Schedae Informaticae Vol. 24 (2015): 73–82
doi: 10.4467/20838476SI.15.007.3029

Looking for the Right Time to Shift Strategy in the Exploration-exploitation Dilemma

Filipo S. Perotto
IRIT
University of Toulouse
1 Capitole Toulouse, France
e-mail: *filipo.perotto@irit.fr*

Abstract. Balancing exploratory and exploitative behavior is an essential dilemma faced by adaptive agents. The challenge of finding a good trade-off between exploration (learn new things) and exploitation (act optimally based on what is already known) has been largely studied for decision-making problems where the agent must learn a policy of actions. In this paper we propose the *engaged climber method*, designed for solving the exploration-exploitation dilemma. The solution consists in explicitly creating two different policies (for exploring or for exploiting), and to determine the good moments to shift from the one to the other by the use of notions like engagement and curiosity.

Keywords: learning and adaptation, Markovian Decision Process, Exploration-Exploitation Dilemma.

1. Introduction

The *Exploration-Exploitation Dilemma* is a fundamental challenge faced by any kind of adaptive agent. The agent must, at the same time: (a) presenting a good performance when acting on its environment for achieving its goals, and (b) experiencing unknown or misunderstood situations in order to learn new knowledge and potentially discover better ways for achieving its goals. The problem of finding a good trade-off between exploration and exploitation is generally studied using *Reinforcement Learning* (RL) algorithms associated to *Markovian Decision Processes* (MDP). On that subjects, see, e.g., [1, 2, 3].

In model-based RL, the agent behavior stems from the models it learns. In this way, the first two questions are: (a) how to learn a model of the world? (i.e. what

methods for approximating reward and transition functions from observation)? and (*b*) how to exploit the knowledge? (i.e. what method for planning or constructing a policy of actions in order to maximize rewards?). Taken together, these two questions cause a cyclical dependence: the way the agent behaves conditions what it can learn, and what it knows conditions how it behaves. That is the seed of the exploration-exploitation dilemma, and it implies two additional questions: (*c*) how to balance exploration and exploitation? (i.e. what method for determining when to shift from one strategy to the other) and (*d*) how to explore the world? (i.e. what method for gaining knowledge efficiently?). We are concerned with the last two questions.

In this paper we approach an underdeveloped problem: deciding when to shift between exploration and exploitation. The motivation for such research comes from the intuitive notion that, in general, when the agent is following a policy, it is chaining actions, probably passing through indifferent or even penalizing states, in order to achieve some interesting rewarding state at the end of the sequence. Once engaged in a sequence of actions (that we can think of as a short-term plan) it is not very interesting to change the strategy before achieving the reward: the invested effort could be wasted. It is the same insight for exploring: when the agent spends its time seeking a situation where a new experience will allow to learn some important knowledge about the world, it should not miss such an opportunity by changing the strategy before realizing the experience.

The contribution of this paper is the *engaged climber method*, designed for governing the equilibrium between exploration and exploitation based on the concept of *policy engagement* (or *policy commitment*), bringing the notion of policy closer to the notion of plan. In the proposed solution, the agent maintains two different policies (for exploiting or exploring), based on two different estimated state utilities: the *expected reward utility* and the *expected learning utility*. A *greedy* behavior leads the agent to search for rewards, and a *curious* behavior leads it to search for knowledge. An *engagement* is associated to each behavior. The idea is to stay focused on the same strategy to be able to reach the next *peak* (a big reward, or a big discovery), then consummating the expected utility.

This paper is organized as follows: section 2 presents the RL and MDP background and the state of the art on methods for balancing exploration and exploitation; section 3 introduce the research problem, the proposed method for solving the dilemma, and preliminary experimental results comparing different methods; section 4 presents some conclusions and perspectives.

2. Reinforcement Learning

Reinforcement Learning (RL) problems within *Artificial Intelligence* are originally inspired on classic behaviorist experiments, where a sequence of actions must be learned through rewards and punishments in order to solve a maze. The computational version of that experiment corresponds to an agent governing an unknown *Markovian*

Decision Process. At each time step, the agent observes the state of the process, executes some action, and eventually receives positive or negative rewards. It must learn to coordinate its actions autonomously, by trial and error. There is no previous separated time for training: the agent needs to perform and to improve its behavior at the same time, actively and on-line, only based on its own experiences. After a sufficient number of episodes, the agent is expected to learn a policy of actions that maximizes the estimated average reward for subsequent episodes.

Model-free RL methods are able to directly learn a policy of actions that solves a given MDP without modeling it, but they need to visit each state many times to converge. In opposition, model-based learning mechanisms try to discover the effect of the agent actions on the system by incrementally constructing a model of the world. These mechanisms are able to learn a policy more efficiently because they can plan over the model.

2.1. Markovian Decision Processes

The *Markovian Decision Process* (MDP) framework constitutes a widely-used set of representations for modeling *decision-making* and *planning* problems. An MDP is typically represented as a discrete stochastic finite state machine: at each time step the machine is in some state s, the agent interacts with the process by choosing some action a to perform, then the machine changes into a new state s' and gives the agent a corresponding reward r.

In an MDP, the flow of the process (the transition between states) only depends on the system's current state and on the action taken by the agent at the time. The state of the whole process must can be represented without referring to the past. In order to represent sequential decision problems, the history of states and actions (if important) must be represented as a part of the state description. A general MDP can be formally defined as a tuple $\{S, A, T, R\}$ where:

- $S = \{s_1, s_2, ..., s_{|S|}\}$ is the finite non-empty set of system states;
- $A = \{a_1, a_2, ..., a_{|A|}\}$ is the finite non-empty set of agent actions;
- $T : S \times A \times S \rightarrow \mathbb{R}^{[0,1]}$ is the transition function, where $T(s, a, s') = Pr(s'|s, a)$;
- $R : S \times A \times S \times \mathbb{R} \rightarrow \mathbb{R}^{[0,1]}$ is the reward function, where $R(s, a, s', r) = Pr(r|s, a, s')$.

The transition function T defines the system dynamics by determining the next state s' given the current state s and the executed action a. The reward function R defines the immediate reward after transition from state s to s' with action a. A deterministic policy $\pi : S \rightarrow A$, where $\pi(s) = a$, defines the agent behavior by indicating the action a to carry out depending on the system state s.

Solving an MDP means finding the policy of actions that maximizes the rewards received by the agent over time. The optimal policy is often computationally hard to calculate, and then in practice good algorithms are expected to guarantee a near-optimal policy with high probability. When reward and transition functions are given,

the MDP can be mathematically solved by *dynamic programming*. When these elements are ignored, the MDP can be solved by *model-free RL methods*, which directly learn a policy of actions on-line. In the first case, a policy can be constructed by planning over the model, in the second case, the policy is updated after each interaction of the agent with the system. Alternatively, *model-based RL methods* constitute an intermediate solution, where the agent uses its experiences to approximate the parameters of both MDP transition and reward functions, and then solve it with some dynamic programming method, by incrementally estimating the utility of state-actions pairs and then mapping states to actions [1].

In a finite time horizon, considering n steps, the mean reward $\langle r \rangle$ can be calculated by the sum of each received reward r_t divided by the total number of time steps:

$$\langle r \rangle = \frac{\sum_{t=1}^{n} r_t}{n}. \tag{1}$$

In infinite time horizon, it is common to replace the mean reward for the discounted cumulated reward v, using a discount factor $\gamma \in \mathbb{R}^{[0,1]}$. It enables to ponder between future and immediate rewards, and that value can be calculated by:

$$v = \sum_{t=1}^{\infty} \gamma^t . r_t. \tag{2}$$

From T and R models, a policy can be calculated by *value-iteration*. That method extracts the policy from a value function V, which is iteratively updated until convergence, as follows:

$$V'(s) \leftarrow max_a \sum T(s, a, s')[R(s') + \gamma V(s')]. \tag{3}$$

2.2. Exploration-Exploitation Dilemma

Classic methods for balancing exploration and exploitation use hand-tuned parameters. Methods like *ε-greedy* and *softmax* use a fixed proportion of time to explore or to exploit. In the *ε-greedy* method, at each time step, the agent can either execute a random action with probability ε, or follow the current optimal policy otherwise. In the *softmax* method, the agent selects among the possible actions proportionally to their estimated reward utility. In this way, the likely best actions are executed with higher probabilities, but less rewarded actions are tried from time to time. A parameter τ is used to define how the difference between the estimated action utilities affect the choice. High τ or ε values result in more exploratory behavior, whereas low values cause more greedy action selection.

The problem with these basic methods is that the exploration parameter is not adjusted during the learning process. To make them more adaptable, a meta-parameter can be included to regulate the exploratory parameter [4]. It is the case for *ε-first*, that performs full exploration during a predetermined amount of time, and then switches to full exploitation. Alternatively, the *decreasing-ε* method proposes the gradual reduction of the ε parameter, so that the exploration probability decreases in a rate of

$1/n$, where n is the elapsed time (in cycles). The *softmax* method can be modified in the same way, making the τ value decrease over time. The decreasing rate is, however, fixed.

In order to dynamically regulate the ε decreasing rate, the *VDBE* [5, 6] algorithms utilize a state-dependent $\varepsilon(s)$ parameter that measures the uncertainty on the utility associated to each state in function of the fluctuations in the temporal-difference error. The idea is to make the agent more explorative on the regions of the environment where the reward estimation is still unstable, and to connect the adjustment on the exploration rate sensible to how difficult is to learn the model.

Another approach to solve the exploration-exploitation dilemma is the optimistic initialization of estimations. The idea is to be optimistic in face of uncertainty. It consists in adding a bonus utility value to the less frequent experienced state-action pairs [7]. It can be done by defining a priori optimistic reward estimations and then choosing actions greedily. Actions that lead to unobserved situations will be probably preferred. The more a situation is tried, the closer the estimated reward becomes to the true average reward. As a consequence these algorithms present an initial phase of exploration, that is gradually replaced by an exploitative behavior, as the modeled reward and transition functions converge. Such approach is used in well-established model-based methods like E^3 [8], *R-Max* [9], and *Bayesian-RL* [10]. Directly or indirectly, these methods take into account how often a state-action pair has been visited. Less visited states have an exploration bonus added to the estimated reward utility. The estimated models can be guaranteed to be probably approximately correct after a sufficient and efficient number of experiences.

2.3. Conducting exploration

Once the method for balancing exploration and exploitation is chosen, another problem is to define how to explore the world. ε or *softmax* based methods use undirected exploration [11], simply adding a degree of randomness during the action selection process. Undirected exploration means executing random actions from time to time. In this way, the agent can waste its exploring action with already well modeled state-action pairs, or it can be leaded toward known strongly penalizing states.

Directed exploration methods prefer trying less visited state-action pairs, pondered with their estimated reward utility. This is the case of *R-Max*, E^3, *Dyna*, or *Prioritized Sweeping*. Optimistic prior beliefs are associated to infrequent states in order to induce the agent to try them (if they are not too far, and if there is no early and strong negative reward evidence). Directed methods mix the exploration bonus with the reward estimation, forcing the agent to an initial exploration phase that ceases when the world is sufficiently experimented.

Another technique is to base exploration on a second-order information about the uncertainty of the estimated reward utility. The *UCB* [4] algorithm combines optimism over uncertainty with confidence bounds for each estimation. A similar strategy is used in *Bayes-Adaptive* algorithms [12], and ε-VDBE algorithms [6].

3. Designed problem and proposed solution

In this section we present the *engaged climber method*. The insight is that a short-term plan is a set of actions that leads the agent to some interesting rewarded state in the next future. A policy does the same without an explicit record of what is the goal, just by pointing at the direction of such rewarded state calculating the utility of local observable state-action pairs. Our solution preserves the spirit of model-based RL methods, but adding some information to make the policy look like a plan.

Firstly, the models $\widehat{T}$ (which estimates the transitions) and $\widehat{R}$ (which estimates the reward) must be extended to include the confidence parameter σ. $\widehat{T} : S \times A \times S \rightarrow \mathbb{R}^{[0,1]} \times \mathbb{R}^{[0,1]}$ represents the probability of the transition and the confidence on the prediction. $\widehat{R} : S \times A \times S \rightarrow \mathbb{R} \times \mathbb{R}^{[0,1]}$ represents the estimated reward and the confidence on the estimation. The value of σ is defined by the fluctuations in the temporal-difference errors, like in [5].

To calculate the policies of actions, the algorithm incrementally updates utility functions. The function $V_K : S \rightarrow \mathbb{R}$ gives the *learning utility* of a given state s. The function $V_R : S \rightarrow \mathbb{R}$ gives the *reward utility* of the state s. V_K represents the possible future gain of knowledge for s, and V_R represents the possible future average reward for s. The *engaged climber method* presents the same loop of most model-based RL methods, where the agent interacts with the environment, then uses the experience to update its models, calculate utilities, and redefine policies. The difference is that it maintains explicitly two types of utilities and policies for two distinct objectives: to explore or to exploit. The first part of the algorithm verifies if the agent should stay committed with the current strategy (exploring or exploiting), or if it should reevaluate such engagement. When the agent is not engaged, it can analyze the situation and choose the better strategy to follow from the given time.

The methods for learning the models can variate. In the machine learning literature, several algorithms are proposed for constructing models of the transition and the reward functions. In our experimental scenarios we have used naive deterministic estimators to learn $\widehat{T}$ and $\widehat{R}$ (based on the first observation). The definition of the policy is also based in classic *argmax* comparisons, i.e. for a given state s, the algorithm chooses the action a that maximizes the utility. The difference here is that we have two different utilities, depending on the running strategy (to explore or to exploit).

The *engaged climber method* chooses an strategy (exploring or exploiting) and remains committed with it. An engagement is naturally broken when the agent reaches a peak (in this case, we can consider that the strategy succeeded) or when too much time has been spent on the same strategy with no gains (timeout). When one of these situations follows, the agent redefines its strategy (taking a new engagement). When both strategies seem interesting, the method uses an ε parameter to ponder between learning or reward utilities.

Sometimes the current strategy makes no more sense (exploring when a critical alert exists, exploring when there is nothing more to discover from the current state, or exploiting when the current best policy expects negative rewards), in these cases, the engagement is forcibly broken. The *engaged climber method* considers the cumulated

Algorithm 1 Agent Lifecycle

Initialize() ▷ initialize variables and tables
loop
  if $(\pi = \pi_R)$ **and** $(V_R(s) < 0)$ **and** $(V_K(s) > 0)$ **then**
    $\pi \leftarrow \pi_K$; $\xi \leftarrow 0$ ▷ change to exploration
  else if $(\pi = \pi_K)$ **and** $(V_K(s) = 0)$
    or $(\pi = \pi_K)$ **and** $(\$ < \$_{min})$ **and** $(V_R(s) > 0)$ **then**
    $\pi \leftarrow \pi_R$; $\xi \leftarrow 0$ ▷ change to exploitation
  else if $(\xi > \xi_{max})$
    or $(\pi = \pi_R)$ **and** $(r > 0)$
    or $(\pi = \pi_K)$ **and** $(\widehat{R} \neq \widehat{R}')$ **then**
    if $(rnd() > \varepsilon)$ **then** $\pi \leftarrow \pi_R$ ▷ use epsilon
    else $\pi \leftarrow \pi_K$; $\xi \leftarrow 0$
  else ▷ unchanged strategy
    $\xi \leftarrow \xi + 1$
  end if
  $s \leftarrow World.ObserveState()$ ▷ observe current state
  $a \leftarrow \pi(s)$ ▷ select action
  $World.ExecuteAction(a)$ ▷ carry out action
  $s' \leftarrow World.ObserveState()$ ▷ observe next state
  $r \leftarrow World.GetReward()$ ▷ receive immediate reward
  $\widehat{T} \leftarrow LearnT(\widehat{T}, s, a, s')$ ▷ update transition model
  $\widehat{R} \leftarrow LearnR(\widehat{R}, s, a, s', r)$ ▷ update reward model
  $V_K \leftarrow UpdateV_K(\widehat{T}, \widehat{R})$ ▷ calculate exploration utility
  $V_R \leftarrow UpdateV_R(\widehat{T}, \widehat{R})$ ▷ calculate exploitation utility
  $\pi_K \leftarrow DefinePolicy_K(V_K)$ ▷ redefine exploration policy
  $\pi_R \leftarrow DefinePolicy_R(V_R)$ ▷ redefine exploitation policy
end loop

reward as the agent's score \$. When the agent's score becomes lower than a certain critical level, the exploitation strategy is activated in order to bring the score back to a non-critical state. When one strategy is clearly better than the other from the current state, such strategy is chosen (e.g. nothing new to discover but reward gains in sight, or inversely, no positive rewards are expected, but some state to explore on the horizon).

The utilities are updated by the methods $UpdateV_K(\widehat{T}, \widehat{R})$ and $UpdateV_R(\widehat{T}, \widehat{R})$ using an adapted *value-iteration* algorithm, following the given equations:

$$V_K(s) = max_a \left[\sum T(s, a, s'). \frac{(\sigma(s') + (V_K(s').\xi(s'))}{\xi(s') + 1} \right] \tag{4}$$

$$V_R(s) = max_a \left[\sum T(s, a, s'). \frac{(R(s') + (V_R(s').\xi(s'))}{\xi(s') + 1} \right] \tag{5}$$

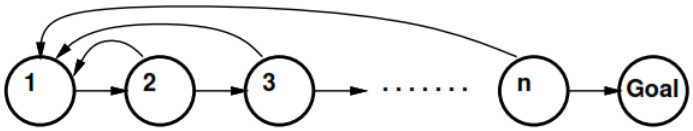

Figure 1. Chain Experiment.

3.1. Preliminary experimental results

The chain experiment [11] is a delayed reward setting where the agent must discover the unique sequence of actions that leads it to a single distant rewarded state, like illustrated in Figure 1. A single wrong action within the sequence puts the agent back to the starting point. Random exploration (the type used in the *ε-family* methods) would take $O(2^{|S|})$ steps to reach the goal even once, whereas directed exploration would require only $O(|S|^2)$ steps. If there are no costs or limited time for exploring, methods like *R-Max*, optimistic in face of uncertainty, will discover almost the entirely environment first, for then exploiting it. However, for several different problem settings, a more equilibrated balancing between exploration and exploitation is needed.

In the proposed scenario, the transition and reward functions are deterministic. The agent knows the transition model but it needs to learn the reward model. A 10-states chain gives a negative reward (−1.0) when the agent steps toward the final state, and a small positive reward (+0.5) when it takes the action that puts it back to the first state. When the agent carries out the correct sequence of actions, passing by all the states, it receives the big reward (+20.0). Thus, the way toward the goal imposes a sequence of negative rewards before arriving to the goal. The confidence value σ is set to 1 when a state-action pair is observed, and it is 0 when the transition has never been tried. The agent's score $\$ = 20 + \sum_{t=1}^{n} r_t$ are equivalent to the cumulated reward added with an initial positive credit of 20. On the *engaged climber method*, the maximum engagement ξ_{max} has been defined as 10 (equivalent to the number of states), and any positive reward has been considered as a peak. An agent's score $ lower than 10 represents a critical level. We would like to have an intelligent agent, able to quickly find a good policy, but avoiding the situation where the score becomes negative. In preliminary experiments, our *engaged climber method* reached interesting

Table 1. Experimental results.

Method	Time to discover the goal	Minimum score
ε-Greedy	≈ 1200	≈ -400
Optimistic-Greedy	≈ 60	≈ -30
Engaged-Climber	≈ 70	≈ 0

performances. It is able to outperform the classic ε-greedy algorithm simply because it uses directed exploration. When compared to classic *optimistic under uncertainty* methods, *engaged climber method* presents the advantage of managing constraints like

exploration costs. Table 1 shows the mean performances of the three methods on 20 simulations. For ε-greedy we used value-iteration with $\varepsilon = 0.99$ (almost completely random exploration). The optimistic initialization method can quickly reach the goal, but its way of exploring does not takes into account the costs. The engaged climber needs more time to reach the goal, but it is due to the management of costs, avoiding negative scores.

4. Conclusion and perspectives

Popular reinforcement learning methods approach poorly the question of what is the right time to shift between exploring and exploiting. If the agent is able to estimate the value of the information, comparing its importance to the value of next rewards, it can balance better between exploration and exploitation. It means to be able to quantify how much the agent pays in terms of reward regret in order to obtain new knowledge, and if such information is worth.

The use of exploration bonuses for infrequent states (optimism under uncertainty) makes an inherent compensation between the value of future rewards and the value of future learning. Methods such *R-Max* mix *reward utility* with *learning utility* in order to be able to compare learning costs to reward regrets. A consequence of it is a long primarily exploration phase. Depending on the problem settings, it could not be possible to wait for. It is the case for scenarios where safety and risk, costs of exploring, or even true non-determinism, are considered.

In the method proposed in this paper, a notion of *curiosity* is associated to a specific exploration policy. The agent explores the environment actively, planning and taking decisions seeking to increase its knowledge about the world dynamics. Preliminary experiments have demonstrated interesting results, even if much more experimentation in different scenarios are still needed, as well a deeper analysis in terms of algorithm complexity and robustness.

The exploration-exploitation dilemma is not a new subject, but many questions are still open. Interesting directions to further research are, for example, the difference between confidence and certainty, i.e. how to be sure of recognizing the probability distribution of a stochastic function from observation? Furthermore, the dilemma deserves more accurate studies for other kinds of settings, with different definitions of optimality.

Acknowledgements

Thanks to Sylvie Doutre, Umberto Grandi, Jean-Marc Thévénin, Laurent Perrussel (the fabulous IRIT-LILAC-UT1 research group), and also to Bruno Castro da Silva and Bertrand Saïd, for the inspiring conversations about AI.

5. References

[1] Sutton R., Barto A., *Introduction to Reinforcement Learning*. 1st edn. MIT Press, Cambridge, MA, USA, 1998.

[2] Sigaud O., Buffet O., eds. *Markov Decision Processes in Artificial Intelligence*. iSTE - Wiley, 2010.

[3] Wiering M., Otterlo M., Reinforcement learning and markov decision processes. In: *Reinforcement Learning: State-of-the-Art*. Springer, Berlin/Heidelberg 2012, pp. 3–42.

[4] Auer P., Cesa-Bianchi N., Fischer P., *Finite-time analysis of the multiarmed bandit problem*. 2002, 47(2–3), pp. 235–256.

[5] Tokic M., Adaptive epsilon-greedy exploration in reinforcement learning based on value difference. In: *KI 2010*, Berlin, Heidelberg, Springer-Verlag, 2010, pp. 203–210.

[6] Tokic M., Palm G., Value-difference based exploration: Adaptive control between epsilon-greedy and softmax. In: *KI 2011*, Berlin, Heidelberg, Springer-Verlag, 2011, pp. 335–346.

[7] Meuleau N., Bourgine P., *Exploration of multi-state environments: Local measures and back-propagation of uncertainty*. May 1999, 35(2), pp. 117–154.

[8] Kearns M., Singh S., *Near-optimal reinforcement learning in polynomial time*. November 2002, 49(2-3), pp. 209–232.

[9] Brafman R., Tennenholtz M., *R-max – a general polynomial time algorithm for near-optimal reinforcement learning*. J. Mach. Learn. Res., 2002, 3, pp. 213–231.

[10] Poupart P., Vlassis N., Hoey J., Regan K., An analytic solution to discrete bayesian reinforcement learning. In: *Proc. of the 23rd ICML*. ICML '06, New York, ACM, 2006, pp. 697–704.

[11] Kaelbling L., Littman M., Moore A., *Reinforcement learning: A survey*. May 1996, 4(1), pp. 237–285.

[12] Guez A., Silver D., Dayan P., *Scalable and efficient bayes-adaptive reinforcement learning based on monte-carlo tree search*. J. Artif. Int. Res., 2013, 48, pp. 841–883.