September 1, 2012

Université de Liège

Learning for exploration/exploitation in reinforcement learning

Castronovo Michael
2nd master in computer science

Supervisor: Ernst Damien

September 1, 2012

Université de Liège

Learning for exploration/exploitation in reinforcement learning

Castronovo Michael
2nd master in computer science

Supervisor: Ernst Damien

Abstract

We address in this thesis the original problem of reinforcement learning, where an agent interacts with a Markov Decision Process so as to collect, over an infinite length episode, the largest sum of discounted rewards. The agent is assumed to know nothing about the MDP at the beginning of the interaction.

The main difficulty of this problem is that the agent faces two opposite goals. On one side, he must take decisions to ensure that he rapidly explores the MDP in order to identify rapidly a near-optimal policy. On the other side, he must also exploit its current knowledge of the MDP at best to get high rewards as quickly as possible.

Current strategies for tackling this exploration/exploitation dilemma often give poor performances as soon as the size of the MDP starts growing and/or when dealing with MDPs with a relatively small discount factor. This is mainly due to the fact that the problem is difficult since when dealing with discounted rewards as performance criterion, an agent cannot easily recover from early mistakes.

In this thesis, we have assumed, as we believe it is the case for many practical cases, that prior information on the MDP to be played is also available. In such a context, we have proposed a strategy that exploits this prior information for tackling the original problem of RL.

The results reported in this thesis show that our strategy works very well. In particular, it outperforms by far other well-known strategies such as ϵ-Greedy and R-max.

Castronovo Michael

Acknowledgements

It was a pleasure for me to work on this master thesis. I would like to sincerely thank Prof. D. Ernst, my supervisor. I enjoyed his enthusiasm, and I thank him for all of his meticulous feedbacks, which allowed me to improve strongly the global quality of this thesis as well as my own writing skills.

I also extend my thanks to Dr. F. Maes and Dr. R. Fonteneau for their support, their scientific advice and for helping me with the writing of a research paper based on the materiel of this master thesis.

I would like also to thank Mr. P. Di Carlo for advising me to do my master thesis in the exciting field of reinforcement learning.

I also would like to thank all my friends for their support.

Finally, I would like to thank my family and, in particular, my parents for teaching me the value of education.

Contents

Chapter 1

Introduction

Learning is the basic process of all living beings. Many forms of learning inspired popular learning paradigms (e.g. *Evolutionary Computation* or *Reinforcement Learning*). One of the most frequently used and spontaneous learning process in the nature is mimicry: the ability to acquire new skills by observing and reproducing others' actions.

The main application of mimicry in computer sciences is to elaborate automatically good solutions to difficult problems. The reason why one wants such a procedure is generally because the problem is too hard to be analysed, or because it would cost too much resources and/or time to find a satisfying solution. Automatic approaches are attractive. Our ideal is to provide a description of the problem, and let the computer solve it on its own. From this point of view, *Constraint Programming* is very close to this: it consists to solve a problem by describing it through a list of mathematical constraints. However, this approach is difficult to apply to complex problems, since in such cases elaborating a convenient list of constraints can be a very difficult task.

The *Reinforcement Learning* field is probably the closest to mimicry. The computer is directly confronted to the problem, and has the ability to interact with it in order to learn the best way to proceed. The only thing needed by the system is a reproduction of the real problem. *Reinforcement learning* techniques are based on a non-optimal model, from which a better approach is inducted. An agent can either choose to exploit it at best or try to improve the current model. By repeating this process several times, one hopes to obtain a good solving procedure: it consists to learn the decisions to take and not to take.

Formally, one considers two components: an agent, representing the computer, and an environment, representing the problem. An agent chooses actions to perform, while the environment grants him with some rewards, knowing that each action modifies the state in which the agent stands. The only piece of information known by an agent is his current situation. The objective shared by all *Reinforcement Learning* techniques is to determine which is the best action to perform for each possible situation. Usually, an environment is called a *Markov Decision Process.*

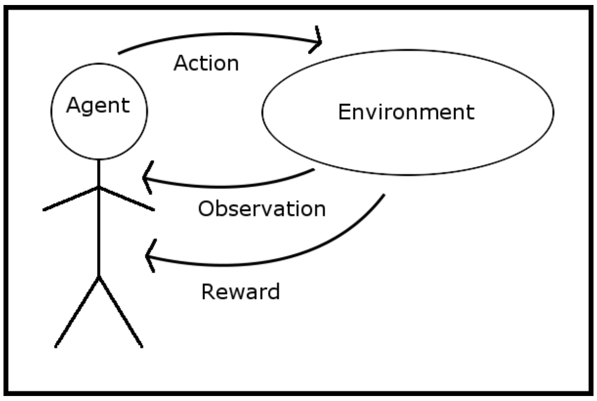

Figure 1.1: Interaction between an agent and its environment.

The process of mimicry is natural. A child learns in which situation saying "Hello" is appropriate, and when it is not, thanks to his parents. They act as the environment for an agent, granting the child with their approval (or disapproval) according to the pertinence of the performed action.

One great challenge in copying mimicry is to transcribe this feedback and interpret it in the best possible way. This is not trivial since from environment to another, the feedback is more less explicit. Sometimes, the agent wastes a lot of time before finding the best sequence of actions. Generally, it happens when an agent has to take several bad decisions in order to land in a state from which he can make a really good one.

It is a common situation in board games like chess, when the optimal sequence of actions consists in sacrificing two or three pawns in order to place the opponent's king in check. The sacrifice is not considered immediately as a good decision, but it can lead to a situation where the agent can take a great advantage on his opponent.

Notice that the feedback provided by the environment is very critical in the learning process. Think about a man learning to play guitar by himself. For example, without any supervisor, he is likely to learn wrong things about how to put his fingers on the strings, until he encounters a song that he is not able to play. At this point, he will probably consider another stance for his fingers. A supervisor would have told him earlier which stance is optimal. Notice that having a bad professor teaching an incorrect stance is dangerous as well.

This example shows us that the environment is only a representation of a concrete problem, and the choice of the feedback signal is critical, just like the choice of the supervisor for the young guitarist. An expert can provide a signal, led by a concrete representation of the feedbacks (e.g. money). If such an expert does not exist, it is hard-coded within the environment. In that case, the choice of the feedback signal is a problem itself. However, this master thesis is not focusing on the transcription of a problem, but rather on the building of an agent that can learn well from a given feedback signal.

One more critical point about *Reinforcement Learning* is the exploration/exploitation dilemma. Consider a chess player, facing a given specific situation. He will make a choice and proceed until the end of the game. At this point, he is wondering if the decision he took earlier was optimal. If the answer is yes, he will associate this action to this situation. If not, he will try another one if the situation occurs again. By playing more and more, he will encounter this particular situation many times. But when can he be convinced that a particular action is the optimal choice? When does he need to stop testing other actions?

This is the exploration/exploitation dilemma. If he explores too much, he will rarely win, or miss promising areas of the state space. If he does not explore, he will make bad conclusions, leading him to a suboptimal behaviour. A trade-off between the two is therefore needed.

Another thing to consider is the case when the environment does not provide the same feedback in the same conditions. This is a common situation in real life. A child is doing something wrong. The first time, the parents will not be too harsh with him. If he repeats the same mistake again and again, the parents will punish him. Sometimes, his parents will not see him and the child will receive no punishment.

Current strategies for tackling this exploration/exploitation dilemma often give poor performances as soon as the size of the MDP starts growing and/or when dealing with MDPs with a relatively small discount factor. This is mainly due to the fact that the problem is difficult since when dealing with discounted rewards as performance criterion, an agent cannot easily recover from early mistakes.

In this thesis, we have assumed, as we believe it is the case for many practical cases, that prior information on the MDP to be played is also available. In such a context, the objective is to clearly outperform classical Reinforcement Learning techniques.

The structure of this master thesis is as follows: Chapter 2 presents the general Reinforcement Learning problem, and details formally the agent and the environment. Chapter 3 describes classical Reinforcement Learning techniques. Chapter 4 concerns a specific type of problem not well solved by usual approaches, and shows a general approach to address it. Chapter 5 exposes an application of it on a specific class of random MDPs. Chapter 6 concludes.

The work presented in this thesis has led to a scientific publication [3]. This publication is given in Appendix A and can also be downloaded from my website:

`http://www.student.montefiore.ulg.ac.be/~s070130`

Chapter 2

Problem statement

The system is composed of an agent and an environment. Let $\mathcal{S}$ be the set of states, and $\mathcal{A}$ be the set of actions. At each time-step $t \in \{0, 1, \cdots\}$, the agent stands in a state $s_t \in \mathcal{S}$. He has to choose an action $a_t \in \mathcal{A}$ to perform. The environment m gives him in return a feedback called a reward r_t, and moves the agent on a new state $s_{t+1} \in \mathcal{S}$. He collects the corresponding transition $T_m^t = (s_t, a_t, s_{t+1}, r_t) \in \mathcal{T}_m$, where $\mathcal{T}_m$ is the set of all possible transitions over m. Initially, the agent is placed in an initial state $s_0 \in \mathcal{S}$, which is imposed by the environment.

The collected transitions represent his knowledge of the system. The transitions can be collected over one or several episodes. This thesis is only focussing on the single trajectory problem. We denote by $H_t = \{T_m^0, T_m^1, \cdots, T_m^{t-1}\} \in \mathcal{H}$, the set of transitions collected by the agent in time-step t, where $\mathcal{H}$ is the set of all possible sets of transitions.

$$
\begin{aligned}
&H_0 = \emptyset \\
&H_t = H_{t-1} \cup \{T_m^{t-1}\}
\end{aligned}
$$

2.1 Agent

An agent takes decisions based on a function called a policy. It is a mapping between the information available to the agent and an action. When the agent is confronted to a given situation $s_t \in \mathcal{S}$, he chooses to perform the action $a_t \in \mathcal{A}$ given by π[1]:

$$
\begin{aligned}
&\pi(.,.) : \mathcal{H} \times \mathcal{S} \to \mathcal{A} \\
&a_t = \pi(H_t, s_t),\ H_t \in \mathcal{H}, s_t \in \mathcal{S}
\end{aligned}
$$

[1]The actual knowledge of the system obviously includes the current state. However, we thought it is more convenient to include it explicitly in the definition of π.

2.2 Environment

The environment m is defined by:

- a reward function $\rho_m(.,.) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$;

  Used to determine the feedback to return (r_t), depending on the current state (s_t) and the action chosen by the agent (a_t). If the rewards are stochastic, this function is replaced by a reward distribution.

  - **Deterministic case:**

    $$r_t = \rho_m(s_t, a_t)$$

  - **Stochastic case:**

    $$r_t \sim \rho_m(s_t, a_t)$$

  It can also depends on the next state s_{t+1} according to the considered environment. In such a case, we have in the deterministic case:

  $$r_t = \rho_m(s_t, a_t, s_{t+1})$$

  and in the stochastic case:

  $$r_t \sim \rho_m(s_t, a_t, s_{t+1})$$

- and a transition law $\tau_m(.,.) : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$.

  Used to determine the state in which the agent has to be moved on (s_{t+1}), depending on the current state (s_t) and the action chosen by the agent (a_t). It can be either deterministic or stochastic.

  - **Deterministic case:**

    $$s_{t+1} = \tau_m(s_t, a_t)$$

  - **Stochastic case:**

    $$s_{t+1} \sim \tau_m(s_t, a_t)$$

The environment is told stochastic if either the reward or the transition law is stochastic; it is called deterministic in the other case. In this thesis, we will assume that $\mathcal{S}$ and $\mathcal{A}$ are discrete sets.

2.3 Quality criterion

In order to produce a well-performing agent, there is a necessity to define precisely how to determine whether an agent is good or not. In reinforcement learning, this quality criterion is a function of the rewards collected over an optimization horizon n:

$$\text{quality criterion} \equiv R_m^\pi(r_0, r_1, \cdots, r_{n-1})$$

Moreover, in reinforcement learning, it is often assumed that this quality criterion has an additive nature such as:

$$R_m^\pi(r_0, r_1, \cdots, r_{n-1}) = \sum_{t=0}^{n} r_t$$

It is also convenient to use a discounted sum of rewards:

$$R_m^\pi(r_0, r_1, \cdots, r_{n-1}) = \sum_{t=0}^{n} \gamma^t \, r_t, \ \gamma \in [0; 1[$$

so that, when dealing with an infinite or large horizon ($n \to +\infty$), no convergence problem occurs.

Actually, we decided to focus on this latter criterion in the infinite case ($n \to +\infty$), which is one of the most studied in reinforcement learning. One of the reason that may explain why this criterion is so popular is certainly related to the fact that, for such a criterion, there exists in the set of all possible polices, a stationary one, which is optimal.

This criterion penalizes strongly early mistakes, forcing the agent to converge to an optimal policy as fast as possible. To approximate at best its value, one usually truncates the infinite discounted sum of rewards such as the desired accuracy Δ is obtained, assuming that the infinite sum is bounded:

$$\exists C \in \mathbb{R}_0^+ : \sum_{t=0}^{+\infty} \gamma^t \, r_t < C$$

This can be ensured if the rewards are themselves bounded, since $0 \leq \gamma < 1$:

$$\exists B \in \mathbb{R}_0^+ : 0 \leq r_t < B$$

In order to get the precision Δ, the horizon limit n, used to truncate the infinite sum, has to be chosen so that:

$$\begin{aligned}
&\left| \left(\sum_{t=0}^{n} \gamma^t \, r_t \right) - \left(\sum_{t=0}^{+\infty} \gamma^t \, r_t \right) \right| \leq \Delta \\
\iff & \left(\sum_{t=0}^{+\infty} \gamma^t \, r_t \right) - \left(\sum_{t=0}^{n} \gamma^t \, r_t \right) \leq \Delta \text{ (because } \gamma^t \geq 0,\ r_t \geq 0,\ \forall t) \\
\Longleftarrow & \left(\sum_{t=0}^{+\infty} \gamma^t \, B \right) - \left(\sum_{t=0}^{n} \gamma^t \, B \right) \leq \Delta \text{ (because } \exists B \in \mathbb{R}_0^+ : 0 \leq r_t < B,\ \forall t) \\
\iff & \left(\frac{B}{1-\gamma} \right) - \left(\frac{B - B\,\gamma^n}{1-\gamma} \right) \leq \Delta \\
\iff & \frac{B\,\gamma^n}{1-\gamma} \leq \Delta \\
\iff & \gamma^n \leq \frac{1-\gamma}{B}\,\Delta \\
\iff & n \leq \log_\gamma \left(\frac{1-\gamma}{B}\,\Delta \right)
\end{aligned}$$

Which is true if the horizon limit n is greater or equal to:

$$n \geq \left\lceil \log_\gamma \left(\frac{1-\gamma}{B}\,\Delta \right) \right\rceil$$

2.3.1 Optimality

Let Π be the set of all possible policies on problem m. Let J_m^π be the expected sum of rewards obtained by an agent following a policy $\pi \in \Pi$ on m:

$$J_m^\pi = \mathop{\mathbb{E}}_{m}(R_m^\pi)$$

The policy $\pi^* \in \Pi$ is optimal on m if:

$$J_m^{\pi^*} \geq J_m^\pi,\ \forall \pi \in \Pi$$

Formally, assuming a perfect knowledge of m, and knowing both the reward function/distribution ρ_m and the transition law τ_m, one can compute an optimal stationary policy π^* by first computing a value function, according to the Value Iteration algorithm [?] that works as follows:

Algorithm 1 $V_m^*(.)$ computation

```
$i \leftarrow 0$
for all $s \in \mathcal{S}$ do
  $V_m^{(0)}(s) \leftarrow 0$
end for
repeat
  for all $s \in \mathcal{S}$ do
    $V_m^{(i+1)}(s) \leftarrow \sum_{s' \in \mathcal{S}} P(\tau_m(s,a) = s') \; (\mathbb{E}[\rho_m(s,a,s')] + \gamma \; V_m^{(i)}(s'))$
  end for
  $i \leftarrow (i+1)$
until "Stopping conditions are reached"
```

We will come back on these stopping conditions in Chapter 3. The policy π^* is inferred from V^* according to:

$$\pi^*(s) = \arg\max_{a \in \mathcal{A}} \left[\sum_{s' \in \mathcal{S}} P(\tau_m(s,a) = s') \; (\mathbb{E}[\rho_m(s,a,s')] + \gamma \; V^*(s')) \right], \forall s \in \mathcal{S}$$

A variant of the Value Iteration algorithm is the Q-Iteration algorithm:

Algorithm 2 $Q_m^*(.,.)$ computation

```
$i \leftarrow 0$
for all $(s,a) \in \mathcal{S} \times \mathcal{A}$ do
  $Q_m^{(0)}(s,a) \leftarrow 0$
end for
repeat
  for all $(s,a) \in \mathcal{S} \times \mathcal{A}$ do
    $Q_m^{(i+1)}(s,a) \leftarrow \sum_{s' \in \mathcal{S}} P(\tau_m(s,a) = s') \; (\mathbb{E}[\rho_m(s,a,s')] + \gamma \max_{a' \in \mathcal{A}} Q_m^{(i)}(s',a'))$
  end for
  $i \leftarrow (i+1)$
until "Stopping conditions are reached"
```

The main advantage of the Q-function over the value function is that the optimal policy can be deduced from it, without knowing m:

$$\pi^*(s) = \arg\max_{a \in \mathcal{A}} Q^*(s,a)$$

This Q-function was first introduced by Watkins [8]. Many works in reinforcement learning have sought to compute this function from H. That is also a strategy we will adopt here to find a good exploration/exploitation strategy.

The next chapter concerns the general techniques of reinforcement learning used to compute an optimal policy.

Chapter 3

RL techniques

By taking decisions, an agent builds an historic H by accumulating the transitions provided by the environment. A common approach consists to use a *batch mode* tool, a method allowing to compute an estimation of a utility function from a given historic. First, we will describe some popular *batch mode* approaches. Then, we will discuss about how to use them efficiently to deal with the exploration/exploitation dilemma.

3.1 Batch mode

There are two main families of batch mode methods: *model-free* and *model-learning*. The first one consists to improve the current estimation of a utility function, by using directly the provided transition. The second one builds a model, through the estimation of both the reward and the transition law of the system, from which a utility function is computed.

A Q-function is the utility function generally used for the *model-free* methods. In a *model-learning* setting, it is also common to use dynamic programming, principally to extract from the model a Q-function or a value function from which a policy, which is optimal with respect to the learned model, can be extracted in a straightforward way.

However, in such a setting, other techniques, such as direct policy search techniques could also be considered. This thesis is focusing on the case where both $\mathcal{S}$ and $\mathcal{A}$ are finite.

3.1.1 Model-free approaches

This approach has many advantages. By defining an update procedure, one can improve a previous policy by simply providing a new transition. This means that there is no need to keep in memory any already used transition, leading to a memory-effective approach. However, these techniques are known to be slower than model-learning approaches [4].

One starts to compute a Q-function. Its role is to evaluate each state-action pair, which is basically a way to evaluate the quality of an action according to a state. Let Q_m be this function over problem m. A policy π_{Q_m} is computed as follows:

$$\pi_{Q_m}(s) = \arg\max_{a \in \mathcal{A}} Q_m(s,a), \forall s \in \mathcal{S}$$

The Q-function evaluates the quality of each action, according to the state from which this action is taken. Model-free approaches updates incrementally this function, whenever a new transition is provided, in order to improve the current estimation of Q^*. Many algorithms follow this scheme. They only differ by the choice of the update procedure f.

Formally, let $Q_m^{(i)}$ be the i^{th} Q-function computed over problem m. Let T_m^i be the i^{th} transition provided by the exploration component. The $Q_m^{(i+1)}$ Q-function is obtained by applying f, the update procedure, to every state-action pair:

$$\begin{aligned}
&f(.,.,.,.) : \mathcal{S} \times \mathcal{A} \times \mathbb{R} \times \mathcal{T}_m \to \mathbb{R} \\
&T_m^i = (s', a', s'', r),\ s', s'' \in \mathcal{S}, a' \in \mathcal{A}, r \in \mathbb{R} \\
&Q_m^{(i+1)}(s,a) = f(s, a, Q_m^{(i)}(s,a), T_m^i), \forall s \in \mathcal{S}, a \in \mathcal{A}
\end{aligned}$$

One has to be careful about how he chooses f, $Q_m^{(0)}$ and how he produces the T_m^i. Besides, f can also depends on i, and there is a necessity to ensure the convergence to the optimal policy π^*:

$$\pi_{Q_m^{(i \to +\infty)}} \equiv \pi^*$$

This is achieved when:

$$Q_m^{(i \to +\infty)} \equiv Q^*$$

Moreover, one can decide to apply the update procedure to all state-action pairs, or only on a few of them. The well-known Q-Learning approach is presented below to illustrate it.

3.1.1.1 Q-Learning [7]

Q-Learning consists to update the state-action pair corresponding to the given transition. Let $Q_m^{(i)}$ the current Q-function over problem m, T_M^i be the provided transition and f_i, the update procedure to use for the i^{th} update. $Q_m^{(i+1)}$ is computed as follows:

$$\begin{aligned}
T_m^i &= (s, a, s', r),\ s, s' \in \mathcal{S}, a \in \mathcal{A}, r \in \mathbb{R} \\
Q_m^{(i+1)}(s,a) &= f_i(s, a, Q_m^{(i)}(s,a), T_m^i) \\
&= Q_m^{(i)}(s,a) + \alpha^{(i)}[r + \gamma \max_{a'} Q_m^{(i)}(s', a') - Q_m^{(i)}(s,a)],\ \alpha^{(i)} \in]0;1] \\
Q_m^{(i+1)}(\bar{s}, \bar{a}) &= Q_m^{(i)}(\bar{s}, \bar{a}),\ \forall (\bar{s}, \bar{a}) \neq (s,a)
\end{aligned}$$

where $\alpha^{(0)}, \alpha^{(1)}, \cdots$ is a decreasing arithmetic series, γ is the discount factor.

There is only one state-action pair updated for each transition provided (the (s, a) pair).

Algorithm 3 Q-Learning algorithm (model-free)

```
{Initialize $Q_m^{(0)}$}
{Process}
for all $t = 0, 1, \cdots$ do
    {Retrieve transition $T_m^t = (s_t, a_t, s_{t+1}, r_t)$}
    $Q_m^{(t+1)}(s_t, a_t) \leftarrow Q_m^{(t)}(s_t, a_t) + \alpha^{(t)}[r_t + \gamma \max_{a'} Q_m^{(t)}(s_{t+1}, a') - Q_m^{(t)}(s_t, a_t)]$
end for
```

3.1.2 Model-learning approaches

A model-learning approach consists in the computation of an abstract object called a model. A model is a representation of an environment. It is computed by incrementally improving a previous model, whenever a new transition is provided.

Formally, let Ψ, the set of all possible models, $\psi_m^{(i)}$ be the current model of environment m. Let T_m^i be the i^{th} transition provided by the exploration component and g, the model update procedure. The computation of $\psi_m^{(i+1)}$ is performed as follows:

$$\begin{aligned} & g(.,.) : \Psi \times \mathcal{T}_m \rightarrow \Psi \\ & T_m^i = (s', a', s'', r),\ s', s'' \in \mathcal{S}, a' \in \mathcal{A}, r \in \mathbb{R} \\ & \psi_m^{(i+1)} = g(\psi_m^{(i)}, T_m^i) \end{aligned}$$

The second step consists in computing a V-function, evaluating a utility value for each state. It measures the utility to be in a given a state. Let h be a function computing a V-function from a model. Let $\psi_m^{(i)}$ be the i^{th} model computed. $V_{\psi_m^{(i)}}$ is defined by:

$$\begin{aligned} & h(.) : \mathcal{S} \times \Psi \rightarrow \mathbb{R} \\ & V_{\psi_m^{(i)}}(s) = h(s, \psi_m^{(i)}), \forall s \in \mathcal{S} \end{aligned}$$

The final step consists on defining a policy based on a V-function. A model is an estimation of an environment m. Let ρ_m and τ_m be respectively the reward distribution and the transition law of the environment m. Given a ψ model of m, one can compute an estimation of them, called respectively ρ_ψ and τ_ψ. A policy π_{V_ψ} defined by V_ψ, the estimation of V^* based on model ψ, is defined as follows:

$$\pi_{V_\psi}(s) = \arg\max_{a\in\mathcal{A}} \left[\sum_{s'\in\mathcal{S}} P(\tau_\psi(s,a)=s')\ (\mathbb{E}[\rho_\psi(s,a,s')] + \gamma\ V_\psi(s')) \right], \forall s \in \mathcal{S}$$

where $P(\tau_\psi(s,a) = s')$ is the probability that an agent choosing to perform action a in state s leads him to state s'.

It consists in choosing the action that leads to the most useful state (estimated by the V-function), without forgetting the immediate expected reward, balanced by the probability that such a transition occurs.

One has to ensure the convergence to the optimal policy π^*. It implies three elements:

- Considering the perfect model ψ^* of the environment m, the elements of sequence $\psi_m^{(0)}, \psi_m^{(1)}, \cdots$ of $\psi_m^{(i)}$, computed by the model update procedure g, have to converge to the perfect model when $i \to +\infty$:

  $$\psi_m^{(i\to+\infty)} \equiv \psi^*$$

  Implying that:

  $$\rho_{\psi_m^{(i\to+\infty)}} \equiv \rho_m$$
  $$\tau_{\psi_m^{(i\to+\infty)}} \equiv \tau_m$$

- The estimation of V^*, based on a model ψ_m converging to ψ_m^*, is equivalent to V^*

  $$V^*_{(\psi_m\to\psi_m^*)} \equiv V^*$$

- The policy defined by a V-function, computed using the induction procedure h, has to converge to the optimal policy π^* over environment m when the model $\psi \to \psi^*$:

  $$\pi_{V^*_{(\psi_m\to\psi_m^*)}} \equiv \pi^*$$

3.1.2.1 Value Iteration [7]

A model is learned through the computation of three functions. Let $n_a^{(i)}(.,.)$ and $n_b^{(i)}(.,.,.)$ be functions counting the number of times a state-action pair or a state-action-state triplet has been observed before the i^{th} transition T_m^i has been provided, and $c^{(i)}(.,.,.)$ be the sum of rewards observed for this triplet. The update of these functions is performed as follows:

$$
\begin{aligned}
&T_m^i = (s_i, a_i, s_{i+1}, r_i)\\
&n_a^{(i+1)}(s_i, a_i) = n_a^{(i)}(s_i, a_i) + 1\\
&n_a^{(i+1)}(s, a) = n_a^{(i)}(s, a),\ \forall (s, a) \neq (s_i, a_i)\\
&n_b^{(i+1)}(s_i, a_i, s_{i+1}) = n_b^{(i)}(s_i, a_i, s_{i+1}) + 1\\
&n_b^{(i+1)}(s, a, s') = n_b^{(i)}(s, a, s'),\ \forall (s, a, s') \neq (s_i, a_i, s_{i+1})\\
&c^{(i+1)}(s_i, a_i, s_{i+1}) = c^{(i)}(s_i, a_i, s_{i+1}) + r_i\\
&c^{(i+1)}(s, a, s') = c^{(i)}(s, a, s'),\ \forall (s, a, s') \neq (s_i, a_i, s_{i+1})
\end{aligned}
$$

From these functions, one can compute an estimation of $P(\tau_m(s,a) = s')$, the probability to observe the transition (s, a, s') and $\mathbb{E}[\rho_m(s, a, s')]$, the mean reward observed for the transition (s, a, s'), referred by $\tilde{P}^{(i)}(s, a, s')$ and $\tilde{R}^{(i)}(s, a, s')$, respectively:

$$
\begin{aligned}
\tilde{P}^{(i)}(s, a, s') &= \frac{n_b^{(i)}(s, a, s')}{n_a^{(i)}(s, a)},\ \forall s, s' \in \mathcal{S}, a \in \mathcal{A}\\
\tilde{R}^{(i)}(s, a, s') &= \frac{c^{(i)}(s, a, s')}{n_b^{(i)}(s, a, s')},\ \forall s, s' \in \mathcal{S}, a \in \mathcal{A}
\end{aligned}
$$

The Value Iteration algorithm is used to compute a V-function according to the current model $\psi^{(t)}$.

Algorithm 4 VALUE ITERATION algorithm (model-learning)

```
{Initialize $n_a^{(0)}$, $n_b^{(0)}$, $c^{(0)}$ and $V^*_{\psi_m^{(0)}}$}

{Process}
for all $t = 0, 1, \cdots$ do
  {Retrieve transition $T_m^t = (s_t, a_t, s_{t+1}, r_t)$}

  {Compute $\psi_m^{(t+1)}$}
  $n_a^{(t+1)}(s_t, a_t) \leftarrow n_a^{(t)}(s_t, a_t) + 1$
  $n_b^{(t+1)}(s_t, a_t, s_{t+1}) \leftarrow n_b^{(t)}(s_t, a_t, s_{t+1}) + 1$
  $c^{(t+1)}(s_t, a_t, s_{t+1}) \leftarrow c^{(t)}(s_t, a_t, s_{t+1}) + r_t$
  for all $(s, a, s') \in \mathcal{S} \times \mathcal{A} \times \mathcal{S}$ do
    $\tilde{P}^{(t+1)}(s, a, s') \leftarrow \frac{n_b^{(t+1)}(s,a,s')}{n_a^{(t+1)}(s,a)}$
    $\tilde{R}^{(t+1)}(s, a, s') \leftarrow \frac{c^{(t+1)}(s,a,s')}{n_b^{(t+1)}(s,a,s')}$
  end for

  {'Compute $V^*_{\psi_m^{(t+1)}}$'}
  for all $s \in \mathcal{S}$ do
    $V^{(0)}_{\psi_m^{(t+1)}}(s) \leftarrow 0$
  end for

  $i \leftarrow 0$
  repeat
    for all $s \in \mathcal{S}$ do
      $V^{(i+1)}_{\psi_m^{(t+1)}}(s) \leftarrow \max_{a \in \mathcal{A}} \left[ \sum_{s' \in \mathcal{S}} \tilde{P}^{(t+1)}(s, a, s') \; (\tilde{R}^{(t+1)}(s, a, s') + \gamma \, V^{(i)}_{\psi_m^{(t+1)}}(s')) \right]$
    end for
    $i \leftarrow i + 1$
  until "Stopping conditions are reached"

  for all $s \in \mathcal{S}$ do
    $V^*_{\psi_m^{(t+1)}}(s) \leftarrow V^{(i)}_{\psi_m^{(t+1)}}(s)$
  end for
end for
```

It is a usual *Dynamic Programming* algorithm, improving iteratively a previous function in order to converge to the optimal one. Because of the nature of such a process, one needs stopping conditions when the computed function is considered to be good enough.

There are two stopping conditions commonly used:

1. Given a precision factor θ, one can stop the process when any value of V did not change more than θ:

$$|V^{(i)}_{\psi_m^{(t+1)}}(s) - V^{(i+1)}_{\psi_m^{(t+1)}}(s)| \leq \theta,\ \forall s \in \mathcal{S}$$

2. One can stop the process after T iterations:

$$i = T$$

There are many possibilities to perform the update of the function. For example, the *Prioritized Sweeping* [1] approach consists to determine a priority among the value to be updated, which could be really useful to improve the convergence speed. Besides, one can also consider not to update every state (like in Q-Learning, only updating the value associated to the state appearing in the last transition provided).

3.2 E/E dilemma

A common way to use a batch mode technique consists to compute, at each time-step, the utility function corresponding to the current historic. Then, the inferred policy is used by the agent to choose the action to perform.

This approach is commonly called Greedy, since the exploration of the MDP is performed thanks to a suboptimal policy, which is converging an optimal one. However, is it the best way to proceed? A batch mode technique is not efficient if the provided historic is not convenient. There is a necessity to collect enough information about each state, or state-action pair, in order to be confident about the quality of the utility function computed.

If an agent has the choice between an unknown state-action pair or another state-action pair not well-known, which one has to be experienced at first? This is the exploration/exploitation dilemma, where the agent has to choose between exploiting his current knowledge, or taking the risk to explore the unknown.

3.2.1 Random

The Random approach consists to take decisions randomly. This way, all state-action pairs are tested uniformly when the number of decisions taken is big enough. The main drawback of this approach is the complete lack of control, leading to unnecessary tests of well-known state-action pairs. It is a waste of resources, which can be critical depending on the problem faced.

3.2.2 Soft-max [7]

The SOFT-MAX approach is an algorithm that chooses the action to perform randomly, by favouring the more promising ones.

A temperature parameter is used to control the impact of the utility value on the probabilities. A high temperature leads to a more uniform distribution, while a low temperature favours the best actions according to the utility function.

Formally, let $\tau \in \mathbb{R}_0^+$ be the temperature parameter. Let $u(.,.)$ be a function evaluating the quality of a state-action pair. The probability to choose action a in state s, $P_u(s,a)$, is given by:

$$u(.,.) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$$

$$P_u(s,a) = \frac{\exp\left(\frac{u(s,a)}{\tau}\right)}{\sum\limits_{a' \in \mathcal{A}} \exp\left(\frac{u(s,a')}{\tau}\right)}$$

Generally, the function $u(.,.)$ is depending on a Q-function learned in a model-free setting, or on a Q-function (or V-function) learned in a model-learning setting, given an environment m.

- **Model-free**

$$u(s,a) = Q_m(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}$$

- **Model learning**

$$u(s,a) = \sum_{s' \in \mathcal{S}} P(\tau_\psi(s,a) = s')\ (\mathbb{E}[\rho_\psi(s,a,s')] + \gamma\ V_m(s')), \forall s \in \mathcal{S}, a \in \mathcal{A}$$

  where $\rho_\psi(.,.,.)$ and $\tau_\psi(.,.)$ are computed from ψ, a model of m.

When $\tau \to +\infty$, the SOFT-MAX approach tends to the RANDOM approach. When $\tau \to 0$, the SOFT-MAX approach tends to the GREEDY approach.

3.2.3 ϵ-Greedy [7]

The ϵ-GREEDY approach consist in switching between RANDOM and GREEDY approaches. A parameter $\epsilon \in [0;1]$ is introduced. Its role is to determine which approach to use. The RANDOM approach is used with a probability of ϵ, and the GREEDY approach is used with a probability of $(1-\epsilon)$. This way, ϵ controls the exploration/exploitation balance.

For a given state, let rand(.) be the action to be performed according to a RANDOM approach, greed(.) be the action to be performed according to a GREEDY approach. The action a chosen by the ϵ-GREEDY approach when standing in state s is:

$$\begin{aligned}
&\text{rand}(.) : \mathcal{S} \rightarrow \mathcal{A} \\
&\text{greed}(.) : \mathcal{S} \rightarrow \mathcal{A} \\
&a = \text{rand}(s), \text{with a probability of } \epsilon \\
&a = \text{greed}(s), \text{with a probability of } (1 - \epsilon)
\end{aligned}$$

When $\epsilon \rightarrow 1$, the ϵ-GREEDY approach tends to the RANDOM approach.
When $\epsilon \rightarrow 0$, the ϵ-GREEDY approach tends to the GREEDY approach.

3.2.4 R-max approach [2]

The R-MAX approach is similar to a GREEDY approach, where the update of the underlying Q-function is controlled in order to manage the exploration/-exploitation balance.

Initially, the value of each state-action given by the Q-function is maximal. In this case, the agent chooses randomly which action to perform, since there is no decision better than another according to the Q-function. The transitions provided by the environment are used to update a model. When a state-action pair has been observed exactly m times, the current model is used to update the Q-function through a Q-ITERATION process. Observing again the same state-action pair will not modify the model nor the Q-function.

The agent will start to explore randomly each state-action pair. The exploration will not end until each reachable state-action pair has been tested m times, since the value computed by the Q-function is maximal when a state-action has been encountered less then m times. Then, the exploration will stop since the agent will consider there is enough data collected, and he will start to exploit what he knows.

Generally, the maximal reachable value is chosen as the discounted sum of rewards, where the rewards obtained are equal to r_{max}, the maximal reward provided by the environment

$$\sum_{t}^{+\infty} \gamma^t \, r_{max} = \frac{r_{max}}{1 - \gamma}$$

A model is learned through the computation of two functions. Let $n^{(j)}(.,.)$ be a function counting the number of times a state-action pair has been observed after the j^{th} update of the model, and $c^{(j)}(.,.)$ be the sum of rewards observed for this pair. The $(j+1)^{\text{th}}$ update of these functions is performed as follows:

$$\begin{aligned}
&T^t_m = (s_t, a_t, s_{t+1}, r_t)\\
&n^{(j+1)}(s_t, a_t) = n^{(j)}(s_t, a_t) + 1\\
&n^{(j+1)}(s, a) = n^{(j)}(s, a),\ \forall (s, a) \neq (s_t, a_t)\\
&c^{(j+1)}(s_t, a_t) = c^{(j)}(s_t, a_t) + r_t\\
&c^{(j+1)}(s, a) = c^{(j)}(s, a),\ \forall (s, a) \neq (s_t, a_t)
\end{aligned}$$

From these functions, one can compute an estimation of $\mathbb{E}[\rho_m(s, a)]$, the mean reward observed for the state-action pair (s, a) after the j^{th} update, referred by $\tilde{R}^{(j)}(s, a)$:

$$\tilde{R}^{(j)}(s, a) = \frac{c^{(j)}(s, a)}{n^{(j)}(s, a)},\ \forall s \in \mathcal{S}, a \in \mathcal{A}$$

The R-max algorithm is summarized below:

Algorithm 5 R-MAX algorithm

```
{Initialize $n^{(0)}$, $c^{(0)}$ and $Q^*_{\psi_m^{(0)}}$}
for all $(s,a) \in \mathcal{S} \times \mathcal{A}$ do
    $n^{(0)}(s,a) \leftarrow 0$
    $c^{(0)}(s,a) \leftarrow \frac{r_{max}}{1-\gamma}$
    $Q^*_{\psi_m^{(0)}}(s,a) \leftarrow \frac{r_{max}}{1-\gamma}$
end for

{Process}
$j \leftarrow 0$
for all $t = 0, 1, \cdots$ do
    {Retrieve transition $T_m^t = (s_t, a_t, s_{t+1}, r_t)$}
    if $n^{(j)}(s_t, a_t) < m$ then
        {Compute $\psi_m^{(j+1)}$}
        $n^{(j+1)}(s_t, a_t) \leftarrow n^{(j)}(s_t, a_t) + 1$
        $c^{(j+1)}(s_t, a_t) \leftarrow c^{(j)}(s_t, a_t) + r_t$
        $j \leftarrow j + 1$

        if $n^{(j)}(s_t, a_t) = m$ then
            for all $(s, a, s') \in \mathcal{S} \times \mathcal{A} \times \mathcal{S}$ do
                $\tilde{R}^{(j)}(s,a) \leftarrow \frac{c^{(j)}(s,a)}{n^{(j)}(s,a)}$
            end for

            {'Compute $Q^*_{\psi_m^{(j)}}$'}
        end if
    end if
end for
```

Algorithm 6 $Q^*_{\psi_m^{(j)}}$ computation

```
{'Compute $Q^*_{\psi_m^{(j)}}$'}
for all $s \in \mathcal{S}$ do
  $Q^{(0)}_{\psi_m^{(j)}}(s) \leftarrow \frac{r_{max}}{1-\gamma}$
end for

$i \leftarrow 0$
repeat
  for all $(s, a) \in \mathcal{S} \times \mathcal{A}$ do
    $Q^{(i+1)}_{\psi_m^{(j)}}(s, a) \leftarrow \tilde{R}^{(j)}(s, a) + \gamma \max_{a' \in \mathcal{A}} Q^{(i)}_{\psi^{(j)}m}(s_{t+1}, a')$
  end for
  $i \leftarrow (i + 1)$
until "Stopping conditions are reached"

for all $(s, a) \in \mathcal{S} \times \mathcal{A}$ do
  $Q^*_{\psi_m^{(j)}}(s, a) \leftarrow Q^{(i)}_{\psi_m^{(j)}}(s, a)$
end for
```

Chapter 4

Exploiting prior knowledge for E/E in a MDP

This chapter is focusing on how to outperform classical Reinforcement Learning techniques, when prior knowledge is available on the MDP. We suppose that this prior knowledge is encoded in the form of a probability distribution of the MDP to be played.

The approach presented here is an instantiation of the learning approach for exploration/exploitation developed this last year at the University of Liège to the case where the environment is a MDP. This approach was first tested for multi-armed bandit problems which are MDPs having only one single state [5].

This chapter is divided into two main sections. Section 4.1 carefully states the problem addressed in this thesis. Section 4.2 presents the general approach. The empirical evaluation of this approach is reported in Chapter 5.

4.1 Desired optimality

Let $H_m^i = \{T_m^0, T_m^1, \cdots, T_m^{i-1}\}$ be the historic obtained over problem m before the i^{th} decision a_i. Let e be an E/E strategy, and π^e be the policy using the E/E strategy e:

$$\begin{aligned} a_i &= \pi^e(H_m^i, s_i) \\ s_{i+1} &\sim \tau_m(s_i, a_i) \\ r_i &= \rho_m(s_i, a_i, s_{i+1}) \end{aligned}$$

where τ_m is the transition law of m, and ρ_m its reward distribution.

The return of an agent following these policies over problem m is defined by:

$$R_m^{\pi^e} = \sum_{t=1}^{+\infty} \gamma \, r_t$$

where $\gamma \in]0;1[$ is the discount factor.

The expected discounted sum of rewards over any problem m, drawn from the same distribution $p_{\mathcal{M}}(.)$, obtained by an agent using the E/E strategy e, is given by:

$$J^{\pi^e}_{p_{\mathcal{M}}(.)} = \underset{m \sim p_{\mathcal{M}}(.)}{\mathbb{E}} (R^{\pi^e}_m)$$

Let $\mathcal{E}$ be a set of E/E strategies. The E/E strategy $e^* \in \mathcal{E}$ is optimal if:

$$J^{\pi^{e^*}}_{p_{\mathcal{M}}(.)} \geq J^{\pi^e}_{p_{\mathcal{M}}(.)}, \forall e \in \mathcal{E}$$

4.2 General approach

The main idea behind our approach consists in two main phases:

1. Defining a large and rich set of E/E strategies $\mathcal{E}$.
2. Finding the best E/E strategy of $\mathcal{E}$.

Soft-max and ϵ-Greedy are two types possible strategies. The choice of the underlying induction procedure affects directly the quality of the E/E strategy used. Besides, there is many other approaches that could be referred as greedy, but where the induction procedure is dealing with the E/E dilemma by itself.

The following chapter is structured as follows: Subsection 4.2.1 describes a way to define $\mathcal{E}$, while Subsection 4.2.2 concerns the searching algorithm used to find e^* in $\mathcal{E}$.

4.2.1 Formula-based E/E strategies

Our approach consists to build a set of strategies, in which one can hope to find a promising one. The main idea is to use a formula to define an E/E strategy, which will be called a formula-based E/E strategy. By defining a rich set of formulas, one hopes to define a rich set of strategies.

Formally, let $p_{\mathcal{M}}(.)$ be the distribution of the considered class of MDPs. A formula-based E/E strategy is a strategy which will choose the most promising action on problem $m \in \mathcal{M}$ by following the return of a given formula $F_{\mathcal{M}}$, which is depending on the current historic of the system $H^{(t-1)}_m = \{T^{(1)}_m, \cdots, T^{(t-1)}_m\}$, the current time-step t, the current state s_t and an action $a \in \mathcal{A}$. Let $\pi_{F_{\mathcal{M}}}$ be the E/E strategy defined by the formula $F_{\mathcal{M}}$:

$$F_{\mathcal{M}}(.,.,.,.) : \mathcal{H}_{\mathcal{M}} \times \mathbb{N} \times \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$$
$$\pi_{F_{\mathcal{M}}}(s_t) = \underset{a \in \mathcal{A}}{\arg\max}\, F_{\mathcal{M}}(H^{(t-1)}_m, t, s_t, a)$$

where $\mathcal{H}_{\mathcal{M}}$ is the set of all possible histories on $\mathcal{M}$.

When a tie occurs, the action to perform is chosen randomly among the most promising actions. Such an approach has already been studied as an index-based strategy. In that case, the formula is an estimation the action-value function. Many algorithms are randomized versions using ϵ-GREEDY or SOFT-MAX approaches.

Defining a rich set of strategies can be achieved by defining a rich set of formulas. In this master thesis, a discrete set of formulas is studied. The main advantage of using small formulas leads to underlying interpretable strategies.

4.2.1.1 Formula definition

Let m be the environment, H_m an historic, t a strictly positive integer, s a state and a an action. Formally, a formula $F_{\mathcal{M}}$ is either:

- A binary expression: $F_{\mathcal{M}}(H_m, t, s, a) = B(F'_{\mathcal{M}}(H_m, t, s, a), F''_{\mathcal{M}}(H_m, t, s, a))$, where $F'_{\mathcal{M}}$ and $F''_{\mathcal{M}}$ are both formulas, B is a binary operator;
- A unary expression: $F_{\mathcal{M}}(H_m, t, s, a) = U(F'_{\mathcal{M}}(H_m, t, s, a))$, where $F'_{\mathcal{M}}$ is a formula, U is a unary operator;
- A constant: $F_{\mathcal{M}}(H_m, t, s, a) = C$, where C is a constant or;
- A variable: $F_{\mathcal{M}}(H_m, t, s, a) = V(H_m, t, s, a)$ where V is variable. A variable is a function depending on H_m, t, s and a.

Let $|F_{\mathcal{M}}|$ be the size of formula $F_{\mathcal{M}}$, which can be seen as the total number of operators, constants and variables appearing in it. We are trying to define $\mathbb{F}^K_{\mathcal{M}}$, the set of all formulas of size not greater than K.

Building $\mathbb{F}^K_{\mathcal{M}}$ is not our objective. Indeed, what we need is a set of formulas defining strictly different policies. It is easy to find two formulas $F^1_{\mathcal{M}}$ and $F^2_{\mathcal{M}}$ for which $\pi_{F^1_{\mathcal{M}}} \equiv \pi_{F^2_{\mathcal{M}}}$. This is why we are instead searching for $\bar{\mathbb{F}}^K_{\mathcal{M}}$, the set of all formulas of $\mathbb{F}^K_{\mathcal{M}}$ that produces strictly different E/E strategies.

4.2.1.2 $\bar{\mathbb{F}}^K_{\mathcal{M}}$, the reduction of $\mathbb{F}^K_{\mathcal{M}}$

There is a necessity to determine when two different formulas, $F^1_{\mathcal{M}}$ and $F^2_{\mathcal{M}}$, leads to equivalent strategies:

$$\pi_{F^1_{\mathcal{M}}} \equiv \pi_{F^2_{\mathcal{M}}}$$

The values computed by $F^1_{\mathcal{M}}$ and $F^2_{\mathcal{M}}$ does not matter, only the ordering implicitly defined when the action varies.

$\bar{\mathbb{F}}^K_{\mathcal{M}}$ can be obtained by partitioning the formulas of $\mathbb{F}^K_{\mathcal{M}}$ into equivalence classes. Taking one formula of each equivalence class is enough to define a valid $\bar{\mathbb{F}}^K_{\mathcal{M}}$. For optimisation concerns, we decided to take the shortest formula of each equivalence class in order to build $\bar{\mathbb{F}}^K_{\mathcal{M}}$.

This classification is not trivial: it would imply strong mathematical analysis such as commutativity, associativity, operator-specific rules and any increasing transformation. Because we cannot afford a too costly process, we are going to build $\tilde{\mathbb{F}}^K_{\mathcal{M}}$, an approximation of $\bar{\mathbb{F}}^K_{\mathcal{M}}$.

4.2.1.3 $\tilde{\mathbb{F}}^K_{\mathcal{M}}$, the approximation of $\bar{\mathbb{F}}^K_{\mathcal{M}}$

For each formula of $\bar{\mathbb{F}}^K_{\mathcal{M}}$, a key is computed according to how they rank d samples, each defining an different attribution of values for all variables. Formulas sharing the same key represents a different partition of $\mathbb{F}^K_{\mathcal{M}}$. Formally we proceed as follows:

1. Building $\mathbb{F}^K_{\mathcal{M}}$;
2. Defining randomly d attributions of values, according to the validity domain of each variable:
$$\Theta_1, \cdots, \Theta_d$$
3. The formulas of $\mathbb{F}^K_{\mathcal{M}}$ are partitioned according to the following rule: two formulas $F_{\mathcal{M}}$ and $F'_{\mathcal{M}}$ belong to the same partition if and only if they rank all the Θ_i points in the same order:
$$\forall i, j \in \{1, \ldots, d\}, i \neq j, F(\Theta_i) \geq \ F_{\mathcal{M}}(\Theta_j) \iff F'_{\mathcal{M}}(\Theta_i) \geq \ F'_{\mathcal{M}}(\Theta_j)$$
4. Selecting the shortest formula of each partition, and adding it to $\tilde{\mathbb{F}}^K_{\mathcal{M}}$.

Notice that when the set of samples corresponds to all possible attributions of values, $\tilde{\mathbb{F}}^K_{\mathcal{M}} = \bar{\mathbb{F}}^K_{\mathcal{M}}$. Moreover, choosing a good d is not easy. One could think that the greater it is, the more accurate our approximation will be. However, accuracy errors would be more and more often when d increases, which could lead to a slightly different ranking for mathematically equivalent formulas, leading to the introduction of unnecessary new partitions.

4.2.2 Search algorithm

Let assume the existence of a learning set of p samples:

$$M = \{m_1, \cdots, m_p\}$$

where $m_i \sim p_{\mathcal{M}}(.)$, $\forall i \in [1, p]$.

It represents the prior knowledge on the considered problem class. In our case, we assumed the knowledge of the problem distribution $p_{\mathcal{M}}$. A learning set of p instances has been built by simply drawing p problem instances from it. If $\mathcal{E}$ is the set of the considered E/E strategies, our goal is to find the strategy e^* for which we have:

$$J_{m \in M}^{\pi_{e^*}} \geq J_{m \in M}^{\pi_e}, \forall e \in \mathcal{E}$$

By doing so, one hopes to obtain the best possible E/E strategy not only on M, but also for any MDP that can be drawn by $p_{\mathcal{M}}(.)$.

Our approach consists to transcribe our problem into a multi-armed bandit problem, for which many good algorithms exists. A multi-armed bandit problem is a problem where the state space $\mathcal{S}$ is reduced to one single state. Maximizing the sum of rewards consists to find the best action of $\mathcal{A}$ as quickly as possible.

In our context, each E/E strategy of $\mathcal{E}$ corresponds to an action of $\mathcal{A}$:

$$\mathcal{E} = \{e_1, e_2, \cdots\}$$
$$\mathcal{A} = \{a_1, \cdots, a_{|\mathcal{E}|}\}$$
$$a_i \equiv e_i$$

Choosing the action a_i corresponds to choose to test the strategy e_i. Performing an action consists to test the E/E strategy e_i on an instance $m \in M$. The reward r, retrieved for the multi-armed bandit problem, is the discounted sum of rewards obtained by the agent, using the E/E strategy e_i:

$$r = R_m^{\pi_{e_i}}$$

Many algorithms performs well to solve the multi-armed bandit problem (such as UCB1). However, we decided to use an algorithm developed by Maes Francis [6], with a few changes to improve its efficiency. Our choice was driven by the necessity of using a fast algorithm, since the process is really costly. It appears to be a good choice since it proves to give good results, with the advantage to be *multi-threaded*, which is necessary in our application.

4.2.2.1 Search as a multi-armed bandit problem

The objective is to determine, as quickly as possible, the most promising E/E strategies of $\mathcal{E} = \{e_1, e_2, \cdots\}$ over $M = \{m_1, \cdots, m_p\}$, for which we compute an accurate estimation of $J_M^{\pi_e}$. The more promising a strategy is, the most it will be tested. A strategy is tested by using it over a problem instance $m \in M$. The E/E strategy $e_k \in \mathcal{E}$ is characterized by two variables:

- θ_k: the number of times e_k has been tested.
- μ_k: the current estimation of $J_M^{\pi_{e_k}}$.

Initially, each strategy e_k is tested over m_1. All strategies are placed on a priority queue. Each process retrieves the most promising strategy e_j remaining on the queue. The retrieved strategy is then tested over $m_{([(\theta_i+1) \mod p]+1)}$. Both θ_j and μ_j are updated before the e_j strategy is inserted back to the queue.

The most promising strategy is determined by an index-function $I_\alpha(.,.)$:

$$I_\alpha(.,.) : \mathbb{R} \times \mathbb{N} \rightarrow \mathbb{R}$$
$$I_\alpha(\mu_i, \theta_i) = \mu_i + \frac{\alpha}{\theta_i}$$

where α is a parameter controlling the exploration rate of the Search algorithm.

After T_b tests, the research is terminated. The best strategy $e^* = e_j \in \mathcal{E}$ is considered to be the one with the highest μ_j:

$$\mu_j >= \mu_i, \forall i$$

Chapter 5

Experiments

This chapter reports the results obtained by applying the approach presented in Chapter 4 on test problems. The structure of this chapter is as follows: the prior knowledge on the considered class of MDPs is presented in Section 5.1. Section 5.2 summarizes the set of E/E strategies considered, and lists the discovered formulas. Afterwards, some baseline strategies are presented in Section 5.3 and a comparison with the most promising discovered formula is performed in Section 5.4.

5.1 Prior knowledge

Let $p_{\mathcal{M}}(.)$ be the the distribution of the considered class of MDPs, $\mathcal{S} = \{s_1, s_2, \cdots\}$ be the set of states, $\mathcal{A} = \{a_1, a_2, \cdots\}$ be the set of actions, $n_{\mathcal{S}} = |\mathcal{S}|$ be the number of states and $n_{\mathcal{A}} = |\mathcal{A}|$ be the number of actions. Each state-action pair can lead to 10% of all possible states, which are chosen randomly. It implies that the transition law is stochastic. Each possible transition provides a deterministic reward between 0 and 1.

Formally, for each state-action pair $(s, a) \in \mathcal{S} \times \mathcal{A}$, we define $\mathcal{S}'_{(s,a)} \subset \mathcal{S}$, a list of states. Its size is arbitrarily fixed to $|\mathcal{S}'_{(s,a)}| = 0.1\ n_{\mathcal{S}}$. The transition law $\tau_{\mathcal{M}}(s, a)$ and the reward distribution $\rho_{\mathcal{M}}(s, a)$ are defined as follows:

$$
\begin{aligned}
&P(\tau_{\mathcal{M}}(s,a) = s') \neq 0, \forall s' \in \mathcal{S}'_{(s,a)} \\
&P(\tau_{\mathcal{M}}(s,a) = s') = 0, \forall s' \in \mathcal{S} \setminus \mathcal{S}'_{(s,a)} \\
&\rho_{\mathcal{M}}(s,a) \in]0;1], \forall s' \in \mathcal{S}'_{(s,a)} \\
&\rho_{\mathcal{M}}(s,a) = 0, \forall s' \in \mathcal{S} \setminus \mathcal{S}'_{(s,a)}
\end{aligned}
$$

The learning set M is composed of 10 000 instances, for which $n_{\mathcal{S}} = 20$, $n_{\mathcal{A}} = 5$. The discounted factor γ is set to 0.995.

5.2 Discovered formula

The chosen set of formulas $\mathbb{F}^5_{\mathcal{M}}$ has been obtained by combining, in all possible ways, at most 5 elements, which are operators, constants or variables:

- Operators
  - Unary
    $|\,.\,|$, $\sqrt{.}$, $\log(.)$
  - Binary
    $+$, $-$, $\times$, $/$, $\min(.,.)$, $\max(.,.)$
- Constants
  1, 2, 3, 5, 7
- Variables
  - $\tilde{\rho}(s,a)$: The mean reward obtained for the current state-action pair.
  - $n(s,a)$: The number of times the current state-action pair has been encountered before.
  - $\hat{Q}(s,a)$: Q-function value for the current state-action pair.
  - $\hat{V}(s)$: V-function value for the current state.
  - t: The current time-step.
  - γ^t: The current discount factor.

The V-function is computed based on a model ψ, with a precision factor $\theta = 0.001$. The Q-function is obtained from it:

$$\hat{Q}(s,a) = \sum_{s' \in \mathcal{S}} P(\tau_\psi(s,a) = s')\,(\mathbb{E}[\rho_\psi(s,a)] + \hat{V}(s'))$$

The search algorithm has been used with $\alpha = 150.0$ and $T_b = 1\,000\,000$. An E/E formula-based strategy e is evaluated by the discounted sum of rewards:

$$R^{\pi^e} = \sum_{t=1}^{n} \gamma^t\, r_t$$

where n has been chosen to ensure a precision of 0.001:

$$\begin{aligned} n &= \lceil \log_\gamma((1-\gamma)\,\Delta) \rceil \\ &= \lceil \log_{0.995}((1-0.995)\,0.001) \rceil \\ &= 2436 \end{aligned}$$

The following formulas have been discovered:

Rank	Formula
1	$(N(s,a)\ \times\ \hat{Q}(s,a))\ -\ N(s,a)$
2	$\max(1,\ (N(s,a)\ \times\ \hat{Q}(s,a)))$
3	$\hat{Q}(s,a)$ (= Greedy)
4	$\min(\gamma^t,\ (\hat{Q}(s,a)\ -\ \hat{V}(s)))$
5	$\min(\hat{\rho}(s,a),\ (\hat{Q}(s,a)\ -\ \hat{V}(s)))$

5.3 Baselines

The following techniques have been selected for the comparison:

- Optimal
  A greedy approach based on the exact model.
- Random
  Take decisions randomly.
- Greedy
  A greedy approach based on an approximated model.
- R-max $(m = 1)$
  Classical R-max technique.

To ensure an objective comparison, the RL techniques based on one or several parameters have been tuned, using the same learning set M as the one used for the discovery of the formulas. The tuning of the ϵ-Greedy algorithm led to $\epsilon = 0$, which is equivalent to the Greedy approach.

5.4 Comparison

We measured the quality of a strategy over a set of problem instances by the mean of the returns obtained over each instances of it. Five problem sets of 2 000 instances have been considered, differing by the size of their state set (10, 20, 30, 40 and 50). This way, one can observe the quality of a policy, tuned for a specific size of the state set, and measure its robustness.

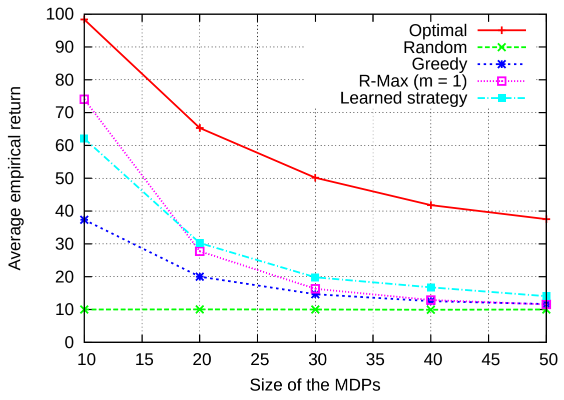

Figure 5.1: Performances of the learned and the baseline strategies for different distributions of MDPs that differ by the size of the MDPs belonging to their support.

Chapter 6

Conclusions

In this thesis, we have sought to develop an approach able to exploit prior knowledge to compute good exploration/exploitation (E/E) strategies for finite MDPs. This prior knowledge was encoded as a probability distribution over the MDP to be played. The approach works by (i) defining a rich of E/E strategies based on small interpretable formulas (ii) searching using a customized optimisation algorithm for the strategy that was working at best in average on MDPs drawn for the distribution encoding the prior knowledge.

Our tests revealed that the discovered strategy outperforms state-of-the-art reinforcement learning algorithms on MDPs drawn from the same distribution. Besides, it appears that making slight errors on the knowledge of the problem distribution does not decrease the efficiency of our approach for problem instances using a larger state space, ensuring a certain robustness. Moreover, the use of small formulas rather than "black-box functions" for representing the strategies leads to a better interpretability of the discovered strategies.

This work could be extended along several lines. First, it is worth noticing that our search algorithm can only be used in a finite configuration: the state/action spaces as well as the set of considered E/E strategies have to be finite because of the search algorithm, which is limited to the discrete case. An extension of this work to the infinite case setting would certainly be interesting. A search algorithm able to solve *multi-armed bandit problem* in the continuous case could be considered. Defining a richer sets of E/E strategies could be done by using an infinite set of formulas to define those strategies. One could imagine to parametrize the formulas, add other original variables, or simply consider a completely new type of strategies. Finally, it would also be worth improving the optimisation strategy at the heart of our approach. Indeed, the CPU time it requires for discovering a good strategy can become highly prohibitive as soon as the size of the MDP, the discount factor or the size of the formulas start growing too much.

In conclusion, this research lead to an original procedure to discover new strategies to address the original Reinforcement Learning problem. However, it certainly raises more new research questions than it has answered to.

Bibliography

[1] Chris Atkeson Andrew Moore. Prioritized sweeping: Reinforcement learning with less data and less real time. *Machine Learning*, 13:103–130, 1993.

[2] R.I. Brafman and M. Tennenholtz. R-max – a general polynomial time algorithm for near-optimal reinforcement learning. *The Journal of Machine Learning Research*, 3:213–231, 2002.

[3] Michael Castronovo, Francis Maes, Raphael Fonteneau, and Damien Ernst. Learning exploration/exploitation strategies for single trajectory reinforcement learning. In *Proceedings of the 10th European Workshop on Reinforcement Learning (EWRL 2012)*, Edinburgh, Scotland, June 30-July 1 2012.

[4] Damien Ernst. *Near Optimal Closed-Loop Control. Application to Electric Power Systems.* PhD thesis, University of Liège, Belgium, March 2003.

[5] L. Wehenkel F. Maes, R. Fonteneau and D. Ernst. Policy search in a space of simple closed-form formulas: towards interpretability of reinforcement learning. In *Submitted*, 2012.

[6] Francis Maes, Louis Wehenkel, and Damien Ernst. Automatic discovery of ranking formulas for playing with multi-armed bandits. In *9th European workshop on reinforcement learning (EWRL'11)*, Athens, Greece, September 2011.

[7] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction.* MIT Press, Cambridge, MA, USA, 1st edition, 1998.

[8] C. Watkins. *Learning from Delayed Rewards.* PhD thesis, University of Cambridge, England, 1989.

Appendix A

Publication

The work presented in this thesis has led to a research paper that has been accepted for publication in the proceedings of the 2012 *European Workshop on Reinforcement Learning.* This paper, as it will be published in these proceedings, is given hereafter.

At the end of this Appendix, we also give the comments of the reviewers we got for this paper.

Learning Exploration/Exploitation Strategies for Single Trajectory Reinforcement Learning

Michael Castronovo MICHAEL.CASTRONOVO@STUDENT.ULG.AC.BE
Francis Maes FRANCIS.MAES@ULG.AC.BE
Raphael Fonteneau RAPHAEL.FONTENEAU@ULG.AC.BE
Damien Ernst DERNST@ULG.AC.BE
Department of Electrical Engineering and Computer Science, University of Liège, BELGIUM

Editor: Marc Deisenroth

Abstract

We consider the problem of learning high-performance Exploration/Exploitation (E/E) strategies for finite Markov Decision Processes (MDPs) when the MDP to be controlled is supposed to be drawn from a known probability distribution $p_{\mathcal{M}}(\cdot)$. The performance criterion is the sum of discounted rewards collected by the E/E strategy over an infinite length trajectory. We propose an approach for solving this problem that works by considering a rich set of candidate E/E strategies and by looking for the one that gives the best average performances on MDPs drawn according to $p_{\mathcal{M}}(\cdot)$. As candidate E/E strategies, we consider index-based strategies parametrized by small formulas combining variables that include the estimated reward function, the number of times each transition has occurred and the optimal value functions $\hat{V}$ and $\hat{Q}$ of the estimated MDP (obtained through value iteration). The search for the best formula is formalized as a multi-armed bandit problem, each arm being associated with a formula. We experimentally compare the performances of the approach with R-MAX as well as with ϵ-GREEDY strategies and the results are promising.

Keywords: Reinforcement Learning, Exploration/Exploitation dilemma, Formula discovery

1. Introduction

Most Reinforcement Learning (RL) techniques focus on determining high-performance policies maximizing the expected discounted sum of rewards to come using several episodes. The quality of such a learning process is often evaluated through the performances of the final policy regardless the rewards that have been gathered during learning. Some approaches have been proposed to take these rewards into account by minimizing the undiscounted regret (Kearn and Singh (2002); Brafman and Tennenholtz (2002); Auer and Ortner (2007); Jaksch et al. (2010)), but RL algorithms have troubles solving the *original RL problem* of maximizing the expected discounted return over a single trajectory. This problem is almost intractable in the general case because the discounted nature of the regret makes early mistakes - often due to hazardous exploration - almost impossible to recover. Roughly speaking, the agent needs to learn very fast in one pass. One of the best solution to face this Exploration/Exploitation (E/E) dilemma is the R-MAX algorithm (Brafman and Tennenholtz (2002)) which combines model learning and dynamic programming with the "optimism in the face of uncertainty" principle. However, except in the case where the underlying Markov Decision Problem (MDP) comes with a small number of states and a discount factor very close to 1 (which corresponds to giving more chance to recover from bad initial decisions), the performance of R-max is still very far from the optimal (more details in Section 5).

In this paper, we assume some prior knowledge about the targeted class of MDPs, expressed in the form of a probability distribution over a set of MDPs. We propose a scheme for learning E/E strategies that makes use of this probability distribution to sample training MDPs. Note that this assumption is quite realistic, since before truly interacting with the MDP, it is often possible to have some prior knowledge concerning the number of states and actions of the MDP and/or the way rewards and transitions are distributed.

To instantiate our learning approach, we consider a rich set of candidate E/E strategies built around parametrized index-functions. Given the current state, such index-functions rely on all transitions observed so far to compute E/E scores associated to each possible action. The corresponding E/E strategies work by selecting actions that maximize these scores. Since most previous RL algorithms make use of small formulas to solve the E/E dilemma, we focus on the class of index-functions that can be described by a large set of such small formulas. We construct our E/E formulas with variables including the estimated reward function of the MDP (obtained from observations), the number of times each transition has occurred and the estimated optimal value functions $\hat{V}$ and $\hat{Q}$ (computed through off-line value iteration) associated with the estimated MDP. We then formalize the search for an optimal formula within that space as a multi-armed bandit problem, each formula being associated to an arm.

Since it assumes some prior knowledge given in the form of a probability distribution over possible underlying MDPs, our approach is related to Bayesian RL (BRL) approaches (Poupart et al. (2006); Asmuth et al. (2009)) that address the E/E trade-off by (i) assuming a prior over possible MDP models and (ii) maintaining - from observations - a posterior probability distribution (i.e., "refining the prior"). In other words, the prior is used to reduce the number of samples required to construct a good estimate of the underlying MDP and the E/E strategy itself is chosen a priori following Bayesian principles and does not depend on the targeted class of MDPs. Our approach is specific in the sense that the prior is not used for better estimating the underlying MDP but rather for identifying the best E/E strategy for a given class of targeted MDPs, among a large class of diverse strategies. We therefore follow the work of Maes et al. (2012), which already proposed to learn E/E strategies in the context of multi-armed bandit problems, which can be sought as state-less MDPs.

This paper is organized as follows. Section 2 formalizes the E/E strategy learning problem. Section 3 describes the space of formula-based E/E strategies that we consider in this paper. Section 4 details our algorithm for efficiently learning formula-based E/E strategies. Our approach is illustrated and empirically compared with R-max as well as with ϵ-Greedy strategies in Section 5. Finally, Section 6 concludes.

2. Background

Let $M = (\mathcal{S}, \mathcal{A}, p_{M,f}(\cdot), \rho_M, p_{M,0}(\cdot), \gamma)$ be a MDP. $\mathcal{S} = \left\{s^{(1)}, \dots, s^{(n_\mathcal{S})}\right\}$ is its state space and $\mathcal{A} = \left\{a^{(1)}, \dots, a^{(n_\mathcal{A})}\right\}$ its action space. When the MDP is in state s_t at time t and action a_t is selected, the MDP moves to a next state s_{t+1} drawn according to the probability distribution $p_{M,f}(\cdot|s_t, a_t)$. A deterministic instantaneous scalar reward $r_t = \rho_M(s_t, a_t, s_{t+1})$ is associated with the stochastic transition (s_t, a_t, s_{t+1}).

$H_t = [s_0, a_0, r_0, \ldots, s_t, a_t, r_t]$ is a vector that gathers the history over the first t steps and we denote by $\mathcal{H}$ the set of all possible histories of any length. An exploration / exploitation (E/E) strategy is a stochastic algorithm π that, given the current state s_t, processes at time t the vector H_{t-1} to select an action $a_t \in \mathcal{A}$: $a_t \sim \pi(H_{t-1}, s_t)$. Given the probability distribution over initial states $p_{M,0}(\cdot)$, the performance/return of a given E/E strategy π with respect to the MDP M can be defined as: $J^\pi_M = \underset{p_{M,0}(\cdot), p_{M,f}(\cdot)}{\mathbb{E}} [\mathcal{R}^\pi_M(s_0)]$ where $\mathcal{R}^\pi_M(s_0)$ is the stochastic discounted return of the E/E strategy π when starting from the state s_0. This return is defined as:

$$\mathcal{R}^\pi_M(s_0) = \sum_{t=0}^{\infty} \gamma^t r_t \ ,$$

where $r_t = \rho_M(s_t, \pi(H_{t-1}, s_t), s_{t+1})$ and $s_{t+1} \sim p_{M,f}(.|s_t, \pi(H_{t-1}, s_t))$ $\forall t \in \mathbb{N}$ and where the discount factor γ belongs to $[0, 1)$. Let $p_{\mathcal{M}}(\cdot)$ be a probability distribution over MDPs, from which we assume that the actual underlying MDP M is drawn. Our goal is to learn a high performance finite E/E strategy π given the prior $p_{\mathcal{M}}(\cdot)$, i.e. an E/E strategy that maximizes the following criterion:

$$J^\pi = \underset{M' \sim p_{\mathcal{M}}(\cdot)}{\mathbb{E}} [J^\pi_{M'}] \ . \qquad (1)$$

3. Formula-based E/E strategies

In this section, we describe the set of E/E strategies that are considered in this paper.

3.1. Index-based E/E strategies

Index-based E/E strategies are implicitly defined by maximizing history-dependent state-action index functions. Formally, we call a history-dependent state-action index function any mapping $I : \mathcal{H} \times \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. Given such an index function I, a decision can be taken at time t in the state $s_t \in \mathcal{S}$ by drawing an optimal action according to I: $\pi(H_{t-1}, s_t) \in \underset{a \in \mathcal{A}}{\arg\max}\ I(H_{t-1}, s_t, a)$[1]. Such a procedure has already been vastly used in the particular case where the index function is an estimate of the action-value function, eventually randomized using $\epsilon-$greedy or Boltzmann exploration, as in Q-Learning (Watkins and Dayan (1992)).

3.2. Formula-based E/E strategies

We consider in this paper index functions that are given in the form of small, closed-form formulas. This leads to a very rich set of candidate E/E strategies that have the advantage of being easily interpretable by humans. Formally, a formula $F \in \mathbb{F}$ is:

• either a binary expression $F = B(F', F'')$, where B belongs to a set of binary operators $\mathbb{B}$ and F' and F'' are also formulas from $\mathbb{F}$,

1. Ties are broken randomly in our experiments.

• or a unary expression $F = U(F')$ where U belongs to a set of unary operators $\mathbb{U}$ and $F' \in \mathbb{F}$,

• or an atomic variable $F = V$, where V belongs to a set of variables $\mathbb{V}$ depending on the history H_{t-1}, the state s_t and the action a,

• or a constant $F = C$, where C belongs to a set of constants $\mathbb{C}$.

Since it is high dimensional data of variable length, the history H_{t-1} is non-trivial to use directly inside E/E index-functions. We proceed as follows to transform the information contained in H_{t-1} into a small set of relevant variables. We first compute an estimated model of the MDP $\hat{M}$ that differs from the original M due to the fact that the transition probabilities and the reward function are not known and need to be learned from the history H_{t-1}. Let $\hat{P}(s,a,s')$ and $\hat{\rho}(s,a)$ be the transition probabilities and the reward function of this estimated model. $\hat{P}(s,a,s')$ is learned by computing the empirical frequency of jumping to state s' when taking action a in state s and $\hat{\rho}(s,a)$ is learned by computing the empirical mean reward associated to all transitions originating from (s,a)[2]. Given the estimated MDP, we run a value iteration algorithm to compute the estimated optimal value functions $\hat{V}(\cdot)$ and $\hat{Q}(\cdot,\cdot)$. Our set of variables is then defined as: $\mathbb{V} = \left\{\hat{\rho}(s_t,a), N(s_t,a), \hat{Q}(s_t,a), \hat{V}(s_t), t, \gamma^t\right\}$ where $N(s,a)$ is the number of times a transition starting from (s,a) has been observed in H_{t-1}.

We consider a set of operators and constants that provides a good compromise between high expressiveness and low cardinality of $\mathbb{F}$. The set of binary operators $\mathbb{B}$ includes the four elementary mathematical operations and the min and max operators: $\mathbb{B} = \{+, -, \times, \div, \min, \max\}$. The set of unary operators $\mathbb{U}$ contains the square root, the logarithm and the absolute value: $\mathbb{U} = \left\{\sqrt{\cdot}, \ln(\cdot), |\cdot|\right\}$. The set of constants is: $\mathbb{C} = \{1, 2, 3, 5, 7\}$.

In the following, we denote by π^F the E/E strategy induced by formula F:

$$\pi^F(H_{t-1}, s_t) \in \underset{a \in \mathcal{A}}{\arg\max}\ F\left(\hat{\rho}(s_t,a), N(s_t,a), \hat{Q}(s_t,a), \hat{V}(s_t), t, \gamma^t\right)$$

We denote by $|F|$ the description length of the formula F, i.e. the total number of operators, constants and variables occurring in F. Let K be a maximal formula length. We denote by $\mathbb{F}^K$ the set of formulas whose length is not greater than K. This defines our so-called set of small formulas.

4. Finding a high-performance formula-based E/E strategy for a given class of MDPs

We look for a formula F^* whose corresponding E/E strategy is specifically efficient for the subclass of MDPs implicitly defined by the probability distribution $p_{\mathcal{M}}(\cdot)$. We first describe a procedure for accelerating the search in the space $\mathbb{F}^K$ by eliminating equivalent formulas in Section 4.1. We then describe our optimization scheme for finding a high-performance E/E strategy in Section 4.2.

2. If a pair (s,a) has not been visited, we consider the following default values: $\hat{\rho}(s,a) = 0$, $\hat{P}(s,a,s) = 1$ and $\hat{P}(s,a,s') = 0, \forall s' \neq s$.

4.1. Reducing $\mathbb{F}^K$

Notice first that several formulas $\mathbb{F}^K$ can lead to the same policy. All formulas that rank all state-action pairs $(s, a) \in \mathcal{S} \times \mathcal{A}$ in the same order define the same policy. We partition the set $\mathbb{F}^K$ into equivalence classes, two formulas being equivalent if and only if they lead to the same policy. For each equivalence class, we then consider one member of minimal length, and we gather all those minimal members into a set $\bar{\mathbb{F}}^K$.

Computing the set $\bar{\mathbb{F}}^K$ is not trivial: given a formula, equivalent formulas can be obtained through commutativity, associativity, operator-specific rules and through any increasing transformation. We thus propose to approximately discriminate between formulas by comparing how they rank (in terms of values returned by the formula) a set of d random samples of the variables $\hat{\rho}(\cdot,\cdot), N(\cdot,\cdot), \hat{Q}(\cdot,\cdot), \hat{V}(\cdot), t, \gamma^t$. More formally, the procedure is the following:

• we first build $\mathbb{F}^K$, the space of all formulas such that $|F| \le K$;

• for $i = 1 \ldots d$, we uniformly draw (within their respective domains) some random realizations of the variables $\hat{\rho}(\cdot,\cdot), N(\cdot,\cdot), \hat{Q}(\cdot,\cdot), \hat{V}(\cdot), t, \gamma^t$ that we concatenate into a vector Θ_i;

• we cluster all formulas from $\mathbb{F}^K$ according to the following rule: two formulas F and F' belong to the same cluster if and only if they rank all the Θ_i points in the same order, i.e.: $\forall i, j \in \{1, \ldots, d\}, i \neq j, F(\Theta_i) \ge F(\Theta_j) \iff F'(\Theta_i) \ge F'(\Theta_j)$. Formulas leading to invalid index functions (caused for instance by division by zero or logarithm of negative values) are discarded;

• among each cluster, we select one formula of minimal length;

• we gather all the selected minimal length formulas into an approximated reduced set of formulas $\tilde{\mathbb{F}}^K$.

In the following, we denote by N the cardinality of the approximate set of formulas $\tilde{\mathbb{F}}^K = \{F_1, \ldots, F_N\}$.

4.2. Finding a high-performance formula

A naive approach for determining a high-performance formula $F^* \in \tilde{\mathbb{F}}^K$ would be to perform Monte-Carlo simulations for all candidate formulas in $\tilde{\mathbb{F}}^K$. Such an approach could reveal itself to be time-inefficient in case of spaces $\tilde{\mathbb{F}}^K$ of large cardinality.

We propose instead to formalize the problem of finding a high-performance formula-based E/E strategy in $\tilde{\mathbb{F}}^K$ as a $N-$armed bandit problem. To each formula $F_n \in \tilde{\mathbb{F}}^K$ ($n \in \{1, \ldots, N\}$), we associate an arm. Pulling the arm n consists first in randomly drawing a MDP M according to $p_{\mathcal{M}}(\cdot)$ and an initial state s_0 for this MDP according to $p_{M,0}(\cdot)$. Afterwards, an episode starting from this initial state is generated with the E/E strategy π^{F_n} until a truncated time horizon T. This leads to a reward associated to arm n whose value is the discounted return $\mathcal{R}^{\pi}_{M}(s_0)$ observed during the episode. The purpose of multi-armed bandit algorithms is here to process the sequence of such observed rewards to select in a smart way the next arm to be played so that when the budget of pulls has been exhausted, one (or several) high-quality formula(s) can be identified.

Multi-armed bandit problems have been vastly studied, and several algorithms have been proposed, such as for instance all UCB-type algorithms (Auer et al. (2002); Audibert et al. (2007)). New approaches have also recently been proposed for identifying automatically empirically efficient algorithms for playing multi-armed bandit problems (Maes et al. (2011)).

5. Experimental results

In this section, we empirically analyze our approach on a specific class of random MDPs defined hereafter.

Random MDPs. MDPs generated by our prior $p_{\mathcal{M}}(\cdot)$ have $n_{\mathcal{S}} = 20$ states and $n_{\mathcal{A}} = 5$ actions. When drawing a MDP according to this prior, the following procedure is called for generating $p_{M,f}(\cdot)$ and $\rho_M(\cdot,\cdot,\cdot)$. For every state-action pair (s,a) : (i) it randomly selects 10% of the states to form a set of successor states $Succ(s,a) \subset \mathcal{S}$ (ii) it sets $p_{M,f}(s'|s,a) = 0$ for each $s' \in \mathcal{S} \setminus Succ(s,a)$ (iii) for each $s' \in Succ(s,a)$, it draws a number $N(s')$ at random in $[0,1]$ and sets $p_{M,f}(s'|s,a) = \frac{N(s')}{\sum_{s'' \in Succ(s,a)} N(s'')}$ (iv) for each $s' \in Succ(s,a)$, it sets $\rho_M(s,a,s')$ equal to a number chosen at random in $[0,1]$ with a 0.1 probability and to zero otherwise. The distribution $p_{M,0}(\cdot)$ of initial states is chosen uniform over $\mathcal{S}$. The value of γ is equal to 0.995.

Learning protocol. In our experiments, we consider a maximal formula length of $K = 5$ and use $d = 1000$ samples to discriminate between formulas, which leads to a total number of candidate E/E strategies $N = 3834$. For solving the multi-armed bandit problem described in Section 4.2, we use an Upper Confidence Bound (UCB) algorithm (Auer et al. (2002)). The total budget allocated to the search of a high-performance policy is set to $T_b = 10^6$. We use a truncated optimization horizon $T = \log_\gamma((1-\gamma)\delta)$ for estimating the stochastic discounted return of an E/E strategy where $\delta = 0.001$ is the chosen precision (which is also used as stopping condition in the off-line value iteration algorithm for computing $\hat{Q}$ and $\hat{V}$). At the end of the T_b plays, the five E/E strategies that have the highest empirical return mean are returned.

Baselines. Our first baseline, the Optimal strategy, consists in using for each test MDP, a corresponding optimal policy. The next baselines, the Random and Greedy strategies perform pure exploration and pure exploitation, respectively. The Greedy strategy is equivalent to an index-based E/E strategy with formula $\hat{Q}(s,a)$. The last two baselines are classical E/E strategies whose parameters have been tuned so as to give the best performances on MDPs drawn from $p_{\mathcal{M}}(\cdot)$: ϵ-Greedy and R-max. For ϵ-Greedy, the best value we found was $\epsilon = 0$ in which case it behaves as the Greedy strategy. This confirms that hazardous exploration is particularly harmful in the context of single trajectory RL with discounted return. Consistently with this result, we observed that R-max works at its best when it performs the least exploration ($m = 1$).

Results. Table 1 reports the mean empirical returns obtained by the E/E strategies on a set of 2000 test MDPs drawn from $p_{\mathcal{M}}(\cdot)$. Note that these MDPs are different from those used during learning and tuning. As we can see, the best E/E strategy that has been learned performs better than all baselines (except the Optimal), including the state-of-the-art approach R-max.

We may wonder to what extent the E/E strategies found by our learning procedure would perform well on MDPs which are not generated by $p_{\mathcal{M}}(\cdot)$. As a preliminary step to answer this question, we have evaluated the mean return of our policies on sets of 2000

Baselines		Learned strategies	
Name	J^π	Formula	J^π
Optimal	65.3	$(N(s,a) \times \hat{Q}(s,a)) - N(s,a)$	30.3
Random	10.1	$\max(1, (N(s,a) \times \hat{Q}(s,a)))$	22.6
Greedy	20.0	$\hat{Q}(s,a)$ (= Greedy)	20.0
ϵ-Greedy($\epsilon = 0$)	20.0	$\min(\gamma^t, (\hat{Q}(s,a) - \hat{V}(s)))$	19.4
R-max ($m = 1$)	27.7	$\min(\hat{\rho}(s,a), (\hat{Q}(s,a) - \hat{V}(s)))$	19.4

Table 1: Performance of the top-5 learned strategies with respect to baseline strategies.

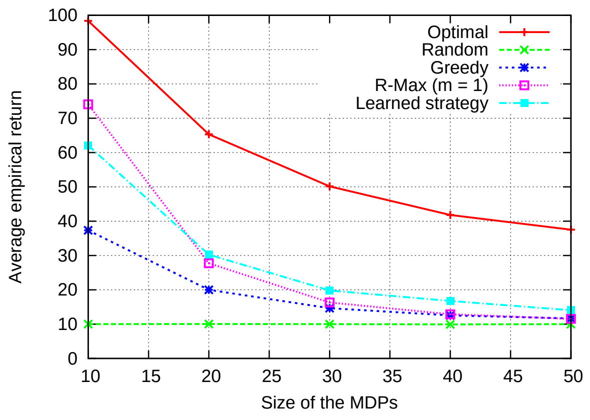

Figure 1: Performances of the learned and the baseline strategies for different distributions of MDPs that differ by the size of the MDPs belonging to their support.

MDPs drawn from slightly different distributions as the one used for learning: we changed the number of states $n_{\mathcal{S}}$ to different values in $\{10, 20, \ldots, 50\}$. The results are reported in Figure 1. We observe that, except in the case $n_{\mathcal{S}} = 10$, our best E/E strategy still performs better than the R-max and the ϵ-Greedy strategies tuned on the original distribution $p_{\mathcal{M}}(\cdot)$ that generates MDPs with 20 states. We also observe that for larger values of $n_{\mathcal{S}}$, the performances of R-max become very close to those of Greedy, whereas the performances of our best E/E strategy remain clearly above. Investigating why this formula performs well is left for future work, but we notice that it is analog to the formula $t_k(r_k - C)$ that was automatically discovered as being well-performing in the context of multi-armed bandit problems (Maes et al. (2011)).

6. Conclusions

In this paper, we have proposed an approach for learning E/E strategies for MDPs when the MDP to be controlled is supposed to be drawn from a known probability distribution $p_{\mathcal{M}}(\cdot)$. The strategies are learned from a set of training MDPs (drawn from $p_{\mathcal{M}}(\cdot)$) whose size depends on the computational budget allocated to the learning phase. Our results show that the learned strategies perform very well on test problems generated from the same distribution. In particular, they outperform on these problems R-max and ϵ-Greedy policies. Interestingly, the strategies also generalize well to MDPs that do not belong to the support of $p_{\mathcal{M}}(\cdot)$. This is demonstrated by the good results obtained on MDPs having a larger number of states than those belonging to $p_{\mathcal{M}}(\cdot)$'s support.

These encouraging results suggest several future research direction. First, it would be interesting to better study the generalization performances of our approach either theoretically or empirically. Second, we believe that our approach could still be improved by considering richer sets of formulas w.r.t. the length of the formulas and the number of variables extracted from the history. Finally, it would be worth investigating ways to improve the optimization procedure upon which our learning approach is based so as to be able to deal with spaces of candidate E/E strategies that are so large that even running once every strategy on a single training problem would be impossible.

References

J. Asmuth, L. Li, M.L. Littman, A. Nouri, and D. Wingate. A Bayesian sampling approach to exploration in reinforcement learning. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 19–26. AUAI Press, 2009.

J.Y. Audibert, R. Munos, and C. Szepesvári. Tuning bandit algorithms in stochastic environments. In *Algorithmic Learning Theory*, pages 150–165. Springer, 2007.

P. Auer and R. Ortner. Logarithmic online regret bounds for undiscounted reinforcement learning. In *Advances in Neural Information Processing Systems 19: Proceedings of the 2006 Conference*, volume 19, page 49. The MIT Press, 2007.

P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2):235–256, 2002.

R.I. Brafman and M. Tennenholtz. R-max – a general polynomial time algorithm for near-optimal reinforcement learning. *The Journal of Machine Learning Research*, 3:213–231, 2002.

T. Jaksch, R. Ortner, and P. Auer. Near-optimal regret bounds for reinforcement learning. *The Journal of Machine Learning Research*, 11:1563–1600, 2010.

M. Kearn and S. Singh. Near-optimal reinforcement learning in polynomial time. *Machine Learning*, 49:209–232, 2002.

F. Maes, L. Wehenkel, and D. Ernst. Automatic discovery of ranking formulas for playing with multi-armed bandits. In *9th European workshop on reinforcement learning*, Athens, Greece, September 2011.

F. Maes, L. Wehenkel, and D. Ernst. Learning to play K-armed bandit problems. In *International Conference on Agents and Artificial Intelligence*, Vilamoura, Algarve, Portugal, February 2012.

P. Poupart, N. Vlassis, J. Hoey, and K. Regan. An analytic solution to discrete Bayesian reinforcement learning. In *Proceedings of the 23rd international conference on Machine learning*, pages 697–704. ACM, 2006.

C.J. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8(3-4):179–192, 1992.

A.1 Reviewers' comments

```
----------------- REVIEW 1 ---------------
PAPER: 5
TITLE: Learning Exploration/Exploitation Strategies
   for Single Trajectory Reinforcement Learning
AUTHORS: Michael Castronovo, Francis Maes, Raphael
   Fonteneau and Damien Ernst

OVERALL RATING: 1 (weak accept)
REVIEWER'S CONFIDENCE: 2 (medium)

The paper does not appear to be a double submission.
I recommend that the paper be presented as a talk,
   either long or short.

The article introduce a method for finding good index
    formulas for MDPs drawn from a chosen prior over
   the class of MDPs. Such formulas are otherwise
   often designed by hand such that bounds can be
   proven. This article continues and extends work by
    Maes et. al. on finding good index formulas for
   Bandit problems. That paper by Maes et. al. used
   small collections of Bandits and I think it is far
    more satisfactory to simply draw MDPs from a
   prior as the authors of this article is doing.

The empirical results are good and the found formula
   outperforms generic methods like R-max when MDPs
   are sampled from the given prior, however, the
   authors do not compare to Bayesian RL methods
   despite referencing them and pointing out that
   they are the most closely related methods. To base
    a Bayesian RL method on the same prior would
   allow the competition the same information that is
    provided to the introduced method which would
   make it a truly fair comparison.

The paper is well presented.
```

```
------------------ REVIEW 2 ----------------
PAPER: 5
TITLE: Learning Exploration/Exploitation Strategies for Single Trajectory Reinforcement Learning
AUTHORS: Michael Castronovo, Francis Maes, Raphael Fonteneau and Damien Ernst

OVERALL RATING: 3 (strong accept)
REVIEWER'S CONFIDENCE: 4 (expert)

A Is this paper a Double-Submission?
No, I do not think so.

B If the paper gets accepted, what do you recommend:
(i)   Long Talk + poster
(ii)  Short Talk + poster
(iii) Poster

I recommend (i) Long Talk + poster.


I Summary of the Paper

The paper presents a method for learning exploration/exploitation strategies for solving
Markov Decision Processes using Reinforcement Learning. This method uses a set of formulas
that are used on the interaction history of the RL agent to compute the best action to
select. The system is trained on a large number of different MDPs taken from some prior
probability distribution, which circumvents the method to be tailored only to one specific
MDP. The results show that the approach learns exploration/exploitation strategies that
are better than R-MAX, or epsilon-greedy.


II Contribution of the Paper

The main contribution is the use of formula discovery for learning an exploration/exploitation
strategy. It is often difficult to construct new exploration/exploitation strategies, and the
proposed method allows the agent to learn novel strategies itself. This is very original.
```

```
The algorithm generates a large number of different formulas and selects to use each time
one of them using UCB. It would be interesting to use the same formula discovery also for
this multi-armed bandit problem.

III Clarity and Presentation

The paper is very well written and very interesting to read. It describes very original work
and the obtained results are very good as well.

IV Relevance and Significance

The paper is very relevant for EWRL, and has a high significance. It is interesting original
research that will be interesting for a large RL audience.


V Strengths and Weaknesses

The strengths of the paper are:

1) Very original approach
2) Very interesting for a large RL audience
3) Very well written paper and very nice to read
4) Very good results

I do not see any weaknesses, except maybe the difficulty to prove regret bounds with the
found formulas which are much more complex to analyze than humans could come up with.


VI Open Questions and Detailed Comments

I found some typos:

Page 2. and that action a_t is selected -> remove that
Page 3. we call history-dependent -> a history-dependent
Page 3. such an index functions -> function
Page 5. we clusterize -> we cluster
Page 6. \rho_M(s,a,s') -> before the reward function is based on (s,a) so: \rho_M(s,a)
Page 6. 2000 tests MPDs -> MDPs
Page 6. reported on Figure 1 -> reported in


VII Summary of the Review

Excellent paper describing very original research
```