Algorithm selection of reinforcement learning algorithms

Romain Laroche
Microsoft Maluuba
2000 rue Peel, Montreal QC, Canada
romain.laroche@microsoft.com

Abstract

Dialogue systems rely on a careful reinforcement learning (RL) design: the learning algorithm and its state space representation. In lack of more rigorous knowledge, the designer resorts to its practical experience to choose the best option. In order to automate and to improve the performance of the aforementioned process, this article tackles the problem of online RL algorithm selection. A meta-algorithm is given for input a portfolio constituted of several off-policy RL algorithms. It then determines at the beginning of each new trajectory, which algorithm in the portfolio is in control of the behaviour during the next trajectory, in order to maximise the return. The article presents a novel meta-algorithm, called Epochal Stochastic Bandit Algorithm Selection (ESBAS). Its principle is to freeze the policy updates at each epoch, and to leave a rebooted stochastic bandit in charge of the algorithm selection. The algorithm comes with theoretical guarantees and proves to be practically efficient on a simulated dialogue task, even outperforming the best algorithm in the portfolio in most settings.

Keywords: Reinforcement Learning; Algorithm Selection; Ensemble Learning

1 Introduction

Reinforcement Learning (RL) is a machine learning framework intending to optimise the behaviour of an agent interacting with an unknown environment. For the most practical problems, trajectory collection is costly and sample efficiency is the main key performance indicator. This is for instance the case for dialogue systems [1] and robotics [2]. As a consequence, when applying RL to a new problem, one must carefully choose in advance a model, an optimisation technique, and their parameters in order to learn an adequate behaviour given the limited sample set at hand.

In particular, for 20 years [3], dialogue research has developed and applied RL algorithms, involving a large range of models and algorithms to optimise them. Just to cite a few of them: Monte Carlo, Q-Learning, SARSA, MVDP algorithms, Kalman Temporal Difference, Fitted-Q Iteration, Gaussian Process RL, and more recently Deep RL algorithms. Additionally, most of them require the setting of hyper parameters and a state space representation. When applying these research results to a new problem, these choices may dramatically affect the speed of convergence and therefore, the dialogue system performance. Facing the complexity of choice, RL and dialogue expertise is not sufficient. Confronted to the cost of data, the popular *trial and error* approach shows its limits.

Algorithm selection [4] (AS) is a framework for comparing several algorithms on a given *problem instance*. The algorithms are tested on several problem instances, and hopefully, the AS meta-algorithm learns from those experiences which algorithm should be the most efficient given a new problem instance. In our setting, only one problem instance is considered, but several experiments are led to determine the fittest algorithm to deal with it. Thus, we developed an *online* learning version [5] of AS. It consists in testing several algorithms on the task and in selecting the best one at a given time. Indeed, it is crucial to notice that, as new data is collected, the algorithms improve their performance and that an algorithm might be the worst at a short-term horizon, but the best at a longer-term horizon. For clarity, throughout the whole article, the algorithm selector is called a *meta-algorithm*, and the set of algorithms available to the meta-algorithm is called a *portfolio*. Defined as an online learning problem, the AS aims to minimise the expected regret.

At each online selection, only the selected algorithm is experienced. Since the algorithms learn from their experience, it implies a requirement for a fair budget allocation between the algorithms, so that they can be equitably evaluated and compared. Budget fairness is in direct contradiction with the expected regret minimisation objective. In order to circumvent this, the reinforcement algorithms in the portfolio are assumed to be *off-policy*, meaning that they can learn from experiences generated from an arbitrary non-stationary behavioural policy. Section 2 provides a unifying view of RL algorithms, that allows information sharing between all algorithms of the portfolio, whatever their decision processes, their state representations, and their optimisation techniques. Then, Section 3 formalises the problem of online selection of off-policy RL algorithms.

Beyond the sample efficiency issues, the online AS approach addresses furthermore four distinct practical problems for spoken dialogue systems and online RL-based systems more generally. First, it enables a systematic benchmark of models and algorithms for a better understanding of their strengths and weaknesses. Second, it improves robustness against implementation bugs: if an algorithm fails to terminate, or converges to an aberrant policy, it will be dismissed and others will be selected instead. Third, convergence guarantees and empirical efficiency may be united by covering the empirically efficient algorithms with slower algorithms that have convergence guarantees. Fourth, it enables staggered learning: shallow models converge the fastest and consequently control the policy in the early stages, while deep models discover the best solution later and control the policy in late stages.

Afterwards, Section 4 presents the Epochal Stochastic Bandit AS (ESBAS), a novel meta-algorithm addressing the online off-policy RL AS problem. Its principle is to divide the time-scale into epochs of exponential length inside which the algorithms are not allowed to update their policies. During each epoch, the algorithms have therefore a constant policy and a stochastic multi-armed bandit can be in charge of the AS with strong pseudo-regret theoretical guaranties. Finally, Section 5 describes the dialogue experiment and explains the empirical success of ESBAS, which even outperforms the best algorithm in the portfolio in most cases.

2 Unifying view of RL algorithms

The goal of this section is to enable information sharing between algorithms, even though they are considered as black boxes. We propose to share their trajectories expressed in a universal format: the *interaction process*.

Reinforcement Learning (RL) consists in learning through trial and error to control an agent behaviour in a stochastic environment. More formally, at each time step $t \in \mathbb{N}$, the agent performs an action $a(t) \in \mathcal{A}$, and then perceives from its environment a signal $o(t) \in \Omega$ called observation, and receives a reward $R(t) \in \mathbb{R}$. Figure 1 illustrates the RL framework. This interaction process is not Markovian: the agent may use an internal memory.

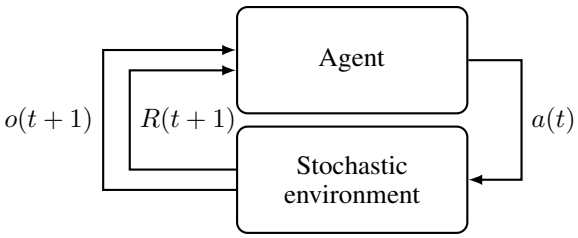


Figure 1: RL framework: after performing action $a(t)$, the agent perceives observation $o(t+1)$ and receives reward $R(t+1)$.

In this article, the reward function is assumed to be bounded between R_{min} and R_{max}, and we define the RL problem as episodic. Let us introduce two time scales with different notations. First, let us define *meta-time* as the time scale for AS: at one meta-time τ corresponds a meta-algorithm decision, *i.e.* the choice of an algorithm and the generation of a full episode controlled with the policy determined by the chosen algorithm. Its realisation is called a *trajectory*. Second, *RL-time* is defined as the time scale inside a trajectory, at one RL-time t corresponds one triplet composed of an observation, an action, and a reward.

Let $\mathscr{E}$ denote the space of trajectories. A *trajectory* $\varepsilon_\tau \in \mathscr{E}$ collected at meta-time τ is formalised as a sequence of (observation, action, reward) triplets: $\varepsilon_\tau = \langle o_\tau(t), a_\tau(t), R_\tau(t)\rangle_{t\in[\![1,|\varepsilon_\tau|]\!]} \in \mathscr{E}$, where $|\varepsilon_\tau|$ is the length of trajectory ε_τ. The *trajectory set* at meta-time T is denoted by: $\mathcal{D}_T = \{\varepsilon_\tau\}_{\tau\in[\![1,T]\!]} \in \mathscr{E}^T$. A sub-trajectory of ε_τ until RL-time t is called the *history* at RL-time t and written $\varepsilon_\tau(t)$ with $t \leq |\varepsilon_\tau|$. The history records what happened in episode ε_τ until RL-time t: $\varepsilon_\tau(t) = \langle o_\tau(t'), a_\tau(t'), R_\tau(t')\rangle_{t'\in[\![1,t]\!]} \in \mathscr{E}$.

The objective is, given a discount factor $0 \leq \gamma < 1$, to generate trajectories with high discounted cumulative reward, also called *return*, and noted $\mu(\varepsilon_\tau) = \sum_{t=1}^{|\varepsilon_\tau|} \gamma^{t-1} R_\tau(t)$. Since $\gamma < 1$ and $R_{min} \leq R \leq R_{max}$, this return is bounded. The goal of each RL algorithm α is therefore to find a policy $\pi^* : \mathscr{E} \to \mathcal{A}$ which yields optimal expected returns. Such an algorithm α is viewed as a black box that takes as an input a trajectory set $\mathcal{D} \in \mathscr{E}^+$, where $\mathscr{E}^+$ is the ensemble of trajectory sets of undetermined size: $\mathscr{E}^+ = \bigcup_{T\in\mathbb{N}} \mathscr{E}^T$, and that outputs a policy $\pi^\alpha_\mathcal{D}$, formalised as follows: $\alpha : \mathscr{E}^+ \to (\mathscr{E} \to \mathcal{A})$.

Such a high level definition of the RL algorithms allows to share trajectories between algorithms: a history as a sequence of observations, actions, and rewards can be interpreted by any algorithm in its own decision process and state representation. For instance, RL algorithms classically rely on an MDP defined on a state space representation $\mathcal{S}^\alpha_\mathcal{D}$ thanks to a projection $\Phi^\alpha_\mathcal{D} : \mathscr{E} \to \mathcal{S}^\alpha_\mathcal{D}$.

The state representation may be built dynamically as the trajectories are collected. Many algorithms doing so can be found in the literature [6]. Then, α may learn its policy $\pi^\alpha_{\mathcal{D}_T}$ from the trajectories projected on its state space representation and saved as a transition set: a transition is defined as a quadruplet $\langle s_\tau(t), a_\tau(t), R_\tau(t), s_\tau(t+1)\rangle$, with state $s_\tau(t) \in \mathcal{S}^\alpha_\mathcal{D}$, action $a_\tau(t) \in \mathcal{A}$, reward $R_\tau(t) \in \mathbb{R}$, and next state $s_\tau(t+1) \in \mathcal{S}^\alpha_\mathcal{D}$. Off-policy RL optimisation techniques compatible with this approach are numerous in the literature [7]: Q-Learning, Fitted-Q Iteration, Kalman Temporal Difference, etc. Another option would be to perform end-to-end RL [8]. As well, any post-treatment of the state set, any alternative decision process model, such as POMDPs, and any off-policy technique for control optimisation may be used. The algorithms are defined here as black boxes and the considered meta-algorithms will be indifferent to how the algorithms compute their policies.

3 Algorithm selection model

Algorithm selection [4] for combinatorial search consists in deciding which experiments should be carried out, given a problem instance and a fixed amount of computational resources: generally speaking computer-time, memory resources, time and/or money. Algorithms are considered efficient if they consume little resource. AS applied to machine learning, also called *meta-learning*, is mostly dedicated to error minimisation given a corpus of limited size. In the classical batch setting, the notations of the machine learning AS problem are described in [9] as follows: $\mathcal{I}$ is the space of problem instances; $\mathcal{P}$ is the *portfolio*, *i.e.* the collection of available algorithms; $\mu : \mathcal{I} \times \mathcal{P} \to \mathbb{R}$ is the *objective function*, *i.e.* a performance metrics enabling to rate an algorithm on a given instance; and $\Psi : \mathcal{I} \to \mathbb{R}^k$ are the features characterising the properties of problem instances. The principle consists in collecting problem instances and in solving them with the algorithms in the portfolio $\mathcal{P}$. The measures μ provide evaluations of the algorithms on those instances. Then, the aggregation of their features Ψ with the measures μ constitutes a training set. Finally, any supervised learning techniques can be used to learn an optimised mapping between the instances and the algorithms.

In our case, $\mathcal{I}$ is not large enough to learn an efficient model, it might even be a singleton. Consequently, it is not possible to regress a general knowledge from a parametrisation Ψ. This is the reason why the online learning approach is tackled in this article: different algorithms are experienced and evaluated during the data collection. Since it boils down to a classical exploration/exploitation trade-off, multi-armed bandit have been used for combinatorial search AS [5]. In the online setting, the AS problem for off-policy RL is new and we define it as follows: $\mathcal{D} \in \mathscr{E}^+$ is the *trajectory set*; $\mathcal{P} = \{\alpha^k\}_{k\in[\![1,K]\!]}$ is the *portfolio*; $\mu : \mathscr{E} \to \mathbb{R}$ is the *objective function*.

Pseudo-code 1 formalises the online AS setting. A *meta-algorithm* is defined as a function from a trajectory set to the selection of an algorithm: $\sigma : \mathscr{E}^+ \to \mathcal{P}$. The meta-algorithm is queried at each meta-time $\tau = |\mathcal{D}_{\tau-1}|+1$, with input $\mathcal{D}_{\tau-1}$, and it ouputs algorithm $\sigma(\mathcal{D}_{\tau-1}) = \sigma(\tau) \in \mathcal{P}$ controlling with its policy $\pi^{\sigma(\tau)}_{\mathcal{D}_{\tau-1}}$ the generation of the trajectory $\varepsilon^{\sigma(\tau)}_\tau$ in the stochastic environment. The final goal is to optimise the cumulative expected return. It is the expectation of the sum of rewards obtained after a run of T trajectories: $\mathbb{E}_\sigma\left[\sum_{\tau=1}^T \mu\left(\varepsilon^{\sigma(\tau)}_\tau\right)\right]$.

Pseudo-code 1 Online AS setting

Data: $\mathcal{P} \leftarrow \{\alpha^k\}_{k\in[\![1,K]\!]}$: algorithm portfolio
Data: $\mathcal{D}_0 \leftarrow \emptyset$: trajectory set
for $\tau \leftarrow 1$ **to** ∞ **do**
  Select $\sigma(\mathcal{D}_{\tau-1}) = \sigma(\tau) \in \mathcal{P}$;
  Generate trajectory ε_τ with policy $\pi^{\sigma(\tau)}_{\mathcal{D}_{\tau-1}}$;
  Get return $\mu(\varepsilon_\tau)$;
  $\mathcal{D}_\tau \leftarrow \mathcal{D}_{\tau-1} \cup \{\varepsilon_\tau\}$;
end

Nota bene: there are three levels of decision: meta-algorithm σ selects algorithm α that computes policy π that is in control. The focus is on the meta-algorithm level.

The *fairness of budget distribution* has been formalised in [10]. It is the property stating that every algorithm in the portfolio has as much resources as the others, in terms of computational time and data. It is an issue in most online AS problems, since the algorithm that has been the most selected has the most data, and therefore must be the most advanced one. A way to solve this issue is to select them equally, but, in an online setting, the goal of AS is precisely to select the best algorithm as often as possible. In short, exploration and evaluation require to be fair and exploitation implies to be unfair. Our answer is to require that all algorithms in the portfolio are learning *off-policy*, *i.e.* without bias induced by the behavioural policy used in the learning dataset. By assuming that all algorithms learn off-policy, we allow *information sharing* [10] between algorithms. They share the trajectories they generate. As a consequence, we can assume that each algorithm, the least or the most selected one, learns from the same trajectory set. Therefore, the control

unbalance does not directly lead to unfairness in algorithms performances: all algorithms learn equally from all trajectories. However, unbalance might still remain in the exploration strategy if, for instance, an algorithm takes more benefit from the exploration it has chosen than the one chosen by another algorithm. Here, we speculate that this chosen-exploration side effect is negligible.

4 Epochal stochastic bandit algorithm selection

An intuitive way to solve the AS problem is to consider algorithms as arms in a multi-armed bandit setting. The bandit meta-algorithm selects the algorithm controlling the next trajectory ε and the trajectory return $\mu(\varepsilon)$ constitutes the reward of the bandit. However, a stochastic bandit cannot be *directly* used because the algorithms performances vary and improve with time. Adversarial multi-arm bandits are designed for non-stationary environments [11], but the exploitation the structure of the AS problem makes it possible to obtain pseudo-regrets of order of magnitude lower than $\Omega(\sqrt{T})$. Alternatively, one might consider using policy search methods [12], but they rely on a state space representation in order to apply the policy gradient. And in our case, neither policies do share any, nor the meta-algorithm does have at disposal any other than the intractable histories $\varepsilon_\tau(t)$.

Pseudo-code 2 ESBAS with UCB1

Data: $\mathcal{P} \leftarrow \{\alpha^k\}_{k\in[\![1,K]\!]}$
Data: $\mathcal{D}_0 \leftarrow \emptyset$
for $\beta \leftarrow 0$ **to** ∞ **do**
  for $\alpha^k \in \mathcal{P}$ **do**
    $\pi^k_{\mathcal{D}_{2^\beta-1}}$: policy learnt by α^k on $\mathcal{D}_{2^\beta-1}$
  end
  $n \leftarrow 0$, $\forall \alpha^k \in \mathcal{P}$, $n^k \leftarrow 0$, and $x^k \leftarrow 0$
  for $\tau \leftarrow 2^\beta$ **to** $2^{\beta+1}-1$ **do**
    $\alpha^{k_{max}} = \underset{\alpha^k\in\mathcal{P}}{\operatorname{argmax}} \left(x^k + \sqrt{\xi \frac{\log(n)}{n^k}} \right)$
    Generate trajectory ε_τ with policy $\pi^{k_{max}}_{\mathcal{D}_{2^\beta-1}}$
    Get return $\mu(\varepsilon_\tau)$, $\mathcal{D}_\tau \leftarrow \mathcal{D}_{\tau-1} \cup \{\varepsilon_\tau\}$
    $x^{k_{max}} \leftarrow \frac{n^{k_{max}} x^{k_{max}} + \mu(\varepsilon_\tau)}{n^{k_{max}}+1}$
    $n^{k_{max}} \leftarrow n^{k_{max}} + 1$ and $n \leftarrow n+1$
  end
end

To solve the off-policy RL AS problem, we propose a novel meta-algorithm called Epochal Stochastic Bandit AS (ESBAS). Because of the non-stationary algorithms' learning, the stochastic bandit cannot directly select algorithms. Instead, the stochastic bandit can choose among fixed policies. To comply with this constraint, the meta-time scale is divided into epochs inside which the algorithms policies are not allowed to be updated: the algorithms optimise their policies only when epochs start, in such a way that the policies are constant inside each epoch. As a consequence, at each new epoch, the problem can be cast into an independent stochastic K-armed bandit Ξ, with $K = |\mathcal{P}|$. The ESBAS meta-algorithm is formally sketched in Pseudo-code 2 with UCB1 as the stochastic K-armed bandit Ξ. The meta-algorithm takes as an input the set of algorithms in the portfolio. Meta-time scale is fragmented into epochs of exponential size. The β^{th} epoch lasts 2^β meta-time steps, so that, at meta-time $\tau = 2^\beta$, epoch β starts. At the beginning of each epoch, the ESBAS meta-algorithm asks each algorithm in the portfolio to update their current policy. Inside an epoch, the policy is never updated anymore. At the beginning of each epoch, a new Ξ instance is reset and run. During the whole epoch, Ξ decides at each meta-time step which algorithm is in control of the next trajectory.

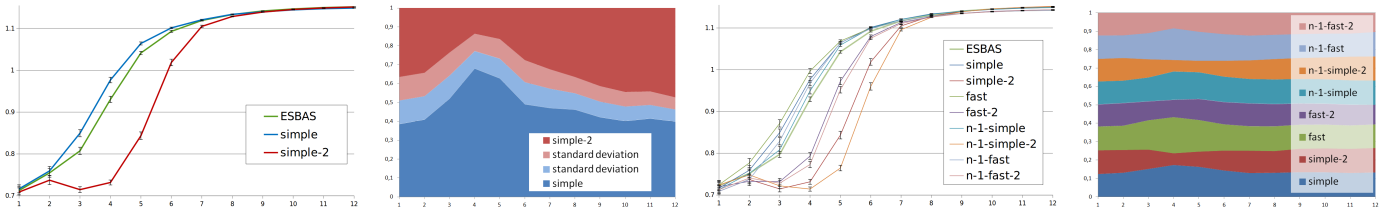
Under some assumptions on the algorithms learning behaviour, near tight bounds for the ESBAS meta-algorithm have been derived, in function of the order of magnitude of the asymptotic convergence of the algorithms in the portfolio. Please refer to [13] for details.

5 Dialogue experiment

ESBAS algorithm for off-policy RL AS may and was meant to be applied to dialogue systems. Thus, its practical efficiency is illustrated on a dialogue negotiation game [14] that involves two players: the system p_s and a user p_u. Their goal is to reach an agreement. 4 options are considered, and at each new dialogue, for each option η, players have a private uniformly drawn cost $\nu^p_\eta \sim \mathcal{U}[0,1]$ to agree on it. Each player is considered fully empathetic to the other one. As a result, if the players come to an agreement, the system's immediate reward at the end of the dialogue is: $R^{p_s}(s_f) = 2 - \nu^{p_s}_\eta - \nu^{p_u}_\eta$, where s_f is the last state reached by player p_s at the end of the dialogue, and η is the agreed option; if the players fail to agree, the final immediate reward is: $R^{p_s}(s_f) = 0$; and finally, if one player misunderstands and agrees on a wrong option, the system gets the cost of selecting option η without the reward of successfully reaching an agreement: $R^{p_s}(s_f) = -\nu^{p_s}_\eta - \nu^{p_u}_{\eta'}$.

Players act each one in turn, starting randomly by one or the other. They have four possible actions: REFPROP(η): the player makes a proposition: option η, if there was any option previously proposed by the other player, the player refuses it; ASKREPEAT: the player asks the other player to repeat its proposition; ACCEPT(η): the player accepts option η that was understood to be proposed by the other player, this act ends the dialogue either way: whether the understood proposition was the right one or not; ENDDIAL: the player does not want to negotiate anymore and ends the dialogue with a null reward. Understanding through speech recognition of system p_s is assumed to be noisy: with a sentence error rate of probability $SER^u_s = 0.3$, an error is made. Please, refer to [14, 13] for more details.

The performance figures plot the curves of algorithms individual performance against the ESBAS portfolio control in function of the epoch (the scale is therefore logarithmic in meta-time). The performance is the average return $\gamma^{|\epsilon|} R^{p_s}(s_f)$. The ratio figures plot the average algorithm selection proportions of ESBAS at each epoch. Two experience results are presented in this extended abstract. More can be found in [13].



(a) *simple* vs *simple-2*: performance (b) *simple* vs *simple-2*: ratios (c) 8 learners: performance (d) 8 learners: ratios

Figure 2: The figures on the left plot the performance over time. The learning curves display the average performance of each algorithm/portoflio over the epochs. The figures on the right show the ESBAS selection ratios over the epochs. Figure 2b also shows the selection ratio standard deviation from one run to another around the mean which is at the frontier between the blue and the red areas. Since the abscissæ are in epochs, all figure are logarithmic in meta-time. As a consequence, Epoch 12 is 1024 times larger than Epoch 2, and this explains for instance that *simple-2* has actually a higher cumulative returns over the 12 epochs than *simple*.

Figures 2a and 2b plot the typical curves obtained with ESBAS selecting from a portfolio of two learning algorithms. On Figure 2a, the ESBAS curve tends to reach more or less the best algorithm in each point as expected. Surprisingly, Figure 2b reveals that the algorithm selection ratios are not very strong in favour of one or another at any time. Indeed, the variance in trajectory set collection makes *simple* better on some runs until the end. ESBAS proves to be efficient at selecting the best algorithm for each run and although the ESBAS meta-algorithm does not intend to do better than the best algorithm in the portfolio, it obtains a significantly higher performance than the best algorithm's.

ESBAS also performs well on larger portfolios of 8 learners (see Figure 2c), even if the algorithms are, on average, almost selected uniformly as Figure 2d reveals. ESBAS offers some staggered learning, but more importantly, early bad policy accidents in learners are avoided. The same kind of results are obtained with 4-learner portfolios.

We interpret ESBAS's success at reliably outperforming the best algorithm in the portfolio as the result of four following potential added values: First, **calibrated learning**: ESBAS selects the algorithm that is the most fitted with the data size. This property allows for instance to use shallow algorithms when having only a few data and deep algorithms once collected a lot. Second, **diversified policies**: ESBAS computes and experiments several policies. Those diversified policies generate trajectories that are less redundant, and therefore more informational. As a result, the later policies trained on these diverse trajectories should be more efficient. Third, **robustness**: if one algorithm learns a terrible policy, it will soon be left apart until the next policy update. This property prevents the agent from repeating again and again the same blatant mistakes. Fourth, **run adaptation**: obviously, there has to be an algorithm that is the best on average for one given task at one given meta-time. But depending on the variance in the trajectory collection, it is not necessarily the best one for each run. The ESBAS meta-algorithm tries and selects the algorithm that is the best at each run.

These properties relate to the ensemble RL ones. [15] uses combinations of a set of RL algorithms to build its online control such as policy voting or value function averaging. This approach shows good results when all the algorithms are efficient, but not when some of them are underachieving. Hence, no convergence bound has been proven with this family of meta-algorithms when some algorithms are underachieving. In comparison, ESBAS policy does not emerge from a consensus between the learners, but rather empirically evaluates them to keep the best one(s). We invite once more the interested reader to consult the full paper [13].

References

[1] Satinder P. Singh, Michael J. Kearns, Diane J. Litman, and Marilyn A. Walker. Reinforcement learning for spoken dialogue systems. In *Proceedings of the 13th Annual Conference on Neural Information Processing Systems (NIPS)*, 1999.

[2] Tim Brys, Anna Harutyunyan, Halit Bener Suay, Sonia Chernova, Matthew E Taylor, and Ann Nowé. Reinforcement learning from demonstration through shaping. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI)*, 2015.

[3] Esther Levin and Roberto Pieraccini. A stochastic model of computer-human interaction for learning dialogue strategies. In *Proceedings of the 5th European Conference on Speech Communication and Technology (Eurospeech)*, 1997.

[4] John R. Rice. The algorithm selection problem. *Advances in Computers*, 1976.

[5] Matteo Gagliolo and Jürgen Schmidhuber. Learning dynamic algorithm portfolios. *Annals of Mathematics and Artificial Intelligence*, 2006.

[6] Wendelin Böhmer, Jost Tobias Springenberg, Joschka Boedecker, Martin Riedmiller, and Klaus Obermayer. Autonomous learning of state representations for control: An emerging field aims to autonomously learn state representations for reinforcement learning agents from their real-world sensor observations. *KI-Künstliche Intelligenz*, 2015.

[7] Hamid R Maei, Csaba Szepesvári, Shalabh Bhatnagar, and Richard S Sutton. Toward off-policy learning control with function approximation. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 719–726, 2010.

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

[9] Kate A. Smith-Miles. Cross-disciplinary perspectives on meta-learning for algorithm selection. *ACM Computational Survey*, 2009.

[10] Marie-Liesse Cauwet, Jialin Liu, Baptiste Rozière, and Olivier Teytaud. Algorithm Portfolios for Noisy Optimization. *ArXiv e-prints*, November 2015.

[11] Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E Schapire. The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing*, 32(1):48–77, 2002.

[12] Andrew Y Ng and Michael Jordan. Pegasus: A policy search method for large mdps and pomdps. In *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*, pages 406–415. Morgan Kaufmann Publishers Inc., 2000.

[13] Romain Laroche and Raphaël Féraud. Algorithm selection of off-policy reinforcement learning algorithm. *arXiv preprint arXiv:1701.08810*, 2017.

[14] Romain Laroche and Aude Genevay. A negotiation dialogue game. In *Proceedings of the 7th International Workshop on Spoken Dialogue Systems (IWSDS)*, Finland, 2016.

[15] Marco A Wiering and Hado Van Hasselt. Ensemble algorithms in reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 38(4):930–936, 2008.