Configurable Environments in Reinforcement Learning: An Overview

Alberto Maria Metelli

Abstract Reinforcement Learning (RL) has emerged as an effective approach to address a variety of complex control tasks. In a typical RL problem, an agent interacts with the environment by perceiving observations and performing actions, with the ultimate goal of maximizing the cumulative reward. In the traditional formulation, the environment is assumed to be a fixed entity that cannot be externally controlled. However, there exist several real-world scenarios in which the environment offers the opportunity to *configure* some of its parameters, with diverse effects on the agent's learning process. In this contribution, we provide an overview of the main aspects of environment configurability. We start by introducing the formalism of the Configurable Markov Decision Processes (Conf-MDPs) and we illustrate the solutions concepts. Then, we revise the algorithms for solving the learning problem in Conf-MDPs. Finally, we present two applications of Conf-MDPs: policy space identification and control frequency adaptation.

1 Introduction

Artificial Intelligence (AI) [32] and *Machine Learning* (ML) [23] are becoming terms widespread in our daily life. The progressive increase of the amount of available *data* and the evolution of the computing systems have enabled ML to become a powerful and effective *decision-making* tool. The traditional taxonomy of ML paradigms includes *supervised*, *unsupervised*, and *reinforcement learning*. Nowadays, the former two have reached an almost mature level of development, having also achieved marvelous results, especially in image classification [16], hand-written text recognition [29], and recommendation systems [1]. Instead, *Reinforcement Learning* (RL) [36] has only recently emerged, beyond the research field, as a valuable approach for several real-world applications. RL can be thought of as the most com-

A. M. Metelli (✉)
Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Via Ponzio 34/5, 20133 Milano, Italy
e-mail: albertomaria.metelli@polimi.it

L. Piroddi (ed.), *Special Topics in Information Technology*,
PoliMI SpringerBriefs, https://doi.org/10.1007/978-3-030-85918-3_9

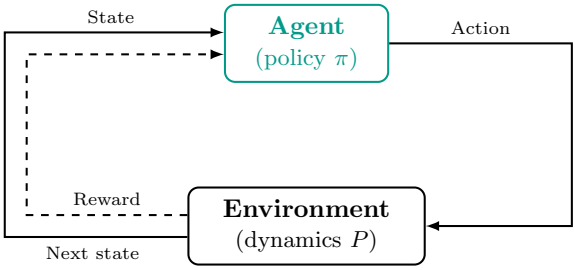

Fig. 1 Graphical representation of the interaction between an agent and an environment in an MDP

plete and general ML paradigm, to which supervised and unsupervised learning can be reduced. Furthermore, it can be considered the paradigm closest to the intuitive idea of the *learning* process, typical of biological entities.

The term *reinforcement* was introduced by Burrhus F. Skinner in behavioral psychology to denote the attitude of organisms to strengthen a behavior if associated to some desirable consequence [35]. Consider, for instance, a baby, an example of a biological agent, learning how to walk. She will interact with the surrounding environment in a *trial and error* fashion, receiving positive and negative feedback. Progressively, she will learn the proper movements in order to stay upright and, finally, effectively walk. In the context of AI, RL refers to the *computational* approach to learning for artificial agents, in which the *interaction* between the agent and the environment plays a central role.

The RL setting is composed of an (artificial) *agent* and an *environment* interacting with one another [36]. The agent senses the state of the environment and performs actions. Every action causes a transition of the environment to a new state, governed by its dynamics (or transition model) P. The agent also receives from the environment a *reward* signal R. The agent-environment interaction proceeds in several (possibly infinite) epochs. The ultimate goal of the agent consists in determining a *policy* π, i.e., a prescription (possibly stochastic) telling which action to play in every state. This form of interaction encodes a *sequential* decision-making problem, typically modeled with the mathematical formalism of the *Markov Decision Processes* (MDPs) [30] (Fig. 1). The distinctive feature of RL, compared to the other ML paradigms (supervised and unsupervised), is that the agent has to plan over a possibly long horizon since the reward can be delayed. As a consequence, it might be beneficial to sacrifice some reward achievable in the immediate future in order to reach a more profitable region in the far future [36]. In the last decades, RL has obtained remarkable success in several fields, including autonomous driving [13], robotic locomotion [11], and video games [24], to mention a few.

Most of the RL literature considers the environment as a fixed entity, out of any control. However, there exist several real-world scenarios in which a partial intervention on the environment dynamics is allowed. Consider, for instance, the task of learning how to drive a Formula 1 vehicle. The vehicle is a portion of the environment and the driver has at her disposal several vehicle settings that can be *configured*, while other parts of the environment are immutable. We call *environment configuration* the activity of altering some environmental parameters.

In this contribution, we provide a summary of the Ph.D. dissertation entitled "Exploiting Environment Configurability in Reinforcement Learning" [18],[1] focused on studying the different aspects of environment configurability. The structure of the present contribution reflects the subdivision in parts of the dissertation. After having introduced, in Sect. 2, the idea of "environment configuration" and having provided some motivational examples, we outline the main results of the dissertation. In Sect. 3, we present the formalization of environment configurability, based on the novel Configurable Markov Decision Process (Conf-MDP) framework. Then, in Sect. 4, we briefly outline the approaches for learning in cooperative Conf-MDPs and focus on the experiment of car configuration based on TORCS. Finally, in Sect. 5, we present two applications of the Conf-MDP framework. We conclude, in Sect. 6, summarizing the results and discussing some future research directions.

2 Configurable Environments

As we mentioned in Sect. 1, the majority of the RL literature disregards the opportunity of configuring the environment, even when possible in the specific case of application. Indeed, traditionally, the modification of the environment dynamics during learning is considered the effect of a *non-stationary* process [2], i.e., a natural evolution of the environment. Instead, the possibility to *strategically* act on the environmental dynamics is studied in a limited number of works only. Some approaches belonging to the planning area [12, 38], some are constrained to specific forms of environment configurability [8, 9, 34], and others based on the *curriculum learning* framework [4, 7]. The goal of the dissertation [18] is to provide a uniform treatment of environment configurability in its diverse aspects. Before moving to the summary of the contributions, we present three motivational examples of environment configuration.

Example 1 (Car Configuration) Consider a Formula 1 driver that has to learn how to drive a Formula 1 car. The driver is the agent and the vehicle is a part of the environment, which is composed of other elements, like the road. The vehicle represents a configurable part of the environment since it is possible to change some of its settings (e.g., the wing orientation, the kind of tiers, and the brake repartition), whereas other parts of the environment are not configurable, such as psychical laws. The goals of the configuration may be different. First, we might want to find the vehicle settings that best fit the agent's needs, and allow she to learn the best performing policy. Second, we might want to train the agent with different vehicle configurations in order to speed up the learning process in a curriculum learning fashion. Notice that the vehicle configuration can be carried out by the agent itself, i.e., the driver can change some settings from its driving console, or by an external *configurator*, like a

[1] Online: http://hdl.handle.net/10589/170616.

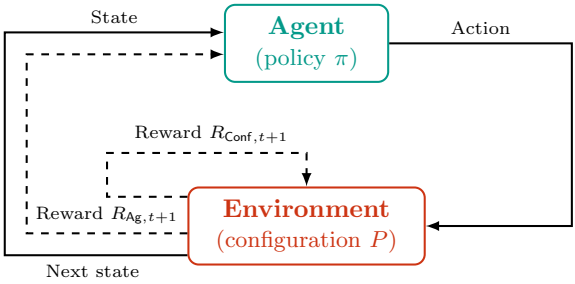

Fig. 2 Graphical representation of the interaction between an agent and an environment in a Conf-MDP

track engineer. Finally, both the agent and the external configurator share the same goals: improve the agent's learning experience, i.e., they act in what we denote as *cooperative* setting.

Example 2 (Teacher-Student) Another example of cooperative behavior is the interaction between a student and a teacher. They both aim at maximizing the knowledge acquired by the student, i.e., the agent. We can think of the teacher as either a physical person or an online teaching platform. In both cases, from the student's viewpoint, the *teaching style* is part of the environment and can be configured and should be tailored to the peculiarities of the student. Therefore, to select a suitable configuration, the teacher has to be aware of the student's capabilities that are to be inferred by interaction with the student herself.

Example 3 (Supermarket) We now consider an example in which agent and configurator no longer interact in a cooperative way. Suppose we are the owner of a supermarket and we have to configure the placement of the goods on the shelves in order to maximize our profit, so inducing customers to buy more. The customers, i.e., the agents, might have a different interest compared to that of the supermarket owner. Maybe they want to find the products they are interested in, in the smallest amount of time or buy certain goods only. This is an example in which the agents, the customers, and the configurator, the supermarket owner, show diverging interests. Thus, we are in what we denote as *non-cooperative* setting [31]. Moreover, we can distinguish between whether the agents are either aware or not of the configurator presence, leading to different levels of strategical behavior.

3 Modeling Environment Configurability

In this section, we outline the main contributions related to the modelization of configurable environments and the proposal of the corresponding solution concepts (Part I of [18]).

3.1 Configurable Markov Decision Processes

In order to represent the configuration opportunities the environment offers, we introduce an extension of the MDP framework: the *Configurable Markov Decision Process* (Conf-MDP) [22]. The main modification, compared to the traditional MDP, is that we no longer have a transition model P, governing the dynamics of the environment since environment configuration has the precise effect of changing it. Instead, we look at P just like the policy π, as an element that has to be determined as an output of the learning process. In the following, we will refer to P as environment *configuration*, instead of transition model, to highlight the features of the considered setting. Furthermore, to account for the possibly different interests of agent and configurator, we consider two reward functions R_{Ag} and R_{Conf} for agent and configurator, respectively (Fig. 2). With these two reward functions, we can define the performance indexes: the *expected returns* $J_{\mathsf{Ag}}^{\pi,P}$ and $J_{\mathsf{Conf}}^{\pi,P}$, i.e., the expected discounted sum of the rewards collected during the interaction with the environment:

$$J_{\mathsf{Ag}}^{\pi,P} = \mathbb{E}_{\mathsf{Ag}}^{\pi,P}\left[\sum_{t=0}^{\infty}\gamma^t R_{\mathsf{Ag},t+1}\right] \quad \text{and} \quad J_{\mathsf{Conf}}^{\pi,P} = \mathbb{E}_{\mathsf{Conf}}^{\pi,P}\left[\sum_{t=0}^{\infty}\gamma^t R_{\mathsf{Conf},t+1}\right],$$

where $\gamma \in [0, 1]$ is the discount factor that provides the relative importance between the reward collected in the present and those that will be collected in the future. The general goal in a Conf-MDP consists in finding an optimal policy π^* together with an optimal configuration P^*. These tasks are carried out by the agent and the configurator respectively. However, the notion of optimality strictly depends on the kind of interaction taking place between the agent and the configurator, as we discuss in the following section.

3.2 Solution Concepts

As we have illustrated in the examples presented in Sect. 2, the interaction between the agent and the configurator can take place in different forms. In particular, we distinguish between *cooperative* and *non-cooperative* Conf-MDPs.

In the cooperative setting, like in Examples 1 and 2, agent and configurator share the same objectives. In other words, they have the same reward function $R := R_{\mathsf{Ag}} = R_{\mathsf{Conf}}$. In such a case, defining a suitable solution concept is straightforward, as we look for an optimal policy π^* and an optimal configuration P^* they jointly maximize the expected return $J^{\pi,P}$, defined in terms of the unique reward function R:

$$(\pi^*, P^*) \in \arg\max_{(\pi,P)\in\Pi\times\mathcal{P}} \left\{J^{\pi,P}\right\}. \tag{1}$$

It is worth noting that the search of the optimal policy π^* and the optimal configuration P^* is constrained in specific policy Π and configuration $\mathcal{P}$ spaces. The choice of

these elements is highly relevant since it defines the actuations and the configuration possibilities. Indeed, while in a large body of the RL literature, it is assumed that the agent can play any policy (restriction are enforced for safety constraints in industrial applications [10]), it is in general unreasonable that the configurator can change arbitrarily the transition model. Indeed, in many real-world scenarios, the transition model groups portions of the environment that are immutable, like physical laws, and some that are mutable and, therefore, configurable (Example 1).

In the non-cooperative setting, like in Example 3, the agent and configurator reward functions R_{Ag} and R_{Conf} are kept distinct, to model situations in which the two entities have diverging interests. In this setting, in order to define suitable solution concepts, we have to resort to game-theoretic *equilibria* [33]. Moreover, the most suitable solution concept depends on the degree of *awareness* of the two entities about the presence of the other. While it is reasonable to assume that the configurator is always aware of the agent presence, the reverse might not be the case. The simplest situation is when the agent is unaware of the configurator presence. In such a case, it will react to any modification of the environment, perceived as a non-stationary evolution, with its optimal policy, that we call *best-response* policy, according to the game-theoretic terminology. Thus, we can map this setting to a *leader-follower* game in which the configurator (leader) wants to find the best configuration according to its reward function R_{Conf}, assuming that the agent (follower) will react with a best-response policy. The solution concept suitable for this setting is the *Stackelberg* equilibrium [37]:

$$P^* \in \arg\max_{P \in \mathcal{P}} \left\{ J_{\mathsf{Conf}}^{\beta_{\mathsf{Ag}}(P), P} \right\},$$

where $\beta_{\mathsf{Ag}}(P) \in \arg\max_{\pi \in \Pi} \left\{ J_{\mathsf{Ag}}^{\pi, P} \right\}$ is a best-response function that maps every configuration P to a an optimal policy $\beta_{\mathsf{Ag}}(P) \in \Pi$ under P.[2] Differently, when the agent is aware of the configurator presence, we are in a more symmetric setting that can be mapped to a *simultaneous* game. In such a case, it is reasonable to consider the *Nash* equilibrium as a solution concept [26], in which we look for a policy-configuration pair such that neither the agent nor the configurator has an interest in unilaterally diverge from the equilibrium:

$$\pi^* \in \arg\max_{\pi \in \Pi} \left\{ J_{\mathsf{Ag}}^{\pi, P^*} \right\} \quad \text{and} \quad P^* \in \arg\max_{P \in \mathcal{P}} \left\{ J_{\mathsf{Conf}}^{\pi^*, P} \right\}.$$

[2] The need for defining such a function derives from the fact that multiple best responses might generate different expected returns when evaluated with the configurator reward. In the game theory literature, common approaches consist in breaking ties in favor of the leader (strong Stackelberg equilibrium) or in favor of the follower (weak Stackelberg equilibrium) [5].

4 Learning in the Cooperative Configurable Markov Decision Processes

In this section, we provide a brief outline of the learning algorithms for cooperative Conf-MDPs, with particular reference to a car configuration example based on the TORCS simulator (Part II of [18]).

In the context of cooperative Conf-MDPs, the learning problem consists in finding an optimal policy π^* together with an optimal environment configuration P^* so that they jointly maximize the expected return, as in Equation (1). In [18], two algorithms are presented to tackle this problem. In the dissertation, we first focus on Conf-MDPs with finite state-action spaces and we propose a safe algorithm, *Safe Policy Model Iteration* (SPMI) [22], endowed with strong theoretical guarantees on the performance improvement. However, despite being the first attempt to solve the learning problem in Conf-MDPs, SPMI displays some limitations. First, it can be employed in finite Conf-MDPs only. Second, it requires the full knowledge of the environment dynamics, i.e., the configurator has to know not only the configurable parameters but also their effect on the transition dynamics. In order to overcome these limitations, we introduce a new algorithm, *Relative Entropy Model Policy Search* (REMPS) [19]. REMPS applies to continuous state-actions and no longer requires the knowledge of the transition model. The only assumption is that the configurator must know which are the configurable environmental parameters, while their effect on the transition model is learned from samples.

We tested REMPS on a simulated car configuration task based on the TORCS simulator [15]. The agent has access to a low-dimensional state representation based on the cars sensors (e.g., speed, focus, wheel speeds) and it can act on low-level controllers (acceleration, braking, and steering). The agent's goal consists of driving the car minimizing the lap time. The configurator, instead, is allowed to modify three configurable environmental parameters: rear wing orientation, front wing orientation, and front-rear brake repartition. In Fig. 3, we show the expected return and the average lap duration comparing REMPS, REPS (Relative Entropy Policy Search) [28], in

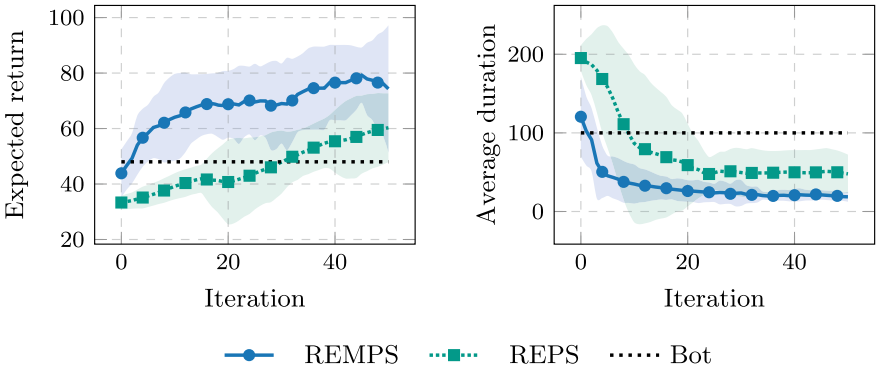

Fig. 3 Expected return and episode duration as a function of the number of iterations in the TORCS experiment (10 runs, 80% c.i.)

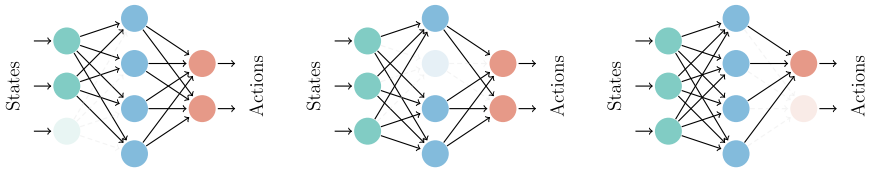

Fig. 4 An example of policy space modeled as a neural network with limitations in perceptions (left), mapping (center), and actuation (right)

which the car is not configured and the initial configuration is kept fixed, and a bot baseline. We can see that REMPS is able to outperform both REPS and the bot in terms of final policy performance and in terms of learning speed, showing that environment configuration can be beneficial also for speeding up the learning process.

5 Applications of Configurable Markov Decision Processes

In this section, we outline two cases of applications of Conf-MDPs in which the environment configuration opportunities can play a relevant role (Part III of [18]): *policy space identification* (Sect. 5.1) and *control frequency adaptation* (Sect. 5.2).

5.1 Policy Space Identification

In Example 2, about the teacher-student interaction, we have illustrated that in order to wisely choose the environment configuration, i.e., the teaching style, the configurator (teacher) has to be aware of the agent's (student) capabilities. More formally, the agent's capabilities are related to its perceptions, actuations, and ability to map states into actions. These three elements define the space of policies the agent can play, i.e., the *agent's policy space*. In this part of the dissertation [20], we study how to identify the agent's policy space by observing its behavior. Besides configurable MDPs, knowing the policy space of an expert agent might be of interest also in the imitation learning field [27] in order to prevent possible overfitting/underfitting phenomena.

We assume that the agent's policy is parametric, i.e., the policy $\pi_{\boldsymbol{\theta}}$ depends on a parameter vector $\boldsymbol{\theta} \in \Theta$. Among the $\boldsymbol{\theta}$ parameters, the agent controls just a subset of them, where "controls" means that the agent can change their value, while the others are conventionally set to a fixed value. This represents a way of restricting the agent's policy space. For instance, suppose the agent is equipped with a neural network policy and does not perceive a state variable. This can be represented by

setting to zero the weights related to that state variable. A similar construction can be performed to represent limitations in the agent's actuation and ability to map states to actions (Fig. 4).

In the dissertation [18], we propose an approach based on *generalized likelihood ratio* tests [3] to identify the set of parameters that the agent controls. Furthermore, we provide guarantees in terms of the probability of misidentification and numerical simulations on benchmark domains.

5.2 Control Frequency Adaptation

The typical RL setting deals with *discrete-time* problems that are obtained from the time discretization of a *continuous-time* problem [17]. Time discretization requires selecting a *control frequency* that is a design choice and represents, in all regards, configurable environmental parameter. This problem arises in several real-world domains, including robot control [14] and trading [25]. On one hand, we might be tempted to prefer high frequencies because they provide better control opportunities, leading to possibly more performing policies. However, in such a case, actions will last for a small time interval, thus, their effect on the environment will not be very clear. On the contrary, low frequencies sacrifice some control opportunities, but each action will last longer, making its effect more visible, with a possible benefit on the sample complexity. The question we address in this part of the dissertation [18] is whether we can exploit this trade-off to define a notion of optimal control frequency.

We propose to model the adaptation of the control frequency by means of *action persistence* [21], which consists in the repetition of each action for multiple k consecutive time steps. We focus on the trade-off in the choice of the persistence k. We first show that by increasing k, we give up control opportunities and, consequently, the optimal policy performance decreases. To visualize the beneficial effect of a large k on the sample complexity, we propose a novel algorithm, *Persistent Fitted Q-Iterations* (PFQI) [21]. PFQI is a batch RL algorithm and extends the classical Fitted Q-Iterations (FQI) [6] to account for action persistence. PFQI enjoys a sample complexity that decreases with the persistence k. Therefore, we observe that the optimal value of k depends on the number of samples (batch size) available for learning. Intuitively, when the batch size is large, we can afford a small value of k, whereas with few samples, we benefit from the regularization effect of using a high persistence. Furthermore, we propose a heuristic approach to suggest an approximately optimal value of k.

We tested PFQI on a simple forex trading simulator. In Fig. 5, we show the performance of the policy learned with PFQI for different batch sizes and different persistence values. First of all, we notice a generally improving trend as the batch size increases. If we look at small batch sizes, we observe that the best performance is obtained with high values of persistence ($k = 4$ or $k = 8$), whereas as the batch size increases, the best performance is attained by small values of persistence, namely $k = 1$ for batch size 400.

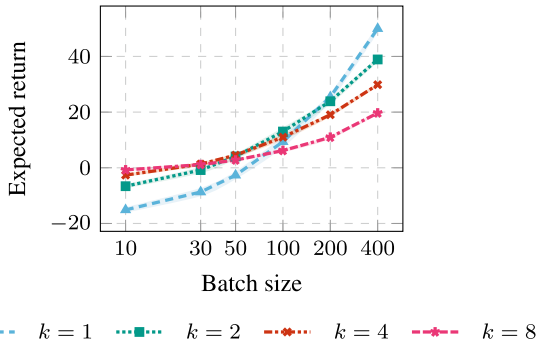

Fig. 5 Expected return as a function of the batch size in the forex experiment (10 runs, 95% c.i.)

6 Conclusions

In this contribution, we summarized the results of the Ph.D. dissertation [18]. We started by introducing the novel framework of the Conf-MDPs to model several real-world situations in which agent interacts with a configurator, entitled to modifying some environmental parameters. Then, we provided learning algorithms for the cooperative setting, showing that environment configuration can be beneficial for the performance of the agent's policy. We have seen that knowing the agent's policy space is important for suitably choosing an environment configuration and we presented an approach for identifying it from samples. Finally, we have studied the effects on the learning performance of a specific configurable environmental parameter, namely the control frequency.

We hope, with this dissertation, to have shed light on a novel topic of interest in the RL community. Numerous directions could be further explored. From the modelization standpoint, it would be worth considering the possibility of having multiple agents interacting with multiple configutators acting in the same environment. Concerning the learning problem, online approaches for searching for the best configuration in the cooperative setting should be investigated, while the learning problem for the non-cooperative setting has to be further deepened. Finally, concerning action persistence, our approach is limited to a fixed persistence. It would be interesting studying the possibility of having a control frequency that dynamically changes during the learning process.

Acknowledgements The author wishes to thank Prof. Marcello Restelli for the continuous support and guidance.

References

1. J. Bobadilla, F. Ortega, A. Hernando, A. Gutiérrez, Recommender systems survey. Knowl. Based Syst. **46**, 109–132 (2013)
2. B.L. Bowerman, *Nonstationary Markov Decision Processes and Related Topics in Nonstationary Markov Chains* (1974)
3. G. Casella, R.L. Berger, *Statistical Inference*, vol. 2 (Duxbury Pacific Grove, CA, 2002)
4. K.A. Ciosek, S. Whiteson, OFFER: off-environment reinforcement learning, in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, ed. by S.P. Singh, S. Markovitch (AAAI Press, 2017), pp. 1819–1825
5. V. Conitzer, T. Sandholm, Computing the optimal strategy to commit to, in *Proceedings 7th ACM Conference on Electronic Commerce (EC-2006), Ann Arbor, Michigan, USA, June 11-15, 2006*, ed. by J. Feigenbaum, J.C.-I. Chuang, D.M. Pennock (ACM, 2006), pp. 82–90
6. D. Ernst, P. Geurts, L. Wehenkel, Tree-based batch mode reinforcement learning. J. Mach. Learn. Res. **6**, 503–556 (2005)
7. C. Florensa, D. Held, M. Wulfmeier, M. Zhang, P. Abbeel, Reverse curriculum generation for reinforcement learning, in *1st Annual Conference on Robot Learning, CoRL 2017, Mountain View, California, USA, November 13-15, 2017, Proceedings*, vol. 78 of *Proceedings of Machine Learning Research* (PMLR, 2017), pp. 482–495
8. V. Gallego, R. Naveiro, D.R. Insua, Reinforcement learning under threats, in *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019* (AAAI Press, 2019), pp. 9939–9940
9. V. Gallego, R. Naveiro, D.R. Insua, D. Gómez-Ullate, Opponent aware reinforcement learning, in *CoRR*, abs/1908.08773 (2019)
10. J. García, F. Fernández, A comprehensive survey on safe reinforcement learning. J. Mach. Learn. Res. **16**, 1437–1480 (2015)
11. T. Haarnoja, S. Ha, A. Zhou, J. Tan, G. Tucker, S. Levine, Learning to walk via deep reinforcement learning, in *Robotics: Science and Systems XV, University of Freiburg, Freiburg im Breisgau, Germany, June 22-26, 2019*, ed. by A. Bicchi, H. Kress-Gazit, S. Hutchinson (2019)
12. S. Keren, L.E. Pineda, A. Gal, E. Karpas, S. Zilberstein, Equi-reward utility maximizing design in stochastic environments, in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, ed. by C. Sierra, ijcai.org (2017), pp. 4353–4360
13. B. Ravi Kiran, I. Sobh, V. Talpaert, P. Mannion, A.A. Al Sallab, S.K. Yogamani, P. Pérez, Deep reinforcement learning for autonomous driving: a survey, in *CoRR*. abs/2002.00444 (2020)
14. J. Kober, J. Andrew Bagnell, J. Peters, Reinforcement learning in robotics: a survey. I. J. Robotics Res. **32**(11), 1238–1274 (2013)
15. D. Loiacono, A. Prete, P.L. Lanzi, L. Cardamone, Learning to overtake in TORCS using simple reinforcement learning, in *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2010, Barcelona, Spain, 18-23 July 2010* (IEEE, 2010), pp. 1–8
16. D. Lu, Q. Weng, A survey of image classification methods and techniques for improving classification performance. Int. J. Remote Sensing **28**(5), 823–870 (2007)
17. D.G. Luenberger, Introduction to dynamic systems; theory, models, and applications. Technical report (1979)
18. A.M. Metelli, *Exploiting Environment Configurability in Reinforcement Learning*. PhD thesis, Politecnico di Milano, March 2021
19. A.M. Metelli, E. Ghelfi, M. Restelli, Reinforcement learning in configurable continuous environments, in *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, ed. by K. Chaudhuri, R. Salakhutdinov, vol. 97 of *Proceedings of Machine Learning Research* (PMLR, 2019), pp. 4546–4555
20. A.M. Metelli, G. Manneschi, M. Restelli, Policy space identification in configurable environments, in *CoRR*, abs/1909.03984 (2019)

21. A.M. Metelli, F. Mazzolini, L. Bisi, L. Sabbioni, M. Restelli, Control frequency adaptation via action persistence in batch reinforcement learning, in *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, vol. 119 of *Proceedings of Machine Learning Research* (PMLR, 2020), pp. 6862–6873
22. A.M. Metelli, M. Mutti, M. Restelli, Configurable markov decision processes, in *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, ed. by J.G. Dy A. Krause, vol. 80 of *Proceedings of Machine Learning Research* (PMLR, 2018), 3488–3497
23. T.M. Mitchell. *Machine Learning*, International Edition. McGraw-Hill Series in Computer Science (McGraw-Hill, 1997)
24. V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, M.A. Riedmiller, Playing atari with deep reinforcement learning, in *CoRR*, abs/1312.5602 (2013)
25. J.E. Moody, M. Saffell, Learning to trade via direct reinforcement. IEEE Trans. Neural Netw. **12**(4), 875–889 (2001)
26. J. Nash, Non-cooperative games. Ann. Math. **54**(2), 286–295 (1951)
27. T. Osa, J. Pajarinen, G. Neumann, J. Andrew Bagnell, P. Abbeel, J. Peters, An algorithmic perspective on imitation learning. Foundations Trends Robot. **7**(1–2), 1–179 (2018)
28. J. Peters, K. Mülling, Y. Altun, Relative entropy policy search, in *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010*, ed. by M. Fox, D. Poole (AAAI Press, 2010)
29. J. Puigcerver, Are multidimensional recurrent layers really necessary for handwritten text recognition?, in *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, vol. 1, pp. 67–72 (IEEE, 2017)
30. M.L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons (2014)
31. G. Ramponi, A.M. Metelli, A. Concetti, M. Restelli, Online learning in non-cooperative configurable Markov decision process, in *AAAI-21 Workshop on Reinforcement Learning in Games* (2021)
32. S.J. Russell, P. Norvig, *Artificial Intelligence– - Modern Approach* (Third International Edition, Pearson Education, 2010)
33. L.S Shapley, Stochastic games. Proc. Natl. Acad. Sci. **39**(10), 1095–1100 (1953)
34. R. Silva, F.S. Melo, M. Veloso, What if the world were different? gradient-based exploration for new optimal policies, in *GCAI-2018, 4th Global Conference on Artificial Intelligence, Luxembourg, September 18-21, 2018*, ed. by D.D. Lee, A. Steen, T. Walsh, vol. 55 of *EPiC Series in Computing* (EasyChair, 2018), pp. 229–242
35. B.F. Skinner, *The Behavior of Organisms: An Experimental Analysis* (1938)
36. R.S. Sutton, A.G. Barto, *Reinforcement Learning: An Introduction*. MIT press (2018)
37. H. Von Stackelberg, *Marktform und gleichgewicht*. J. Springer (1934)
38. H. Zhang, Y. Chen, D.C. Parkes, A general approach to environment design with one agent, in *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009*, ed. by C. Boutilier (2009), pp. 2002–2014