ICICT 2024

International Congress & Excellence Awards

9th International Congress on Information and Communication Technology, London, UK

Design of cluster-computing architecture to improve training speed of the Neuroevolution algorithm

Iaroslav Omelianenko
Institute of Software Systems, National Academy of Science of Ukraine

19-22 February, 2024

Introduction

Genetic algorithms are a promising field of machine learning, particularly suitable for tasks related to optimization and control, reinforcement learning, artificial life simulation, coevolution of swarm agents, and so on.

One of the most popular and powerful among them is a **Neuroevolution of Augmenting Topologies** (NEAT). NEAT algorithm and its extensions allow us to apply creative forces of natural evolution to design compact and energy efficient topologies of the Artificial Neural Networks (ANN). Controllers based on such ANNs can then be used for inference in environments with very limited resources, such as: robots, UAVs, networking edge devices, IoT devices, etc.

In this presentation, I'll briefly describe the fundamentals of the NEAT algorithm. After that, the proposed architecture design of cluster-computing using the Ray framework will be presented.

NEAT Fundamentals

The most important feature of the NEAT algorithm, that defines its potential, is an ability to evolve the topology of Artificial Neural Networks during the learning process. The importance of gradual augmentation of network topology became obvious if we consider how it is implemented by NEAT. When exploring search space of solution, NEAT algorithm starts with simplest basic network topology that comprises only input and output nodes of the solver ANN. After that, with each epoch of evolution, the topology of solvers become more complex and the most complex multidimensional topologies are evaluated only at the final stages of evolutionary process. Furthermore, the most suitable topological structures found during evolution are preserved by NEAT algorithm through generations due to its inherent features.

Such approach of gradual complexification of solver's topology considerably reduces the solution search space during the training. Wide solution's search space is a huge drawback of other evolutionary algorithms which effectively addressed by NEAT due to its key design features, which we consider next.

NEAT Key Features

- **Topology augmentation through gradual complexification.** Learning process starts with basic, simple topology and gradually increases its complexity based on results of analysis of solution search space. At the same time, NEAT has ability to preserve minimal found topology that is able to generate ultimate solution.

- **Speciation.** It separates population into evolutionary niches using genetic distance between organisms (similarity of genomes). Its main goal is to reduce negative adaptation pressure caused by increased complexity of novel organism when new node or connection added to the genome of the offspring.

- **Mutation.** It plays an important role in keeping genetic diversity of the population during evolution and prevents objective function from stalling at local minima when the chromosomes of organisms in the population become too similar. Mutation operator changes one or more genes in chromosomes according to the mutation probability defined by experimenter.

NEAT Key Features

- **Innovation numbers.** It is the most prominent feature of the NEAT algorithm that allows to solve issue with overlapping parts of genomes of ANN solvers. The overlapping genome parts produce similar ANN topologies that estimate the same function, but were embedded in different genome structures during reproductive crossover. With innovation numbers, we can easily differentiate them without complex topological analysis.

- **Crossover (recombination of genomes).** Allows for two or more champion organisms within specific population niche to mate and produce offspring that inherit important characteristics of both parents. It is during this process that the mentioned above innovation numbers allow us to identify overlapping (matching), excess or disjoint genes.

NEAT Drawbacks

- Genome of each organism in the population directly encodes a certain topology of the ANN solver. As a result, the population size and the size of encoded ANN are limiting factors. With a larger population, we need to have more computing resources to evaluate the evolutionary process. Thus, we can only work effectively with relatively small ANN topologies and moderate populations.
- Chosen objective function applied to calculate the fitness of organisms in a population works well for simple tasks, but often collapses into local minima traps for more complex tasks.
- Slow training associated with a large number of calculations at the stage of fitness evaluation of each organism in the population.
- First two drawbacks can be addressed using extensions of the NEAT algorithm and the latter is the subject of interest in this presentation.

NEAT distributed training with Ray framework

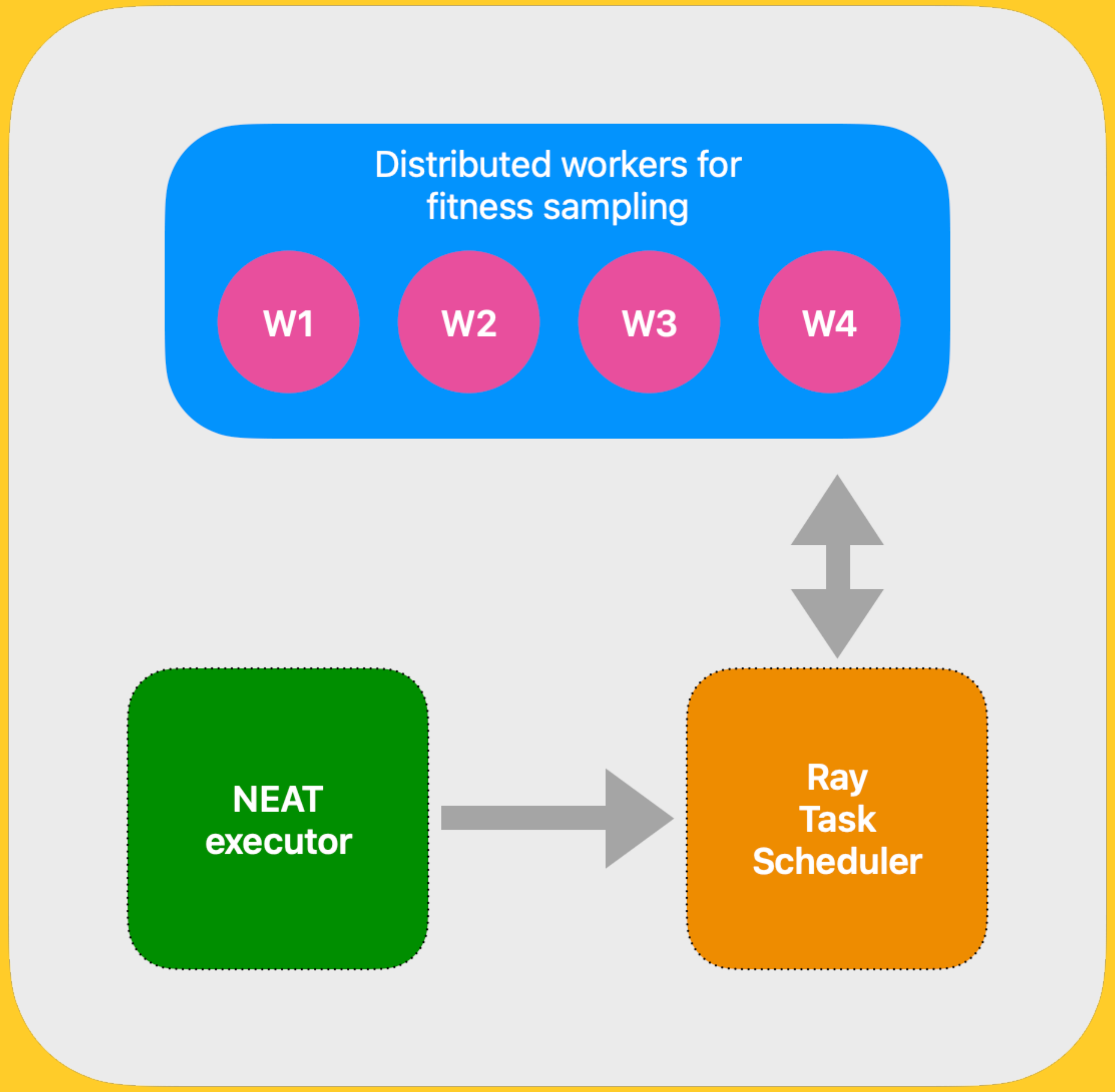
- Ray is a general-purpose distributed cluster-computing framework developed within University of California, Berkley to address the ever-growing need for distributed parallel computing for Reinforcement Learning tasks. The main goal of the Ray framework was to develop a framework capable of processing heterogenous task in a massively distributed manner to handle over a million scheduled tasks per second with milliseconds-level latencies. Heterogeneity of tasks considered both in the terms of execution time (e.g. a task takes milliseconds or hours) and resource usage (e.g. tasks can be executed by GPUs, CPUs, or TPUs).

- Our proposal is to use the Ray framework to handle the most time-consuming task in the neuroevolution algorithm - simulation of the objective function execution to estimate the fitness of each organism in the population at each epoch of evolution.

NEAT distributed training with Ray framework

The following architecture is proposed to support distributed training:

- **NEAT executor** - main process executing evolutionary training using NEAT GoLang library.
- **Ray Task Scheduler** - Ray driver application scheduling remote tasks executions and exposing RPC API interface to the NEAT executor.
- **Distributed workers** - cluster with Ray stateless workers to perform fitness estimation simulation for each organism in the population.



NEAT distributed training with Ray framework

- The NEAT algorithm implements a direct genome encoding scheme, which allows generating a solver ANN directly from an organism's genome. At the same time, the implemented genome encoding scheme has a very compact textual representation that can be easily encapsulated in standard POST HTTP requests.
- Using this coding scheme, we can easily encode the genomes of a single organism or all organisms in a population together. As a result, at the end of each evolutionary epoch, we can send the encoded genomes of all organisms in the population for evaluation by distributed Ray workers in just one network call, which greatly reduces the latency of this operation.
- After receiving the encoded genomes of all organisms in the population, the *Ray Task Scheduler* reads the encoding of each individual genome and schedules tasks for remote fitness evaluation by *stateless cluster workers*.

NEAT distributed training with Ray framework

- Each remote worker in the Ray cluster receives the encoded genome of the organism as input, decodes it into an ANN solver, and applies it to solve the simulated objective function. The estimated fitness of the organism and a flag indicating whether the objective function has been solved are then returned to the *Ray Task Scheduler.*

- Ultimately, the *Ray Task Scheduler* collects data from all remote workers and returns it to the *NEAT Executor* in one batch. Using the obtained fitness values of each organism in the population, the *NEAT Executor* creates a new population of organisms for the next epoch of evolution, or terminates evolution if a solution has already been found.

- For a detailed description of the proposed algorithms, please refer to my conference paper.

Conclusions

- Despite all the powerful features inherent in evolutionary algorithms, they are still not widely used due to their slow training speed. We demonstrate how this obstacle can be mitigated by using a state-of-the-art cluster computing framework that allows the most time-consuming steps of evolutionary learning to be transferred to an almost limitless cloud infrastructure.

- Using the proposed design, it is possible to develop industrial micro-service systems with trained NEAT models without any additional model tuning. This can significantly reduce the maintenance time of the ML operation cycle. In addition, this approach allows implementing self-supervised learning systems that continuously train deployed models.

- The source code of the implementation of the NEAT library using the GO programming language is available in the GitHub repository

- **https://github.com/yaricom/goNEAT**

Thank you!