How to evolve a neuron

W. Garrett Michener
College of Charleston
Charleston, South Carolina
MitchenerG@cofc.edu

ABSTRACT

A natural neuron can function as a nonlinear down-sampler, or in the extreme, as a coincidence detector. I will describe artificial life experiments in which agents capable of general computation evolve an artificial reaction network capable of performing coincidence detection. They also develop spike-timing-dependent plasticity, thought to be an important mechanism for learning.

CCS CONCEPTS

• **Applied computing** → **Systems biology**; **Population genetics**.

KEYWORDS

artificial life, evolution, neuroscience, systems biology

ACM Reference Format:
W. Garrett Michener. 2021. How to evolve a neuron. 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/10.1145/3449726.3459547

1 INTRODUCTION

Neurons are living cells that perform computations on the millisecond time scale using discrete electrical pulses called *spikes* or *action potentials.* A neuron can serve as a non-linear low-pass filter or coincidence detector. Such a neuron spikes once only when several pulses of excitatory neurotransmitter arrive in a very short time.

Furthermore, a neuron can exhibit *spike-timing dependent plasticity* [2, 4]. If it repeatedly spikes just after receiving excitatory input, the synapse temporarily becomes more receptive to excitation. This is thought to be a key step in learning.

In an attempt to understand something about how such a biochemical computational unit could evolve, let us consider an *artificial life* (*a-life*) simulation, and focus on the challenge of how to specify a fitness function that leads agents to evolve neuron-like behavior. Specifically, they must perform coincidence detection and exhibit something akin to spike-timing dependent plasticity.

The desired results can be achieved by presenting an agent with a long, fixed list of inputs, each of which is a time series of spikes arriving at various times. Points are awarded for firing shortly after a coincidence in the input. Points must also be awarded for *not* firing when the input contains widely separated spikes. Points are also awarded for firing when the input contains two spikes separated by an intermediate duration *only* when preceded by several spike-inducing coincidences. Under these conditions, agents evolve into computational units that mimic the coincidence-detection ability and plasticity of living neurons.

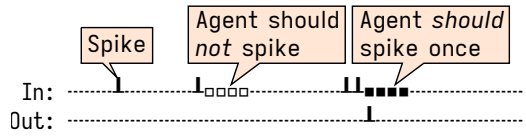

Figure 1: Example input spike train and desired output.

2 SIMULATION FRAMEWORK IN BRIEF

The experiment takes place in an a-life simulation, similar to that described in [3]. The simulation is designed to mimic certain aspects of living organisms so that it may serve as a tool for understanding how species discover and refine computational mechanisms.

Each agent consists of a single cell whose mutable state is an array of activation levels $A[]$. The entry $A[k]$ represents how many units of abstract biochemical k are present in the cell. Initially, A is all 0s. Elements 3 and 41 are designated for input and output.

Each agent has a bit-string genome, which specifies an *artificial regulatory network* (ARN) similar to the *gene regulatory network* (GRN) of a living cell. Blocks of bits serve as genes and are decoded into tuples of integers $(p_{\text{sw}}, \theta, p_{\text{up}}, p_{\text{dn}})$, each of which is interpreted as an instruction:

> If $A[p_{\text{sw}}] \geq \theta$ then
> add 1 to $A[p_{\text{up}}]$
> and subtract 1 from $A[p_{\text{dn}}]$.

This instruction format is an abstraction of the construction and action of a protein. The switch pattern p_{sw} is analogous to a promoter in DNA. The threshold θ controls how much of k must be present to trigger transcription. The integers p_{up} and p_{dn} represent the action of the protein produced from this gene, as it creates one unit of p_{up} and destroys one unit of p_{dn}.

An agent's *program* is the set of all instructions from all chromosomes. Program execution proceeds in discrete time steps. At the beginning of a time step, external input to the cell is handled by adding 8 to $A[3]$. Then all instructions in the program are evaluated simultaneously, in analogy with proteins in solution. Finally, if $A[41] \geq 2$, the cell produces an output spike, and 2 is subtracted from $A[41]$ to represent expended energy. Otherwise, the agent produces no output for this time step.

The selection-mutation process advances in discrete generations as follows. Each agent has a numerical fitness rating. Living agents are sorted by rating, and the highest-scoring 400 remain alive. Another 600 are generated by choosing pairs of parents, and forming

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

ACM ISBN 978-1-4503-8351-6/21/07.
https://doi.org/10.1145/3449726.3459547

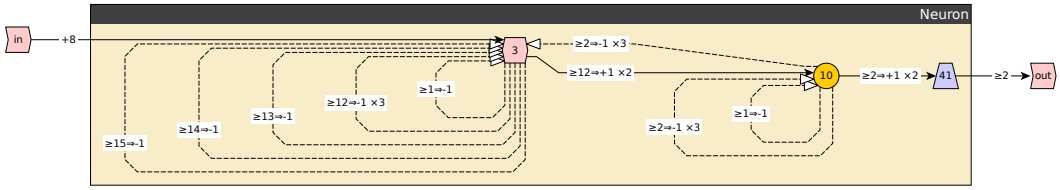

Figure 2: Example perfect solution

a child agent. Each parent is selected by a *tournament:* Two candidates are picked uniformly at random from the population, and with probability 0.6 the higher-scoring one is chosen. Once two parents are selected, the child is generated by a process modeled on diploid organisms. For each parent, each pair of bit-string chromosomes is aligned, and a single point crossover is performed to produce a single chromosome, resulting in a haploid gamete. Mutations are performed at random, including bit flips, gene deletions, and gene duplications. The chromosomes from each gamete are combined to construct the child's genome.

3 THE RATING FUNCTION IN BRIEF

An agent's fitness rating is the sum of points scored from a list of *tasklets*. Each tasklet specifies a spike train in the form of a finite sequence of bits $(b_0, b_1, \ldots, b_n)$ with $b_j = 1$ meaning that an excitatory input spike arrives at time step j. An example of an input spike train and desired output is shown in fig. 1. The first two input spikes are widely separated and should not trigger an output spike, but the following two spikes in consecutive time steps should.

The scoring system is designed to encourage the population not to remain stuck at a local maximum where no output spikes are generated. Assuming an output spike indicates recognition of danger, the scoring system prefers excess spikes (false positives) to never spiking at all. For a biological analogy, Mauthner neurons in fish produce a single spike that triggers an escape reflex [1]. Thus, many points are given for spiking during a short interval after two input spikes arrive in quick succession, and some are given for *not* spiking when the input spikes are widely separated. A few points are lost for excess spikes.

Starting with a few short tasklets as in fig. 1, a set of longer tasklets is produced by concatenating copies of those foundational tasklets. These penalize mechanisms that can handle one or two pairs of input spikes, but do not properly reset themselves, and give incorrect results when presented with long spike trains. Instead, mechanisms that can react correctly to any number of stimuli are favored. The training set so formed is large enough that agents are unable to memorize it.

A second set of tasklets adds the complication of spike-timing dependent plasticity. There are *strengthening* tasklets, in which many pairs of input spikes separated by short intervals arrive in a relatively short period of time, followed by a final pair of spikes separated by just a few too many time steps to trigger a spike if they occurred in isolation. The correct response here is to produce an output spike after that last pair, analogous to a strengthened synapse. Conversely, there are *weakening* tasklets, in which a very long interval with only a few, very widely separated input spikes is followed by a final pair of spikes separated by an intermediate time interval. The correct response is not to produce an output spike after that last pair, analogous to a weakened synapse.

4 RESULTS & DISCUSSION

For each of two variants of the scoring system, 100 samples were run for 20,000 generations. Each generation takes about 15 seconds, so some of these samples took several days to complete. The two sets included 9 and 13 samples in which a perfectly scoring solution was discovered and swept through the population. Other samples were nearly perfect, but had trouble with the weakening task, or produced excess output spikes.

An example of a perfect solution is shown in fig. 2. The nodes labeled 3, 10, and 41 inside the box represent $A[3]$, $A[10]$, and $A[41]$. It is pre-specified that input spikes cause 8 to be added $A[3]$, and that when $A[41] \geq 2$, an output spike is generated and $A[41]$ is reduced by 2. These features are represented by arrows from the *in* node and to the *out* node. Each genetically specified instruction $A[p_{\text{sw}}] \geq \theta \Rightarrow A[p_{\text{up}}]{+}{=}1, A[p_{\text{dn}}]{-}{=}1$ is represented by two arrows, a solid one from p_{sw} to p_{up} and a dotted one from p_{sw} to p_{dn}, each labeled $\geq \theta$. A label with $\times n$ indicates n copies of that arrow. Only relevant arrows are shown. The many inhibitory (dotted) arrows from 3 to itself allow output spikes only when two closely spaced input spikes arrive, followed by a reset of $A[3]$. After many input spikes arrive in a short time, $A[3]$ temporarily holds a greater value, so output spikes are produced more readily. The instructions involving $A[10]$ pass activity from $A[3]$ to $A[41]$ but with a one time step delay, preventing output spikes in the same time step in which the second input spike of a pair arrives. The input synapse is generally in a weakened state, and only becomes strengthened after high levels of input, analogous to how NMDA receptors work [2]. This suggests that the rating function is biologically reasonable.

REFERENCES

[1] Henri Korn and Donald S. Faber. 2005. The Mauthner Cell Half a Century Later: A Neurobiological Model for Decision-Making? *Neuron* 47, 1 (July 2005), 13–28. https://doi.org/10.1016/j.neuron.2005.05.019

[2] Fei Li and Joe Z. Tsien. 2009. Memory and the NMDA Receptors. *The New England journal of medicine* 361, 3 (July 2009), 302–303. https://doi.org/10.1056/NEJMcibr0902052

[3] W. Garrett Mitchener. 2014. Evolution of Communication Protocols Using an Artificial Regulatory Network. *Artificial Life* 20, 4 (Aug. 2014), 491–530. https://doi.org/10.1162/ARTL_a_00146

[4] J. David Sweatt. 2003. *Mechanisms of Memory.* Academic Press, San Diego, Calif.