigma\delta$-DQN still settled for a sub-optimal equilibrium every now and then, large $\sigma\delta$-DQN was able to learn the optimal policy always. However, this more complex network was also observed to sporadically unlearn the optimal policy, before quickly re-learning it.

11.2 Discussion and Future Work

The methods we have investigated in this Thesis are quite general, and can in principle be applied to any Dec-POMDP. Of course, the main limitation to our approach is the complexity of updating the sufficient statistic at each timestep: $\mathcal{O}\left((|\mathcal{A}^*|) \cdot |\Omega^*|)^{n(t+1)} \cdot |\mathcal{S}|^2\right)$. There is one interesting line of future work to address it. In our experiments, we have delineated how deterministic decision rules can be used even during learning. Nevertheless, the agents still used the individual action-observation histories as input to the DQNs. This, in turn, required the sufficient statistic for general policies $\sigma_t(s_t, \overline{\theta}^i_t)$. We have argued why agents need to use the action-observation history as input, instead of the observation history (as is common for deterministic decision rules). The reason is that, if the agents instead condition on the observation history, such an OH might imply a different AOH at a later timestep during learning. When training the DQN, the same observation history could therefore be sampled twice from its memory, where in each one different actions were taken, without the agent being able to differentiate between them.

However, by conditioning on the sufficient statistic, the agents *can* actually differentiate between such observation histories. Since the sufficient statistic is a summary of the past joint policy, an agent will be able to differentiate between an observation history that implies multiple action-observation histories through σ_t. We can therefore potentially use the observation history as input, and instead maintain the sufficient statistic for deterministic policies $\sigma_t(s_t, \bar{o}_t)$. This significantly reduces the complexity of updating σ_t to $\mathcal{O}\left(|\Omega^*|^{n(t+1)} \cdot |\mathcal{S}|^2\right)$.

We would also like to point towards some work that reduce the dimensionality of the sufficient statistic, by identifying various equivalence relations [44][46]. As a result, both the decision rules and the sufficient statistic can be represented more compactly. However, one problem with such equivalence relations is that they are usually computed after forming the sufficient statistic, and thus might not directly improve on the complexity of computing the sufficient statistic. We already use a function approximator to handle the large dimensionality of the sufficient statistic. Nevertheless. the use of such compact representations can be further investigated within our setting.

Bibliography

[1] Ranjit Nair, Milind Tambe, Makoto Yokoo, David Pynadath, and Stacy Marsella. Taming decentralized pomdps: Towards efficient policy computation for multiagent settings. In *IJCAI*, volume 3, pages 705–711, 2003.

[2] Eric A Hansen, Daniel S Bernstein, and Shlomo Zilberstein. Dynamic programming for partially observable stochastic games. In *AAAI*, volume 4, pages 709–715, 2004.

[3] Daniel Szer, François Charpillet, and Shlomo Zilberstein. Maa*: A heuristic search algorithm for solving decentralized pomdps. *arXiv preprint arXiv:1207.1359*, 2012.

[4] Christopher Amato, Jilles Steeve Dibangoye, and Shlomo Zilberstein. Incremental policy generation for finite-horizon dec-pomdps. In *Nineteenth International Conference on Automated Planning and Scheduling*, 2009.

[5] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[6] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[7] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

[8] Richard Bellman. A markovian decision process. *Journal of mathematics and mechanics*, pages 679–684, 1957.

[9] Martin L Puterman. *Markov Decision Processes.: Discrete Stochastic Dynamic Programming.* John Wiley & Sons, 2014.

[10] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction.* MIT press, 2018.

[11] Karl J Astrom. Optimal control of markov processes with incomplete state information. *Journal of mathematical analysis and applications*, 10(1):174–205, 1965.

[12] Richard D Smallwood and Edward J Sondik. The optimal control of partially observable markov processes over a finite horizon. *Operations research*, 21(5):1071–1088, 1973.

[13] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

[14] Eric A Hansen. Solving pomdps by searching in policy space. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pages 211–219. Morgan Kaufmann Publishers Inc., 1998.

[15] Matthijs TJ Spaan. Partially observable markov decision processes. In *Reinforcement Learning*, pages 387–414. Springer, 2012.

[16] Satinder P Singh, Tommi Jaakkola, and Michael I Jordan. Learning without state-estimation in partially observable markovian decision processes. In *Machine Learning Proceedings 1994*, pages 284–292. Elsevier, 1994.

[17] Edward J Sondik. The optimal control of partially observable markov processes over the infinite horizon: Discounted costs. *Operations research*, 26(2):282–304, 1978.

[18] Frans A Oliehoek, Christopher Amato, et al. *A concise introduction to decentralized POMDPs*, volume 1. Springer, 2016.

[19] Craig Boutilier. Sequential optimality and coordination in multiagent systems. In *IJCAI*, volume 99, pages 478–485, 1999.

[20] Daniel S Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. The complexity of decentralized control of markov decision processes. *Mathematics of operations research*, 27(4):819–840, 2002.

[21] Frans A Oliehoek, Matthijs TJ Spaan, and Nikos Vlassis. Optimal and approximate q-value functions for decentralized pomdps. *Journal of Artificial Intelligence Research*, 32:289–353, 2008.

[22] Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning*, pages 330–337, 1993.

[23] Stevo Bozinovski. A self-learning system using secondary reinforcement. *Cybernetics and Systems Research*, pages 397–402, 1982.

[24] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[25] Caroline Claus and Craig Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. *AAAI/IAAI*, 1998(746-752):2, 1998.

[26] Long-Ji Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8(3-4):293–321, 1992.

[27] Landon Kraemer and Bikramjit Banerjee. Multi-agent reinforcement learning as a rehearsal for decentralized planning. *Neurocomputing*, 190:82–94, 2016.

[28] Frans Adriaan Oliehoek. Sufficient plan-time statistics for decentralized pomdps. In *Twenty-Third International Joint Conference on Artificial Intelligence*, 2013.

[29] Frans A Oliehoek and Christopher Amato. Dec-pomdps as non-observable mdps. 2014.

[30] Christopher John Cornish Hellaby Watkins. Learning from delayed rewards. 1989.

[31] Ashutosh Nayyar, Aditya Mahajan, and Demosthenis Teneketzis. Decentralized stochastic control with partial history sharing: A common information approach. *IEEE Transactions on Automatic Control*, 58(7):1644–1658, 2013.

[32] Jakob N Foerster, Francis Song, Edward Hughes, Neil Burch, Iain Dunning, Shimon Whiteson, Matthew Botvinick, and Michael Bowling. Bayesian action decoder for deep multi-agent reinforcement learning. *arXiv preprint arXiv:1811.01458*, 2018.

[33] Laetitia Matignon, Guillaume J Laurent, and Nadine Le Fort-Piat. Independent reinforcement learners in cooperative markov games: a survey regarding coordination problems. 2012.

[34] Martin Lauer and Martin 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.

[35] Laëtitia Matignon, Guillaume J Laurent, and Nadine Le Fort-Piat. Hysteretic q-learning: an algorithm for decentralized reinforcement learning in cooperative multi-agent teams. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 64–69. IEEE, 2007.

[36] Liviu Panait, Keith Sullivan, and Sean Luke. Lenient learners in cooperative multiagent systems. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 801–803, 2006.

[37] Bikramjit Banerjee, Jeremy Lyle, Landon Kraemer, and Rajesh Yellamraju. Sample bounded distributed reinforcement learning for decentralized pomdps. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[38] Jakob N Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[39] Guillaume Bono, Jilles Dibangoye, Laëtitia Matignon, Florian Pereyron, and Olivier Simonin. On the study of cooperative multi-agent policy gradient. 2018.

[40] Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in neural information processing systems*, pages 6379–6390, 2017.

[41] Jakob N Foerster, Yannis M Assael, Nando de Freitas, and Shimon Whiteson. Learning to communicate to solve riddles with deep distributed recurrent q-networks. *arXiv preprint arXiv:1602.02672*, 2016.

[42] Matthew Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps. In *2015 AAAI Fall Symposium Series*, 2015.

[43] Shayegan Omidshafiei, Jason Pazis, Christopher Amato, Jonathan P How, and John Vian. Deep decentralized multi-task multi-agent reinforcement learning under partial observability. *arXiv preprint arXiv:1703.06182*, 2017.

[44] Jilles Steeve Dibangoye, Christopher Amato, Olivier Buffet, and François Charpillet. Optimally solving dec-pomdps as continuous-state mdps: Theory and algorithms. 2014.

[45] Adam Lerer, Hengyuan Hu, Jakob N Foerster, and Noam Brown. Improving policies via search in cooperative partially observable games. In *AAAI*, pages 7187–7194, 2020.

[46] Shimon Whiteson. Incremental clustering and expansion for faster optimal planning in decentralized pomdps. *Journal of Artificial Intelligence Research*, 46, 2013.