A Reinforcement Learning Approach to Dynamic Norm Generation

by

Hadi Hosseini

Bachelor of Engineering, Computer Software Engineering
Amirkabir University of Technology (Tehran Polytechnic), 2008

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Computer Science

In the Graduate Academic Unit of Computer Science

THE UNIVERSITY OF NEW BRUNSWICK

August, 2010

Dedication

To my parents, Nasser and Rezvan, for their continuous love and support. I would never have been able to make it this far without you.

To my lovely fiance, Mehrnaz, for her presence in every moment of my life, her unconditional love, and her patience with me during the past two years.

Abstract

This thesis proposes a two-level learning framework for dynamic norm generation. This framework uses the Bayesian reinforcement learning technique to extract behavioral norms and domain-dependent knowledge in a certain environment and later incorporates them into the learning agents in different settings. Reinforcement learning (RL) and norms are mutually beneficial: norms can be extracted through RL, and RL can be improved by incorporating behavioral norms as prior probability distributions into learning agents. An agent should be confident about its beliefs in order to generalize them and use them in future settings. The confidence level is developed by checking two conditions: how familiar the agent is with the current world and its dynamics (including the norm system), and whether it has converged to an optimal policy.

A Bayesian dynamic programming technique is implemented and then compared to other methods such as Q-learning and Dyna. It is shown that Bayesian RL outperforms other techniques in finding the best equilibrium for the exploration-exploitation problem. This thesis demonstrates how an agent can extract behavioral norms and adapt its beliefs based on the domain knowledge it has acquired through the learning process.

Scenarios with different percentages of similarity and goals are examined. The experimental results show that the normative agent, having been trained in an initial environment, is able to adjust its beliefs about the dynamics and behavioral norms in a new environment, and thus it converges to the optimal policy more quickly, especially in the early stages of learning.

Acknowledgements

I would like to thank my supervisor, Dr. Ulieru, and my committee members Dr. Fleming, Dr. Dueck, and Dr. Biletskiy. I extend my appreciation to Dr. Michael Fleming for his constructive discussions and his fruitful comments.

Table of Contents

List of Tables

List of Figures

Abbreviations

DNG	Dynamic Norm Generation
DP	Dynamic Programming
LOE	Level of Exploration
MDP	Markov Decision Process
POMDP	Partially Observable Markov Decision Process
RL	Reinforcement Learning
TD	Temporal Difference
VPI	Value of Perfect Information

Chapter 1

Introduction

"Learning is not child's play; we cannot learn without pain."
Aristotle

In today's world of science and technology, there is always a necessity for urgent and precise decision-making mechanisms. Every decision support system needs a way of finding the best patterns of actions in order to push the overall behavior of agents and organizations to follow an optimal policy. This must be accomplished not only at a reasonable pace but also with a high degree of accuracy. However, these situations are not quite known from the beginning in any environmental or organizational decision-making case, thus yielding to a complex decision-making problem. There are four key requirements of decision making in complex environments: recognizing the necessity of a decision to be made, identifying relevant options [set of actions], determining when a decision will be made, and taking appropriate actions [22].

Automated decision making provides a rich ground in human-centered services and systems. Humans are usually affected by the impacts of the environment, organizational factors, and human factors such as emotions, risk, and stress. Therefore, humans are likely to make wrong decisions that might be irrevocable or expensive to be undone, and subsequently have severe consequences on the future of the system.

Every organization or system has a set of rules that are defined prior to any situation. Rules are usually set to be followed strictly by members of the system when a decision has to be made. On the other hand, every system develops its norms based on the behavior of its members (agents) and the outcome of their actions. Norms or conventions routinely guide the choice of behaviors in human societies, and conformity to norms reduces social frictions, relieves the cognitive load on humans, and facilitates coordination and decision making [61][46]. These norms differ in various situations depending on the environment's dynamics, behaviors of other agents (including peers and superiors), and many other factors affecting them. For instance, in a crisis situation caused by flooding or an earthquake, first responders are responsible to control and (sometimes) enforce some rules to the people such as evacuating the area or preventing people from looting shops. However, a first responder might decide to let people break into a drug store (against the dictated norms) in order to get medical equipment. As another example, a civilian may decide to help an elderly person or a child rather than escaping to a safe place to survive.

Norms can be violated or even changed depending on an environment's dynamics and many other factors. Whenever a decision has to be made, either in high levels of organizations or in lower levels, many aspects such as dynamics, former experiences, norms, and organizational culture should be considered. Due to this complexity, important questions will arise: How to gather information about dynamics? How to include experiences from previous events? How norms emerge, change, and get adapted to new circumstances? When to disobey norms? How organizational culture affects our decisions? To address these questions, we should automate the process of decision making by enabling our agents to learn and deduce about normative frameworks while accomplishing tasks toward a certain goal.

1.1 Motivation

The process of decision making can be enhanced by applying predefined norms. However, finding a reliable set of norms (rules) to initially code them into agents is a highly difficult task. By applying learning techniques, in fact, we can equip agents with proper tools for setting up new norms in every different environment. These norms emerge throughout the process of decision making in different simulations and can be used or updated by receiving new perception signals from the environment. This work mostly focuses on the individual level of decision making where organizations and agents need to make close-to-optimal decisions, considering all the uncertainties about other organizations' dynamics and the environment's dynamics. Learning and norms are mutually beneficial; learning equips agents with an understanding about dynamics and norms of uncertain environments, and norms enhance the process of decision making.

Agents need to be socialized and trained to be autonomous and self-reliant units, able to make decisions without consulting higher authority or when such authority is absent. In these scenarios agents should act in a more flexible and autonomous way, being able to adapt their beliefs and decisions to the environment's dynamics. On the other hand, learning ensures that organizations and agents are able to come up with policies wherever the need for emergent behaviors arises. Relying on an overly hierarchical system of decision making will not give an organizational responder enough flexibility to change its shape rapidly to suit response and recovery needs or what might be seen as an evolving challenge. Likewise, an overly free system - one completely devoid of structure - would also be flawed [53]. Unlike social laws, which need to be enforced in a top-down manner, norms evolve in a bottom-up manner and are typically more self-enforcing [61].

Reinforcement learning is, in fact, a natural way of learning. This is exactly the way we, as humans, start to learn and interact with our surrounding world. Our interaction with the environment results in negative or positive signals from the environment itself, or other agents that also can be considered to be part of the environment. Essentially, it is a traditional way of "reward & punishment"; getting a positive signal (reward) makes us want to repeat that action, and a negative signal (punishment) prohibits us from doing it again. However, as we get feedback about our actions in different environments, we memorize all the information we have acquired through various situations. This knowledge about the environment then can be used in other situations with different settings. However we do evaluate our actions over time and adjust them to make them suitable in different environments. This process is constantly in action by our brain without our noticing it. Therefore, we are always making our decisions based on knowledge gained through our experiences. This knowledge about the environment and about the consequences of our actions is derived from natural, social, and practical boundaries we face in our lifetime.

This thesis considers the required domain knowledge and guidelines as "behavioral norms", as they affect the behavior of agents and the ways of decision making in every state. Conte [26] and Dignum [34] discussed how norms and normative actions drive (or sometimes affect) the ways of decision making. These norms emphasize autonomy to agents on the side of decision. We believe that the best way to form behavioral norms is through a bottom-up approach (rather than emphasizing norms in a top-down fashion [71]) and the emergence of norms through reinforcement learning. This way we bridge the gap between learning agents [61] and their normative behaviors.

This study applies reinforcement learning (RL) methods to extract domain knowledge from the experiences learned over many simulations. I propose a dynamic norm generation (DNG) system, which is a two-level learning framework for decision making agents. Agents learn the model of the environment by choosing an action that may fulfill their intentions of finding the optimal policy and solving the problem while at the same trying to generalize norms (when they are confident about them) and store them as domain knowledge to be used for future environmental settings. Determining where and when to extract norms is done using probability distributions of every state-action pair.

In reinforcement learning, how to extract truly useful domain knowledge is still an open problem. On the other hand, norms are being designed by domain experts and dictated to agents in a top-down manner. I believe behavioral norms and domain knowledge in RL systems are mutually connected to each other. Norms should be obtained throughout the learning process after each simulation and then incorporated as prior knowledge using Bayesian techniques [29],[63],[20] in future RL settings.

Behavioral norms can always be altered by receiving new perceptual beliefs via interaction with the environment. Likewise, in an RL setting, beliefs are always being updated by perceiving the environment's dynamics and observing the consequences of actions (affected by the model of reward distribution or transition to desired/undesired states). While iterating in each episode, norms get updated and fed back into the agents. At the successful end of each simulation, these norms are extracted and initialized into the agents for other settings/environments. Following the extraction of behavioral norms in an automated approach, the verification and analysis of norms [71] would be a trivial task, consisting of repeated evaluation of norms while accessing and updating new beliefs.

In this thesis, it is shown that Bayesian reinforcement learning is an effective tool to extract norms and would increase the overall performance of the system. After that, the effectiveness of coding behavioral norms as domain knowledge is investigated by attempting to maximize the amount of cumulative reward in each simulation.

1.2 Research Contributions

The main purpose of this thesis is to develop a reinforcement learning framework to dynamically generate norms. In this thesis I extend the state of the art as follows:

- Firstly, I give a detailed overview of the research work that has been done previously in the area of reinforcement learning. I implement model-free and model-based reinforcement algorithms such as Q-learning [76], proposed by Watkins, and the Dyna architecture [66], developed by Sutton. I also provide a review of norms and normative frameworks from the artificial agent's point of view, describing the necessity of defining norms and the previous work done in this area.

This work generalizes the Bayesian reinforcement learning technique. Bayesian RL gives us the ability to code prior knowledge into agents, and models the environment as a Markov Decision Process (MDP). I implement this algorithm and show how well this approach meets our requirements to extract norms.

- Secondly, I provide an extensive comparison between the Bayesian RL technique and other techniques, based on the criteria that are important for norm generation. The results demonstrate how Bayesian reinforcement learning outperforms any other technique in solving decision-making problems and finding an appropriate balance for the exploration-exploitation problem.
- Thirdly, I propose a reinforcement learning framework for dynamic norm generation (DNG). This framework is a two-level learning system to optimize the behavior of agents via learning in an uncertain environment and the emergence of norms. It is shown that Bayesian reinforcement learning is a natural way of extracting behavioral norms.
- Finally, I demonstrate that DNG method is effective in learning the model of the environment, dynamically generating norms that are applicable to various settings, and integrating domain knowledge into

decision-making systems.

1.3 Thesis Structure

In this thesis I expand on each of the above contributions in turn:

- Chapter 2 examines in detail the background to this research with a literature review that highlights our key decisions. The important ideas in the state of the art, both in the reinforcement learning field and studying of norms, are explained.
- Chapter 3 describes the techniques to solve the Bayesian reinforcement learning algorithm and extends this technique's capability to incorporate knowledge into the decision-making process of agents. A two-level reinforcement learning framework is proposed to automate the process of norm generation. The last part of this chapter discusses how this approach enables us to proactively gather information about norms and reuse them as domain knowledge in different settings.
- Chapter 4 shows the experimental results and analysis. Firstly, it shows a comparison between a model-free reinforcement algorithm such as Q-learning [76], a middle-ground approach of Dyna [66], and Bayesian reinforcement learning without considering norms. Secondly, this chapter develops a scenario of two agents, one without any initialized knowledge about the environment and the other one with behavioral norms

learned in previous experiences, and then compares the results based on performance, exploration, and convergence rate.

- Chapter 5 concludes the thesis, outlining the way this approach can be extended to address more sophisticated problems and suggesting a number of directions for future study.

Chapter 2

Literature Review

2.1 Introduction

This chapter provides an overview of the basic concepts related to the problems that are tackled throughout this thesis, specifically background on different approaches in reinforcement learning and normative frameworks. In the process, this chapter draws the connections among those fields, focusing on the concepts and problems to bridge the gap between reinforcement learning and the emergence of norms.

2.2 Autonomous Agents

In artificial intelligence, an intelligent agent (IA) is an autonomous entity that observes and acts upon an environment (i.e., it is an agent) and directs its activity towards achieving goals (i.e., it is rational). Intelligent agents may also learn or use knowledge to achieve their goals. They may be very simple or very complex: a reflex machine such as a thermostat is an intelligent agent, as is a human being, as is a community of human beings working together towards a goal [59].

There are two notions used to describe intelligent agents' characteristics in artificial intelligence communities [80]. The first notion, which was described earlier, is considered to be a weak description of agents; however, it is more common and sensible in terms of usage. The second notion, which is stronger than the first one, considers agents with the properties proposed in the weak notion plus some more human-like characteristics. It characterizes agents with mentalistic (such as knowledge, belief, intention, and obligation [62]) and emotional attributes [2],[3]. These mentalistic and emotional attributes include but are not limited to the following: mobility - the ability of an agent to move around an electronic network [77]; veracity - the assumption that an agent will not knowingly communicate false information [38]; benevolence - the assumption that agents do not have conflicting goals and that every agent will therefore always try to do what is asked of it [58]; and rationality - the assumption that an agent will act in order to achieve its goals and will not act in such a way as to prevent its goals being achieved [38],[80].

However, all the AI communities accept the following properties for agents:

- Autonomy: agents operate without the direct intervention of humans or others and have some kind of control over their actions and internal state [17],[12];
- Social ability: agents interact with other agents (and possibly humans) via some kind of agent-communication language [39];
- Reactivity: agents perceive their environment and respond in a timely fashion to changes that occur in it [44];
- Proactiveness: agents do not simply act in response to their environment; they are able to exhibit goal-directed behavior by taking the initiative [79];
- Situated (embedded) in a particular environment: they receive inputs related to the state of their environment through sensors, and they act on the environment through effectors [44];
- Purposeful (or intentional): they have particular objectives (goals) to achieve and are designed to fulfill a specific purpose [44].

It is argued that the development of robust and scalable software systems requires autonomous agents that can complete their objectives while situated in a dynamic and uncertain environment, that can engage in rich, high-level social interactions, and that can operate within flexible organizational structures [44].

As a definition of what agenthood is actually about, an increasing number of researchers find the following characterization useful [79]:

> *An agent is an encapsulated computer system that is situated in some environment and that is capable of flexible, autonomous action in that environment in order to meet its design objectives.*

As it is unrealistic to know *a priori* about the environment and all potential links and interactions in a system, it is impossible to try to predict or analyze all the possibilities at design-time. Rather, it is more realistic to endow the components with the ability to make decisions about the nature and scope of their interactions at run-time [44]. Thus, agents should be able to deal with unanticipated events in uncertain environments where their internal and external states are subject to change at any time. All agents are continuously active, and any coordination that is required is handled bottom-up through inter-agent interaction. Therefore, the ordering of the system's top-level goals is no longer something that has to be rigidly prescribed at design time. Rather, it becomes something that is handled in a context-sensitive manner at run-time [44]. We discuss this approach in Section 2.4.

When reasoning, an agent may maintain an explicit world model, or it may leave the model implicit as it reasons about plans. In this setting, explicit models have more potential for reasoning about states and behaviors, as they store more information explicitly. However, maintaining explicit models may be computationally and memory intensive [37].

2.3 Deterministic vs. Non-Deterministic Environments

"You cannot be certain about uncertainty."
Frank Knight

Autonomous agents interact in an environment to accomplish a task or fulfil a goal. Thus, they should have an understanding of the world they are interfacing with. Uncertain reasoning considers a deterministic model for the agent's view of its world. It talks about worlds as models with certain dynamics that are able to evolve from one state to another state deterministically. Logic represents uncertainty by disjunction; however, it is unable to tell us how likely each condition is to happen. Probability theory provides a quantitative way of encoding the likelihood of each condition. What is the difference between deterministic and non-deterministic processes?

If we have complete information about the environment (including anything), one can predict the upcoming events, and so call this a deterministic process. For example, flipping a coin can be considered deterministic if we know all the forces and all the other factors in the room. Nevertheless, the lack of information in most of the situations makes the process non-deterministic.

It's an important thing to at least keep in mind that the term "uncertainty" is used to really talk about two kinds of things, to talk about real randomness in a process and to talk about our uncertainty about a process. When performing actions under uncertainty, there should be a mechanism to reduce the uncertainty about the environment by perceiving information through signals and performing an action. This process is naturally known as learning. Learning enables agents to initially operate in unknown environments and to become more competent than their initial knowledge alone might allow.

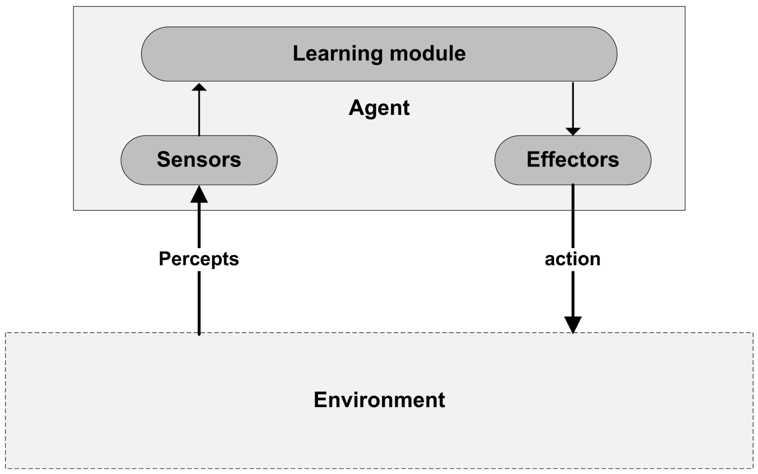


Figure 2.1: Learning agent: agent interaction with unknown environment

2.4 Reinforcement Learning

Learning refers to acquiring new knowledge, behaviors, skills, values, preferences, or understanding and may involve synthesizing different types of information [24],[43]. Reinforcement learning is a promising tool to support the decision-making process. Informally speaking, decision making is the process of identifying "what to do" and "when to do it" to solve the problem of planning ("How to do").

Reinforcement Learning is probably the first and in fact the most effective method of natural learning. The connections we have with our environment produces a wealth of information about cause and effect, the consequences of actions, and our decisions in order to achieve goals [68],[59]. It is, in fact, a form of trial-and-error learning by interaction, in which an agent gets to choose its action among all the possible actions that are permissible in that state. The agent receives rewards when these actions lead to the successful performance of the task [63]. Generally, the amount of control the agent has over its states distinguishes it from its environment. Anything that cannot be changed "directly" by the agent is considered to be outside of it and thus part of its environment.

The agent and environment interact at each of a sequence of discrete time steps, t = 0,1,2,3.... At each time step , the agent receives some representation of the environment's state, $s_t \in S$, where S is the set of possible states, and on that basis selects an action, $a_t \in A(s_t)$, where $A(s_t)$ is the set of actions available in state s_t. One time step later, in part as a consequence of its action, the agent receives a numerical reward, $r_{t+1} \in \Re$, and finds itself in a new state, s_{t+1} [68].

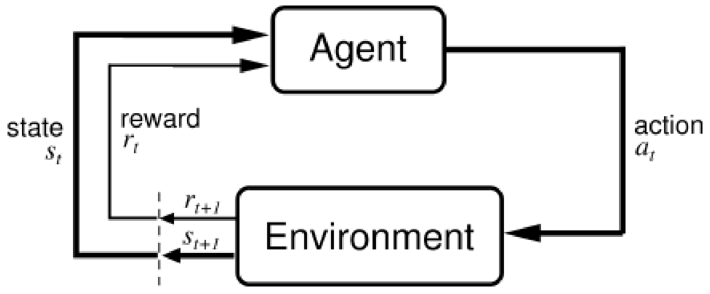


Figure 2.2: The agent-environment interaction in reinforcement learning.

In reinforcement learning, the purpose or goal of the agent is formalized in terms of a special reward signal passing from the environment to the agent. Informally speaking, an agent's goal is to try the best actions in order to maximize the total amount of reward it receives. The reward signal is our way of communicating to the agent what we want it to achieve, not how we want it achieved. A good way to place this kind of prior knowledge is using an initial policy or value function [68][23].

The agent's job is to find a policy π, mapping states to actions, that maximizes some long-run measure of reinforcement. We expect, in general, that the environment will be non-deterministic; that is, taking the same action in the same state on two different occasions may result in different next states and/or different reinforcement values [45].

In most supervised learning problems, the agent is simply given a big collection of data and asked to learn something from it. In reinforcement learning, there is an interesting added dimension: the agent gets to choose its own actions; therefore, it has very direct influence on the data it will receive.

After choosing an action, the agent is told the immediate reward and the subsequent state but is not told which action would have been in its best long-term interest. It is necessary for the agent to gather useful experience about the possible system states, actions, transitions, and rewards actively to act optimally. In the most interesting and challenging cases, actions affect not only the immediate reward but also the next situation and through that all the subsequent rewards. These two characteristics, trial-and-error search and delayed reward, are the two most important distinguishing features of reinforcement learning [67]. There are two possibly opposing reasons for the agent to choose an action: because it thinks the action will have a good result in the world or because it thinks the action will give it more information about how the world works.

Planning AI algorithms are less general than the reinforcement learning methods in that they require a predefined model of state transitions and, with a few exceptions, assume determinism. On the other hand, reinforcement learning, at least in the kind of discrete cases for which theory has been developed, assumes that the entire state space can be enumerated and stored in memory [45].

In learning under uncertainty, to obtain a maximum amount of reward, a reinforcement learning agent must prefer actions that it has tried in the past and found to be effective in producing reward. But to discover such actions, it has to try actions that it has not selected before. The agent has to *exploit* what it already knows in order to obtain reward, but also it has to *explore* in order to make better action selections in the future. Thus, the behavior of the agent changes from near-stochastic (exploration) to near-deterministic (exploitation). There is a question of how long you should gather data to estimate the model before it is good enough to use to find a policy [1].

A model of the environment is actually something that represents the behavior of the environment. Models are used for planning, that is, any way of deciding on a course of action by considering possible future situations before they are actually experienced.

2.4.1 Models of Reward

To model any learning algorithm, one should also define its model of optimal behavior. It should be specified how the agent should take the future into account in the decisions it makes about how to behave now. If, at a given moment in time, the agent optimizes its expected reward for the next h steps and needs not worry about what will happen after that, we are facing a *finite-horizon* model. The expected reward for a finite-horizon model is:

$$E(\sum_{t=0}^{h} r_t) \tag{2.1}$$

On the contrary, if our model takes the long-run reward of the agent into

[1]We will discuss this further in Section 2.4.8.

account, it is considered to be an *infinite-horizon* model. However, the rewards that are received in the future should be discounted, as they need to be weighted down compared to the immediate rewards. Therefore, the expected reward in an infinite-horizon model, where the discount factor is $0 \leq \gamma \leq 1$, is:

$$E(\sum_{t=0}^{\infty} \gamma^t r_t) \tag{2.2}$$

Another optimality criterion is the average-reward model in which the agent is supposed to take actions that optimize its long-run average reward:

$$\lim_{h \to \infty} E(\frac{1}{h} \sum_{t=0}^{h} r_t) \tag{2.3}$$

Such a policy is referred to as a gain optimal policy; it can be seen as the limiting case of the infinite-horizon discounted model as the discount factor approaches 1 [8].

2.4.2 The Markov Property

When making decisions to find an optimal policy in reinforcement learning, it is of a great importance to consider the effects of previous states and chosen actions as well as keeping an eye on future states. This property is called *The Markov Property*. An environment satisfies the Markov Property (or is said to be Markov) if its state signal compactly summarizes the past without degrading the ability to predict the future. In most cases this can not be achieved precisely, but often nearly so; the definition of environment and state signal should be constructed in a way that it holds the Markov Property as nearly as possible [68].

The model of long-run optimality that the agent is using determines exactly how it should take the value of the future into account. The agent will have to be able to learn from delayed reinforcement: it may take a long sequence of actions, receiving insignificant reinforcement, then finally arrive at a state with high reinforcement. The agent must be able to learn which of its actions are desirable based on rewards that can take place arbitrarily far in the future. Problems with delayed reinforcement where the Markov property does hold are well modeled as a *Markov decision process* (MDP).

2.4.3 Markov Decision Processes

The Markov decision process (MDP) [45] forms the theoretical foundation for reinforcement learning problems when the environmental dynamics (reward function and/or transition function) are unknown [4] [13] [68]. In an MDP, the agent perceives the state of the worlds through its sensory inputs and decides on its immediate action based on this state.

We assume a learning agent controlling a stochastic environment modeled as an MDP. An MDP is a 4-tuple $\langle S, A, \mathrm{P}_T, \mathrm{P}_R \rangle$ where S is a finite set of states, A is a set of actions, P_R is a reward function, and P_T is a transition model that captures the probability of reaching state $s' \in S$ after we execute action $a \in A$ at state $s \in S$. Thus, $P_R(s, a, r)$ denotes the probability of obtaining reward r after executing a at s. We focus on infinite-horizon MDPs with a discounted factor $0 \leq \gamma \leq 1$. The agent is charged with constructing an optimal Markovian policy $\pi : S \mapsto A$ that maximizes the expected sum of future discounted rewards over an infinite horizon.

Letting $V^*(s)$ at each $s \in S$ denote the optimal expected discounted reward achievable from state s and $Q^*(s, a)$ denote the value of executing action a at s, we have the standard Bellman equations [4]:

$$V^*(s) = max_{a \in A} Q^*(s, a) \tag{2.4}$$

$$Q^*(s, a) = E_{P_R(s,a,r)}[r|s, a] + \gamma \sum_{s' \, \in \, S} P_T(s, a, s') V^*(s') \tag{2.5}$$

These equations show that the quality Q of a state-action pair is the immediate reward plus the expected discounted value of all succeeding states weighted by their likelihood, and that the value V of a state is the quality of the best action for that state. Once one has V^*, it is easy to determine an optimal policy: any policy that is greedy with respect to V^* is an optimal policy.

Maximizing long-term rewards, either in an infinite or finite horizon, is a good measure to assess the policies learned by a given algorithm. Moreover, the quality of learning is a non-negligible factor in measuring algorithms. Important issues to note are convergence of one policy to an optimal behavior, or the rate of convergence (whether the algorithm converges fast to optimality or near-optimality), and the regret of the algorithm (i.e the expected decrease in reward gained due to executing one algorithm instead of having the system behave optimally from the very beginning).

There are two major classes of algorithm in solving MDPs: model-based and model-free algorithms. Model-based methods are well developed and proven mathematically but require a complete and accurate model of the environment (or a well defined conceptual/imaginary model). Model-free algorithms, however, do not require a model and are conceptually simple but are not well suited for step-by-step incremental computation.

Model-free approaches attempt to learn near-optimal policies without explicitly estimating the dynamics of the surrounding environment. This is usually done by directly approximating a value function that measures the desirability of each environment state. On the other hand, model-based approaches attempt to estimate a model of the environment's dynamics and use it to compute an estimate of the expected value of actions in the environment [29][67].

2.4.4 Model-Based Algorithms

The class of dynamic programming methods are well known as model-based algorithms. Dynamic programming algorithms require a model of the environment. This makes it useful when there is a need to learn the environment and gain some knowledge about the environment.

In the case that transition and reward models are given, this policy and its value, $V^*(s)$ at each $s \in S$, can be computed using standard dynamic programming techniques [4] such as value or policy iteration. Value iteration starts with estimates Q and V of Q^* and V^*, respectively, and updates them repeatedly by applying the previous equations to get new values for Q and V [45] [64]. It has been shown that the estimated values for Q and V converge to their true values [7] [56].

Policy iteration, on the other hand, manipulates the policy directly rather than finding it indirectly via the state-value function. The value function of a policy, V_π, is the expected infinite discounted reward that will be gained at each state by the execution of that policy. It can be computed by solving a set of linear equations. This constitutes the policy evaluation step of the algorithm. Once the value of each state under the current policy is known, a policy improvement step is tried by changing the first action taken when in a state. If the value of the state can be improved, the new action is adopted by the policy; thus, the policy is strictly improved. The algorithm iterates policy evaluation and improvement steps, until no further improvements are

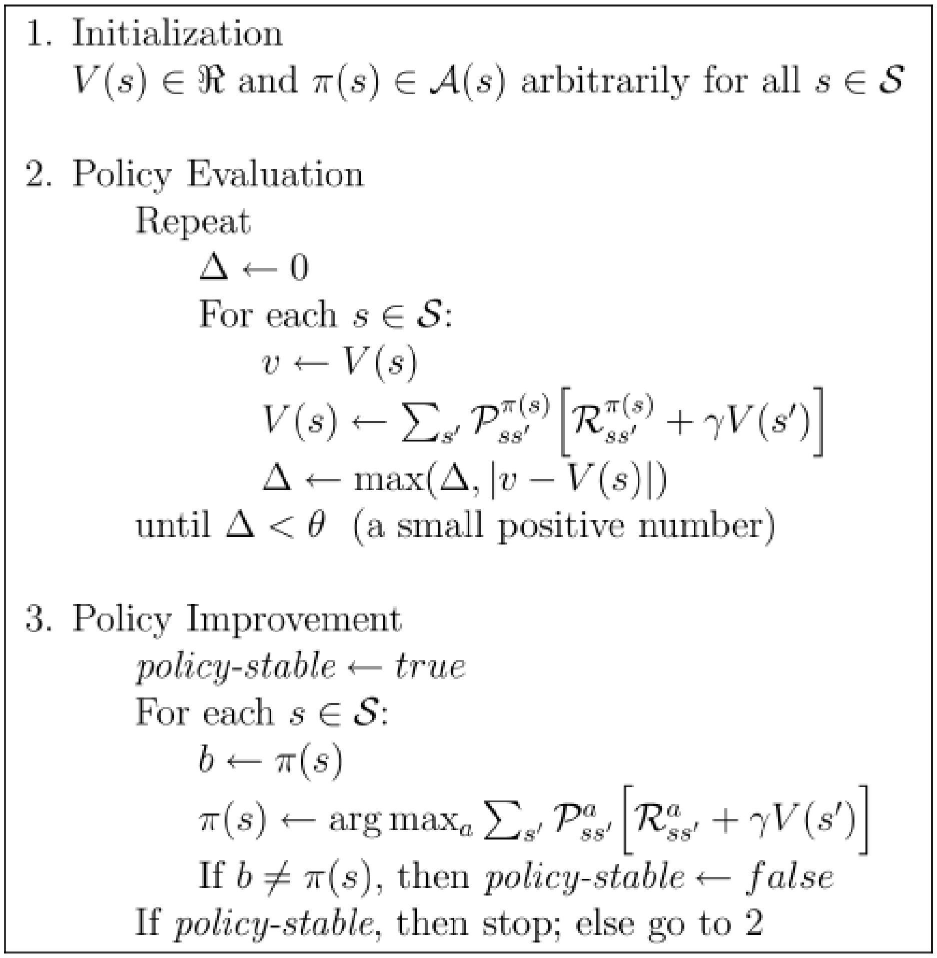
```
Initialize $V$ arbitrarily, e.g.,$V(s) = 0$, for all $s \in \mathcal{S}^+$

Repeat
    $\Delta \leftarrow 0$
    For each $s \in \mathcal{S}$:
        $v \leftarrow V(s)$
        $V(s) \leftarrow \max_a \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V(s')\right]$
        $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)

Output a deterministic policy, $\pi$, such that
    $\pi(s) = \arg\max_a \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V(s')\right]$
```

Figure 2.3: The value iteration algorithm [68]

possible. The policy is then guaranteed to be optimal [41].

All of the DP algorithms update estimates of the values of states based on estimates of the values of successor states, that is, they update estimates on the basis of other estimates. This idea is called bootstrapping and many reinforcement learning algorithms perform this, even those that do not require a complete and accurate model of the environment. Moreover, some model-free algorithms do not bootstrap (such as Monte Carlo methods [68]) and some do bootstrap, even though they do not require the model of the environment.

Another well-known model-based RL algorithm is prioritized sweeping [50]. Prioritized sweeping is a model-based approach in which the number of Bell-



```
1. Initialization
   $V(s) \in \Re$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
       Repeat
           $\Delta \leftarrow 0$
           For each $s \in \mathcal{S}$:
               $v \leftarrow V(s)$
               $V(s) \leftarrow \sum_{s'} \mathcal{P}_{ss'}^{\pi(s)} \left[ \mathcal{R}_{ss'}^{\pi(s)} + \gamma V(s') \right]$
               $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
       until $\Delta < \theta$ (a small positive number)

3. Policy Improvement
       *policy-stable* $\leftarrow$ *true*
       For each $s \in \mathcal{S}$:
           $b \leftarrow \pi(s)$
           $\pi(s) \leftarrow \arg\max_a \sum_{s'} \mathcal{P}_{ss'}^{a} \left[ \mathcal{R}_{ss'}^{a} + \gamma V(s') \right]$
           If $b \neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*
       If *policy-stable*, then stop; else go to 2
```

Figure 2.4: The policy iteration algorithm [68]

man backups is applied to the states that seem to be more important.

Last but not least, Dearden et al. [29] have presented a way to be Bayesian when employing model-based RL. Chapter 3 will describe the Bayesian RL approach in detail.

2.4.5 Model-Free Algorithms

Model-free delayed RL methods are derived by making suitable approximations to the computations in value iteration and policy iteration so as to eliminate the need for a system model. Two important methods result from such approximations: Barto, Sutton, and Anderson's actor-critic method [68] and Watkin's Q-Learning method [76].

Model-free algorithms try to learn an optimal policy without having to explicitly learn and use a model of the environment. These models are used whenever there is no model or there is just an incomplete model of the environment.

Sutton's TD(0) [65] is a model-free method that learns the value of a policy π by using the update rule:

$$V(s) = V(s) + \alpha(r + \gamma V(s') - V(s)) \tag{2.6}$$

Whenever a state s is visited, its estimated value is updated to be closer to $r + \gamma V(s')$, since r is the instantaneous reward received and $V(s')$ is the estimated value of the actually occurring next state.

TD(0) looks only one step ahead when adjusting the value estimates. Its multi-step version, TD(λ) [65], uses an update rule that is similar to the one used by TD(0):

$$V(u) = V(u) + \alpha(r + \gamma V(s') - V(s))e(u), \tag{2.7}$$

The rule is applied not just to the immediately previous state but to every state u, according to its eligibility e(u), which shows the degree to which it has been visited in the past and is a function of λ, where $0 \leq \lambda \leq 1$. TD(λ) is a way of averaging the n-step backups. The one-step return has weight $\lambda - 1$, the two-step return has weight $(\lambda - 1)\lambda$, the three-step return has weight $(\lambda - 1)\lambda^2$ and so on. Thus, for $\lambda = 0$, TD(λ) becomes TD(0). When $\lambda = 1$, it is roughly equivalent to updating all the states according to the number of times they were visited by the end of a run [19].

One of the most interesting model-free algorithms is the Q-learning algorithm [76], which uses an update method to update the Q-values based on the actions chosen previously. Q-learning is an off-policy TD control method; it estimates the Q-values online[2] by using essentially TD(0), but at the same time it uses them to define a policy. An action is chosen just by acting greedily with respect to the Q-values. An update is executed by an agent whenever it receives a reward of r by taking action a and making a transition from s to s'. The update rule is:

$$Q(s, a) = Q(s, a) + \alpha(r + \gamma max_{a'} Q(s', a') - Q(s, a)) \tag{2.8}$$

[2]It is online as it is getting updated constantly when choosing actions.

The probability with which this happens is precisely $P_T(s, a, s')$, which is why it is possible for an agent to carry out the appropriate update without using a transition model. The learned Q-value function directly approximates Q^* independently of the policy being followed, but of course the policy followed has an effect on which state-action pairs are visited and updated. In any case, if each action is tried infinitely often and α is decayed appropriately, the Q-values will converge to Q^* [76].

The conventional approach to solve this problem is to consider some occasional random actions or some other ways of enhancing exploration by sampling [68]. Two commonly used methods are semi-uniform random exploration [78] and Boltzman exploration [75]. The former assigns the probability of p to the best action, and $1-p$ for choosing an action randomly. The latter, which is more sophisticated, uses the temperature parameter that can be decreased slowly over time, resulting in less exploration. However, both of these methods do not consider the prior knowledge of exploring information in an optimized way.

2.4.6 Dyna Architecture

Sutton's Dyna architecture [66] exploits a middle ground, yielding strategies that are both more effective than model-free learning and more computationally efficient than the certainty-equivalence approach. It simultaneously uses experience to build a model ($\hat{T}$ and $\hat{R}$), uses experience to adjust the policy,

```
Initialize $Q(s, a)$ arbitrarily
Repeat (for each episode):
    Initialize $s$
    Repeat (for each step of episode):
        Choose $a$ from $s$ using policy derived from $Q$ (e.g., $\varepsilon$-greedy)
        Take action $a$, observe $r$, $s'$
        $Q(s, a) \leftarrow Q(s, a) + \alpha\big[r + \gamma \max_{a'} Q(s', a') - Q(s, a)\big]$
        $s \leftarrow s'$;
    until $s$ is terminal
```

Figure 2.5: Q-learning algorithm: a model-free delayed reinforcement learning approach [68]

and uses the model to adjust the policy. Dyna operates in a loop of interaction with the environment using an experience tuple. It first updates the policy based on the updated model using value-iteration updates for Q values, then performs k random state-action pair selections and updates them using the same value-iteration update, and finally chooses the action a' to perform in state s' based on the Q values modified by an exploration strategy.

Given an experience tuple $\langle s, a, s', r \rangle$:

1. Update the model, incrementing statistics for the transition and the reward to get ($\hat{T}$ and $\hat{R}$).

2. Update the policy at state s based on the newly updated model using value-iteration update.

3. Perform k additional updates by choosing k state-action pairs randomly using the same value-iteration update.

4. Based on the Q values, choose an action a' with an exploration strategy (like an exploration bonus: every once in a while randomly explore).

The Dyna algorithm requires about k times the computation of Q-learning per instance, but this is typically vastly less than for the naive model-based method. A reasonable value of k can be determined based on the relative speeds of computation and of taking action [45]. To keep the exploration-exploitation trade-off in a good balance (and possibly to be usable in non-stationary environments), the Dyna architecture uses an exploration strategy. This strategy considers an exploration bonus element that make the Dyna agent periodically explore different states every time it exploits some known states for more rewards.

Dyna suffers from being relatively undirected, and it is particularly unhelpful when the goal has just been reached or when the agent is stuck in a dead end. In this situation, the Dyna agent continues to update random state-action pairs rather than concentrating on the interesting parts of the state space [45][66]. To overcome this problem, prioritized sweeping [50] and Queue-Dyna [54] were developed. These are methods that estimate to what extent states would change their value as a consequence of new knowledge of the MDP dynamics or previous value propagations. States are assigned priorities based on the expected size of changes in their values, and states with the highest priority are the ones for which we perform value propagation [29].

2.4.7 Partially Observable Markov Decision Processes

In this section, another class of MDPs, called Partially Observable Markov Decision Processes (POMDP) [49] [45] [35], will be briefly discussed. When dealing with MDPs, it is assumed that the current state is fully observable. On the other hand, a POMDP refers to an MDP in which the agent is unable to observe its current state and is just able to make some probabilistic observations about it. Any POMDP can be represented as a belief-state MDP because its states are actually belief states. Belief states are probability distributions over the states of the world, and they include a sufficient statistic for the past history. Therefore, as discussed in [1], the optimal solution of the belief-state MDP gives rise to optimal behavior of the original POMDP.

2.4.8 Exploration-Exploitation Problem

In learning under uncertainty, not all the information about the model of the environment is available. Hence, there should be a mechanism for agents to gather data by searching through different states and forming its own experience to learn the model of the environment and find the rewards. Also, to obtain a lot of reward, a reinforcement learning agent must prefer actions that it has tried in the past and found to be effective in producing rewards. But to discover such actions, it has to try actions that it has not selected before. The agent has to exploit what it already knows in order to obtain reward, but also it has to explore in order to make better action selections in the future. This is called the Exploration-Exploitation dilemma [68][69], which is a trade-off between exploiting current knowledge versus exploring new states to gain more knowledge about the environment. On a stochastic task, each action must be tried many times to gain a reliable estimate of its expected reward.

There is a question of how long an agent should gather data to estimate the model before it is good enough to use to find a policy. One way to get around this problem is to re-estimate the model on every step. Each time an agent gets a piece of experience, it updates its model estimates. Then, the agent runs value iteration on the updated model. This may be too computationally expensive; if so, the agent can run value iteration over the continually updated model as a kind of background job at whatever rate it can afford computationally.

It has been proven that the Bayesian approach to reinforcement learning provides an optimal solution to the exploration-exploitation problem. It is argued in [55], that an optimal policy of the POMDP formulation of the Bayesian RL optimizes the exploration-exploitation trade-off simply based on the fact that such a policy maximizes the expected total return.

2.4.9 Bayesian Inference and Learning

In previous sections, two major classes in reinforcement learning were discussed: model-free and model-based methods. Model-free approaches attempt to learn optimal policies without explicitly estimating the model and dynamics of the environment. On the contrary, model-based approaches attempt to compute an estimate of the expected value of actions by learning the dynamics of the environment (both in stationary or non-stationary environments).

Different estimation methods (such as point estimates) are being used to capture the model of the environment; however, most of them ignore the agent's uncertainty about the environment's dynamics. Dearden et al. in [30] showed that, under some reasonable assumptions and given past experiences, it is possible to represent the posterior distribution over possible models of the environment. By representing a distribution over possible models, one can quantify the agent's uncertainty about the dynamics of the environment and determine the best actions to perform in every state. Their Bayesian approach to model-free and model-based reinforcement learning is based on simple Bayesian inference. Bayesian inference is statistical inference in which evidence or observations are used to update or to newly infer the probability that a hypothesis may be true. The name "Bayesian" comes from the use of Bayes' theorem in the updating process. Bayes' theorem was introduced by Reverend Thomas Bayes in 1764.

Bayes' theorem shows the relation between one conditional probability and its inverse; for example, the probability of a hypothesis given observed evidence and the probability of that evidence given the hypothesis. Bayes' theorem adjusts probabilities given new evidence in the following way:

$$P(H|E) = \frac{P(E|H) \cdot P(H)}{P(E)} \tag{2.9}$$

where H represents a specific hypothesis, and E is the observed evidence, and:

- $P(H|E)$ is the conditional probability of H, given E. It is also called the posterior probability because it is derived from or depends upon the specified value of E,
- $P(E|H)$ is called the conditional probability of seeing the evidence E if the hypothesis H happens to be true. It is also called a likelihood function when it is considered as a function of H for fixed E,
- P(H) is called the prior probability of H that was inferred before new evidence, E, became available (also called *epistemological* uncertainty[3]),
- P(E) is the prior or marginal probability of E and acts as a normalizing constant.

Bayesian inference uses a numerical estimate of the degree of belief in a

[3]Epistemology or theory of knowledge is the branch of philosophy concerned with the nature and scope (limitations) of knowledge [40][14].

hypothesis before evidence has been observed and calculates a numerical estimate of the degree of belief in the hypothesis after evidence has been observed.

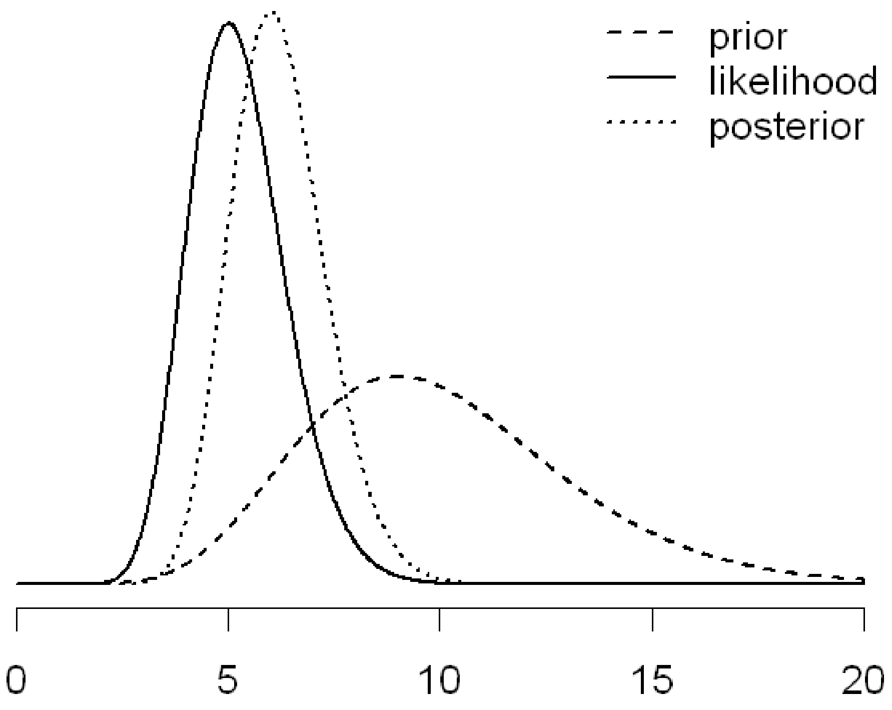


Figure 2.6: Prior, likelihood and posterior distributions for a simple example.

Prior, likelihood and posterior distributions of a simple example are shown in Figure 2.6. It shows how the shape of the likelihood distribution affects the prior distribution, to form the posterior distribution. The posterior is primarily influenced by the likelihood function but is shrunk towards the prior distribution. This reflects the fact that the expectation based on external evidence was a higher rate than actually observed. The posterior distribution is a formal compromise between the likelihood, summarizing the evidence in the data alone, and the prior distribution, which summarizes external evidence which suggested higher rates.

Using Bayes' updates, Dearden et al. introduced a Bayesian model-free approach [30] in which uncertainty about Q-values of actions is represented by probability distributions. By explicitly reasoning using uncertainty about Q-values, they direct exploration specifically toward unvisited regions of the state space. This is a good solution to overcome the exploration-exploitation dilemma. Their method is based on a decision-theoretic approach [42] to action selection, where the agent chooses actions based on the value of the information it can expect to learn by performing them. The Bayesian approach to Q-learning is not solely based on local Q-value information; however, it keeps distributions over the Q-value and propagates them, rather than just estimating points. In this approach, one usually estimates Q-values by treating each step in the environment as a sample from the underlying dynamics. These samples are then used for updating Q-values based on the Bellman equations and propagating the distributions.

Although model-free methods are computationally easier to implement, learning a model of the environment avoids costly repetition of steps in model-based approaches.

To avoid costly repetition of steps in model-free methods, a model-based approach to learning is introduced. The Bayesian model-based approach was first introduced by Dearden et al. in an attempt to balance exploration of untested actions against exploitation of actions that are known to be good. The benefit of exploration is estimated using the classical notion of value of information. This method explicitly represents uncertainty about the parameters of the model and builds probability distributions over Q-values based on them. These distributions are used to compute a myopic approximation to the value of information for each action.

Representing Q-values using probability distributions and estimating the world using the possible MDPs are beneficial to learn the model of the environment as well as giving us an opportunity to study the agent-environment interaction. As discussed earlier, estimating the model of the environment is helpful to finally find a compromised solution for the exploration-exploitation problem. However, the Bayesian approach gives us an opportunity to add prior knowledge in the shape of probability distributions. To consider the prior experiences, the Bayesian techniques have been developed that use Bayesian inference over the agent's beliefs. This approach perfectly solves the exploration-exploitation problem by considering the distribution of rewards and transitions. It is in fact a systematic method of inclusion and update of prior knowledge and domain assumptions.

By representing a distribution over possible models, an agent's uncertainty can be quantified, which can in turn be used to inform it as to what are the best actions to perform. More detail on this will be provided in Chapter 3 while modifications and enhancements to this algorithm are discussed.

2.4.10 Domain Knowledge

Reinforcement learning agents always try to complete their set of beliefs about the reward distribution and state transition by constant interaction with the environment. The behavior of the agent changes from near-stochastic (exploration) to near-deterministic (exploitation) over time. Bayesian RL is a systematic way of equipping agents with useful initial knowledge about the environment's dynamics, social behaviors, interaction patterns, *etc.* As discussed in the previous section, the prior probability of a hypothesis is where external knowledge can be coded. When agents have no prior information about their beliefs, prior probability will have no useful effect. However, if we have some information from a previously performed test or an external knowledge base this can be taken into account in the computations. By providing agents with knowledge about the domain and some general guidelines, we can reduce the random behavior of the agents and significantly speed up the process of learning.

2.5 Norms in Artificial Intelligence

Norms are a widely observed mechanism for enforcing discipline and prescribing uniform behavior in human societies. Norms specify the way the members of a society should behave and help societies to improve co-operation and collaboration among their members [10].

Norms have various definitions based on the academic fields and research community in which they have been studied. According to sociology, “a norm is a rule or standard of behavior shared by members of a social group” (Encyclopedia Britannica [16]). Norms can be seen as describing expected or standard ways of behavior [27]. According to philosophy, “a norm is an authoritative rule or standard by which something is judged and, on that basis, approved or disapproved” (Columbia Encyclopedia [25]). According to economics, a norm (from norma, Latin for carpenter’s rule) is a model of what should exist or be followed, or an average of what currently does exist in some context, such as an average salary among members of a large group [72].

There is a significant difference between social laws (rules) and norms. Some norms are explicit and can be enforced by law. For example, people are always obliged to follow the orders of a police officer. Evacuating an area in a crisis situation is an example of such norm. These norms are enforced from higher authorities and mostly are known as social laws (or rules). On the contrary, some norms are established more implicitly and are being followed by individuals based on their discretion and their understanding from the circumstances. In a crisis situation, helping elder people or injured people is an instance of such norm. These norms mostly rely on social connections, behavioral interactions, and societal and environmental impacts. Both norms are incentives to behave in a certain way but rely on different ways to exercise their influence [33].

Since norms are not pure constraints on behaviors, there is always a possibility for them to be violated. Any system that tries to model norms should provide a mechanism for individuals to alter or violate a norm based on their understanding from the environment. Norms often arise from observing and copying the behavior of others or through interactions that are beneficial, and thus, lead to repetition of these behaviors. This is exactly what makes norms strongly crucial in any society. As individuals interact with the environment or other individuals, they incrementally learn about dominant norms in the respective normative framework.

In [33], Dignum describes two distinguished types of norms: deontic norms (that prescribe desired/required behavior) and social norms (that emerge from collective behavior). Deontic norms can be imposed either explicitly in a society (like a law) or as the effect of an order from higher authority. Norms, and especially those that are imposed from outside (deontic norms), do not automatically lead to the behavior prescribed by the norm.

Since norms arise based on interactions with the environment, they are very likely to be changed when there is a change in interaction patterns, goals, and beliefs. Also, conditions will change, which may lead to different behavior of the agents.

Culture and norms are tightly coupled in every society. Culture is defined as a set of shared attitudes, values, goals, and practices that characterizes an institution, organization, or a group. Culture both emerges from and sets the behavior of a group. In a collectivist culture an individual will look after the interest of the social group it belongs to and thus will very likely follow all social norms pertaining to that group [33].

Current modeling tools and ways of analyzing norms only consider either the forced behaviors of agents or completely free individuals. However, emergence of norms through interaction is hardly being studied. Most of the work in this area focuses on how to impose norms or force the individuals to follow (or in some cases violate) norms, but there is not enough focus on how norms emerge or can be changed in agent societies, and how individuals actively acquire social or behavioral norms in different environments.

2.5.1 Norms and Autonomous Agents

Norms in artificial intelligence and agent communities are interesting and are fairly new concepts. Different definitions of norms are used in multi-agent systems and intelligent agent communities. Three different views are considered: norms as constraints on behavior, norms as ends(goals), and norms as obligations (rules) [27]. Most research focuses on norms as constraints on behavior via social laws. These social laws are designed off-line, and agents are not allowed to deviate from the social laws [74]. The social laws are designed to avoid problems caused by interacting autonomous selfish agents, thus improving cooperation and coordination by constraining the agents' action choices.

Autonomy should be considered both at the individual level and at the multi-agent level. The most important levels of autonomy in an artificial system are norm autonomy and designer autonomy. The latter refers to the autonomy of the agents forming an agent-based system with respect to the designer(s). Such autonomy is related to the self-organizing capabilities of a multi-agent system, which may lead to emergent behaviors. Norm autonomy is the highest level of autonomy, and it refers to social impacts on agents' choices. At this level, agents choose which goals are legitimate to pursue based on a given set of norms. Such agents (called norm autonomous agents or deliberative normative agents [18][9]) may judge the legitimacy of their own and other agents' goals. Verhagen defines autonomy at this level as the agent's capability to change its norm system when a goal conflict arises, thereby changing priorities of goals, abandoning a goal, generating another goal, *etc.* F. Dignum, in [32], provides another view of autonomy at the norm level, allowing the agents to violate a norm in order to adhere to a private goal that they consider to be more profitable, including in such consideration the negative value of the repercussions such violation may have.

In another approach proposed by Briggs and Cook in [15], less restrictive sets of social norms may be chosen by agents if they are unable to find a solution based on the current social norms. This introduces a possibility for deviating from the predefined social norms; however, it is not the laws or norms that are flexible, it is the way they are applied. The laws do not change, it is the agent who decides to apply them or not. The agent is only allowed to deviate from a social law if it cannot act under the current limitations. A similar study is done in [11] where sets of norms are used by an artificial agent decision-support system (pronouncer) to reorder decision trees with the agent having the possibility to refrain from using the reordered decision tree.

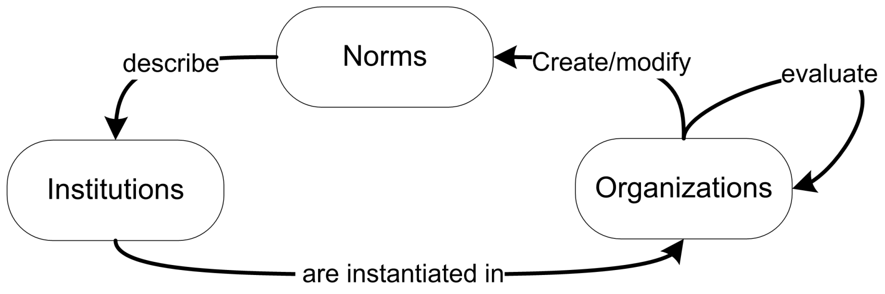


Figure 2.7: Norms, institutions, and organizations

North [52] makes a distinction between institutions and organizations. While institutions are abstract entities that define sets of constraints, organizations are instances of such abstract entities. The parties are members of an organization (not members of an institution) that should follow the institutional framework defined inside the organization. Figure 2.7 depicts the relation between norms, institutions, and organizations. In [72], it is proposed to have a link among procedures and the related norms. This allows having quite simple agents that usually follow the procedures but, whenever they want, can obtain the related norms to reason about them and make better choices. These agents are called flexible normative agents.

In [74], Verhagen views agents as having personal norms and coalition norms.

The coalition norms are subjective; therefore, every agent has an individual view on each norm of the coalition. The personal norms emerge from the interaction with the environment. The coalition norms emerge from interaction with the other agents. From the learning perspective, this can be viewed as emergence of norms (from a game-theoretic point of view) and acceptance of norms (individual level of agents) [70][28].

Studies have been done on emergence of norms based on role models that provide advice to other software agents. This helps agents to build up their social links with other agents. While researchers have studied the emergence of norms in agent populations, they typically assume access to a significant amount of global knowledge. However, in real life experiences, system experts are responsible for gathering data, evaluating the decision and actions, and setting them as norms into the system for individual agents [60].

2.5.2 Emergence of Norms

Sen et al. in [61] studied the emergence of norms in a game-theoretic approach where individual agents learn social norms by interactions with other agents. Moreover, in [51], the emergence of social norms in heterogenous agent societies have been studied. They explored the evolution of social conventions based on repeated distributed interactions between agents in a society. They considered that norms evolve as agents learn from their interactions with other agents in the society using multi-agent reinforcement learning algorithms [61].

As individual agents are able to adapt their behavior or strategy based on the interaction they have with the environment or other agents [51], the knowledge they gain throughout this process can be assessed to form their normative behaviors.

Unlike [36] that studies norm adaptation and effects of thinking in norms using computational approaches, we are interested in using the very natural way of learning used by humans. In Bayesian RL, agents are able to gather information about different environments and settings. After many experiences, this information leads to knowledge of the domain in which the agents are mostly working. Since the domain knowledge gives an understanding about typical norms and normative behaviors, it can perfectly address the issue of our bottom-up approach to generating norms. Bayesian reinforcement learning is the best way to develop automated agents able to learn normative frameworks.

We are interested in focusing on how dynamic norm generation changes the decision-making behavior of the agents by changing the agents' normative behaviors and beliefs. Meanwhile, having a mechanism to build up the agents' normative framework, we are able to capture the domain knowledge and increase agents' performance in finding solutions. Chapter 3 discusses this approach in detail.

2.6 Summary

This chapter gave a broad overview to reinforcement learning techniques and normative systems. Basic concepts in reinforcement learning such as properties of stochastic environments, model of reward, Markov property, and Markov decision processes are discussed in detail. It was shown that there are two main classes of algorithms in RL. Model-free algorithms, such as Q-learning, do not require a model of the environment and instead directly approximate a value function that measures the desirability of each environment state. Model-based algorithms estimate the value function for each state-action pair by modeling the environment. These algorithms try to estimate the reward model and the transition model in order to find an optimal policy. Bayesian inference is then described in detail. It is discussed that Bayesian RL outperforms other RL techniques in finding the best equilibrium for the exploration-exploitation problem.

Finally, the use of norms and normative frameworks in artificial intelligence was reviewed. Norms are studied in different schools of thoughts such as sociology, philosophy, economics, and artificial agent society. Most of the research in artificial intelligence has studied norms and their impacts on agent societies from the designers' point of view. It was discussed how and when norms affect the process of decision making. Emergence of norms in the social context was addressed by [61]. This approach uses reinforcement learning; however, the problem of how behavioral norms are generated in agents' beliefs and how agents maintain their beliefs about norms are still important questions to address.

Chapter 3

A Proposed RL Framework for Dynamic Norm Generation

3.1 Overview

This chapter proposes a two-level learning framework based on the ideas of reinforcement learning. This framework is based on the natural element of prior knowledge in the Bayesian reinforcement learning approach. It is shown how one can extract behavioral norms and study the interactions in every environment and later code them into agents as experiences in domain knowledge.

Initially, this chapter discusses the model-based Bayesian approach and its methodology for learning the model of the environment. Some modifications and enhancements to this algorithm are also discussed. The intention of the proposed framework is to show that *(1)* Bayesian reinforcement learning outperforms other learning methods in solving the exploration-exploitation problem, and *(2)* initializing domain-dependent knowledge as normative behaviors gained over several complete simulations into agents enhances our learning algorithm by considering the extracted norms throughout the previous experiments.

3.2 Two-level Learning Framework

This approach is looking for a framework that is able to learn the system's dynamics, specifically the environment's dynamics and interaction patterns for each setting. As it was motivated in Chapter 2, a key factor for optimizing the performance of agents is to provide them with knowledge about the dynamics of the environment and behavioral norms as well as giving them information about the social norms, interaction patterns, etc. Although social norms and behavioral norms can be generated using the very same techniques, it is not this thesis's concern to focus on the social aspects of learning. Nevertheless, social norms and their relations with behavioral norms will be mentioned throughout this thesis whenever it is helpful.

Two types of learning are considered in this framework: first, learning while the agent is exploring and exploiting rewards in each episode of the same simulation (in the same environment) and trying to learn the environment's dynamics and second, a high-level approach to capture the domain's specific normative behaviors.

For the first level, various different techniques are proposed, which were discussed in Chapter 2, to optimize the behavior of RL agents and find the optimal policy. Dearden et al. applied the Bayesian technique to the model-free Q-learning algorithm. In this method no explicit mathematical model of the agent-environment interaction is utilized. Model-free approaches are unable to maintain a model for the environment's dynamics. To preserve the model of the environment, a Bayesian exploration technique was modeled based on the decision-theoretic ideas of value of information that models the world as a Markov decision process [29].

As it is important to explore unknown states while exploiting the known states in order to get more rewards, this approach gives a relative importance to the exploration of states and balances the expected gain from exploration to find the best equilibrium for the exploration-exploitation problem. Moreover, a Bayesian dynamic programming framework was proposed in [63] that generates a hypothesis to obtain a particular example for an MDP, which could explain the set of observations made so far. This algorithm yields a proper policy for action-selection at each interaction step.

The second type of learning is a higher level approach to analyze and learn the domain's specific norms applicable to different situations. Sen and Airiau in [61] focused on the emergence of social norms through constant interaction with other homogenous agents. Interactions between agents is a useful way to understand social norms in a multi-agent environment. However, there is still a need to automatically learn the behavioral norms by studying an agent's interaction with its environment. It is interesting to study how agents maintain their beliefs about their environments and capture the behaviors that are known as norms and are able to reuse them in various situations.

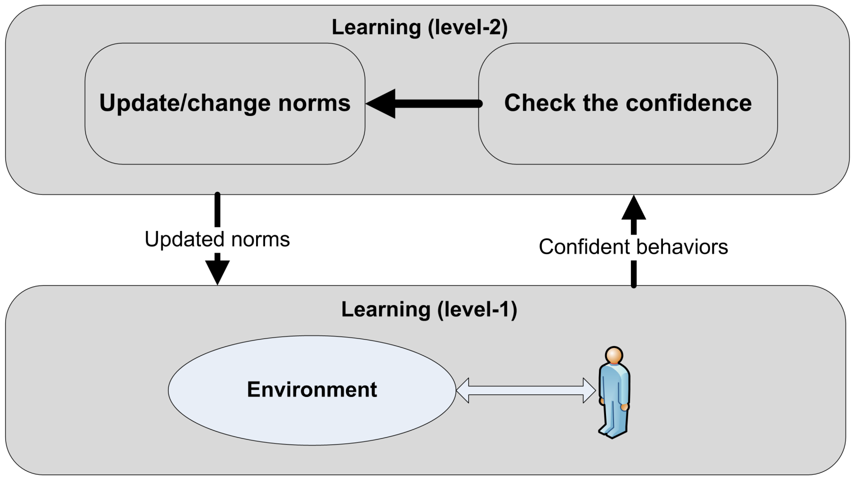


Figure 3.1: Simple sketch of the two-level learning framework

As it is shown in Figure 3.1, the first level is learning the value of each action in a certain setting/ making better decisions (Level-1). This uses the classical reinforcement learning algorithm of Bayesian dynamic programming. The second level involves learning best policies/set of norms and using them in other settings (Level-2). Behavioral norms about the environment's dynamics can be extracted using the probability distribution of each state-action pair after agents get into a reasonable confidence rate about their beliefs.

Afterwards, this knowledge gets updated and added to all the previous data gained in past experiences. The overall knowledge represents the agent's belief about the normative actions and can be incorporated into agents as prior knowledge. Having an understanding (or beliefs) about the environment, even if the beliefs are not completely accurate, helps agents to avoid random behaviors at the beginning of simulations. This approach is like showing a brief map of a city to the first responders before deploying them to a mission. This way, they all have a general understanding of the environment's dynamics such as roads, residential areas, important infrastructures, etc., although this information is like a mental blueprint.

The upcoming sections first focus on learning in a certain setting using Bayesian model learning. Then it continues to discuss norm generation using the knowledge gained through the agent's interaction with its environment.

3.3 Bayesian Model Learning

The Bayesian approach is a principled, non problem-specific approach that provides an optimal solution to the action choice problem in RL. The optimal solution to the RL action selection problem or optimal learning is the pattern of behavior that maximizes performance over the entire history of interactions of an agent with the world [30][29][21]. With Bayesian learning techniques, an agent stores a probability distribution over all possible models, in the form of a belief state [29]. The underlying (unknown) MDP, thus, induces a belief-state MDP. The transition function from belief state to belief state is defined by Bayes' rule, with the observations being the state and reward signals arising from each environmental transition.

We want to model a rescue agent trying to find possible goals in the environment to increase its cumulative reward. This agent needs to visit all the unknown states to become familiar with the reward distribution in states and then use its semi-developed knowledge of the environment to collect rewards from the already known states. Assume an agent is learning to control a stochastic environment modeled as a Markov Decision Process (MDP), which is a 4-tuple $\langle S, A, P_T, P_R \rangle$ with finite state and action sets S, A, transition dynamics P_T and reward model P_R (as described in Chapter 2). An agent's objective is to act so as to maximize the expected sum of its future discounted rewards in an infinite horizon with a discount factor $0 \leq \gamma \leq 1$:

$$E(\sum_{t=0}^{\infty} \gamma^t r_t) \tag{3.1}$$

In a stochastic and non-deterministic environment, the reward distribution and transition probabilities are not known, and so an agent must learn a policy online based on its interactions with the environment. As discussed in Section 2.4.4, model-based techniques such as policy or value iteration can be used to learn an optimal policy and its value V^{π} at each $s \in S$ [68].

At each step in the environment, the learner maintains an estimated MDP $\langle S, A, \widehat{P_T}, \widehat{P_R} \rangle$ based on an experience tuple of $\langle s, a, r, t \rangle$; that is, at each step in the environment the learner starts at state s, chooses an action a, and then observes a new state t and a reward of r. This MDP then can be solved at each stage approximately or precisely depending on an agent's familiarity with state and reward distributions. Bayesian methods [30][29] allow agents to incorporate priors to represent their beliefs over all the possible MDPs (models) that may be describing the environment and explore optimally by updating these priors as they gain more knowledge. The Bayesian framework is considered to be the appropriate framework for dealing with optimal learning, meaning that an agent maximizes its performance by setting an appropriate balance between gaining short-term and long-term rewards. A Bayesian agent models the uncertainty about the environment (discovering P_T and P_R) and takes it into account when calculating value functions. In theory, once the uncertainty is fully incorporated into the model, acting greedily with respect to these value functions is the optimal policy for the agent, the policy that will enable it to optimize its performance while learning. It is well known that Bayesian exploration is the optimal solution to the exploration-exploitation problem, meaning that there is no other method that can outperform the Bayesian solution in expectation while using the same model space and same prior knowledge [48][5].

In the Bayesian approach a belief state over the possible MDPs is maintained. A belief state defines a probability density. Bayesian methods assume some prior density P over possible dynamics D and reward distributions R, which is updated with an experience tuple $\langle s, a, r, t\rangle$. Given this experience tuple, one can compute a posterior belief state using Bayes' rule. We are looking for the posterior over reward model distribution and also the posterior for the transition model, given an observed history of H. Considering H to be the state-action history of the observer, an agent can compute the posterior $P(T, R|H)$ to determine an appropriate action at each stage. As the density P is the product of two other densities $P(T^{s,a})$ and $P(R^{s,a})$, that is, the probability density of choosing action a in state s and the probability density of getting the reward of r by choosing an action a when in state s, we should make an assumption to simplify this calculation.

Based on [29], our prior satisfies parameter independence, and thus the prior distribution over the parameters of each local probability term in the MDP is independent of the prior over the others. This means that the density P is factored over R and T with $P(T|R)$ being the product of independent local densities $P(T^{s,a})$ and $P(R^{s,a})$ for each transition and each reward distribution. It turns out that this form is maintained as we incorporate evidence. The learning agent uses the formulation of [29] to update these estimates using Bayes' rule:

$$P(T^{s,a}|H^{s,a}) = zP(H^{s,a}|T^{s,a})P(T^{s,a}), \tag{3.2}$$

$$P(R^{s,a}|H^{s,a}) = zP(H^{s,a}|R^{s,a})P(R^{s,a}), \tag{3.3}$$

where $H^{s,a}$ is the history of taking action a in state s, and z is a normalizing constant.

It has been assumed that each density $P(T^{s,a})$ and $P(R^{s,a})$ is a Dirichlet [31] as the transition and reward models are *discrete multinomials*. These priors are conjugate[1], and thus the posterior after each observed experience tuple will also be a Dirichlet distribution [29][21].

In probability and statistics, the Dirichlet distribution is of a family of multivariate probability distributions. It is, in fact, the multivariate generalization of the beta distribution, and conjugate prior of the categorical distribution and multinomial distribution in Bayesian statistics. That is, its probability density function returns the belief that the probabilities of k rival events are x_i given that each event has been observed $\alpha_i - 1$ times. We take this as an opportunity to enhance our Bayesian method.

Dirichlet distributions consider the number of actions to be done in every state as a multivariate probability distribution. This makes the implementation and tracking of the algorithm quite hard because the transition model will be sparse, with only a few states that can result from a particular action at a particular state. If the state space is large, learning with a Dirichlet prior can require many examples to recognize that most possible states are

[1]If the posterior distributions $p(a|x)$ are in the same family as the prior probability distribution $p(a)$, the prior and posterior are then called conjugate distributions, and the prior is called a conjugate prior for the likelihood. For example, the Gaussian family is conjugate to itself (or self-conjugate) with respect to a Gaussian likelihood function: if the likelihood function is Gaussian, choosing a Gaussian prior over the mean will ensure that the posterior distribution is also Gaussian [57].

highly unlikely [29][63].

To overcome this problem, we suggest using beta distributions for every state and action. As described in [6], when solving a problem using model-based methods, it is always useful to consider state-action pairs when running the value iteration algorithm to compute the Q-values. In Bayesian statistics, it can be seen as the posterior distribution of the parameter p of a binomial distribution after observing $\alpha - 1$ independent events with probability p and $\beta - 1$ with probability $1 - p$, if there is no other information regarding the distribution of p.

We consider a binomial probability distribution for every state-action pair. These distributions actually show us the number of times in which every state-action pair succeeds or fails during the simulation. We need to maintain the number of times, $N(s \xrightarrow{a} s')$, state s is successful to make transition to s' when action a is chosen, and similarly, $N(s \xrightarrow{a} r)$ for rewards. With the prior distributions over the parameters of the MDP, these counts define a posterior distribution over MDPs.

To model $P(T^{s,a})$, a binomial likelihood of a distribution is used. Given this probability distribution model, one of the parameters we want to estimate using our data is the probability of success for a given question, where success (denoted as p) can be defined as the probability that state-action is chosen; therefore, it is a success. The letter q can be used to denote “failure” and has a probability value equal to $1 - p$.

The maximum likelihood estimate of the transition probability $T(s \xrightarrow{a} s')$ is the proportion of times that action a in state s led to s'. The maximum likelihood estimate of $E[R(s, a)]$ is the average of the rewards received when action a was taken in state s. Hence, dynamic programming provides a solution to the reinforcement learning problem without the need for a learning rate (as opposed to the Q-learning approach).

A drawback to the dynamic programming approach is that it requires an assumption that the underlying reward distributions and transition probabilities are statistically stationary. The implication of this assumption is that retraining the system is required whenever the environment changes significantly [63]. We will discuss non-stationary environments later in this chapter and discuss the fact that learning domain-dependent norms can significantly help agents to act in non-stationary environments, due to the fact that they can extract general guidelines and norms for their actions and use them in different settings under the same domain.

3.3.1 Value of Information Exploration

To capture the uncertainty about the model, the estimation of a distribution over MDPs is used at each stage. Then to improve exploration one can use this knowledge of uncertainty. As discussed earlier, one important question to address is to find a balance between the expected gain from exploration, resulting in more information about state space and thus having more improved policies, and the expected cost of doing a potentially suboptimal action. To compute this value of new information, a technique based on the decision-theoretic ideas of *value of perfect information* [42] was proposed by [29][30]. This exploration method is known as VPI exploration.

Let $q_{s,a}$ be a possible value of $Q^*(s, a)$ in one of the possible MDPs that is a random variable depending on an agent's belief state. If this knowledge does not change the agent's policy, then future rewards would not change, and it has no interest for us. However, when the new knowledge changes the agent's policy, learning the true value $q^*_{s,a}$ of $q_{s,a}$ would have some positive gain for the agent.

The value of perfect information gives an upper bound on the myopic value of information for exploring action a [29]. The expected cost for this exploration is the difference between the value of action a and the value of the current best action. Therefore, the agent should always choose the action that maximizes this addition:

$$E[q_{s,a} + VPI(s, a)]. \tag{3.4}$$

where $VPI(s, a)$ is the expected value of perfect information when the agent does not know about the value of $q^*_{s,a}$ and needs to compute the expected gain given our prior beliefs. When the agent is confident in the estimated Q-values, the VPI of each action is close to zero, causing the agent to always choose the action with the highest expected value.

Suppose that, given the agent's current belief state, a_1 is the action with the highest expected Q-value at state s and a_2 is the second-highest. The gain associated with learning the true value of $q_{s,a}$ is:

$$Gain_{s,a}(q^*_{s,a}) = \begin{cases} E[q_{s,a_2}] - q^*_{s,a} & \text{if } a = a_1 \text{ and } q^*_{s,a} < E[q_{s,a_2}] \\ q^*_{s,a} - E[q_{s,a_1}] & \text{if } a \neq a_1 \text{ and } q^*_{s,a} > E[q_{s,a_1}] \\ 0 & \text{otherwise} \end{cases} \tag{3.5}$$

In the first and the second case, the new information affects the decision process and changes the agent's policy. For the first case, suppose that a_1 is the action with the highest expected value and a_2 is the second best action. The new knowledge indicates that the expected value of taking this action $E[q_{s,a_2}]$ is bigger than q_{s,a_1}; thus, action a_2 should be performed resulting in the expected gain of $E[q_{s,a_2}] - q^*_{s,a}$. For the second case, the new information shows that even though a_1 was supposed to be the best action (for all actions a'), action a (which was considered sub-optimal) is a better action to take compared to the previous best action a_1. Thus, the amount an agent gains from learning this new knowledge and changing the action is $q^*_{s,a} - E[q_{s,a_1}]$. Obviously, if the information has no impact on the decision at s, it is considered to have zero gain.

3.3.2 Estimating Q-value Distributions

There are many different ways to estimate the Q-value distribution such as *naive sampling*, *importance sampling*, *global sampling with repair*, and *local sampling*. One simple solution proposed by [29] to this problem is *naive sampling*. This approach samples k MDPs from the density describing the agent's belief state. For each state-action pair (s, a) there exists k solutions $q^1_{s,a}, ..., q^k_{s,a}$. Every MDP can be easily solved using the value iteration algorithm. The mean Q-value is estimated as follows:

$$E[q_{s,a}] \approx \frac{1}{\sum_i w^i} \sum_i w^i q^i_{s,a} \tag{3.6}$$

Similarly, the VPI can be estimated by summing over the MDPs:

$$VPI(s, a) \approx \frac{1}{\sum_i w^i} \sum_i w^i Gain_{s,a} q^i_{s,a} \tag{3.7}$$

where w^i denotes the weight of each sample (for simplicity all weights can be set to 1), and $q^i_{s,a}$ is the optimal Q-value given the ith MDP. Since our prior satisfies parameter independence as discussed in Section 3.3, then we can sample each distribution independently of the rest. Thus, the sampling problem reduces to sampling from simple posterior distributions.

3.3.3 Complexity

By considering state-action pairs in binomial distributions, we keep the same complexity in solving the learning problem. If $|S|$ is the number of states in our state space, and $|A|$ is the number of permissible actions in each state, the number of state-action pairs that should be maintained using binomial distributions are $|S||A|$. The same complexity is assumed when considering a Dirichlet distribution. The computational complexity of the value iteration algorithm, per iteration, is linear in the number of actions and quadratic in the number of states ($O(|A||S|^2)$). However, the number of iterations required can grow exponentially in the discount factor because, as the discount factor approaches 1, the decisions must be based on results that happen further and further into the future [47] [19].

3.4 Dynamic Norm Generation

3.4.1 Introduction

As discussed in Chapter 2, there are many different classifications defined in the AI literature for norms and normative frameworks. However, two major categories (which include all the other classifications) are more interesting. In every community of individual agents, norms can be considered either in social or individual levels. Social norms mostly refer to the norms between agents in a society. They define the normative behavior of agents when interacting with other agents or communicating with other intelligent entities. In any multi-agent setting, social norms effectively impact the behavior of agents when interacting with other agents. Norms such as trust, reputation, social acceptance, etc., are examples of social norms.

On the other hand, behavioral norms mostly affect individual agents' beliefs. It is the way an agent learns to behave and act in an environment without considering the communications with other agents (communication is considered to be active and aware interactions between agents; anything besides that is considered to be part of the environment). These norms are usually developed based on the perceptions that an individual has when interacting with its environment. These norms mostly affect the decision making of agents and their beliefs toward the circumstances. Behavioral norms are domain-dependent and context-sensitive norms, meaning that in every situation based on the signals one perceives from its environment, these norms can be changed or altered.

Figure 3.2 shows the two classes of social and behavioral norms and their respective interaction patterns.

The need for effective norms to control an agent's behavior is well-recognized in multi-agent and intelligent-agent societies [10],[73]. Norms and normative actions drive the behavior of agents and their ways of decision making. Thus, norms are key issues in optimizing the behavior of agents. Most of the

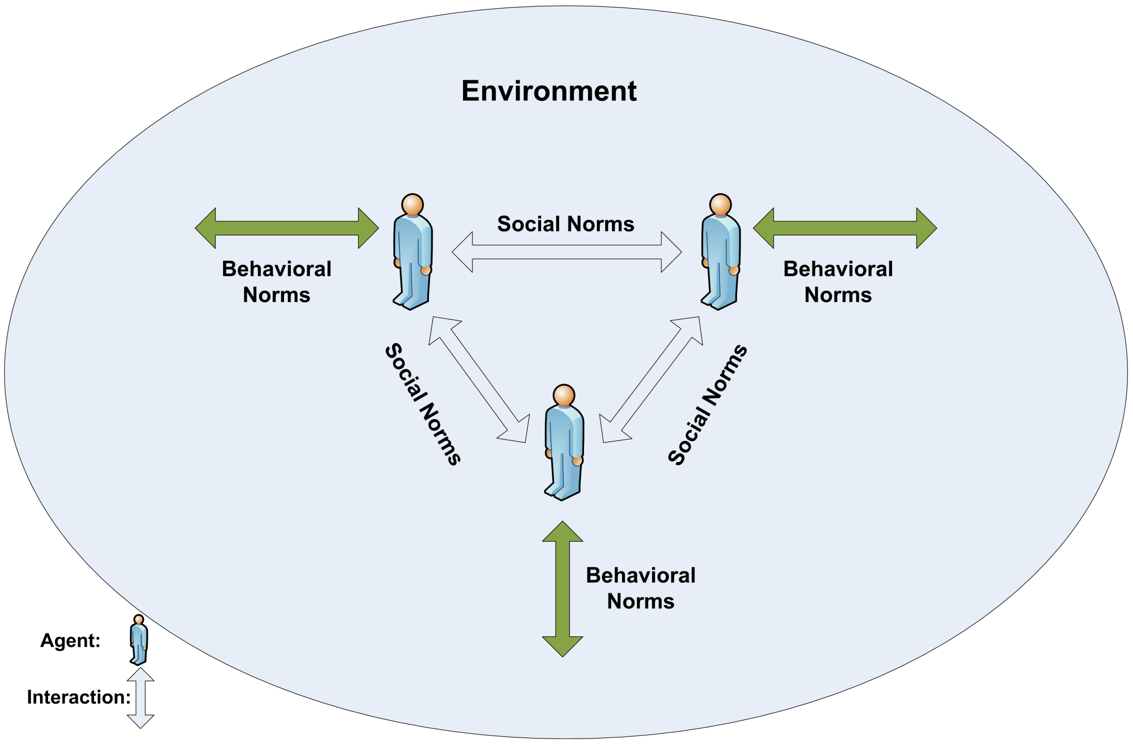


Figure 3.2: Social norms and behavioral norms

research in artificial intelligence on norms has focused deeply on logic or rule-based specification and the enforcement of norms [73]. Other works with a game-theoretic approach also concentrate on norm derivation and enforcement. They lack the ability to describe the ways norms and normative frameworks are established and assume a centralized authority that defines the rules and dictates norms to the agents.

Conventional approaches to normative frameworks assume that every system has a predefined set of rules and norms that are static. It is the system experts' responsibility to extract the domain-dependent norms by carefully

Conventional Methods	Dynamic Norm Generation
Top-down	Bottom-up
Predefined rules/norms	Self-organized norms
Set by system experts	Set by individual agents
Static	Dynamic
Offline	Online
Testing, updating: time consuming	Testing, updating: constantly via interaction

Table 3.1: Conventional methods vs. Dynamic Norm Generation

studying the system's dynamics, applying statistical analysis, considering permissible actions and level of control, and defining roles in agent societies. Also, several experiments and training should be done to evaluate and test these extracted norms. This is a time-consuming process that may or may not lead to a robust framework. Moreover, the offline nature of norm creation prevents any proactivity and flexibility on the side of agents.

In reality, norms often evolve in a bottom-up manner. Agents have a quite useful understanding of the environments, roles and goals, possible actions, and policies as they are constantly interacting with the environment and other homogenous (or heterogenous) agents. Since agents have the best view of the problems and are actually in action, they can easily obtain information about the domain where they are active. This way, norms can be established in their natural form and can be updated or even altered without the heavy expenses of top-down decision-making procedures. Norms are being created on the fly and can be dynamically changed depending on the environment's dynamics.

3.4.2 Generating Behavioral Norms

Every domain has its specific set of norms (known as behavioral norms) that can be generally valid in other environments. As discussed earlier, these norms arise from the agent-environment interaction and usually guide the behavior of agents. There is a mutual connection between behavioral norms and domain-dependent knowledge in reinforcement learning. Norms can be extracted through reinforcement learning (RL), and RL can be improved by incorporating behavioral norms as prior probability distributions into learning agents.

Section 3.3 discussed that the Bayesian approach allows the natural incorporation of prior knowledge as a prior probability distribution over all possible MDPs; also, approximations to optimal Bayesian exploration can take advantage of this model, enabling the mass of the posterior to become progressively focused on those MDPs in which the observed experience tuples are most probable [29]. The distribution maintained over possible MDPs and the Q-values of each of these MDPs, induce a distribution over the Q-values at each state-action pair. The Q-value distribution is then used for action selection. This can be done by incorporating the actions' value of perfect information [42],[29] in the agent's exploratory policy.

Agents gain knowledge about the environment's dynamics using dynamic programming iterations and updates. By visiting every state or choosing actions, agents gradually build up their knowledge about the environment as probability distributions over state-action pairs. This information can be considered to be incomplete or false during the simulation until agents are confident about their beliefs. From the exploration-exploitation perspective, this confidence is gained when the agent has knowledge about most of the states and the permissible actions in them or the value of each action in every state. Thus, agents are said to be confident on their beliefs when (1) The algorithm has converged into an optimal policy (or cumulative reward becomes steady in the recent episodes), and (2) Most of the states have been visited by the agent.

In the first condition, it is not really easy to understand whether an algorithm will converge to an optimal policy or not. It needs complicated and time-consuming mathematical calculations. Bayesian dynamic programming is proved to converge [7]. As it is complicated to check this criterion, another approach will be used.

I introduce an element to check at the end of each episode. When an episode is finished, the goal state is reached, and we are able to look at the cumulative reward gained in that episode by our agent. If this cumulative reward is in a steady state in recent episodes, it is a good measure to be sure that our Bayesian algorithm is in a reliable state, meaning that the algorithm is in equilibrium.

The amount of cumulative reward or the number of steps to the goal is not solely a good metric to measure the level of confidence. What also is important for agents is to make sure that they have at least some sort of sufficient information about the world and the majority of states. This can be measured by counting the number of explored states so far, indicating how many states have been visited by an agent.

The level of exploration (LOE) is defined simply as follows:

$$LOE = \frac{E}{N} \tag{3.8}$$

where E is the number of explored states so far in the simulation, and N is the total number of states. LOE is always smaller than or equal to 1. The closer it gets to 1, more states of the environment have been explored.

It is proposed that the agent can be confident about its beliefs when $LOE \geqslant 0.9$ and CR_n satisfies equation 3.9.

$$CR_n = [\sum_{i=n-k}^{n-1} CR_i/k] \pm (1 - LOE + \epsilon) \tag{3.9}$$

where CR_n is the cumulative reward gained in the n_{th} episode, and k is a desired number of recent episodes. Based on every experiment and the size of the state-space, one can decide to consider k previous cumulative rewards to average them (In this thesis $k = 5$).

The cumulative reward gained in each episode can be different even after converging to the optimal policy, as the agent is always in the learning process and may explore some other states. Therefore, the value of CR_n should fall into a plus/minus interval to be acceptable. This interval depends on the value of LOE. If not many of the states have been explored so far, the interval gets larger. The cumulative rewards become closer and closer to each other when the majority of states have been covered. In a nutshell, the more states that have been explored by an agent, the smaller the interval gets. Although LOE rarely reaches 1, the ϵ in this formula makes sure that there is always an interval even when $1 - LOE$ is equal to 0.

When an agent meets these two conditions and becomes confident about its information on normative behaviors, it should simply update its belief state and add this newly learned knowledge to its knowledge base.

3.4.3 Updating Prior Knowledge as Norms

An agent's knowledge is considered to be probability distributions over state-action pairs, which are represented in the form of prior probabilities. By the ending of each episode, if an agent is confident about its knowledge gained so far, it should update its beliefs by adding the new information to its belief. When working with Bayesian inference, every density for transition and reward (i.e., $P(T^{s,a})$ and $P(R^{s,a})$) is considered to be in the form of a beta distribution[2], which is a discrete binomial. Updating these priors is

[2] The beta distribution is a family of continuous probability distributions defined on the interval (0, 1) parameterized by two positive shape parameters, typically denoted by a and b. It is the special case of the Dirichlet distribution with only two parameters.

very simple and easy as they are conjugate, and thus the posterior after each observed experience tuple is also in the form of a beta distribution.

Updating the Bayes parameter estimate with new information is easy by using the concept of a conjugate prior. The parameter estimate obtained from the previous episodes should be combined with the estimates an agent already has about its states and actions. Essentially, a conjugate prior allows agents to represent the Bayes parameter estimation formula in simple terms using the beta parameters a and b:

$$\begin{aligned} a_{posterior} &= a_{prior} + a_{data} \\ b_{posterior} &= b_{prior} + b_{data} \end{aligned} \tag{3.10}$$

This thesis considers state-action pairs as binomial probability distributions showing us the number of times each state-action succeeds or fails. The beta parameters in beta distributions are the number of successes and the number of failures. The posterior is simply given by adding the prior parameter and data parameter (the number of successful transitions from state s to s' under a).

Updating norms is exactly the same as updating posteriors. Agents are continuously building and updating their posteriors using the aforementioned process. As this information is obtained by agents interacting in the environment (to solve a problem or to pursue a goal), it is representative of the environment's dynamics and norms. When an agent is in a confident level about its knowledge, it keeps a copy of the reward and transition models and then updates its posterior by replacing the posterior gained so far with the prior distribution of tested data (data parameter).

This way, whenever an agent is confident in an environment about its norms and the environment's dynamics, it automatically captures this information and uses it as domain knowledge or domain-dependent behavioral norms. Testing and changing norms is a trivial task due to the fact that agents are constantly updating and altering their beliefs about behavioral norms.

3.4.4 Representing Norms

Although generating norms as domain knowledge using reinforcement learning leaves no need for representing norms in a logical form, one might want to represent results in such form. We use an example to show such analysis to represent norms using descriptive logic:

Let's consider an example of first responders in a crisis situation. First responders such as firefighters, police officers, etc., are members of their respective organizations who should follow orders from higher authorities in their own organizations. If a crisis such as flooding or an earthquake happens, individual agents try to find victims, apply basic medical care, and help people by taking them to safe places. The condition of the environment is not known to the agents; however, they can learn by the interaction they have with the environment and then decide to take an appropriate action. If a road is blocked, they tend to explore the zone behind the blocked area to find injured people. The agents learn this behavioral norm by dynamically generating it through many simulations, training, and real-life incidents.

Chapter 4 describes the experiments and some analysis based on a grid-world scenario. The following helps to better associate our experiment with the aforementioned example:

An agent's goal is to find a policy that maximizes the cumulative rewards while learning the world's dynamics. The agent is trying to find a state with a positive reward (people who need help) by exploring unknown states. After it finds a state with some amount of positive reward, it still explores the other poorly known states in order to find more rewards (and eventually increase its cumulative reward). For instance, if the agent finds more rewards close to the blocked states during a simulation and is confident about this knowledge, it adds this information to its belief state. The agent preserves the probability distribution of each state-action pair and updates them as described in the previous section. When this agent is set to accomplish a task in another setting, the information about actions in each state help him to push the mass of posterior in a way that makes him explore the parts closer to the blocked states.

3.4.5 Behavioral Norms and Organizational Norms

There is a tight connection between the methods used to generate norms in organizations and behavioral norms. Organizations set predefined policies as norms for members to follow in different situations. These norms (which assign a predefined action to every state) are usually extracted by system experts and experienced members of organizations with higher rankings. After many simulations and training sessions and also analyzing the observed behavior of decision making in real practices, a set of norms is set to be generally defined by expert members [72] [28]. As this process is usually done in every organization manually, there is usually a chance of misinterpretation or faults in setting these organizational norms.

Automating the process of generating and validating organizational norms can be done using similar learning methods used to extract behavioral norms. For simplicity, every organization is considered to be a set of decision nodes with permissible actions with a probability. Taking each action in a decision node will get the overall organization into another state. This way, one can use the dynamic norm generation framework and extract norms in an automated process. The generated set of norms is shown to be the most optimized policy for the organization.

Moreover, one can abstract each organization and treat them as individual agents. The relation between all the organizations can be considered as an open multi-agent system where there are several heterogeneous agents with different goals, and sometimes these goals may interfere or may cause them to compete with other agents or organizations.

3.4.6 Sensitivity Analysis

Bayesian analysis is perhaps best seen as a process for obtaining posterior distributions or predictions based on a range of assumptions about both prior distributions and likelihoods: arguing in this way, sensitivity analysis and reasoned justification for both prior and likelihood become vital.

When computing the posteriors and other probability distributions for state-action pairs, there is a chance of getting into errors. This makes it reasonable to have a mechanism to take the error density out of the results. In this thesis, we have used the averaging system to make sure that the results are consistent. Every example is run many times with exactly the same elements, and then the average of the results is considered to be the final solution. This approach is not a precise and robust technique; however, it helps our results to have a better and more reliable form.

3.4.7 Norms and Culture

Culture can be seen as an expression of a certain set of values common to a group or society [34]. Thus, there is a connection between culture and behavior in every society. Agents learn social or behavioral norms by interacting with the environment or other agents. This leads them to integrate into new societies and worlds. Individuals try to learn norms in order to understand the whole normative system. This normative system can be seen as culture, as it is usually the dominant behavior of individuals in a society or the main interactions between the members of an organization. On the other hand, culture is established by the aggregation of social and behavioral norms in a normative framework. This explicitly shows the mutual connection between norms and culture[3].

3.5 Summary

This chapter explained the main ideas in our approach to dynamic norm generation using Bayesian reinforcement learning. A two-level reinforcement learning framework that uses the ideas and concepts in Bayesian RL was proposed. First-level learning refers to learning a certain environment with specific dynamics. The second level uses the results from the first-level learning in different environments and generalizes this knowledge to set them as domain-dependent behavioral norms. Bayesian RL is a systematic way to incorporate prior knowledge about the domain into the learning iterations.

Generating norms in an autonomous way using learning agents was shown to have many advantages over the conventional methods of norm assign-

[3]Chapter 5 draws an interesting line of research as a future work on cultures and norms.

ments. Better interactions, better decision-making processes, automated updates and changes, proactive flexible agents, and easier analysis of norms and domain knowledge are some of the reasons to use this framework. Moreover, it helps to better understand the ways our beliefs about behavioral norms and our environment's dynamics become established.

Chapter 4

Experimental Results and Analysis

4.1 Overview

This chapter evaluates the ideas proposed in this thesis on the introduced two-level learning framework for dynamic norm generation. A series of experiments are shown to test the performance and the effectiveness of these approaches. The first section provides an introduction to these experiments and defines some of the terms repeatedly used in this chapter. The second section gives a broad comparison between three different reinforcement learning techniques. Finally, the third section sets up three different experiments to fully explore the performance of the proposed framework where the agent is asked to adapt its beliefs in a world with different dynamics and different norms.

4.2 Introduction

This chapter uses a simple maze problem to experiment with the proposed ideas on behavioral norms and also to show how these norms can affect the performance of learning agents. Although real-world problems of norm generation are much more complicated, representing the world and its dynamics in a simple way can help us show a proof of concept. Furthermore, every decision-making situation where a learning agent needs to take an action under uncertainty can be easily mapped into a belief-state MDP. Then, using the proposed techniques, an agent will be able to solve the MDP, learn the model of the environment, and generate norms if the confidence level is reached.

In the upcoming examples, one can think of behavioral norms as a series of actions an agent takes in every state based on its discretion and its previously gained beliefs. The agent should take an action based on the learned norms that are believed to get the agent closer to accomplish its goals, although this action is not necessarily the optimized action. If the chosen action is found to be not exactly aligned with the agent's intentions, the agent updates its normative system in order to adjust its beliefs to the new dynamics.

To experiment with the ideas and advantages of the framework proposed in this thesis, I choose a world represented in the form of a grid world. The same techniques and ideas can be applied to any problem involving learning agents and decision making. The implementation framework that is used to code these ideas is the one developed by Sutton in the RLAI lab[1]. This framework provides the basic tools to implement any desired algorithm.

Figure 4.1 shows the maze problem. The agent can move left, right, up, or down by one square in the maze. If it attempts to move into a wall, its action has no effect. The problem is to find a navigation path from the start state ('S') to the goal state ('G') with the fewest possible steps and the highest cumulative reward. The agent also should gather as much information as possible about the environment and its dynamics. When it reaches the goal, the agent receives a reward equal to 1, and the problem is then reset. Any step has a small negative reward of -0.05. The agent's goal is to find the optimal policy that maximizes its cumulative reward. A policy is a series of actions that an agent takes in different states to reach its goal. The problem is made more difficult by assuming that the agent occasionally "slips" and moves in a direction perpendicular to the desired direction (with probability 0.1 in each perpendicular direction). The world's state is not observable, so it must be inferred from the history of sensory input and action.

Throughout this section, some terms will be used that are defined below:

[1]http://rlai.cs.ualberta.ca/

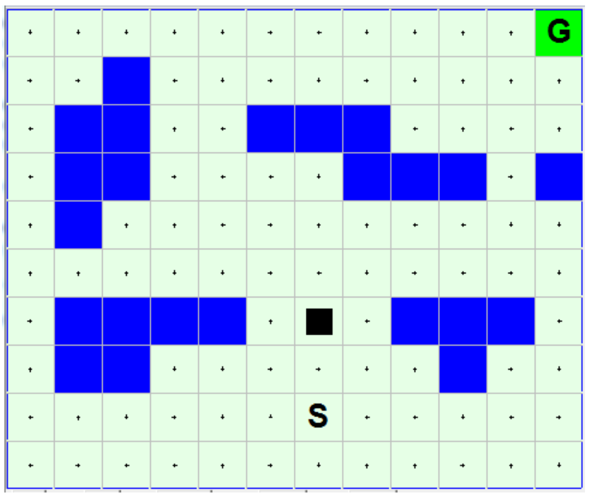


Figure 4.1: A small grid world

- Episode: every trial or iteration that starts from the start state and ends at the goal state,
- Steps: the number of steps an agent takes to find the goal state in each episode,
- Explored states: the number of explored states throughout the simulation in all the episodes so far,
- Cumulative reward: the amount of reward that an agent receives in each episode by trying a policy.

4.3 Comparison Between Different RL Techniques

This section provides some comparisons between different reinforcement learning (RL) techniques in environments with a small number of states and environments with many states. The advantages and disadvantages of each technique will be discussed later in this section.

4.3.1 Small State-Space

As shown in figure 4.2, there are 98 reachable locations in the maze (including the goal and the start state) and there are 4 permissible actions in each state. The agent was given no layout information in the beginning of simulation and has no knowledge about the domain. Three standard algorithms were simulated and their learning performances were compared:

1. Q-learning algorithm
2. Dyna architecture
3. Bayesian DP with VPI

The experiment computes averages for cumulative rewards and the explored states to evaluate these algorithms. This is to ensure that the results are consistent and did not happen by chance (results are not coincidental).

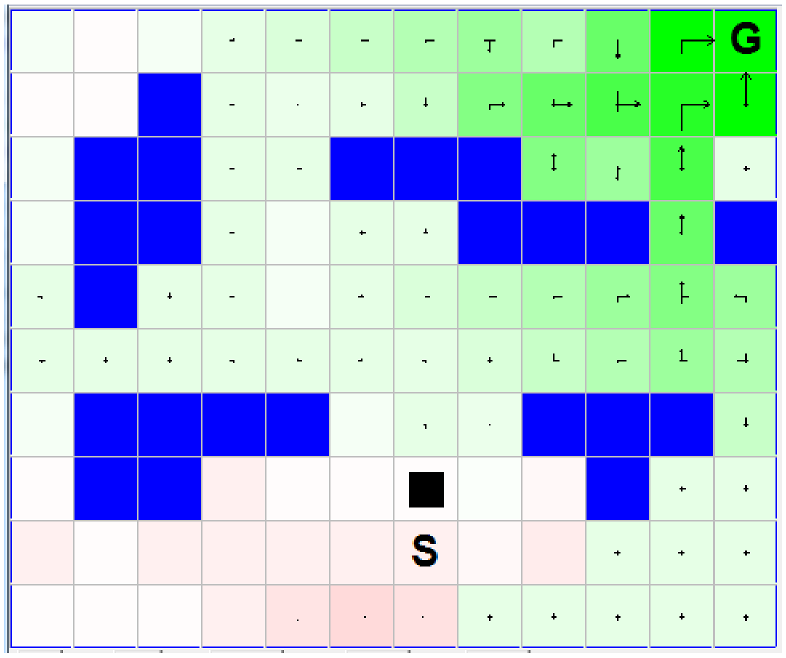


Figure 4.2: Experiment 1- a map of a 12x10 grid world

This experiment evaluates the algorithms by computing the average over 10 runs for each of the results. The experimental results over 50 episodes show differences in cumulative reward, the number of explored states, and the number of steps to the goal state. The agent is trying to find an optimal policy in each state while learning the environment's dynamics. The figures below show the comparison of these factors.

As demonstrated by the figures, the Bayesian dynamic programming algorithm with VPI has slightly better results compared to the Dyna approach. Both of these approaches outperform the model-free approach of Q-learning with a significant difference. This is due to the fact that Q-learning does not try to learn the model of the environment.

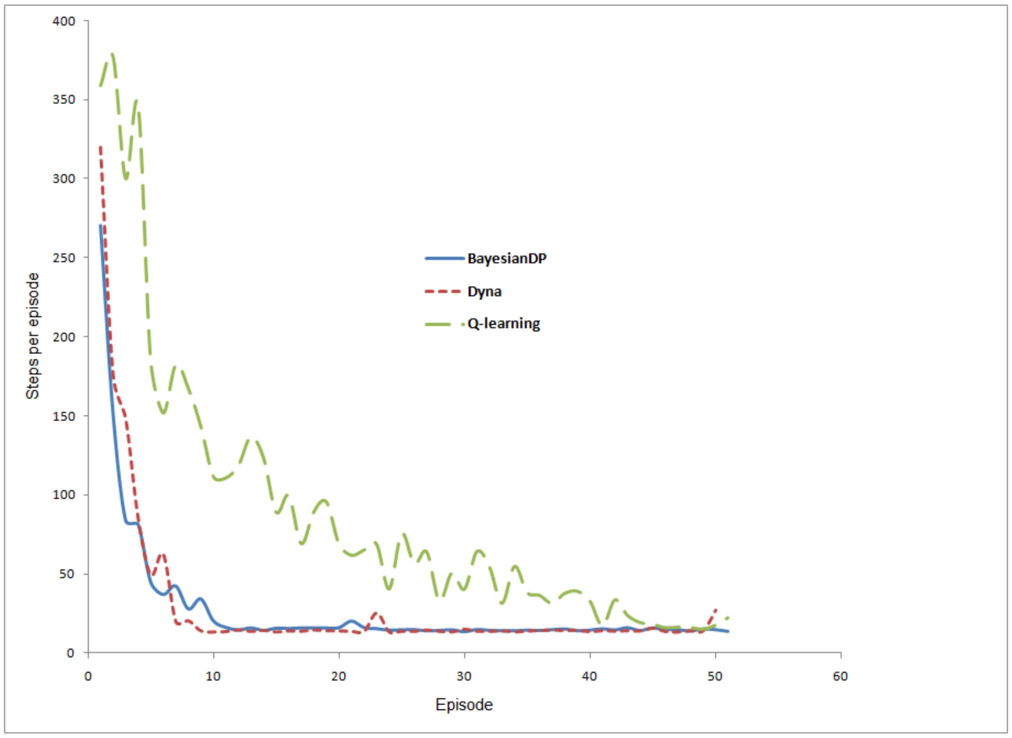


Figure 4.3: A comparison between the number of steps to the goal for Bayesian, Dyna, and Q-learning agents (average over 10 runs)

The number of steps to the goal state converges rapidly in the beginning after about 10 episodes for the Bayesian and Dyna agent; however, the Bayesian agent finds the best policy, to a small extent, faster than the Dyna agent. The Q-learning agent starts with a higher number of steps, and it finds the best policy very slowly.

In the small grid world, all three techniques perform similarly in exploring states; however, early in the learning process the Q-learning agent has a higher exploration rate. Because of the trial-and-error nature of this algorithm, the agent has no planning strategy and randomly visits states. The

Bayesian agent, on the other hand, intentionally is looking to explore the world while trying to find the optimal policy. In other settings, if the start state and the goal state are set to be close to each other, the Q-learning agent will dramatically underperform in exploring the world.

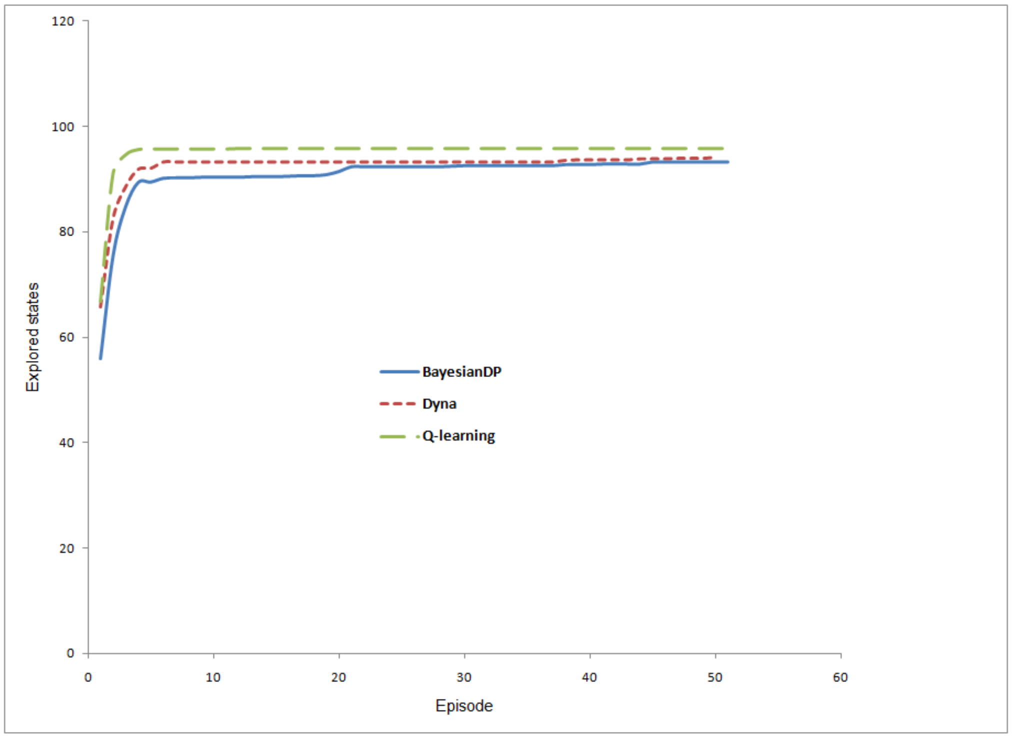


Figure 4.4: A comparison between the number of explored states: Bayesian, Dyna, and Q-learning agents (average over 10 runs)

Figure 4.5 shows the cumulative reward gained by the agents in 50 episodes. As the graph shows, the Bayesian technique performs better compared to all the other ones in terms of cumulative rewards gained over each episode. Although Dyna has a very good performance, the Bayesian agent starts to learn the model of the environment faster, and hence it has slightly better results in general.

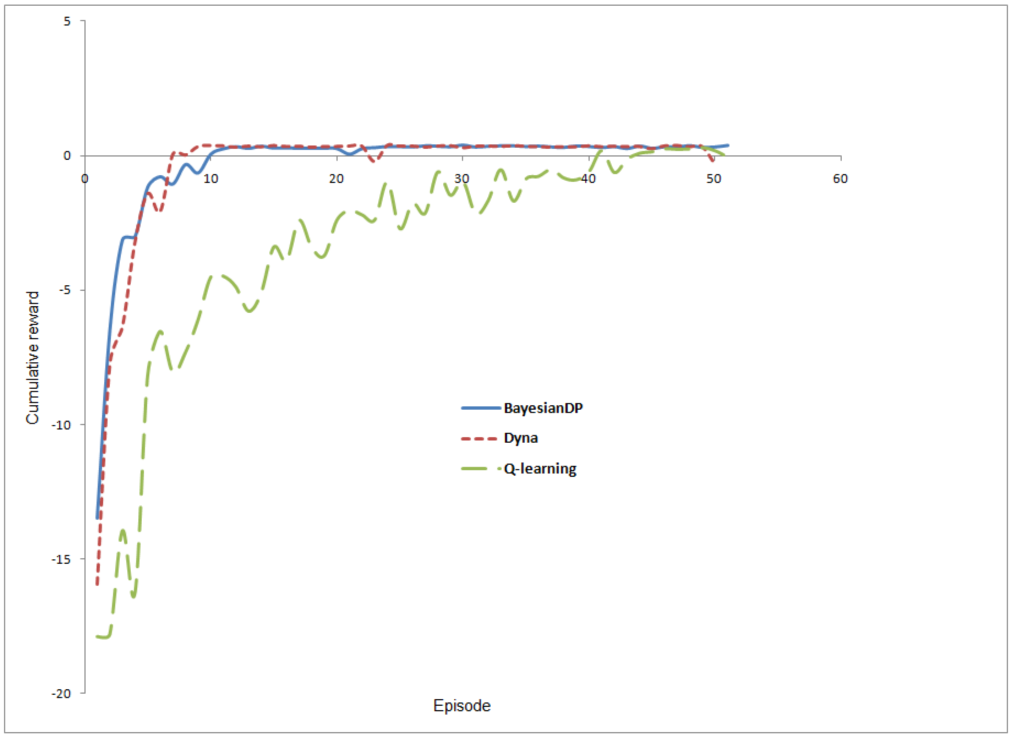


Figure 4.5: A comparison for the cumulative rewards: Bayesian, Dyna, and Q-learning agents (average over 10 runs)

4.3.2 Large State-Space

For this experiment, a larger grid-world maze is considered to test the three techniques. State space and its complexity plays an important role in the performance of the agents, and so I decided to study the performance of each algorithm in a larger state space. As shown in figure 4.6, this scenario is in a 20x20 scale maze with 365 reachable states. As with the previous experiment, the agent can choose 4 permissible actions in each state.

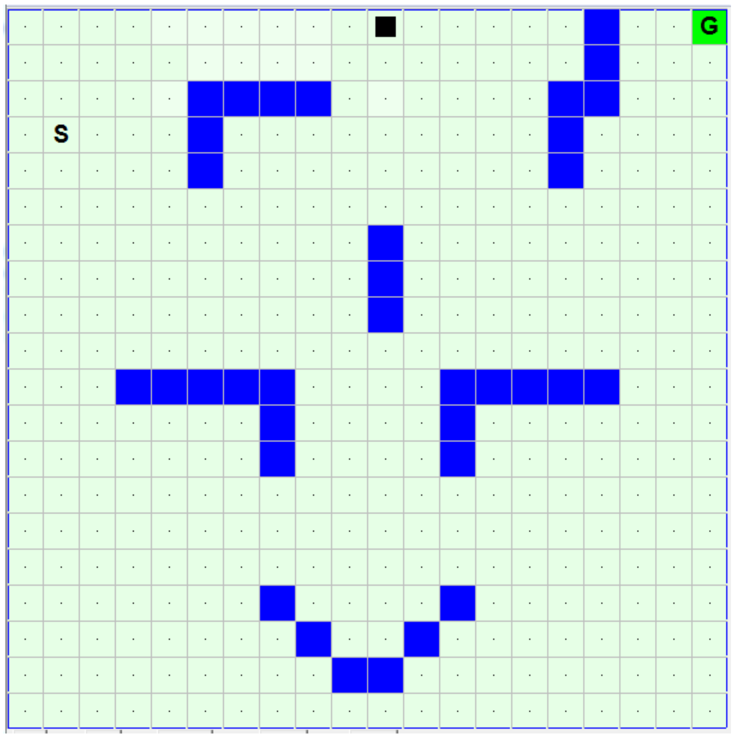


Figure 4.6: Experiment 2: A map of a 20x20 grid world

Figure 4.7 shows that the number of steps to the goal state converges rapidly in the beginning after about 10 episodes for the Bayesian and Dyna agent; however, the Bayesian agent has a better performance in finding the best policy. This is due to its ability to learn the model of the environment and make some hypotheses based on the Bayesian inference. The Q-learning agent starts with a higher number of steps. After 50 episodes, it is still unable to reach an optimal policy. It finds the best policy after 90 episodes (The experiment has been tested for 100 iterations; however, the figure does not show more than 50 episodes).

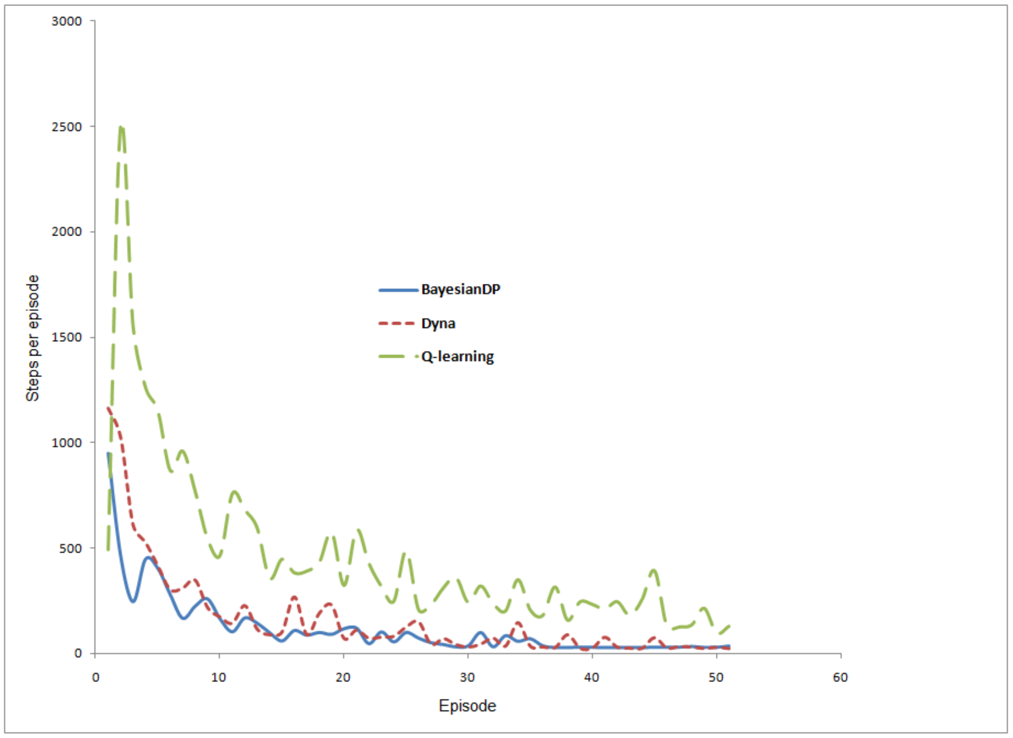


Figure 4.7: A comparison between the number of steps to the goal for Bayesian, Dyna, and Q-learning agents (average over 10 runs)

As in the previous experiment, Figure 4.8 shows that the Q-learning agent is better in exploring most states; however, this is due to the random nature of Q-learning, so it cannot be considered as a "smart" action.

Figure 4.9 shows the cumulative reward for the three techniques. The Bayesian technique surpasses other techniques throughout the learning process. Dyna has a reasonable rate of convergence and performs much better than Q-learning. However, it has many small fluctuations overall. The performance of the Bayesian agent is better than the performance of Q-learning (or Dyna) as it converges faster to an optimal policy. Moreover, a paired t-test demon-

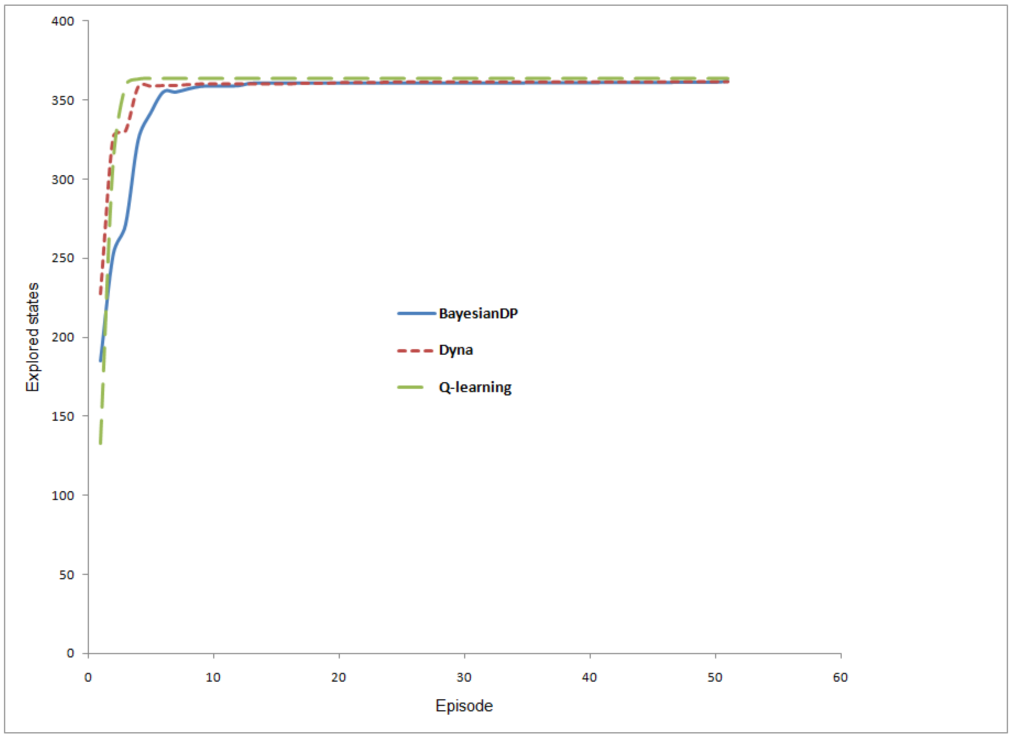


Figure 4.8: A comparison between the number of explored states: Bayesian, Dyna, and Q-learning agents (average over 10 runs)

strates that the difference in means between Bayesian DP and Q-learning is statistically significant with p = 1.88521E-06 (in this thesis, a 0.05 level of statistical significance is being implied throughout the experiments). In statistics, a result is called statistically significant if it is unlikely to have occurred by chance. The amount of evidence required to accept that an event is unlikely to have arisen by chance is known as the significance level or critical p-value.

As with Figure 4.7, the Q-learning gets to an optimal policy after more than 90 episodes. It converges very slowly and gradually overall; however, due to its planless characteristic, it has large fluctuations throughout the learning process.

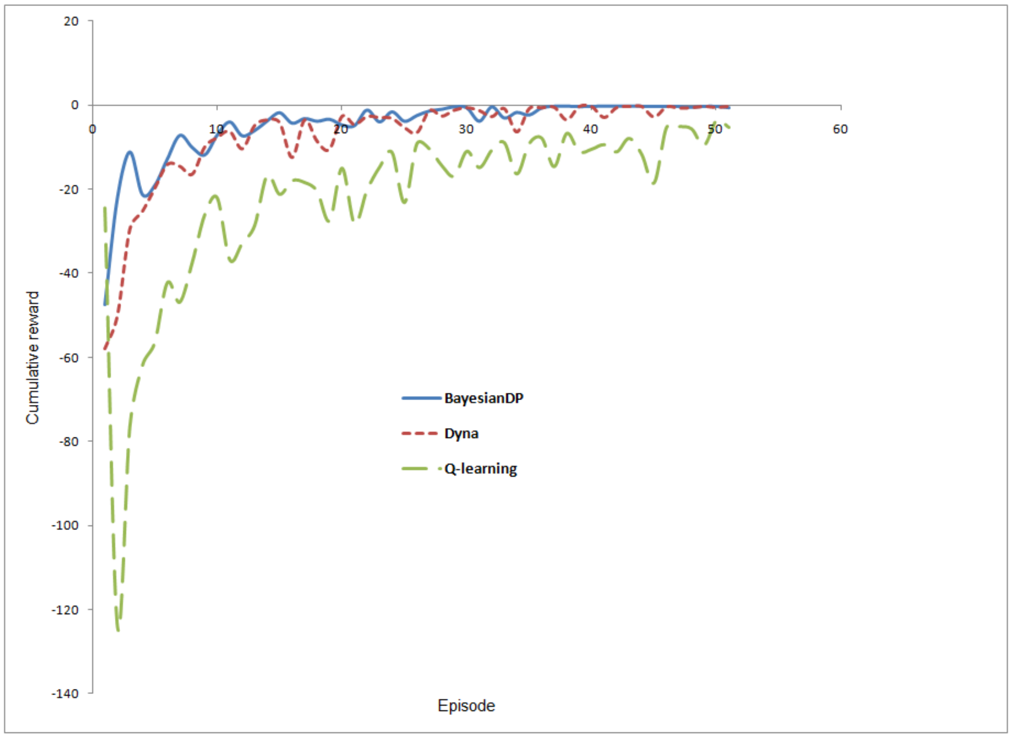


Figure 4.9: A comparison for the cumulative reward: Bayesian, Dyna, and Q-learning agents (average over 10 runs)

The Bayesian technique converges more gradually and quickly than Dyna. Although Dyna has a fast and reasonable convergence overall, we notice wavery changes with some irregularities. This is because Dyna chooses k random samples (in this example 20) and tries these hypotheses to find the best policy. The randomness of choosing these samples is the reason for these small irregularities throughout the learning process. On the other hand, the Bayesian technique converges more steadily because it considers several hypothetical MDPs to solve based on the probability distributions of each state-action and the natural inference in Bayesian updates.

4.3.3 Discussion

Table 4.1 shows the three different techniques in reinforcement learning and their characteristics. Overall, the Bayesian approach has a better performance than Dyna and Q-learning. It finds the best policy quickly and explores the environment to form its belief about the model of the world. The Bayesian technique uses the value of perfect information (VPI), as described in Chapter 3, to examine the value of each new piece of information and to avoid taking suboptimal actions.

RL technique	Bayesian RL	Dyna	Q-learning
Type	model-based	mid-ground	model-free
Exploration rate	high (smart)	high (smart)	high (dumb)
Convergence	rapid	rapid(with fluctuations)	slow
Model learning	intelligent(Bayes' updates)	random samples	None

Table 4.1: The comparison of three different RL techniques

Although Dyna also converges really fast, it has some small fluctuations throughout the learning process. It tries to learn the model of the environment simply by considering random samples and solving the MDPs based on these samples.

The Q-learning approach converges slowly especially in larger-space environments where it cannot rapidly update the Q-values based on the previous values. This technique does not try to learn the model of the environment explicitly; therefore, its exploration is solely near stochastic. Nevertheless, it is one of the most interesting reinforcement learning techniques, as it is computationally easier to implement.

4.4 Normative Agents With Prior Knowledge

The previous section examined the performance of the Bayesian approach in learning the model of the environment. It showed that Bayesian model learning with the VPI approach [29] outperforms the model-free approach of Q-learning and the Dyna approach, which is a middle-ground between the model-free and model-based RL approaches. However, these agents preserve no knowledge after each simulation and start up from scratch in a new environment. They have no prior over the environment's dynamics and behavioral norms; therefore, they start exploring with absolutely no information about the domain or the environment, and their behavior is random early in the learning process.

This section experiments with the effectiveness of the two-level reinforcement learning framework to dynamically generate norms. As described in detail in Section 3.4, this thesis focuses on the behavioral norms that affect the behavior of agents in the process of decision making based on their interactions with the environment. An agent's behavior in any environment is tightly dependent on its understanding of the surrounding environment. Norms extracted from the interactions with the environment are incorporated as domain-dependent knowledge into agents. Agents build up their beliefs about the normative framework of the environment and its dynamics and then use this aggregated knowledge in other settings with different dynamics. This will help the agent to avoid random behaviors in the beginning of a mission under a new and unknown dynamic.

Three different experiments are considered with two agents: a Bayesian agent with no prior knowledge about the dynamics and behavioral norms, and a Bayesian agent with some training in a different environment under the same domain. The environment's dynamics and its behavioral norms will be changed to study which agent better performs when confronting a new setting.

An interesting approach to study this difference is to consider the differences based on the percentage of changes in settings. This way we are able to study the effectiveness of the learned normative behaviors in different environments. Nonetheless, as it was emphasized earlier, the domain in which the agent is finding an optimal policy to the goal state will remain the same. In these experiments, changes can occur in every element of the environment such as blocked states, goal states, start states, etc. Three different experiments have been done based on the percentage of changes:

- Only change in the goal state
- 20% change in the environment
- 50% change of dynamics + change in the goal state

4.4.1 Experiment 1: Goal Change

This experiment tests the rate of convergence to the optimal policy, and the cumulative reward for an agent with no prior knowledge over the behavioral norms and the system's dynamics versus an agent trained in another setting with semi-developed knowledge about the domain. I would like to study how fast the agents can adapt their belief to the changes in the goal.

Here in this experiment, I consider a map of 12x10 with 98 reachable states (the exact map used in Section 4.3.1). I change the goal state from the top-right of the map to the top-left.

Figures 4.10 and 4.11 show the performance of both agents with regard to steps to the goal and cumulative rewards gained in each episode. The results are averages over 10 runs. Both of the agents find the best policy quickly in fewer than 15 trials. The normative agent starts up with a worse result compared to the Bayesian agent with no prior. This is due to the fact that the normative agent needs some exploration to adapt its beliefs to the new environment's dynamics, so it has to update its beliefs about the environment. However, after the first exploration of the map it rapidly finds the

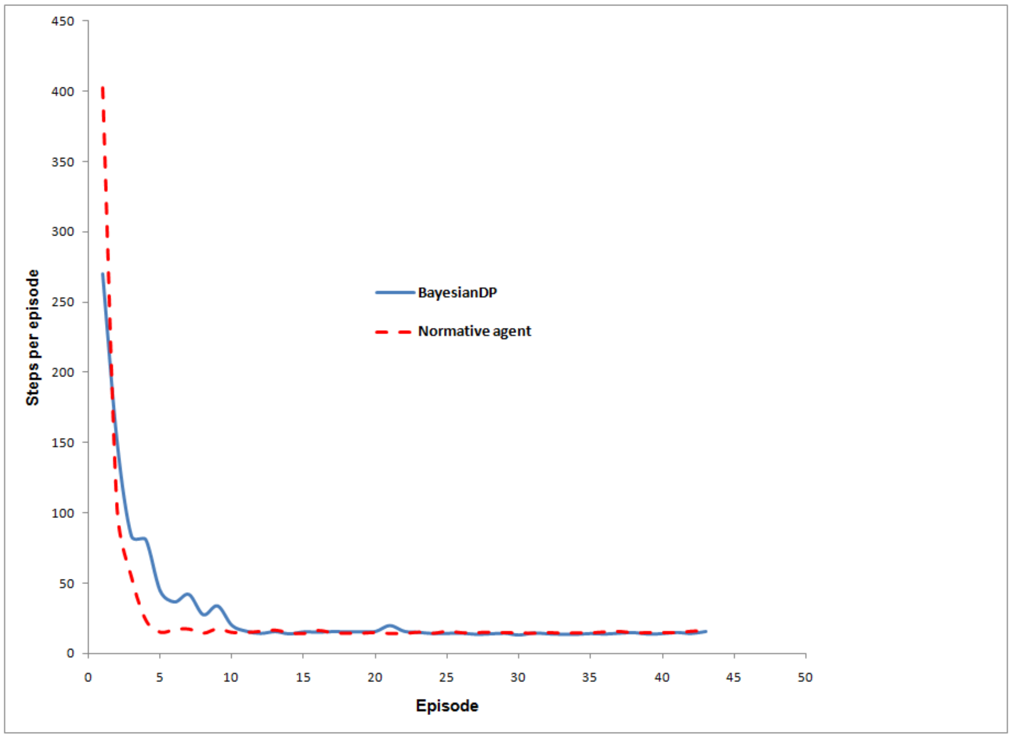


Figure 4.10: The number of steps to the goal for an agent with no prior knowledge vs. a normative agent with prior knowledge (average over 10 runs)

best policy and converges after 5 trials, as opposed to the Bayesian agent with no training.

The next section studies to what degree the generated norms and knowledge will help an agent to adapt to the new environment's dynamics.

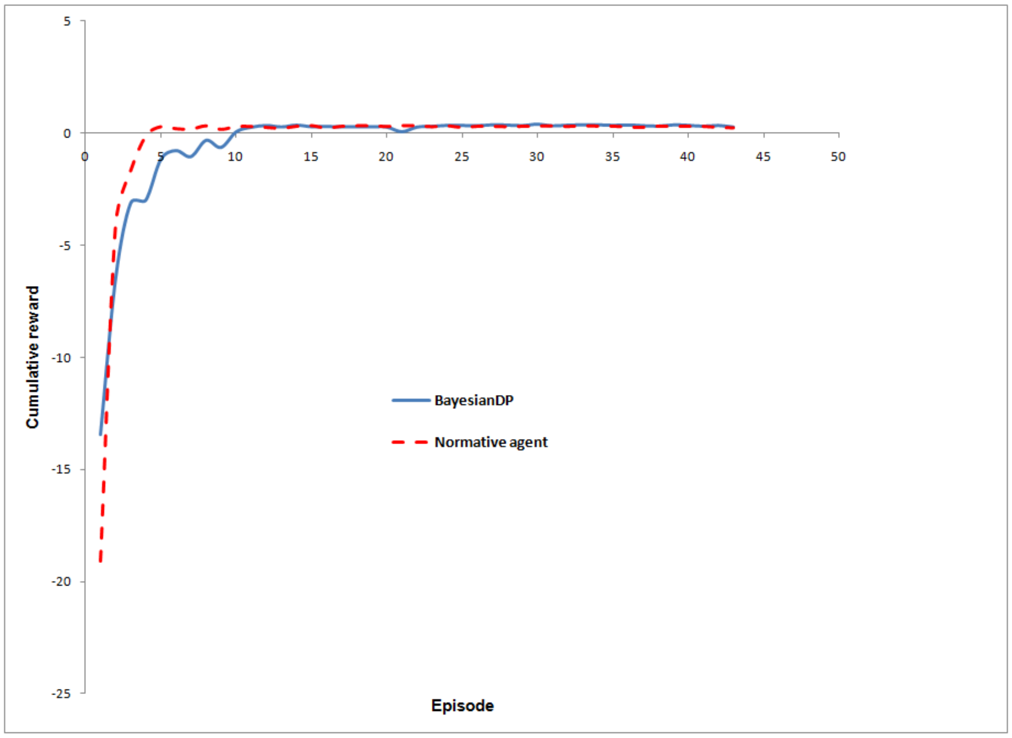


Figure 4.11: Cumulative reward for an agent with no prior knowledge vs. a normative agent with prior knowledge (average over 10 runs)

4.4.2 Experiment 2: World Change (20%)

To better study the effectiveness of having prior information on a certain domain, I make more changes (20% change) and see how a normative agent adapts to the environment when it is set to work in an environment with a 20% margin of difference. As shown in Figure 4.13, the changes are in the locations of blocked states and the location of start state, however, the goal state remains the same. This represents a world in which the dynamics are 20% different from the dynamics of the previously learned world, but the agent's goal does not change (in this example, the agent's purpose is to find the goal state while maximizing the overall cumulative reward).

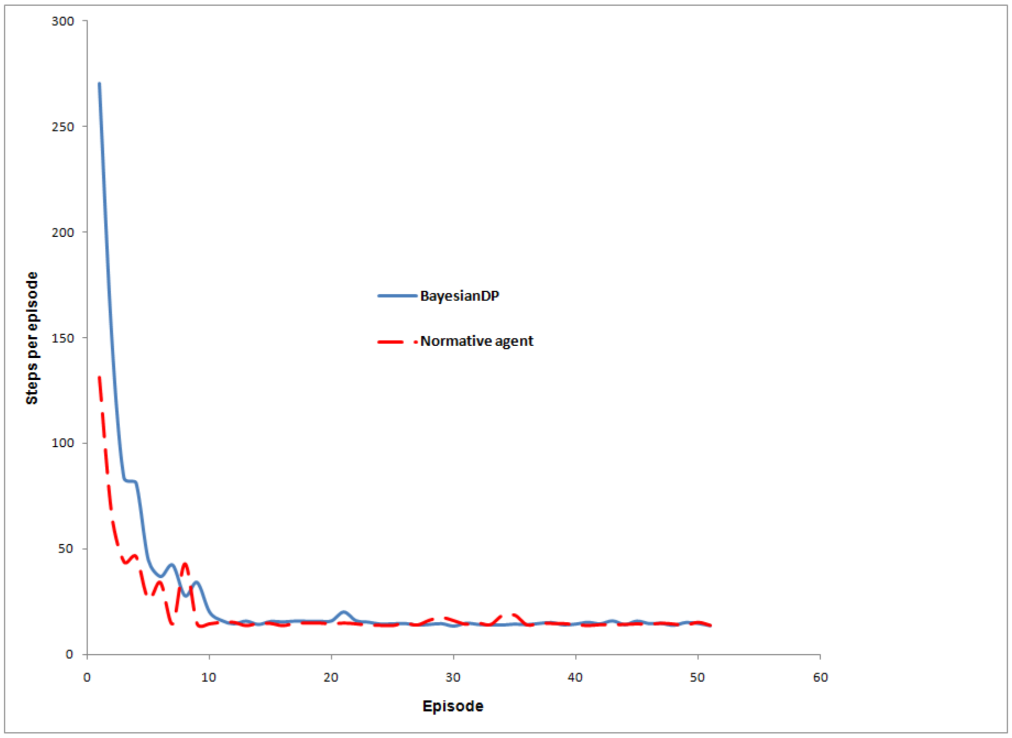


Figure 4.12: 20-percent difference- the number of steps to the goal for both agents (average over 10 runs)

The agent is trained and learned in the map from Figure 4.2 and tested to learn the new environment's dynamics and adjust its behavioral norms (Figure 4.13).

The result of this experiment shows the importance of incorporating prior information especially in the early stages of the learning process. The agent that has practiced and captured knowledge about the domain has some sort of understanding of the environment and has less random behavior.

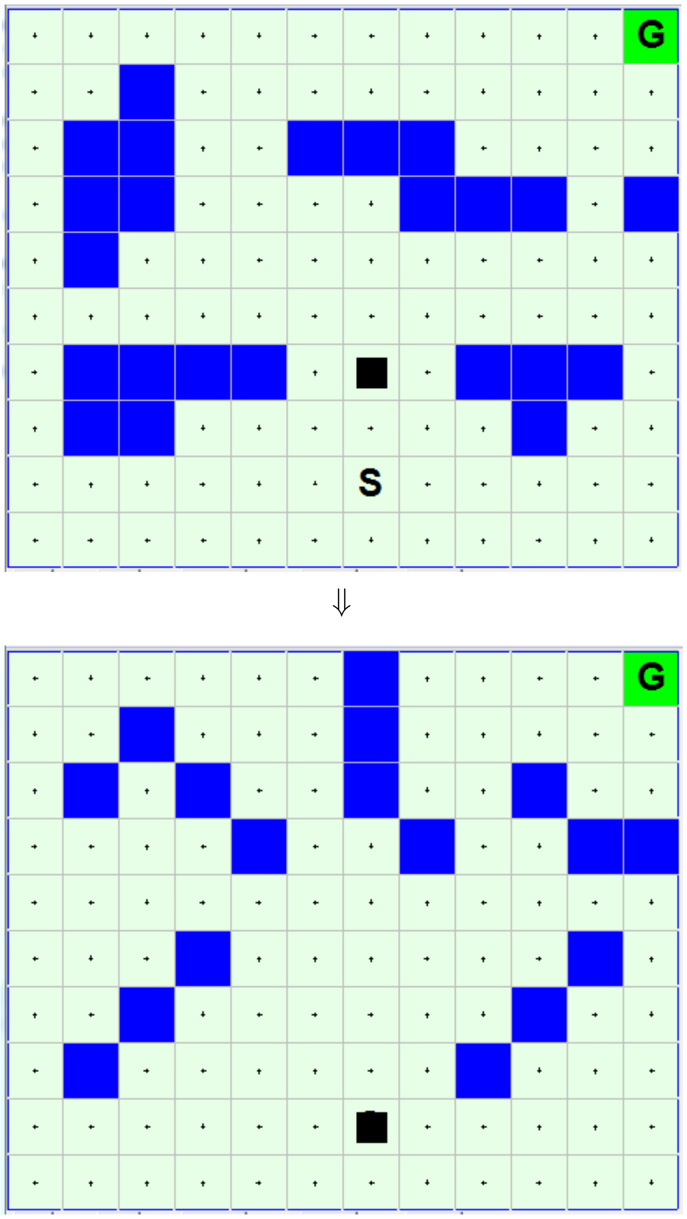


Figure 4.13: The agent learns and extracts knowledge in the first map and then uses it in the second environment

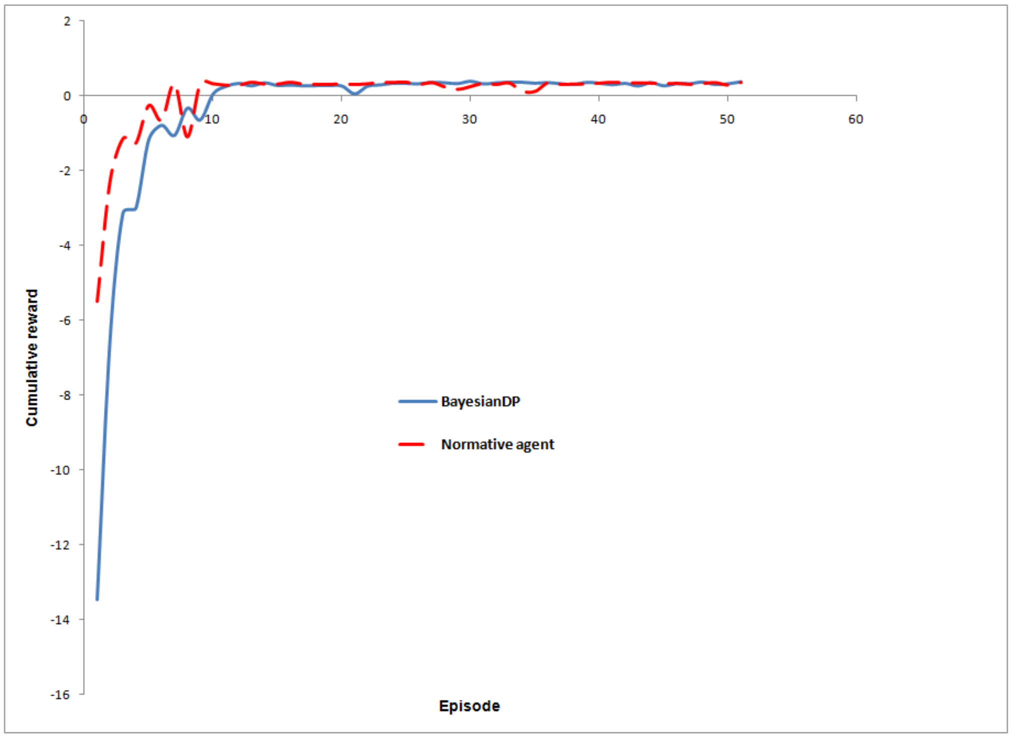


Figure 4.14: 20-percent difference- the cumulative reward for both agents (average over 10 runs)

In the very first trials of learning, the normative agent starts with finding the new optimal policy. On the other hand, some fluctuations in early phases show the agent's attempts to explore the new environment and find out the dynamics as well as exploiting the already known states. In the early learning process, the performance of the normative agent with prior knowledge is statistically significant ($p = 0.027152704$) compared to the Bayesian agent with no knowledge about the normative actions.

A trained agent learns the probability of finding the goal state in each zone of the map so the agent focuses more on the areas that have been learned to be more probable in containing the goal state. In this example, this leads the agent to focus more on the central areas and avoid exploring behind the blocked states in right and left side of the map.

The agent is able to adjust itself to the new environment's dynamics with 20% of change. The next section evaluates this idea under the situations where 50% of dynamics are different. Also, to make it more difficult, the goal will be changed to another state.

4.4.3 Experiment 3: Goal and World Change (50%)

To make a thorough conclusion about the normative agents, a 50% change in the environment is considered. The agent trains in an environment with certain dynamics and tries to learn the behavioral norms in this setting, and then is asked to perform under a different circumstance where there is just 50-percent similarity in the locations of blocked states and the location of start state(Figure 4.17). To make it more difficult, the goal state has also been changed to another state.

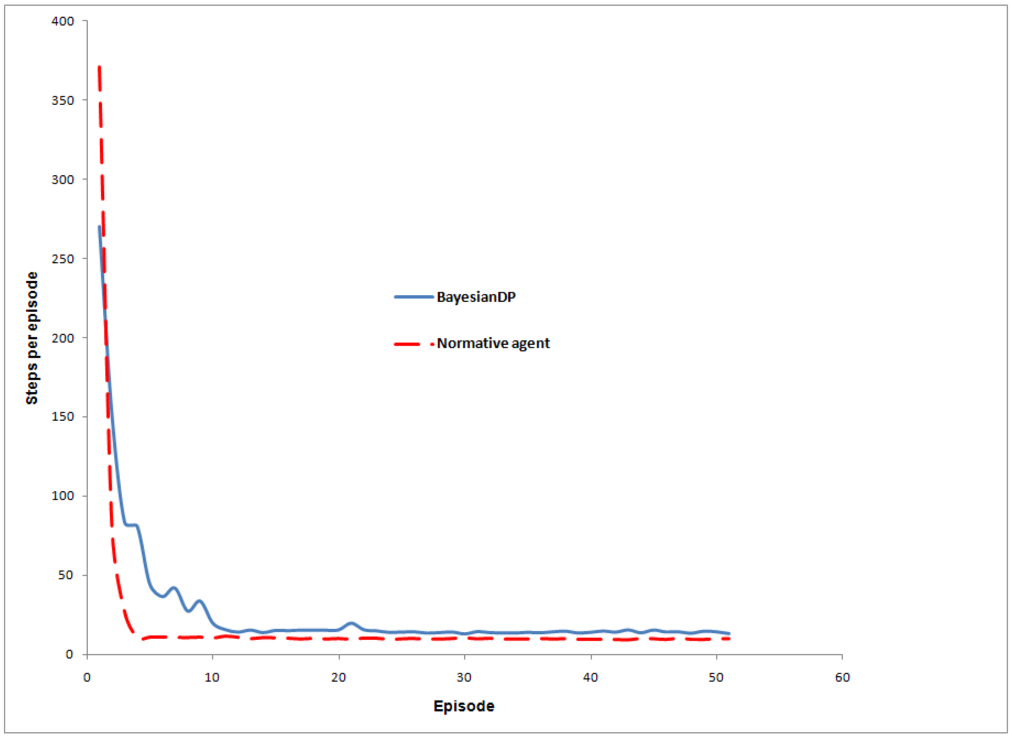


Figure 4.15: 50-percent difference- the number of steps to the goal for both agents (average over 10 runs)

As soon as the normative agent extracts the behavioral norms and learns the system's dynamics, and the level of confidence about its beliefs meets the requirements (as described in Chapter 3), it is deployed to a different environment with more than 50% of dissimilarities.

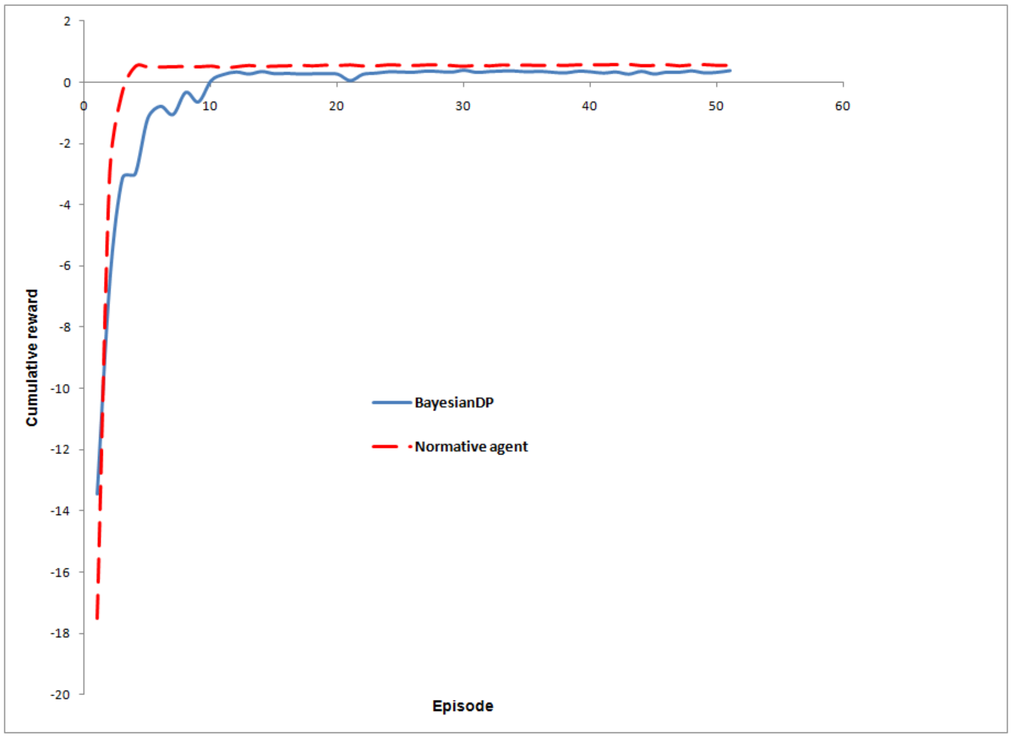


Figure 4.16: 50-percent difference- the cumulative reward for both agents (average over 10 runs)

Although the normative agent starts with a higher number of steps to the goal, Figure 4.15 shows a considerable performance early in the learning process. It converges quickly after two or three episodes of exploration. During this process, the agent adjusts its beliefs based on the new environment and quickly updates its normative system.

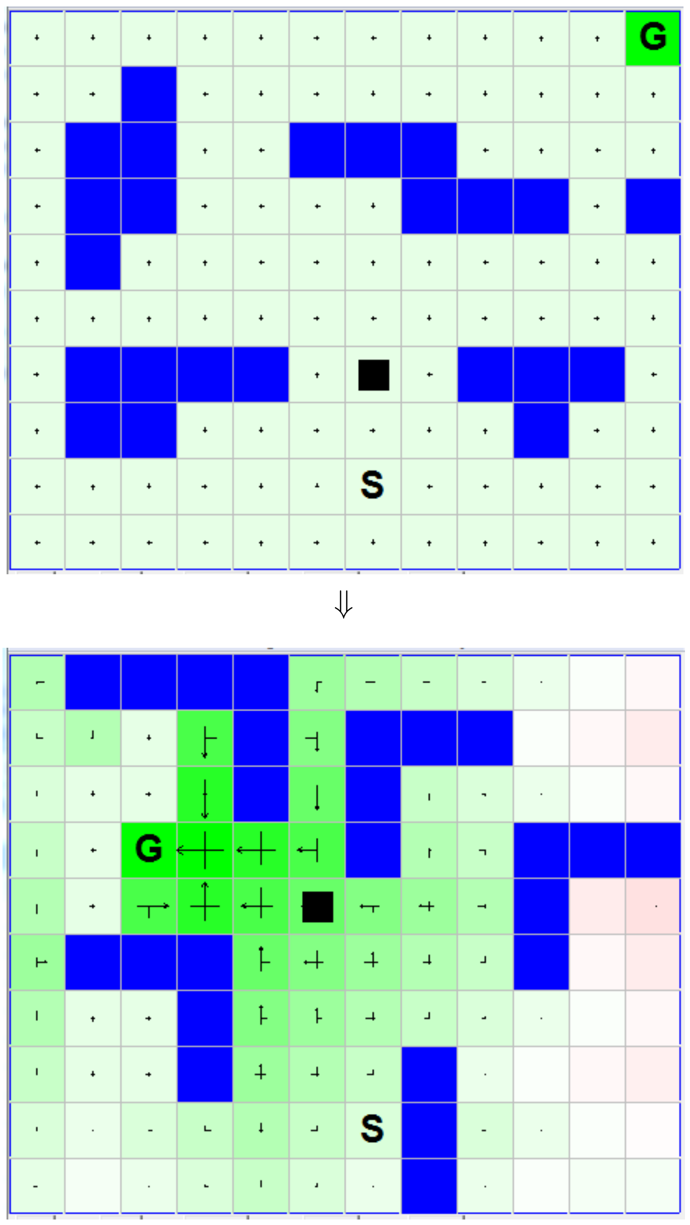


Figure 4.17: The agent learns and extracts knowledge in the first map and then uses it in the second environment

As shown in Figure 4.16, we notice some increased drop in the value of cumulative reward in the first episodes because the agent is adapting its belief state under the new dynamics. However, the value of cumulative reward rises more rapidly and converges to the value of the optimal policy after about 5 episodes. This proves the effectiveness of having prior knowledge about the domain-dependent norms even if the environment changes over time and the agent wants to start learning in a world with different dynamics and different normative system. A paired t-test demonstrates that the difference in means between the normative agent and the agent with no prior knowledge is statistically significant (p = [0.022022831]).

4.4.4 Discussion

The performance of an agent, whether it has prior knowledge about the normative behaviors or not, converges at some point at a reasonable pace. However, an important factor is to avoid any random exploratory behavior at the beginning of a simulation. More specifically, reinforcement learning agents (even the ones that are very quick) spend some time at the start of every new simulation to find out about the dynamics and the environment by visiting states randomly.

The purpose of the proposed learning framework essentially is to endow agents with some prior knowledge about the domain and the dynamics, including the normative framework. To automate the process of norm generation, learning agents should have a mechanism to perceive from the environment, process the new knowledge, update their beliefs, and reuse this information when dealing with new problems. This can be using the reinforcement learning techniques and their ability to learn through interactions. The Agent builds up its beliefs on behavioral norms in a certain environment by representing the state-action pairs in the form of probability distributions. Although this knowledge about the behavioral norms is different in any other environment, the agent will adjust the learned norms to the norms in a new environment; therefore, it finds the optimal policy and the model of the environment more quickly. The experiments showed a considerable amount of performance early in the process of learning.

As we can see in Figure 4.18, the normative agent performs better both in gaining cumulative reward and finding the optimal policy to the goal. The more similar the new environment is to the environment where the agent has been trained, the faster and better it can adjust its beliefs to the new situations. Another interesting observation is that whenever the goal state is very different from the one learned by the agent, the agent has to violate or alter its beliefs to the new situations. Thus, this adjustment process makes the agent override some of the behavioral norms and spend some time to explore the new environment. However, as the agent carries its domain knowledge from the previous experiments, it easily adapts its normative system after just a couple of episodes. The more it takes for the agent to find the best policy, the more it should update/alter its belief systems on behavioral norms.

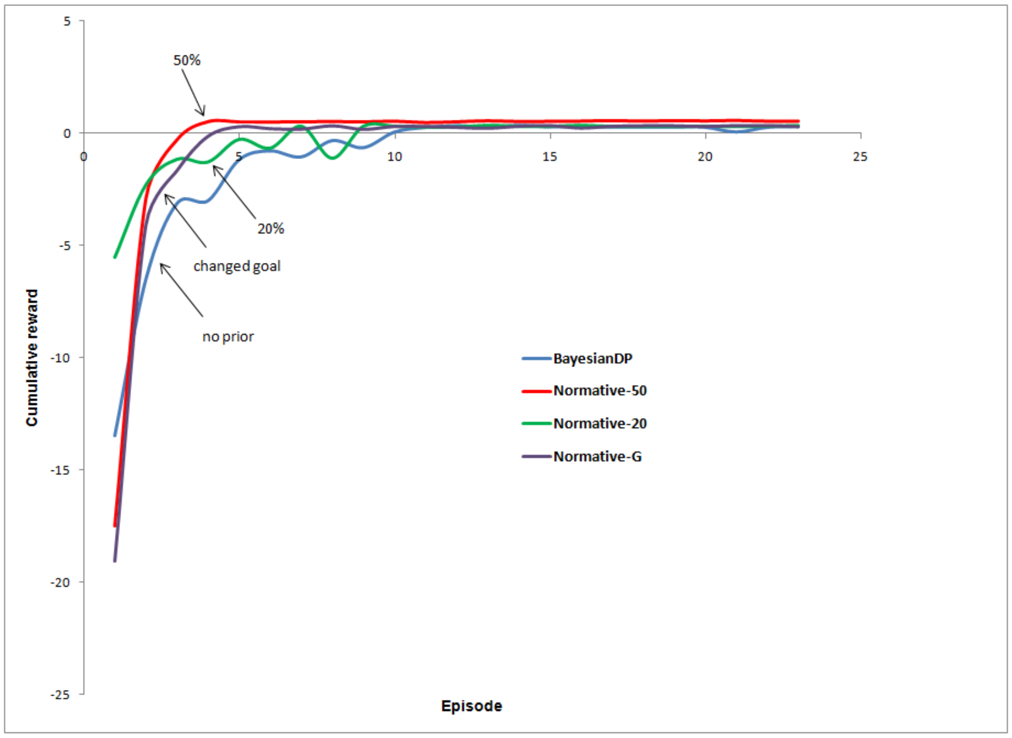


Figure 4.18: The comparison between different values of change

The figure shows that the agent performs better in an environment with 20% change in its dynamics. It adapts itself under the new dynamics rapidly and then starts to visit other states in order to update its beliefs and behavioral norms. On the other hand, when the agent has to perform in an environment with 50% change, it takes more stages at the beginning for the agent to adjust its knowledge to the new environment. Moreover, in the early stages of learning (3 or 4 episodes), the agent gets a highly negative reward as the goal has been changed, and the agent needs to explore and unlearn its current beliefs. By reducing the similarities between the environments, in the early process of learning the behavior of the normative agent changes from high performance to near stochastic as the agent should alter most of its beliefs when confronting an entirely different environment.

4.5 Summary

This chapter illustrated the experimental results and analysis to the introduced framework for generating and using norms. First, it compared three different learning algorithms (Bayesian RL, Dyna, and Q-learning). It is shown that either in a small state-space or large state-space, the Bayesian agent outperforms other learning techniques by learning the model of the environment and the world's dynamics.

Second, the ideas of generating norms dynamically and incorporating them into agents are discussed and examined in various experiments. Each of the experiments showed the efficiency of a trained agent working in the environments with different dynamics. When the world's dynamics are similar to those of the previous environment, the agent adjusts itself rapidly to the new settings by updating its belief set. It was shown that a normative agent can adjust its behavioral norms and its belief rapidly when operating in a new environment. Thus, the performance of the agent increases, especially in the early stages of the learning process.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

This thesis presented a broad overview of reinforcement learning (RL) techniques and normative systems. Two classes of algorithms (model-free and model-based) in RL were described. A middle-ground approach of Dyna and its pros and cons were explained. Bayesian inference was then described in detail. It was discussed that Bayesian RL outperforms other RL techniques in finding the best equilibrium for the exploration-exploitation problem. Then the use of norms and normative frameworks in artificial intelligence was reviewed.

Later, in Chapter 3, I explained my ideas in building a two-level learning framework for dynamic norm generation. This is an automatic process to for extracting norms using reinforcement learning techniques and then using this knowledge to enhance the behavior of the learning agents. First-level learning refers to learning a certain environment with specific dynamics. The second level uses the results from the first-level learning in different environments and generalizes this knowledge to set them as domain-dependent behavioral norms. Bayesian RL is a systematic way to incorporate prior knowledge about the domain into the learning iterations. The agent should be confident about its beliefs in order to generalize them and use them in the future settings. This confidence level is obtained by checking two conditions: how familiar the agent is with the current world and its dynamics (including the norm system) and whether it has converged to an optimal policy.

Generating norms in an autonomous way using learning agents was shown to have many advantages over the conventional methods of norm assignments. Better interactions, better decision-making processes, automated updates and changes, proactive flexible agents, and easier analysis of norms and domain knowledge are some of the reasons to use this framework. Moreover, it helps to better understand the ways our beliefs about behavioral norms and our environment's dynamics become established.

Chapter 4 of this thesis reported the effectiveness of the proposed framework. First, the three different techniques in reinforcement learning were compared in a small state-space and a large state-space to depict the performance of each algorithm. It was discussed that the Bayesian approach to model learning outperforms the other techniques as it learns the model of the world using Bayes' updates.

Second, the ideas of generating norms dynamically and incorporating them into agents are discussed and examined in various experiments. Each of the experiments demonstrated a high efficiency of a trained agent working in the environments with different dynamics. When the world's dynamics are similar to those of the previous environment, the agent adjusts itself rapidly to the new settings by updating its belief set. It was shown that a normative agent can adjust its behavioral norms and its beliefs rapidly when operating in a new environment. Thus, the performance of the agent increases, especially in the early stages of the learning process.

5.2 Future Work

In this thesis, I proposed a two-level learning approach that is applicable to any decision-making problem where agents need to learn domain-dependent knowledge about some dynamics. There are many ways to extend this work, either from the theoretical or practical point of view. I will try to sketch some of these ideas for future work:

- Multi-agent systems

  An interesting topic to tackle would be to apply the proposed framework to a multi-agent system. The multi-agent view of systems is becoming more important because most of the communities feel the need to consider other entities or agents while making decisions or taking actions. There have been much work done in the multi-agent reinforcement learning area. One can use the introduced framework and run many agents in a world to learn the world's dynamics and its normative system. This might need a small change in the method for updating the beliefs because the correct view of the world would be the aggregation of all the agents' beliefs. Applying the proposed framework into these systems will bring up other issues such as trust, reputation, benevolence, malevolence, etc.

- Conflicting norms

  Norms can be inconsistent. This happens especially when norms have different origins. As it was shown in [33], the problem of these conflicts is not that they are general (logical) conflicts between the norms, but that they are only conflicts in very specific situations or even in ways in which norms are fulfilled. Conflicts might arise when individuals are members of several groups in which conflicting norms emerge due to the different circumstances in those groups. An important question is how one can handle these conflicting norms when agents confront groups or societies with completely opposite norms.

- Behavioral norms and social norms

  This thesis discussed how behavioral norms can be learned and incorporated into agents. One can make use of this approach and study the behaviors of agents by considering social and behavioral norms at the same time. Using a simulation purely based on individual agents would not be correct. This would model a situation where there are no social structures that connect and limit individual autonomy. Thus, one can use a modeling framework that combines both a societal and an individual point of view. Of course these two categories overlap at some points, but because norms, in general, guide the behavior of agents and the ways agents make decisions, there are many ways that social and behavioral norms affect each other.

- Extracting domain culture

  Culture can be seen as an expression of a certain set of values common to a group or society. Thus, there is a connection between culture and behavior in every society. Agents learn social or behavioral norms by interacting with the environment or other agents. This leads them to integrate into new societies and worlds. Individuals try to learn norms in order to understand the whole normative system. One can use the techniques provided in this thesis in order to learn the culture among the agents of a community. Any agent has its beliefs and normative behaviors; however, averaging these norms and trying to find the normative framework of the majority of agents will lead to the emergence of the dominant culture between the agents of a certain society.

Bibliography

[1] K.J. Astrom, *Optimal control of Markov decision processes with incomplete state estimation*, Journal of Mathematical Analysis and Applications **10** (1965), 174–205.

[2] J. Bates, *The role of emotion in believable agents*, Communications of the ACM **37** (1994), no. 7, 122–125.

[3] J. Bates, A. Loyall, and W. Reilly, *An architecture for action, emotion, and social behavior*, Artificial social systems, 55–68.

[4] R. Bellman, *Dynamic Programming, Princeton*, NJ: Princeton UP (1957).

[5] RE Bellman, *Adaptive control processes: A guided tour*, (1961).

[6] RE Bellman and LA Zadeh, *Decision-making in a fuzzy environment*, Management science **17** (1970), no. 4, 141–164.

[7] D.P. Bertsekas, *Dynamic programming: deterministic and stochastic models*, Prentice-Hall, Inc. Upper Saddle River, NJ, USA, 1987.

[8] DP Bertsekas, *Dynamic programming and optimal control*, (1995).

[9] G. Boella and L. Lesmo, *Deliberate normative agents*, Social Order in Multiagent Systems.

[10] G. Boella, L. Van Der Torre, and H. Verhagen, *Introduction to normative multiagent systems*, Computational & Mathematical Organization Theory **12** (2006), no. 2, 71–79.

[11] M. Boman, *Norms in artificial decision making*, Artificial Intelligence and Law **7** (1999), no. 1, 17–35.

[12] A.H. Bond and E. Les Gasser, *Readings in distributed artificial intelligence*, Morgan Kaufmann San Mateo, CA, 1988.

[13] C. Boutilier, T. Dean, and S. Hanks, *Decision-theoretic planning: Structural assumptions and computational leverage*, Journal of Artificial Intelligence Research **11** (1999), no. 1, 94.

[14] L. Bovens and S. Hartmann, *Bayesian epistemology*, Oxford University Press, USA, 2003.

[15] W. Briggs and D. Cook, *Flexible social laws*, INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE, vol. 14, LAWRENCE ERLBAUM ASSOCIATES LTD, 1995, pp. 688–693.

[16] Encyclopdia Britannica, *norm*, 2010, [Encyclopdia Britannica Online. 29 Jul. 2010].

[17] C. Castelfranchi, *Guarantees for autonomy in cognitive agent architecture*, Intelligent Agents (1995), 56–70.

[18] C. Castelfranchi, F. Dignum, C. Jonker, and J. Treur, *Deliberative normative agents: Principles and architecture*, Intelligent Agents VI. Agent Theories Architectures, and Languages (2000), 364–378.

[19] G. Chalkiadakis, *A Bayesian approach to multiagent reinforcement learning and coalition formation under uncertainty*, Ph.D. thesis, Citeseer, 2007.

[20] G. Chalkiadakis and C. Boutilier, *Coordination in multiagent reinforcement learning: A bayesian approach*, Proceedings of the second international joint conference on Autonomous agents and multiagent systems, ACM, 2003, p. 716.

[21] G. Chalkiadakis and C. Boutilier, *Coalitional bargaining with agent type uncertainty*, Proc. 20th IJCAI, 2007.

[22] P. Chouinard, *Meta-Organizational Decision Making and Collaboration*, Defence Research and Development Canada, 2009.

[23] J.A. Clouse, *Learning from an automated training agent*, Adaptation and Learning in Multiagent Systems. Springer Verlag, Berlin (1996), 195.

[24] W.M. Cohen and D.A. Levinthal, *Absorptive capacity: a new perspective on learning and innovation*, Administrative science quarterly **35** (1990),

no. 1.

[25] Encyclopedia Columbia, *norm*, 2010, [The Columbia Electronic Encyclopedia 6th ed, 29 Jul. 2010].

[26] Castelfranchi C. Conte, R. and F. Dignum, *Autonomous norm acceptance*, Lecture notes in computer science (1999), 99–112.

[27] R. Conte and C. Castelfranchi, *Cognitive and social action*, Garland Science, 1995.

[28] R. Conte, C. Castelfranchi, and F. Dignum, *Autonomous norm acceptance*, Intelligent Agents V. Agent Theories, Architectures, and Languages: 5th International Workshop, ATAL'98, Paris, France, July 1998. Proceedings, Springer, 2000, pp. 66–66.

[29] R. Dearden, N. Friedman, and D. Andre, *Model based Bayesian exploration*, Proceedings of the fifteenth Conference on Uncertainty in Artificial Intelligence, Citeseer, 1999, pp. 150–159.

[30] R. Dearden, N. Friedman, and S. Russell, *Bayesian Q-learning*, PROCEEDINGS OF THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE, JOHN WILEY & SONS LTD, 1998, pp. 761–768.

[31] M.H. DeGroot, *Optimal statistical decisions*, Wiley-IEEE, 2004.

[32] F. Dignum, *Autonomous agents with norms*, Artificial Intelligence and Law **7** (1999), no. 1, 69–79.

[33] F. Dignum and V. Dignum, *Emergence and enforcement of social behavior*, 18th World IMACS Congress and MODSIM09 International Congress on Modelling and Simulation (2009).

[34] F. Dignum, V. Dignum, and C.M. Jonker, *Towards agents for policy making*, Proceedings of the Multiagent-based Simulation, MABS (2008).

[35] A. Dutech, *Solving POMDPs using selected past events*, European Conference on Artificial Intelligence **3** (2008).

[36] J.M. Epstein, *Learning to be thoughtless: Social norms and individual computation*, Computational Economics **18** (2001), no. 1, 9–24.

[37] C.B. Excelente-Toledo and N.R. Jennings, *Using reinforcement learning to coordinate better*, COMPUTATIONAL INTELLIGENCE-OTTAWA THEN BOSTON AND OXFORD- **21** (2005), no. 3, 217.

[38] JR Galliers, *A theoretical framework for computer models of cooperative dialogue, acknowledging multi-agent conflict*, (1988).

[39] M.R. Genesereth and S.P. Ketchpel, *Software agents*, Commun. ACM **37** (1994), no. 7, 48–53.

[40] A. Hájek and S. Hartmann, *Bayesian Epistemology*, Blackwell Companion to Epistemology (2008).

[41] R.A. Howard, *Dynamic programming and Markov process*, MIT press, 1960.

[42] RA Howard, *Information value theory*, IEEE Transactions on systems science and cybernetics **2** (1966), no. 1, 22–26.

[43] S. Huang and D. Distante, *On Practice-Oriented Software Engineering Education*, CONFERENCE ON SOFTWARE ENGINEERING EDUCATION & TRAINING WORKSHOPS–CSEETW, vol. 6, Citeseer, 2006, p. 19.

[44] N.R. Jennings, *On agent-based software engineering* 1*, Artificial Intelligence **117** (2000), no. 2, 277–296.

[45] L.P. Kaelbling, M.L. Littman, and A.W. Moore, *Reinforcement learning: A survey*, Journal of Artificial Intelligence **4** (1996), no. 1, 237–285.

[46] D.K. Lewis, *Convention: A philosophical study*, Wiley-Blackwell, 2002.

[47] M.L. Littman, T.L. Dean, and L.P. Kaelbling, *On the complexity of solving Markov decision problems*, Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence, Citeseer, 1995, pp. 394–402.

[48] JJ Martin, *Bayesian decision problems and Markov chains*, (1967).

[49] G.E. Monahan, *A survey of partially observable Markov decision processes: Theory, models, and algorithms*, Management Science (1982), 1–16.

[50] A.W. Moore and C.G. Atkeson, *Prioritized sweeping: Reinforcement learning with less data and less time*, Machine Learning **13** (1993), no. 1,

103–130.

[51] P. Mukherjee, S. Sen, and S. Airiau, *Emergence of Norms with Biased Interactions in Heterogeneous Agent Societies*, Proceedings of the 2007 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology-Workshops, IEEE Computer Society, 2007, pp. 512–515.

[52] D.C. North, *Institutions, institutional change, and economic performance*, Cambridge Univ Pr, 1990.

[53] Sean Norton, *Adapting and Evolving in Emergency Response: The Case for More Complex Multi-Organizational Partnerships*, Defence Research and Development Canada, 2008.

[54] J. Peng and R.J. Williams, *Efficient learning and planning within the Dyna framework*, From animals to animats 2: proceedings of the Second International Conference on Simulation of Adaptive Behavior, The MIT Press, 1993, p. 281.

[55] P. Poupart, N. Vlassis, J. Hoey, and K. Regan, *An analytic solution to discrete Bayesian reinforcement learning*, Proceedings of the 23rd international conference on Machine learning, ACM, 2006, p. 704.

[56] M.L. Puterman, *Markov decision processes: Discrete stochastic dynamic programming*, IMA Journal of Management Mathematics.

[57] H. Raiffa and R. Schlaifer, *Applied statistical decision theory*, Wiley New York, 2000.

[58] J.S. Rosenschein and M.R. Genesereth, *Deals among rational agents*, 1985.

[59] S. Russell and P. Norvig, *Artificial intelligence: a modern approach*, New Jersey (1995).

[60] B.T.R. Savarimuthu, S. Cranefield, M. Purvis, and M. Purvis, *Role model based mechanism for norm emergence in artificial agent societies*, Proceedings of the 2007 international conference on Coordination, organizations, institutions, and norms in agent systems III, Springer-Verlag, 2007, pp. 203–217.

[61] S. Sen and S. Airiau, *Emergence of norms through social learning*, Proceedings of the Twentieth International Joint Conference on Artificial Intelligence, 2007, pp. 1507–1512.

[62] Y. Shoham, *An overview of agent-oriented programming*, Software agents (1997), 271–290.

[63] M. Strens, *A Bayesian framework for reinforcement learning*, MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-, Citeseer, 2000, pp. 943–950.

[64] J. Suijs and P. Borm, *Stochastic cooperative games: superadditivity, convexity, and certainty equivalents*, Games and Economic Behavior **27**

(1999), no. 2, 331–345.

[65] R.S. Sutton, *Learning to predict by the methods of temporal differences*, Machine learning **3** (1988), no. 1, 9–44.

[66] R.S Sutton, *Integrated architectures for learning, planning, and reacting based on approximating dynamic programming*, Proceedings of the Seventh International Conference on Machine Learning, vol. 216, Citeseer, 1990, p. 224.

[67] R.S Sutton, *Reinforcement learning architectures*, Proceedings ISKIT **92** (1992).

[68] R.S. Sutton and A.G. Barto, *Introduction to reinforcement learning*, (1998).

[69] S. Thrun, *The role of exploration in learning control*, Handbook of intelligent control: Neural, fuzzy and adaptive approaches (1992), 527–559.

[70] R. Tuomela, *The importance of us: A philosophical study of basic social notions*, Stanford Univ Pr, 1995.

[71] W.W. Vasconcelos, *Norm verification and analysis of electronic institutions*, Procs. AAMAS **4**.

[72] J. Vázquez-Salceda, *The role of Norms and Electronic Institutions in Multi-Agent Systems applied to complex domains. The HARMONIA framework*, AI Communications **16** (2003), no. 3, 209–212.

[73] J. Vázquez-Salceda, H. Aldewereld, and F. Dignum, *Norms in multiagent systems: from theory to practice*, International Journal of Computer Systems Science & Engineering **20** (2005), no. 4, 225–236.

[74] H. Verhagen, *Norms and artificial agents*, Sixth Meeting of the Special Interest Group on Agent-Based Social Simulation, ESPRIT Network of Excellence on Agent-Based Computing, 2001.

[75] C.J. Watkins, *Models of delayed reinforcement learning*, Ph.D. thesis, Ph. D. thesis, Cambridge University, 1989.

[76] C.J.C.H. Watkins and P. Dayan, *Q-learning*, Machine learning **8** (1992), no. 3, 279–292.

[77] J.E. White, *Telescript technology: The foundation for the electronic marketplace*, General Magic white paper (1994).

[78] S.D. Whitehead and D.H. Ballard, *Learning to perceive and act by trial and error*, Machine Learning **7** (1991), no. 1, 45–83.

[79] M. Wooldridge, *Agent-based software engineering*, IEE Proceedings] Software Engineering. IEE Proceedings-[see also Software **144** (1997), no. 1, 26–37.

[80] M. Wooldridge and N.R. Jennings, *Intelligent agents: Theory and practice*, The knowledge engineering review **10** (2009), no. 02, 115–152.

Vita

Candidate's full name: Hadi Hosseini

Universities attended:

University of New Brunswick

Master of Computer Science

2008-2010

Amirkabir University of Technology (Tehran Polytechnic)

Bachelor of Engineering, Computer Software Engineering

2003-2008

Publications:

- H.Hosseini, G. W. Dueck *"Building Large Toffoli Gates: A Billiard Ball Model Approach"*, Proceedings of 9th International Workshop on Boolean Problems, Freiberg, Germany, September 2010.

- H.Hosseini *"Adaptive Decision Making Under Uncertainty"*, Graduate Research Conference, University of New Brunswick, May 2010.
- H. Hosseini, G. W. Dueck, *"Toffoli Gate Implementation Using The Billiard Ball Model"*, 40th International Symposium on Multiple-Valued Logic (IEEE-ISMVL), Casa Convalescncia, Barcelona, Spain, May 2010.
- H. Hosseini, Z.Noorian, M.Ulieru, *"An Autonomous Agent-based Framework for Self-Healing Power Grid"*, IEEE Systems, Man, and Cybernetics 2009 Conference(IEEE-SMC), October 11-14, San Antonio, Texas, USA.
- H.Hosseini, M.Ulieru, *"Modeling and Simulation of Norms and Institutions in Multi-organizational Systems Using BDI Framework"(Extended Abstract)*, Atlantic Provinces Council on the Sciences(APICS) 2009, Dalhousie University, Halifax, NS, Canada.
- Hadi Hosseini, *"Redesigning and Extending of a Robotic Arm Simulator, Using Java3D with Standard Object Oriented Design Patterns"* BSWE Thesis, Amirkabir University of Technology, Tehran, Iran, 2008.