Evo-RL: Evolutionary-Driven Reinforcement Learning

Ahmed Hallawa
RWTH Aachen University
Aachen, Germany
hallawa@ice.rwth-aachen.de

Thorsten Born
RWTH Aachen University
Aachen, Germany
thorsten.born@rwth-aachen.de

Anke Schmeink
RWTH Aachen University
Aachen, Germany
anke.schmeink@rwth-aachen.de

Guido Dartmann
Trier University of Applied Sciences
Trier, Germany
g.dartmann@umwelt-campus.de

Arne Peine
University Hospital Aachen
Aachen, Germany
apeine@ukaachen.de

Lukas Martin
University Hospital Aachen
Aachen, Germany
lmartin@ukaachen.de

Giovanni Iacca
University of Trento
Trento, Italy
giovanni.iacca@unitn.it

A. E. Eiben
Vrije Universiteit Amsterdam
Amsterdam, The Netherlands
a.e.eiben@vu.nl

Gerd Ascheid
RWTH Aachen University
Aachen, Germany
ascheid@ice.rwth-aachen.de

ABSTRACT

In this work, we propose a novel approach for reinforcement learning driven by evolutionary computation. Our algorithm, dubbed as Evolutionary-Driven Reinforcement Learning (Evo-RL), embeds the reinforcement learning algorithm in an evolutionary cycle, where we distinctly differentiate between purely evolvable (instinctive) behaviour versus purely learnable behaviour. Furthermore, we propose that this distinction is decided by the evolutionary process, thus allowing Evo-RL to be adaptive to different environments. In addition, Evo-RL facilitates learning on environments with rewardless states, which makes it more suited for real-world problems with incomplete information. To show that Evo-RL leads to state-of-the-art performance, we present the performance of different state-of-the-art reinforcement learning algorithms when operating within Evo-RL and compare it with the case when these same algorithms are executed independently. Results show that reinforcement learning algorithms embedded within our Evo-RL approach significantly outperform the stand-alone versions of the same RL algorithms on OpenAI Gym control problems with rewardless states constrained by the same computational budget.

CCS CONCEPTS

• **Computer systems organization** → **Robotic autonomy**; **Evolutionary robotics**;

KEYWORDS

Evolutionary computation, Artificial life, Reinforcement learning

ACM Reference Format:
Ahmed Hallawa, Thorsten Born, Anke Schmeink, Guido Dartmann, Arne Peine, Lukas Martin, Giovanni Iacca, A. E. Eiben, and Gerd Ascheid. 2021.

GECCO '21 Companion, July 10–14, 2021, Lille, France

ACM ISBN 978-1-4503-8351-6/21/07.
https://doi.org/10.1145/3449726.3459475

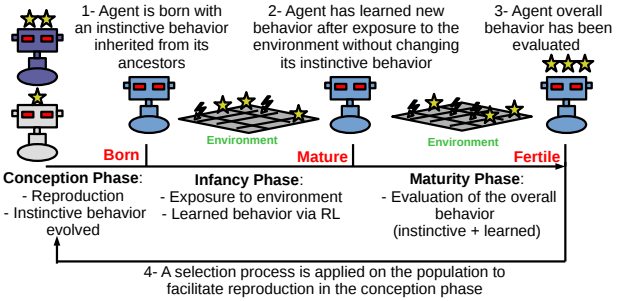

Figure 1: The Evo-RL scheme showing the agent life-cycle, highlighting its different phases and states.

Evo-RL: Evolutionary-Driven Reinforcement Learning. In *2021 Genetic and Evolutionary Computation Conference Companion (GECCO '21 Companion), July 10–14, 2021, Lille, France.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3449726.3459475

Proposed Model: We identify two types of behaviors: a purely evolved behavior, and a learnable behavior, which is driven by the agent's experience in its lifetime. The first one is dubbed as *instinctive behavior*. We formally define an *instinctive behaviour* as the evolved part of the agent's behaviour that is inherited from its ancestors and cannot be changed during the learning process within the lifetime of the agent. As for the second behavior, the learned behavior, we picture it as an extension to the evolved instinctive behavior. We define a *learnable behaviour* as the behaviour learned by the agent during its lifetime, as a result of its exposure to the environment. It should be noted that the learned behaviour cannot alter the instinctive behaviour. Finally, we define the *overall behaviour* as the combination of the agent's instinctive and learned behaviour, integrated together during the agent's lifetime. Furthermore, inspired by [1], we identify three states for the agent: the *born* state, which means that an agent has already an instinctive behaviour, but no learned behaviour. Note that the agent in this state is not exposed to the environment yet. The second state, dubbed as *mature*, means that the agent is already trained on the environment and has now an instinctive and a learned behaviour. Finally, the *fertile* state, means that the agent overall behaviour is already evaluated, therefore, a score reflecting its performance relative to a

Table 1: Final rewards after 60,000 evaluations (mean and standard error over 10 trials).

Environment		EA-Only	PPO	ePPO (ours)	DQN	eDQN (ours)
CartPole	0%	**140.9** ±20.2	**195.2** ±0.0 @ 4,370	**196.9** ±0.3 @ 14,400	**195.5** ±0.1 @ 160	**198.8** ±0.6 @ 1200
	10%	**115.1** ±19.8	**151.9** ±20.9	**197.0** ±0.5 @ 56,100	**195.5** ±0.1 @ 880	**199.5** ±0.4 @ 900
	20%	**115.1** ±19.8	**125.6** ±18.9	**196.8** ±0.5 @ 51,900	**195.6** ±0.1 @ 3680	**199.5** ±0.2 @ 900
	30%	**155.1** ±17.1	**114.6** ±18.8	**196.1** ±0.2 @ 51,600	**142.9** ±25.4	**199.2** ±0.5 @ 2100
	40%	N.A.	**112.6** ±16.5	**198.0** ±0.5 @ 47,100	**139.6** ±26.9	**198.1** ±0.6 @ 32100
	50%	N.A.	**81.0** ±20.5	**196.9** ±0.3 @ 31,500	**121.0** ±28.8	**198.5** ±0.6 @ 45600
Acrobot	0%	**-109.5** ±6.5	**-99.0** ±0.2 @ 12,300	**-99.0** ±0.2 @ 12,300	**-99.7** ±0.1 @ 1,270	**-95.8** ±0.8 @ 1,500
	10%	**-106.5** ±5.9	**-179.1** ±50.7	**-97.3** ±1.0 @ 47,100	**-118.0** ±17.7	**-94.1** ±1.3 @ 2,400
	20%	**-108.2** ±5.3	**-259.3** ±61.9	**-97.2** ±0.9 @ 5,700	**-99.8** ±0.1 @ 13,120	**-96.3** ±0.9 @ 1,500
	30%	**-99.1** ±5.8 @ 10,860	**-299.7** ±63.3	**-98.3** ±0.4 @ 15,000	**-99.7** ±0.1 @ 19,340	**-90.3** ±2.1 @ 7,800
	40%	N.A.	**-353.6** ±52.4	**-97.3** ±0.7 @ 50,400	**-139.5** ±38.0	**-93.1** ±1.3 @ 6,300
	50%	N.A.	**-427.5** ±46.2	**-101.2** ±5.5	**-175.6** ±45.9	**-90.8** ±1.5 @ 6,600

pre-defined objective can be computed. The outer loop of Evo-RL is an evolutionary algorithm. Similar to any EA, Evo-RL starts by initializing a set of individuals (agents), thus, producing a population of agents. In the first iteration, each agent has a randomly initialized behaviour. For example, the phenotype can be an Artificial Neural Network (ANN) representing the agent behaviour, while the genotype can be a set of binary numbers that when decoded produce the ANN of that agent's behaviour. After this initialization, each agent in the population is considered in the born state, and has an instinctive behaviour. As shown in Figure 1, after birth, the agent starts to be exposed to the environment. In this *infancy phase*, reinforcement learning is executed. However, the agent cannot overwrite its instinctive behaviour, thus, if the agent is in a state where the instinctive behaviour has already defined an action to execute, the agent executes this action and no learning is done with respect to this state. On the other hand, if the agent is in a state where the instinctive behaviour does not define what should be done, then the agent proceeds with its learning algorithm normally. After the infancy phase ends, due to resource or time constraints for example, the agent reaches the mature stage and is now ready to be evaluated. In the *maturity phase*, the agent overall behaviour is evaluated with respect to pre-set objectives. A score that measures its performance is then calculated. To summarize, the overall approach is an evolutionary computation approach. However, the novelty can be highlighted in the following design axioms: (1) The choice of which part of the overall behaviour is instinctive, and which is not, is decided by the evolutionary process. In other words, the line between what is instinctive (fully evolvable) and learnable, is evolved. This is achieved by not allowing the learning process to overwrite the instinctive behaviour. Evolution dictates which region of states it operates on. (2) The overall fitness of an agent considers both behaviors, i.e., instinctive plus learnable. This is facilitated by conducting the evaluation of the behaviour after the learning process is conducted. (3) In the conception phase, only the instinctive behaviour is evolved, but the learned behaviour is transferred to the offspring to allow plasticity in the learned behaviour as long as the instinctive behaviour allows it. In case of conflict, the instinctive behaviour overwrites any learnable behaviour. For evaluation purposes, we have implemented the algorithm as follows. Firstly, we used the EA in the form of Genetic Programming (GP). However, as for what concerns the representation of the instinctive (evolved) behaviour, we adopted *behaviour trees* (BTs). These fit well with GP and, unlike ANN, are much easier to interpret. As for the learned behaviour, we adopted two possibilities, one tabular representation used when testing our approach with Q-learning, and another ANN representation when testing our approach with Proximal Policy Optimization (PPO) and Deep Q-Network (DQN).

Experimental Results: We designed our experimental evaluation to test the three following hypotheses: (1) The performance of reinforcement learning algorithms is enhanced when embedded in the Evo-RL approach for environments with rewardless states, with the same fixed computational budget. (2) The performance of Evo-RL is better than the evolutionary algorithm part alone (i.e., Evo-RL without the reinforcement learning). In other words, we want to show that instinctive behaviour plus learnable behaviour (Evo-RL) outperforms adopting only instinctive behaviour (EA-only) or only learnable behaviour (RL-only). (3) As the rewardless states increase in an environment, the ratio of instinctive behaviour executed, compared to the learnable one, increases as well. This shows that the instinctive behaviour is necessary to handle more efficiently the rewardless states. In order to facilitate testing on environments with a rewardless state, we modified three OpenAI gym control problems to obtain rewardless state problems: Cartpole, Acrobot and MountainCar. The modifications can be summarized as follows. The state space of each problem is discretized into bins. Whenever a problem is initialized, a predefined percentage of these bins is marked as rewardless states. If, while learning, an agent reaches one of those states, no feedback from the environment is given back to the agent. More precisely, the reward of rewardless states is not zero, it is just a state with no reward signal, i.e. no new information given from the environment to the learning process at this state. In our experiments, we tried setting the percentage of rewardless states to 0%, 10%, 20%, 30%, 40% and 50%. Table 1 summarize the final rewards after the last evaluation for all problems and algorithms. The number after the "@" denotes the number of evaluations needed for solving the problem. Blue indicates that the problem is solved, while red indicates that it was not solved.

REFERENCES

[1] AE Eiben, Nicolas Bredeche, M Hoogendoorn, J Stradner, J Timmis, A Tyrrell, A Winfield, et al. 2013. The triangle of life: Evolving robots in real-time and real-space. *Advances in artificial life, ECAL* 2013 (2013), 1056.