2023 62nd IEEE Conference on Decision and Control (CDC)
December 13-15, 2023. Marina Bay Sands, Singapore

Computationally Efficient Reinforcement Learning: Targeted Exploration leveraging Simple Rules

Loris Di Natale, Bratislav Svetozarevic, Philipp Heer, and Colin N. Jones

***Abstract*— Model-free Reinforcement Learning (RL) generally suffers from poor sample complexity, mostly due to the need to exhaustively explore the state-action space to find well-performing policies. On the other hand, we postulate that expert knowledge of the system often allows us to design simple rules we expect good policies to follow at all times. In this work, we hence propose a simple yet effective modification of continuous actor-critic frameworks to incorporate such rules and avoid regions of the state-action space that are known to be suboptimal, thereby significantly accelerating the convergence of RL agents. Concretely, we saturate the actions chosen by the agent if they do not comply with our intuition and, critically, modify the gradient update step of the policy to ensure the learning process is not affected by the saturation step. On a room temperature control case study, it allows agents to converge to well-performing policies up to $6-7\times$ faster than classical agents without computational overhead and while retaining good final performance.**

I. Introduction

Despite its success in many applications [1], [2], model-free Reinforcement Learning (RL), and in particular deep RL (DRL), usually suffer from *data inefficiency*, i.e., they require a significant number of interactions with the environment to converge [3], [4]. This stems from the necessity to explore the state-action space to find optimal policies and leads to significant computational costs. It also limits the deployment of DRL methods on physical systems without pretraining in simulation: learning a building temperature control policy from scratch can for example take years of data [5], [6].

To speed up the training of DRL agents, researchers have for example investigated how to leverage expert demonstrations [7], [8], but this requires access to an expert policy which is not always available in practice. Instead, we postulate that prior knowledge of physical systems often allows us to design simple rules that agents should follow *a priori*, such as *"Do not heat the room if it is already* $26\,^{\circ}\mathrm{C}$*"*; we indeed know this action will *always* be suboptimal in that state, there is no need for agents to explore its consequences.

In this paper, we hence propose modifications of actor-critic algorithms to encode simple rules in RL agents, introducing *state-dependent constraints* on the agents' actions to restrict exploration to interesting regions of the state-action space. In other words, the key idea is to avoid visiting state-action pairs that are known to be suboptimal by the expert to accelerate the convergence towards meaningful solutions and thus increase the efficiency of (D)RL. Note that while state-dependent bounds were concurrently introduced in [9], they are enforced *a posteriori* in the environment instead of directly modifying the agent's behavior and do not necessarily improve convergence. In another line of work, prior knowledge successfully accelerated learning in [10], but relying on fuzzy rather than direct rule integration.

This research was supported by the Swiss National Science Foundation under NCCR Automation, grant agreement 51NF40_180545, and in part by the Swiss Data Science Center, grant agreement C20-13.

L. Di Natale, B. Svetozarevic, and P. Heer are with the Urban Energy Systems Lab, Empa, Dübendorf, Switzerland. {loris.dinatale, bratislav.svetozarevic, philipp.heer}@empa.ch.

L. Di Natale and C. N. Jones are with the Laboratoire d'Automatique, EPFL, Lausanne, Switzerland. colin.jones@epfl.ch.

A. Constraining RL agents

To bound the decisions taken by RL agents, one typically defines some *constrained set* of actions and either project the actions of the agents on this set at each time step or switch to a fallback controller when needed [11], [12], [13], [14]. The main challenge with these operations is that they are usually not differentiable and hence cannot be learned by RL agents, with the notable exceptions of [15], [16], [17], who leveraged differentiable optimization layers [15], modified the policy updates to account for projections [16], or derived a closed-form solution of the projection step [17]. However, these methods either entail additional computational burden [15], [16] or rely on a learned linearization of the constraints [17]. Alternatively, one could also apply tools from the safe RL literature [18], [19], [20], typically relying on constrained policy optimization [21], [22]. However, this would again introduce both engineering and computational overhead.

The complexity of the methods discussed above often stems from the fact that they are designed to impose *state constraints* on DRL agents, which is a more challenging problem in general since it leads to complex action bounds for the agent at each step. Here, however, we argue that prior knowledge can straightforwardly be used to accelerate the training of DRL agents through simple state-dependent *box constraints on their actions*, which allows us to leverage less computationally expensive tools.

To alleviate the issue of non-differentiability without increasing either the engineering or the computational burden, Reward Shaping (RS) heuristics might be used in various forms to penalize agents when constraints are violated, let them know when a fallback controller was used or they were saturated, or introduce prior knowledge about the task to solve [14], [23], [24]. While such methods might accelerate the learning process to some extent, they are however indirect, i.e., they only influence the learned policies through the reward function that the agent will learn to optimize over time. Moreover, shaping the reward function simultaneously impacts the learning process of both the actor *and* the critic.

979-8-3503-0123-6/23/$31.00 ©2023 IEEE

B. Contribution

In this work, we propose to clip DRL agents' actions according to simple expert-designed state-dependent bounds and then modify the gradient update step of the actor to let agents learn from their mistakes and accelerate their convergence to expected actions. To explain the effectiveness of the gradient modification, we provide an intuitive analytical analysis of its impact on the learning of DRL agents. Importantly, contrary to RS, our modifications only affect the actor, allowing the critic to learn the true Q-values.

Remarkably, our method bypasses the need for complex projection steps and does not require access to a fallback controller or an expert policy. Moreover and critically, the proposed modifications do not impact the computational complexity of the algorithm, are straightforward to design and implement, and can be coupled with any actor-critic algorithms. Note that, in contrast with [10], where the expert knowledge is potentially overridden by the policy, our method enforces the wanted behaviors on agents *at all times*.

The effectiveness of the proposed Efficient Agents (EAs) is demonstrated in simulation on a room temperature control case study, where they converge up to 6 – 7 times faster than classical ones and 2 – 3 times faster than RS-based agents while retaining good final performance. This hints at how the proposed modifications can provide a simple yet effective and computationally inexpensive mean to leverage expert knowledge to accelerate DRL algorithms.

II. PRELIMINARIES

A. Reinforcement Learning

At each time step t, given an observation s_t of the state of the environment, an RL agent chooses an action a_t. The environment then transitions to s_{t+1} according to the transition probabilities $P(s_t, a_t)$ and sends the new state and the reward signal $r(s_t, a_t)$ to the agent. The objective of any deterministic RL algorithm[1] is to find a policy $\pi(s_t)$ that maximizes the expected discounted cumulative returns:

$$J(\pi) = \mathbb{E}_{a_t \sim \pi(s_t), s_{t+1} \sim P(s_t, a_t)} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right], \quad (1)$$

where γ is the discount factor trading off near- and long-term rewards, and the initial state $s_0 \sim \rho$ is sampled from the corresponding initial distribution. With a slight abuse of notation on the expectation for clarity, we can define the Q-function of any state-action pair (s, a):

$$Q^\pi(s, a) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) | s_0 = s, a_0 = a \right], \quad (2)$$

which captures the expected returns when action a is chosen in state s and the policy π is followed thereafter.

In our simulations, we let agents explore the environment with the ϵ-greedy exploration strategy, which means we apply the following action to the environment:

$$a(s) = \text{clip}(\pi(s) + \epsilon, a^{low}, a^{up}), \qquad \epsilon \sim \mathcal{N}(0, \sigma), \quad (3)$$

[1]While the presented analyses deal with deterministic actor-critic agents for clarity, the results can easily be extended to the stochastic case.

where the noisy actions are clipped elementwise between a^{low} and a^{up}, the predefined action bounds from the environment, and ϵ is the Gaussian exploration noise with a standard deviation of σ. All the transition tuples (s, a, r, s') observed by the agent are stored in a replay buffer.

B. Actor-critic algorithms

In practice, policies and Q-functions are often parametrized with Neural Networks (NNs) as π_θ and Q_ϕ, respectively, leading to DRL, and numerous algorithms have been developed to maximize (1) [25]. In this work, we are interested in deterministic actor-critic methods stemming from [26], where both the actor π_θ (also referred to as the policy) and the critic Q_ϕ are optimized in parallel leveraging gradient descent. While different flavors exist, most algorithms compute the gradient of the critic using the Temporal Difference (TD) loss [26]:

$$\hat{\nabla}_\phi Q_\phi = \nabla_\phi \left[\frac{1}{|B|} \sum_{b=(s,a,r,s') \in B} \left(Q_\phi(s, a) - y(b)\right)^2 \right], \quad (4)$$

with $y(b) = (r + \gamma \max_{a'} Q_\phi(s', a'))$ and where a batch B of past transitions is sampled from the replay buffer and used to estimate expectations. Leveraging the policy gradient theorem [27], one can similarly use the critic to estimate the actor gradient as:

$$\hat{\nabla}_\theta \pi_\theta = -\nabla_\theta \left[\frac{1}{|B|} \sum_{s \in B} Q_\phi(s, \pi_\theta(s)) \right]. \quad (5)$$

Note that these gradients are easily computed using automatic differentiation when the actor and the critic are parametrized with NNs.

In this paper, we rely on the Twin Delayed Deep Deterministic (TD3) policy gradient algorithm, which introduces a few modifications to limit the well-known overestimation bias of Q-functions plaguing vanilla actor-critic algorithms [28]. Remarkably, however, these adjustments do not impact the actor gradient in (5), allowing us to seamlessly integrate the proposed modifications detailed in Section III.

III. METHODS

A. State-dependent action saturation

In many cases, prior knowledge allows us to design state-dependent upper and lower bounds $a^{max}(s)$ and $a^{min}(s)$, respectively, on the actions we expect well-performing control policies to take in a given state s, with:

$$a^{low} \leq a^{min}(s) \leq a^{max}(s) \leq a^{up}. \quad (6)$$

To limit the exploration of known suboptimal state-action pairs, we can then modify (3) accordingly to:

$$a(s) = \text{clip}(\pi_\theta(s) + \epsilon, a^{min}(s), a^{max}(s)). \quad (7)$$

Note that these bounds stemming from prior knowledge are also enforced at test time when $\epsilon = 0$ to ensure an agent would *never* turn left if whenever there is a wall in that direction, for example, neither during the training nor the deployment phase.

B. Actor gradient modification

The major problem with the clipping operation in (7) is its non-differentiability. Worse yet, its subdifferentials go to zero whenever agents are saturated (see (9) in Section III-C), making any backward flow of information on the overriding process impossible. As a countermeasure, to let agents learn from their mistakes, we also modify the actor gradient (5) to:

$$\hat{\nabla}_\theta^{EA}\pi_\theta = -\nabla_\theta\Bigg(\frac{1}{|B|}\sum_{(s,a)\in B}\Big[Q_\phi(s,\pi_\theta(s)) - \frac{\lambda}{2}\left(\pi_\theta(s)-a(s)\right)^2\Big]\Bigg), \tag{8}$$

where λ is a hyperparameter. The last term in (8) penalizes actions chosen by the policy $\pi_\theta(s)$ if they deviate from the constrained action $a(s)$ that was applied to the environment, thus steering the agent's decisions towards expected actions.[2]

Alternatively, we note here that one could instead modify the reward function to include this penalty (RS) and then maximize (1). Remarkably, however, the latter also impacts the learning process of the critic in (4) when applied to actor-critic frameworks, contrary to our method. We will show empirical benefits of the proposed modification (8) over RS-based penalties in terms of convergence speed in Section V.

C. Implications of the modified gradients

Let $C(s) = \left\{a\in\mathbb{R} : a^{min}(s)\le a\le a^{max}(s)\right\}$ for any given state s.[3] Grouping all the parameters θ in a vector and recalling the definition of the action $a(s)$ applied to the environment in state s from (7), we can define its subgradient $\nabla_\theta a(s)$ as:

$$a(s) = \begin{cases} a^{min}(s), & \text{if } \pi(s) < a^{min}(s),\\ \pi_\theta(s)+\epsilon, & \text{if } \pi_\theta(s)\in C(s),\\ a^{max}(s), & \text{if } \pi(s) > a^{max}(s).\end{cases}$$

$$\implies \nabla_\theta a(s) = \begin{cases}\nabla_\theta\pi_\theta(s), & \text{if } \pi_\theta(s)\in C(s),\\ 0, & \text{else,}\end{cases} \tag{9}$$

where $\nabla_\theta\pi_\theta(s)$ is the actor gradient. We can then rewrite the gradient of EAs (8) as:

$$\begin{aligned}\hat{\nabla}_\theta^{EA}\pi_\theta &= -\frac{1}{|B|}\sum_{(s,a)\in B}\Big[\nabla_\theta Q_\phi(s,\pi_\theta(s)) - \nabla_\theta\left(\frac{\lambda}{2}\left(\pi_\theta(s)-a(s)\right)^2\right)\Big]\\ &= -\frac{1}{|B|}\sum_{(s,a)\in B}\Big[\nabla_\theta Q_\phi(s,\pi_\theta(s)) - \Big(\lambda\left(\nabla_\theta\pi_\theta(s)-\nabla_\theta a(s)\right)^\top\left(e_\theta(s)\right)\Big)\Big],\end{aligned}$$

where we introduce the error term $e_\theta(s)=\pi_\theta(s)-a(s)$. We hence get the following modified actor gradient, where we omit $(s,a)\in B$ for clarity:

$$\hat{\nabla}_\theta^{EA}\pi_\theta = \begin{cases} -\frac{1}{|B|}\sum_B\Big[\nabla_\theta Q_\phi(s,\pi_\theta(s))\Big], & \text{if } \pi_\theta(s)\in C(s),\\ -\frac{1}{|B|}\sum_B\Big[\nabla_\theta Q_\phi(s,\pi_\theta(s)) -\lambda\nabla_\theta\pi_\theta(s)^\top e_\theta(s)\Big], & \text{else.}\end{cases}$$

Remarkably, the additional penalty term in (8) hence allows us to solve the issue of the subdifferentials of the clipping operator being zero when actions are saturated, modifying the gradients *only when the constraints are not met*. Indeed, as long as the action chosen by the agent respects the constraints provided by the expert, the classical gradient (5) is used. On the other hand, as soon as the constraints are not met, the gradient is modified in the direction $e_\theta(s)$ to accelerate the convergence of $\pi_\theta(s)$ to $C(s)$ despite the subdifferential of the clipped action being zero, confirming the graphical intuition from [15, Fig. 2]. This allows EAs to learn from their mistakes and — we hypothesize — helps them rapidly converge to meaningful policies.

IV. Room temperature control case study

To assess the effectiveness of the proposed method, we apply it to a temperature control case study, where the objective is to minimize the energy consumption of a room while maintaining the comfort of the occupants, represented by predefined temperature bounds that should not be exceeded.

A. Reinforcement Learning framework

The continuous action space of the agents corresponds to how much heating or cooling power, should be applied at each time step, normalized between $a^{low}=-1$ and $a^{up}=1$. During the heating season, a^{low} corresponds to the heating being turned off and a^{up} to full heating, and the contrary in the cooling case. Physically Consistent Neural Networks (PCNNs) [29] are used to simulate one bedroom in the NEST building [30] and s_t gathers time, weather, temperature, and comfort bound information (see [6] for details). The reward function is defined as the negative weighted sum of energy consumption E_t and comfort violations, i.e. how far from the designed bounds the temperature inside the room is:

$$r(s_t,a_t) = -\max\left\{L_t-T_t, T_t-U_t, 0\right\} - \alpha E_t, \tag{10}$$

$$E_t = \begin{cases}\frac{a_t+1}{2}\,E_{heat}^{max}, & \text{in the heating season,}\\ \frac{1-a_t}{2}\,E_{cool}^{max}, & \text{in the cooling season.}\end{cases}$$

where L_t and U_t represent the lower and upper comfort bounds on the temperature T_t at time t, respectively, α is a weighting factor, and E_{heat}^{max} and E_{cool}^{max} stand for the maximal heating and cooling power, respectively.

B. Design of the saturation rules

In the context of room temperature control, we intuitively know that an optimal policy should gradually stop heating when the temperature reaches the upper comfort bound and gradually start heating as soon as the lower bound is not met

[2]In another line of work, this penalty was also used in [15] to improve the robustness of differentiable layer-based RL for state-constrained problems.

[3]Without loss of generality, we assume that $a\in\mathbb{R}$ in this section for clarity. This assumption can easily be lifted for multi-dimensional problems.

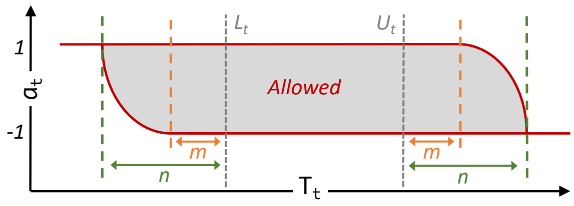

Fig. 1: Representation of the action bounds used in this work.

(and vice versa for cooling). To encode these simple rules, we design state-dependent action bounds as follows:

$$a^{min}(s_t) = \text{clip}\left(\frac{(L_t - m) - T_t}{n - m},\ 0,\ 1\right)^2 * 2 - 1 \qquad (11)$$

$$a^{max}(s_t) = 1 - 2 * \text{clip}\left(\frac{T_t - (U_t + m)}{n - m},\ 0,\ 1\right)^2, \qquad (12)$$

with $n \geq m \geq 0$ representing design parameters to leave more or less freedom to the agents. In words, we start constraining the action of the agents as soon as the temperature deviates from the bounds for more than m degrees and then quadratically increase the constraint until n degrees have been reached, where the agent is forced to use the maximum or minimum power, as pictured in Fig. 1.

V. Results

To investigate the influence of m and n, which measure how much prior knowledge is transmitted to DRL agents, we train different EAs (EA m / n). For comparison purposes, we also train agents with the classical actor gradient (5), introducing the additional squared penalty $\frac{\lambda}{2}\left(\pi_\theta(s) - a(s)\right)^2$ in the reward function instead as another computationally inexpensive means to incorporate prior knowledge in agents (RS m / n). Finally, we also analyze two classical DRL agents with different random seeds (Classical 1 and 2).[4]

A. Final performance

All the agents were trained on up to three-day-long episodes randomly sampled from three years of data. They were evaluated after each 96 steps of 15 min, i.e. one day's worth of data, hereafter also referred to as an *epoch*, on a testing set of 50 unseen sequences of three days. They all use the same hyperparameters as in [6]. We manually set $\lambda = 100$ for EAs to ensure the constraints are enforced as fast as possible and $\lambda = 10$ for RSs since larger penalties led to instability. While we empirically observed more robust performance of EAs with respect to λ compared to RSs, a complete sensitivity analysis is left for future work.

The best reward obtained by all the trained agents over the first 500 epochs can be found in Table I, and the corresponding trade-off between energy consumption and comfort violations is plotted in Fig. 2. These results illustrate how tighter parameters m and n, i.e., higher levels of prior knowledge, allow EAs and RSs to converge to better solutions in this limited training regime. In particular, it allows EAs to reduce the amount of comfort violations without

[4]The code and data are available on https://gitlab.nccr-automation.ch/loris.dinatale/efficient-drl.

TABLE I: Best reward obtained by each agent on the test set over the first 500 epochs.

Agent	Rew.	Agent	Rew.	Agent	Rew.
Classical 1	-2.64	Classical 2	-2.75		
RS 0 / 1	-2.58	EA 0 / 1	-2.85	EA 0.5 / 1	-2.83
RS 0 / 0.5	-2.44	EA 0 / 0.5	-2.58	EA 0.25 / 0.5	-2.74
RS 0 / 0.25	**-2.37**	EA 0 / 0.25	-2.51	EA 0.2 / 0.25	-2.62
RS 0 / 0.1	-3.69	EA 0 / 0.1	**-2.46**	EA 0.075 / 0.1	**-2.46**

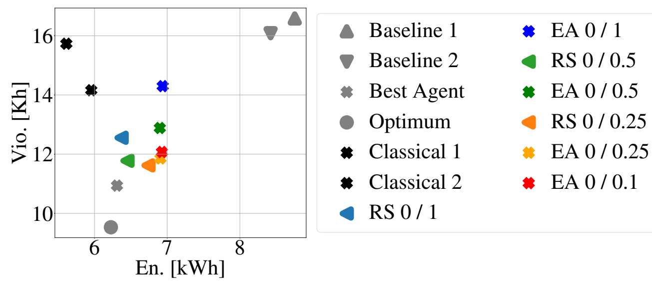

Fig. 2: Average energy consumption (En.) and comfort violations (Vio.) over the test set corresponding to each agent in Table I. The performance of two industrial baselines, of an agent trained for 125,000 epochs (Best Agent), and the optimal performance achievable (Optimum), all computed as in [6], are reported in gray.

significantly increasing the energy consumption. Classical DRL agents on the other hand usually use less energy at the cost of additional comfort violations in this early phase of learning before converging to near-optimal solutions after longer training times [6].

B. Visualization of the impact of prior knowledge

To intuitively understand the effect of action saturation, we visualize its impact on some EAs in Fig. 3, where the behavior of all agents is plotted *before* training on the left, and *after* on the right, for the same three days during the heating season. Focusing on the left plot, we see the untrained classical DRL agent in black letting the temperature diverge to an uncomfortably high range (out of the bounds of the plot) as it starts exploring the state space using roughly constant heating power. On the other hand, all the EAs are forced to stop heating once they are n degrees out of bounds. Consequently, even before training, such agents will not overheat the room and keep it at acceptable temperatures for the occupants, corresponding to what we expect from good controllers. However, note that EAs can present control input oscillations due to the impact of external disturbances, mainly the solar gains around noon, triggering the saturation mechanism on and off.

On the right plot, after training, one can observe that all EAs generally take comparable decisions — still being sometimes saturated, which ensures compliance with prior expert knowledge — leading to similar temperature patterns. On the other hand, the classical agent presents a slightly different behavior, with smoother decision patterns. Interestingly, this agent is the only one heating in the early afternoon, while the EAs wait until the end of the afternoon to heat the room with high power and meet the comfort bound tightening at 8pm. This allows the classical agent to use less energy than EAs over these three days but can incur additional comfort violations (Table II), as expected from Fig. 2.

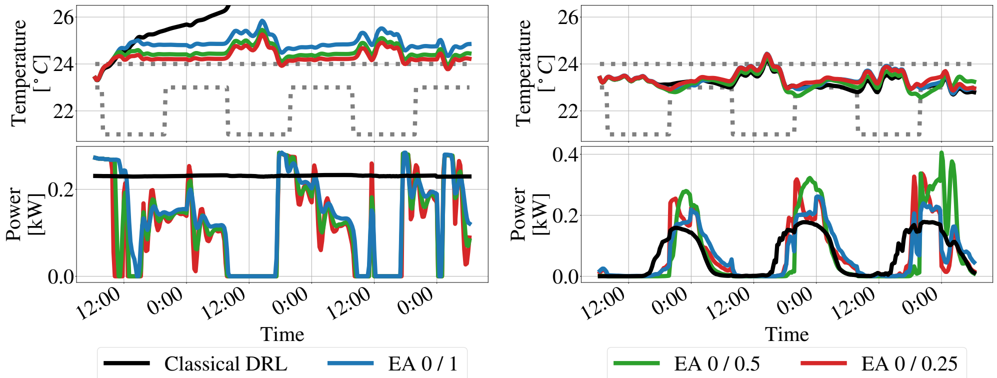

Fig. 3: Behavior of a classical agent and EAs with various m and n parameters (ours) minimizing the heating power consumption (bottom) while maintaining the temperature in the grey dotted bounds (top). **Left:** Performance *before* training, where EAs are saturated once they exceed the bounds by n degrees. **Right:** Performance *after* training, showing how all agents converged to similar solutions (Table II).

TABLE II: Reward, sum of comfort violations (Vio.), and energy consumption (En.) of each agent over the three days depicted on the right of Fig. 3.

Agent	Classical 1	EA (ours)		
m / n	-	0 / 1	0 / 0.5	0 / 0.25
Reward	-0.68	-0.69	-0.89	-0.60
Vio. [Kh]	1.28	1.18	2.23	0.65
En. [kWh]	5.03	5.38	5.54	5.46

C. Data efficiency of the proposed gradient modification

A comparison of the convergence speed of various agents over the first 300 epochs is plotted in Fig. 4, where the vertical lines and annotations illustrate the number of days required to attain performance on par with rule-based on-off industrial baselines from [6]. In general, we observe that all the EAs attain returns on par with the baselines significantly earlier than classical DRL agents, in as little as 29 days instead of roughly 200, an improvement of almost an order of magnitude. In particular, the smaller n is chosen (from left to right in Fig. 4), the faster the convergence of the EAs in green and blue. Intuitively, this makes sense, as tighter constraints introduce more prior knowledge to the EAs, thereby allowing them to find interesting solutions faster, without losing time exploring suboptimal state-action pairs. On the other hand, the influence of m is less marked, with $m \neq 0$ (blue) and $m = 0$ (green) leading to very similar convergence patterns in the bottom row of plots in Fig. 4.

Remarkably, RS does not seem to drastically speed the training up in this case study (red). While RS 0 / 0.25 does converge twice as fast as the classical DRL agents, RS 0 / 0.1 does not converge at all, hinting at the fragility of RS in general. Even when they converge, RSs are still two to three times slower than their EA counterparts. On the other hand, RS seems to lead to more consistent performance than classical agents and EAs after a few hundred epochs, which is confirmed by their good final performance in Fig. 2.

Overall, these results support our claim that, as long as the rules provided to the agents are well-defined and correspond to expected behaviors, the proposed modifications can indeed greatly accelerate the convergence of DRL agents. Critically, this does not significantly impact the quality of the final solution (Sec. V-A). Interestingly, incorporating more specific expert knowledge in EAs — through smaller m and n — further accelerates their convergence. This corresponds to our intuition: better-defined rules help agents more. Remarkably, the modifications proposed in Sec. III provide the desired speedup for a wide variety of parameters m and n, contrary to RS, hinting at the robustness of the proposed scheme.

VI. Conclusion

Starting from the postulate that prior expert knowledge often gives us an intuition of how good control policies should behave, we presented a scheme to encode it in actor-critic frameworks through simple rules to accelerate learning and decrease the associated computational load. These rules take the form of bounds on the agent's actions that can directly be enforced during training and online operations. To ensure agents learn from their mistakes, we also modified the actor gradients to steer control policies towards expected behaviors, limiting the exploration of known suboptimal state-action pairs. Critically, both these operations are computationally inexpensive, ensuring the gains in sample complexity positively impact the training time of the agents.

On a room temperature control case study, this scheme allowed us to accelerate the convergence speed of DRL agents by up to $6-7\times$. Furthermore, modifying actor gradients proved to be $2-3$ times more effective and more robust than the widespread reward shaping method. Remarkably, this was done without suffering from a significant drop in the final performance of the control policies, illustrating how prior knowledge can help alleviate the computational burden of DRL. This represents an interesting first step towards efficient agents that can be deployed and learned from scratch on physical systems, potentially bypassing the need for complex simulators. In future work, it would be interesting to investigate annealing strategies on λ or leverage primal-dual optimization tools to adaptively tune the influence of the additional penalty in the actor gradient and let agents learn more expressive policies after the initial exploration phase.

References

[1] A. Coronato, M. Naeem, G. De Pietro, and G. Paragliola, "Reinforcement learning for intelligent healthcare applications: A survey," *Artificial Intelligence in Medicine*, vol. 109, p. 101964, 2020.

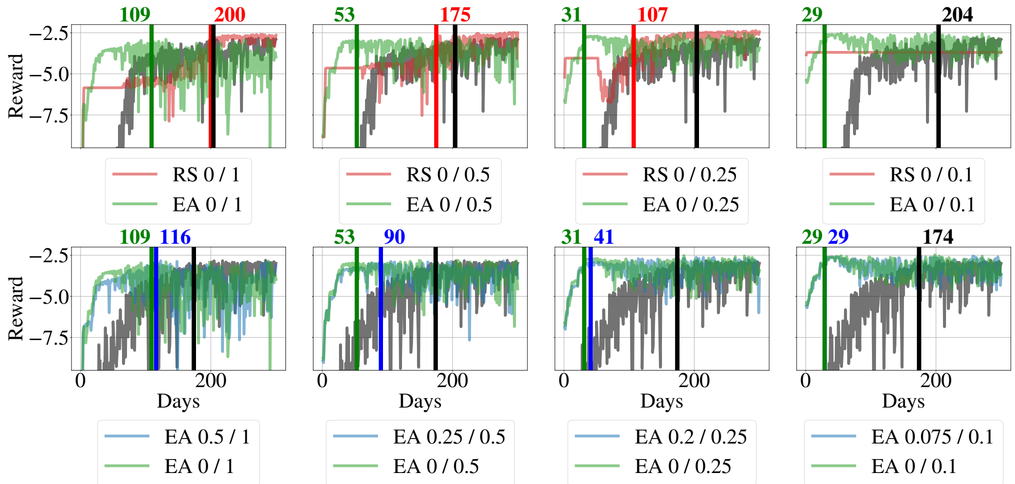

Fig. 4: Convergence speed of various EAs with different m and n parameters (ours), compared to agents using RS and two classical agents in black (one in the top plots, one in the bottom ones). The vertical lines and annotations specify the number of days of data required to obtain a reward of -2.95 for each agent, which corresponds to the performance of two industrial rule-based baselines.

[2] Z. Zhang, D. Zhang, and R. C. Qiu, "Deep reinforcement learning for power system applications: An overview," *CSEE Journal of Power and Energy Systems*, vol. 6, no. 1, pp. 213–225, 2019.

[3] Y. Duan, J. Schulman, X. Chen, P. L. Bartlett, I. Sutskever, and P. Abbeel, "RL^2: Fast reinforcement learning via slow reinforcement learning," *arXiv preprint arXiv:1611.02779*, 2016.

[4] M. Schwarzer, N. Rajkumar, M. Noukhovitch, A. Anand, L. Charlin, R. D. Hjelm, P. Bachman, and A. C. Courville, "Pretraining representations for data-efficient reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[5] Z. Wang and T. Hong, "Reinforcement learning for building controls: The opportunities and challenges," *Applied Energy*, vol. 269, p. 115036, 2020.

[6] L. Di Natale, B. Svetozarevic, P. Heer, and C. N. Jones, "Near-optimal Deep Reinforcement Learning Policies from Data for Zone Temperature Control," *arXiv preprint arXiv:2203.05434*, 2022.

[7] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Overcoming exploration in reinforcement learning with demonstrations," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 6292–6299.

[8] T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, D. Horgan, J. Quan, A. Sendonaris, I. Osband *et al.*, "Deep q-learning from demonstrations," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[9] B. De Cooman, J. Suykens, and A. Ortseifen, "Enforcing Hard State-Dependent Action Bounds on Deep Reinforcement Learning Policies," in *Machine Learning, Optimization, and Data Science: 8th International Workshop, LOD 2022, Certosa di Pontignano, Italy, September 19–22, 2022, Revised Selected Papers, Part II*. Springer, 2023, pp. 193–218.

[10] P. Zhang, J. Hao, W. Wang, H. Tang, Y. Ma, Y. Duan, and Y. Zheng, "KoGuN: accelerating deep reinforcement learning via integrating human suboptimal knowledge," *arXiv preprint arXiv:2002.07418*, 2020.

[11] R. Wang, X. Zhang, X. Zhou, Y. Wen, and R. Tan, "Toward Physics-Guided Safe Deep Reinforcement Learning for Green Data Center Cooling Control," in *2022 ACM/IEEE 13th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, 2022, pp. 159–169.

[12] H. Mao, M. Schwarzkopf, H. He, and M. Alizadeh, "Towards safe online reinforcement learning in computer systems," in *33rd conference on neural information processing systems (NeurIPS 2019)*, 2019.

[13] R. Bautista-Montesano, R. Galluzzi, K. Ruan, Y. Fu, and X. Di, "Autonomous navigation at unsignalized intersections: A coupled reinforcement learning and model predictive control approach," *Transportation Research Part C: Emerging Technologies*, vol. 139, p. 103662, 2022.

[14] H. H. Goh, Y. Huang, C. S. Lim, D. Zhang, H. Liu, W. Dai, T. A. Kurniawan, and S. Rahman, "An Assessment of Multi-Stage Reward Function Design for Deep Reinforcement Learning-Based Microgrid Energy Management," *IEEE Transactions on Smart Grid*, 2022.

[15] B. Chen, P. L. Donti, K. Baker, J. Z. Kolter, and M. Bergés, "Enforcing policy feasibility constraints through differentiable projection for energy optimization," in *Proceedings of the Twelfth ACM International Conference on Future Energy Systems*, 2021, pp. 199–210.

[16] S. Gros, M. Zanon, and A. Bemporad, "Safe reinforcement learning via projection on a safe set: How to achieve optimality?" *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 8076–8081, 2020.

[17] G. Dalal, K. Dvijotham, M. Vecerik, T. Hester, C. Paduraru, and Y. Tassa, "Safe exploration in continuous action spaces," *arXiv preprint arXiv:1801.08757*, 2018.

[18] P. Osinenko, D. Dobriborsci, and W. Aumer, "Reinforcement learning with guarantees: a review," *IFAC-PapersOnLine*, vol. 55, no. 15, pp. 123–128, 2022.

[19] S. Gu, L. Yang, Y. Du, G. Chen, F. Walter, J. Wang, Y. Yang, and A. Knoll, "A Review of Safe Reinforcement Learning: Methods, Theory and Applications," *arXiv preprint arXiv:2205.10330*, 2022.

[20] J. Garcıa and F. Fernández, "A comprehensive survey on safe reinforcement learning," *Journal of Machine Learning Research*, vol. 16, no. 1, pp. 1437–1480, 2015.

[21] T. D. Simão, N. Jansen, and M. T. Spaan, "AlwaysSafe: Reinforcement learning without safety constraint violations during training," in *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2021.

[22] J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained policy optimization," in *International conference on machine learning*. PMLR, 2017, pp. 22–31.

[23] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu, "Safe reinforcement learning via shielding," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[24] Y. Hu, W. Wang, H. Jia, Y. Wang, Y. Chen, J. Hao, F. Wu, and C. Fan, "Learning to utilize shaping rewards: A new approach of reward shaping," *Advances in Neural Information Processing Systems*, vol. 33, pp. 15 931–15 941, 2020.

[25] OpenAI, "Part 2: Kinds of RL Algorithms," 2018, accessed: 25.11.2022. [Online]. Available: https://spinningup.openai.com/en/latest/spinningup/rl_intro2.html#a-taxonomy-of-rl-algorithms

[26] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[27] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *International conference on machine learning*. PMLR, 2014, pp. 387–395.

[28] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International Conference on Machine Learning*. PMLR, 2018, pp. 1587–1596.

[29] L. Di Natale, B. Svetozarevic, P. Heer, and C. N. Jones, "Physically consistent neural networks for building thermal modeling: theory and analysis," *Applied Energy*, vol. 325, p. 119806, 2022.

[30] Empa, "NEST," 2022, accessed: 25.11.2022. [Online]. Available: https://www.empa.ch/web/nest/overview