# Impact of Selection Strength on the Evolution of Regulatory Networks

W. Garrett Mitchener
College of Charleston Mathematics Department
MitchenerG@cofc.edu
http://mitchenerg.people.cofc.edu

## Introduction

The problem that led to this project was a result from Plotkin and Nowak [2000], in which an inequality from information theory puts an upper bound on the fitness of a communication system with a one-to-one mapping of signal to meaning. If there are enough topics of conversation, switching to a sequential coding system (string-of-signals to one meaning) yields a fitness advantage that overcomes that bound. See also Zuidema and de Boer [2009]. Evolutionary game theory can explain the survival and spread of such a system, but not its *origin* from organisms that do not have it.

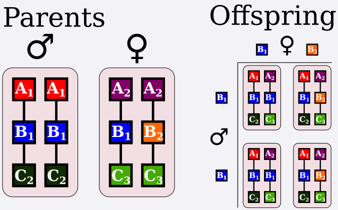
The basis of biological computation is the reaction or regulatory network. How are such networks discovered by selection-mutation processes?

The Utrecht Machine (UM) is a discrete abstraction of a gene regulatory network. For this project, an evolutionary simulation is used to discover UM-based agents that solve a data encoding problem. Details of the selection process have a significant impact on population dynamics: Weak selection leads to shorter overall times to first perfect solution, shorter genomes, fewer repairs to auxiliary mechanisms after an improvement to the main mechanism, and fewer atypical synaptic codes, and is more likely to discover improved partial solutions by breeding outliers. The times at which key innovations appear and spread depends on the population state in surprising ways.

FYI: I call it the Utrecht Machine because the conference where I first presented it was EvoLang 2010 in Utrecht.

# Traditional population genetics

Abstracts away the gene regulatory network. Models the genotype as a list of alleles, mapped to a numerical fitness for the phenotype. Markovian & memoryless.



Fitness table

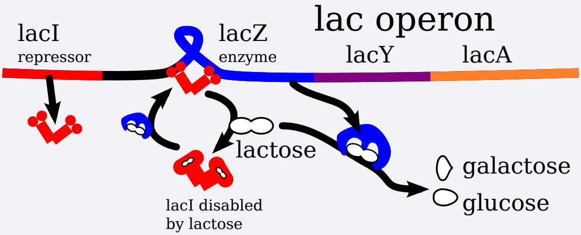
$F[\ldots B_1B_1\ldots]=4.5$
$F[\ldots B_1B_2\ldots]=4.8$
…

Mutation table

$P[B_1 \rightsquigarrow B_2]=0.01$
…

# Systems biology

Genes produce functional products (enzymes, etc.) but also regulatory products that promote or repress other genes. Complex networks of interacting molecules.



## Population game dynamics: replicator-mutator

$$(1) \qquad \dot{x}_j = \sum_{k=1}^{K} x_k F_k Q_{kj} - x_j \phi$$

- $k = 1, 2, \ldots K$: types of individuals
- $x_k$ = fraction of population of type $k$
- $B_{kj}$ = payoff for type $k$ interacting with type $j$
- $F = Bx$: vector of average payoffs by type
- $\phi = x^T F$: overall average payoff
- $Q_{kj}$ = probability offspring of type $k$ is type $j$

## Population genetics: Moran-Mendel

The population is finite. Dynamics form a Markov chain: Kill one agent uniformly, replace by cloning another chosen in proportion to fitness.

- $A_1, A_2, \ldots; B_1, B_2 \ldots$: alleles at loci $A$ & $B$
- $P_{hi}, \ldots$: prob of mutation from $A_h$ to $A_i$ …
- $F_{j_1 j_2 k_1 k_2}$ = fitness of genotype $A_{j_1} A_{j_2} B_{k_1} B_{k_2}$

# How to combine selection-mutation population dynamics with relevant molecular details?

- Genome = inherited record of instructions:
  Should be a string of letters, subject to mutation and recombination like DNA

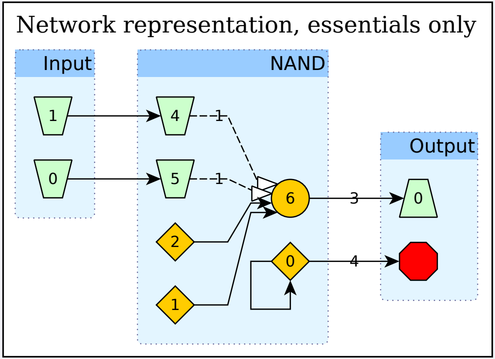
- Virtual machine = how to build the phenotype:
  Should be an artificial regulatory network

The *Utrecht Machine* or *UM* is designed to meet those requirements. The state of the UM is a table mapping *patterns* $p$ to integer *activation levels* $A_p$. Each pattern is metaphorically a protein that binds to a promoter or inhibitor sequence on DNA. The activation level of a pattern is how many units of it are present.

Each UM reaction instruction $[A_s \geqslant \theta \implies \text{inc}\, A_p, \text{dec}\, A_q]$ is parameterized by four integers: a *switch pattern* $s$, a *threshold* $\theta$, a pattern $p$ to activate called *p-up*, and a pattern $q$ to inhibit called *p-down*. An instruction is active if the activation level of its switch pattern $A_s$ satisfies $A_s \geqslant \theta$. An active instruction adds 1 to the activation of its p-up pattern and subtracts 1 from the activation of its p-down pattern. All active instructions are performed simultaneously.

A bit of input is provided by increasing the activation of a particular pattern during each time step the input bit is set.

# Example: UM for binary NAND



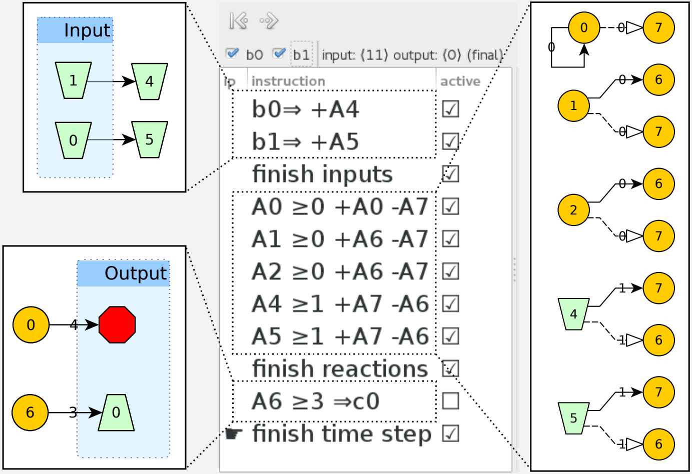
| $b_0$ | $b_1$ | $\neg(b_0 \wedge b_1)$ |
|---|---|---|
| F | F | T |
| F | T | T |
| T | F | T |
| T | T | F |

$$A_0 \geqslant 0 \implies \text{inc}\, A_0, \text{dec}\, A_7$$
$$A_1 \geqslant 0 \implies \text{inc}\, A_6, \text{dec}\, A_7$$
$$A_2 \geqslant 0 \implies \text{inc}\, A_6, \text{dec}\, A_7$$
$$A_4 \geqslant 1 \implies \text{inc}\, A_7, \text{dec}\, A_6$$
$$A_5 \geqslant 1 \implies \text{inc}\, A_7, \text{dec}\, A_6$$

Two bits of input supplied through patterns 4 and 5, one bit of output read as $\mathbf{1}(A_6 \geqslant 3)$. Stop when $A_0 \geqslant 4$ or after a maximum of 10 time steps.

UM reaction instructions are just tuples of 4 integers, and their effects are independent of the order in which they are listed. They can therefore be encoded as a binary genome and subject to biologically realistic mutation and recombination.

# Building the network representation



☑ b0 ☑ b1 input: ⟨11⟩ output: ⟨0⟩ (final)

| ip | instruction | active |
|---|---|---|
| | b0⇒ +A4 | ☑ |
| | b1⇒ +A5 | ☑ |
| | finish inputs | ☑ |
| | A0 ≥0 +A0 -A7 | ☑ |
| | A1 ≥0 +A6 -A7 | ☑ |
| | A2 ≥0 +A6 -A7 | ☑ |
| | A4 ≥1 +A7 -A6 | ☑ |
| | A5 ≥1 +A7 -A6 | ☑ |
| | finish reactions | ☑ |
| | A6 ≥3 ⇒c0 | □ |
| ☛ | finish time step | ☑ |

One link for each. . .

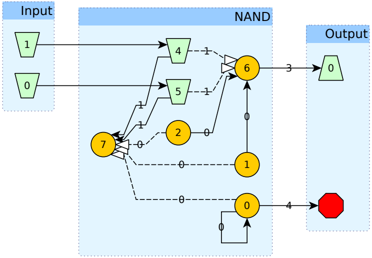
- input bit
- output bit
- stop signal

Each instruction generates two links from the switch pattern node: one for p-up, one for p-down.

Links for p-up, input, & output are solid with black arrowhead. Links for p-down are dashed with white arrowhead. Pattern nodes are circular for most patterns, keystones for those involved with input and output. Numbers on links indicate a threshold. A link with no number is activated by an input bit instead of a threshold.

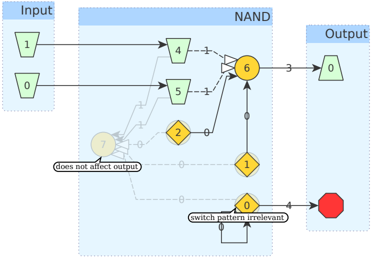
# Reduction from full to essential network

Network representation, all details



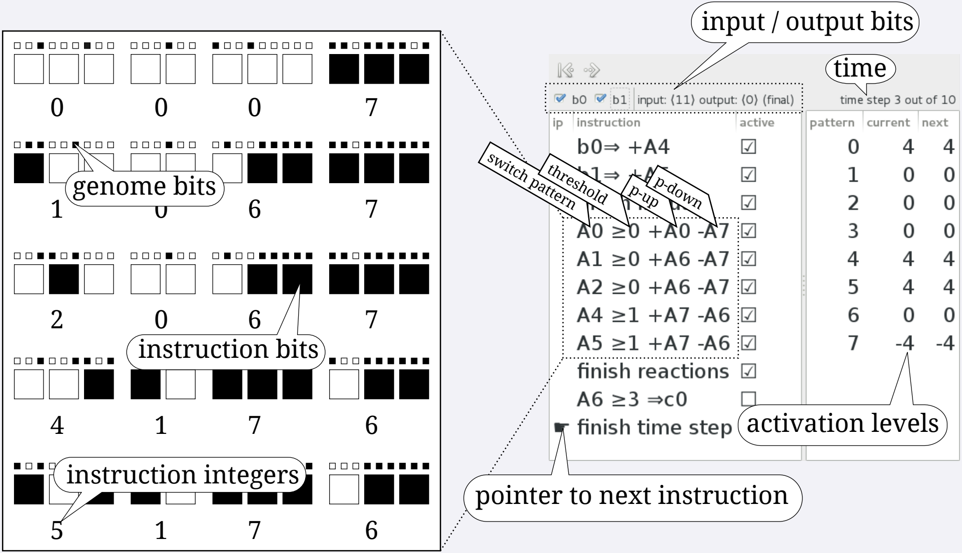
Merge pattern nodes, layout so that information mostly flows left to right.

Trim non-essential details...



Remove pattern nodes that don’t matter, like 7, which is used as a no-op or bit bucket in this machine. Turn nodes where the pattern doesn’t really matter into diamonds. This happens to patterns 0, 1 and 2 because all instructions with these for the switch pattern happen to always be active. Then neaten up the layout.

# Example: NAND with genome and step-by-step interface



|   |   |   |   |
|---|---|---|---|
| 0 | 0 | 0 | 7 |
| 1 | 0 | 6 | 7 |
| 2 | 0 | 6 | 7 |
| 4 | 1 | 7 | 6 |
| 5 | 1 | 7 | 6 |

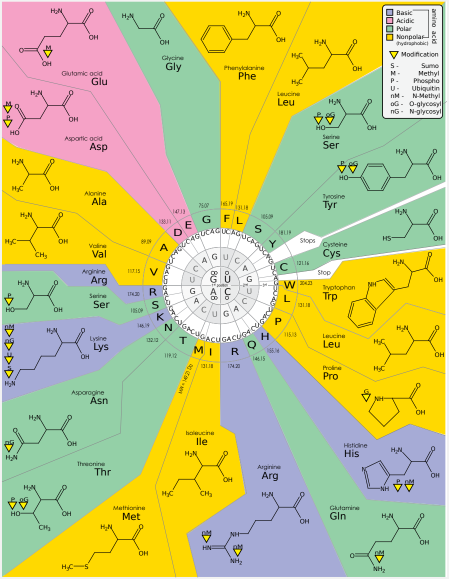
☑ b0 ☑ b1 input: ⟨11⟩ output: ⟨0⟩ (final)

time step 3 out of 10

| ip | instruction | active | pattern | current | next |
|---|---|---|---|---|---|
| | b0⇒ +A4 | ☑ | 0 | 4 | 4 |
| | b1⇒ +A5 | ☑ | 1 | 0 | 0 |
| | [illegible] | ☑ | 2 | 0 | 0 |
| | A0 ≥0 +A0 -A7 | ☑ | 3 | 0 | 0 |
| | A1 ≥0 +A6 -A7 | ☑ | 4 | 4 | 4 |
| | A2 ≥0 +A6 -A7 | ☑ | 5 | 4 | 4 |
| | A4 ≥1 +A7 -A6 | ☑ | 6 | 0 | 0 |
| | A5 ≥1 +A7 -A6 | ☑ | 7 | -4 | -4 |
| | finish reactions | ☑ | | | |
| | A6 ≥3 ⇒c0 | ☐ | | | |
| ☛ | finish time step | | | | |

Triples of *genome bits* are decoded using majority-of-three into *instruction bits,* which are read left-to-right as the binary representations of integers.

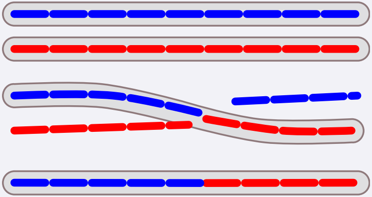
# Why the extra genome encoding layer, majority-of-three?



Picture by Kosi Gramatikoff and Seth Miller, posted on Wikipedia

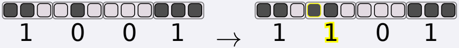
The natural genetic code maps 64 three-letter codons to 20 or so amino acids plus a few control codes. Most amino acids are represented by several synonymous codons. Some of these codons are *robust* in that substitution mutations are likely to result in another synonymous codon, leaving the resulting protein unchanged. Ex: CUU ⇝ CUC, both code for leucine. Other codons are *fragile* in that they are more likely to mutate to a different amino acid. Ex: UUA ⇝ UUC switches from leucine to phenylalanine. An excess of fragile codons is thought to be evidence that a gene has recently been subject to selection [Plotkin and Kudla, 2011]. The majority-of-three encoding in the simulation is so that a future project can address this phenomenon.

## Recombination via haploid crossover

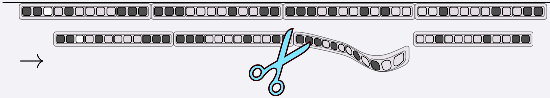


Agents in the simulation have a single chromosome. They reproduce in pairs. The genomes are aligned at the beginning and split at a random location. The first part of one (blue) is attached to the second part of the other (red) to form the offspring genome.

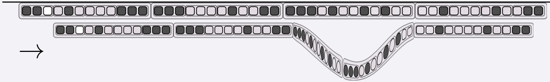
## Supported mutations



single bit substitutions
prob 0.005 per genome bit



gene (instruction) deletion
prob 0.001 per gene

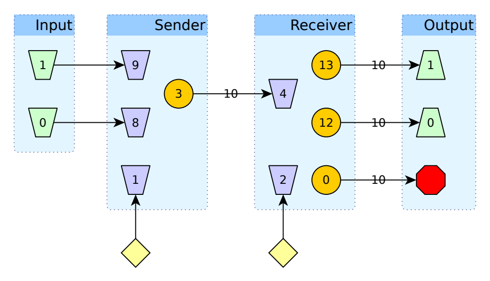


gene (instruction) duplication
prob 0.001 per gene

# Experimental task: Transmit 2 bits over time

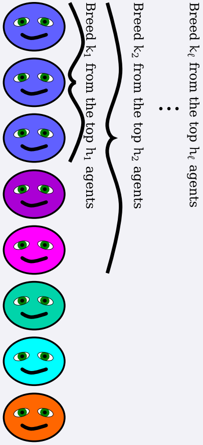
We use a selection-mutation process to evolve solutions to a sequential coding problem. A genome is built into an *agent* follows: Two UMs are built from the genome, one sender and one receiver. The sender gets two bits of input, plus constant input into its role pattern 1. The receiver must generate two bits of output that reproduce the input, and signal when to stop. The receiver gets input from the sender through a single synapse, plus constant input into its role pattern 2.

## Sketch of agent & scoring details



Each agent is presented with all four possible input words $\langle \mathtt{00} \rangle$, $\langle \mathtt{10} \rangle$, $\langle \mathtt{01} \rangle$, $\langle \mathtt{11} \rangle$, starting at a zero state for each and running for up to 100 time steps. It earns $10,000$ points for each bit correctly transmitted, plus $100 - \max(20, t)$ each time it stops after $t$ steps. Maximum possible score: $10,000 \times 4 \times 2 + 80 \times 4 = 80,320$.

# Selection protocols


Breed $k_1$ from the top $h_1$ agents
Breed $k_2$ from the top $h_2$ agents
. . .
Breed $k_\ell$ from the top $h_\ell$ agents

Agents are sorted in descending order by score. Then 300 new agents are created as follows: $k_1$ new agents are created by breeding pairs selected uniformly at random from among the top $h_1$, then $k_2$ are bred from the top $h_2$, . . . . New agents are added to the top* of the list, and agents on the bottom are killed to maintain a constant population of 500. The list is re-sorted, preserving the original order* in case of a tie.

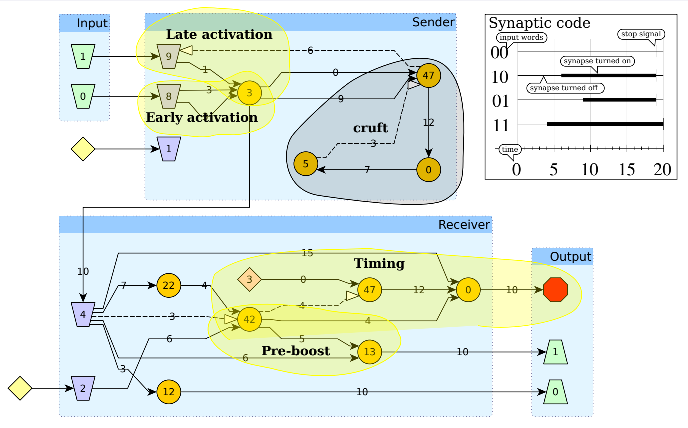
*These details prevent stagnation.

**Selection strengths:**

| Protocol name | $(h_1, k_1), (h_2, k_2) \ldots$ |
|---:|:---|
| Very weak | $(300, 500)$—all agents breed equally |
| Weak | $(100, 200), (500, 100)$ |
| Strong | $(100, 300)$—only top 100 breed |
| Very strong | $(10, 300)$—only top 10 breed |

Even under very weak selection there is still *some* selection because agents with low ratings are more likely to end up at the bottom of the list and die. All of these protocols eventually yield maximum-scoring solutions starting from an initial population of 500 genomes each with 32 genes generated uniformly at random. Patterns have six bits (0-63) and thresholds have four bits (0-15). Each gene has 22 instruction bits, which are encoded by majority-of-three as 66 genome bits.

# Example solution to bit transmission problem

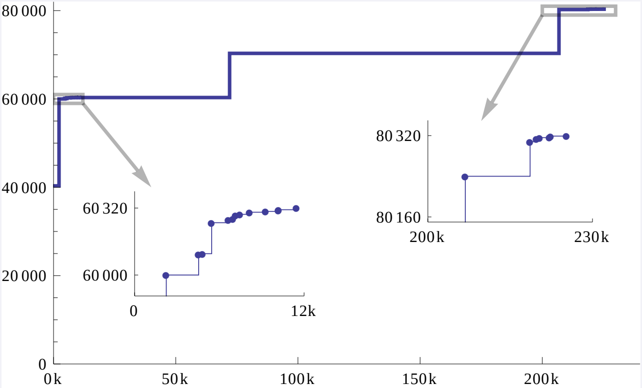


**Typical synaptic code:**

⟨00⟩ is represented by never activating the synapse. ⟨11⟩ is represented by very early activation of the synapse. ⟨10⟩ and ⟨01⟩ are represented by intermediate activation times. This code is typical of solutions evolved by the simulation.

- Chicken-and-egg problem solved incrementally
- Timing mechanism entangled with other mechanisms
- Spurious but harmless activation of receiver mechanism in sender
- Recombination assembles new combinations of genes, discovers better networks, often from sub-optimal parents

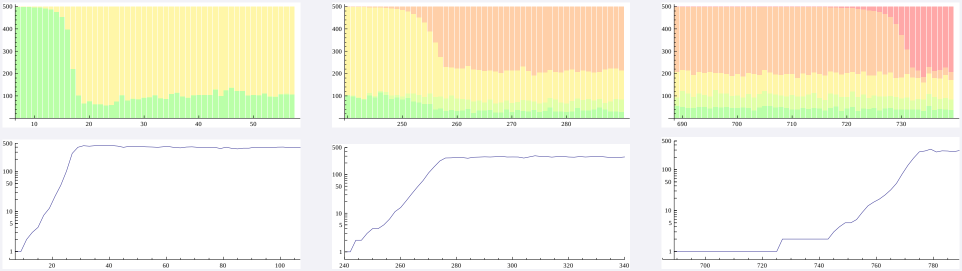
# Rating vs. time



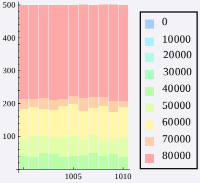
Maximum rating as a function of agent ID number. Agents are numbered as they are born, so ID number measures time. Large jumps occur at each *major innovation*, that is, when an agent appears that transmits an additional bit correctly. Reductions in time spent yield smaller jumps, shown as inserts.

Typical trajectory is as follows: (1) Initially, most agents always output $\langle \texttt{00} \rangle$, scoring $40,000$ = half credit. (2) Recombination joins a link from one input bit directly to the synapse with a link from the synapse directly to the corresponding output bit, scoring $60,000$. (3) A sender mechanism appears that generates a fully informative synaptic code, but no benefit is realized until. . . (4) A mechanism appears that connects to the other output bit, scoring $70,000$. (5) Some sort of complex mechanism is finally discovered that transmits all 8 bits correctly. After each major innovation, the timing mechanism may need to be repaired to get the other 320 points.

# Distribution of ratings by generation



**Color scheme:**



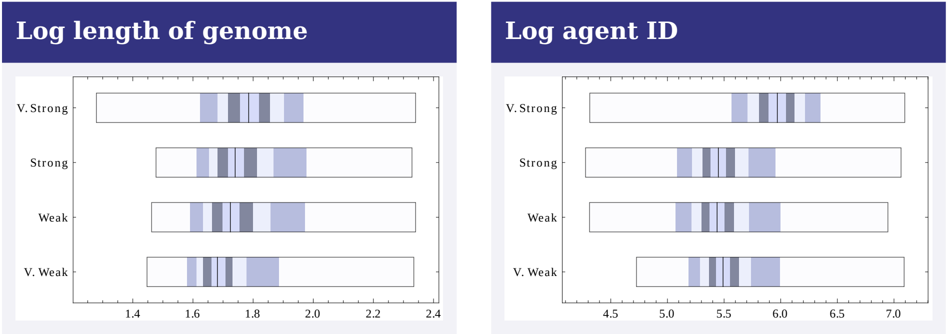
**Upper row:** Stacked bar chart of counts of agents whose score is between $10,000n$ and $10,000(n+1)$ in each generation for 50 generations after each major innovation.

**Lower row:** Log-scale plot of count of agents whose score is nearly maximal in each generation for 50 generations after each major innovation. (That is, the count in the highest corresponding sub-bar in the chart above it.)

Each major innovation is more difficult than the previous ones. When an agent that can transmit an additional bit appears, that ability spreads exponentially, but for higher-scoring innovations, the initial delay is longer, the growth rate is lower, and the equilibrium fraction of high-scoring agents is lower.

# Varying selection strength

Increasing selection strength reduces the number of agents that can reproduce, which has some counter-intuitive consequences. Aggregating observations from $10,000$ runs of the simulation:



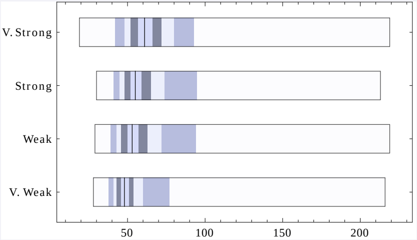
Log length of genome

Log agent ID

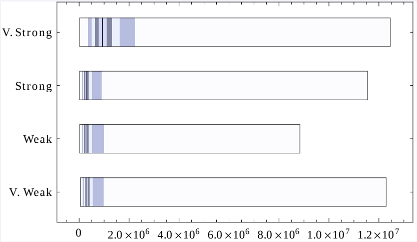
Distributions of the $\log_{10}$ length of the genome (left) and ID number (right) of the first agent in each sample run to achieve the maximum score. Bands indicate deciles. The correlation is imperfect, but in general stronger selection leads to longer genomes, and more agents searched (= more search time).

# Unscaled distribution charts

## Length of genome

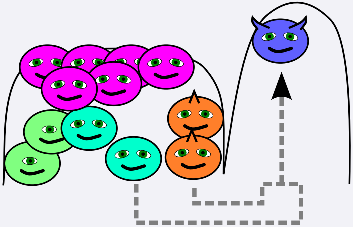


## Agent ID



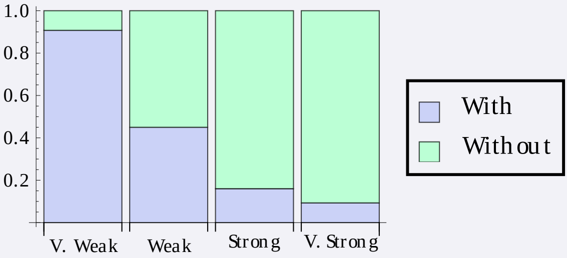
Distributions of unscaled genome length and agent ID number. Bands indicate deciles. A few especially long genomes and long-running samples draw out the uppermost decile.

# Innovations through outliers



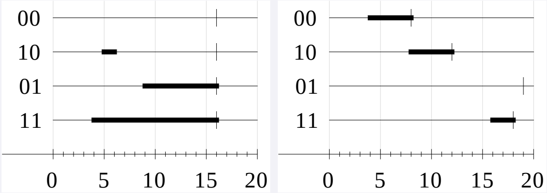
During the long equilibrium phases, most of the population hovers near the peak of a ridge in the fitness landscape. Genetic diversity ensures that there are always a few *outliers* that don't achieve the highest score present in the population at that time. But they are more likely to be near the edge of the zone of attraction of the ridge, in which case their offspring are more likely to jump to another ridge—an innovation.

In weaker protocols, there are more samples with at least one major innovation where at least one parent was an outlier. This phenomenon appears to be a consequence of the fact that weaker selection maintains greater diversity.



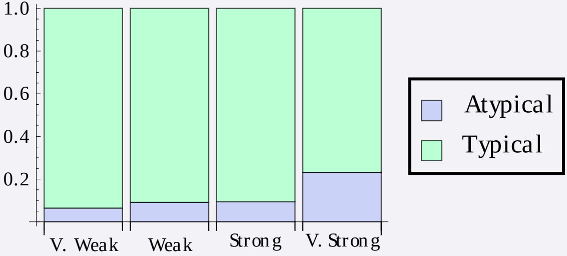
# Typical vs. atypical synaptic codes

**Example atypical synaptic codes:**

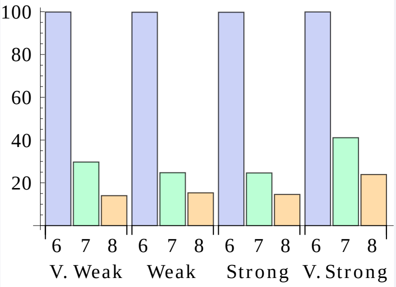


Most solutions use *typical synaptic codes* something like the example solution. But 5-20% of solutions end up with something very different, and often with much more complex mechanisms.

In stronger protocols, a greater fraction of runs settle on atypical synaptic codes. Possibly, the reduced diversity associated with stronger selection increases the likelihood that the population gets trapped in a ridge with a smaller zone of attraction, which yields more atypical results.



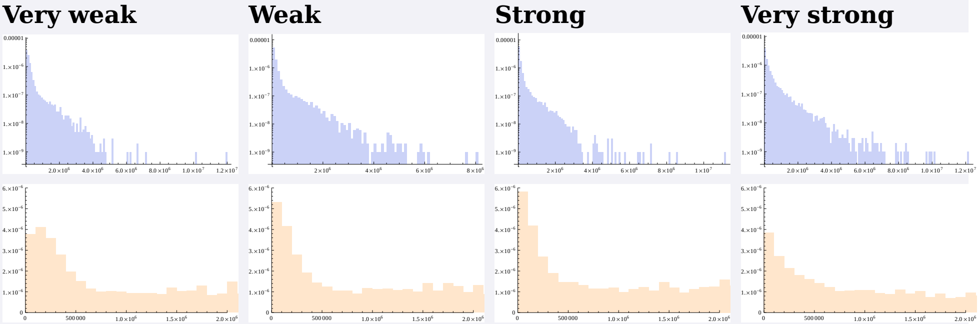
# Timing repairs



Percent of runs that required repair or tweaking of the timing mechanism after the innovation that enables the transmission of an additional bit: The height of the bar over $n$ is the percent of runs that jump from a rating of $10,000n + 320$ to between $10,000(n + 1)$ and $10,000(n + 1) + 319$.

Frequently, a mutation that improves the bit transmission mechanism disrupts the timing mechanism, sometimes stealing part of its structure. Such an event yields a jump of just under $10,000$ in score, followed by minor jumps that restore the additional 320 timing points. Almost all runs have to repair (or initially discover) the timing mechanism after the first major innovation, but only 10-40% do so after the others. The very strong selection protocol has to make the most repairs, but the trend is imperfect: the very weak protocol makes more repairs than the weak and strong protocols after the innovation at $n = 7$. This means that it is not at all unusual (1) for evolution to sacrifice one functional mechanism to build another, and (2) for the resulting mechanisms to be entangled.

# Time for last major innovation



Very weak

Weak

Strong

Very strong

**Upper row:** Log-scale PDF histogram of number of agents searched between next-to-last and last major innovation. **Lower row:** Hazard function histogram of the same.

In traditional population dynamics, mutations are modeled with a Markov chain: An allele $A_j$ in a parent mutates to $A_k$ in a child with a probability that depends only on $j$ and $k$. Starting from a population of all $A_j$, the time to first appearance of $A_k$ is approximately exponentially distributed. However, in the simulation, the time required for the last major innovation (jump from a maximum rating of $70,000\pm$ to $80,000$) is not exponentially distributed: Its PDF is not linear on a log plot, and its hazard function is not constant.

## Hazard function

For a random variable $T$ with density function $f$ and cumulative distribution function $F$, the *hazard function* is

$$h(t) = \lim_{\Delta t \to 0} \frac{\mathbb{P}\left(t \leqslant T \leqslant T + \Delta t \mid T \geqslant t\right)}{\Delta t} = \frac{f(t)}{\mathbb{P}\left(T \geqslant t\right)} = \frac{f(t)}{1 - F(t)}$$

The name comes from the idea that $T$ is the time at which a device fails, so the hazard function is the instantaneous failure rate of the device at time $t$ given that it has survived to time $t$. If $T$ has an exponential distribution, a calculation shows that its hazard function is a constant, equal to the reciprocal of its mean.

# Discussion

These simulations show that the origin story must be complicated. What seems to be happening: The population goes through periods of neutral drift and diversification. Recombination tries many combinations of genetic variants of partially successful genes, including those from parents that score lower than the current maximum (epistasis). Innovations frequently take the form of entangled mechanisms that often must be tuned or repaired.

Stronger selection results in generally longer genomes and greater search time. There are also indications that it traps the population tightly near ridges in the fitness landscape with small zones of attraction (fewer innovations through outliers, more atypical synaptic codes). A possible explanation is that strong selection lowers genetic diversity (founder effect).

The time between innovations is not exponentially distributed, which indicates some memory effect: Shortly after one innovation, the population is more likely to have another. Possibly during a selective sweep (time during which an innovation is spreading) the population retains some diversity which increases the chance of another innovation. If the sweep runs to completion, the process is reset and becomes memoryless.

# Next steps

- Find ways to measure diversity directly and check these possibilities. The genomes are all different lengths with different amounts of space between useful genes, so an alignment algorithm is needed to measure genotypic distances.
- Implement fully diploid genomes. This seems to be an important detail: Recombination may be more likely to find useful combinations of genes when they are at a distance on the genome. Under haploid recombination, it is likely that none of the offspring of such an individual will inherit both parts because crossover will occur in between them. But with diploid recombination, roughly 1/4 of the offspring will inherit both parts.
- Model regulatory network formation more formally. This will require Markov chains with huge state space.

# References

W. Garrett Mitchener. A discrete artificial regulatory network for simulating the evolution of computation. In *EvoNet2012: Evolving Networks, from Systems/Synthetic Biology to Computational Neuroscience*, East Lansing, Michigan, USA, 2012.

Joshua B. Plotkin and Grzegorz Kudla. Synonymous but not the same: the causes and consequences of codon bias. *Nat Rev Genet*, 12(1):32–42, January 2011.

Joshua B. Plotkin and Martin A. Nowak. Language evolution and information theory. *Journal of Theoretical Biology*, 205(1):147–159, July 2000.

Willem Zuidema and Bart de Boer. The evolution of combinatorial phonology. *Journal of Phonetics*, 37(2):125–144, April 2009.