Proc. of the Nineteenth Intl. Conf. on Computational Intelligence and Security (CIS 2023)
Editors Yuping Wang, Xiao-zhi Gao, Lalit Mohan Patnaik, David Camacho, and Hailin Liu

doi: 10.3850/978-981-18-8813-7_P236-cd

CIS 2023
Haikou, China

Evolutionary Reinforcement Learning by Rank-one Evolution Strategy with Population Size Control

Shuo Wang

School of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China. E-mail: wangshuo_77@nuaa.edu.cn

Zhenhua Li

School of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China. E-mail: zhenhua.li@nuaa.edu.cn

Minghui Hu

School of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China. E-mail: minghuihu@nuaa.edu.cn

Deep reinforcement learning has made significant breakthroughs in numerous fields and is considered a meaningful way to achieve general artificial intelligence. However, the existing deep reinforcement learning algorithms still face some difficulties in dealing with real problems, such as long time range and sparse reward credit allocation, lack of effective diversified exploration and sensitivity to the selection of hyperparameters. To this end, we propose an evolutionary reinforcement learning algorithm, RPSA-RL, which combines an evolutionary algorithm based on population search with a deep reinforcement learning algorithm that utilizes problem gradient information. We evaluated the performance of RPSA-RL with state-of-the-art reinforcement learning algorithms on six common DRL continuous control tasks in the OpenAI Gym test bed. The results show that the proposed algorithm is superior to or better than the most advanced algorithm, which proves the effectiveness of the proposed algorithm.

Keywords: Deep reinforcement learning, evolutionary algorithm, noise processing method, population size adaptation.

1. Introduction

Deep Reinforcement Learning (DRL) has yielded significant advancements within the domain of large-scale real-time strategy games[Mnih et al. (2013, 2015); Silver et al. (2016)], robotic control paradigms, machine vision applications, and various cognate fields. Nevertheless, the problem of applying these techniques to a wide range of real-world applications still faces the difficulties of assigning credits with long time scales and sparse rewards, lack of effective diversity exploration, and sensitivity to the choice of hyperparameters.

Evolutionary algorithms offer a compelling avenue for the principled resolution of the above-mentioned challenges within the deep reinforcement learning domain[Fogel (2006)]. Evolutionary algorithms use the overall reward value of an episode as the fitness of the intelligence, making it less susceptible to the problems of sparse reward allocation and long-time horizon credit allocation[Such et al. (2017)]. Furthermore, the inherent stochasticity within the population dynamics enhances the evolutionary algorithm's capacity for robustness and steadfast convergence when confronting intricate problem domains. Uncertainty and noise exist in the reward and state transfer distribution functions of deep reinforcement learning algorithms[Sutton and Barto (2018)]. OpenAI introduces a simplistic evolution strategy (OpenAI-ES) to solve reinforcement learning problems. This algorithm uses a simple Gaussian distribution for sampling, randomly selects policy from the current distribution and calculates fitness by interacting with the environment. The stochastic gradient is then calculated based on the fitness and used to update the distribution parameters[Salimans et al. (2017)]. However, the introduction of noise processing mechanism is rarely considered in the

current research on the development of deep reinforcement learning algorithms using evolutionary algorithms, and only a few studies use uncertainty processing technology[Müller and Glasmachers (2018)] and restart strategy[Chen et al. (2019)] to improve the performance of evolutionary algorithms. For search space with high latitude, R1-ES[Li and Zhang (2017)] is the simplest method to deal with such high dimensional problems.

In response to the above difficulties faced by deep reinforcement learning, the present study introduces an evolutionary reinforcement learning construct coined as RPSA-RL. By combining the noise processing method RPSA with the existing large-scale optimization algorithm R1-ES[Li and Zhang (2017)] and the mainstream deep reinforcement learning algorithm SAC(Soft Actor-Critic)[Haarnoja et al. (2018)] based on gradient optimization to form the evolutionary reinforcement learning algorithm RPSA-RL. The main features of RPSA-RL are summarized as follows.

(1) RPSA-RL effectively alleviates the problems faced by the mainstream DRL algorithm, such as too long and sparse reward credit allocation time, lack of practical diversity exploration, and sensitivity to hyperparameters. The RPSA mechanism further improves the algorithm's performance in DRL problems with high noise intensity.

(2) Evolutionary reinforcement learning algorithm RPSA-RL is a general framework. It can use other advanced evolutionary algorithms for solving large-scale optimization problems[Mei et al. (2016); Loshchilov (2017); Sun et al. (2017); Li et al. (2018)] and off-policy deep reinforcement learning algorithms[Mnih et al. (2016); Fujimoto et al. (2018)] to replace the corresponding algorithms R1-ES and SAC in RPSA-RL.

The subsequent discourse of this exposition is structured as follows. Section 3 proffers an incisive exposition of the introduced RPSA-RL algorithm. In Section 4, the performance of the RPSA-RL algorithm is validated and evaluated on continuous control tasks commonly used by DRL using the OpenAI Gym test platform[Brockman et al. (2016)]. The summative synthesis of the paper's essence converges within Section 5.

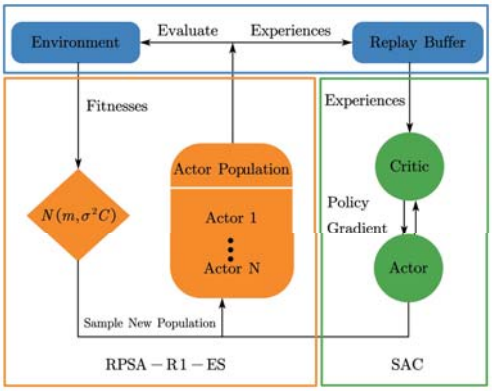


Fig. 1. Schematic diagram of RPSA-RL algorithm.

2. Background and Prerequisite

The goal of reinforcement learning is to maximise the rewards an intelligent body gains during its interaction with the environment, usually modelled as a Markov decision process (MDP). The MDP is represented by a quaternion (S, A, T, R), where S is the state space, A is the action space, T is a state transfer function indicating that the execution of action a from the current state s will transition to the next state s', and R is a reward function defined at the immediate reward r obtained by acting a in the current state s.

The total reward for an intelligent body interacting with the environment is expressed in terms of payoffs, and the discounted payoffs earned from moment t to the end of the round are:

$$U_t = \sum_{k=0}^{\infty} \gamma^k \cdot R_{t+k}, \tag{1}$$

where $\gamma \in [0, 1]$ is a discount factor to adjust the weight of future rewards received.

The action value function is the expectation of the reward that an intelligent body can obtain by acting a in state s, defined as:

$$\begin{aligned} Q_\pi(s_t, a_t) =& \mathbb{E}_{S_{t+1}, A_{t+1}, \cdot\cdot, S_n, A_n} \\ & [U_t | S_t = s_t, A_t = a_t], \end{aligned} \tag{2}$$

where the action choice of an intelligent body is determined by the strategy function π. Therefore, the action value function depends on the strategy π.

The state value function is the expectation of the payoff that the intelligence can obtain in state s based on the strategy π, defined as:

$$V_\pi(s_t) = \mathbb{E}_{A_t, S_{t+1}, A_{t+1}, \cdots, S_n, A_n}[U_t | S_t = s_t]. \tag{3}$$

The policy learning approach uses a neural network to approximate the policy function, derived directly by solving an optimisation problem. The optimal policy should have a large mean value of the state values for all states, and therefore, the expectation of the state values is used as the objective function:

$$J(\theta) = \mathbb{E}_S[V_\pi(S)], \tag{4}$$

where θ is a parameter of the policy network, an approximation of the optimal policy function can be derived by maximising this objective function. Maximising the objective function can be done using the gradient ascent method, where the objective function gradients over the parameters θ to obtain the policy gradient:

$$\frac{\partial J(\theta)}{\partial \theta} = \mathbb{E}_S[\mathbb{E}_{A\sim\pi(\cdot|S;\theta)}[\frac{\partial \ln(A|S;\theta)}{\partial \theta}] \cdot Q_\pi(S, A)]. \tag{5}$$

Reinforcement learning actor and critic parameter updates are described below.

(1) Actor π_θ interacts with the environment and creates the experience of a tuple (s, a, r, s', d) in replay buffer R. B trajectory cells $\{(s, a, r, s', d)\}$ are randomly sampled from the replay buffer R.

(2) Update all actors and critics of gradient-based deep reinforcement learning algorithm.

Calculate the goals of the critic:

$$y(r, s', d) = r + \gamma(1-d)(\min_{k=1,2} Q_{\phi_{\text{uxy},k}}(s', \tilde{a}') - \alpha \log \pi_\theta(\tilde{a}'|s')), \tilde{a}' \sim \pi_\theta(\cdot|s') \tag{6}$$

Update critic:

$$\nabla_{\phi_k} \frac{1}{|B|} \sum_{(s,a,r,s',d)\in B} (Q_{\phi_k}(s, a) - y(r, s', d))^2, \quad \text{for } k = 1, 2 \tag{7}$$

Update actor:

$$\nabla_\theta \frac{1}{|B|} \sum_{s\in B} (\min_{k=1,2} Q_{\phi_k}(s, \tilde{a}_\theta(s)) - \alpha \log \pi_\theta (\tilde{a}_\theta(s) \mid s)), \tilde{a}_\theta(s) \sim \pi_\theta(\cdot|s) \tag{8}$$

Update the target evaluator:

$$\phi_{\text{targ},k} \leftarrow \rho\phi_{\text{targ},k} + (1-\rho)\phi_k, \text{ for } k = 1, 2 \tag{9}$$

(3) Introduce gradient information into evolutionary algorithms and periodically replace the actor of reinforcement learning algorithms with the one with the lowest fit in evolutionary algorithms.

3. The Proposed Method

In this section, we describe the proposed RPSA-RL in detail.

3.1. *Rank One Es*

R1-ES uses sparse and low-rank decomposition of covariance to achieve low computational complexity. It has linear time complexity O(n) and low space complexity. R1-ES uses the evolutionary path as the primary search direction. The evolutionary path accumulates a natural gradient of expected fitness relative to the mean of the distribution and acts as a momentum term under stationary conditions.

At iteration t, R1-ES samples a population of λ new candidate solutions from the current distribution as

$$\pi_i = \mathbf{m}_t + \sigma_t \mathbf{y}_i, i = 1, \ldots, \lambda, \tag{10}$$

where $\mathbf{y}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{C}_t)$ is a search direction. These candidate solutions are evaluated by the objective function and sorted according to the noisy objective values $\widetilde{f}(\pi_{1;\lambda}) \leq \widetilde{f}(\pi_{2;\lambda}) \leq \cdots \leq \widetilde{f}(\pi_{\lambda;\lambda})$, where the subscript $i : \lambda$ denotes the i-th best candidate solution.

The distribution mean is updated by the weighted sum of the best $\mu = \lfloor \frac{\lambda}{2} \rfloor$ candidate solutions as

$$\mathbf{m}_{t+1} = \sum_{i=1}^{\mu} \omega_i \pi_{i:\lambda} \tag{11}$$

where $w_i > 0, i = 1, \ldots, \mu$ are the weights for multi-recombination. The principal search direction $\mathbf{p}$ constructed by accumulating the movements of $\mathbf{m}_t$ and adapt the covariance matrix as

$$\mathbf{p}_{t+1} = (1-c)\mathbf{p}_t + \sqrt{c(2-c)\mu_w} \frac{\mathbf{m}_{t+1} - \mathbf{m}_t}{\sigma_t} \tag{12}$$

with changing rate $c \in (0, 1)$ and $\mu_w = 1/\sum_{i=1}^{\mu} w^2$. Compute the rank difference

$$q = \frac{1}{\mu} \sum_{i=1}^{\mu} \omega_i \left(R_t(i) - R_{t+1}(i)\right) \tag{13}$$

where the weights $w_i > 0, i = 1, \ldots, \mu$, and the factor $\frac{1}{\mu}$ is used to normalize $q \in [-1, 1]$. Compute the cumulative rank rate

$$s_{t+1} = (1-c_s)s_t + c_s(q - q^*) \tag{14}$$

where $q^* \in (0, 0.5)$ is a constant target ratio, $c_s > 0$ is a constant changing rate, and $s_0 = 0$. Adapt the mutation strength as

$$\sigma_{t+1} = \sigma_t \exp\left(\frac{s_{t+1}}{d_\sigma}\right) \tag{15}$$

where $d_\sigma \geq 1$ is a damping factor.

3.2. *Population Size Control Based on Rank Change*

We estimate noise levels by tracking the average difference in rank between two consecutive generations and adjusting the population size based on the estimated noise levels.

(1) Set $F = F_t \cup F_{t+1}$, where F_t and F_{t+1} denote a population of objective values at iteration t and $(t+1)$, respectively. All the elements in F are then ranked in terms of their objective values. The ranks of the i-th best solutions from F_t and F_{t+1} are denoted by $R_t(i)$ and $R_{t+1}(i)$, respectively.

(2) Calculate the average rank of each population

$$A_t = \frac{1}{\lambda_{t-1}} \sum_{i=1}^{\lambda_{t-1}} R_t(i), \tag{16}$$

$$A_{t+1} = \frac{1}{\lambda_t} \sum_{i=1}^{\lambda_t} R_{t+1}(i). \tag{17}$$

(3) Calculate and normalize the average rank difference

$$r = \frac{2}{\lambda_{t-1} + \lambda_t} \cdot (A_{t+1} - A_t), \tag{18}$$

(4) The rank difference is smoothed over generations as

$$\psi_{t+1} = (1 - c_\lambda)\psi_t + c_\lambda(r - r^*), \tag{19}$$

where the parameter $r^* \in (0, 0.5)$ controls the acceptance threshold for the estimated average rank difference. In practice, it also denotes the noise level we can tolerate.

(5) Update the population size according to the estimated cumulative noise level ψ_{t+1} as follows

$$\lambda_{t+1} = \lambda_t \cdot \exp\left(\frac{\psi_{t+1}}{d_\lambda}\right) \tag{20}$$

The damping parameter $d_\lambda \geq 1$ reduces the variance of the population size change, and a large d_λ can slow down the adaptation. In this article, we set $d_\lambda = 1$ for rapid adaptation.

Algorithm 1 The Framework of RPSA-RL

Use weights $\theta^\pi, \phi_1, \phi_2$ to initialize actor π_θ and the two critical $\mathcal{Q}_{\phi_1}, \mathcal{Q}_{\phi_2}$
Use weights ϕ_1, ϕ_2 to initialize the weight of the two target critics $\phi_{\text{targ},1} \leftarrow \phi_1, \phi_{\text{targ},2} \leftarrow \phi_2$
Initializes the parameter mean $\mathbf{m}_0$ of R1-ES, step size σ_0, evolutionary path $\mathbf{p}_0$, set of fit values F_0, and an empty loop replay buffer R
for $t = 0$ **to** ∞ **do**
  // Sample λ candidate solutions as actors
  for i = 1 to λ_t **do**
    $\mathbf{z}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I}),\ r_i \sim \mathcal{N}(0, 1)$
    $\pi_i = \mathbf{m}_t + \sigma_t \cdot (\sqrt{1 - c_{cov}}\mathbf{z}_i + \sqrt{c_{cov}} r_i \mathbf{p}_t)$
    // Evaluate the newly generated actor
    $f(\pi_i)$, R = Evaluate(π_i, R)
  end for
  // R1-ES parameter update
  sort π_i as $f(\pi_{1:\lambda}) \leq f(\pi_{2:\lambda}) \leq \cdots \leq f(\pi_{\lambda:\lambda})$
  $\mathbf{m}_{t+1} = \sum_{i=1}^{\mu} \omega_i \pi_{i:\lambda}$
  $\mathbf{p}_{t+1} = (1-c)\mathbf{p}_t + \sqrt{c(2-c)\mu_w}\frac{\mathbf{m}_{t+1} - \mathbf{m}_t}{\sigma_t}$
  $R_t, R_{t+1} \leftarrow$ ranks of F_t, F_{t+1} in $F_t \cup F_{t+1}$
  $q = \frac{1}{\mu}\sum_{i=1}^{\mu} \omega_i \left(R_t(i) - R_{t+1}(i)\right)$
  $s_{t+1} = (1 - c_s)s_t + c_s(q - q^*)$
  $\sigma_{t+1} = \sigma_t \exp\left(\frac{s_{t+1}}{d_\sigma}\right)$
  // Noise processing method RPSA
  λ_{t+1} = RPSA (F_t, F_{t+1}, λ_t)
  // Evaluate the actor of RL
  _, R = Evaluate(π_θ, R)
  // Update parameters for all actors and critics of RL
  for j = 1 to number of updates **do**
    Procedure mentioned in Section 2.
  end for
  if $t \bmod \omega = 0$ **then**
    Replace the actor of RL with the actor of the lowest fitness value in R1-ES: $\theta' \Leftarrow \theta$
  end if
  t = t + 1
end for

3.3. *The Framework of RPSA-RL*

The evolutionary reinforcement learning algorithm RPSA-RL combines evolutionary and deep reinforcement learning algorithms. RPSA-R1-ES generates a large number of experiences of diversity exploration in the process of interacting with the environment. These experiences are stored in the replay buffer for the train-

ing of the SAC of the DRL algorithm.

Figure 1 shows the overall framework of the RPSA-RL algorithm. The following delineated steps primarily characterize the algorithmic exposition.

(1) Initialize the parameters (lines 1-3), initialize all the Actors and Critics in the SAC, initialize the R1-ES with parameter mean $\mathbf{m}_0$, step size σ_0, evolutionary path σ_0, set of adaptation values F_0, and an empty cyclic replay buffer R for storing experience.

(2) Generate λ_t new solutions as new actors from the current distribution of R1-ES and interact with the environment by taking the resulting empirical tuple (s, a, r, s', d) in the replay buffer R, where d indicates whether the prologue is over or not ($d = 0$ means it has not been ended, $d = 1$ means ended). The total reward value obtained is the adaptation value (Lines 5-9). Based on the newly generated actuator and adaptation values, update the parameters of R1-ES (lines 10-16).

(3) Perform the population size-based adaptive noise processing method RPSA (line 17).

(4) Lines 18-25 are introduced in Section 2.

Parameter Settings: RPSA-RL uses Adam[Kingma and Ba (2014)] as the neural network parameter optimizer, and the learning rate of the actuator and evaluator is set to $5e^{-5}$ and $5e^{-4}$, respectively. The initial population size of R1-ES λ_0 is set to 10. The size of the replay buffer R is set to $1e^6$, and the batch size is 128. SAC's discount factor γ is set to 0.99. The synchronization period ω is set to 1.

4. Experimental Studies

This section comprehensively evaluates the evolutionary reinforcement learning paradigm, RPSA-RL, across six continuous control tasks emblematic of the deep reinforcement learning domain. The assessment uses the OpenAI Gym test platform[Brockman et al. (2016)].

This experiment compared RPSA-RL with R1-ES-RL, R1-ES, and SAC algorithms. The graphical depiction in Figure 2 delineates the learning trajectory of all algorithms in the continuous task test suite, where the abscissa signifies the number of computational steps undertaken. At the same time, the ordinate denotes the magnitudes of rewards garnered. Every algorithm is independently executed five times in each iteration, with the continuous line denoting the mean trajectory and the shaded segment encapsulating the variance. The main observations and discussions of this experiment are summarized as follows.

- RPSA-RL outperforms R1-ES-RL, R1-ES, and SAC on all continuous control tasks, especially on Swimmer-v2, Ant-v2, and Walker2d-v2 tasks, where the performance gap is more pronounced.
- RPSA-RL and R1-ES-RL are hybrid algorithms that outperform their key components, R1-ES and SAC, on all tasks, which validates the effectiveness of combining evolutionary and deep reinforcement learning algorithms.
- RPSA-RL is better than R1-ES-RL on the continuous task test set, especially on the tasks Swimmer-v2, Ant-v2, and Walker2D-v2, where the performance gap between the two is more prominent. Compared with R1-ES-RL, RPSA-RL introduces an additional noise processing method RPSA, and the experimental results verify the effectiveness of RPSA.

5. Conclusion

In this paper, we firstly combine the RPSA noise processing method proposed in this paper with the existing large-scale optimization algorithm R1-ES, propose a large-scale noise evolutionary optimization algorithm RPSA-R1-ES, and then combine it with the mainstream deep reinforcement learning algorithm SAC based on gradient optimization to form the evolutionary reinforcement learning algorithm RPSA-RL. RPSA-R1-ES proposes compelling mitigation strategies for the various challenges plaguing mainstream deep reinforcement learning paradigms, the most notable of which address the difficulties faced by mainstream deep reinforcement learning algorithms such as longer and sparse reward credit allocation, lack of practical diversity exploration and sensitivity to hyper-parameters, and at the same time, deep reinforcement learning algorithms based on gradient optimization can make full use of the gradient information of the problem to improve the utilization rate of samples, and thus accelerate the convergence of the algorithms. The empirical validation of the efficacy of RPSA-RL materializes within the crucible of the DRL continuous control task test suite, thereby confirming its potent efficacy.

Acknowledgement

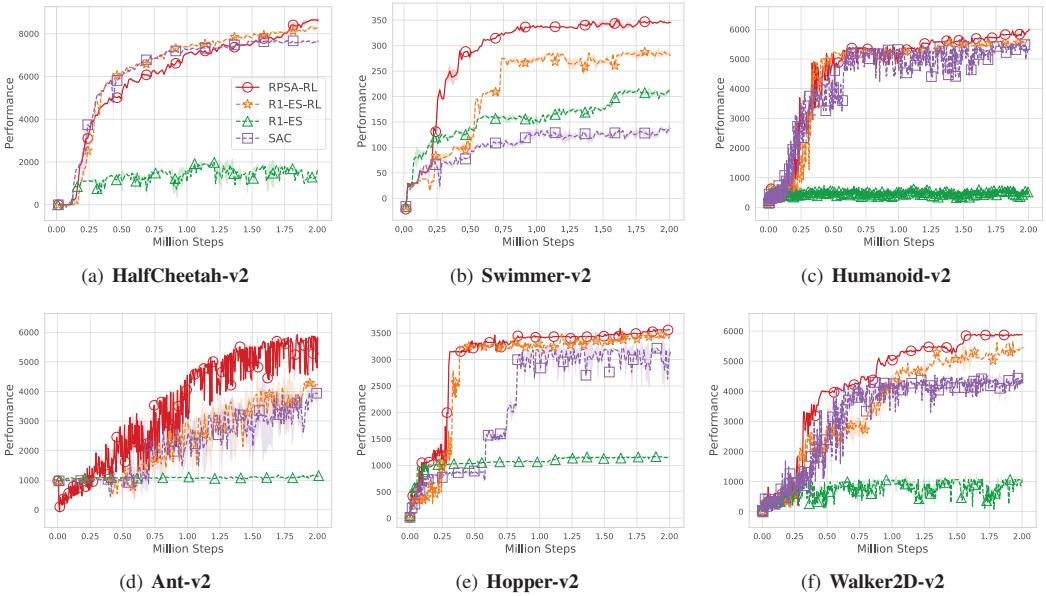
This work was supported by the National Natural Science Foundation (Grant No.62102178) and Natural Science Foundation of Jiangsu Province (Grant No.BK20200443).

References

Brockman, G., V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba (2016). Openai gym. *ArXiv abs/1606.01540*.

Chen, Z., Y. Zhou, X. He, and S. Jiang (2019). A restart-based rank-1 evolution strategy for reinforcement learning. In *IJCAI*, pp. 2130–2136.

Fogel, D. B. (2006). *Evolutionary computation: toward a new philosophy of machine intelligence*. John Wiley & Sons.



(a) **HalfCheetah-v2** (b) **Swimmer-v2** (c) **Humanoid-v2**

(d) **Ant-v2** (e) **Hopper-v2** (f) **Walker2D-v2**

Fig. 2. Compare the learning curve of the algorithm on continuous control tasks.

Fujimoto, S., H. Hoof, and D. Meger (2018). Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pp. 1587–1596. PMLR.

Haarnoja, T., A. Zhou, P. Abbeel, and S. Levine (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. PMLR.

Kingma, D. P. and J. Ba (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Li, Z. and Q. Zhang (2017). A simple yet efficient evolution strategy for large-scale black-box optimization. *IEEE Transactions on Evolutionary Computation 22*(5), 637–646.

Li, Z., Q. Zhang, X. Lin, and H.-L. Zhen (2018). Fast covariance matrix adaptation for large-scale black-box optimization. *IEEE transactions on cybernetics 50*(5), 2073–2083.

Loshchilov, I. (2017). Lm-cma: An alternative to l-bfgs for large-scale black box optimization. *Evolutionary computation 25*(1), 143–171.

Mei, Y., M. N. Omidvar, X. Li, and X. Yao (2016). A competitive divide-and-conquer algorithm for unconstrained large-scale black-box optimization. *ACM Transactions on Mathematical Software (TOMS) 42*(2), 1–24.

Mnih, V., A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu (2016). Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937. PMLR.

Mnih, V., K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Mnih, V., K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. (2015). Human-level control through deep reinforcement learning. *nature 518*(7540), 529–533.

Müller, N. and T. Glasmachers (2018). Challenges in high-dimensional reinforcement learning with evolution strategies. In *Parallel Problem Solving from Nature–PPSN XV: 15th International Conference, Coimbra, Portugal, September 8–12, 2018, Proceedings, Part II 15*, pp. 411–423. Springer.

Salimans, T., J. Ho, X. Chen, S. Sidor, and I. Sutskever (2017). Evolution strategies as a scalable alternative to reinforcement learning. arxiv 2017. *arXiv preprint arXiv:1703.03864*.

Silver, D., A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. (2016). Mastering the game of go with deep neural networks and tree search. *nature 529*(7587), 484–489.

Such, F. P., V. Madhavan, E. Conti, J. Lehman, K. O. Stanley, and J. Clune (2017). Deep neuroevolu-

tion: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. *arXiv preprint arXiv:1712.06567*.

Sun, Y., M. Kirley, and S. K. Halgamuge (2017). A recursive decomposition method for large scale continuous optimization. *IEEE Transactions on Evolutionary Computation 22*(5), 647–661.

Sutton, R. S. and A. G. Barto (2018). *Reinforcement learning: An introduction*. MIT press.