EasyChair Preprint

№ 5309

# Distributed Species-Based Genetic Algorithm for Reinforcement Learning Problems

Anirudh Seth, Alexandros Nikou and Marios Daoutis



April 8, 2021

# Distributed species-based genetic algorithm for reinforcement learning problems

Anirudh Seth[1,2][0000−0003−3762−2578], Alexandros Nikou[2][0000−0002−8696−1536], and Marios Daoutis[2][0000−0003−0401−6897]

[1] KTH Royal Institute of Technology, Brinellvägen 8, 114 28 Stockholm, Sweden
[2] Ericsson Research, Torshamnsgatan 23, 16440, Stockholm, Sweden
{anirudh.seth, alexandros.nikou, marios.daoutis}@ericsson.com

**Abstract.** Reinforcement Learning (RL) offers a promising solution when dealing with the general problem of optimal decision and control of agents that interact with uncertain environments. A major challenge of existing algorithms is the slow rate of convergence and long training times especially when dealing with high-dimensional state and action spaces. In our work, we leverage evolutionary computing as a competitive alternative to training deep neural networks for RL problems. We present a novel distributed algorithm based on efficient model encoding which enables the intuitive application of genetic operators. Another contribution is the application of crossover operator in two neural networks in the encoded space. Preliminary results demonstrate a considerable reduction of trainable parameters and memory requirements while maintaining comparable performance with DQN and A3C when evaluated on Atari games, resulting in an overall significant speedup.

**Keywords:** Neuro-evolution strategies · Model Encoding · Distributed Speciation · Reinforcement Learning · Genetic Algorithms

## 1 Introduction

Reinforcement Learning (RL) is an active area of research within artificial intelligence routinely applied in a wide range of domains, for example robotic manipulation [1], personalized recommendations [10] and recently in novel areas such as telecommunication [6]. Existing methods, e.g. Q-learning [4] and policy gradients [3], train deep neural networks by back-propagation of gradients. Such approaches usually entail expensive computations, often not trivially parallelizable, which result in hours or even days of training in order to obtain desirable results especially when solving complex problems with large state-spaces [3, 4].

Alternative approaches to solve RL problems include Augmented Random Search (ARS) [2], evolution strategies [7] and deep neuro-evolution [9]. The latter two approaches belong to a class of black-box optimisation techniques which are based on principles of evolutionary computation. The results highlight the robustness of these approaches to sparse/dense rewards, tolerance to arbitrarily long time horizons as a consequence of gradient free learning and significant speedup due to distributed training [7, 9].

In our work, we propose a novel distributed species-based genetic algorithm aimed at solving RL problems, which trains a deep neural network using the genetic operators mutation, selection and crossover. The algorithm relies on an

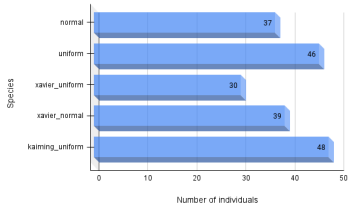


*Fig. 1: Distribution of species in the initial population.*

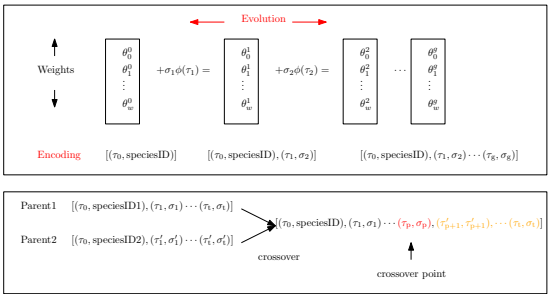


*Fig. 2: Graphical representation of the model encoding, mutation and crossover.*

efficient and compact model encoding that scales well in memory and utilizes extremely low bandwidth making it highly scalable. The performance is presently evaluated in Atari 2600 benchmarks and is compared to established algorithms such as DQN [5] and A3C [3].

## 2 Methods

As in any other typical genetic algorithm or population based optimisation method, we evolve a population $\mathcal{P}$ of $\mathcal{N}$ individuals (candidate solutions to the problems – i.e. a *DQN* network). The concept of speciation to cluster the individuals with topological similarities was first introduced in NEAT [8]. We build upon the same idea and randomly generate $\mathcal{S}$ classes of species, with each class represented by a unique NN layer weight initialisation strategy, that promotes diversity and protects innovation within the population pool. We call this algorithm *Sp-GA* in the following text.

At each generation, we evaluate the entire population using a fitness function (cumulative reward for a RL problem). A fraction of the population $\mathcal{T}$ with highest fitness scores are selected as parents for the current generation. These parents then generate the offsprings for the subsequent generation using genetic operators like mutations, crossover and selection. The mutation operator selects a single parent uniformly at random from $\mathcal{T}$ and updates all the parameters by applying additive Gaussian noise as $\theta' = \theta + \sigma\epsilon$ where $\epsilon \sim \mathcal{N}(0, I)$ and $\sigma$ is the mutation power. The individual with the highest fitness is copied as-is to the next generation, a concept called elitism. Crossover operation selects a pair from $\mathcal{T}$ to generate a new individual with features from each parent. The parameter $\psi$ represents the probability of using mutation and $1 - \psi$ for crossover.

In our work, we propose a novel way to perform single point crossover of two neural networks to produce one child neural network in the encoded space. This evolutionary process is repeated until a new population of size $\mathcal{P} + 1$(elite) is generated completing one epoch/generation of the genetic algorithm.

### 2.1 Model encoding and genetic operators

Authors in [7, 9] employ lists of random seeds for distributed computing but these approaches are limited to a vanilla genetic algorithm without crossover. In our work, we propose an enhanced model encoding that scales well in memory, can be easily serialized, utilizes low bandwidth and can also work with a wide variety of genetic algorithms. An individual from the population (a NN) can be represented as a list of tuples (Fig. 2), the first tuple contains the specie identifier

| Hyperparameter | Value |
|---|---|
| Population | 200 |
| Mutation Power | 0.002 |
| No of elites | 10% |
| $\Psi$ | 0.75 |
| Species | 5 |

*Fig. 3: Hyperparameters used to train Sp-GA*

| | DQN | A3C | Sp-GA |
|---|---|---|---|
| No. of workers | 1 CPU | 8 workers - 1 CPU each | 10 workers - 4 CPU each |
| Frames | 5M | 5M | 5M |
| Training time | $\sim$ 9hr | $\sim$ 3hr45min | $\sim$ 55min |
| frostbite | 240 | 180 | **270** |
| spaceinvader | 585 | 555 | **1020** |

*Fig. 4: Maximum episodic reward for DQN,A3C and elite's performance for Sp-GA after training on $\sim$ 5 million frames on 2 Atari 2600 games.*

(class name), initialisation seed and the subsequent tuples represent the mutation power, seeds utilized for evolutionary operations at each generation. Given this encoded representation, a worker can decode the model by iterating through this list of seeds and applying iterative mutations.

Crossover of two neural networks is a challenging task to implement in a distributed manner. Previous work [7, 9] only rely on mutations when evolving the neural network. The chromosomal crossover and recombination of genes in cells motivated us to apply the same approach to the encoded representation of the networks. A crossover point is sampled and a new encoding is formed by merging the two encodings along the sampled point. The resulting encoding represents a valid individual which contains features from each parent (Fig. 2).

## 3 Results

The performance of the proposed algorithm (*Sp-GA*) was evaluated on Atari 2600 games. Due to limited computational resources, we compare the results from 2 games, i.e. *space-invaders* and *frostbite* with reference implementations [3] of *DQN* and *A3C*. The tuned hyper-parameters were obtained from the benchmarked results [4]. The following steps in our algorithm are identical to *DQN* and *A3C*: 1) data pre-processing 2) network architecture 3) 30 random, initial no-op operations The hyper-parameters used for training are summarized in Fig. 4 and the initial distribution and kinds of species are shown in Fig. 1. Each episode (*DQN* and *A3C*), generation *Sp-GA*) was capped at maximum of 10k frames. The maximum episodic reward (*DQN* and *A3C*) and elite individual's reward (*Sp-GA*) after training for $\sim$ 5 million frames on a Kubernetes cluster is reported in Fig. 4. The elite model for frostbite belongs to the species 'xavier_normal' and for space-invader it belongs to the species 'kaiming_uniform'. *Sp-GA* outperforms *DQN* and *A3C* on both games and is at least 4 times fast. However, all these comparisons are preliminary and need further investigation. The key takeaway is that *Sp-GA* is a comparable competitor to *DQN* and *A3C* and is significantly scalable.

## 4 Discussion

In this work we presented a distributed species-based genetic algorithm as a scalable alternative to RL algorithms like *DQN* and *A3C*. A novel framework to apply crossover in the encoding space is also proposed. Our results show comparable performance and significant speedup when compared to Markov Decision Process (*MDP*) based approaches. The results in our work are limited to 2 games and training on $\sim$ 5 million frames and require further verification. We have also demonstrated how different variants of genetic algorithms can be scaled and

[3] https://docs.ray.io/en/master/rllib.html
[4] https://github.com/ray-project/rl-experiments

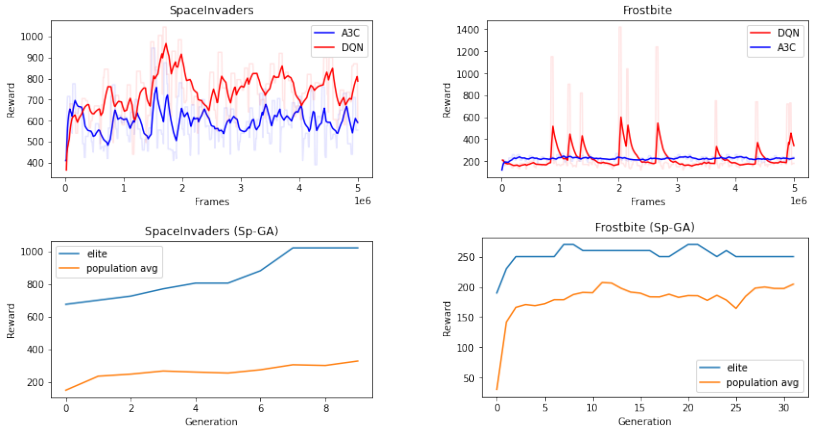


*Fig. 5: Performance of DQN, A3C (max score over an episode), Sp-GA(top elite's score) with increasing epochs on Atari 2600 games.*

quickly solve RL problems with a large state space. Future efforts will be devoted towards training our algorithm on a wider set of games for more training frames and tuning the hyper-parameters for each use case. It is also interesting to see how this algorithm would perform in RL problems from other domains such as from robotics and telecommunication. Overall it is noteworthy to see a simple algorithm performing surprisingly well paving the way to novel variants such as a hybrid approach combining *Sp-GA* with a *MDP*-based algorithm.

## References

1. Kalashnikov, D., Irpan, A., Pastor, P., Ibarz, J., Herzog, A., Jang, E., Quillen, D., Holly, E., Kalakrishnan, M., Vanhoucke, V., Levine, S.: Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation (2018)
2. Mania, H., Guy, A., Recht, B.: Simple random search provides a competitive approach to reinforcement learning (2018)
3. Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T.P., Harley, T., Silver, D., Kavukcuoglu, K.: Asynchronous methods for deep reinforcement learning (2016)
4. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M.: Playing atari with deep reinforcement learning (2013)
5. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al.: Human-level control through deep reinforcement learning. nature **518**(7540), 529–533 (2015)
6. Nikou, A., Mujumdar, A., Orlic, M., Feljan, A.V.: Symbolic reinforcement larning for safe ran control. Internaltinal Conference of Autonomous Agents and Multi Agent systems (AAMAS) (to appear) (2021)
7. Salimans, T., Ho, J., Chen, X., Sidor, S., Sutskever, I.: Evolution strategies as a scalable alternative to reinforcement learning (2017)
8. Stanley, K.O., Miikkulainen, R.: Evolving neural networks through augmenting topologies. Evol. Comput. **10**(2), 99–127 (Jun 2002)
9. Such, F.P., Madhavan, V., Conti, E., Lehman, J., Stanley, K.O., Clune, J.: Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning (2018)
10. Zheng, G., Zhang, F., Zheng, Z., Xiang, Y., Yuan, N., Xie, X., Li, Z.: Drn: A deep reinforcement learning framework for news recommendation. pp. 167–176 (04 2018)