Decentralized Hybrid-Model Evolutionary Computation using Blockchain and an Optimization Proof Of Work

Harvey D. Bastidas C.
Email: harveybc@ingeni-us.com
GitHub: https://github.com/harveybc
Master on Engineering Student at Pontificia Universidad Javeriana – Cali

Introduction

- The *Evolutionary Computation (EC)* techniques such as genetic algorithms or neuroevolution are optimization methods that use a population of candidate solutions.
- Candidate solutions evolve in a search space in a way inspired by biological evolution principles like competition, selection or reproduction.
- There are several architectural models for implementing EC techniques in distributed processing architectures (dEC), the models with better fault tolerance and lower communicational cost are the hybrid models based on the so-called island model.

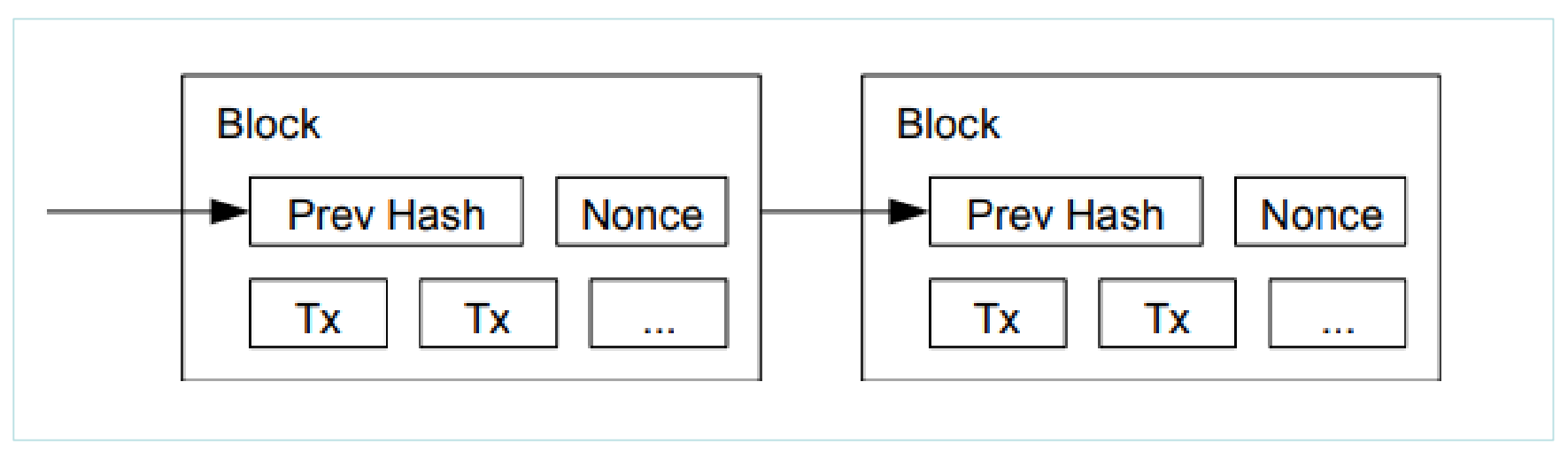

Figure 1. Blockchain Diagram - Source: "Bitcoin: A Peer-to-Peer Electronic Cash System", Satoshi Nakamoto, 2008

- This project uses a blockchain (Figure 1) with *an optimization proof-of-work* (OPoW) for block-time control in a decentralized network to implement traceability of optimization states in dEC with hybrid models.
- OPoW uses for optimization the computational capacity that is used for hash-based proof-of-work generation in other blockchain based decentralized networks like Bitcoin and allows consensus on the order of the operations made in the network (devices may have misconfigured clocks) but with a security tradeoff.

Design

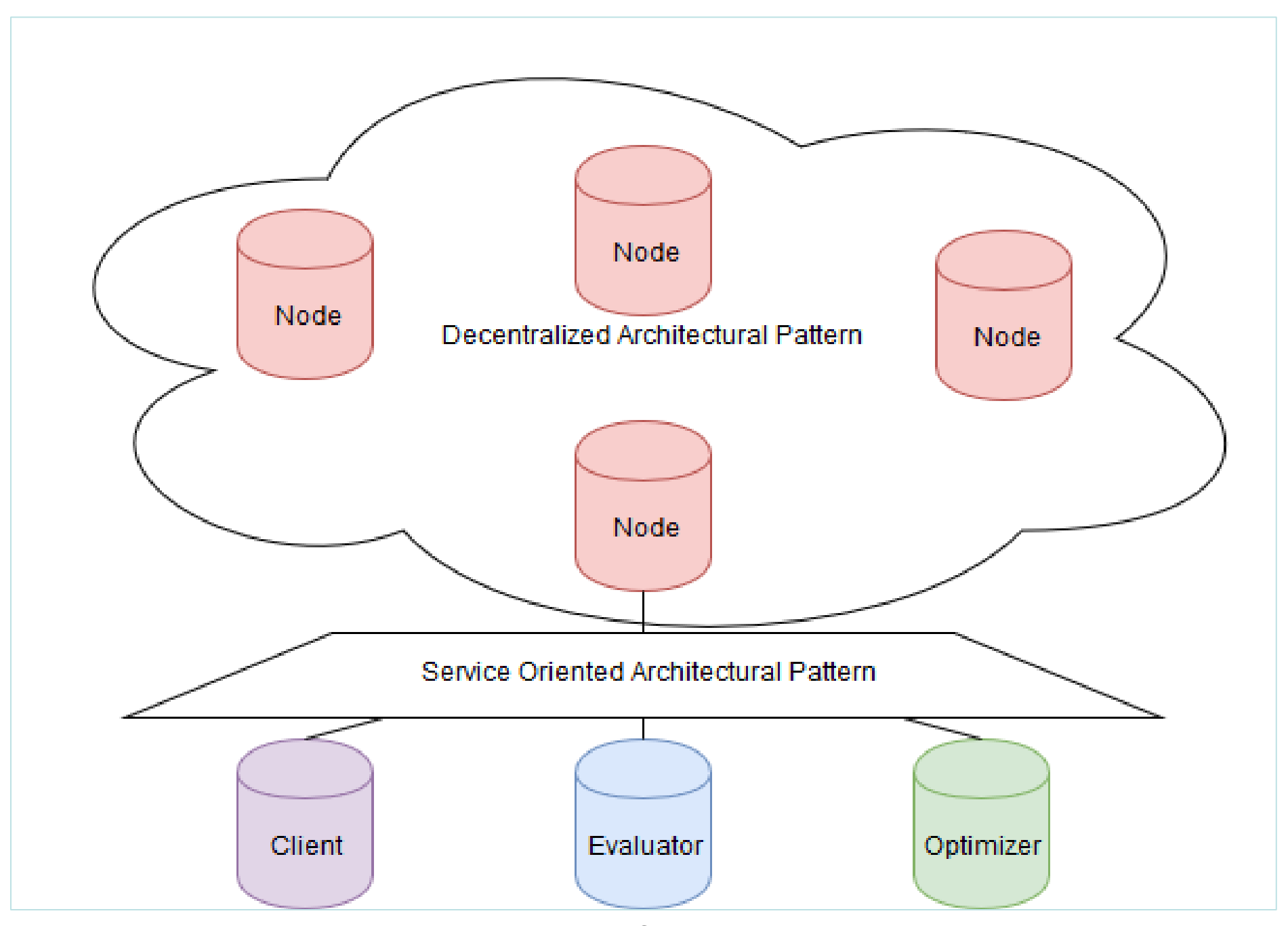
Node
Node
Decentralized Architectural Pattern
Node
Node
Service Oriented Architectural Pattern
Client
Evaluator
Optimizer

Figure 3. Platform Architecture

The nodes are continuously receiving requests from clients, evaluators and optimizers and flooding them to other nodes. (Figure 3).

When an optimizer finds new optimal parameters, it sends them to a node and after verification, it creates a new block containing all the information received by the node from the last block.

Client: Sends inputs to be remotely evaluated by model with optimized parameters.

Evaluator: Produces an output given a mathematical model, its parameters and some input.

Optimizer: Implements an EA for optimizing model's parameters must include an HTTP request between iterations for optimization state synchronization. Generates the OPoW.

Node: Orchestrates the communications between devices and other nodes, implements traceability of actions performed by devices in the network, have authentication and authorization for controlling the access to the network, perform controlled flooding and they use MVC pattern. (Figure 4).

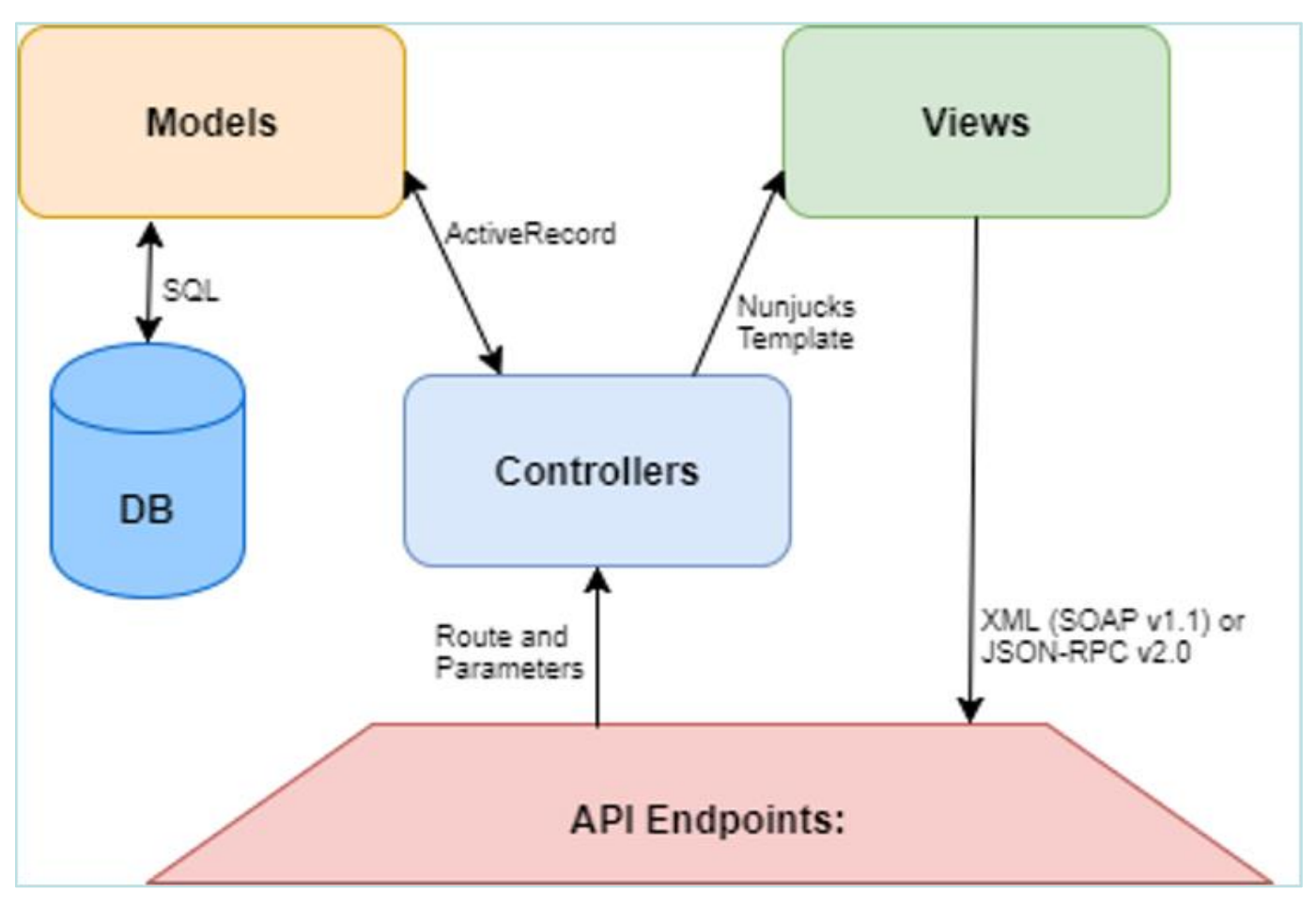

Figure 4. Node Architecture

Optimization Proof-Of-Work

- Bitcoin uses the number of leading zero bits of a PoW (hash of block contents + nonce) as the difficulty of the block and it serves to control the block-time, targeting an average number of blocks per hour.
- In this project the difficulty of the block is the increment in performance of the parameters of a mathematical model being optimized.
- As with the PoW used in Bitcoin, the OPoW is easy to validate from optimizers in the network but hard to produce, since it requires a large computational capacity to increase the performance of the mathematical model being optimized.
- Since the optimization increments produced by EC are of variable magnitude (Figure 2), the method to control the block-time in Bitcoin based on moving averages may not work with OPoW.

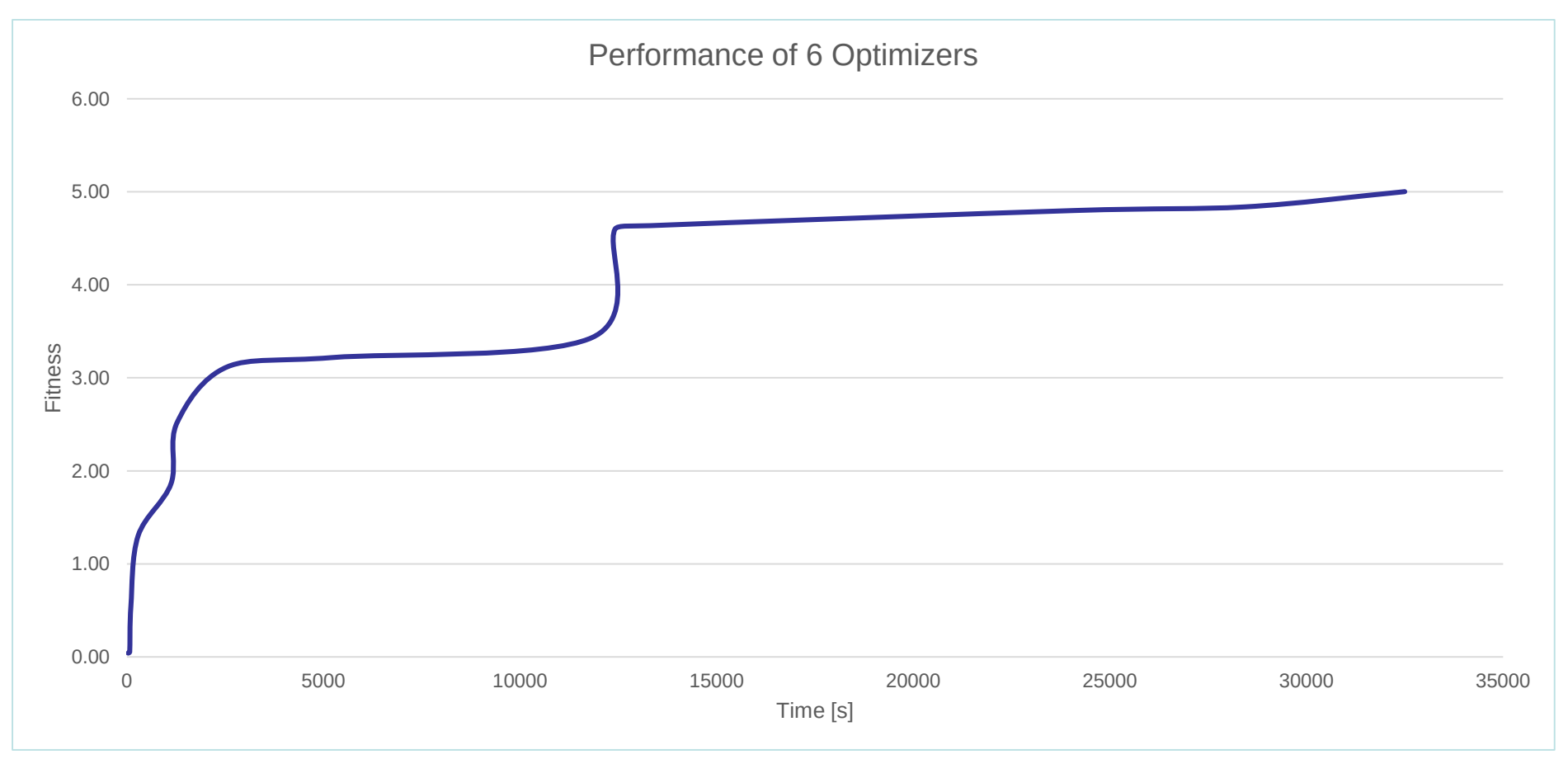

Figure 2. Performance of a reinforcement learning experiment

- Depending on the application, there are two options for producing blocks using the performance increments as difficulty:
  - To use a variable block-time and generate a block in every increment of performance.
  - To use a control system to find a threshold that the performance must reach in order to target an average number of blocks per hour.

Implementation

- The OPoW is implemented in a software platform for extending existing Evolutionary Algorithms (EA) to use a decentralized architecture using a REST API (Node.js) between iterations to synchronize states with the network.
- The network uses TTL and Sequence Number Controlled Flooding (SNCF).
- The consensus on the order of the operations in a block is that the correct order is the one in the accounting transactions of optimizer that finds a new optimization proof-of-work, other nodes sync with it.
- Active Record was used, so the platform is database-engine portable.

Stage	Number of Optimizers	Avg. Performance
1	6	3.28
2	1	0.76

Table 1. Results of a scalability test executed during 3 hours

An application for reinforcement learning in the Forex trading automation domain was made for empirically validating the platform, it uses a Forex account simulator OpenAI Gym environment and optimizes the weights and topology of a neural network that controls the actions of an agent in the simulation using Neuroevolution (NEAT-Python).

Conclusions

- It is possible to use an optimization-based proof-of-work (OPoW) to control the block-time of a blockchain because a new optimized parameter is hard to produce but easy to verify by optimizers.
- Existing evolutionary algorithms can be extended to a decentralized architecture for scalability, fault tolerance and invalid result rejection using OPoW.
- The tradeoff using of a useful proof of work is that the OPoW is not a function of the block contents but this tradeoff probably can be mitigated using additional security mechanisms.