University of Modena and Reggio Emilia

Department of Sciences and Methods for Engineering

Master of Science in Mechatronics Engineering

Integrating Causality into Q-Learning for Adaptive Control in Dynamic Environments

Integrazione della Causalità in Q-Learning per il Controllo Adattivo in Ambienti Dinamici

Advisor:
Dott. Ing. Marco Lippi

Candidate:
Giovanni Briglia

Academic Year 2022/2023

To all young students, take your time;
To myself, be proud;
To those who love me, Thank you.

Abstract

This research thesis is dedicated to explore the integration of causal models into Reinforcement Learning (RL) algorithms, a concept known as Causal Reinforcement Learning. This approach holds significant promise across several domains such as healthcare, robotics, environment and economics. By combining data-driven learning with causal reasoning, Causal RL has the potential to create adaptable, robust, and trustworthy learning systems capable of excelling in complex and uncertain real-world scenarios.

Specifically, this thesis focuses on integrating causal knowledge into the Q-Learning algorithm, a form of Temporal-Difference learning algorithm widely used in RL. While Q-Learning may not be well-suited for dynamic environments due to its construction, introducing causal knowledge has shown the ability to enhance its performance. The causal knowledge is extracted through causal inference, employing operations like do-calculus to deduce the posterior probabilities of variables after specific actions. The primary objective of this work is to provide a comprehensive framework that seamlessly integrates causal knowledge into the Q-Learning algorithm. Additionally, the concepts of explainability and trustworthiness have been taken into account to guarantee the interpretability and transparent comprehension of the agent's decision-making process.

The approach outlined in this thesis consists of two primary components. The initial phase entails the agent's navigation within a compact environment featuring a solitary randomly moving enemy. During this phase, the agent leverages causal inference to understand the outcomes resulting from its actions, and these outcomes are stored for subsequent use. In the latter phase, the agent operates within larger environments of diverse sizes and configurations of enemies, with the overarching objective of reaching designated goal positions.

The evaluation of the proposed approach is facilitated through the creation of a game environment. The game involves an agent navigating an environment to reach a goal position, without having any prior knowledge of the environment's layout. An in-depth comparison is conducted between the performance of the classic Q-Learning algorithm and the novel Causal Q-Learning algorithms; it is achieved through an extensive experimental campaign encompassing various scenarios, ranging from easy to challenging. The comparison of results, focuses on several key metrics, including the average reward per episode step, the number of steps necessary to finish an episode, the average time taken to complete a game using each designed approach, the frequency of defeats for each algorithm, and the occurrences of timeout conditions for each algorithm. The experimental results highlight the significant performance advantage of agents equipped with causal knowledge compared to those using only classic Q-Learning. Notably, the performance gap widens as task complexity increases.

The importance of this endeavor rests in the advancement of Causal Reinforcement

Learning and Causal Artificial Intelligence, which adds to the progress of scientific research by introducing a novel class of AI algorithms. These algorithms are able to understand and evaluate complex environments, all while maintaining the capacity to continuously acquire new concepts. The learning and comprehension mechanisms of these Causal Reinforcement Learning algorithms closely mirror human cognitive processes.

Contents

Nomenclature

AI	Artificial Intelligence
ML	Machine Learning
DL	Deep Learning
RL	Reinforcement Learning
MDP	Markov Decision Process
TD	Temporal Difference
MC	Monte Carlo
DP	Dynamic Programming
BL	Bayesian Learning
BN	Bayesian Network
QL	Q-Learning
CQL	Causal Q-Learning

List of Tables

List of Algorithms

List of Figures

Chapter 1

Introduction

Reinforcement Learning (RL) stands as a prominent facet of artificial intelligence, focusing on the acquisition of sequential decision-making policies to maximize cumulative rewards [5]. While the traditional RL approach centers on individual-agent scenarios, real-world contexts often involve multiple agents interacting within a shared environment. These shared environments manifest in diverse applications like IoT, Smart Homes, Personal Assistants, Smart Factories, Robotics, and Self-Driving Cars. Such scenarios need to learn from mistakes, a common human endeavor, while simultaneously comprehending the environmental outcomes resulting from taken actions [7].

Simultaneously, Causal Models provide a strong framework for comprehending and reasoning about causal associations [8], able to enhancing RL algorithms in both single-agent and multi-agent settings. Finally, the difference between machine/deep learning and causal models lies in the absence of a reality model for the former. Just like thirty years ago, machine learning programs (including those with deep neural networks) operate almost entirely in an associative mode. They continue to improve in accuracy as more data are fitted, but they do not benefit from the "super-evolutionary speedup" [9]; the integration of artificial intelligence algorithms and causal models can achieve this significant acceleration.

Furthermore, the combination of data-driven models with model-based counterparts presents a fascinating strategy for addressing complex problems involving wide datasets and sequential decision-making. The development of such applications carries substantial potential for enhancing decision trustworthiness, augmenting the explainability of decisions made.

An active research group in the domain of Causal RL is the *CausalAI Lab*[1] at Columbia University, under the leadership of Elias Bareinboim. Over recent years, the lab has pioneered the development of the Causal RL framework, merging the structural invariances of causal inference with the sampling efficiency inherent in reinforcement learning.

The focal point of this thesis resides in exploring the fusion of data-driven models, particularly reinforcement learning, with model-based paradigms such as causal models. The objective is to advance the creation of explainable solutions, encompassing techniques for both single-agent systems and multi-agent systems concerning coordination, communication, and generalization. This novel category of algorithms is aptly termed *Causal Reinforcement Learning*, which holds the promise of revolutionizing the contem-

[1] `https://causalai.net/`

porary landscape. These algorithms exhibit strong adaptability, leveraging a blend of reinforcement learning's trial-and-error approach and the causal knowledge engendered by causal models to understand and predict actions' repercussions. The core objectives of this master's thesis encompass the creation of novel and trustworthy solutions tailored for reinforcement learning scenarios. These solutions are distinguished by their robust explainability, remarkable flexibility, and expansive generalization capabilities. Additionally, the thesis delves into investigating the integration of causal models into single-agent scenarios, empowering effective decision-making within dynamically evolving environments. An integral facet of the thesis revolves around the development of innovative data-driven models, notably reinforcement learning algorithms. These algorithms draw upon previously acquired causal knowledge, allowing them to enhance the coordination, efficiency, and decision-making potential of both single-agent (and multi-agent, not faced in this thesis) systems.

To effectively evaluate the devised algorithms, a versatile game environment has been defined. This environment's formulation is parameterized, capable of transforming into either a grid-based or maze-like structure of varying dimensions and varying number of enemies. The final algorithm itself comprises two distinct components: the first component pertains to the extraction of causal knowledge from a generic environment; this causal understanding empowers the agent to discern the ramifications of its actions, thereby comprehending the factors leading to game over scenarios. The second component involves the agent actively participating in the actual game environment, wherein it is tasked with reaching its goal, although without knowing in advance the position of the target or even the number of enemies.

The acquired results yield interesting insights. By utilizing conventional RL metrics, it becomes evident that the agent guided by causal knowledge attains superior performance compared to its counterpart lacking such knowledge. Significantly, the most noteworthy emphasis is that the intensification of the complexity of the task leads to the improvement of the performance for the causal knowing agent. This salient trend indicates that as challenges become more complex, the agent's possession of causal knowledge continues to translate into increasingly enhanced outcomes.

The structure of this thesis is organized within the following framework:

1. *Background*, this chapter serves as a foundational introduction, clarifying key concepts pertinent to the thesis. It delves into the overarching field of Artificial Intelligence (AI), outlines the characteristics of Reinforcement Learning (RL), elucidates the structuring of RL problems, highlights specific RL algorithms embraced within this thesis, exhibits the implementation of causal models, and assesses the advantages and limitations inherent to such models.

2. *Literature Review*, this chapter undertakes a comprehensive examination of the various works explored in the course of this thesis. It accentuates the salient aspects of each work, identifying points of interest and relevance to the present research.

3. *Minigame*, within this chapter, the construction of the game environment is documented. It encompasses the process of building the game, details the methodology employed for implementing the causal model crucial for extracting causal knowledge, and expounds on the work of the Causal Q-Learning algorithms deployed.

4. *Experiments and Results*, this chapter serves to discuss and analyze the results obtained. The collected findings are reviewed and analyzed, offering insights into the implications and significance of the findings.

5. *Conclusions*, the concluding chapter encapsulates the entirety of the thesis, summarizing its various facets and contributions. It presents a comprehensive synthesis of the work conducted, highlighting key takeaways and potential avenues for future research.

In addition, the appendix section features a concise summary of the thesis in the Italian language.

Chapter 2

Background

In this chapter, a background on the topics covered in the research is provided, in order to give the tools useful for understanding the thesis.

2.1 Artificial Intelligence

What is *Artificial Intelligence* (AI)? What can AI do? How does AI work?

2.1.1 Introduction and Canonical Definitions

At European level, following the great rise of AI that we are witnessing in recent years, the European Union commission, in 2018, provides a definition of AI: *"Artificial Intelligence refers to systems that show a clever behaviour, by analyzing the environment and by doing actions, to achieve specifics goals with a certain level of autonomy."*

Moreover, it is possible to find also the definition given by Wikipedia: *"Artificial Intelligence is a discipline belonging to computer science that studies the theoretical foundations, methodologies and techniques that allow the design of hardware systems and software program systems capable of providing the electronic computer with performances that, to a common observer, would seem belong exclusively to human intelligence."* [10]

Inside [1], through a table reported in Figure 2.1, principal AI definitions, available up to that time, are given by representing them with four categories/approaches. The definitions on the left measure success by similarity to a *human* performance, while those on the right use the ideal concept of intelligence, which we generally call **rationality**, as a term of comparison. A system is rational if, given its knowledge, it "does the right thing".

2.1.2 Building Intelligent Machines to Transform Data into Knowledge

In this technological age, there is a resource available in large quantities: there are several structured and not structured data; this aspect was very helpful for the AI development, Machine Learning (ML), one of the AI fields, regard the development of self-learning algorithms able to acquire/extract knowledge from the data, for the purpose of making predictions. [11]

Systems that think like humans	Systems that think rationally
"The exciting new effort to make computers think ... *machines with minds,* in the full and literal sense." (Haugeland, 1985)	"The study of mental faculties through the use of computational models." (Chamiak and McDermott, 1985)
"[The automation of] activities that we associate with human thinking, activities such as decision-making, problem solving, learning ..." (Bellman, 1978)	"The study of the computations that make it possible to perceive, reason, and act." (Winston, 1992)
Systems that act like humans	**Systems that act rationally**
"The art of creating machines that perform functions that require intelligence when performed by people." (Kurzweil, 1990)	"Computational Intelligence is the study of the design of intelligent agents." (Poole *et al.*, 1998)
"The study of how to make computers do things at which, at the moment, people are better." (Rich and Knight, 1991)	"AI ...is concerned with intelligent behavior in artifacts." (Nilsson, 1998)

Figure 1.1 Some definitions of artificial intelligence, organized into four categories.

Figure 2.1: *Artificial Intelligence definitions, four main categories [1].*

Tom Mitchell, an American computer scientist and the Founders University Professor at Carnegie Mellon University (CMU), provided a definition of ML that has become popular: "*A computer program is said to learn from experience E with respect to any task class T and performance measure P if its task exception in T, measured at P, improves experience E.*" [12]

Machine Learning is closely related to pattern recognition and to computational learning theory, it explores the study and the construction of algorithms that could learn from data and that could making predictions on these, by inductively building a model based on samples.

Main Paradigms of Machine Learning

The three main ML paradigms[1] are reported in this list, they differ from each other in the type of learning adopted.

- **Supervised Learning** (SL), its main purpose is to build a model from labeled training data, which allows you to make predictions about unavailable or future data. The term "supervised" refers to the fact that in the set of samples, the desired output signals (the labels) are already known.
- **Unsupervised Learning** (UL),on the contrary, we are dealing with unlabelled data or data with an unknown structure. Using unsupervised learning techniques we are able to observe the structure of our data, to extract from them information full of meaning without however being able to rely on the guidance of a known variable relating to the result, nor a reward function.

[1] A **paradigm**, in common parlance, it is a reference model, a term of comparison.

Application	AI Planning	SL	UL	RL	IL
Optimization	x			x	x
Learning from experience		x	x	x	x
Generalization	x	x	x	x	x
Delayed consequence	x		x	x	
Exploration				x	

Table 2.1: *Comparison of possible applications of ML paradigms.*

- **Reinforcement Learning** (RL), has the objective of developing a system (*agent*) that improves its performance on the basis of interactions with the environment; is the paradigm with which we worked in this thesis.

Figure 2.2 reports a conceptual graph related to cited ML paradigms with their applications whereas, Table 2.1 provides an overview of possible application scenarios, to which it can be added a fourth technique called **Imitation Learning (IL)**, where agents have access to samples related to expert behaviours.

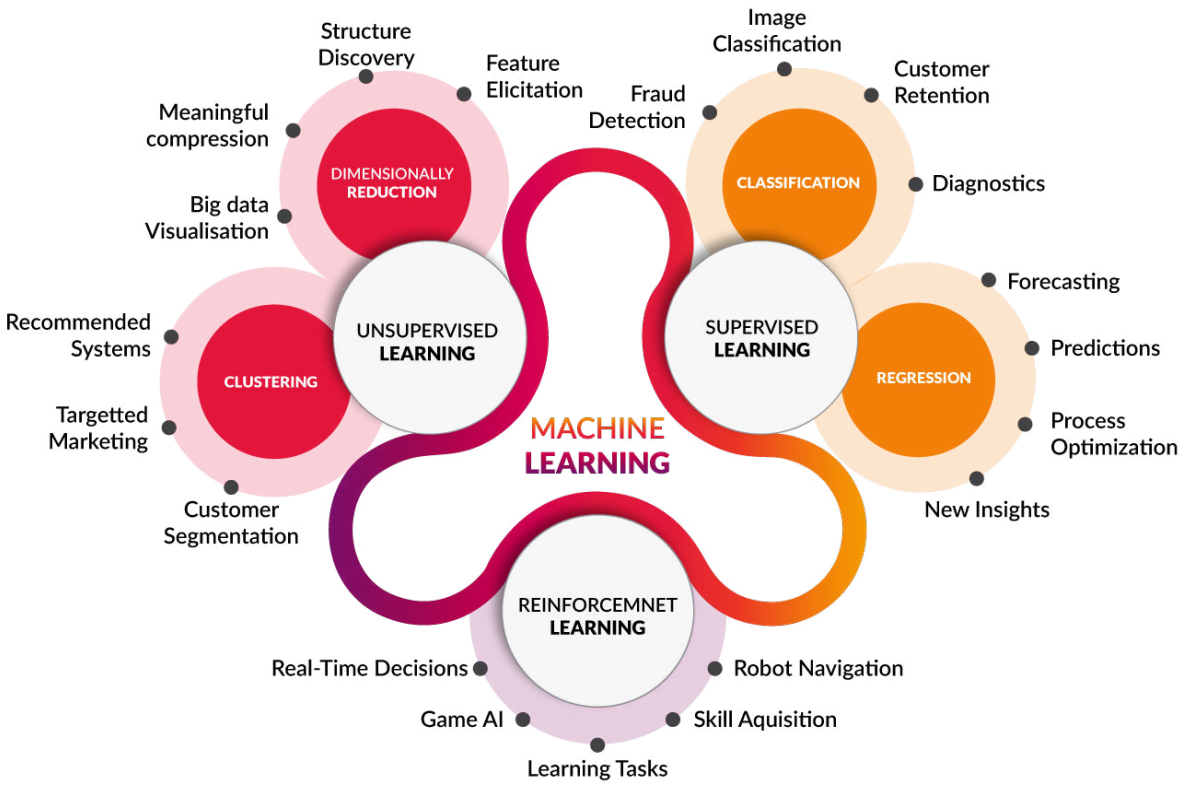


Figure 2.2: *Main ML paradigms with related fields of application [2].*

2.2 Reinforcement Learning (RL)

2.2.1 Overview

The idea that one learns by interacting with an environment is probably the first that comes to mind when thinking about the nature of learning. RL is to learn what to do - how to map situations to actions - in order to maximize a numerical reward signal. The agent who wants to learn is not told which actions to take, instead he must find out which ones give the greatest reward by *trying them*. In the most interesting and challenging cases, actions can influence not only the immediate reward but also the subsequent situation and, through it, all subsequent rewards. These two features, **trial search** and **delayed reward**, are the two most important distinguishing features of RL.

An RL problem is formalized using concepts deriving from the theory of dynamical systems, in particular that relating to the optimal control of not completely known Markov decision-making processes (MDP). The idea behind this type of problem is that the most important aspects of the real problem faced by a learning agent who interacts over time with its environment to achieve a goal are captured. Clearly, such an agent must be able to perceive the state of the environment to some extent and must be able to take actions that affect the state. The agent must also have one or more objectives related to the state of the environment. The formulation intends to include only these three aspects, **feeling**, **action** and **purpose**, in their simplest possible forms without trivializing any of them. [5]

Another definition of RL is provided by Sergey Levine, an associate professor in the Department of Electrical Engineering and Computer Science at UC Berkeley:
"*Reinforcement learning in general is learning to act through trial and error with no models, labels, demonstrations or supervisory cues provided plus delayed rewards for the agent's actions.*"

2.2.2 Applications

What has been described so far bears proofs to the fact that reinforcement learning has a truly interesting and vigorous underlying idea: finding the best solution to a problem, learning more and more from the mistakes made (more or less what we human beings should do everyday).

RL systems, as they are defined, can be adopted in many applications. In [7] it is possible to find an exhaustive description of the RL theory and a detailed list of its possible applications with accurate descriptions; in particular in the field:

- **Healthcare**, the use of machine learning (ML) has shown extraordinary results in the field of healthcare. Initially, researchers focused on using ML algorithms to diagnose conditions or predict treatment outcomes. These tasks are also important, but determining the best treatment policy to use for a specific patient is a challenging task and cannot be accomplished using traditional approaches. Reinforcement learning is a promising candidate for solving such human health problems as well.

- **Robotics and Autonomous Driving**, many autonomous driving and robotics problems can be modeled as a RL problem. The trial-and-error mechanism of the RL helps the robot and the control system to autonomously learn the optimal behavior by interacting with the environment.

- **Autonomous IoT**, the integration of the IoT with an autonomous control system, leads to what is called AIoT system (Autonomous Internet of Things). RL algorithms introduce *ambient intelligence* into IoT systems, providing the tools to solve closed-loop problems to process sensory data, in particular, these algorithms are used to make decisions.

- **Video-games and Self-Organizing Systems**, the RL is having a fundamental contribution for the creation of efficient algorithms in the video game field; many times the work is done by training the agent to interact with the environment using certain characteristics in order to develop a super intelligence. Also RL systems have recently surpassed human intelligence and reached super intelligence in strategy games. These strategic features can also be useful in self-organizing systems (such as cellular networks etc.) to improve their performance.

- **Communication and Network**, the RL, and even more the Deep RL, can be applied to solve many challenges and problems in telecommunications and data exchange. Modern architectures such as UAVs (Unmanned Aerial Vehicles), Heterogeneous Networks (HeNets) and IoT (Internet of Things) can be autonomous, ad-hoc designed and decentralized if the entities present in the network are able to make autonomous decisions .

Please refer to the article for further information. The Figure 2.3 also shows a very interesting image, which encompasses the macro-areas where it is possible to apply the RL algorithms and where these can provide great results. As further evidence of the importance and scientific interest that reinforcement learning is having, the graph relating to the number of articles published per year in recent years is shown in Figure 2.4; this growth is due, similarly to ML and DL, to the possession of ever more powerful and affordable computing units.

2.2.3 Markov Decision Processes (MDPs)

Markov Decision Processes (MDPs) are a classical formalization of sequential decision making, where actions influence not just immediate rewards, but also subsequent situations, or states, and through those future rewards. Thus MDPs involve delayed reward and the need to trade-off immediate and delayed reward.

MDPs are a mathematically idealized form of the reinforcement learning problem for which precise theoretical statements can be made.

Agent-Environment Interface

MDPs are meant to be a straightforward framing of the problem of learning from interaction to achieve a goal. The learner and decision maker is called *agent*. The thing it interacts with, comprising everything outside the agent, is called *environment*. These interact continually, the agent selecting actions and the environment responding to these actions and presenting new situations to the agent [2]. The environment also gives rise to

[2]Terms *agent*, *environment* and *action* are used instead of adopting the terms *controller*, *controlled system* and *control signal*; but it is the same conceptual framework.

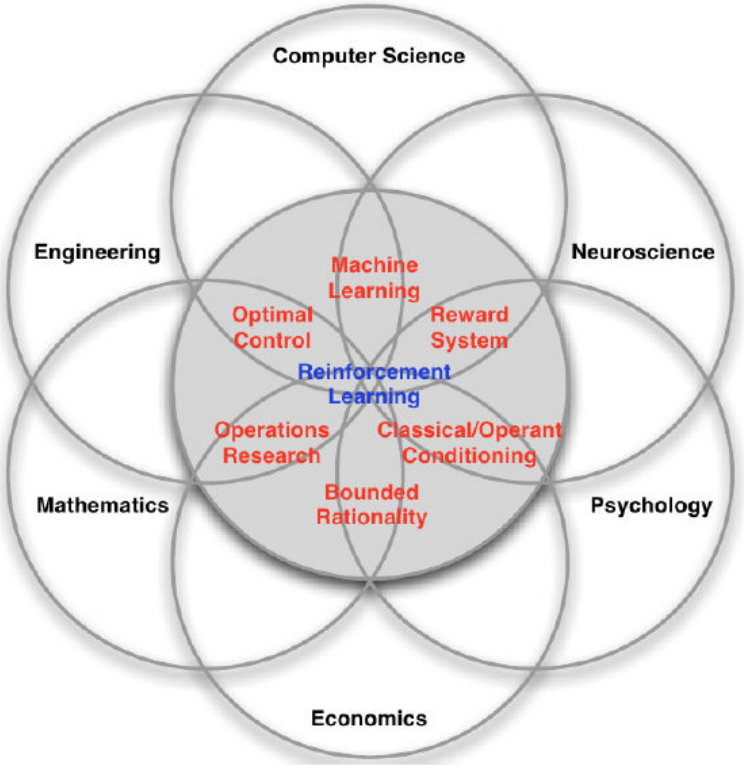


Figure 2.3: *Fields of application of Reinforcement Learning [3].*

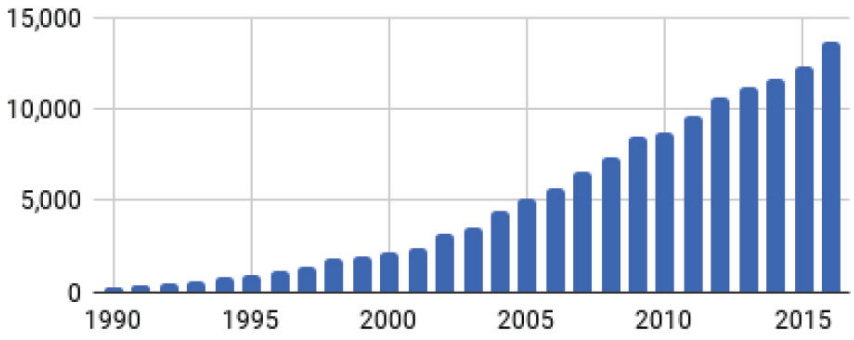


Figure 2.4: *Number of publications per year in the field of Reinforcement Learning [4].*

rewards, special numerical values that the agent seeks to maximize over time through its choice of actions.

More specifically, in first approximation, you can consider the interaction between agent and environment of a sequence of a discrete time steps. At each time step t, the agent receives some representation of the environment's **state**, $S_t \in \mathbb{S}$, and on that basis selects and **action**, $A_t \in \mathbb{A}(s)$. One time step later, in part as a consequence of its action, the agent receives a numerical **reward**, $R_{t+1} \in \mathbb{R}$, and finds itself in a new state, S_{t+1}. This process is summarized in Figure 2.5.

The transition of the process into its new state, denoted as s', is directly impacted by the chosen action. This influence is precisely quantified by the *state transition function* $P_a(s, s')$. Consequently, the subsequent state s' is reliant upon both the present state s

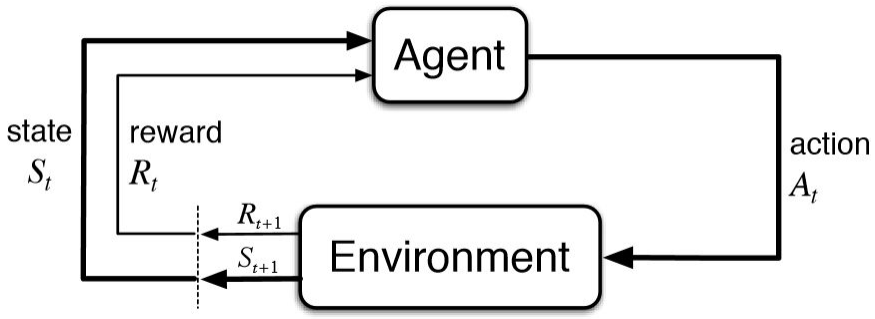


Figure 2.5: *The agent-environment interaction in a Markov Decision Process [5].*

and the action a taken by the decision maker. However, it is important to note that given the current state s and action a, the next state s' becomes conditionally independent of all preceding states and actions. In simpler terms, the progression of states in a Markov Decision Process (MDP) adheres to the *Markov property*, which means that the current state encapsulates all the pertinent information from past interactions between the agent and the environment that will impact future outcomes.

The MDP framework is abstract and flexible and can be applied to many different problems in many different ways.

Goals and Rewards

Within the domain of Reinforcement Learning (RL), the agent's objective or target is expressed through a distinctive indicator known as the *reward*. This signal is conveyed from the environment to the agent, with each time step providing a singular numerical value, $R_t \in \mathbb{R}$. Informally, the agent's ambition can be distilled as the pursuit of amplifying the aggregate reward it collects. This pursuit doesn't solely revolve around immediate rewards, but rather centers on the accumulation of rewards over an extended span. This underlying principle can be concisely phrased as the *reward hypothesis. That all of what we mean by goals and purpose can be well thought of as the maximization of the expected value of the cumulative sum of a received scalar signal (called reward).*

Employing a reward signal to formalize the concept of an objective stands out as one of the most distinguishing attributes of Reinforcement Learning (RL). The agent's consistent aim is to optimize its reward. To accomplish this, it is essential to supply the agent with rewards in a manner that aligns with our overarching objectives when maximized. Therefore, ensuring that the rewards we establish accurately reflect the desired accomplishments becomes paramount. It's worth emphasizing that the reward signal is not intended to impart prior knowledge to the agent regarding the procedural methods for attaining our goals. Rather, the reward signal serves as a means to convey *what* outcomes are desired, not *how* these outcomes are to be achieved.

Currently, the quest to identify the optimal reward framework remains an ongoing challenge. Notably, even Richard Sutton dedicated a chapter in his book specifically to try to address this matter.

2.2.4 Exploration vs. Exploitation Trade-off

One of the distinctive challenges that emerges in Reinforcement Learning (RL), as opposed to other types of learning, is the intricate balance between exploration and exploitation. In order to obtain significant rewards, an RL agent must demonstrate a preference for actions it has previously tested and confirmed as effective in generating rewards. However, in the pursuit of identifying such actions, the agent must also venture into uncharted state by trying actions it has not selected before. This creates a dynamic where the agent must both *exploit* its established experiences to secure rewards and *explore* new possibilities to enhance future decision-making. The complexity lies in the fact that exclusive dedication to either exploration or exploitation leads to sub-optimal outcomes.

Facing this challenge requires the agent to continually experiment with various states, gradually favoring those that show the most promise. Despite extensive study by mathematicians over decades, the exploration-exploitation predicament on stochastic tasks continues to elude a definitive resolution. Presently, it is essential to acknowledge that this challenge persists, unlike in supervised and unsupervised learning, where the dynamics tend to be more straightforward in their purest manifestations.

ϵ-Greedy Method

The epsilon-greedy method is a fundamental exploration strategy employed in reinforcement learning, particularly in algorithms like Q-Learning. It balances the trade-off between exploration and exploitation by guiding an agent's actions in an environment. The core idea is to ensure a balance between selecting actions that are believed to yield the highest immediate rewards (exploitation) and choosing actions that have not been extensively tried before (exploration).

2.2.5 Temporal Difference Learning

Temporal Difference (TD) learning is a combination of Monte Carlo[3] ideas and dynamic programming[4] ideas. Like MC methods, TD methods can learn directly from raw experience without a model of the environment's dynamics. Like DP, TD methods update estimates based in part on other learned estimates, without waiting for a final outcome (they bootstrap).

Simple every-visit Monte Carlo method suitable for non-stationary environments is

$$V(S_t) \longleftarrow V(S_t) + \alpha[G_t - V(S_t)] \tag{2.1}$$

where G_t is the actual return following time t, and α is a constant step-size parameter. Let us call this method *constant-α MC*.

Whereas MC methods must wait until the end of the episode to determine the increment to $V(S_t)$ (only then is G_t known), TD methods need to wait only until the next

[3] *Monte Carlo* (MC) methods are a broad class of computational algorithms that rely on repeated random sampling to obtain numerical results. The underlying concept is to use randomness to solve problems that might be deterministic in principle

[4] *Dynamic programming* (DP) refers to a collection of algorithms that can be used to compute policies given a perfect model of the environment as a MDP. The key idea of DP is the use of value functions to organize and structure the search for good policies.

time step. At time t+1 they immediately form a target and make a useful update using the observed reward R_{t+1} and the estimate $V(S_{t+1}$. The simplest TD method makes the update

$$V(S_t) \longleftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1} - V(S_t)] \tag{2.2}$$

immediately on transition to S_{t+1} and receiving R_{t+1}. In effect, the target for the MC update is G_t, whereas the target for the TD update is $R_{t+1} + \gamma V(S_{t+1})$. This TD method is called *TD(0)*, or *one-step TD*, because it is a special case of the TD(λ).

TD methods update their estimates based in part on other estimates. They learn a guess from a guess, hey bootstrap. What advantage do TD methods have over MC and DP methods?

TD methods have an advantage over DP methods in that they do not require a model of the environment, of its reward and next-state probability distributions.

The subsequent notable advantage of Temporal Difference (TD) methods compared to Monte Carlo methods lies in their innate capability for online, incremental execution. In contrast to Monte Carlo methods where the return becomes known only at the episode's conclusion, TD methods necessitate waiting for only a single time step. This unexpectedly proves crucial in many scenarios. Some applications entail lengthy episodes, rendering the postponement of all learning until an episode's culmination impractical. Other applications entail ongoing tasks that lack distinct episodes. Additionally, as previously highlighted, certain Monte Carlo methods must either disregard or assign less weight to episodes involving experimental actions, causing a notable deceleration in learning. TD methods exhibit greater resilience to these challenges as they assimilate knowledge from each transition, regardless of the subsequent actions taken.

Convergence and Optimality of TD(0)

Let us consider a scenario where only a finite amount of experience is available—perhaps 10 episodes or 100 time steps. In such cases, a common strategy involving incremental learning methods is to repeatedly present the available experience until the method converges towards a solution. Assuming an approximate value function, V, the increments defined by Equations 2.1 and 2.2 are computed for each time step, denoted as t, in which a non terminal state is encountered. However, the actual alteration of the value function occurs only once, wherein the summation of all increments contributes to this change. Subsequently, all available experience is once again processed using the updated value function, yielding a new comprehensive increment. This process iterates until the value function reaches convergence.

This iterative technique is termed *batch updating* as updates are performed exclusively after processing each complete batch of training data. When applying batch updating, the TD(0) method converges deterministically towards a specific solution, irrespective of the step-size parameter (denoted as α), provided *alpha* is judiciously chosen to be sufficiently small. Similarly, the constant-α MC method also achieves deterministic convergence under the same conditions, albeit yielding a distinct solution. Understanding these two outcomes is pivotal in grasping the divergence between these two methods.

It is important to note that during regular updating, the methods do not directly attain their respective batch answers. Instead, they can be seen as taking steps in the directions

of these batch answers to some extent. Prior to delving into a general understanding of the two outcomes across all potential tasks, let us first explore a few examples for clarification.

On-Policy and Off-Policy Methods

A **policy** serves as the agent's strategic approach for achieving its objectives. Consequently, a policy is called *optimal* if, across all states, the expected reward is either greater than or equal to rewards obtained through alternative policies. Two primary categories characterize policy methods:

- *On-Policy methods* involve evaluating or enhancing the policy that is directly employed for decision-making.
- *Off-Policy methods* encompass two distinct policies: the *behavioral policy* that guides the agent's interaction with the environment, and the *target policy* that is learned and optimized. The behavioral policy dictates the agent's actions in various states, thus driving the decision-making process. Conversely, the target policy serves as the basis for learning from the rewards generated by the agent's actions. This information is then employed to update a component that integrates past actions and rewards.

The fundamental distinction lies in the relationship between the behavior policy and the target policy. In on-policy methods, these policies are identical (there's only one), while off-policy methods employ two distinct policies. On-policy methods are applicable to both model-based and model-free RL, whereas off-policy methods find their utility primarily in the model-free RL context.

Q-Learning Algorithm: Off-Policy TD Control

Q-Learning is one of most known reinforcement learning algorithms [13], it is is a form of model-free RL. It can also be viewed as a method of asynchronous DP. It provides agents with the capability of learning to act optimally in Markovian domains by experiencing the con- sequences of actions, without requiring them to build maps of the domains.

First of all, you must define an entity aims to store the expected cumulative reward of taking an action in a particular state, this entity is called *Q-Table*; a Q-Table in the context of Q-Learning is a fundamental data structure used to represent and store action-value estimates for different state-action pairs in a reinforcement learning environment. Q-Learning is a model-free reinforcement learning algorithm that aims to find an optimal policy for an agent to maximize its cumulative rewards over time.

The Q-Table is essentially a matrix where rows correspond to different states in the environment, and columns correspond to different actions that the agent can take. Each cell in the Q-Table holds an estimate of the expected cumulative reward that the agent can achieve by taking a specific action from a specific state. This estimate is known as the Q-value. An example is given in Figure 2.6. During the Q-Learning process, the agent interacts with the environment, observes the outcomes of its actions, and updates the Q-Table based on the rewards it receives. The Q-Table is updated using the Bellman equation, which combines the immediate reward received with the estimated maximum

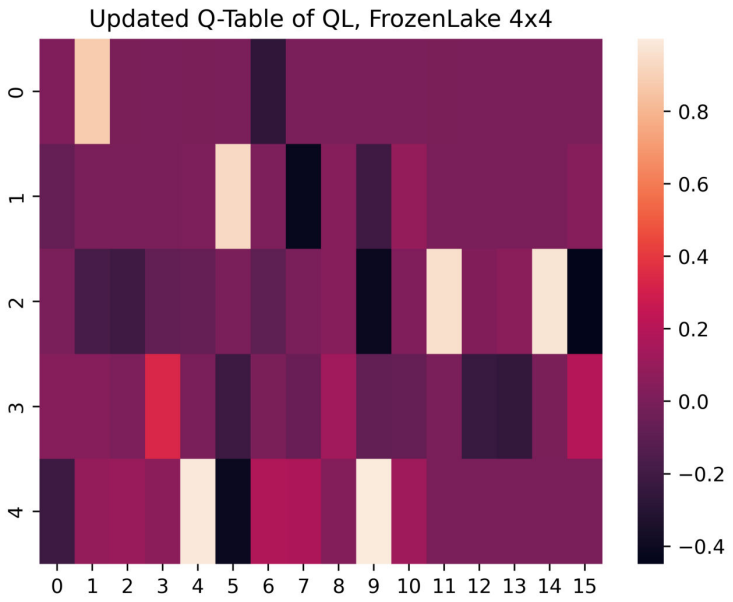


Figure 2.6: *Example of a Q-Table generated in this thesis, implemented in a Frozen Lake-like environment. The agent plays within this grid, having 5 distinct actions available.*

future rewards from the next state. This iterative update process helps the Q-Table converge to the optimal Q-values over time, allowing the agent to make better decisions. As the agent learns more about the environment through exploration and exploitation, the Q-Table becomes a valuable reference for deciding which actions to take in different states. Ultimately, the agent's policy for action selection is guided by the Q-Table's Q-values, helping it make informed decisions to achieve its goals efficiently.

The Q-Learning algorithm is reported below (Algorithm 1), pay attention to the Q-Table update step, it updates a state-action pair (Equation 2.3).

This algorithm was chosen for the development of this thesis due to its fairly simple construction, so as to be able to manipulate it in a not too complex way.

In the context of Q-Learning, which aims to find the optimal policy for an agent in a given environment, the *epsilon-greedy method* involves the following steps:

1. *Exploration* $\rightarrow$ random action: with probability ϵ, the agent chooses a random action from the available action space. This encourages the agent to explore unfamiliar actions and states, facilitating a more comprehensive understanding of the environment.

2. *Exploitation* $\rightarrow$ best-known action: with probability (1 - ϵ), the agent selects the action that corresponds to the highest estimated Q-value for the current state. This action maximizes the agent's short-term reward and is based on the agent's current knowledge of the environment.

The parameter ϵ is crucial in controlling the balance between exploration and exploitation. A higher value of ϵ encourages more exploration, which can be beneficial in the initial stages of learning or when dealing with uncertain environments. As the agent's

Algorithm 1 Q-Learning Algorithm

Require: Initialize arbitrarily Q(s, a), for all s $\in \mathbb{S}^+$, a $\in \mathbb{A}(s)$
  Step size (learning rate) $\alpha \in (0, 1]$
  Discount factor $\gamma <\sim 1$
  for each episode **do**
    Initialize state S
    while each step for episode until S is terminal **do**
      Choose A from S using policy derived from Q (e.g., ϵ-greedy)
      Take action A, observe R, S'
      Update:

$$Q(S_t, A_t) \longleftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S', a) - Q(S_t, A_t)] \tag{2.3}$$

      S $\longleftarrow$ S'
    end while
  end for

knowledge grows, decreasing ϵ gradually shifts the focus towards exploitation, as the agent relies more on the learned Q-values.

Challenges and Extensions of Q-Learning

Below some challenges and extensions of Q-Learning algorithm are reported:

- *Exploration-Exploitation Trade-off*, as described previously, Q-Learning faces the challenge of balancing exploration (trying new actions to discover their rewards) and exploitation (choosing actions based on known high rewards). Striking the right balance is crucial for effective learning and optimal policy discovery.

- *Large State Spaces*: Q-Learning's Q-Table grows with the number of states and actions, making it impractical or infeasible for environments with large state spaces. Techniques such as function approximation or deep reinforcement learning (DRL) can address this challenge by using neural networks to estimate Q-values.

- *Continuous Action Spaces*: Q-Learning is inherently designed for discrete action spaces. Adapting it to continuous action spaces requires modifications, such as discretization or using algorithms like Deep Deterministic Policy Gradients (DDPG).

- *Delayed Rewards and Sparse Rewards*: Q-Learning can struggle when rewards are delayed or sparse. It might fail to associate actions with long-term consequences. Techniques like reward shaping, eligibility traces, and prioritized experience replay can help address this issue.

- *Non-Stationary Environments*: in environments where dynamics change over time (non-stationary), Q-Learning might have trouble adapting. Techniques like online learning, adaptive learning rates, and incorporating uncertainty can help improve performance in such cases.

- *Multi-Agent Reinforcement Learning*: Q-Learning's application in multi-agent environments can lead to challenges such as unstable learning due to the non-stationary nature of other agents. Algorithms like Multi-Agent DDPG (MADDPG) and Multi-Agent Actor-Critic (MAAC) extend Q-Learning to multi-agent scenarios [14].
- *Reward Design*: designing suitable reward functions is critical in Q-Learning. Poorly designed rewards can lead to sub-optimal or undesirable policies. Careful consideration of rewards is essential for successful Q-Learning.

2.3 Causal Inference

Before introducing some concepts related to causal inference, we talk about who is considered one of the pioneers of this current of thought, **Judea Pearl** [15]. Judea Pearl is a prominent computer scientist and philosopher known for his groundbreaking work in the field of artificial intelligence, particularly in the area of probabilistic reasoning, causal inference [16–18], and Bayesian networks. He has made significant contributions to developing algorithms and methodologies that enable computers to reason about uncertainty and make decisions in uncertain environments.

Judea Pearl's contributions have not only advanced the field of artificial intelligence but have also found applications in various disciplines, including medicine, social sciences, economics, and more. His work has significantly impacted the way researchers and practitioners approach complex systems, uncertainty, and causal relationships.

2.3.1 Introduction to Causal Inference

Causal Inference refers to the process of determining the cause-and-effect relationship between variables or events in a system. It aims to establish whether changes in one variable directly lead to changes in another, going beyond mere correlation to understand the underlying mechanisms that drive observed outcomes. Causal inference is vital for making informed decisions, predicting outcomes, and understanding the fundamental drivers of complex systems. This topic is of vital importance because it concerns many fundamental sectors, such as:

- *Medical research*, in clinical trials, causal inference is crucial for determining the effectiveness of treatments. Researchers assess whether a treatment leads to improved health outcomes compared to a control group, accounting for confounding factors.

- *Environmental Studies*: causal inference is used to understand the effects of environmental factors on ecosystems. It helps determine whether changes in environmental variables directly impact biodiversity, climate patterns, and ecological stability.

- *Policy Making*: causal inference guides evidence-based policy-making. By establishing causality, policymakers can design interventions and regulations that address specific problems effectively. This approach enhances the likelihood of positive outcomes

- *Artificial Intelligence*: in AI, causal inference is essential for building robust and interpretable models. It helps uncover causal relationships in complex data, guiding decisions and predictions. Causality can aid in understanding feature importance, making AI models more reliable and accountable.

- *Social Sciences*: causal inference is central to social sciences for exploring the effects of policies, interventions, and societal changes. Researchers assess the impact of educational reforms, economic policies, and social programs to understand their causal effects on outcomes like poverty reduction, educational attainment, and crime rates.

- *Epidemiology*: in epidemiology, understanding causal relationships is critical for identifying risk factors and interventions in public health. Causal inference helps determine whether certain factors, such as exposure to toxins or behaviors like smoking, directly lead to specific health outcomes, such as diseases.

In all these fields, causal inference is pivotal for making sound decisions, avoiding erroneous conclusions, and building accurate models of reality. It goes beyond observing correlations to unveil the intricate causal relationships that drive the phenomena under study.

Correlation and Causation

Correlation and causation are related concepts, but they have distinct meanings: *correlation* refers to a statistical relationship between two variables where changes in one variable tend to correspond with changes in another variable. A correlation indicates that there is a pattern of association between the variables. However, correlation alone does not imply a cause-and-effect relationship. Variables can be correlated without one causing the other. Correlation can be positive (both variables increase together) or negative (one variable increases as the other decreases). *Causation*, on the other hand, implies a direct cause-and-effect relationship between two variables. Changes in one variable lead to changes in the other variable. Establishing causation requires more than just observing a correlation; it requires evidence of a mechanism or process that connects the two variables in a way that one variable directly influences the other.

2.3.2 Causal Models and Causal Graphs

The concept of causal models serves as a powerful framework for representing and comprehending the intricate causal relationships that exist between variables in complex systems. Causal models provide a structured approach to understanding how changes in one variable can influence other variables and lead to specific outcomes. These models not only facilitate a deeper grasp of causal mechanisms but also offer a foundation for making predictions, performing interventions, and guiding decision-making in a wide range of domains.

One key method for visually representing causal relationships within causal models is through the use of directed acyclic graphs (DAGs), commonly referred to as causal graphs. These graphs serve as graphical tools that capture the cause-and-effect associations between variables while also depicting the directionality and dependencies among them. In a causal graph, nodes represent variables, and directed edges between nodes indicate causal relationships, with the direction of the edge indicating the direction of causation.

The acyclic nature of these graphs is crucial: the absence of cycles ensures that causal relationships can be unambiguously understood and prevents circular causality, where variables cause each other in a loop. This acyclicity aligns with the fundamental principle that causes precede their effects in time, promoting a coherent depiction of causality.

Components of Causal Graphs

Causal graphs provide a visual representation that helps researchers and analysts discern how changes in one variable can directly or indirectly influence other variables, thereby

shedding light on the underlying causal mechanisms at play. Furthermore, these graphs allow for the identification of confounding variables, which are common sources of bias and can lead to incorrect causal inferences. By explicitly illustrating the relationships and potential confounders, causal graphs offer a means to systematically account for such complexities.

In essence, the utilization of causal graphs in the context of causal models empowers us to not only represent causal relationships but also to scrutinize, manipulate, and explore the underlying mechanisms governing these relationships. As a result, they serve as a crucial tool for advancing our understanding of causality, supporting evidence-based decision-making, and fostering the development of sophisticated interventions in various fields such as medicine, social sciences, economics, and artificial intelligence.

2.3.3 Counterfactual and the Fundamental Problem of Causal Inference

The concept of counterfactuals is rooted in the idea of imagining what would have happened if a particular event or action had not occurred or had occurred differently. Counterfactuals refer to outcomes that would have materialized under alternative conditions or scenarios, distinct from the actual circumstances. They involve the comparison of observed reality with a hypothetical reality that did not occur, allowing us to reason about causality, make predictions, and assess the effects of interventions.

The fundamental problem of causal inference is intrinsically tied to counterfactuals. It revolves around the challenge of simultaneously observing both the treated and untreated states of a single unit or entity. In essence, this challenge stems from the fact that, in any given situation, we can only observe one outcome—the outcome that actually transpired. However, to comprehensively understand causal relationships and assess the impact of interventions, we require the ability to compare what happened with what could have happened under different conditions.

For example, consider a medical study evaluating the efficacy of a new drug. The goal is to determine whether the drug caused a change in health outcomes for patients. To do so, researchers need to compare the outcomes of patients who received the drug (the "treated" group) with the outcomes of those who did not receive the drug (the "untreated" group). This comparison allows them to isolate the causal effect of the drug itself, taking into account other variables that might influence the outcomes. However, the challenge arises because each patient can only be in one state—either treated or untreated. It is not possible to directly observe both potential outcomes for the same patient. This dilemma forms the core of the counterfactual problem in causal inference. How can we accurately estimate the difference between the treated and untreated states when we can only observe one of these states for each unit? To address this challenge, causal inference methods leverage statistical techniques, experimental designs, and assumptions to infer causal relationships from observed data. Counterfactuals play a crucial role in these methods, as they allow researchers to construct hypothetical scenarios and estimate the effects of interventions by comparing observed outcomes with what might have happened in alternative scenarios.

In summary, counterfactuals are essential in causal inference because they enable us to reason about the effects of interventions and to understand causality by comparing what

did happen with what could have happened under different conditions. The challenge of observing both treated and untreated states for a single unit underscores the complexity of causal inference and motivates the development of innovative methodologies to bridge this gap and unlock insights into cause-and-effect relationships.

2.3.4 Intervention and "Do" Operator

The concept of interventions is a fundamental aspect of causal inference, particularly when exploring cause-and-effect relationships in complex systems. Interventions involve deliberately manipulating certain variables within a system to observe the effects of specific actions. This process allows researchers to investigate how changes in one variable directly lead to changes in another variable, thereby unraveling the causal relationships at play.

In the context of intervention, two key terms are often used: *treatment variable (or intervention variable)* and the *outcome variable*. The first is the variable that is deliberately manipulated or changed during the intervention. It represents the specific action being taken to explore its causal effects. The second is the variable that researchers are interested in understanding how it responds to the intervention. It represents the effect of the action taken on the treatment variable.

Interventions enable researchers to move beyond merely observing correlations between variables and to determine whether a particular variable's changes directly cause changes in another variable. By manipulating variables in a controlled manner, researchers can establish a cause-and-effect relationship, allowing them to make more informed decisions and predictions about the system.

The *"do" operator* is a powerful notation used in causal inference to denote interventions and distinguish them from observational distributions. It's a way of explicitly indicating that a particular variable has been manipulated or set to a specific value, allowing us to reason about the causal effects of interventions.

Using the "do" operator, we can express statements like "$P(Y \mid do(X = x))$," which represents the probability distribution of variable Y given that we have intervened to set variable X to the value x. This explicitly captures the causal effect of changing X on the distribution of Y, rather than relying solely on observed correlations.

Do-Calculus and Causal Identification

The do-calculus is a set of rules and techniques used in causal inference to manipulate causal graphs and derive causal effects from both observational and interventional data. Causal graphs, often represented as directed acyclic graphs (DAGs), provide a graphical framework for expressing causal relationships between variables. The do-calculus helps harness the information embedded in these graphs to uncover cause-and-effect relationships, even in the presence of confounding factors.

The primary power of the do-calculus lies in its ability to reason about causal effects in complex systems under specific assumptions. These assumptions include:

1. *Causal Sufficiency*, the do-calculus assumes that the causal graph is sufficient to capture all relevant causal relationships between variables. This means that all confounding variables and causal paths are correctly represented in the graph.

2. *No measured confounders*, the do-calculus assumes that there are no unmeasured confounding variables—variables that are related to both the treatment and the outcome but are not explicitly included in the graph.

3. *Stable Unit Treatment Value Assumptions (SUTVA)*, this assumption asserts that the treatment assigned to one unit does not affect the potential outcomes of other units.

2.3.5 Causal Graphical Models and Bayesian Networks

Causal graphs and Bayesian networks are closely connected concepts, both serving as probabilistic representations of causal relationships within a system. They provide a graphical framework to visually and mathematically model the dependencies between variables and to make inferences about these dependencies using probabilistic reasoning.

Causal graphs are directed acyclic graphs (DAGs) that depict causal relationships among variables. In a causal graph, nodes represent variables, and directed edges represent causal links between them. This graphical representation highlights how changes in one variable directly influence others.

Bayesian networks, on the other hand, are a type of probabilistic graphical model that extends the idea of causal graphs to incorporate probabilistic information. In Bayesian networks, each node represents a variable, and edges not only depict causal relationships but also probabilistic dependencies. Conditional probability distributions quantify how each variable depends on its parents in the graph.

The connection between causal graphs and Bayesian networks lies in the fact that causal graphs can be used as the foundation for constructing Bayesian networks. Causal graphs provide the structural backbone that defines the causal relationships between variables, and Bayesian networks extend this structure to include probabilistic information, making them a versatile tool for causal inference.

The advantages of using probabilistic graphical models for causal inference can be:

- *Causal Reasoning*: probabilistic graphical models facilitate systematic causal reasoning by allowing researchers to represent and manipulate causal relationships explicitly. This enables them to ask and answer "what if" questions about interventions and counterfactual scenarios.

- *Clarity and Visualization*: both causal graphs and Bayesian networks offer a visual representation of complex causal relationships. This visual clarity helps researchers and practitioners understand the structure of the system and the dependencies between variables.

- *Interdisciplinary Applications*: probabilistic graphical models find applications in various fields, including medicine, economics, social sciences, and artificial intelligence. They provide a common framework for causal analysis across different domains.

- *Counterfactual Analysis*: probabilistic graphical models enable counterfactual analysis—examining the outcomes that would have occurred under different conditions or interventions.

- *Accounting for Uncertainty*: probabilistic graphical models naturally accommodate uncertainty by allowing the incorporation of probability distributions. This is essential for representing the inherent uncertainty in real-world causal relationships.

2.3.6 Resume

Understanding causal inference is crucial for making informed decisions, predicting outcomes, and uncovering the mechanisms that drive complex systems. Causal relationships are at the heart of various scientific disciplines, from medicine to social sciences to artificial intelligence. Accurate causal inference enables us to move beyond mere correlation, leading to better policy-making, improved interventions, and more reliable predictive models.

The do-calculus plays a pivotal role in advancing causal research by providing a systematic and formal framework for reasoning about causal relationships. It offers a set of rules and techniques to manipulate causal graphs, derive causal effects from data, and distinguish between observational and interventional distributions. The do-calculus bridges the gap between observational data and counterfactual scenarios, enabling researchers to make stronger claims about causal relationships in complex systems.

Chapter 3

Literature Review

This chapter describes the works related to the topics covered, extrapolating the highlights; in particular, the focus is placed on the use of reinforcement learning, causal models and bayesian networks.

3.1 Related Works

In the domain of machine learning, the work presented in [19] stands as a foundational contribution, delving into probabilistic graphical models and their crucial role in enabling intelligent systems to reason effectively amid uncertainty. The central focus of the book revolves around Bayesian networks, a powerful tool for representing and manipulating probabilistic relationships among variables. By harnessing probabilities, evidence, and causal links, these networks facilitate coherent and rational inference. The book highlights their significance in capturing intricate dependencies and aiding decision-making across various domains, including artificial intelligence, medicine, and engineering. Merging theoretical insights with practical applications, the work serves as a cornerstone reference for researchers, practitioners, and students seeking a profound grasp of probabilistic reasoning within intelligent systems.

In [20] a general-purpose system is proposed to build models of context and associated sense of agency; the experiments were performed in a smart-home scenario. The sense of agency is the subjective awareness of initiating, executing, and controlling one's actions in the world. The aim of the work is to lay the foundations for the construction of a complete structure for the definition and development of the sense of agency.

The approach to learning this awareness begins with the consideration of the fact that, without a specific assigned goal, an agent can freely explore the effects of his actions on the environment and on himself (what he perceives). The proposed approach aims to learn how to interpret the world model, where there are obviously dependencies between the actions one chooses to do and the variables one observes (consequences and more).

Three stages can be distinguished:

1. *Problem definition*, in which we consider an agent who chooses to implement a certain action, which will immediately produce a result observable by the agent. The observations (results) can be enclosed in a vector of variables which can be associated with a low-level internal sensor; just as, with an increasing level of abstraction, they can be associated with a higher level situation.

2. *Sense of agency ladder*, in which the learning of models of the world and the acquisition of a sense of agency can be conceived as an incremental process that takes place both at the individual and at the systemic (collective) level. The steps necessary for the development of each level are described in the table 3.1 and in the figure 3.1 how these are connected.

Individual Level	**Collective Level**
Perception	Recogniton of Non-Self
Exploration	Recognition of Self-Limitations
Planning	Strategic Thinking
	Multi-Agent Interaction

Table 3.1: *Individual and Collective levels: steps needed for the definition.*

3. *Structure learning of Bayesian Network*, in this approach, it is assumed that the agent has prior knowledge of what its actuators are, in this way the agent knows which are the variables representing the actuators, so that it can directly influence them (actions). This allows the agent to infer, after some experience, the existence of a cause-and-effect relationship between actions and consequences. In essence, the links present in the NL can reasonably be considered (to the best of the agents' knowledge) not simply as correlations, but as *causal relationships*.

Building upon Judea Pearl's insights on the *causal hierarchy* outlined in his work [17], the agent's awareness of the extent to which it can manipulate its environment through actuators transforms the process of learning causal relationships. It transcends the mere acquisition of associations (level 1 on the causality scale, P(y|x)) and evolves into the realm of *learning interventions* (level 2 on the causality scale, P(y|do(x), z)). This transition not only enriches the agent's learning experience but also introduces the potential for *counterfactual analysis* (level 3 on the causality scale), enabling the agent to reason and plan by exploring alternative scenarios that were not directly experienced. This progression from association to intervention to counterfactual analysis signifies a deeper engagement with the underlying mechanisms governing the environment, unlocking the agent's capacity to understand, influence, and anticipate outcomes in a more sophisticated manner.

Expanding upon the preceding discussion, an operational framework [6] has been formulated, illustrated in Figure 3.1, concerning the notion of *self-development*[1]. This concept is intimately tied to the concept of causal models, primarily due to the endeavor to engage with an entity of unknown attributes, striving to fathom its inherent traits.

Moreover, it is feasible to discern the essential methodologies for enacting self-development, grounded in the stages of effort underscored earlier within the operational framework—refer to Figure 3.2.

Furthermore, a demonstration of the application of causal models in a smart factory context is presented in [22]. In this scenario, a simulated manufacturing environment

[1]The term *self-development* is employed to denote the process undertaken by children during the initial phases of their existence [21]. Nevertheless, in a broader context, it can also be associated with the evolving nature of agents who exist and engage within an unfamiliar environment.

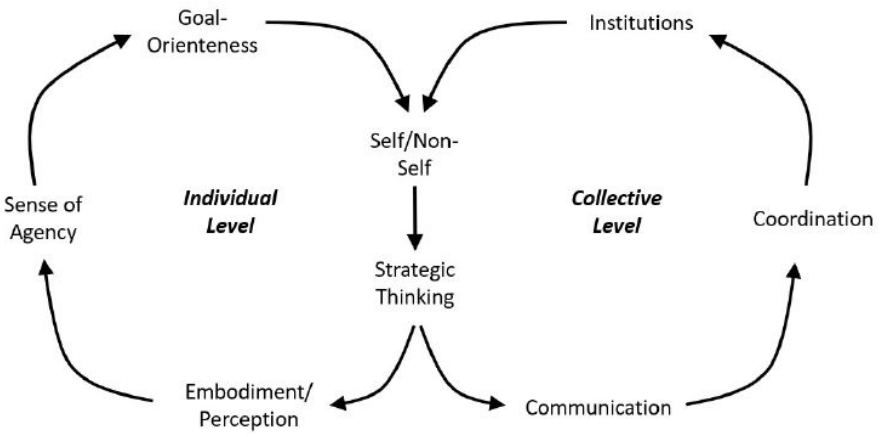


Figure 3.1: *Conceptual-operational structure of self-development [6].*

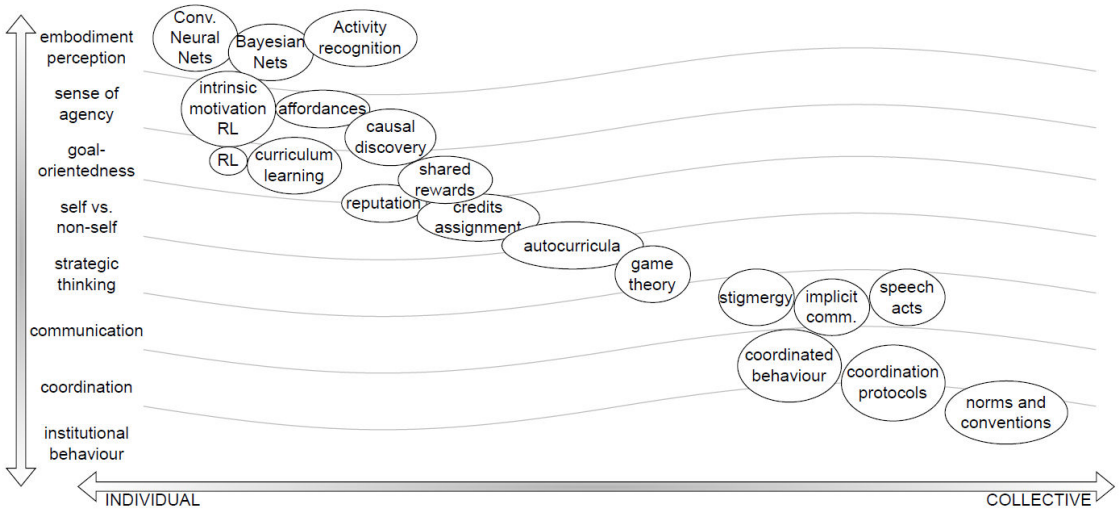


Figure 3.2: *Self-development: main techniques [6].*

is used as the backdrop, with a primary focus on acquiring causal representations of the environment. These causal models serve as interpretable frameworks that facilitate human understanding, aiding in analysis and decision-making processes. These frameworks are designed to capture the components and events of causality, abstracting away from the intricate physical implementations.

Notably, manufacturing processes often involve complex arrangements of physical devices such as sensors and actuators. The proposed hierarchical architecture offers a means to separate the responsibilities of individual components. Through the utilization of Digital Twins (DTs), each device can be effectively modeled. As a result, the hierarchical structure facilitated by DTs naturally introduces an abstraction layer. This abstraction allows for the linkage of variables and connections within the causal model to higher-level concepts.

The upper layers of this hierarchical arrangement possess a broader perspective on the events unfolding within the considered scenario. Consequently, these higher layers are adept at capturing dependencies between a more extensive set of variables, enabling a

more comprehensive understanding of the interrelationships at play.

Expanding upon Judea Pearl's concept of the "causal hierarchy," [23] underscores the pivotal role of causal reasoning within RL, particularly in medical challenges. This integration empowers RL agents to infer causal effects within complex real-world scenarios. The framework optimizes treatment strategies by accounting for unobserved confounders, as exemplified in the treatment of sepsis. Causal RL mirrors human learning, where agents learn causal relationships through interactions and optimize policies accordingly. This symbiosis between causal reasoning and RL offers two advantages: data efficiency and minimal change. While conventional RL demands extensive data, causal RL relies on the stability of causal graphs, resulting in efficient answers to interventions and counterfactual queries. The causal structure's robustness supports knowledge transfer, enabling agents to adapt and excel in diverse environments. This approach carries the potential to advance not only healthcare but also all RL domains, bridging causal reasoning and adaptive learning to enhance adaptability, generalization, and transfer in AI systems.

The study presented in [24] explores causal reasoning in the context of meta-reinforcement learning. The challenge revolves around understanding causal relationships within dynamic environments and leveraging this understanding to enhance an agent's adaptability across tasks. By integrating causal inference and RL, the research aims to refine decision-making processes in novel scenarios. This amalgamation empowers agents to learn causal effects through a series of tasks, subsequently improving their adaptability and decision-making capabilities. This work contributes to the exploration of how causal reasoning enriches meta-reinforcement learning, showcasing the role of causal relationships in enhancing an agent's generalization and adaptability across diverse tasks.

[25] introduces an innovative strategy for enhancing online reinforcement learning via counterfactual data fusion. By incorporating counterfactual reasoning into the learning process, the approach enriches the agent's decision-making by considering unobserved or alternative scenarios. This technique offers a more comprehensive perspective, aiding agents in handling dynamic environments and making informed decisions based on a wider range of information. The integration of counterfactual reasoning bridges reinforcement learning and causal inference, showcasing the potential of this fusion in refining learning outcomes. Through its application in online reinforcement learning, the approach highlights the significance of counterfactual data fusion in improving adaptability, decision-making accuracy, and overall performance.

[26] delves into the development of optimal dynamic treatment regimes using a causal reinforcement learning approach. The study addresses the challenge of designing treatment plans that adapt over time, particularly in medical contexts. By leveraging causal inference and reinforcement learning, the research aims to enhance the design of treatment strategies that cater to evolving patient conditions. This integration enables tailored treatment plans that consider causal relationships and optimize long-term outcomes. By combining these domains, the study contributes to the advancement of medical decision-making and underscores the significance of considering both causality and reinforcement learning for personalized healthcare interventions.

The paper [27] focuses on rendering reinforcement learning (RL) more interpretable and transparent by adopting a causal lens; it tackles the challenge of comprehending and explaining the decision-making process of RL algorithms, particularly in complex domains. The authors propose an approach that blends causal reasoning with RL to

enhance the transparency of RL models. By examining the causal relationships between actions and outcomes, this framework enables better insights into the rationale behind RL decisions. This causal perspective fosters understanding of the underlying mechanisms and factors driving agent behavior. The integration of causality and RL contributes to the field of explainable AI, bridging sophisticated RL techniques with transparency needs in AI systems.

[28] explores the fusion of Bayesian networks and reinforcement learning for regulating group emotions in response to sensory stimuli. The research tackles the challenge of managing emotions within a collective context and explores the combination of these techniques for effective emotional control. The authors propose an approach that employs Bayesian networks to model relationships between sensory stimuli and group emotions. Reinforcement learning is then applied to adapt and optimize emotional responses over time. By merging these methodologies, the study offers a framework for influencing and managing collective emotions. This fusion of Bayesian networks and reinforcement learning contributes to understanding and influencing collective emotions, showcasing the potential of combining probabilistic modeling and adaptive learning techniques in emotional regulation.

[29] introduces a methodology to enhance reinforcement learning from demonstrations using Bayesian network-based knowledge extraction. The study addresses the challenge of efficiently learning from demonstrated behaviors to improve reinforcement learning outcomes. The authors propose a technique that leverages Bayesian networks to extract knowledge from demonstrated actions and integrates this knowledge into the reinforcement learning process. By combining these techniques, the approach accelerates the learning process and enhances the agent's efficiency in acquiring optimal policies. This fusion of Bayesian networks and reinforcement learning contributes to the development of efficient learning paradigms, demonstrating how prior knowledge from demonstrations can expedite learning and improve reinforcement learning agent performance.

[30] explores the application of transfer learning in multi-armed bandit problems using a causal approach. The study addresses the challenge of transferring knowledge between different bandit tasks by leveraging causal reasoning. By understanding the causal relationships between actions and rewards, the approach aims to improve the efficiency of transferring learned policies to new scenarios. This fusion of causal reasoning and transfer learning enhances multi-armed bandit strategies, showcasing the potential of causal insights to improve knowledge transfer across tasks.

[31] delves into the integration of Bayesian methods with reinforcement learning techniques. The study addresses the challenge of incorporating uncertainty and probabilistic reasoning into the learning and decision-making process within dynamic environments. By applying Bayesian techniques, the approach provides more robust and adaptive solutions in uncertain and evolving scenarios, showcasing the value of incorporating Bayesian perspectives in reinforcement learning research.

Finally, [32] tackles the challenge of balancing exploration and exploitation in reinforcement learning by incorporating causal models into the learning process. This work proposes a novel approach to address this challenge by incorporating causal models into the agent's learning process. The introduction of causal models aims to refine the learning procedure by constraining the agent's exploration space through the utilization of interventional queries. The study centers around the classic taxi problem from OpenAI

and explores the integration of causal models into the action selection step of Q-learning. By leveraging causal models, the approach offers a mechanism to narrow down the set of possible actions an agent can take. This restriction proves beneficial in enhancing the efficiency of the learning process, yielding both higher and faster rewards and convergence rates. Through empirical evaluation on the taxi problem, the study demonstrates the effectiveness of causal models in improving the performance of Q-learning. By leveraging the insights provided by causal relations, the proposed approach showcases its potential to address the exploration-exploitation trade-off challenge, leading to more efficient and effective learning outcomes in reinforcement learning scenarios. Conventional strategies for exploration and exploitation lack directionality and do not explicitly facilitate interesting state transitions. The adoption of predictive models presents a promising avenue to address this challenge. These models have the potential to incorporate causal knowledge, which involves causal relationships between variables. An essential premise of their approach is that a causal model, even if incomplete, can be harnessed to augment a reinforcement learning (RL) algorithm. Specifically, the authors emphasize that the causal model can capture partial relationships between variables, possibly corresponding to specific subtasks within the broader task they aimed to improve. The core hypothesis of their work is that causal inference can significantly enhance RL by leveraging causal connections between state variables or between actions and state variables. This, in turn, leads to a substantial reduction in the dimensionality of the state or action space. Their method involves a modification to the exploration phase of Q-learning: upon observing a state, the agent employs queries to one or multiple causal models to intelligently select an action that is likely to facilitate goal attainment. The parameters of these causal models are defined through a probabilistic semantic structure.

3.2 Extracted Concepts and Thesis Goals

The review of the existing literature has clarified the scope and direction that this thesis should encompass. Below, several pivotal concepts that will play a significant role in the development of the thesis are reported:

- *Causal Reasoning in Reinforcement Learning*: investigate the integration of causal reasoning into RL algorithms and explore how causal relationships can enhance the decision-making process of RL agents. Highlight the advantages of leveraging causal inference in improving adaptability, generalization and decision-making efficiency within dynamic environments.

- *Sense of Agency and Causal Understanding*: examine the relationship between an agent's sense of agency and its ability to comprehend causal relationships. Discuss how an agent's awareness of its own actions and their consequences contributes to the acquisition of causal knowledge and reasoning capabilities.

- *Causal Inference and Explainability*: explore how causal inference can enhance the explainability of RL models. Investigate how causal contributes to the acquisition of causal knowledge and reasoning capabilities.

- *Counterfactual Analysis and Learning from Demonstrations*: discuss the significance of counterfactual analysis of RL models. Highlight how causal reasoning enables

agents to explore alternative scenarios and learn more effectively from observed behaviors.

- *Counterfactual Analysis and Learning Interventions* : explore the transition from association-based learning to interventions and counterfactual analysis. Discuss how agents can progress from understanding correlations to planning and reasoning about alternative scenarios using causal relationships.

- *Bayesian Networks and Causal Models*: examine the utilization of Bayesian networks and causal models in enhancing RL outcomes. Discuss how these models can capture complex dependencies and causal relationships, leading to improved decision-making and adaptability in uncertain environments.

By incorporating these key concepts, this thesis can provide a comprehensive and insightful exploration of the integration of causal reasoning into reinforcement learning algorithms and its implications across various domains.

Chapter 4

Minigame

This chapter explains the configuration of the experimental environment employed in this thesis. To facilitate a rigorous assessment, a custom implementation was chosen to ensure a challenging setup. The whole work was carried out using the `Python` programming language.

4.1 Game Implementation

The environment simulates an agent navigating an unfamiliar environment in search of an objective whose location he does not know, in addition there are enemies present.

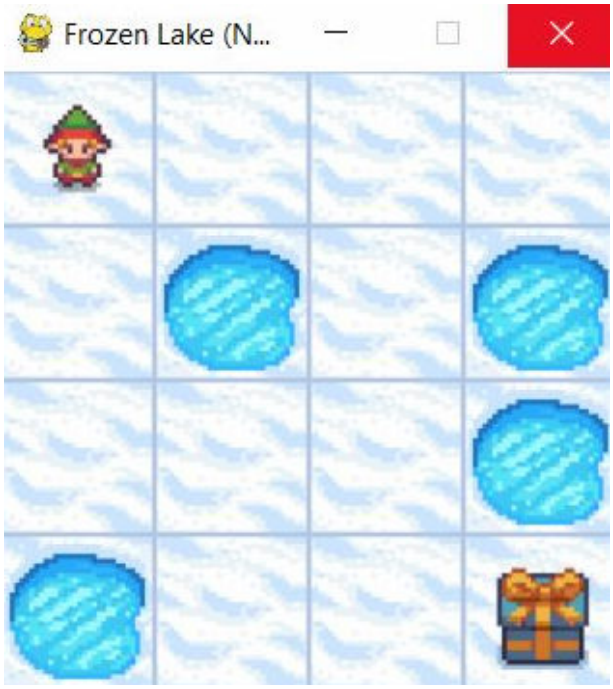


Figure 4.1: *Original graphical user interface of the Frozen lake environment.*

Initially, the focus was on a classic reinforcement learning problem, specifically the Frozen Lake environment from OpenAI [33]. This environment comprises a 4x4 grid containing four types of areas: Start (S), Frozen (F), Hole (H), and Goal (G), as Figure 4.1

shows.

The agent navigates through the grid with the primary objective of reaching the goal in the most efficient manner. Upon successfully reaching the goal, the agent is rewarded with +1, whereas landing on frozen ground or a hole results in a reward of 0. Importantly, there is no punitive negative reward attributed to falling into a hole. As the environment remains unchanged, the locations of both holes and the goal are stationary.

For the scenarios considered in this thesis, to intensify the challenge, dynamic enemy elements have been introduced into the environment. The tasks can be broadly categorized into two groups: the *"easier" tasks* involve scenarios with 1 or 5 enemies in grid-like environments of sizes 10, 20, and 50; whereas, the *more complex tasks* encompass scenarios with 10 and 20 enemies in grid-like environments of sizes 10, 20, 50, and 100. Additionally, there are scenarios with 50 enemies in grid-like environments of sizes 50 and 100.

This heightened complexity is further amplified by incorporating a negative reward of -1 for the agent in case of defeat. This addition elevates the challenge beyond that of the conventional Frozen Lake configuration. Notably, in the larger environments (size 100 mostly, but also size 50), the problem becomes particularly interesting as it falls under the category of a *sparse reward problem*, where the environment rarely provides a useful reward signal. This complexity also reflects in the computation time required to complete the entire task. It is worth mentioning that scenarios where the number of enemies is equal to or greater than the number of rows/columns are particularly challenging.

Similar to the situation in the Frozen Lake-like environment, the current context also involves the agent either achieving victory or encountering defeat. Subsequently, the environment is reset, leading to the repositioning of both the agent and enemies back to their initial starting positions within the environment. Furthermore, similar to the case in the Frozen Lake environment, the agent (along with the enemies) is afforded *five distinct actions*: Stop (0), Right (1), Left (2), Up (3), and Down (4).

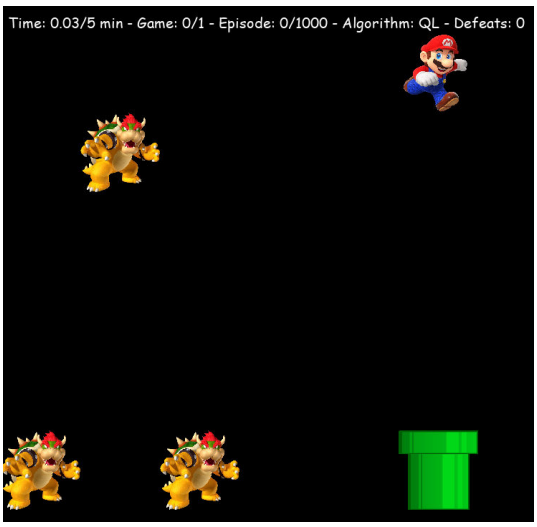


Figure 4.2: *Custom graphical user interface environment: Grid of size 10x10 featuring a single enemy.*

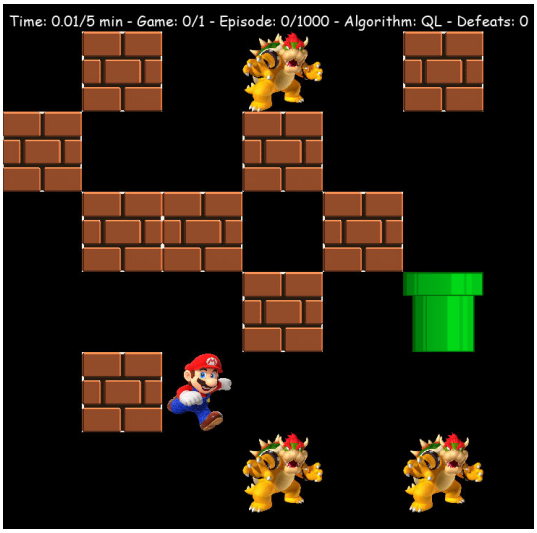


Figure 4.3: *Custom graphical user interface environment: Maze of size 10x10 featuring a single enemy.*

In the newly devised custom game, the agent can lose the game through two distinct

scenarios: firstly, if it collides with an enemy, and secondly, if an enemy occupies the same position as the agent and the agent chooses not to move.

Progressing further in complexity, a maze-like environment was implemented, incorporating randomly generated walls while ensuring that a path to victory exists for the agent. The tasks analyzed in this environment align with those previously described for the grid-like environment.

It's important to observe that in both grid-like and maze-like environments, the initial positions of the agent, goal, enemies, and, in the case of mazes, the walls, are randomly assigned to ensure that none of them start in the same position as another. These experiments progressively introduce higher complexities to the agent's understanding, requiring it to comprehend the environment's dynamics and the consequences of its actions.

A graphical interface was developed to visualize the game environment, allowing for a tangible representation of the agent's interaction. Figure 4.2 and Figure 4.3 display screenshots of the graphical user interface (GUI) designed for grid and maze environments under study. These GUI provide a visual representation of the game progress including various real-time metrics, including the current game status, elapsed time, the number of games played, episodes in progress, the active algorithm, and the count of defeats. As can be imagined, Mario Bros is the agent, Bowser(s) the enemy(ies) and the green pipe the goal, recalling the famous game.

4.2 Causality

The agent's task involves comprehending the underlying dynamics of its surrounding environment. Importantly, the agent approaches this challenge with no prior knowledge of the environment's characteristics.

To achieve an understanding, the agent must unravel the causal model governing the environment, specifically, the causal consequences resulting from its actions. The implementation of the causal model is facilitated through the utilization of the `CausalNex` library. This is a Python library that uses Bayesian Networks to combine machine learning and domain expertise for causal reasoning. You can use CausalNex to uncover structural relationships in your data, learn complex distributions, and observe the effect of potential interventions.

The causality step of investigation centered around extracting causal relationships within a specific context. The chosen environment was a 3x3 grid, featuring a single enemy and no defined goal. The selection of this grid size holds significance: the 3x3 configuration enables numerous interactions between the agent and the enemy. The absence of a goal in this environment aligns with the primary objectives, which focus on comprehending how the agents' actions influence their movement and gaining insights into the factors leading to a "game over" scenario.

The input dataframe is specifically designed to consist of 10k rows, which track the movements of both the agent and the enemy; the actions taken by the agents and by the enemies are completly random (in this section).

To facilitate a better comprehension within the causal model, several key variables are defined; these variables are designed to provide a comprehensive representation of the agent-enemy interaction dynamics in the environment:

- *State*, this variable expresses the differences in X and Y coordinates for both the agent and the enemy.
- *Action*, to accommodate the multiple actions available to both the agent and the enemy, a set of variables is defined, each corresponding to a specific action (standing still is also considered an action).
- *Var*, an important aspect is the definition of parent variables within the model. This step is crucial to ensure an accurate representation of the system. Notably, the arrow relationships are directed from parents to children, aligning with the principle that actions cannot be influenced by other variables. This variable is used to help the model in understanding the link from other variables.
- *Game Over*, this variable is set to True when the agent loses the game.
- *Alive*, conversely, this variable represents the logical opposite, indicating whether the agent is still in the game.
- *Enemies Nearby*, this variable is defined for every possible action that both the agent and the enemies can take. If an enemy reaches a position such that one action can bring it into contact with the agent, the corresponding variable, from the agent's perspective, is set to True.
- *Enemies Attached*, when an enemy makes contact with the agent, this variable becomes True. It's important to note that the agent only loses the game in this scenario if it remains stationary. This variable serves as a specialized representation of the "Enemies Nearby" variable, specifically when the chosen action is to remain stationary: action 0.
- *Contact*, this variable is set to True when the agent chooses an action that results in the Enemies Nearby variable becoming True. Such a situation leads to the agent's loss as it moves toward an enemy.

To obtain the best performance from the Bayesian Networks implemented in the subsequent step, the dataframe is processed by converting its variables into binary format. This transformation helps in defining several variables in such a way that the Bayesian Network can precisely discern which variables influence changes in other variables.

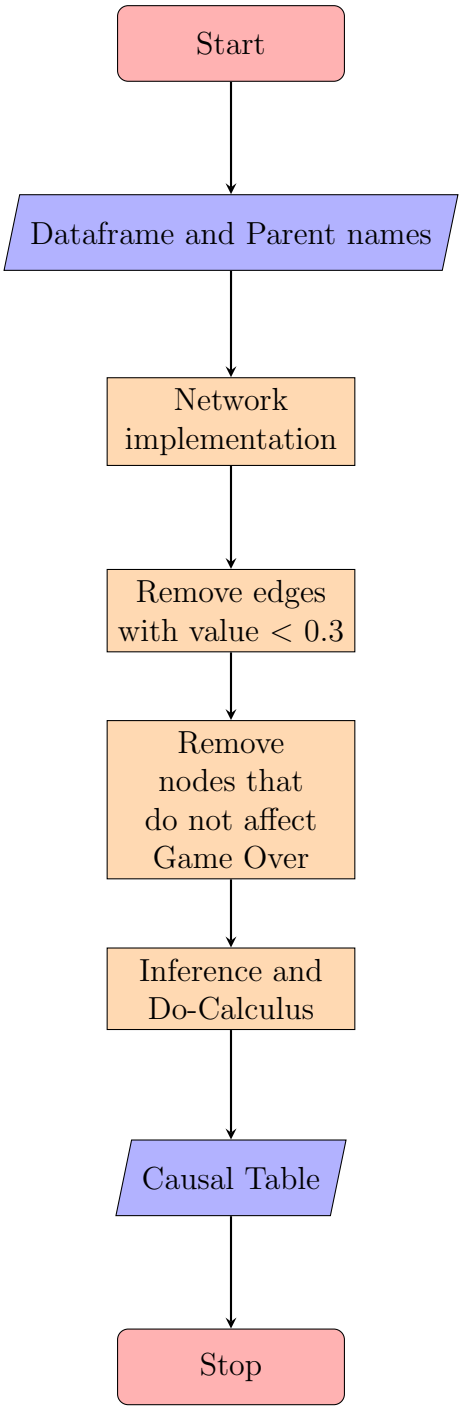
Algorithm 2 provides a summary block diagram that illustrates the implementation process of this part of the thesis. Here, the explanation of them is reported.

The first step involves acquiring the binary dataframe and identifying the parent variables, which include "GameOver", "Alive", "State" and "Var".

Following that, we construct the entire network associated with the dataframe as a "StructureModel" object. This network's structure is depicted visually in Figure 4.4.

After creating a weighted graph, the weights associated with connections between nodes are examined: any edges with an absolute value below 0.3 are removed as they are considered too insignificant.

From the remaining graph, the variables directly connected to "GameOver" are extracted, which include the 'Contact' variables, representing general scenarios leading to a game over (as illustrated in Figure 4.4). Subsequently, variables that lack a link to at least



Algorithm 2: *Block diagram depicting the implementation of a causal model and the causal table extraction.*

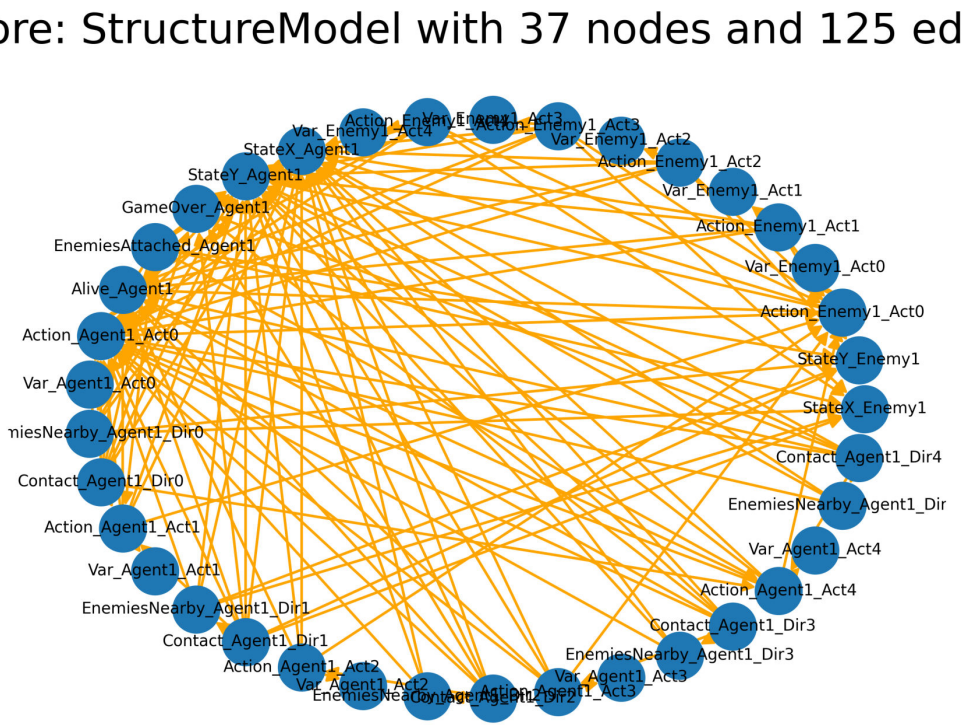


Figure 4.4: *Structure model derived directly from the initial dataframe, without any cleaning procedures applied.*

one of the previously extracted features are removed. In this step, all variables related to "Var" are deleted since they are redundant to action variables, and also variables linked to enemy actions are discarded, as they do not influence the agent's actions or the occurrence of a game over. This adjustment is consistent with the environment's definition, where both agent and enemy actions were randomized. While the initial structure could yield correct results, it would lead to an unacceptable increase in computation time. To put this into perspective, the final structure model comprises 21 nodes and 77 edges, as depicted in Figure 4.5. This cleaning process significantly enhances computational efficiency for the next step.

Afterwards, the Bayesian Network is implemented by using the remained graph and, subsequently, the inference process, followed by do-calculus, is initiated. As previously highlighted in the Background section, the significance of do-calculus is paramount: while Bayesian networks do not inherently impose such constraints, they become essential when the do-calculus operation is considered; this transformation represents a pivotal step, shifting the Bayesian network from a non-causal model to one that encapsulates causal relationships and insights. The do-calculus operation is applied to all remaining variables with themselves, resulting in a causal table that stores the most likely outcomes from the do-calculus operation. The causal table consists of rows representing the occurrence of the reported variable and columns containing the results of other variables under that specific

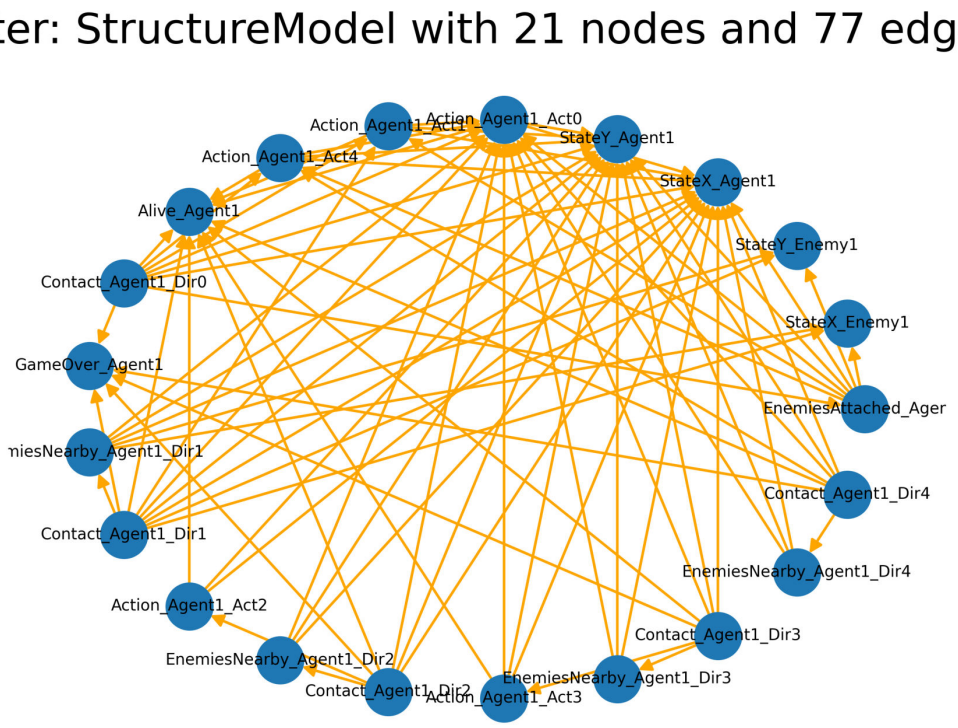


Figure 4.5: *Structure model post-cleaning procedure, prepared for subsequent training and counterfactual steps.*

condition. This table allows for the extraction of information regarding how the agent's actions influence its movements and the occurrence of game over. With this configuration, the complete do-calculus process required only 11 minutes (Computer specifications: equipped with an i7 Processor and 16 GB of RAM).

The final step in this process involves eliminating rows and columns that remain unchanged in the tables. Specifically, the removed columns include 'StateX_Enemy1' and 'StateX_Enemy1', consistent with the previous considerations. Additionally, the eliminated rows are 'GameOver_Agent1' and 'Alive_Agent1', since these two variables serve as the parents of the network. An extract of the causal table is stored as a *pickle* file and it is reported in Table 4.1.

	ΔX	**ΔY**	**GameOver**	**Act3**	**EnNearby_Dir3**	**Contact_Dir3**
Act1	+1	0	F	F	F	F
Contact_Dir3	0	+1	T	T	T	T

Table 4.1: *Extract from the generated causal table: the first row represents a portion of the movement model, while the second row represents a portion of the game-over model. T represents 'True', indicating the occurrence of a condition or action, while F signifies 'False' for non-occurrence.*

4.3 Reinforcement Learning

The conventional Q-Learning algorithm faces limitations in dynamic environments, primarily due to its approach in updating the Q-Table. For instance, considering Equation 2.3, in scenarios where the agent loses, the Q-Table is updated at the point of failure (state S). However, in cases involving dynamic enemies, this update alone does not effectively assist the agent in avoiding the enemy's actions in subsequent steps.

To significantly alleviate this challenge, the agent needs to discern the actions leading to its failure. This understanding can be achieved through the utilization of causal tables derived from the Causality segment. By leveraging the a-posteriori probabilities, the agent can determine the consequences of its chosen actions, the progression of failure states, and strategies for avoiding unfavorable outcomes.

To determine the agent's policy for selecting actions, the ϵ-greedy method was employed: this strategy strikes a balance between exploration and exploitation, and it was tailored to follow a descending exponential pattern. Specifically, as the number of completed (and successful) game episodes increases, the likelihood of engaging in exploration decreases gradually.

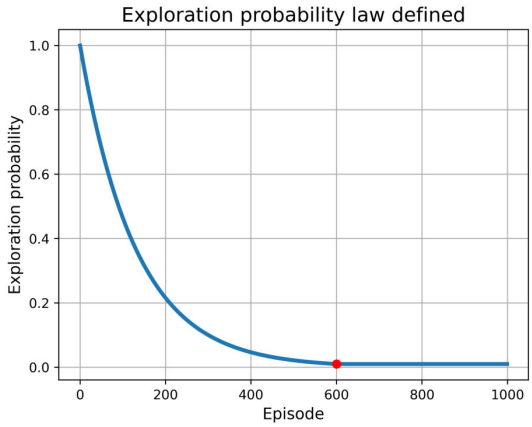


Figure 4.6: *Exploration probability law: when the red point on the graph is reached, the exploration probability is adjusted to the predefined minimum value.*

A minimum exploration probability of 0.01 was established, indicating that in 99 out of 100 instances, the agent would choose to exploit its current knowledge, and in 1 out of 100 instances, it would explore new actions. The reduction in this exploration probability is regulated by a parametric coefficient. This coefficient relies on the total number of episodes required for the game and the specified episode number at which the minimum exploration probability is aimed. In this context, the coefficient was set to 0.6, signifying that, for example, if 1000 episodes were planned, the minimum exploration probability would be encountered starting from the 600th episode. This approach ensures a controlled transition from exploration to exploitation as the agent gains more experience in the game; therefore, the exploration probability factor follows a decreasing exponential law, depending on the number of complete episodes and the decrease constant; the graphical representation of this law defined is depicted in Figure 4.6.

In order to identify the most effective approach for the implementation of Causal

Algorithm 3 Built Algorithms

Require: Initialize empty Q_table(s_x, s_y, a), for all s_x, $s_y \in \mathbb{S}^+$, a $\in \mathbb{A}(s)$
Require: Initialize empty Q_table_track(s_x,s_y,a) for tracking, for all s_x,$s_y \in \mathbb{S}^+$, a $\in \mathbb{A}(s)$
Require: Causal table → causal_table
  Learning rate → $\alpha = 0.01$
  Discount factor → $\gamma = 0.99$
  Percentage of total exploration on the match → $perc\_expl = 0.6$
  Exploration probability → expl_prob = 1
  Minimum exploration probability → min_expl_prob = 0.01
  Number of episode → n_ep = 1000
  Exploration decreasing decay → expl_decay = $-\frac{ln(min\_expl\_prob)}{n\_episodes*perc\_expl} = 0.00767$

  for episode ∈ n_ep **do**
    Initialize state S
    while each step for episode until S is terminal **do**
      Choose action A from S using ϵ-greedy policy:
      if rand(0, 1) < exploration probability **then**
        if approach == CQL1 or approach == CQL1* **then**
          A: CM1, with more exploration
        else if approach == CQL2 or approach == CQL2* **then**
          A: CM1, with more exploration, additional Q-Table update
        else if approach == CQL3 or approach == CQL3* **then**
          A: CM1 and CM2, with more exploration, additional Q-Table update
        else if approach == CQL4 or approach == CQL4* **then**
          A: CM1 and CM2, without more exploration, additional Q-Table update
        else if approach == QL **then**
          A: random action
        end if
      else
        if approach == QL or approach == CQL_ **then**
          A: np.argmax(Q_table[current_stateX, current_stateY, :])
        else
          possible_actions = CM2
          A: np.argmax(Q_table[current_stateX, current_stateY, possible_actions])
        end if
      end if
      Take action A, observe R, S'
      Classic Q-Table update: Equation 2.3
      S ⟵ S'
      $expl\_prob = max(min\_expl\_prob,\ e^{-expl\_decay*episode\_number})$
    end while
  end for

Q-Learning, a comprehensive exploration of four (eight) distinct variants has been conducted. The description of these variants are detailed in Algorithm 3, while the explanation of each component is reported in Algorithms 4, 5, 6, 7 and 8. This implementation heralds the introduction of a novel class of algorithms, signifying the development of a Q-Learning framework capable of comprehending the agent's surroundings and well-suited for dynamic environments. In addition, two distinct types of novel causal approaches were formulated: one focusing solely on modifying the exploration aspect, and the other involving modifications to both exploration and exploitation parts. These methodologies were designed with the intent of understanding the most effective strategies to enhance performance. Through systematic implementation and rigorous evaluation, these variations were explored to establish the most effective way to achieve superior results. Algorithms where both the exploration and exploitation phases are modified are marked with the star ("*").

Algorithm 4 CM1: Causal Model of Movement

Require: Causal Table → causal_table
Require: Current X coordinate → curr_X
Require: Current Y coordinate → curr_Y
Require: Action → act

ΔX = causal_table(action)
ΔY = causal_table(action)
new_X = curr_X + ΔX
new_Y = curr_Y + ΔY
Return new_X, new_Y

Concerning the movement causal model (CM1), the utilization of the causal table in this scenario is straightforward. When a particular action is selected, the process involves intersecting the row associated with that specific action in the causal table with the columns representing the changes in X and Y coordinates (ΔX and ΔY) resulting from that action. Following this intersection, the updated coordinates are computed.

In contrast, employing the causal table to determine which actions must be excluded in order to prevent a game over (CM2) is a somewhat more intricate process. As previously outlined, two variables are utilized to assess the potential occurrence of a game over (they are provided by the environment at each step taken): the presence of enemies in the same location as the agent and the proximity of enemies at a distance of 1 from the agent. In the first case, a game over only arises if the agent remains stationary, whereas in the second case, the agent loses if it selects actions that lead it to the same location as the enemies.

To tackle this, the initial step involves extracting the rows from the causal table where the "GameOver" variable is true. The first aspect addresses the situation where enemies are in the same location as the agent, indicated by the "Enemies_Attached" variable being True. In this instance, the causal table's row concerning the presence of attached enemies is isolated, and the action responsible for the occurrence of game over is removed from the list of possible actions. Naturally, the action that instructs the agent to remain stationary must be removed in such cases. However, it is important to emphasize that this decision

Algorithm 5 CM2: Causal Model of the Game Over

Require: Causal table → causal_table
Require: Enemies_nearby from the environment
Require: Enemies_attached from the environment
Require: Possible actions → pos_acts

```
rows_GameOver_True = causal_table[causal_table['GameOver' == True]]
if Enemies_attached == True then
    row_GameOver_att = rows_GameOver_True['EnemiesAttached' == True]
    for action ∈ pos_acts do
        if row_GameOver_att[action] == True then
            pos_acts.remove(action)
        end if
    end for
end if
for enemy_nearby ∈ Enemies_nearby do
    if Enemies_nearby[enemy] ≠ 50 then
        row_GameOver_nearby = rows_GameOver_True[enemy_nearby == True]
        for action ∈ pos_acts do
            if row_GameOver_nearby[action] == True then
                pos_acts.remove(action)
            end if
        end for
    end if
end for
Return pos_acts
```

Algorithm 6 Additional Q-Table Update

Require: curr_Xcoord, curr_Ycoord
Require: new_Xcoord, new_Ycoord
Require: Action done A, learning rate α, discount factor γ, Q_table Q

reward = 0
$Q[curr\_Xcoord, curr\_Ycoord, A] = (1 - \alpha) * Q[curr\_Xcoord, curr\_Ycoord, A] + \alpha(reward + \gamma \max Q[new\_Xcoord, new\_Ycoord, :]$

Algorithm 7 Without More Exploration

Require: Causal movement model → CM1
Require: Actual coordinates → curr_Xcoord, curr_Ycoord
Require: Possible actions

A: random action(possible actions)
new_Xcoord, new_Ycoord = CM1(causal_table, curr_Xcoord, curr_Ycoord, A)
Return A, new_Xcoord, new_Ycoord

Algorithm 8 With More Exploration

Require: Exploration tries thresholds: th = 10
Require: Causal movement model → CM1
Require: Actual coordinates → curr_Xcoord, curr_Ycoord
Require: Possible actions

```
new_state = False
check_if_new = 0
while not new_state do
    A: random action(possible actions)
    new_Xcoord, new_Ycoord = CM1(causal_table, curr_Xcoord, curr_Ycoord, A)
    if Q_table_track[new_Xcoord, new_Ycoord] == 0 then
        Q_table_tracking[new_Xcoord, new_Ycoord] = 1
        new_state = True
    else
        check_if_new += 1
        if check_if_new == th then
            new = True
        end if
    end if
end while
Return A, new_Xcoord, new_Ycoord
```

is guided by only causal knowledge rather than being driven by pre-defined conditions.

On the other hand, when enemies are in close proximity to the agent, a similar approach is applied. Rows indicating a game over scenario are intersected with columns related to variables "EnemiesNearby_Dir...," enabling the identification of actions that would result in a game over. These actions are then removed from the possible actions. Notably, the variable "Enemies_nearby" is structured as a list, with a length equal to the number of enemies. Each element in the list represents the direction of an enemy as perceived by the agent; If the enemies are positioned at a distance greater than 1 from the agent, their corresponding elements in the list are marked with the value "50".

Conversely, the additional Q-table update step assumes a crucial role in accelerating the learning process. The experimental results will show that the algorithms learn at twice the speed when compared to the traditional algorithm. Furthermore, the computational time necessary for this update is negligible. Consequently, the CQL1 and CQL1* algorithms are expected to behave very similar to the classical Q-Learning.

The "more exploration" component serves the primary purpose of assisting the agent in navigating through unfamiliar states; a lot of works (for example [34]) regard the field of improving exploration in RL algorithms, with the designed implementation of this thesis a easy but strong way has been choose. However, this strategy might not always prove beneficial; to understand this point, CQL4 was introduced, where actions that do not result in a game over are extracted, but the selection of one of these actions is entirely random, disregarding any specific exploration objectives.

Furthermore, in the case of CQL3 and CQL4 algorithms, the agent loses only under

specific conditions. In algorithms where there are no adjustments made to the exploitation phase, the agent's action selection involves choosing a random action, which may result in defeat; whereas, within the modified exploration or modified exploitation phases, the agent is expected to lose only in two particular scenarios:

1. The first scenario occurs when the agent is positioned at the edge of the environment, has an enemy attached (as indicated by Enemies_Nearby_Dir0 = True), and selects an action that moves it towards the edge. In such cases, the agent remains in the same position and subsequently loses the game.

2. The second scenario occurs when an agent is surrounded by enemies, a situation that arises in particularly complex environments and tasks.

Chapter 5

Experiments and Results

In this section, the key findings and highlights of the results are presented; as described previously, several experiments have been considered, ranging from easier to more demanding scenarios.

5.1 Evaluation Setup

First of all, the configuration employed for assessing the performance of the case study is clarified. Following a methodology similar to the one detailed in [4], a comprehensive suite of experiments has been meticulously conducted, covering a spectrum of configurations. Specifically, a game is generated randomly, taking into account variables such as the environmental type, its dimensions, and the number of enemies in play. Subsequently, this generated game undergoes evaluation using each of the aforementioned approaches. This iterative process is replicated *ten* times, with each iteration involving the generation of a distinct randomly constructed game.

Furthermore, an additional evaluation aspect involves the agent's behavior in the presence of two different enemy action scenarios. In the first scenario, simpler in nature, a predetermined sequence of random actions is generated for the enemies, which follow this sequence in the same order, and upon reset, they restart to the initial action defined (note that the actions generated randomly are distinct for each enemy). The second scenario is more challenging, as enemies take random actions at each time step.

Given the variety of environments and tasks investigated, two distinct *timeout* criteria were introduced to evaluate the effectiveness of the proposed solutions in terms of average reward achievement and computational efficiency. These timeout criteria were applied individually to each game and approach. In particular, a 5-minute timeout was implemented for the "easier" tasks, while a 10-minute timeout was enforced for the more complex tasks. These values have been chosen after evaluating the computational time required for each problem under investigation; in scenarios where the environment is relatively limited, these conditions are less demanding, but as the environment widens, they become enough stringent. In the case that an algorithm exceeds the specified time limit during a game, all the games associated with that algorithm are considered invalid, and the label *"-"* is included in the average reward results, which are reported on the following pages.

5.1.1 Metrics

The chosen evaluation metrics encompass fundamental RL measures: the *number of steps required to complete an episode* and the *average reward across episode steps*. The latter metric holds maximum importance for RL problems, as defined in Chapter 2. Notably, the averages of these metrics are derived through the previously mentioned methodology. Subsequent sections will present graphical plots to visually illustrate these aspects. The primary emphasis of this thesis, however, revolves around the frequency of the agent's losses, which is easily distinguishable due to the equivalent rewards for winning and losing conditions. Nevertheless, the rate at which convergence of the number of passages per episode is achieved is also of significant importance.

To summarize the results, tables, plots and bar plots are used. The presented plots showing the average reward and the number of steps required per episode have been generated using a first-order Gaussian filter. This approach has been employed to create smoother and more visually coherent plots, enhancing the clarity of the presented data.

In essence, this entire methodology can be seen as a Monte Carlo simulation, as it involves calculating the average of metrics across 10 simulations to comprehensively evaluate the outcomes.

Additionally, to evaluate the *execution time for individual simulations* under each approach in terms of CPU processing time, graphical representations depicting the average duration in minutes are provided for straightforward comparison. The computer used for these simulations is equipped with an i7 processor and 16 GB of RAM.

Furthermore, to assess the occurrence of *timeouts* with each algorithm during simulations, visual representations are presented to illustrate their frequency. As an alternative perspective on the effectiveness of the proposed algorithms, bar plots display the *average number of defeats in a single game*.

Finally, an important tool for comprehending the mechanisms inside of the designed algorithms is the *Q-Table*. This resource aids in understanding how the reward signal is distributed across the environment, highlighting the consequences of the agent's actions. It is worth noting that in larger environments, the resulting Q-table may become challenging to interpret effectively.

The following pages present and analyze the results. Additionally, for each environmental context and type of enemy behavior, a case of study is selected and examined in depth, aiding understanding through various figures.

5.2 Low Complexity Environments

In this section, the results obtained in less complex environments are presented; relevant observations and reflections on the overall results and the specific case studies chosen are provided.

5.2.1 Grid

Enemies performing same actions in every match

Environment		Average Reward in Algorithms								
Grid size	**Enemies**	**QL**	**CQL1**	**CQL2**	**CQL3**	**CQL4**	**CQL1***	**CQL2***	**CQL3***	**CQL4***
10x10	1	0.951	0.947	0.949	**1**	**1**	0.947	0.951	**1**	**1**
20x20	1	0.966	0.958	0.971	0.9981	0.9984	0.961	0.968	0.9998	**1**
50x50	1	0.916	0.899	0.964	**1**	1.00	0.854	0.957	**1**	**1**
10x10	5	0.797	0.556	0,812	0.979	0.979	0.849	0.813	0.987	**0.988**
20x20	5	0.888	0.827	0.89	0.967	0.964	0.901	0.91	**0.9903**	0.9901
50x50	5	0.733	-0.136	0.888	0.918	0.915	0.763	0.92	**0.96**	0.952

Table 5.1: *Comparison of average reward performance: Classic Q-Learning (QL) vs. Causal Q-Learning Variants with exploration modification (CQL1 to CQL4) vs. Causal Q-Learning Variants with modifications to both exploration and exploitation phases (CQL1* to CQL4*). Evaluation in less challenging grid environments where enemies follow a fixed sequence of actions.*

Table 5.1 shows that all tasks have been successfully completed by each algorithm. In the simplest tasks with only one enemy, Q-Learning (QL) demonstrates strong performance. However, as the complexity of the task increases, its results notably decline, especially when compared to the results achieved with various Causal Q-Learning (CQL) variants, both without modified exploitation and with modified exploitation (the latter yielding the best outcomes).

As case of study 20x20 grid like-environment with 1 enemy is considered; this task is of particular interest because the environment offers few agent-enemy interactions without suffering from the sparse-reward problem often encountered in larger environments. The results are presented in Figure 5.3, which illustrates the average rewards. Additionally, Figure 5.1 showcases the average number of defeats experienced by each algorithm, while Figure 5.2 displays the average computation times required to complete a single game for each algorithm.

The latest implemented algorithm (CQL4*) demonstrates flawless performance as it never loses. On the other hand, the second-to-last algorithm (CQL3*) only loses once, yet its performance remains exceptional, especially when compared to the results achieved by the Q-Learning algorithm. In terms of the computation time required to finish the game, Q-Learning proves to be the fastest in this task, although it is worth noting that all algorithms generally exhibit fast performance.

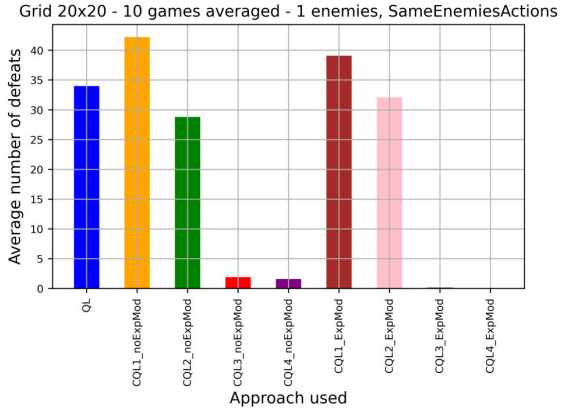


Figure 5.1: *Plot of how many times on average does the single algorithm lose in each game; 20x20 grid-like environment, 1 enemy that follows the same default random action sequence. "_noExpMod" indicates no change in the exploitation phase, while "_ExpMod" indicates the presence of such a change, equivalent to the "*" symbol.*

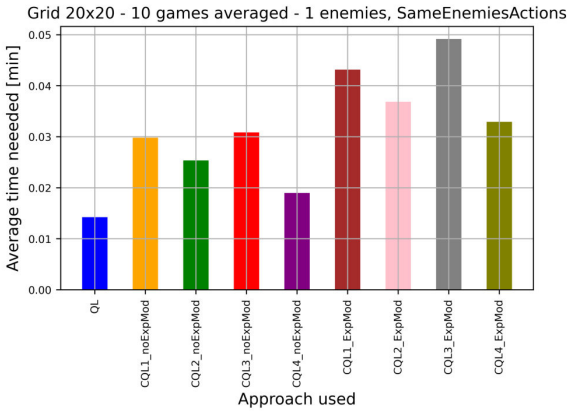


Figure 5.2: *Plot of the average game completion time for the single algorithm; 20x20 grid-like environment, 1 enemy that follows the same default random action sequence. "_noExpMod" indicates no change in the exploitation phase, while "_ExpMod" indicates the presence of such a change, equivalent to the "*" symbol.*

In general, these types of tasks can also be performed using the classic Q-Learning algorithm. However, notice that when there are multiple enemies present, the average reward can significantly diminish. Furthermore, it is worth highlighting that the tasks explored are relatively simple.

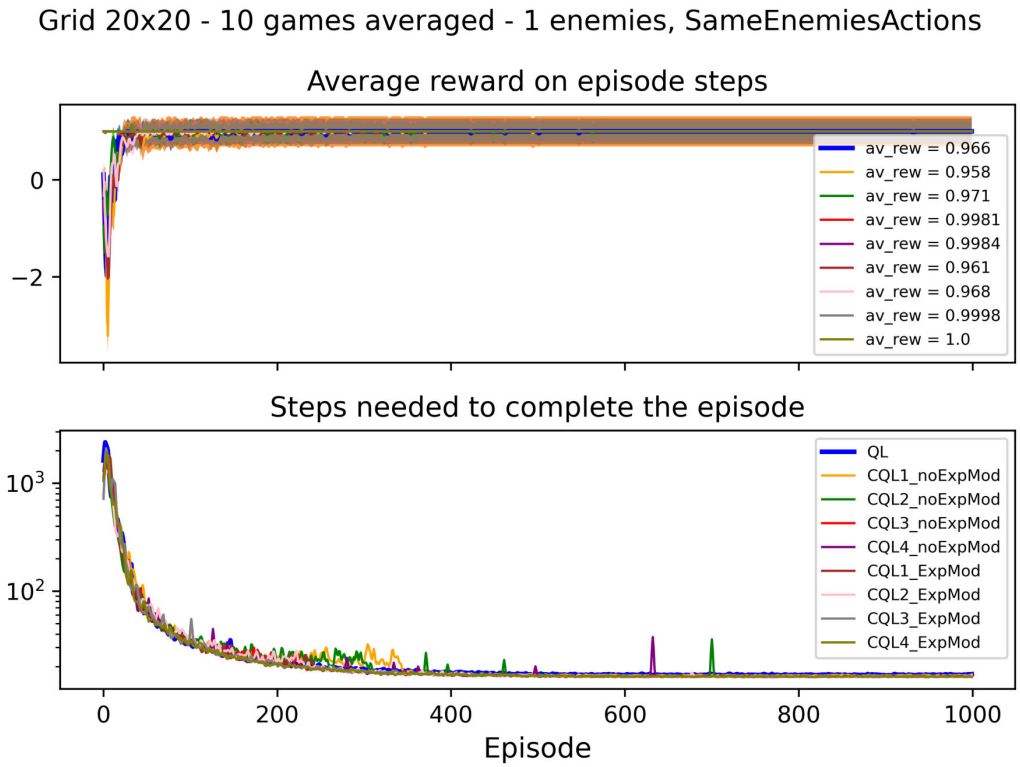


Figure 5.3: *Plot of performance comparison of the average reward and the steps needed for episode; 20x20 grid-like environment, 1 enemy that follows the same default random action sequence. "_noExpMod" indicates no change in the exploitation phase, while "_ExpMod" indicates the presence of such a change, equivalent to the "*" symbol.*

Enemies performing random action in every match

Environment		Average Reward in Algorithms								
Grid size	**Enemies**	**QL**	**CQL1**	**CQL2**	**CQL3**	**CQL4**	**CQL1***	**CQL2***	**CQL3***	**CQL4***
10x10	1	0.899	0.87	0.892	0.938	0.935	0.956	0.956	0.9952	**0.9983**
20x20	1	0.892	0.97	0.944	0.979	0.9958	0.979	0.974	0.9977	**0.998**
50x50	1	0.933	0.938	0.944	0.943	0.948	0.934	0.975	**0.986**	0.98
10x10	5	-0.65	-0.399	-0.589	0.442	0.37	-	0.567	0.982	**0.983**
20x20	5	-0.029	0.134	0.047	0.596	0.609	0.818	0.844	0.9928	**0.993**
50x50	5	-	-	0.834	-	-	0.721	0.898	0.933	**0.93**

Table 5.2: *Comparison of average reward performance: Classic Q-Learning (QL) vs. Causal Q-Learning Variants with exploration modification (CQL1 to CQL4) vs. Causal Q-Learning Variants with modifications to both exploration and exploitation phases (CQL1* to CQL4*). Evaluation in less challenging grid environments where enemies take random actions. The symbol "-" indicates that a timeout occurred.*

Table 5.2 presents the results for these specific tasks. It is evident that the overall performance is lower compared to the previously discussed scenarios. The diminished performance can be linked to the increased task complexity. Specifically, it can be attributed to the initial lack of predictability in enemy movements. In previous scenarios, enemy actions followed a fixed (logical or not) sequence, allowing the agent to discern the con-

sequences of its actions at specific locations and moments. However, in these tasks, such predictability is no longer applicable.

However, as described also before, when the number of enemies becomes greater than 1, the Q-Learning approach and, in this case, also causal Q-Learning algorithms with only the exploration modified, have a significant deterioration in performance, whereas, causal Q-Learning algorithms with modified exploration and exploitation keep excellent performance very close to optimal.

As case of study 10x10 grid like-environment with 5 enemies is considered. The results are presented in Figure 5.6, which illustrates the average rewards. Additionally, Figure 5.4 showcases the average number of defeats experienced by each algorithm, while Figure 5.5 displays the average computation times required to complete a single game for each algorithm.

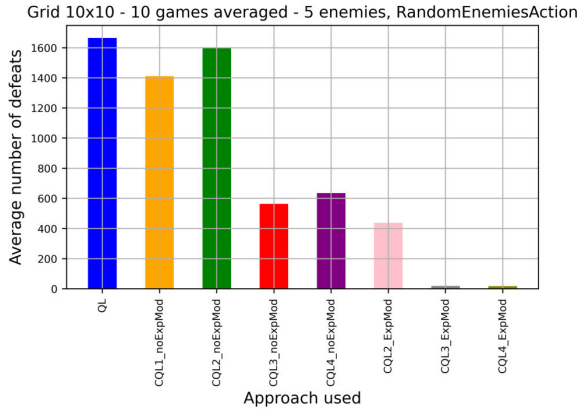


Figure 5.4: *Plot of how many times on average does the single algorithm lose in each game; 10x10 grid-like environment, 5 enemies that take random actions. "_noExpMod" indicates no change in the exploitation phase, while "_ExpMod" indicates the presence of such a change, equivalent to the "*" symbol.*

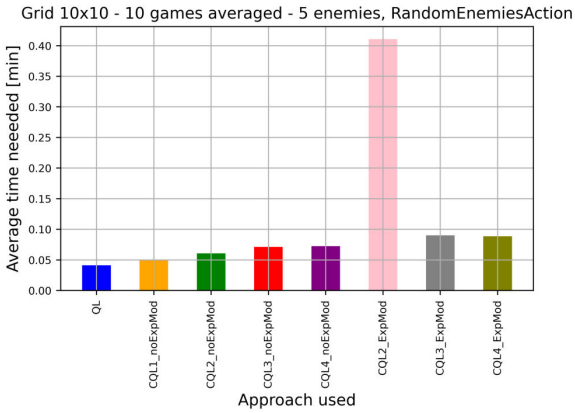


Figure 5.5: *Plot of the average game completion time for the single algorithm; 10x10 grid-like environment, 5 enemies that take random actions. "_noExpMod" indicates no change in the exploitation phase, while "_ExpMod" indicates the presence of such a change, equivalent to the "*" symbol.*

Analyzing the violation of timeout conditions in CQL1* requires considering that both CQL1 and CQL1* closely resemble classic Q-Learning. The primary distinction lies in the choice of actions that lead the agent to unexplored positions. Consequently, if traditional Q-Learning yields poor results, such as the -0.65 score observed here, indicating difficulty in finding quickly a solution due to a high number of defeats, an approach that emphasizes exploration may further extend the time needed to complete the game. This approach may potentially lead to improved average reward performance but necessitates additional time.

The plot provides insights into the behavior of the Q-Learning algorithm. As the environment becomes more dynamic due to its small size and the presence of several enemies, the likelihood of agent-enemy interactions increases. In such scenarios, the traditional Q-Learning algorithm proves to be inadequate. However, despite there being a considerable number of enemies, the causal algorithms, mostly where the exploitation is modified,

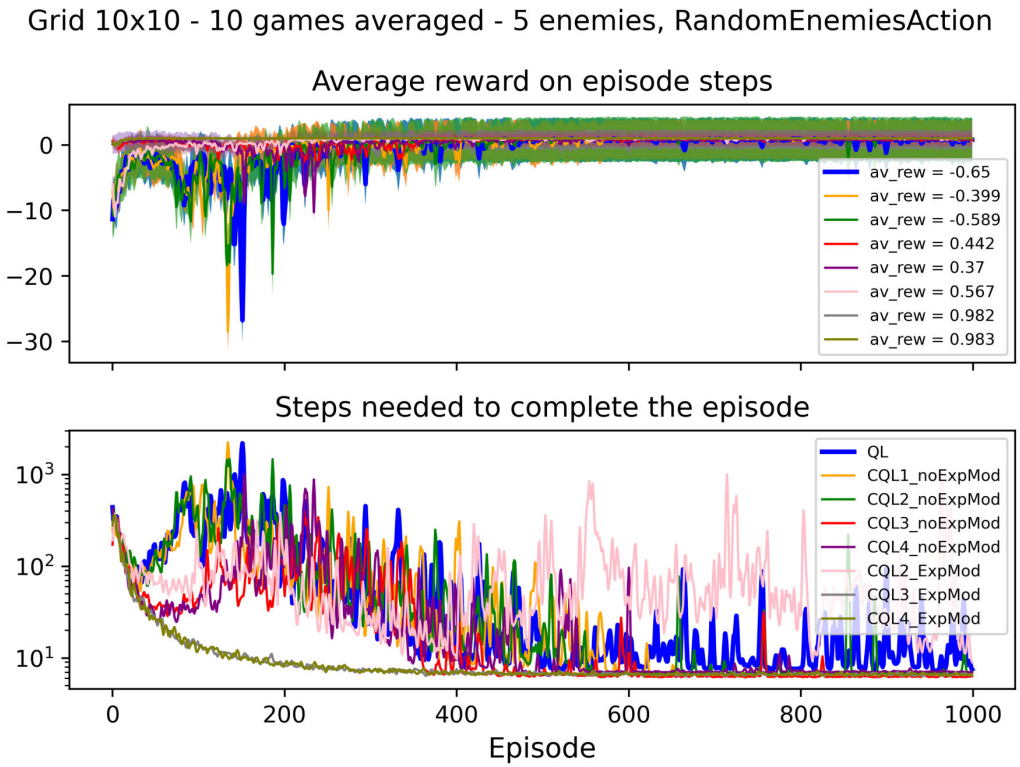


Figure 5.6: *Plot of performance comparison of the average reward and the steps needed for episode; 10x10 grid-like environment, 5 enemies that take random actions. "_noExpMod" indicates no change in the exploitation phase, while "_ExpMod" indicates the presence of such a change, equivalent to the "*" symbol.*

demonstrate strong performance both in terms of minimizing the number of defeats and optimizing computational efficiency.

Notably, exceptions exist in the cases of CQL1 and CQL2 when exploitation is not modified; they yield poor results. In contrast, as highlighted previously, CQL3 and CQL4 with modified exploitation prove to be the most powerful algorithms, reaching near-perfect performance levels; in fact, by considering the trend of the steps needed to complete the episode, these two algorithms have smooth curve. Furthermore, the behavior of these two is remarkably alike, and their performances are, in fact, equivalent.

It is necessary to underline that where the agent loses, the Q-Table is still updated, this leads to a reshuffling of the Q-Table, which therefore becomes more complex to use, it causes the enemies to be mobile and not fixed on the point where the agent lost, also causing further difficulty in reaching the target location. This factor may help explain why CQL1 and CQL2, as well as CQL3 and CQL4, fail to achieve favorable results.

5.2.2 Maze

Enemies performing same actions in every match

Environment		Average Reward in Algorithms								
Maze size	**Enemies**	**QL**	**CQL1**	**CQL2**	**CQL3**	**CQL4**	**CQL1***	**CQL2***	**CQL3***	**CQL4***
10x10	1	0.96	0.952	0.94	**1**	**1**	0.942	0.948	**1**	**1**
20x20	1	0.901	0.977	0.9	0.925	0.924	0.966	0.973	0.9989	**0.9998**
50x50	1	0.922	0.94	0.966	1	1	0.938	0.967	**1**	**1**
10x10	5	-	-	-	-	-	-	-	0.959	**0.966**
20x20	5	0.788	0.77	0.725	0.958	0.952	0.775	0.819	0.975	**0.979**
50x50	5	0.68	0.759	0.869	0.973	0.976	0.798	0.86	**0.978**	0.967

Table 5.3: *Comparison of average reward performance: Classic Q-Learning (QL) vs. Causal Q-Learning Variants with exploration modification (CQL1 to CQL4) vs. Causal Q-Learning Variants with modifications to both exploration and exploitation phases (CQL1* to CQL4*). Evaluation in less challenging maze environments where enemies follow a fixed sequence of actions. The symbol "-" indicates that a timeout occurred.*

Similar to the grid-like environment described previously, where enemies follow a predefined sequence of random actions, the outcomes achieved in these tasks are good for the classic Q-Learning algorithm where only one enemy is in the environment; the results decrease considerably where there are five enemies. It is worth noting that there is an exception when the environment is a 10x10 grid with 5 enemies, in fact, by considering that the agent is enclosed within a maze-like environment, the selected task is notably complex, which is why it has been chosen as a case study below. The other results can be labeled "ordinary" and all comments given in the previous section can be considered.

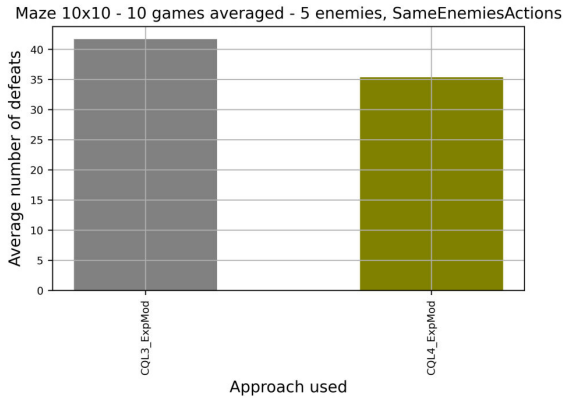


Figure 5.7: *Plot of how many times on average does the single algorithm lose in each game; 10x10 maze-like environment, 5 enemies that follow the same default random action sequence. "_noExpMod" indicates no change in the exploitation phase, while "_ExpMod" indicates the presence of such a change, equivalent to the "*" symbol.*

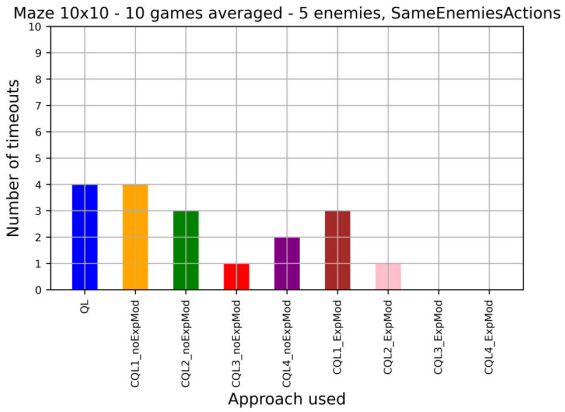


Figure 5.8: *Plot of the number of timeouts for each algorithm; 10x10 maze-like environment, 5 enemies that follow the same default random action sequence. "_noExpMod" indicates no change in the exploitation phase, while "_ExpMod" indicates the presence of such a change, equivalent to the "*" symbol.*

In the selected case of study, 10x10 maze with 5 enemies, only CQL3* and CQL4* managed to stay within the imposed time limit, whereas, the others algorithms exceeded at least one time (CQL3 and CQL2*) or more times, as reported in Figure 5.8.

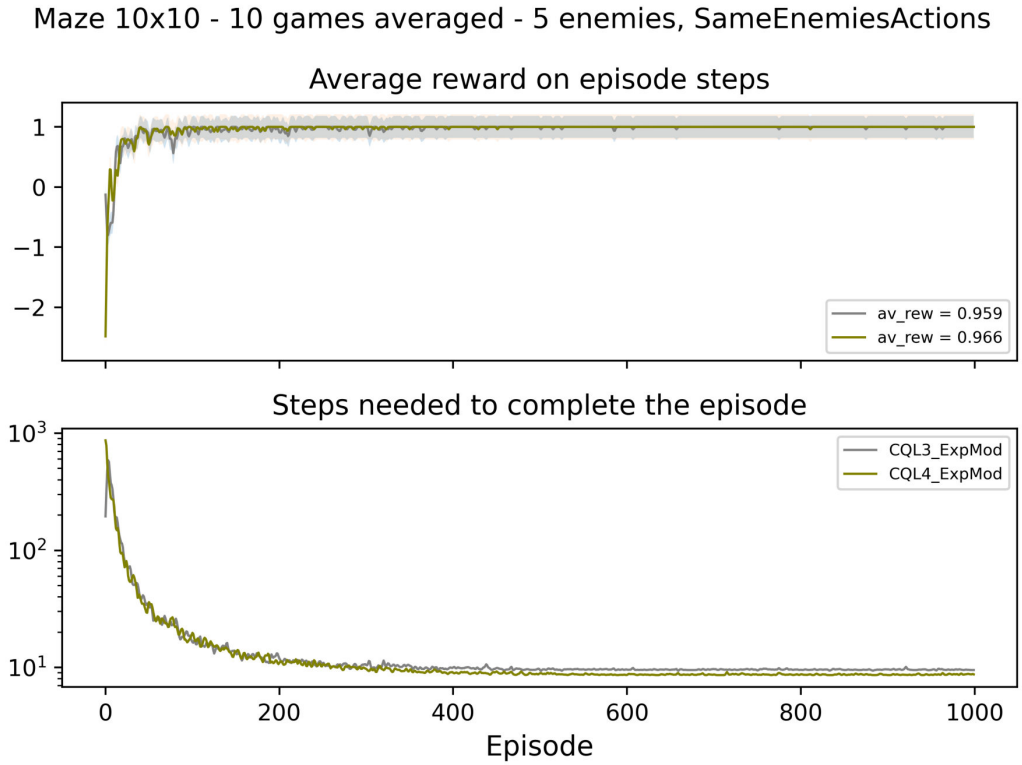


Figure 5.9: *Plot of performance comparison of the average reward and the steps needed for episode; 10x10 maze-like environment, 5 enemies that follow the same default random action sequence. "_noExpMod" indicates no change in the exploitation phase, while "_ExpMod" indicates the presence of such a change, equivalent to the "*" symbol.*

It is interesting to note the impressive capabilities of the algorithms capable of respecting the timeout condition. In Figure 5.9, the trends in the number of steps required to complete the episode appear to be consistently smooth, resulting in a correspondingly stable average reward before the 200th episode. The main instances of defeat are predominantly concentrated in the early stages of the game, as highlighted by the average reward plot. Once again, as previously observed, these trends exhibit remarkable similarities, as do the achieved performances; this observation is reinforced by the fact that while CQL3* is built to explore more than CQL4*, in environments with constraints (walls) and multiple enemies, the available locations for exploration become limited. Consequently, CQL3* exhibits behaviors closely resembling those of CQL4* under such conditions.

Overall, CQL3* and CQL4* consistently demonstrate outstanding performance across all tasks, and apart from the specific scenario examined earlier, CQL3 and CQL4 also obtain excellent results.

Enemies performing random actions in every match

Environment		Average Reward in Algorithms								
Maze size	**Enemies**	**QL**	**CQL1**	**CQL2**	**CQL3**	**CQL4**	**CQL1***	**CQL2***	**CQL3***	**CQL4***
10x10	1	-	-	-	-	-	0.872	-	0.9902	**0.9907**
20x20	1	0.961	0.966	0.973	0.988	0.988	0.976	0.985	0.9937	**0.9961**
50x50	1	-	0.866	0.925	-	-	0.938	**0.976**	0.968	0.972
10x10	5	-0.158	-0.297	-0.142	0.617	0.686	0.21	-0.267	**0.966**	0.965
20x20	5	-	-2.899	-	-1.613	-3.173	0.766	0.645	**0.976**	**0.976**
50x50	5	-	-	-	-	-	0.708	0.858	0.839	**0.91**

Table 5.4: *Comparison of average reward performance: Classic Q-Learning (QL) vs. Causal Q-Learning Variants with exploration modification (CQL1 to CQL4) vs. Causal Q-Learning Variants with modifications to both exploration and exploitation phases (CQL1* to CQL4*). Evaluation in less challenging maze environments where enemies take random actions. The symbol "-" indicates that a timeout occurred.*

In tasks of this nature, the limitations of Q-Learning in dynamic environments become very obvious. Q-Learning manages to meet the required time limit only in two tasks, and it yields satisfactory results in just one of them, specifically in the 20x20 maze with a single enemy scenario. Notably, in the initial and final tasks, as detailed in Table 5.4, only the causal algorithms with modified exploitation do not surpass the fixed time limit.

As a case study, 50x50 maze with 5 enemies is selected, primarily because it's interesting to observe that only the algorithms incorporating the modified exploitation strategy prove to be valids in this context. This underscores the significance of this modification phase. A similar pattern happens in another task, the 10x10 maze with a single enemy, although CQL2* is excluded due to exceeding the time limit.

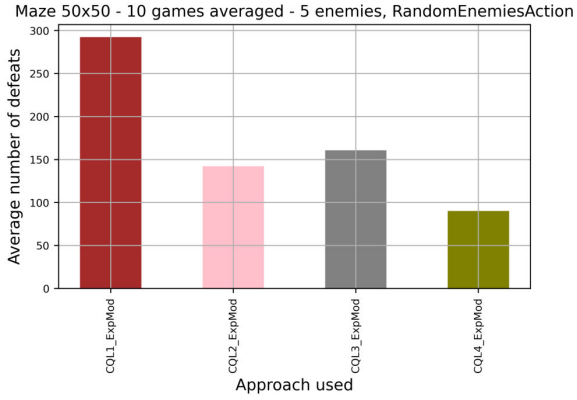


Figure 5.10: *Plot of how many times on average does the single algorithm lose in each game; 50x50 maze-like environment, 5 enemies that take random actions. "_noExpMod" indicates no change in the exploitation phase, while "_ExpMod" indicates the presence of such a change, equivalent to the "*" symbol.*

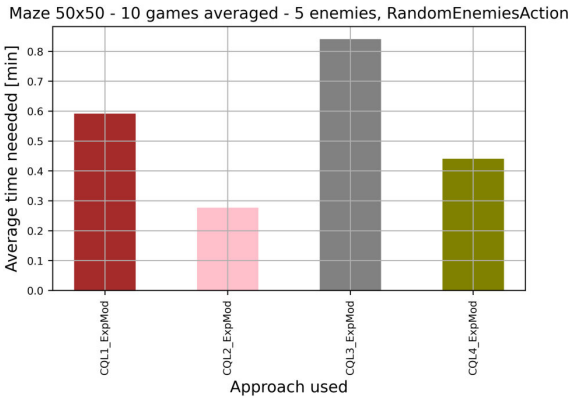


Figure 5.11: *Plot of the average game completion time for the single algorithm; 50x50 maze-like environment, 5 enemies that take random actions. "_noExpMod" indicates no change in the exploitation phase, while "_ExpMod" indicates the presence of such a change, equivalent to the "*" symbol.*

Consequently, the performance, as depicted in terms of average rewards in Figure 5.12, are notably considerable. Once again, as previously explained, the principal defeats are concentrated in the early stages of the game for all four valid algorithms. It's worth noting that while the trend for CQL4* may exhibit less smoothness compared to, for instance, CQL2*, it manages to achieve a higher average reward.

Furthermore, the average number of defeats for each implemented algorithm is presented in Figure 5.10. It's important to highlight that the average time required to complete a game for each algorithm remains under one minute, indicating the remarkable speed and optimization of these algorithms in finding solutions, despite the fact that there are 5 enemies in the environment and that this is of significant size, bringing the problem into the class of low reward problems.

Restricting the focus to CQL3* and CQL4*, it is evident that in all the examined tasks, these algorithms consistently achieve remarkably high average rewards. The key distinguishing factor between them lies in the specific case study selected; the former attains an average reward of 0.839, while the latter reaches 0.91. This substantial difference can be elucidated by considering that CQL4* (and CQL4) prioritizes less exploration of the environment, whereas CQL3* (and CQL3) embraces exploration, resulting in superior performance by directly targeting the objective without risking being trapped in parts of the environment that could lead to defeat.

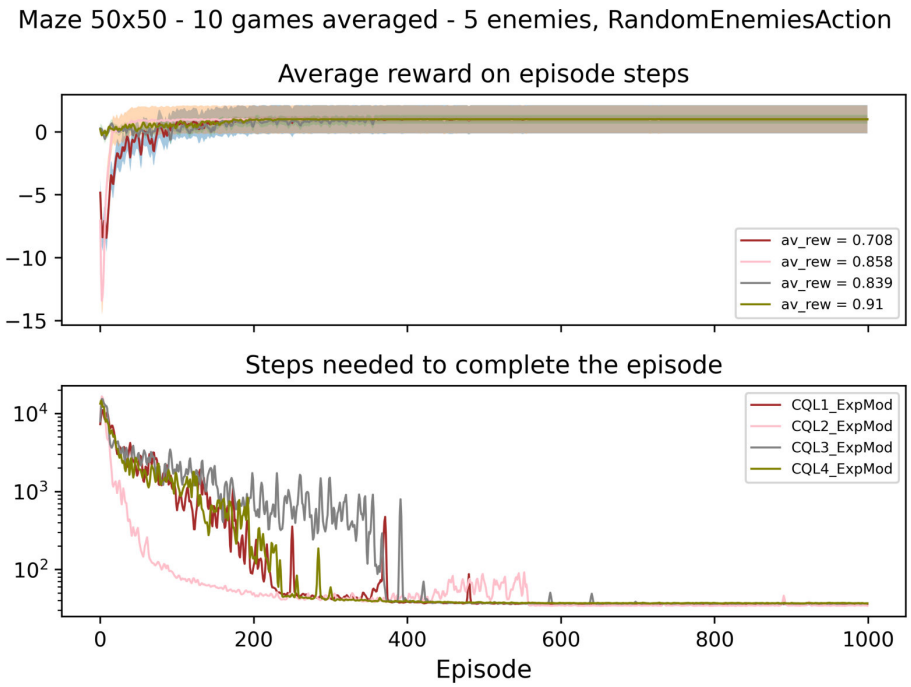


Figure 5.12: *Plot of performance comparison of the average reward and the steps needed for episode; 50x50 maze-like environment, 5 enemies that take random actions. "_noExpMod" indicates no change in the exploitation phase, while "_ExpMod" indicates the presence of such a change, equivalent to the "*" symbol.*

5.3 Highly Complexity Environments

In this section, the results obtained in higher complex environments are presented; relevant observations and reflections on the overall results and the specific case studies chosen are provided.

5.3.1 Grid

Enemies performing same actions in every match

Environment		Average Reward in Algorithms								
Grid size	**Enemies**	**QL**	**CQL1**	**CQL2**	**CQL3**	**CQL4**	**CQL1***	**CQL2***	**CQL3***	**CQL4***
10x10	10	-	-	-	0.889	0.843	0.268	-	**0.968**	0.967
20x20	10	-	-	-	0.942	0.822	0.733	0.749	0.97	**0.974**
50x50	10	0.731	0.71	0.782	0.93	**0.946**	0.729	0.826	0.943	0.943
100x100	10	-	-	0.745	-	-	-	**0.768**	-	-
10x10	20	-	-	-	-	-	-	-	**0.916**	0.913
20x20	20	-	-	-	-	-	0.158	0.188	**0.961**	0.945
50x50	20	-	-	-	-	-	-0.026	0.57	0.753	**0.773**
100x100	20	-	-	-	-	-	-	**0.569**	-	-
50x50	50	-	-	-	-	-	-	-0.512	**0.619**	0.413
100x100	50	-	-	-	-	-	-	-	-	-

Table 5.5: *Comparison of average reward performance: Classic Q-Learning (QL) vs. Causal Q-Learning Variants with exploration modification (CQL1 to CQL4) vs. Causal Q-Learning Variants with modifications to both exploration and exploitation phases (CQL1* to CQL4*). Evaluation in more challenging grid environments where enemies follow a fixed sequence of actions. The symbol "-" indicates that a timeout occurred.*

The outcomes presented in Table 5.5 underscore the significance and stringency of imposing a timeout condition when assessing RL algorithms in environments of this nature. When evaluating the performance of Q-Learning, only one scenario proves valid: the 50x50 grid with 10 enemies. This environment strikes a balance, providing a sufficient degree of complexity to facilitate few agent-enemy interactions while still avoiding the challenges associated with an exceedingly complex sparse reward problem. Nevertheless, the achieved average reward in this case remains significantly lower than that achieved with causal algorithms.

As a case study, the task wherein the agent navigates a 100x100 grid-like environment with 10 enemies is examined. This task is noteworthy because only the CQL2 algorithm, both without modified exploitation and with modified exploitation, produces valid results in terms of computational time. The average reward and the steps required to complete the episode are depicted in Figure 5.13. The performances of these algorithms are generally similar, but the one with modified exploitation outperforms the other because it avoids defeat in the latter stages of the game, as evident in the plot. Conversely, the algorithm with modified exploitation experiences more losses in the early game stages. This suggests a balance, which is also apparent in Figure 5.14, where the average number of defeats per game for each algorithm hovers around 250.

For the remaining algorithms that exceed the prescribed computational time limit, Figure 5.15 provides an overview of the timeouts that occurred, all of which are greater than 4.

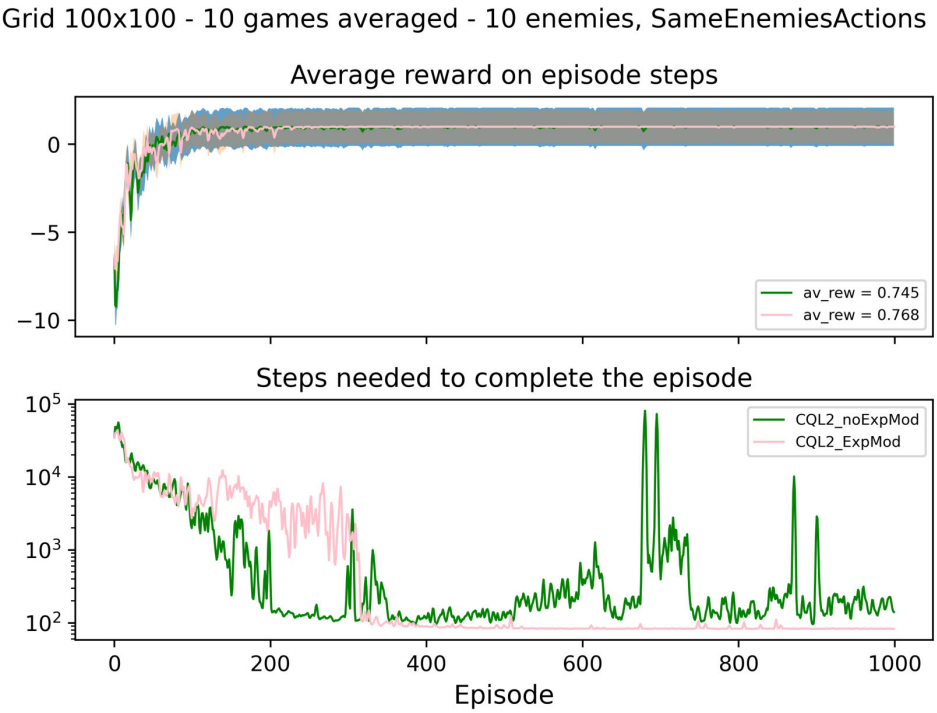


Figure 5.13: *Plot of performance comparison of the average reward and the steps needed for episode; 100x100 grid-like environment, 10 enemies that follow the same default random action sequence. "_noExpMod" indicates no change in the exploitation phase, while "_ExpMod" indicates the presence of such a change, equivalent to the "*" symbol.*

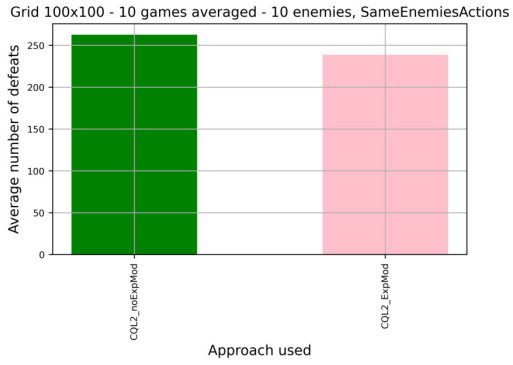


Figure 5.14: *Plot of how many times on average does the single algorithm lose in each game; 100x100 grid-like environment, 10 enemies that follow the same default random action sequence. "_noExpMod" indicates no change in the exploitation phase, while "_ExpMod" indicates the presence of such a change, equivalent to the "*" symbol.*

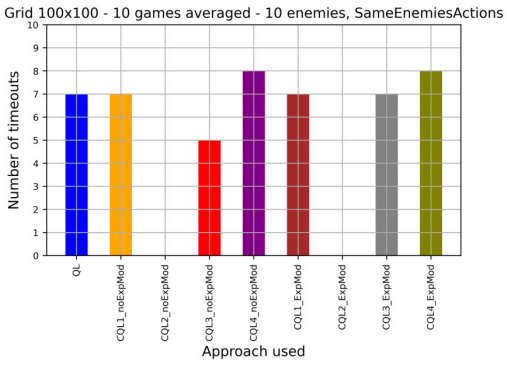


Figure 5.15: *Plot of the number of timeouts for the single algorithm; 100x100 grid-like environment, 10 enemies that follow the same default random action sequence. "_noExpMod" indicates no change in the exploitation phase, while "_ExpMod" indicates the presence of such a change, equivalent to the "*" symbol.*

In general, the CQL2* algorithm demonstrates its suitability and effectiveness primarily in expansive environments. This can be attributed to its inherent design, as it actively seeks to explore the environment comprehensively. Interestingly, during the exploration phase, CQL2* may opt for actions that place it in direct conflict with enemies. In contrast, both CQL3 and CQL4 are programmed to avoid actions leading to a game over scenario, but for these tasks they don't respect the timeout condition. This algorithm's power becomes further evident when examining its performance in a task involving a 100x100 grid with 20 enemies, where it stands as the unique performer with a valid outcome. Nevertheless, it is worth noting that the task involving a 100x100 grid with 50 enemies might prove exceedingly challenging for any algorithm under consideration.

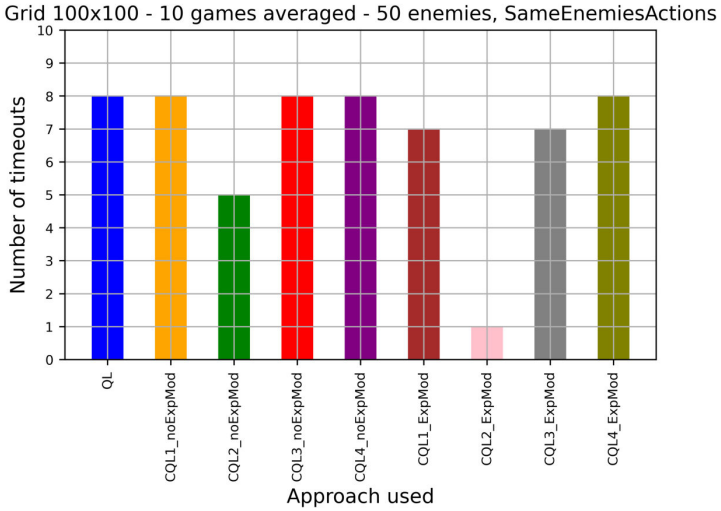


Figure 5.16: *Plot of the number of timeouts for each algorithm; 100x100 grid-like environment, 50 enemies that follow the same default random action sequence. "_noExpMod" indicates no change in the exploitation phase, while "_ExpMod" indicates the presence of such a change, equivalent to the "*" symbol.*

As indicated in Table 5.5, none of the algorithms respect to the time limit in the 100x100 grid-like environment with 50 enemies, which presents a challenge due to its sparse reward nature and the presence of several enemies. Figure 5.16 provides an overview of the number of timeouts that occurred for each algorithm, with only CQL2* showing promising behavior. The conclusions drawn from this task suggest that the implemented algorithms are not sufficiently robust to handle the complexity of this environment. To effectively evaluate the performance of these algorithms in such scenarios, it becomes necessary to extend the timeout limit.

Enemies performing random actions in every match

Environment		Average Reward in Algorithms								
Grid size	**Enemies**	**QL**	**CQL1**	**CQL2**	**CQL3**	**CQL4**	**CQL1***	**CQL2***	**CQL3***	**CQL4***
10x10	10	-3.637	-3.444	-3.625	-1.576	-1.429	-	-0.749	0.961	**0.964**
20x20	10	-1.747	-1.606	-2.121	-1.337	-1.101	0.64	0.666	0.977	**0.978**
50x50	10	-	-	-	-	-	-	0.685	**0.701**	0.668
100x100	10	-	-	-	-	-	-	**0.403**	-	-
10x10	20	-6.923	-6.665	-6.861	-3.876	-4.317	-2.472	-2.619	0.915	**0.918**
20x20	20	-8.7	-8.582	-9.684	-6.39	-5.612	-	-	**0.955**	0.949
50x50	20	-	-	-	-	-	-	0.43	**0.688**	0.662
100x100	20	-	-	-	-	-	-	**-0.503**	-	-
50x50	50	-	-	-	-	-	-	-	**0.153**	-
100x100	50	-	-	-	-	-	-	-	-	-

Table 5.6: *Comparison of average reward performance: Classic Q-Learning (QL) vs. Causal Q-Learning Variants with exploration modification (CQL1 to CQL4) vs. Causal Q-Learning Variants with modifications to both exploration and exploitation phases (CQL1* to CQL4*). Evaluation in more challenging grid environments where enemies take random actions. The symbol "-" indicates that a timeout occurred.*

The situation in these experiments mirrors what it has been explained earlier. Table A.1 displays the achieved average rewards. Similar to previous cases, the environment featuring a 100x100 grid with 50 enemies does not have any algorithm that adheres to the specified computation time limit.

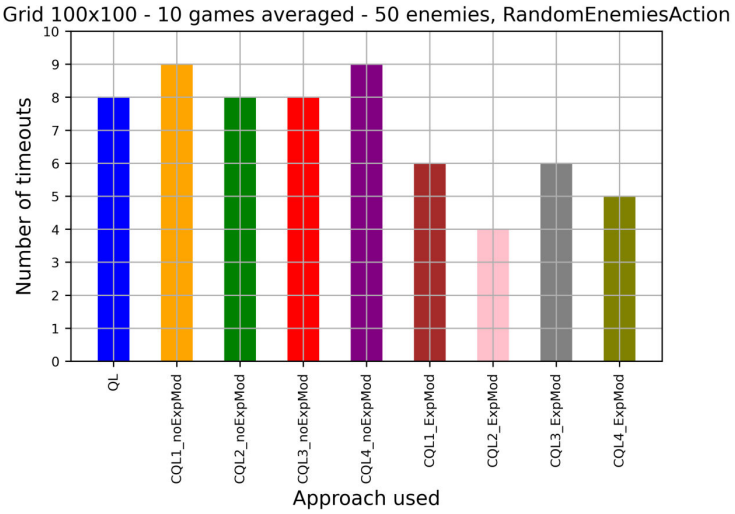


Figure 5.17: *Plot of the number of timeouts for each algorithm; 100x100 grid-like environment, 50 enemies that take random actions. "_noExpMod" indicates no change in the exploitation phase, while "_ExpMod" indicates the presence of such a change, equivalent to the "*" symbol.*

However, in this instance, the number of timeouts for each algorithm is more pronounced, as indicated in Figure 5.17. Specifically, just like before, CQL2* exhibits the lowest number of timeouts, which is four in this case. The same conclusions drawn for this type of task earlier remain applicable; specifically, CQL2* underscores its robustness, particularly when the context involves a wide environment. Notably, in the scenario of a 50x50 grid with 50 enemies, CQL3* emerges as the exclusive algorithm able to achieve satisfactory performance. In general, it is important to recognize that the average rewards in this case are consistently lower compared to the various task types we have considered so far, underscoring the notable level of complexity involved.

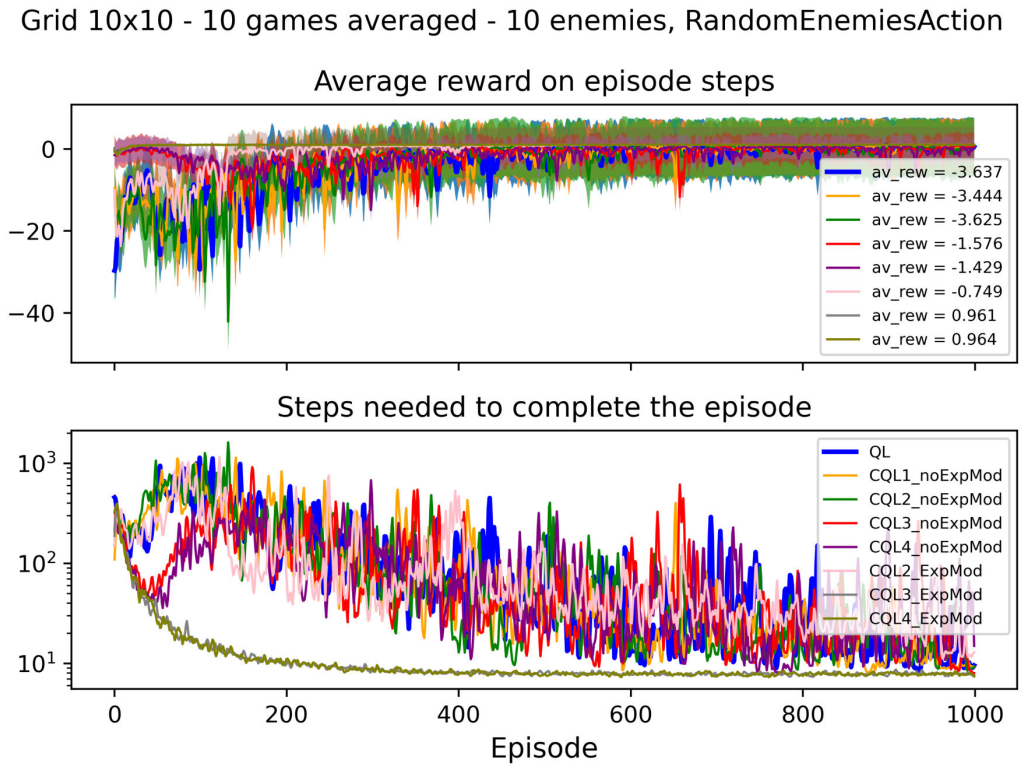


Figure 5.18: *Plot of performance comparison of the average reward and the steps needed for episode; 10x10 grid-like environment, 10 enemies that take random actions. "_noExpMod" indicates no change in the exploitation phase, while "_ExpMod" indicates the presence of such a change, equivalent to the "*" symbol.*

As a case study, the 10x10 grid with 10 enemies is selected to emphasize once more the effectiveness of CQL3* and CQL4*. As depicted in Figure 5.18, these two algorithms stand out as the only ones achieving a positive average reward, and moreover, their average rewards are nearly optimal. Other approaches prove unsuitable for this challenging environment, characterized by its small size and a high density of enemies, as showed also in Figure 5.19 and Figure 5.20.

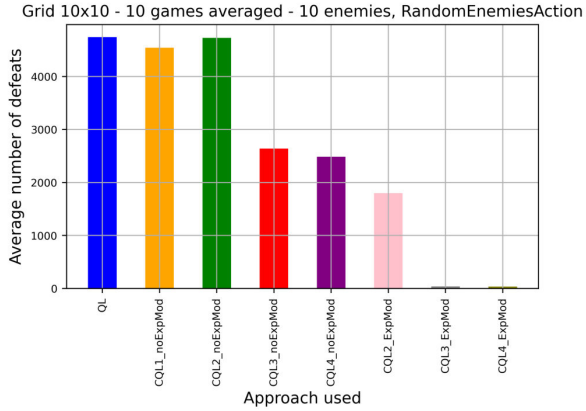


Figure 5.19: *Plot of how many times on average does the single algorithm lose in each game; 10x10 grid-like environment, 10 enemies that take random actions. "_noExpMod" indicates no change in the exploitation phase, while "_ExpMod" indicates the presence of such a change, equivalent to the "*" symbol.*

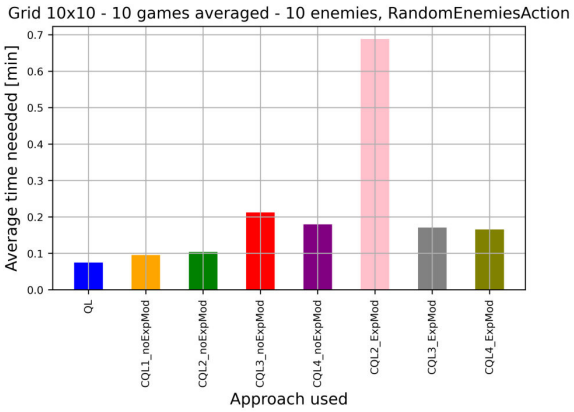


Figure 5.20: *Plot of the average game completion time for the single algorithm; 10x10 grid-like environment, 10 enemies that take random actions. "_noExpMod" indicates no change in the exploitation phase, while "_ExpMod" indicates the presence of such a change, equivalent to the "*" symbol.*

Once again, the trends exhibited by these two algorithms are consistently smooth, and the number of defeats, when compared to alternative approaches, is minimal. This observation holds true for the scenario with 20 enemies in the 10x10 grid as well, where CQL4* attains an impressive average reward of 0.918, signifying outstanding performance. As previously mentioned, in environments characterized by being limited in size and by the presence of numerous opponents, CQL3* and CQL4* exhibit remarkably similar behavior.

5.3.2 Maze

Enemies performing same actions in every match

Environment		Average Reward in Algorithms								
Maze size	**Enemies**	**QL**	**CQL1**	**CQL2**	**CQL3**	**CQL4**	**CQL1***	**CQL2***	**CQL3***	**CQL4***
10x10	10	-	-	-	-	-	-	-	**0.9**	-
20x20	10	0.433	0.579	0.61	0.937	0.924	0.6	0.64	**0.949**	0.942
50x50	10	0.585	0.583	-	0.913	0.899	0.648	0.799	0.906	**0.926**
100x100	10	-	-	-	-	-	-	**0.663**	-	-
10x10	20	-	-	-	-	-	-	-	-	-
20x20	20	-	-	-	-	-	-	-	-	-
50x50	20	-	-	-	0.681	-	-	0.478	**0.749**	0.708
100x100	20	-	-	-	-	-	-	**0.452**	-	-
50x50	50	-	-	-	-	-	-	-	**0.24**	-
100x100	50	-	-	-	-	-	-	-	-	-

Table 5.7: *Comparison of average reward performance: Classic Q-Learning (QL) vs. Causal Q-Learning Variants with exploration modification (CQL1 to CQL4) vs. Causal Q-Learning Variants with modifications to both exploration and exploitation phases (CQL1* to CQL4*). Evaluation in more challenging maze environments where enemies follow a fixed sequence of actions. The symbol "-" indicates that a timeout occurred.*

The results reported in Table 5.7 show an increasing of the complexity of the task examined, in particular Q-Learning is able to respect the required computation time only in two task, which not by chance are the tasks that have fewer enemies and that have the environment neither too small (too many agent-enemy interactions) nor too large (sparse reward problem).

In these experiments, in three tasks, none of the algorithms adhere to the specified computation time limit. For the 10x10 and 20x20 mazes with 20 enemies, the likely reason could be the substantial number of enemies present in these relatively compact environments. The occurrences of timeouts for these two tasks are detailed in Figures 5.21 and 5.22.

As for the 100x100 maze with 50 enemies, the issue remains consistent with what was explained earlier, compounded by the fact that the environment's size contributes to a sparse reward problem. The number of timeouts for each algorithm in this task is displayed in Figure 5.23.

As previously discussed, addressing this challenge may necessitate the development of more powerful approaches and/or increasing the timeout duration; particularly, when dealing with constrained environments, such as 10x10 and 20x20 grids, a logical approach could involve creating a novel algorithm inspired by the operational principles of CQL3* and CQL4*. On the other hand, in wide environments like 100x100 or larger grids/mazes, it might be beneficial to explore strategies akin to those employed by CQL2*. These proposed solution, to keep coherence with the work, must first be tested using the same timeout conditions; if the results cannot yet be appreciated, the computational time limit must be increased.

The proposed case study for tasks of this nature involves a 50x50 maze-like environment with 50 enemies, and the outcome is interesting because only one algorithm managed to respect the timeout condition: the CQL3*. Figure 5.24 makes it evident that the ten randomly generated games were highly challenging. Despite the apparent effectiveness of CQL3* and CQL4*, only the first meets the specified computation time limit, achieving

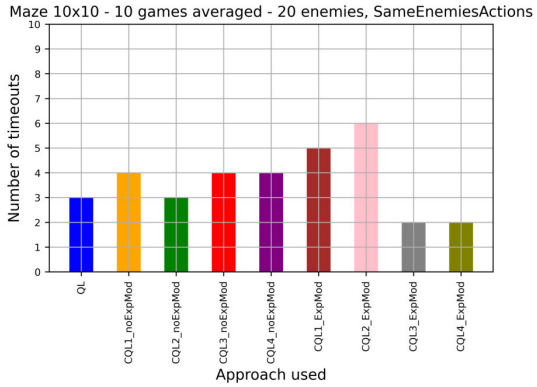


Figure 5.21: *Plot of the number of timeouts for the single algorithm; 10x10 maze-like environment, 20 enemies that follow the same default random action sequence. "_noExpMod" indicates no change in the exploitation phase, while "_ExpMod" indicates the presence of such a change, equivalent to the "*" symbol.*

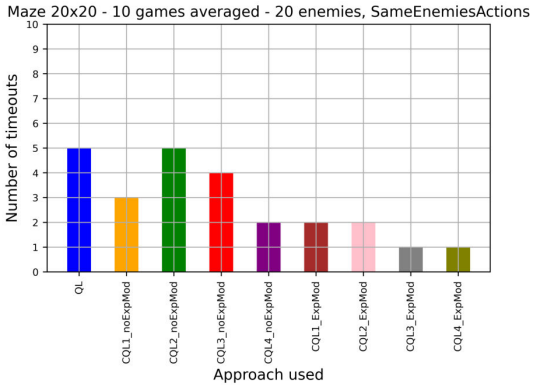


Figure 5.22: *Plot of the number of timeouts for the single algorithm; 20x20 maze-like environment, 20 enemies that follow the same default random action sequence. "_noExpMod" indicates no change in the exploitation phase, while "_ExpMod" indicates the presence of such a change, equivalent to the "*" symbol.*

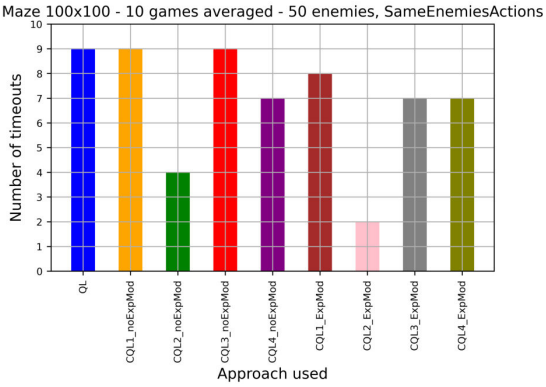


Figure 5.23: *Plot of the number of timeouts for the single algorithm; 100x100 maze-like environment, 50 enemies that follow the same default random action sequence. "_noExpMod" indicates no change in the exploitation phase, while "_ExpMod" indicates the presence of such a change, equivalent to the "*" symbol.*

a modest average reward of 0.24. Additionally, it leads an average of approximately 750 defeats per game, as illustrated in Figure 5.25. The performance trend of the surviving algorithm displays considerable variability in the initial stages, but after around 200 episodes, it converges toward the optimal reward. The jagged trend observed in the later stages of the games aligns with the strategy of avoiding enemies and finding new paths to reach the goal position, both in the exploration and exploitation phases.

Other approaches prove unsuitable, as shown in Figure 5.26, where the number of timeouts for algorithms without modified exploitation, including classic Q-Learning, is unacceptably high; whereas, algorithms with modified exploitation experience fewer timeouts,

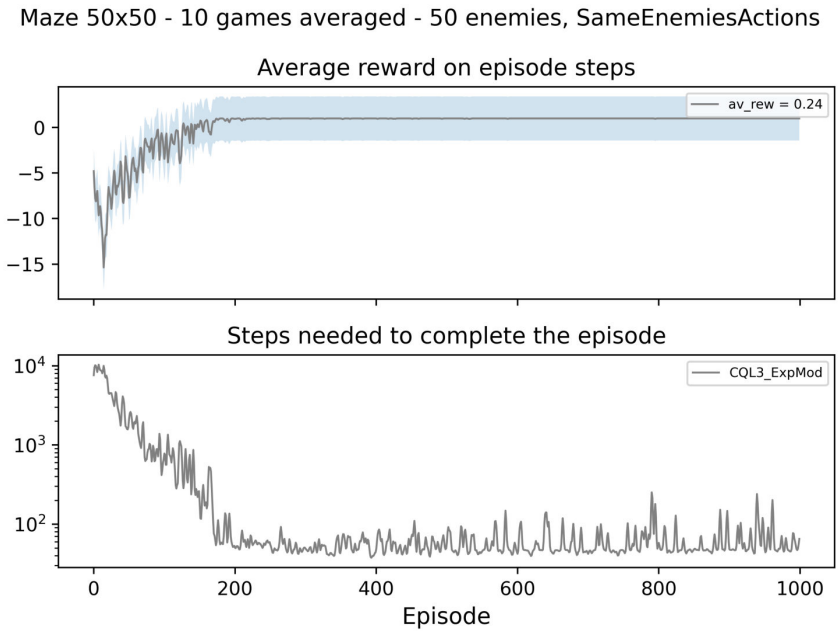


Figure 5.24: *Plot of performance comparison of the average reward and the steps needed for episode; 50x50 maze-like environment, 50 enemies that follow the same default random action sequence. "_noExpMod" indicates no change in the exploitation phase, while "_ExpMod" indicates the presence of such a change, equivalent to the "*" symbol.*

particularly CQL4*.

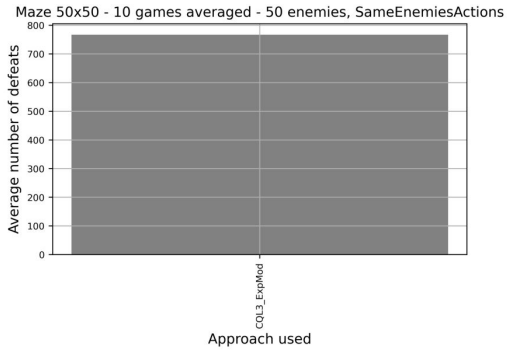


Figure 5.25: *Plot of how many times on average does the single algorithm lose in each game; 50x50 maze-like environment, 50 enemies that follow the same default random action sequence. "_noExpMod" indicates no change in the exploitation phase, while "_ExpMod" indicates the presence of such a change, equivalent to the "*" symbol.*

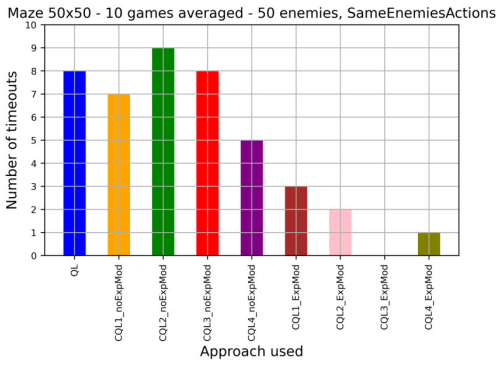


Figure 5.26: *Plot of the number of timeouts for the single algorithm; 50x50 maze-like environment, 50 enemies that follow the same default random action sequence. "_noExpMod" indicates no change in the exploitation phase, while "_ExpMod" indicates the presence of such a change, equivalent to the "*" symbol.*

Enemies performing random actions in every match

Environment		Average Reward in Algorithms								
Maze size	**Enemies**	**QL**	**CQL1**	**CQL2**	**CQL3**	**CQL4**	**CQL1***	**CQL2***	**CQL3***	**CQL4***
10x10	10	-7.216	-6.486	-	-9.453	-	-	-	**0.851**	0.831
20x20	10	-2.532	-2.826	-4.406	-1.21	-1.168	0.276	0.518	**0.956**	0.952
50x50	10	-	-	-	-	-	-0.111	-	0.752	0.524
100x100	10	-	-	-	-	-	-	**0.359**	-	-
10x10	20	-15.433	-15.412	-	-	-	-	-	**0.568**	0.56
20x20	20	-16.231	-16.574	-	-	-	-	-	**0.871**	0.858
50x50	20	-	-	-	-	-	-	0.214	0.379	**0.471**
100x100	20	-	-	-	-	-	-	-	-	-
50x50	50	-	-	-	-	-	-	-	**0.31**	-
100x100	50	-	-	-	-	-	-	-	-	-

Table 5.8: *Comparison of average reward performance: Classic Q-Learning (QL) vs. Causal Q-Learning Variants with exploration modification (CQL1 to CQL4) vs. Causal Q-Learning Variants with modifications to both exploration and exploitation phases (CQL1* to CQL4*). Evaluation in more challenging maze environments where enemies take random actions. The symbol "-" indicates that a timeout occurred.*

Table A.2 illustrates that both classic Q-Learning and Causal Q-Learning approaches without modified exploitation are unsuitable for these highly demanding tasks. These tasks involve the agent navigating through maze environments of varying sizes with a significant number of enemies. Notably, the performance achieved by classic Q-Learning and Causal Q-Learning without modified exploitation are remarkably similar. When they manage to adhere to the specified timeout condition, their average rewards closely align, and when Q-Learning fails to do so, also the Causal Q-Learning algorithms without modified exploitation fail. As previously discussed, the higher the number of early-game defeats, the more complex it becomes to discover the optimal path to reach the goal position. For instance, if we examine CQL3 and CQL4, during the exploration phase, the risk of losing is minimal. However, when moving to the exploitation phase using the ϵ-greedy approach, even for a brief period, the Q-Table deteriorates rapidly, especially given the significant presence of adversaries within the environment. A similar challenge is encountered with classical Q-Learning, which presents a worse situation due to its absence of adjustments in either the exploration or exploitation phases.

On the other hand, Causal Q-Learning algorithms with modified exploitation yield impressive results. The only exceptions are the 100x100 mazes with 20 and 50 enemies, where they struggled to meet the prescribed timeout condition. Figures 5.27 and 5.28 provide insights into the number of timeouts experienced by each algorithm. As observed in previous instances, the CQL2* algorithm serves as a promising foundation for the development of a more powerful algorithm suitable for very large environments. Its minimal number of timeouts suggests its potential as a starting point for such endeavors.

For this category of experiments, two specific case studies were undertaken involving maze-like environments, each featuring 20 enemies, with dimensions of 10x10 and 20x20. Remarkably, the performance outcomes for both cases were highly similar. Among the algorithms assessed, only classic Q-Learning, CQL1, CQL3*, and CQL4* managed to meet the specified timeout condition in both scenarios. Examining the number of defeats incurred in both selected cases, as depicted in Figures 5.29 and 5.30, the situation appears to be equivalent, even when considering only the visual plots. The latter two algorithms

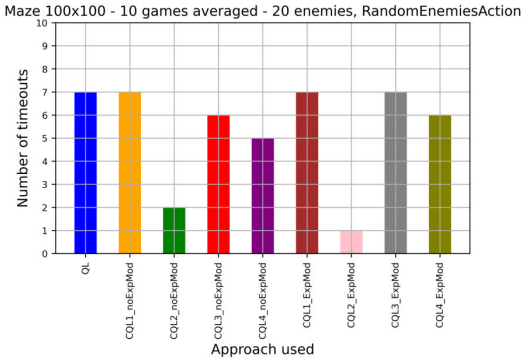


Figure 5.27: *Plot of the number of timeouts for the single algorithm; 100x100 maze-like environment, 20 enemies that take random actions. "_noExpMod" indicates no change in the exploitation phase, while "_ExpMod" indicates the presence of such a change, equivalent to the "*" symbol.*

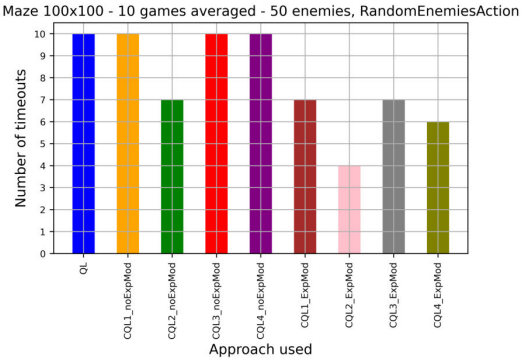


Figure 5.28: *Plot of the number of timeouts for the single algorithm; 100x100 maze-like environment, 50 enemies that take random actions. "_noExpMod" indicates no change in the exploitation phase, while "_ExpMod" indicates the presence of such a change, equivalent to the "*" symbol.*

experience negligible losses, while the first two demonstrate approximately the same number of defeats, around 17500. These results are confirmed by examination Figures 5.31 and 5.32, where average rewards and the steps required to complete episodes are reported. The first two algorithms fail to exhibit a smooth trend, are unable to avoid enemies, and lead unacceptable performance. Conversely, the latter two algorithms demonstrate behavior consistent with the overall thesis: a smooth trend converging around the 300th episode, resulting in excellent results for these highly complex tasks.

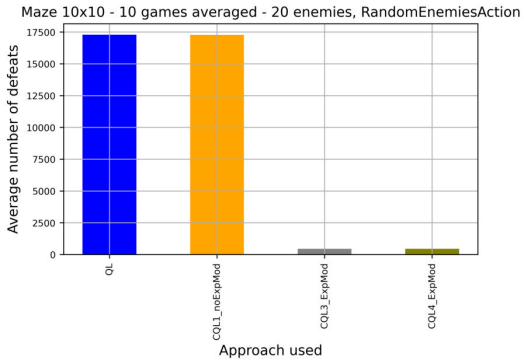


Figure 5.29: *Plot of how many times on average does the single algorithm lose in each game; 10x10 maze-like environment, 20 enemies that take random actions. "_noExpMod" indicates no change in the exploitation phase, while "_ExpMod" indicates the presence of such a change, equivalent to the "*" symbol.*

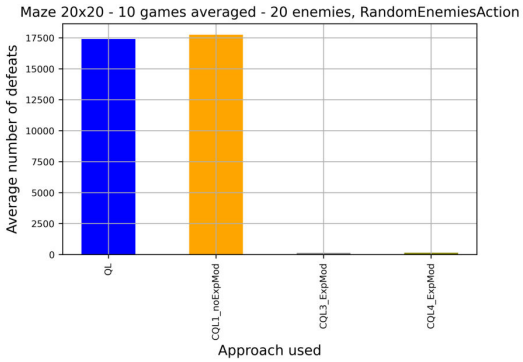


Figure 5.30: *Plot of how many times on average does the single algorithm lose in each game; 20x20 maze-like environment, 50 enemies that take random actions. "_noExpMod" indicates no change in the exploitation phase, while "_ExpMod" indicates the presence of such a change, equivalent to the "*" symbol.*

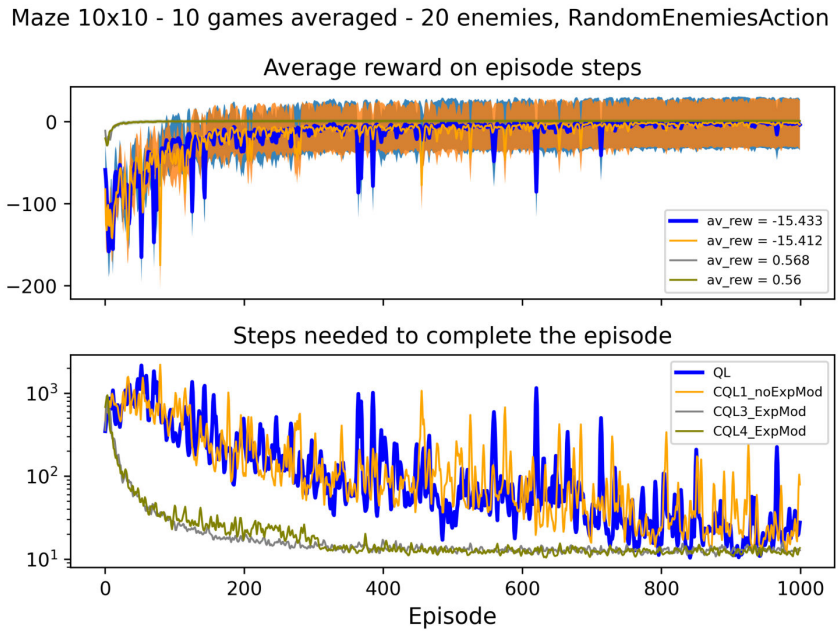


Figure 5.31: *Plot of performance comparison of the average reward and the steps needed for episode; 10x10 maze-like environment, 20 enemies that take random actions. "_noExpMod" indicates no change in the exploitation phase, while "_ExpMod" indicates the presence of such a change, equivalent to the "*" symbol.*

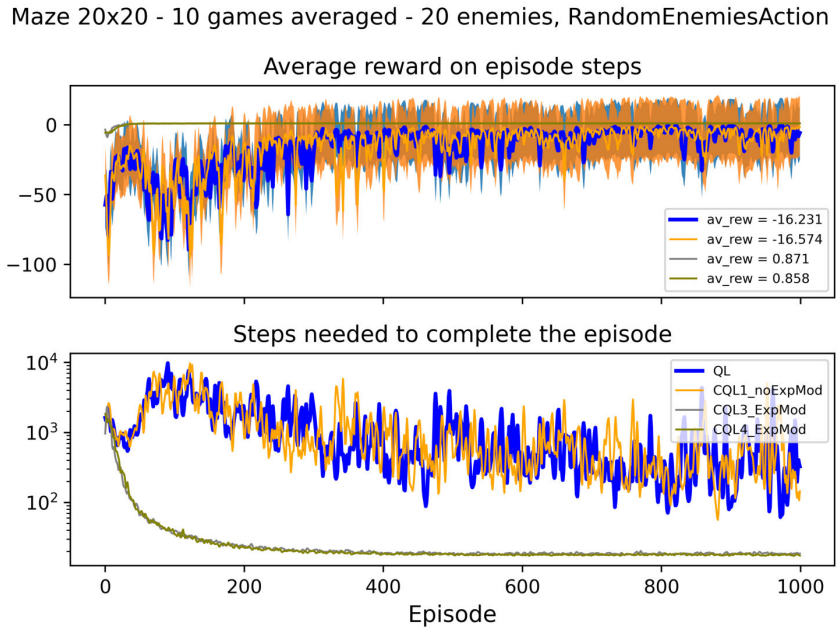


Figure 5.32: *Plot of performance comparison of the average reward and the steps needed for episode; 20x20 maze-like environment, 20 enemies that take random actions. "_noExpMod" indicates no change in the exploitation phase, while "_ExpMod" indicates the presence of such a change, equivalent to the "*" symbol.*

5.4 Frozen Lake

An additional set of experiments was conducted in a Frozen Lake-like environment to provide another point of comparison. This environment was generated using the same script as the others, with the positions of the goal and enemies predefined, and the enemies remaining stationary. Figure 5.33 provides a visual representation of this setup.

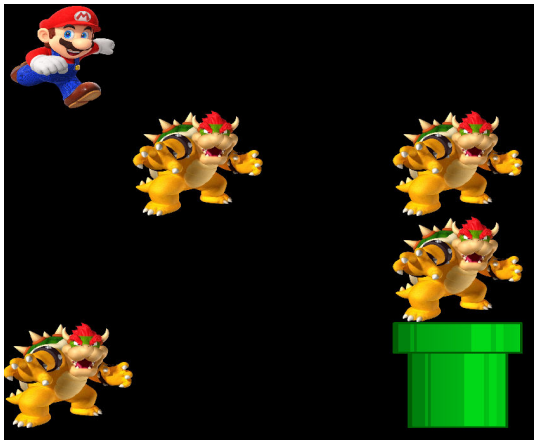


Figure 5.33: *Custom graphical user interface: Frozen Lake environment.*

In this experiment, using the standard Frozen Lake environment gave a reward of 0 on failure. Later, this experiment was extended to include a scenario where the reward for defeat was changed to -1. This alteration was made to enable a comprehensive comparative assessment between the two approaches. It is important to note that in both cases examined, the averages were calculated based on 10 simulations, even though the environment itself remained consistent. To better evaluate the results obtained in the Frozen Lake environment with the typical baseline, with a reward for defeat equal to 0, the average reward obtained, which is always 1, is divided by the number of steps necessary to complete the single episode.

While Figure 5.34 suggests that Q-Learning outperforms other methods in terms of average reward divided by steps per episode, a deeper analysis, excluding CQL1-2 with the exploitation modification, reveals that Q-Learning requires $\frac{1}{0.14915} = 6.7$ steps, whereas, for instance, CQL3 achieves $\frac{1}{0.1477} = 6.77$ steps. Therefore, the differences are relatively minor. However, Figure 5.36 illustrates the average losses, highlighting that classic Q-Learning incurs nearly 2000 losses, emphasizing its inefficiency in this aspect.

In an environment where defeat results in a reward of -1, as depicted in Figure 5.35 and Figure 5.37, Q-Learning's efficiency decreases significantly compared to earlier results. When considering Causal Q-Learning algorithms for this type of environment, it should be noted that achieving optimal performance depends primarily on selecting actions in the exploration phase that do not lead to game over. In contrast, for other algorithms where there is a higher probability that actions in the exploration phase will lead to a game over, their performance tends to align more closely with traditional Q-Learning. To clarify, the current task involves static enemies, simplifying the process of finding the optimal path; given the simplicity of the environment, this can also be done only during the exploration phase, which is the most relevant in the early stages of the game. Although the exploit phase in CQL3 and CQL4 can potentially lead to game over, upgrading the

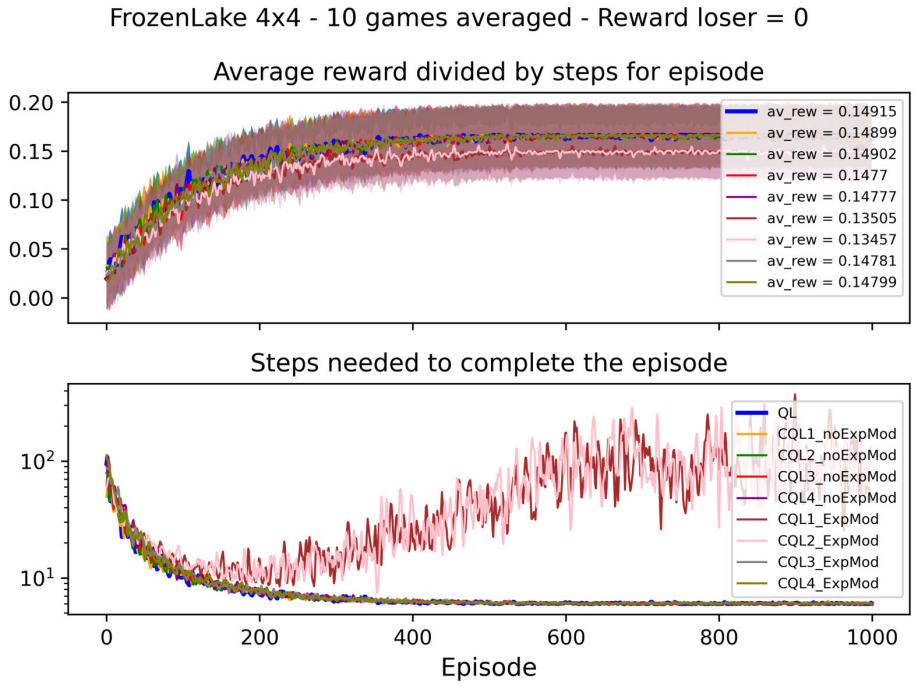


Figure 5.34: *Plot of performance comparison of the average reward divided by the steps taken and the steps needed for episode; Frozen Lake-like environment.*
Reward for defeat = 0. "_noExpMod" indicates no change in the exploitation phase, while "_ExpMod" indicates the presence of such a change, equivalent to the "" symbol.*

Q-Table proves advantageous in these cases, because the enemies are stationary.

It is important to note that CQL3* and CQL4* are practically invulnerable in this environment, whereas CQL3 and CQL4 did experience losses, albeit very rarely. In the graphs presented, it may appear that CQl3 and CQL4 algorithms have zero defeats, but this can be attributed to result approximations in the average reward plot and the higher number of defeats experienced by other algorithms in the defeat graph.

Moreover, the findings, Figure 5.34 and Figure 5.35, suggest that under the most favorable circumstances, the performance attained by causal Q-Learning algorithms exhibits a slight advantage over that achieved using the conventional Q-Learning approach. This observation aligns with the idea that within a relatively compact and static environment, the classic Q-Learning approach already achieves optimal outcomes, rendering the implementation of a causal model unnecessary. Furthermore, this observation is strengthened by the previously demonstrated fact that in a relatively modest and uncomplicated environment, the Q-Learning approach remains adequate and effective. However, it is noteworthy that the performances of causal Q-Learning still show improvements over the baseline.

To facilitate better comprehension, the Q-Tables derived from the initial game (with defeat reward set to -1) have been included. The first Q-Table corresponds to the classic Q-Learning method, as depicted in Figure 5.38, while the second Q-Table pertains to CQL4*, as illustrated in Figure 5.39. This comparison aims to provide a visual representation of the learned action-values in both cases. Specifically, this causal Q-Learning (CQL4*) focuses on a particular aspect: it does not engage with the entire environment; to clarify, it exclusively delves into the section where it can avoid losses, as indicated by its well-defined

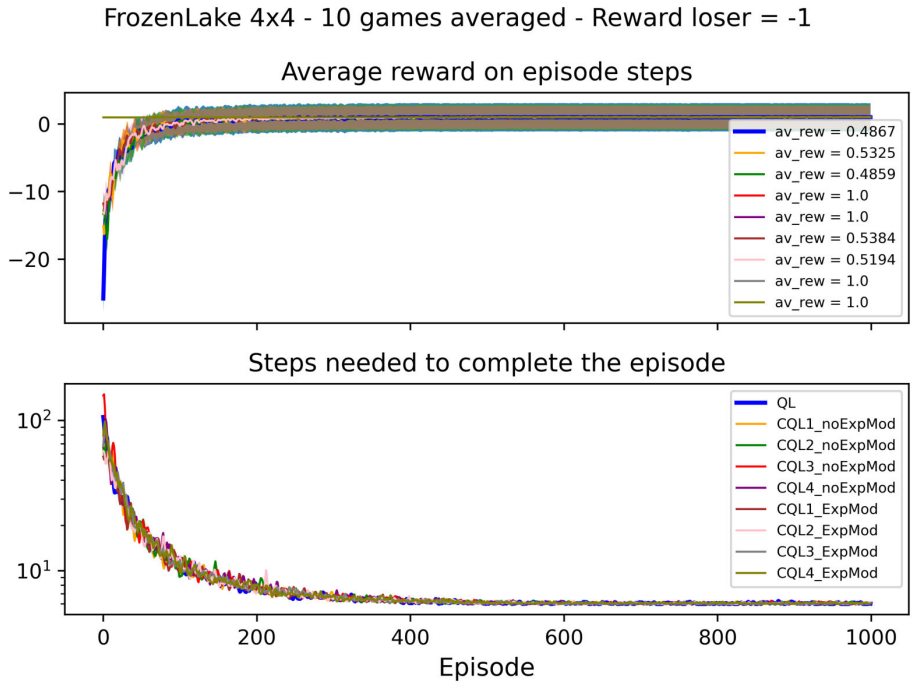


Figure 5.35: *Plot of performance comparison of the average reward and the steps needed for episode; Frozen Lake-like environment. Reward for defeat = -1. "_noExpMod" indicates no change in the exploitation phase, while "_ExpMod" indicates the presence of such a change, equivalent to the "*" symbol.*

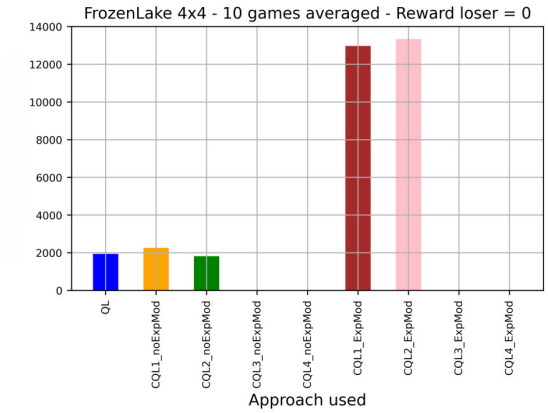


Figure 5.36: *Plot of how many times on average does the single algorithm lose in each game; Frozen Lake-like environment. Reward for defeat = 0. "_noExpMod" indicates no change in the exploitation phase, while "_ExpMod" indicates the presence of such a change, equivalent to the "*" symbol.*

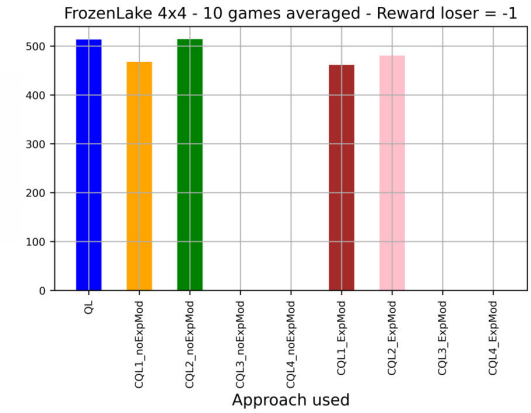


Figure 5.37: *Plot of how many times on average does the single algorithm lose in each game; Frozen Lake-like environment. Reward for defeat = -1. "_noExpMod" indicates no change in the exploitation phase, while "_ExpMod" indicates the presence of such a change, equivalent to the "*" symbol.*

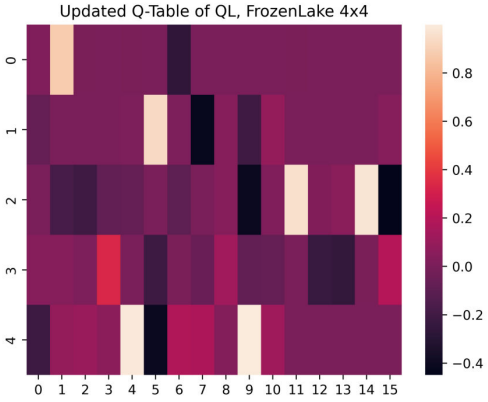


Figure 5.38: *Updated Q-Table by using classic Q-Learning approach.*
Reward for defeat = -1. "_noExpMod" indicates no change in the exploitation phase, while "_ExpMod" indicates the presence of such a change, equivalent to the "" symbol.*

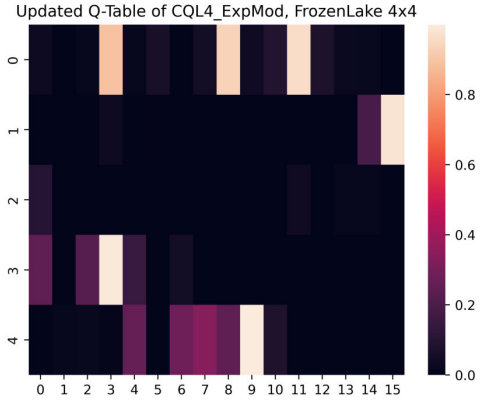


Figure 5.39: *Updated Q-Table by using CQL4 with exploitation modified approach. Reward for defeat = -1. "_noExpMod" indicates no change in the exploitation phase, while "_ExpMod" indicates the presence of such a change, equivalent to the "*" symbol.*

Q-table; due to this factor, the Q-Table provided for this particular algorithm, when the penalty for losing is defined as -1, closely resembles the Q-Table when the reward for the defeat is set at 0.

5.5 Summarized Insights from Extracted Concepts

The objective of this section is to offer a concise recap of the previously presented results and to provide a comprehensive final overview, along with additional insights.

First of all, the introduction of a timeout condition serves as a necessary tool in the evaluation and assessment of RL algorithms. This is due to the inherent requirement for these AI algorithms to engage with the environment for extended duration, which, in certain real-world applications, might become unacceptable. This condition, tailored to the complexity of the task, has provided valuable insights into the limitations of the algorithms under examination. Notably, even though the initial environments considered were relatively manageable (with a 5-minute timeout condition), a not negligible number of timeouts were recorded, especially for the classic Q-Learning approach and also for the causal Q-Learning algorithms lacking the modified exploitation. When the analysis is extended to more complex environments (where the timeout condition is set at 10 minutes), it turns out that some tasks proved to be unsolvable, albeit in limited numbers (where the environment is very challenging). However, many tasks were successfully resolved, primarily by a select few algorithms,m which are the causal Q-Learning algorithms with the modified exploitation strategy. As previously discussed, it is crucial to incrementally adjust the computation time limit to determine the point at which the timeout condition leads to a significant shift in the results.

Still in terms of time considerations, the provided bar plots reveal various aspects related to the computational time required. It is evident from these plots that Q-Learning exhibits in many cases the fastest execution. However, three noteworthy points deserve attention: firstly, all causal Q-Learning algorithms entail consulting the causal model, which is integrated into the causal table. Consequently, in many instances, this factor alone accounts for the divergence in computational time needed. Secondly, as evidenced by the plots depicting the steps required to complete an episode, Q-Learning often displays jagged behavior, with a considerably higher number of steps compared to others. This observation prompts reflection on the effectiveness of the newly developed algorithms and raises questions about the efficiency of Q-Learning in the tasks at hand. Lastly, while Q-Learning may demonstrate a speed advantage (albeit not significantly pronounced), its performance is consistently lower to that of other algorithms. Once more, it is evident that Q-Learning registers the highest occurrence of timeouts, leading to the conclusion that it is ineffective both in terms of performance and time efficiency.

In terms of algorithm selection, the primary observation is that algorithms incorporating the exploitation modification consistently outperform others across all considered tasks. This outcome aligns with expectations since agents consistently make intelligent choices during the exploitation phase. In regard to the exploration phase, the results indicate that for relatively narrow environments, both CQL3* and CQL4* perform exceptionally well. Their behavior closely aligns, especially in confined environments with numerous adversaries. Conversely, in more expansive settings, CQL2* emerges as a strong candidate. However, it is worth remarking that in the most complex tasks, none of the algorithms managed to provide valid solutions within the specified timeout condition. Classic Q-Learning maintains its suitability when dealing with nearly static and narrow environments. Nevertheless, it is crucial to bear in mind that this approach lacks causal knowledge, leading to a significant number of defeats in such scenarios.

All the results have been archived in an open-source repository[1], where the developed code from this thesis work are available for download and utilization.

[1] https://github.com/Giovannibriglia/MasterThesis_CausalQL

Chapter 6

Conclusions

This final chapter encapsulates the conclusions drawn from this master's thesis. It encompasses the primary discoveries made throughout the research, examines their significance, and outlines potential avenues for future advancements.

6.1 Future Developments

This section outlines potential future directions for development, capitalizing on the robust framework established within this work.

First of all, it's clear that reinforcement learning (RL) strongly relies on extensive interactions with the environment. In this work, classic RL algorithms like Q-Learning are employed, operating in a sequential and non-parallelizable manner, which precludes the utilization of GPUs. Despite the relatively moderate computational complexity, the time required for simulations becomes a notable constraint. Consequently, a central way for future development involves the exploration of optimization strategies aimed at speeding up the simulation process. Simultaneously, the facet of causal modeling demands substantial computational resources due to the inherent necessity of establishing correlations between variables. Notably, the instantiation of Bayesian Networks and the computation of do-calculus operations emerge as computationally intensive tasks, incurring both computation and time costs. Furthermore, it is important to note that the utilization of GPU resources remains unattainable also within this context, at least for the present moment. Hence, a promising trajectory for future research lies in the augmentation of the efficiency of these causal inference procedures, thus promoting greater lightness and time-effective analyses. However, it is worth highlighting that achieved results indicate a significant reduction in simulation time when the RL algorithm incorporates the causal modeling component. This stands as one of the prominent achievements of this work, underscoring the potential to leverage causal modeling to mitigate computational bottlenecks.

As a secondary way for potential future development, there is the possibility of increasing the complexity of the tasks under consideration. Additionally, another way worth delving into pertains to the study of dynamic goals within the environment, which could offer captivating insights and outcomes. Instead of having enemies move randomly, it might be quite intriguing to give them more "intelligence". This approach would require the agent to exhibit a high level of cleverness in order to avoid the enemies and successfully reach the goal.

Moreover, an interesting avenue for future development involves constructing the causal model during gameplay, rather than pre-defining it as done in this study. This approach would introduce greater complexity to the framework, but it would closely resemble real-world scenarios. In such cases, the implications related to computational resources and processing times become particularly significant since the do-calculus operation would need to be performed in real-time. A potential strategy to address this challenge might involve updating the Bayesian Networks using a form of bootstrapping technique. This approach could lead to more effective network refinement, benefiting from a larger, but still not too extensive, data set that ensures fast and clear understanding. As a result, the computation time and resources required by the do-calculus operation could potentially be reduced.

Furthermore, a prospective advancement for the causal model could encompass the inclusion of the environment's goal and boundaries within the causal analysis framework (the initial 3x3 grid used for extracting causal tables). The successful integration of this aspect would empower the agent to exhibit behavior that avoids actions leading to the environment's boundaries. Moreover, when the agent approaches the goal, this enriched causal model could facilitate the selection of actions conducive to achieving victory. Additionally, enabling the agent to possess fundamental knowledge of counting and comprehending enemies independently could be another potential avenue for development.

Moreover, a natural extension of this work involves the exploration of alternative temporal-difference algorithms within reinforcement learning, such as SARSA[1]. Similar to the approach undertaken in this work, the integration of causal models can be done. Additionally, a challenging way for further investigation is the comparison between the developed Causal TD-RL algorithms and the DYNA algorithm, given its application in dynamic environments.

Furthermore, the filed of multi-agent systems is particularly intriguing, as it allows for the inclusion of multiple agents in the game. These agents have the potential to collaborate or compete, making it a engaging area of exploration to determine which ensemble strategies can be effectively employed in such scenarios.

Finally, the potential for integrating Deep Reinforcement Learning (DRL) presents itself. However, careful and meticulous design would be requisite for effectively merging DRL with causal knowledge, yielding a fascinating area of exploration.

[1] *SARSA*: State-Action-Reward-State-Action.

6.2 Conclusions

Causal Reinforcement Learning (Causal RL) and, more broadly, Causal Artificial Intelligence stand as highly promising avenues for developing intelligent systems capable of estimating the potential consequences of specific actions. This master's thesis is dedicated to exploring the integration of causal models with the Q-Learning algorithm. A comprehensive elucidation of this fusion is provided, combined by a substantial experimental campaign detailed and analyzed through both textual descriptions and visual plots.

The principal achievements of this research can be encapsulated as the establishment of a framework for embedding causal knowledge within an RL agent, thus providing it with a complete understanding of his surroundings. The results are substantiated by a meticulous comparison of the average reward performance between the traditional Q-Learning algorithm and the newly devised Causal Q-Learning algorithms (varied implementations were examined). These performance comparisons underscore the substantial impact of incorporating causal knowledge in dynamic environments, whereas in static environments like Frozen Lake, the performance remains relatively consistent.

As a general trend, the greater the complexity of the task, the more pronounced the performance differences between Causal and classic Q-Learning become.

Moreover, the generated plots and figures not only enhance explainability but also provide a lucid depiction of the decision-making processes of the RL agent. This thesis also underscores its commitment to designing a trustworthy AI solution.

The implications of this type of research are profound, as elaborated in the "Background" chapter. Just as with general AI systems, the outcomes of this work hold significant potential across numerous domains. Notably, the algorithms examined in this thesis, can be extended to different applications. The unique challenge in adopting these algorithms for different tasks lies in designing the specific environment to which they will be applied.

Future directions have been previously outlined, accompanied by comprehensive explanations for each of them.

```
Persevering in the search for new solutions to increase the quality of
life of all people, loving the environment,
must be one of humanity's main objectives.
```

Bibliography

[1] S. J. Russell, *Artificial intelligence a modern approach*. Pearson Education, Inc., 2010.

[2] K. Krzyk, "Coding deep learning for beginners — medium." [Online]. Available: https://towardsdatascience.com/coding-deep-learning-for-beginners-types-of-machine-learning-b9e651e1ed9d

[3] D. Silver, "Lectures on reinforcement learning," 2015. [Online]. Available: https://www.davidsilver.uk/teaching/

[4] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.

[5] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[6] M. Lippi, S. Mariani, M. Martinelli, and F. Zambonelli, "Individual and collective self-development: Concepts and challenges," in *2022 17th Conference on Computer Science and Intelligence Systems (FedCSIS)*. IEEE, 2022, pp. 15–21.

[7] M. Naeem, S. T. H. Rizvi, and A. Coronato, "A gentle introduction to reinforcement learning and its application in different fields," *IEEE access*, vol. 8, pp. 209 320–209 344, 2020.

[8] J. Pearl, *Causality*. Cambridge university press, 2009.

[9] J. Pearl and D. Mackenzie, "The book of why."

[10] Wikipedia, "Intelligenza artificiale — wikipedia, l'enciclopedia libera." [Online]. Available: http://it.wikipedia.org/w/index.php?title=Intelligenza_artificiale&oldid=133511318

[11] S. Raschka, *Python machine learning*. Packt publishing ltd, 2015.

[12] Wikipedia, "Apprendimento automatico — wikipedia, l'enciclopedia libera." [Online]. Available: http://it.wikipedia.org/w/index.php?title=Apprendimento_automatico&oldid=133328054

[13] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, pp. 279–292, 1992.

[14] L. Buşoniu, R. Babuška, and B. De Schutter, "Multi-agent reinforcement learning: An overview," *Innovations in multi-agent systems and applications-1*, pp. 183–221, 2010.

[15] Wikipedia, "Judea pearl — wikipedia, l'enciclopedia libera." [Online]. Available: http://it.wikipedia.org/w/index.php?title=Judea_Pearl&oldid=132752525

[16] J. Pearl, "The seven tools of causal inference, with reflections on machine learning," *Communications of the ACM*, vol. 62, no. 3, pp. 54–60, 2019.

[17] S. Powell, "The book of why: The new science of cause and effect. pearl, judea, and dana mackenzie. 2018. hachette uk." *Journal of MultiDisciplinary Evaluation*, vol. 14, no. 31, pp. 47–54, 2018.

[18] D. Geiger and J. Pearl, "On the logic of causal models," in *Machine Intelligence and Pattern Recognition*. Elsevier, 1990, vol. 9, pp. 3–14.

[19] J. Pearl, *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan kaufmann, 1988.

[20] M. Lippi, S. Mariani, and F. Zambonelli, "Developing a "sense of agency" in iot systems: Preliminary experiments in a smart home scenario," in *2021 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*. IEEE, 2021, pp. 44–49.

[21] P. Rochat, "Self-perception and action in infancy," *Experimental brain research*, vol. 123, pp. 102–109, 1998.

[22] M. Lippi, M. Martinelli, M. Picone, and F. Zambonelli, "Enabling causality learning in smart factories with hierarchical digital twins," *Computers in Industry*, vol. 148, p. 103892, 2023.

[23] C. Lu, "Introduction to causal reinforcement learning," *Blogpost at causallu.com*, 2018.

[24] I. Dasgupta, J. Wang, S. Chiappa, J. Mitrovic, P. Ortega, D. Raposo, E. Hughes, P. Battaglia, M. Botvinick, and Z. Kurth-Nelson, "Causal reasoning from meta-reinforcement learning," *arXiv preprint arXiv:1901.08162*, 2019.

[25] A. Forney, J. Pearl, and E. Bareinboim, "Counterfactual data-fusion for online reinforcement learners," in *International Conference on Machine Learning*. PMLR, 2017, pp. 1156–1164.

[26] J. Zhang, "Designing optimal dynamic treatment regimes: A causal reinforcement learning approach," in *International Conference on Machine Learning*. PMLR, 2020, pp. 11 012–11 022.

[27] P. Madumal, T. Miller, L. Sonenberg, and F. Vetere, "Explainable reinforcement learning through a causal lens," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 03, 2020, pp. 2493–2500.

[28] S.-G. Choi and S.-B. Cho, "Bayesian networks+ reinforcement learning: Controlling group emotion from sensory stimuli," *Neurocomputing*, vol. 391, pp. 355–364, 2020.

[29] Y. Zhang, Y. Lan, Q. Fang, X. Xu, J. Li, and Y. Zeng, "Efficient reinforcement learning from demonstration via bayesian network-based knowledge extraction," *Computational Intelligence and Neuroscience*, vol. 2021, 2021.

[30] J. Zhang and E. Bareinboim, "Transfer learning in multi-armed bandit: a causal approach," in *Proceedings of the 16th Conference on Autonomous Agents and Multi-Agent Systems*, 2017, pp. 1778–1780.

[31] N. Vlassis, M. Ghavamzadeh, S. Mannor, and P. Poupart, "Bayesian reinforcement learning," *Reinforcement Learning: State-of-the-Art*, pp. 359–386, 2012.

[32] A. M. Molina, I. F. Avelino, E. F. Morales, and L. E. Sucar, "Causal based q-learning," *Research in Computing Science*, vol. 149, pp. 95–104, 2020.

[33] OpenAI, "Frozen lake — openai,." [Online]. Available: https://www.gymlibrary.dev/environments/toy_text/frozen_lake/

[34] R. McFarlane, "A survey of exploration strategies in reinforcement learning," *McGill University*, vol. 3, pp. 17–18, 2018.

Appendix A

Integrazione della Causalità in Q-Learning per il Controllo Adattivo in Ambienti Dinamici

Riassunto in Lingua Italiana

A.1 Sommario

Questa tesi di ricerca è dedicata ad esplorare l'integrazione dei modelli causali negli algoritmi di Reinforcement Learning (RL), lavorando sul concetto noto come Causal Reinforcement Learning. Questo approccio è molto promettente in diversi campi, come la sanità, la robotica, l'ambiente e l'economia. Combinando l'apprendimento guidato dai dati con il ragionamento causale, il Causal RL ha il potenziale per creare sistemi di apprendimento adattivi, robusti e affidabili, in grado di eccellere in scenari complessi, non stazionari e incerti del mondo reale.

In particolare, questa tesi si concentra sull'incorporare una conoscenza causale nell'algoritmo di Q-Learning, una tipologia di algoritmo che fa parte della classe di algoritmi di Temporal Difference (TD) learning, ampiamente usati in Reinforcement Learning. Sebbene Q-Learning possa non essere adatto ad ambienti dinamici a causa della sua struttura, l'introduzione della conoscenza causale ha dimostrato la capacità di migliorarne in maniera importante le prestazioni. La conoscenza causale viene estratta attraverso l'inferenza causale impiegando l'operazione di do-calculus, questa permette di calcolare le probabilità a posteriori delle variabili a seguito di azioni specifiche. L'obiettivo principale di questo lavoro è fornire una struttura completa che integri perfettamente la conoscenza causale nell'algoritmo Q-Learning. Inoltre, vengono considerati i concetti di spiegabilità e affidabilità per garantire l'interpretabilità e la comprensione trasparente del processo decisionale dell'agente.

L'approccio delineato in questa tesi consiste in due componenti primarie. La fase iniziale prevede che l'agente navighi in un ambiente compatto con un nemico solitario, dove entrambi si muovono in maniera casuale. Durante questa fase, l'agente utilizza l'inferenza causale per comprendere i risultati delle sue azioni, che vengono memorizzati per un uso successivo. Nella seconda fase, l'agente opera in ambienti più grandi e complessi, e con diversi nemici, con l'obiettivo generale di raggiungere la posizione obiettivo che non conosce,

tenendo conto che non conosce neppure l'ambiente nella quale è immerso. La valutazione dell'approccio proposto è facilitata dalla creazione di un ambiente di gioco. Viene condotto un confronto approfondito tra le prestazioni dell'algoritmo di Q-Learning classico e quelle del nuovi algoritmi di Q-Learning causale implementati, attraverso un'ampia campagna sperimentale che coinvolge vari scenari, dai più semplici a quelli veramente complessi ed impegnativi. Il confronto dei risultati è approfondito e si concentra su diverse metriche chiave, tra cui la ricompensa media sulla partita, il numero di passi necessari per completare un episodio, il tempo medio necessario per completare la partita, la frequenza delle sconfitte e il verificarsi di condizioni di timeout, ogni metrica è considerata per ciascun algoritmo. I risultati sperimentali evidenziano il significativo gap prestazionale degli agenti dotati di conoscenza causale rispetto a quelli che utilizzano solo il classico algoritmo di Q-Learning. In particolare, il divario di prestazioni aumenta con l'aumentare della complessità del compito.

L'importanza di questo lavoro risiede nell'avanzamento del Causal Reinforcement Learning e dell'Intelligenza Artificiale Causale, questa nuova classe di algoritmi sono una via concreta per incrementare la velocità della ricerche scientifiche e la potenza dei sistemi di Intelligenza Artificiale, questo perché sono in grado di comprendere e valutare ambienti complessi mantenendo la capacità di acquisire continuamente nuovi concetti. I meccanismi di apprendimento e comprensione di questi algoritmi di Causal Reinforcement Learning rispecchiano da vicino i processi cognitivi umani.

A.2 Introduzione

Il principale contributo di questa tesi risiede nell'esplorare la fusione di modelli data-driven, in particolare l'apprendimento per rinforzo, con paradigmi basati su modelli come i modelli causali. L'obiettivo è quello di far progredire la creazione di soluzioni spiegabili, comprendendo tecniche sia per i sistemi a singolo agente sia per i sistemi multi-agente che riguardano la coordinazione, la comunicazione e la generalizzazione. Questa nuova categoria di algoritmi viene definita *Causal Reinforcement Learning*, che promette di rivoluzionare il panorama contemporaneo. Questi algoritmi mostrano una forte adattabilità, sfruttando una combinazione fra un approccio trial-and-error dell'apprendimento per rinforzo e la conoscenza causale generata dai modelli causali per comprendere e prevedere le ripercussioni delle azioni. Gli obiettivi principali di questa tesi di laurea magistrale comprendono la creazione di soluzioni nuove e affidabili per gli scenari di apprendimento per rinforzo. Queste soluzioni si distinguono per la loro robusta spiegabilità, la notevole flessibilità e le ampie capacità di generalizzazione. Inoltre, la tesi si occupa di indagare l'integrazione di modelli causali in scenari a singolo agente, consentendo un processo decisionale efficace in ambienti in evoluzione dinamica. Un aspetto integrale della tesi ruota attorno allo sviluppo di modelli innovativi basati sui dati, in particolare algoritmi di apprendimento per rinforzo. Questi algoritmi si basano sulla conoscenza causale precedentemente acquisita, consentendo di migliorare la coordinazione, l'efficienza e il potenziale decisionale di sistemi a singolo agente (e a più agenti, non affrontati in questa tesi).

Per valutare efficacemente gli algoritmi ideati, è stato definito un ambiente di gioco versatile. La formulazione di questo ambiente è parametrizzata, che permette la trasformazione in una struttura a griglia o a labirinto di dimensioni variabili e con un numero variabile di nemici. L'algoritmo finale comprende due componenti distinte: la prima riguarda l'estrazione di conoscenza causale da un ambiente generico, la quale consente all'agente di discernere le ramificazioni delle sue azioni, comprendendo così i fattori che portano a scenari di game over. La seconda componente prevede che l'agente partecipi attivamente all'ambiente di gioco vero e proprio, dove ha il compito di raggiungere il suo obiettivo, anche se non ne conosce la posizione e, tanto meno, il numero di nemici presenti.

I risultati acquisiti offrono spunti interessanti. Utilizzando le metriche RL convenzionali, risulta evidente che l'agente guidato dalla conoscenza causale ottiene prestazioni superiori rispetto alla sua controparte priva di tale conoscenza. La cosa più significativa è che l'aumento della complessità del compito porta ad un aumento del divario di prestazioni fra le due tipologie di agente costruite (a favore di quello che incorpora la conoscenza causale); questa tendenza indica che, man mano che le sfide diventano più complesse, il possesso della conoscenza causale da parte dell'agente continua a tradursi in risultati sempre migliori.

La tesi è organizzata secondo la seguente struttura:

1. *Background*, questo capitolo per chiarire i concetti chiave pertinenti alla tesi. Approfondisce il campo generale dell'Intelligenza Artificiale (IA), delinea le caratteristiche dell'Apprendimento per Rinforzo (RL), chiarisce come strutturare i problemi di RL, evidenzia gli algoritmi specifici adottati in questa tesi, mostra l'implementazione di modelli causali e valuta i vantaggi e i limiti inerenti a tali modelli.

2. *Related Works*, questo capitolo prende in esame i vari lavori esplorati nel corso di

questa tesi. Mette in evidenza gli aspetti salienti di ciascun lavoro, identificandone i punti di interesse e la rilevanza per la presente ricerca.

3. *Minigame*, in questo capitolo viene documentata la costruzione dell'ambiente di gioco. Il capitolo illustra il processo di costruzione del gioco, dettaglia la metodologia impiegata per l'implementazione del modello causale, cruciale per l'estrazione della conoscenza causale, e illustra li algoritmi di Q-Learning causale costruiti.

4. *Esperimenti e Risultati*, questo capitolo serve a presentare e ad analizzare i risultati ottenuti, offrendo approfondimenti sulle implicazioni e sul significato dei risultati.

5. *Conclusioni*, il capitolo conclusivo presenta una sintesi completa del lavoro svolto, evidenziando i punti chiave e i potenziali percorsi di ricerca futuri.

A.3 Minigame

In questo capitolo viene illustrata la configurazione dell'ambiente sperimentale utilizzato in questa tesi. Per facilitare una valutazione rigorosa, è stata fatta un'implementazione personalizzata per garantire un alto grado di complessità ai problemi considerati. L'intero lavoro è stato svolto utilizzando il linguaggio di programmazione `Python`.

A.3.1 Implementazione del Gioco

L'ambiente simula un agente che naviga in un ambiente che non conosce alla ricerca di un obiettivo del quale non conosce la posizione, in aggiunta sono presenti dei nemici.

Inizialmente, l'attenzione si è concentrata su un classico problema di Reinforcement Learning, nello specifico l'ambiente Frozen Lake di OpenAI [33]. Questo ambiente comprende una griglia 4x4 contenente quattro tipi di aree: Inizio (S), Congelato (F), Buco (H) e Obiettivo (G), come mostra la figura 4.1.

L'agente naviga nella griglia con l'obiettivo primario di raggiungere l'obiettivo nel modo più efficiente. Quando raggiunge l'obiettivo con successo ha una ricompensa di valore +1, mentre se arriva su un terreno ghiacciato o su una buca riceve una ricompensa pari a 0. È importante notare che non c'è una ricompensa negativa "punitiva" attribuita alla caduta in una buca. Poiché l'ambiente è statico, le posizioni delle buche e dell'obiettivo sono stazionarie.

Per gli scenari considerati in questa tesi, in modo da intensificarne la complessità, sono stati introdotti nell'ambiente dei nemici dinamici. Sono state considerate due categorie principali di sfide: quelle "più semplici", che riguardano scenari con 1 o 5 nemici in ambienti a griglia di dimensioni 10, 20 e 50; quelle "più complesse" che comprendono scenari con 10 e 20 nemici in ambienti a griglia di dimensioni 10, 20, 50 e 100.

La complessità considerata è ulteriormente amplificata dall'inserimento di una ricompensa negativa di -1 per l'agente in caso di sconfitta. In particolare, negli ambienti più grandi (per lo più di dimensione 100, ma anche di dimensione 50), il problema diventa particolarmente interessante, poiché rientra nella categoria *sparse reward problem*, in cui l'ambiente fornisce raramente un segnale di ricompensa utile. Questa complessità si riflette anche nel tempo di calcolo necessario per completare l'intero compito. Vale la pena

menzionare che gli scenari in cui il numero di nemici è pari o superiore al numero di righe/colonne sono particolarmente impegnativi.

Analogamente a quanto accade nell'ambiente Frozen Lake, anche il contesto attuale prevede che l'ambiente venga resettato quando l'agente raggiunge la vittoria o la sconfitta, questo comporta il riposizionamento dell'agente e dei nemici nelle loro posizioni iniziali all'interno dell'ambiente. Inoltre, come nel caso classico, l'agente e i nemici hanno a disposizione cinque azioni distinte: Stop (0), Destra (1), Sinistra (2), Su (3) e Giù (4).

In questo nuovo gioco l'agente può perdere la partita in due scenari distinti: in primo luogo, se si scontra con un nemico e, in secondo luogo, se un nemico occupa la stessa posizione dell'agente e quest'ultimo sceglie di non muoversi.

Procedendo ulteriormente nell'aumento di complessità, è stato implementato anche un ambiente di tipo labirinto, che incorpora "muri" generati in modo casuale. I compiti analizzati in questo ambiente sono identici a quelli precedentemente descritti per l'ambiente a griglia.

È importante osservare che sia nell'ambiente a griglia che in quello a labirinto, le posizioni iniziali dell'agente, dell'obiettivo, dei nemici e, nel caso dei labirinti, delle pareti, sono assegnate in modo casuale per garantire che nessuno di essi abbia la stessa posizione iniziale di altri. Questi esperimenti introducono progressivamente una maggiore complessità nel richiedere all'agente di comprendere la dinamica dell'ambiente e le conseguenze delle sue azioni. L'agente, all'iniziare della partita, ha sempre la Q-Table azzerata.

È stata sviluppata un'interfaccia grafica per visualizzare l'ambiente di gioco, consentendo una rappresentazione tangibile di come l'agente si comporta durante la partita. Le Figure 4.2 e 4.3 mostrano le schermate dell'interfaccia grafica (GUI) progettata per gli ambienti a griglia e a labirinto in studio. Questa GUI fornisce una rappresentazione visiva dell'andamento del gioco che include varie metriche in tempo reale, tra cui lo stato attuale del gioco, il tempo trascorso, il numero di partite giocate, l'episodio in corso, l'algoritmo attivo e il conteggio delle sconfitte. Come si può immaginare, Mario Bros è l'agente, Bowser(s) il(i) nemico(i) e il tubo verde l'obiettivo, richiamando il famoso gioco.

A.3.2 Causalità

Il compito dell'agente in questo step consiste nel comprendere le dinamiche dell'ambiente in cui si trova; l'agente affronta questa sfida senza conoscere a priori le caratteristiche dell'ambiente.

Per raggiungere la comprensione, l'agente deve costruire il modello causale che governa l'ambiente, in particolare le conseguenze causali derivanti dalle sue azioni. L'implementazione del modello causale è facilitata dall'utilizzo della libreria `CausalNex`. Si tratta di una libreria Python che utilizza le reti bayesiane per combinare l'apprendimento automatico e l'esperienza in un dominio per il ragionamento causale. È possibile utilizzare CausalNex per scoprire relazioni strutturali nei dati, imparare distribuzioni complesse e osservare l'effetto di potenziali interventi.

Questa fase si è concentrata sull'estrazione di relazioni causali all'interno di un contesto specifico. L'ambiente scelto è una griglia 3x3, con un unico nemico e nessun obiettivo. La scelta di questa dimensione della griglia ha un significato: la configurazione 3x3 consente numerose interazioni tra l'agente e il nemico. L'assenza di un obiettivo in questo ambiente è in linea con gli obiettivi primari, che si concentrano sulla comprensione del modo in cui le

azioni dell'agente ne influenzano il movimento e sulla comprensione dei fattori che portano allo scenario di "game over". Il dataframe di input è di 10k righe, che tengono traccia dei movimenti sia dell'agente che del nemico; le azioni intraprese dagli agenti e dai nemici sono completamente casuali (in questa sezione).

Per facilitare una migliore comprensione all'interno del modello causale, sono state definite diverse variabili chiave; queste variabili sono state progettate per fornire una rappresentazione completa delle dinamiche di interazione agente-nemico nell'ambiente: *State, Action, Var, Game Over, Alive, Enemies Nearby, Enemies Attached and Contact*. La spiegazione dettagliata di queste è fornita nel capitolo 4.

Per ottenere le migliori prestazioni dalla rete bayesiana utilizzata nella fase successiva, il dataframe viene elaborato convertendo le variabili in formato binario. Questa trasformazione aiuta a definire molte più variabili in modo tale che la rete possa discernere con precisione quali variabili influenzano i cambiamenti di altre. L'algoritmo 2 fornisce il diagramma a blocchi riassuntivo che illustra il processo di implementazione di questa sezione.

Il risultato di questo processo è una tabella causale, che ha sulle colonne le conseguenze a seguito di un determinato evento riportato sulle righe. Un estratto è riportato nella Tabella 4.1, nel quale è possibile identificare le due categorie di modello interessanti per la tesi: una parte relativa al movimento dell'agente e una parte relativa a come accade il game over.

A.3.3 Reinforcement Learning

L'algoritmo classico di Q-Learning incontra dei limiti in ambienti dinamici, principalmente a causa del suo approccio all'aggiornamento della Q-Table. Ad esempio, considerando l'equazione 2.3, negli scenari in cui l'agente perde, la Q-Table viene aggiornata al punto di fallimento (stato S). Tuttavia, nei casi che coinvolgono nemici dinamici, questo aggiornamento da solo non aiuta efficacemente l'agente ad evitare i nemici nei passi successivi.

Per ovviare significativamente questo problema, l'agente ha bisogno di comprendere le azioni e le situazioni che lo portano al game over. Questa comprensione può essere ottenuta tramite l'utilizzo della tabella causale derivata dal segmento di causalità precedente. Sfruttando le probabilità a posteriori, l'agente può determinare le conseguenze delle azioni scelte e le strategie per evitare esiti sfavorevoli.

Per determinare la politica di selezione delle azioni dell'agente, è stato utilizzato il metodo ϵ-greedy: questa strategia permette un equilibrio tra esplorazione e sfruttamento; in questa tesi è stata definita in modo che segua un andamento esponenziale decrescente: in particolare, all'aumentare del numero di episodi di gioco completati, la probabilità di esplorare diminuisce gradualmente, favorendo lo sfruttamento; come riportato in Figura 4.6.

Al fine di identificare l'approccio più efficace per l'implementazione del Causal Q-Learning, è stata condotta un'esplorazione completa di quattro (otto) varianti distinte. La descrizione di queste varianti è dettagliata nell'Algoritmo 3, mentre la spiegazione di ogni componente è riportata negli Algoritmi 4, 5, 6 e 7. Queste implementazioni preannunciano l'introduzione di una nuova classe di algoritmi, con framework basato su Q-Learning, in grado di comprendere l'ambiente circostante ed adatta ad ambienti dinamici. Inoltre, sono stati formulati due tipologie distinti di approcci causali: uno incentrato esclusivamente

sulla modifica dell'aspetto esplorativo e l'altro che prevede la modifica sia alla parte esplorativa che a quella di sfruttamento. Queste metodologie sono state progettate con l'intento di comprendere quali siano le più efficaci e in quali situazioni. Gli algoritmi in cui vengono modificate sia la fase di esplorazione che quella di sfruttamento sono contrassegnati dall'asterisco ("*").

A.4 Esperimenti e Risultati

In questa sezione viene presentato il setup sperimentale utilizzato, le metriche considerate per l'analisi e i risultati; come descritto in precedenza, sono stati presi in considerazione diversi scenari, dai più semplici ai più impegnativi.

A.4.1 Setup sperimentale

Innanzitutto, viene chiarita la configurazione utilizzata per valutare le prestazioni dei casi studio considerati. Seguendo una metodologia simile a quella descritta in [4], è stata condotta una serie di esperimenti che coprono un'ampia gamma di configurazioni. In particolare, un gioco viene generato in modo casuale, tenendo conto di variabili quali il tipo di ambiente, le sue dimensioni e il numero di nemici presenti. Successivamente, l'agente all'interno del gioco viene guidato con ogni algoritmo costruito (stesso gioco per ogni algoritmo, uno alla volta) e le sue prestazioni vengono valutate con specifiche metriche. Questo processo iterativo viene replicato per dieci volte, e ogni iterazione comporta la generazione di un gioco distinto, sempre costruito casualmente.

Inoltre, un ulteriore aspetto di valutazione riguarda il comportamento dei nemici, per il quale sono stati sviluppate due casistiche. Nel primo scenario, di natura più semplice, ogni nemico segue una sequenza predeterminata di azioni casuali (una sequenza diversa per ogni nemico) e, al reset, riprendono dall'inizio della sequenza. Il secondo scenario è più impegnativo, poiché i nemici compiono azioni casuali ad ogni passo temporale.

Data la varietà degli ambienti e dei compiti analizzati, sono stati introdotti due distinti criteri di *timeout* per valutare l'efficacia delle soluzioni proposte anche in termini di efficienza computazionale. Questi criteri di timeout sono stati applicati individualmente a ciascun gioco e approccio. In particolare, è stato fissato un timeout di 5 minuti per i compiti più "facili", mentre un timeout di 10 minuti per i compiti più complessi. Questi valori sono stati scelti dopo aver valutato il tempo di calcolo richiesto per ogni problema in esame; negli scenari in cui l'ambiente è relativamente limitato, queste condizioni sono meno impegnative, ma quando l'ambiente si allarga, diventano abbastanza stringenti. Nel caso in cui un algoritmo superi il limite di tempo specificato durante una partita, tutte le partite associate a quell'algoritmo sono considerate non valide e l'etichetta *"-"* è inserita nelle tabelle dei risultati di ricompensa media.

A.4.2 Metriche

Le metriche di valutazione scelte comprendono le misure tipiche degli algoritmi di RL: il *numero di passi necessari per completare un episodio* e la *ricompensa media della partita*. Quest'ultima metrica riveste la massima importanza per i problemi di RL, come chiarito

nel Capitolo 2; nonostante ciò, anche la velocità con cui si raggiunge la convergenza del numero di passaggi per episodio (se si verifica) è di notevole importanza.

Per riassumere i risultati, si utilizzano tabelle, grafici e diagrammi a barre. I grafici presentati, che mostrano la ricompensa media e il numero di passaggi richiesti per episodio, sono stati generati utilizzando un filtro gaussiano del primo ordine. Questo approccio è stato utilizzato per creare grafici più uniformi, migliorando la chiarezza dei dati presentati. In sostanza, l'intera metodologia può essere paragonata a una simulazione Monte Carlo, in quanto prevede il calcolo della media delle metriche di 10 simulazioni per valutare in modo completo i risultati.

Inoltre, per valutare il *tempo di esecuzione delle singole simulazioni* di ciascun approccio, in termini di tempo di elaborazione della CPU, vengono fornite rappresentazioni grafiche che illustrano la durata media in minuti di ogni algoritmo, offrendo un confronto diretto. Il computer utilizzato per queste simulazioni è dotato di un processore i7 e di 16 GB di RAM.

In aggiunta, per valutare l'insorgenza di *timeout* per ciascun algoritmo durante le simulazioni, vengono presentate rappresentazioni visive che ne illustrano la frequenza. Come prospettiva alternativa sull'efficacia degli algoritmi proposti, sono presenti grafici a barre che mostrano il *numero medio di sconfitte in una singola partita per ogni algoritmo.*

Infine, uno strumento importante per comprendere i meccanismi all'interno degli algoritmi progettati è la *Q-Table*. Questa risorsa aiuta a comprendere come il segnale di ricompensa sia distribuito nell'ambiente, aiutando la comprensione di come ogni algoritmo lavora. Da notare che in ambienti molto grandi questa tabella risulta difficile da interpretare.

A.4.3 Risultati

Si riportano due casistiche interessanti tra gli scenari più complessi considerati.

Griglia

Environment		Average Reward in Algorithms								
Grid size	**Enemies**	**QL**	**CQL1**	**CQL2**	**CQL3**	**CQL4**	**CQL1***	**CQL2***	**CQL3***	**CQL4***
10x10	10	-3.637	-3.444	-3.625	-1.576	-1.429	-	-0.749	0.961	**0.964**
20x20	10	-1.747	-1.606	-2.121	-1.337	-1.101	0.64	0.666	0.977	**0.978**
50x50	10	-	-	-	-	-	-	0.685	**0.701**	0.668
100x100	10	-	-	-	-	-	-	**0.403**	-	-
10x10	20	-6.923	-6.665	-6.861	-3.876	-4.317	-2.472	-2.619	0.915	**0.918**
20x20	20	-8.7	-8.582	-9.684	-6.39	-5.612	-	-	**0.955**	0.949
50x50	20	-	-	-	-	-	-	0.43	**0.688**	0.662
100x100	20	-	-	-	-	-	-	**-0.503**	-	-
50x50	50	-	-	-	-	-	-	-	**0.153**	-
100x100	50	-	-	-	-	-	-	-	-	-

Table A.1: *Comparison of average reward performance: Classic Q-Learning (QL) vs. Causal Q-Learning Variants with exploration modification (CQL1 to CQL4) vs. Causal Q-Learning Variants with modifications to both exploration and exploitation phases (CQL1* to CQL4*). Evaluation in more challenging grid environments where enemies take random actions. The symbol "-" indicates that a timeout occurred.*

La tabella mostra le ricompense medie ottenute; come si può notare, i task considerati sono di notevole complessità, in particolare nel task griglia 100x100 con 50 nemici nessun algoritmo ha rispettato il limite di tempo di calcolo specificato; inoltre, in questo caso, il numero di timeout per ciascun algoritmo è più pronunciato, come indicato nella Figura A.1. In particolare, CQL2* presenta il minor numero di timeout, che in questo caso sono quattro; si sottolinea la sua robustezza, soprattutto quando il task è relativo ad un ambiente ampio. Mentre, nello scenario di una griglia 50x50 con 50 nemici, CQL3* emerge come l'unico algoritmo in grado di ottenere prestazioni soddisfacenti.

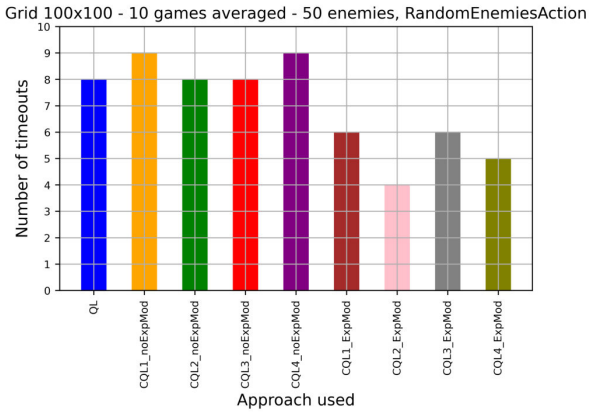


Figure A.1: *Numero di timeout per ciascun algoritmo; ambiente a griglia 100x100, 50 nemici che che compiono azioni casuali. "_noExpMod" indica nessun cambiamento nella fase di sfruttamento, mentre "_ExpMod" indica la presenza di tale cambiamento, equivalente al simbolo "*".*

In generale, è importante notare che le ricompense medie in questi casi sono notevolmente basse, testimoniando la complessità dei task considerati.

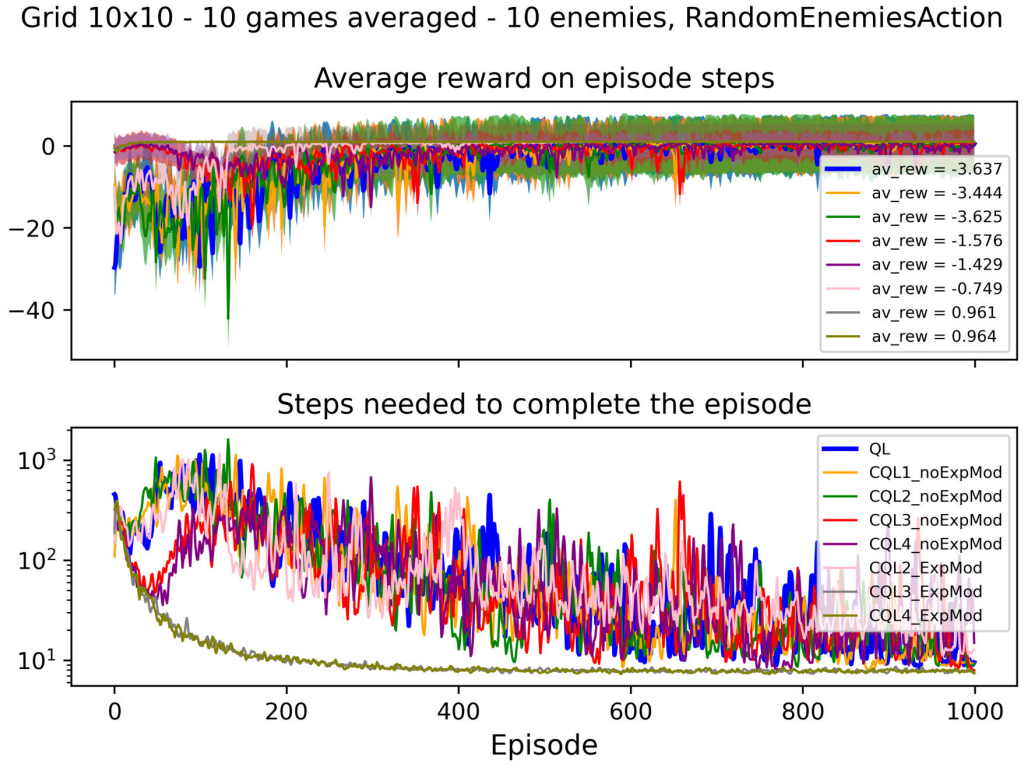


Figure A.2: *Confronto delle prestazioni di ricompensa media e dei passi necessari per completare l'episodio; ambiente a griglia 10x10, 10 nemici che compiono azioni casuali. "_noExpMod" indica nessun cambiamento nella fase di sfruttamento, mentre "_ExpMod" indica la presenza di tale cambiamento, equivalente al simbolo "*".*

Come caso di studio è stata scelta la griglia 10x10 con 10 nemici per sottolineare l'efficacia di CQL3* e CQL4*. Come illustrato nella Figura A.2, questi due algoritmi si distinguono come gli unici in grado di ottenere una ricompensa media positiva e, inoltre, le loro ricompense medie sono quasi ottimali. Altri approcci si dimostrano inadatti a questo ambiente difficile, caratterizzato da dimensioni ridotte e da un'alta densità di nemici. Gli andamenti esibiti da questi due algoritmi sono costantemente regolari e il numero di sconfitte, rispetto agli approcci alternativi, è minimo.

Le Figure A.3 e A.4 mostrano rispettivamente il numero di sconfitte medio di ogni algoritmo e il tempo medio di calcolo necessario per risolvere la configurazione presa in esame. Questa osservazione vale anche per lo scenario con 20 nemici nella griglia 10x10, dove CQL4* raggiunge un'impressionante ricompensa media di 0.918, dimostrando prestazioni eccellenti.

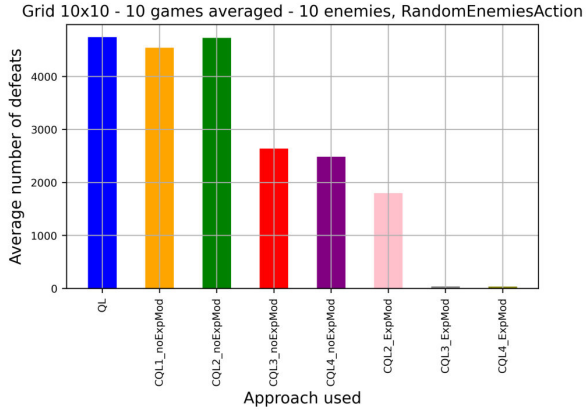


Figure A.3: *Numero di sconfitte medio in una partita per ogni algoritmo; ambiente a griglia 10x10, 10 nemici che compiono azioni casuali. "_noExpMod" indica nessun cambiamento nella fase di sfruttamento, mentre "_ExpMod" indica la presenza di tale cambiamento, equivalente al simbolo "*".*

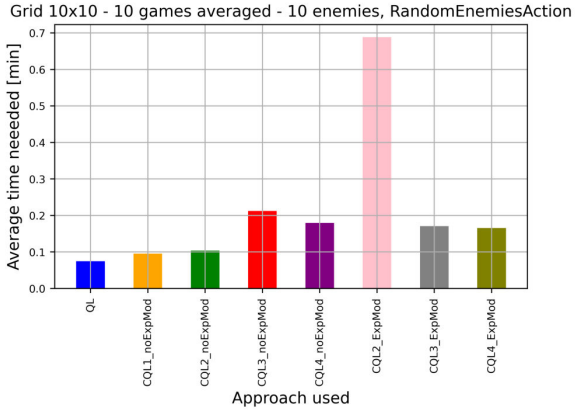


Figure A.4: *Tempo medio di completamento del gioco per ogni algoritmo; ambiente a griglia 10x10, 10 nemici che compiono azioni casuali. "_noExpMod" indica nessun cambiamento nella fase di sfruttamento, mentre "_ExpMod" indica la presenza di tale cambiamento, equivalente al simbolo "*".*

Labirinto

Environment		**Average Reward in Algorithms**								
Maze size	**Enemies**	**QL**	**CQL1**	**CQL2**	**CQL3**	**CQL4**	**CQL1***	**CQL2***	**CQL3***	**CQL4***
10x10	10	-7.216	-6.486	-	-9.453	-	-	-	**0.851**	0.831
20x20	10	-2.532	-2.826	-4.406	-1.21	-1.168	0.276	0.518	**0.956**	0.952
50x50	10	-	-	-	-	-	-0.111	-	0.752	0.524
100x100	10	-	-	-	-	-	-	**0.359**	-	-
10x10	20	-15.433	-15.412	-	-	-	-	-	**0.568**	0.56
20x20	20	-16.231	-16.574	-	-	-	-	-	**0.871**	0.858
50x50	20	-	-	-	-	-	-	0.214	0.379	**0.471**
100x100	20	-	-	-	-	-	-	-	-	-
50x50	50	-	-	-	-	-	-	-	**0.31**	-
100x100	50	-	-	-	-	-	-	-	-	-

Table A.2: *Comparison of average reward performance: Classic Q-Learning (QL) vs. Causal Q-Learning Variants with exploration modification (CQL1 to CQL4) vs. Causal Q-Learning Variants with modifications to both exploration and exploitation phases (CQL1* to CQL4*). Evaluation in more challenging maze environments where enemies take random actions. The symbol "-" indicates that a timeout occurred.*

La tabella illustra che sia l'approccio classico di Q-Learning che quello di Causal Q-Learning senza sfruttamento modificato non sono adatti a questi compiti altamente impegnativi. Questi compiti prevedono che l'agente navighi attraverso ambienti labirintici di varie dimensioni con un numero significativo di nemici. In particolare, le prestazioni ottenute da Q-Learning classico e da Q-Learning causale senza sfruttamento modificato sono notevolmente simili: quando riescono a rispettare la condizione di timeout specificata, le loro ricompense medie si allineano strettamente, mentre quando Q-Learning non riesce a farlo, anche gli algoritmi di Causal Q-Learning senza sfruttamento modificato falliscono.

In generale, più alto è il numero di sconfitte nelle fasi iniziali del gioco, più complesso diventa scoprire il percorso ottimale per raggiungere la posizione obiettivo. Ad esempio, se esaminiamo CQL3 e CQL4, durante la fase di esplorazione, il rischio di perdere è minimo. Tuttavia, quando si passa alla fase di sfruttamento utilizzando l'approccio ϵ-greedy, anche per un breve periodo, la Q-Table si deteriora rapidamente, soprattutto se si considera la presenza significativa di avversari nell'ambiente. Un comportamento simile si incontra con Q-Learning classico, che presenta una situazione anche peggiore a causa dell'assenza della modifica alla fase di esplorazione.

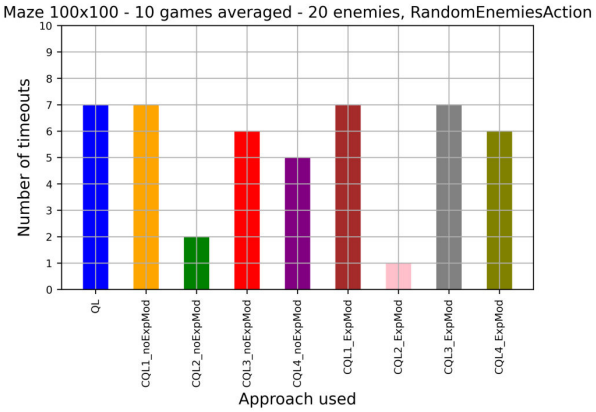


Figure A.5: *Numero di timeout per ciascun algoritmo; ambiente labirinto 100x100, 20 nemici che compiono azioni casuali. "_noExpMod" indica nessun cambiamento nella fase di sfruttamento, mentre "_ExpMod" indica la presenza di tale cambiamento, equivalente al simbolo "*".*

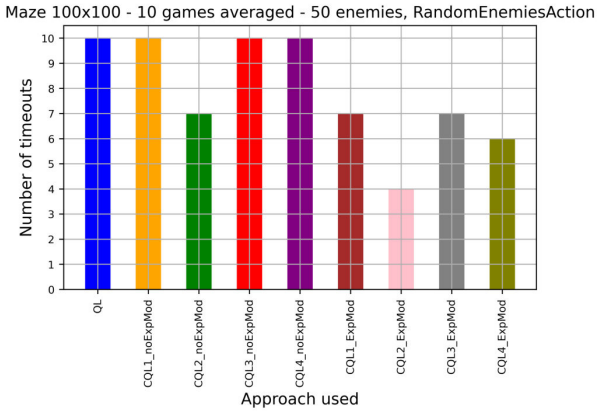


Figure A.6: *Numero di timeout per ciascun algoritmo; ambiente labirinto 100x100, 50 nemici che compiono azioni casuali. "_noExpMod" indica nessun cambiamento nella fase di sfruttamento, mentre "_ExpMod" indica la presenza di tale cambiamento, equivalente al simbolo "*".*

D'altra parte, gli algoritmi di Q-Learning causale con sfruttamento modificato danno risultati impressionanti. Le uniche eccezioni sono i labirinti 100x100 con 20 e 50 nemici, in cui hanno faticato a soddisfare la condizione di timeout prescritta. Le Figure A.5 e A.6 forniscono indicazioni sul numero di timeout di ciascun algoritmo. Come osservato nelle istanze precedenti, l'algoritmo CQL2* è base promettente per lo sviluppo di un algoritmo più potente adatto ad ambienti molto grandi; questo è confermato anche dal numero minimo di timeout.

Per questa categoria di esperimenti, sono stati considerati due casi di studio specifici che coinvolgono task a labirinto simili, ciascuno con 20 nemici, di dimensioni 10x10 e 20x20. È sorprendente notare che i risultati delle prestazioni in entrambi i casi sono stati molto simili. Tra gli algoritmi valutati, solo Q-Learning classico, CQL1, CQL3* e CQL4* sono riusciti a soddisfare la condizione di timeout in entrambi gli scenari.

Esaminando il numero di sconfitte subite nei casi considerati, come mostrato nelle Figure A.7 e A.8, la situazione sembra essere equivalente, anche considerando solo i grafici visivamente. Gli ultimi due algoritmi registrano un numero di perdite trascurabile, mentre i primi due mostrano all'incirca lo stesso numero di sconfitte, circa 17500.

Questi risultati sono confermati dall'esame delle Figure A.9 e A.10, dove sono riportate

le ricompense medie e i passi necessari per completare gli episodi. I primi due algoritmi non mostrano un andamento regolare, non sono in grado di evitare i nemici e hanno prestazioni inaccettabili. Al contrario, gli ultimi due algoritmi mostrano un comportamento coerente con la tesi generale: un andamento regolare che converge intorno al 300° episodio con risultati eccellenti data la complessità del compito.

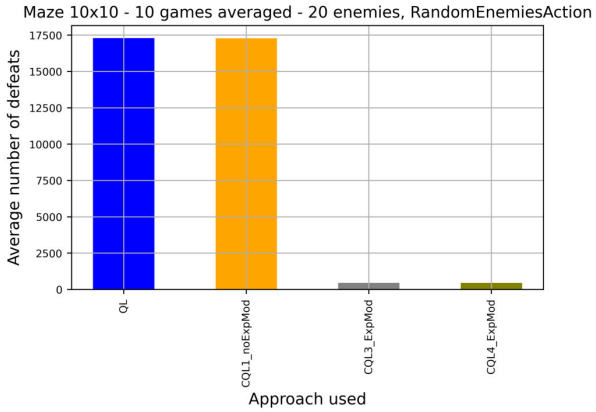


Figure A.7: *Numero di sconfitte medio in una partita per ogni algoritmo; ambiente labirinto 10x10, 20 nemici che compiono azioni casuali. "_noExpMod" indica nessun cambiamento nella fase di sfruttamento, mentre "_ExpMod" indica la presenza di tale cambiamento, equivalente al simbolo "*".*

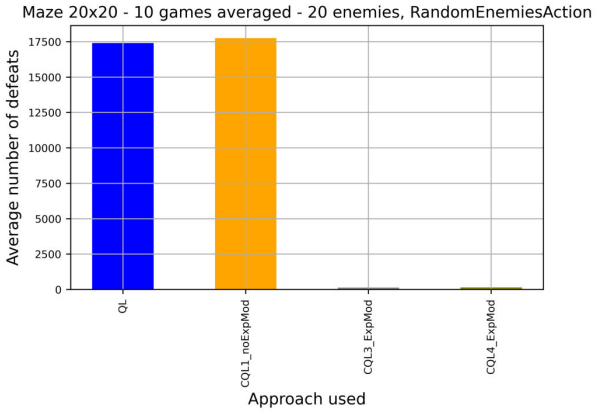


Figure A.8: *Numero di sconfitte medio in una partita per ogni algoritmo; ambiente labirinto 20x20, 20 nemici che compiono azioni casuali. "_noExpMod" indica nessun cambiamento nella fase di sfruttamento, mentre "_ExpMod" indica la presenza di tale cambiamento, equivalente al simbolo "*".*

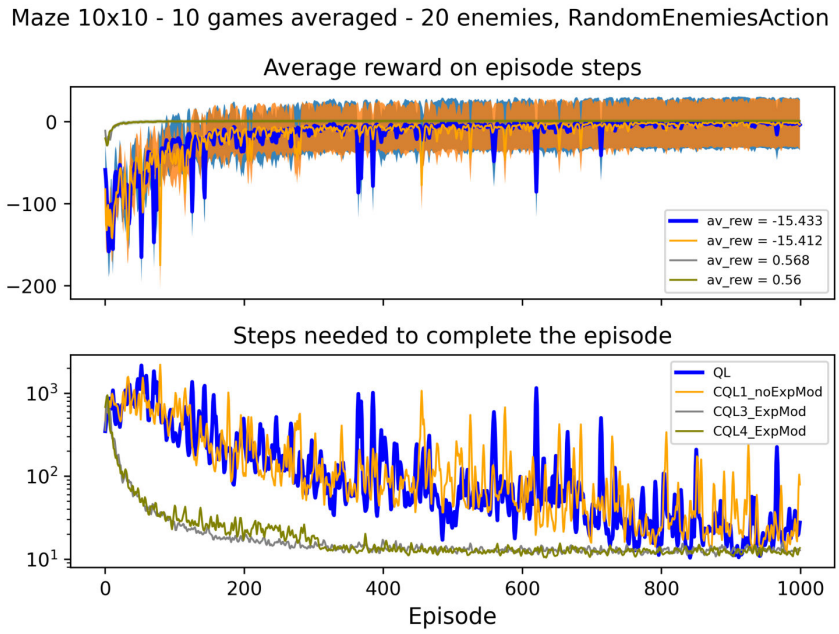


Figure A.9: *Confronto delle prestazioni della ricompensa media e delle passi necessari per completare l'episodio; ambiente labirinto 10x10, 20 nemici che compiono azioni casuali. "_noExpMod" indica nessun cambiamento nella fase di sfruttamento, mentre "_ExpMod" indica la presenza di tale cambiamento, equivalente al simbolo "*".*

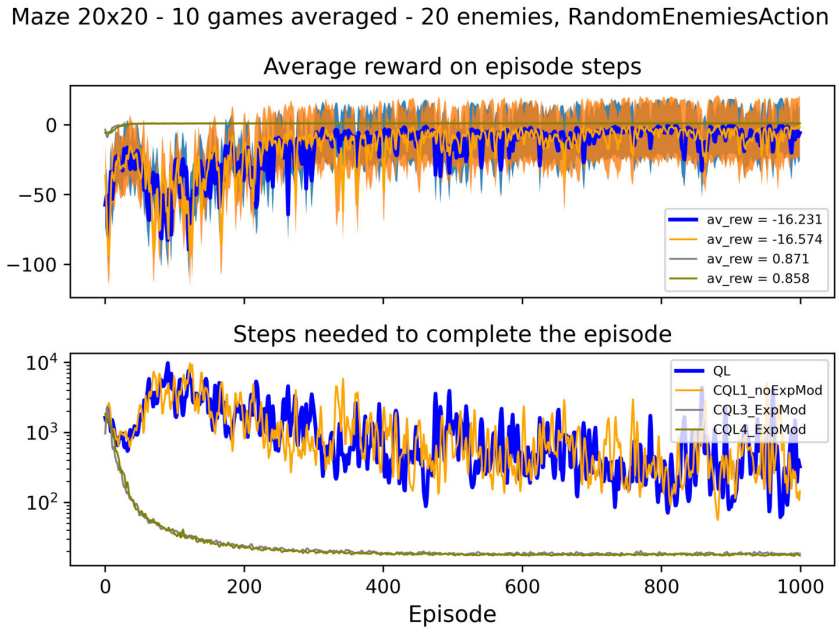


Figure A.10: *Confronto delle prestazioni della ricompensa media e delle passi necessari per completare l'episodio; ambiente labirinto 20x20, 20 nemici che compiono azioni casuali. "_noExpMod" indica nessun cambiamento nella fase di sfruttamento, mentre "_ExpMod" indica la presenza di tale cambiamento, equivalente al simbolo "*".*

A.4.4 Concetti Estrapolati

L'obiettivo di questa sezione è quello di offrire un riassunto sintetico dei risultati e una panoramica finale completa, aggiungendo ulteriori approfondimenti.

Innanzitutto, l'introduzione di una condizione di timeout è uno strumento necessario per la valutazione degli algoritmi di RL. Ciò è dovuto all'approccio di lavoro intrinseco di questi algoritmi di intelligenza artificiale: interagire con l'ambiente per una durata prolungata che, in alcune applicazioni del mondo reale, può diventare inaccettabile. Questa condizione, adattata alla complessità del compito, ha fornito preziose indicazioni sui limiti degli algoritmi in esame. In particolare, anche se gli ambienti iniziali considerati erano relativamente gestibili (condizione di timeout di 5 minuti), è stato registrato un numero non trascurabile di timeout, soprattutto per l'approccio classico di Q-Learning e anche per gli algoritmi di Q-Learning causale privi della parte di sfruttamento modificata. Quando l'analisi viene estesa ad ambienti più complessi (condizione di timeout di 10 minuti), si scopre che alcuni compiti si sono rivelati irrisolvibili, anche se in numero limitato (in particolare dove l'ambiente è molto impegnativo). Comunque, molti compiti sono stati risolti con successo, principalmente dagli algoritmi di Q-Learning causale con anche la fase di sfruttamento modificata. La regolazione in modo incrementale ed iterativo del limite di tempo di calcolo è fondamentale per determinare il punto in cui la condizione di timeout porta a un cambiamento significativo dei risultati.

Sempre in termini di tempo, i grafici a barre presentati rivelano vari aspetti relativi al tempo di calcolo richiesto. È evidente che Q-Learning mostra in molti casi l'esecuzione più veloce, tuttavia tre aspetti degni di nota meritano attenzione: in primo luogo, tutti gli algoritmi di Q-Learning causale comportano la consultazione del modello causale, che è integrato nella relativa tabella. Di conseguenza, in molti casi, questo fattore spiega la differenza di tempo di calcolo necessario. In secondo luogo, come evidenziato dai grafici relativi agli step necessari per completare un episodio, Q-Learning mostra spesso un comportamento frastagliato, con un numero di passi necessari notevolmente superiore rispetto agli altri. Questa osservazione fa riflettere in positivo sull'efficacia degli algoritmi sviluppati e solleva dubbi sull'efficienza di Q-Learning nei compiti da svolgere. Infine, sebbene Q-Learning dimostri un vantaggio in termini di velocità (anche se non significativamente pronunciato, quando riesce), le sue prestazioni sono costantemente inferiori a quelle degli altri algoritmi. Ancora, è evidente che Q-Learning registra il maggior numero di timeout, portando alla conclusione che è inefficace sia in termini di prestazioni che di efficienza temporale.

In termini di selezione degli algoritmi, l'osservazione principale è che gli algoritmi che incorporano la modifica nella fase di sfruttamento superano costantemente gli altri in tutti i compiti considerati. Questo risultato è in linea con le aspettative, poiché gli agenti compiono sempre scelte intelligenti durante la fase di sfruttamento. Per quanto riguarda la fase di esplorazione, i risultati indicano che per ambienti relativamente ristretti, sia CQL3* che CQL4* ottengono risultati eccezionali. Il loro comportamento è strettamente allineato in ambienti ristretti con numerosi avversari. Al contrario, in ambienti più ampi, CQL2* emerge come un candidato principale; tuttavia, è essenziale notare che nei compiti più complessi, nessuno degli algoritmi è riuscito a fornire soluzioni valide entro la condizione di timeout specificata. Q-Learning classico mantiene la sua idoneità quando si tratta di ambienti praticamente statici e ristretti, sempre considerando che questo approccio manca

della conoscenza causale e, di conseguenza, presenterà sempre un numero significativo di sconfitte.

Tutti i risultati sono stati archiviati in una repository open-source[1], nella quale si trova anche il codice sviluppato.

A.5 Conclusioni

L'apprendimento per rinforzo causale (Causal Reinforcement Learning, RL) e, più in generale, l'intelligenza artificiale causale sono strade molto promettenti per lo sviluppo di sistemi intelligenti in grado di stimare le potenziali conseguenze di azioni specifiche e, in seguito, scegliere le azioni migliori per ottenere l'obiettivo. Questa tesi è dedicata all'esplorazione dell'integrazione dei modelli causali con l'algoritmo Q-Learning. Viene fornita una spiegazione completa di questa fusione, abbinata a una consistente campagna sperimentale dettagliata e analizzata attraverso descrizioni testuali e grafici.

I principali risultati di questa ricerca possono essere racchiusi nella creazione di un framework per incorporare la conoscenza causale all'interno di un agente RL, fornendogli così una comprensione completa dell'ambiente circostante. I risultati sono avvalorati da un confronto meticoloso delle prestazioni medie di ricompensa tra il tradizionale algoritmo di Q-Learning e gli algoritmi di Causal Q-Learning di nuova concezione (sono state esaminate diverse implementazioni). Questo confronto sottolinea l'impatto sostanziale dell'incorporazione della conoscenza causale all'interno dell'agente in ambienti dinamici; mentre, in ambienti statici come Frozen Lake, le prestazioni rimangono relativamente equivalenti.

Come tendenza generale, maggiore è la complessità del compito, più pronunciate diventano le differenze di prestazioni tra Causal Q-Learning e Q-Learning classico (a favore di quello causale).

Inoltre, i grafici e le figure generate non solo migliorano la comprensione dei risultati, ma forniscono anche una rappresentazione chiara dei processi decisionali dell'agente. Si sottolinea l'impegno, tramite questa tesi, di progettare una soluzione di IA affidabile.

Le implicazioni di questo tipo di ricerca sono profonde, come spiegato nel capitolo 2. Come per i sistemi di IA in generale, i risultati di questo lavoro hanno un potenziale significativo in numerosi domini; in particolare, gli algoritmi esaminati in questa tesi possono essere estesi a diverse applicazioni. L'unica implicazione nell'adottare questi algoritmi per compiti diversi sta nel progettare l'ambiente specifico a cui saranno applicati.

Le direzioni future sono state delineate nel capitolo 6 e accompagnate da spiegazioni esaustive.

`Perseverare nella ricerca di nuove soluzioni per migliorare la qualità della vita di tutte le persone, volendo bene all'ambiente, deve essere uno dei principali obiettivi dell'umanità.`

[1] `https://github.com/Giovannibriglia/MasterThesis_CausalQL`

A.6 Ringraziamenti

Desidero innanzitutto ringraziare il professor Lippi, relatore di questa tesi, che mi ha dato l'opportunità di approfondire un argomento di mio grande interesse, il Reinforcement Learning, facendomi anche scoprire un'altra corrente di pensiero che mi ha appassionato fin da subito, i modelli causali. Credo che proprio tramite questi si potranno ottenere risultati scientifici rivoluzionari. Il professore è stato un punto di riferimento, con il quale ho sempre avuto la possibilità di confrontarmi nella maniera migliore. Mi ha aiutato a sviluppare un metodo di ragionamento e di lavoro che porterò con me nelle prossime esperienze, ovunque siano. Lo ringrazio per tutte le opportunità che mi ha dato, è stato un piacere lavorare con lui e spero di averne ancora occasione.

Anche se non coinvolti in questa tesi, tengo particolarmente a ringraziare i professori Cocconcelli e Immovilli, che insieme al professor Lippi, mi hanno permesso di continuare la ricerca iniziata nella tesi triennale, sulla quale abbiamo pubblicato due articoli scientifici.

Devo assolutamente ringraziare la mia famiglia, che in tutti questi anni ha diviso con me "rogne" (tante) e "gioie" (tante): dai bei voti al tornare a casa e sentire "ho rifiutato il voto, si riparte a studiare". Credo che una parte considerevole di quanto ottenuto sia merito loro, per avermi aiutato a rimanere sereno e concentrato verso i miei obiettivi. Fare l'università è difficile, ma con una famiglia che comprende i momenti e le dinamiche diventa più semplice; voi l'avete fatto a vostro modo e, per me, l'avete fatto bene. Il debito che ho con voi, sia economico che emotivo è enorme. La famiglia è una delle cose più preziose che ho!! Soprattutto, grazie Mamma, grazie Papà e grazie a tutto il primo piano di casa, vi voglio bene.

Inoltre, complimenti a Peppo che, a 86 anni, ha conseguito la laurea magistrale in formato ridotto; in Erasmus non è riuscito a dare esami causa la difficoltà con la lingua inglese, ma ha trovato il modo per farseli abbonare.

Ringrazio i miei amici, quelli veri, che sono sempre con me, a sentirmi parlare di pizze, simulazioni infinite e grigliate; non tutti hanno amici come i miei, ed io lo so. Continuiamo a credere in qualcosa di migliore. Grazie, vi auguro il meglio!!

Ringrazio i miei nuovi amici, quelli che ho conosciuto in questi anni di università. Alla fine si va all'università per apprendere nuovi concetti, ma anche, e soprattutto, per conoscere persone, persone che ti stimolino e con la quale condividere esperienze. In particolare, ci tengo a fare un grande in bocca al lupo ad Italo e Filippo, anche se non ne hanno minimamente bisogno.

Ringrazio i miei nuovissimi amici conosciuti durante l'Erasmus a Monaco di Baviera, un'esperienza che porto con me con grande orgoglio: ho avuto l'opportunità di studiare in uno dei contesti universitari migliori di Europa, con studenti incredibili, tra cui voi. Siete delle persone stupende, anche se a primo impatto... non avrei detto. Sono stati sei mesi fantastici, grazie anche a voi. In bocca al lupo, siete dei grandi!!

Ringrazio Amanda, perché mi supporta e mi sopporta sempre; non è mai semplice stare accanto, e lei lo fa senza mai chiedere qualcosa in cambio (se non pizze fatte da me). Ci sono stati momenti non facili che mi ha aiutato a vincere. Ormai è diventata anche una profonda esperta di Reinforcement Learning e modelli causali, e, in generale, di Intelligenza Artificiale, a forza di sentirne parlare da me (spero per lei che abbia fatto sempre finta di ascoltarmi). Sa quanto tengo a determinate cose, e una di queste è proprio lei. Grazie e in bocca al lupo anche a te!!

Una considerazione personale, il senso di inadeguatezza e ogni tanto anche di timore attraversato dai noi giovani (universitari o meno) "di cui si sente parlare", è reale; si pensi a quello che succede nel mondo, ricchi idioti senza cognizione che lo distruggono con inquinamento, guerre, discriminazioni, maltrattamento di persone e animali, e tanto altro ancora... basta!! A peggiorare poi la situazione c'è la questione studenti universitari, dimenticati dalle istituzioni e, talvolta, anche malvisti dalla società e dalle stesse famiglie. Credo che un ragazzo debba avere la possibilità di studiare in maniera serena, solo così riuscirà a dare il 110%, solo così riuscirà a trovare la sua strada e, cosa più importante, sarà felice. Per mia fortuna, ho la possibilità di farlo, grazie alle persone citate sopra, la mia famiglia per prima.

In ultimo, ringrazio me stesso, non è facile rimanere sempre concentrati e determinati; qualche "schiaffo" inaspettato arriva, e quando arriva bisogna essere pronti ad incassarlo e pronti soprattutto a ricominciare. Chi mi conosce sa, non riesco ad accontentarmi: questo costituisce forse il mio più grande punto di forza ma, allo stesso tempo, anche un aspetto negativo che non mi fa sempre vedere dalla prospettiva giusta quanto fatto/costruito. In generale, di questo traguardo sono contento, è un percorso dove mi sono impegnato e dove credo di aver ottenuto i risultati voluti (da me). Ora si chiude un capitolo ma se ne apre un altro, probabilmente più complesso ed impegnativo, ma dove voglio prima di tutto divertirmi e svegliarmi al mattino contento di fare ciò che faccio (altrimenti che vado a farci?!).

Se la strada è in salita significa che FORSE sei destinato ad arrivare in alto... ho la testa dura e non mi arrendo, avanti tutta!! :)

Gio