Distribution Agreement

In presenting this thesis or dissertation as a partial fulfillment of the requirements for an advanced degree from Emory University, I hereby grant to Emory University and its agents the non-exclusive license to archive, make accessible, and display my thesis or dissertation in whole or in part in all forms of media, now or hereafter known, including display on the world wide web. I understand that I may select some access restrictions as part of the online submission of this thesis or dissertation. I retain all ownership rights to the copyright of the thesis or dissertation. I also retain the right to use in future works (such as articles or books) all or part of this thesis or dissertation.

Signature:

_____ _____

Steven Riley Date

Approximating McDowell's Evolutionary Theory of Behavior Dynamics with Stochastic Neural Networks

By
Steven Riley
Doctor of Philosophy

Psychology

Jack J. McDowell
Advisor

Gordon J. Berman
Committee Member

Robert C. Liu
Committee Member

Michael T. Treadway
Committee Member

Phillip Wolff
Committee Member

Accepted:

Kimberly Jacob Arriola, Ph.D, MPH
Dean of the James T. Laney School of Graduate Studies

Date

Approximating McDowell's Evolutionary Theory of Behavior Dynamics with Stochastic Neural Networks

By

Steven Riley
M.S., Notre Dame de Namur University, 2015
B.S., Stanford University, 2002

Advisor: J. J McDowell, Ph.D.

An abstract of
A dissertation submitted to the Faculty of the
James T. Laney School of Graduate Studies of Emory University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
in Psychology
2022

Abstract

Approximating McDowell's Evolutionary Theory of Behavior Dynamics with Stochastic Neural Networks

By Steven Riley, M.S.

Behavioral selectionism is the metaphor that learning is like evolution, where successive generations of behaviors develop to increase their demonstrated ability to obtain reinforcement. McDowell's evolutionary theory of behavior dynamics (ETBD) is a selectionist system based on a sexually reproducing population of bitstrings that undergoes successive rounds of emission, selection, recombination, and mutation. The ETBD is consistent with quantitative behavioral findings under variable schedules of reinforcement. However, it lacks the ability to generalize across high-dimensional input spaces, and it is not biologically plausible. Two neural network implementations of the ETBD are presented, which allow for generalization and hierarchical organization of behaviors. Rather than housing a population of behaviors, these networks encode a population within their synapse weights. Network rules acting on these encoded populations are shown to approximate operations on the ETBD's explicit populations. The networks are evaluated against twelve quantitative behavioral findings and found to diverge from the results of the ETBD. Genetic drift in the population of behaviors in the ETBD is shown to be responsible for important features of behavior records. Adding a small amount of reinforcement unconditionally at each time step is shown to approximate the effects of genetic drift and leads to convergence between net one and the ETBD's behavior outputs.

Approximating McDowell's Evolutionary Theory of Behavior Dynamics with Stochastic Neural Networks

By

Steven Riley
M.S., Notre Dame de Namur University, 2015
B.S., Stanford University, 2002

Advisor: J. J McDowell, Ph.D.

A dissertation submitted to the Faculty of the
James T. Laney School of Graduate Studies of Emory University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
in Psychology
2022

Table of Contents

Figures

Tables

Approximating McDowell's Evolutionary Theory of Behavior Dynamics with Stochastic Neural Networks

The analogy between evolution and behavioral learning is a metaphor that psychologists and philosophers have used for over a century (Adams, 1998; Andersson, 2007; Börgers & Sarin, 1997; Catania, 1978; Costa, 2011; Donahoe, Burgos, & Palmer, 1993; Glenn & Field, 1994; Glenn & Madden, 1995; Hull, Langman, & Glenn, 2001; Knudsen, 2004; Pringle, 1951; Szilagyi, Zachar, Fedor, de Vladar, & Szathmary, 2016; Thorndike, 1998; Wasserman, 2012). This metaphor says that unobserved behavior populations undergo a process like evolution, where fitness is a behavior's demonstrated ability to obtain reinforcement. The metaphor is called selectionism, behavioral selectionism, or selection by consequences (Skinner, 1981).

Selectionism

In behaviorism, biology, computer science, and related fields, selectionism encompasses a class of algorithms that search for successively better solutions to a problem. In each selectionist system there is a population of elements, and usually there are birth, death, and change processes that apply to individuals or groups of elements. A selectionist process is "blind," meaning that the population elements do not have access to the space beyond the limited information that they inherit.

Selectionism offers a causal channel that differs from typical theories of the world. Rather than explain with mechanistic or "efficient" causes, selectionism relies upon "final" causes, or causes based on utility (Bacon, 1878; Killeen, 2001). Elements come into being and the environment either nurtures or thwarts them, causing them to survive or die off. In this type of explanation, it is the environment that acts upon the population. This causal mode allows for a

natural link between selectionism and behaviorism, which attempts to explain behavior in terms of its environmental effects (Ringen, 1993; Skinner, 1981; Smith, 1983).

Selection originally and primarily refers to sexual and asexual reproduction. In addition, there are other processes where elements of a population come into being, change, and exit depending on their characteristics. For instance, when planets form from a cloud of dust particles, those particles must be the ones that started with stable orbits; speedy particles escape the solar system while slow ones fall into the sun (Donahoe et al., 1993; Gehrz, Black, & Solomon, 1984). Orbital dynamics act as a filter. As a result, planets are made from particles that have stable orbits, so they in turn will have stable orbits. Depending on the definition in use, one may consider these other processes to be selectionist. A full discussion of the properties of selectionist systems is beyond the scope of this text, and interested readers are referred to Fernando, Szathmary, and Husbands (2012), Hull et al. (2001), and Price (1970, 1972).

There is a major difference between sexual and asexual variants of selectionism. In brief, sexual reproduction allows communication between genomes and causes the population to act as a coordinated whole (Fernando et al., 2012). In contrast, asexual reproduction does not support this type of communication between genomes, so it only allows for search of small areas around the current elements of the population. This paper leverages the geometry inherent in the search accomplished by sexual reproduction and it implements this type of search in neural networks. Studying the geometry of sexual vs. asexual reproduction produces surprising and beautiful results that are beyond the scope of this text. Interested readers are referred to Whitley (1994)’s tutorial on genetic algorithms for a fascinating and highly readable account.

The Evolutionary Theory of Behavior Dynamics

McDowell's evolutionary theory of behavior dynamics (ETBD, McDowell, 2004) is an implementation of behavioral selectionism. The ETBD generates a stream of behavior that is amenable to the same analyses that are applied to the behavior of live organisms. The ETBD is a genetic algorithm with a population of potential behaviors. Successive generations of this population undergo selection, recombination, and mutation. Each generation emits one behavior into a user-defined environment, which may trigger the arrival of a reinforcing or punishing consequence. Behavior streams from the ETBD show remarkable similarity to studies of live animals under variable reinforcement schedules (McDowell, 2019). First, the common implementation of the ETBD is described[1], then its empirical results are discussed.

Implementation of the ETBD

The ETBD consists of two modules. The first, called the artificial organism (AO), supports a population of behaviors, updates them at each time tick, and generates a behavior stream by emitting one behavior per tick. The second module, which simulates an environment, tracks reinforcement schedules and decides if an emitted behavior has produced a reinforcer. These two modules only communicate via emitted behaviors and obtained consequences, so neither has knowledge of the other's implementation.

The Artificial Organism. The commonly used rule set of the AO is as follows.

1) Initialization

[1] Calvin (2019) explores variants of the ETBD and their effects on output. A recent extension to the ETBD is the inclusion of punishment (McDowell & Klapes, 2019). These variants are less commonly used in studies of the ETBD and are not included in the implementations found in this paper.

a. An AO creates a population of one hundred behaviors. Each behavior is a number from 0000000000_2 to 1111111111_2 in binary notation, which is zero to 1023 in decimal notation. The genotype of each behavior is defined as its binary string and its phenotype is defined as its decimal value. The one hundred behaviors are chosen with equal probability from these 1024 possible values.

2) Emission

a. At each tick, the AO emits one behavior with equal probability from its current population of 100 behaviors. The environment decides whether the emitted behavior obtains a reinforcer, and if so, what the magnitude of the reinforcer is.

3) Selection

a. Pairs of behaviors from the current population are selected to be parents of elements in the next generation. There will be one hundred pairs in this intermediate set. One behavior can serve as parent for multiple offspring. The two behaviors in each pair must be different.

b. If the emitted behavior from step two obtains a reinforcer, then the parents of the next generation are chosen based on how close[2] they are to the emitted

[2] Phenotype determines distance. For instance, 511 and 512 are one unit apart, even though their binary strings (0111111111_2 and 1000000000_2, respectively) are opposites. Conversely, 0 and 512 are far apart despite having binary strings that differ by only one bit (0000000000_2 and 1000000000_2, respectively). The topology of the phenotype space is a circle, so the distance between behaviors at 0 and 1023 is only one unit. In general, the distance between decimal integers x and y is $\min(|x-y|, 1024-|x-y|)$.

behavior. Behaviors in the current generation become parents at a rate proportional to a "fitness density function" (FDF) that monotonically decreases with the distance to the reinforced behavior. The FDF is a symmetric triangular distribution[3] with mean and mode equal to zero. The width of the FDF is proportional to a hyperparameter[4] called the "selection strength" or "FDF mean," referred to with the Greek letter μ. It equals the mean absolute deviation of the FDF. The default value of μ is forty units, though μ may vary to simulate changes in reinforcement magnitude. Reinforcer magnitude and μ are inversely related: a smaller μ creates a more exclusive set of parents and therefore a stronger reinforcement effect.

c. If the emitted behavior from step two does not obtain a reinforcer, then all behaviors in the current population are equally likely to become parents for the next generation.

4) Recombination

a. Each pair from step three combines genomes via universal crossover (i.e., independent inheritance by bit, Whitley, 1994) to create one new offspring.

5) Mutation

[3] Other studied FDFs include Gaussian, Laplacian, and uniform distributions. See Calvin (2019) for an analysis of FDF variants and their effects on ETBD outputs.

[4] The term "hyperparameter" is used here to emphasize that it is set in advance (as in, e.g., Claesen & De Moor, 2015); in contrast a parameter is a value that an algorithm learns or calculates at runtime.

a. The mutation rate, referred to with the Greek letter φ, is a second hyperparameter that is usually set to ten percent. Each behavior independently is subjected to mutation with a probability of φ. When mutation occurs in a behavior, a single random bit flips one → zero or zero → one.

6) The new population from step five replaces the old population.

7) Steps two through six are repeated until a user-defined condition is met.

The Environment. There is a separate environment module that schedules and dispenses consequences for emitted behaviors. The environment is meant to simulate an operant chamber with manipulanda such as levers or buttons. The collection of all behaviors that trigger a specific object forms a target class[5]. Most behaviors that an animal can emit in an operant chamber do not trigger the manipulanda, and so most behaviors in the emission range of the ETBD do not belong to a target class. Rather, they correspond to off-target behaviors such as grooming, exploring, or sleeping.

Behaviors in the same target class are similar in effect, and these similarities are captured in the ETBD by defining target classes as small contiguous ranges in the phenotype space. They are typically forty-one units wide.[6] The environment schedules reinforcers that may be obtained by any action within the span of the target class. Each target class may keep a separate schedule, typically a random interval (RI) or random ratio (RR) schedule, and most often these schedules

[5] There are multiple ways to interact with an object that cause the same result. For instance, a rat may use one paw or the other to push a lever. Both actions have the same function, and so both belong to the same target class.

[6] Li, Elliffe, and Hautus (2018) argue for target classes that are only eight units wide, which aligns the rate of target behavior with observed behavior rates of live animals.

are independent of each other. Once a reinforcer is scheduled, the next emitted behavior from the target class obtains the reinforcer and causes the scheduling of the next reinforcer.

In single schedule experiments, there is a single target class. In concurrent schedule experiments, where there are two target classes, the placements of classes will minimize the separation in the phenotype space while maximizing genotype distance. Target classes at 471-511 and 512-552 meet these conditions, forming a Hamming cliff of ten bits at their boundary ($511 = 0111111111_2$ and $512 = 1000000000_2$, which differ on all ten bits).[7]

Empirical Tests of the ETBD

McDowell (2019) described the set of empirical findings from the ETBD and compared them with results from analogous experiments on animals, typically rats or pigeons, in operant chambers. These experiments produce measures of long-term steady-state behavior as well as short-term dynamic behavior. This section summarizes a representative set of twelve findings from these experiments and corresponding tests of the ETBD in the same scenarios. Findings are shown with italics. These twelve findings yield the set of hypotheses that are summarized in Table 1.

[7] Large genotype distances between target classes are necessary to create behavior patterns that look like live animal data. If Hamming cliffs between target classes are too small, then behaviors rapidly switch between targets, and preference for the richer alternative drops (Popa & McDowell, 2010). Analogously, in live animals, if there is no cost associated with switching, then responding tends to alternate rapidly and preference for the richer option is weak (Shull & Pliskoff, 1967). To prevent this outcome in live experiments, schedules usually include a changeover delay, which is a brief period after switching targets where no reinforcement is available. The justification for this delay is that it prevents the reinforcement of the switching itself and counteracts superstitious behavior (Catania & Cutts, 1963). Due to the parallel effects found in these two types of experiments, Popa and McDowell (2010) suggest that the Hamming cliff between target classes acts like a changeover delay. While this interpretation explains the data well, the Hamming cliff is a spatial separation rather than a temporal cost. Spatial separations or barriers between manipulanda in live experiments are also effective at preventing rapid switching or superstitious behavior (Aparicio, 2001). There have been no published studies on the effect of temporal changeover delays in the ETBD.

Important short timescale measurements, or "molecular" values, include preference reversals, which are rapid shifts in distributions of expressed behavior following environmental contingencies, and changeovers, which are successive target behaviors in different target classes. The aggregate measurements of note, so called "molar" values, are the total behaviors emitted in each target class (B_1 and B_2), and the total reinforcers acquired per target class (R_1 and R_2).

Molar behavior.

Bivariate Matching on Concurrent RI Schedules. [8]Herrnstein (1961) showed that pigeons working on concurrent RI schedules of reinforcement distribute behavior between the two options such that the ratio of behaviors

$$\frac{B_1}{B_2}$$

approximated the ratio of reinforcers received on those options

$$\frac{R_1}{R_2}$$

. This equivalence became known as the matching law:

$$\frac{B_1}{B_2} = \frac{R_1}{R_2}$$

.

Baum (1974) analyzed a variety of data showing that experimental subjects systematically deviate from the matching law in two common ways. The first, *bias*, occurs when

[8] RI stands for Random Interval. In this type of schedule, when a reinforcer is obtained, the next reinforcer is scheduled by draw from an exponential distribution with time as its dependent variable. The mean of the distribution is the RI schedule value, e.g., RI 1 minute means time between reinforcers is a random variable t such that $t \sim Exponential(\mu = 1\ minute)$.

there is consistent preference for one alternative. Bias is rare unless reinforcers or manipulanda differ, such as water vs. food, or larger force requirements on one key vs. the other. The second, *undermatching*, is when the subject expresses proportionally more behavior on the option with fewer received reinforcers than would be predicted using Herrnstein's matching law. Undermatching is an example of decreased *sensitivity*, whereas the opposite, *overmatching*, shows increased sensitivity.

Baum (1974) proposed a power function extension of the matching law with the flexibility to accommodate deviations in bias and sensitivity:

$$\frac{B_1}{B_2} = b\left(\frac{R_1}{R_2}\right)^a$$

.

This equation is the *generalized matching law* (GML). The parameters b and a account for bias and sensitivity, respectively. When $b > 1$ there is bias toward the first option, and if there is no bias then $b = 1$. When $a > 1$, the organism shows increased preference for the richer schedule. This is known as *overmatching*, which is rare, but can occur when switching between options is difficult (Aparicio, 2001). A value of $a < 1$ corresponds to the more common outcome of undermatching. Most studies of concurrent RI responding find that *organisms undermatch with values of* $a \approx 0.8$ (McDowell, 2013a), *and studies on humans show values close to* $a \approx 0.7$

(finding a; Kollins, Newland, & Critchfield, 1997). The different sensitivities are attributed to methodological differences between studies[9].

When fitting the GML to data, customary practice is to transform it by taking the logarithm[10] of both sides:

$$\log\left(\frac{B_1}{B_2}\right) = a\log\left(\frac{R_1}{R_2}\right) + \log b$$

. Live data fits the transformed equation well. A meta-analysis (Sutton, Grace, McLean, & Baum, 2008) found that there were no residual trends when fitting the transformed version of the GML, and that residuals were homoscedastic.

Later work showed that perfect matching on RI-RI schedules (i.e., $a = b = 1$) maximizes the expected number of reinforcers gained per behavior (Kubanek, 2017). Since RI schedules suffer from diminishing returns, it is always worth distributing behavior on the leaner RI schedule. Assuming the subject emits behavior at a constant rate, Kubanek (2017) showed analytically that the allocation of behavior with the highest rate of return is where perfect matching occurs.

Undermatching is therefore suboptimal in static environments. However, it is notably pervasive, being found consistently across species and over substantial variations in conditions (McDowell, 2013a). While there is no agreed upon reason for this consistency, one explanation

[9] For instance, experimenters often put animal subjects on food deprivation prior to experiments, which increases the sensitivity of the subject to food reinforcers.
[10] All logarithms in this paper are base 10.

is that undermatching is adaptive in a quickly changing environment and keeps options open (McDowell & Caron, 2007).

Live subjects match behavior to other qualities of reinforcement. The concatenated GML (Davison & McCarthy, 2016), or cGML, extends the GML to allow measurement of sensitivity to these other qualities. For instance, a common experimental manipulation is to vary the magnitude of reinforcement, such as by dispensing more food pellets on one option. In studies that use this manipulation, subjects match their behavior rates to reinforcer magnitudes, although the estimated sensitivity parameter is typically lower. The cGML for rate and magnitude is:

$$\log\left(\frac{B_1}{B_2}\right) = a_r \log\left(\frac{R_1}{R_2}\right) + a_m \log\left(\frac{M_1}{M_2}\right) + \log b$$

.

The parameter a_m is analogous to the sensitivity parameter from the GML but applies to reinforcer magnitudes. This equation uses a_r instead of a to avoid confusion.

A testable assumption of this model is that rate and magnitude independently affect behavior. A meta-analysis by Cording, McLean, and Grace (2011) found that bivariate manipulations of rate and magnitude produced data that fit well with the cGML. *The mean value for a_m in their data set is .60* (finding b), which is lower than the .80 value commonly found when measuring a_r. Values for a_m across individual studies ranged from .26 to .87.[11] The authors found no interaction effects between rate and magnitude, and so Cording et al. (2011)

[11] The studies in this meta-analysis had only thirty-four subjects between them. This small sample size may limit the reliability of its conclusions.

concluded that *the available data support the independence of these two factors on behavior rates* (finding c).

There is no direct equivalent to magnitude in the ETBD. So, before testing for these properties in the ETBD, a proxy for reinforcer magnitude must be established. McDowell, Popa, and Calvin (2012) chose to equate reinforcement magnitude with the reciprocal of the selection strength hyperparameter[12]. With this definition of reinforcer magnitude in place, McDowell et al. (2012) ran AOs on concurrent RI-RI schedules with a range of reinforcer magnitudes, reinforcer rates, and mutation rates. Their studies found that for a range of mutation rates between 7.5% and 14%, the outputs of the ETBD met the criteria for a successful bivariate matching test. For these mutation rates, a_r was between .78 and .85 with mean .83, and a_m was between .64 and .71 with mean .68. Residuals from these fits were homoscedastic with no detectable trends. Effects of rate and magnitude were independent. McDowell et al. (2012) concluded that the ETBD produces bivariate matching data indistinguishable from records of live subjects so long as the mutation rate is within the correct range.

"*Exclusive" Preference on Concurrent RR Schedules.*[13] Unlike RI schedules, RR schedules do not suffer from diminishing returns, and the expected reward per unit of effort is

[12] If μ is small, then fewer behaviors will qualify to be parents in the next generation. Those that become parents will be tightly clustered around the reinforced behavior, and so offspring in the next generation will cluster together as well. Target behaviors will appear with higher probability in the upcoming generations. So, a small μ is consistent with a large reinforcer magnitude. Other functions also show this inverse relationship, though using the reciprocal of the selection strength leads to easy calculations of the magnitude ratio.

[13] Confusingly, "exclusive" and "absolute" preference are not the same. Horner and Staddon (1987, p.62) use "exclusive" to describe behavior where at least 83% of responses are on one alternative (McDowell & Klapes, 2018). Exclusive preference is strong but not absolute. McDowell and Klapes (2018) adopt this terminology from Horner and Staddon (1987) and it is also used here.

constant. This relation would imply that in concurrent RR-RR choice paradigms, also known as "two-armed bandits," devoting all behavior to the richer schedule maximizes reinforcement.

Live subjects do not maximize in these experiments. They express most, but not all, behavior on the richer alternative. Notably, this outcome is not dependent upon having a nervous system. Humans (Shah, Bradshaw, & Szabadi, 2016) and slime molds (Reid et al., 2016) both give exclusive but not absolute preference to the richer ratio. *Strength of preference increases with the ratio between schedule densities* (finding d) *and with the overall reinforcement rate* (finding e).

Interestingly, *exclusive preference also develops when schedules are equal, and depends on overall reinforcement rate* (finding f; Horner & Staddon, 1987). This counterintuitive result shows deficiencies in both matching and maximizing accounts of behavior. Since behavior and reinforcement are proportional in RR schedules (i.e., $B \approx a * R$ where a is the schedule ratio), perfect matching happens regardless of behavior distribution, since

$$\frac{B_1}{B_2} = \frac{a * R_1}{a * R_2} = \frac{R_1}{R_2}$$

. And, since the two schedules have the same reinforcement density, no distribution of behavior is better than any other. When RR schedules are equal, subjects match and maximize regardless of their distribution of behavior, so neither matching nor maximizing can predict what will happen. Since the data show a distinct pattern, another mechanism must be at work.

McDowell and Klapes (2018) found that the ETBD followed the patterns found in Horner and Staddon (1987). Unequal concurrent RR schedules produced exclusive preference correlated

with schedule differences and with overall reinforcement rate. Equal concurrent RR schedules produced exclusive preference correlated with overall reinforcement rate.

Molecular Behavior.

Quadratic Changeovers. McDowell et al. (2012) studied the relation between changeovers, reinforcer rate ratios, and reinforcer magnitude ratios. They fit a two-dimensional quadric surface[14], with form $z = Ax^2 + Bxy + Cy^2 + Dx + Fy + G$, for the data set from each mutation rate. The variable z is the rate of changeovers, which are consecutive pairs of target behaviors that fall into different target classes, and the variables x and y are logarithms of the reinforcement rate and magnitude ratios, respectively. Importantly, to date, no researchers have conducted this type of changeover study in live organisms.

McDowell et al. (2012) found that *the surface that best fit the data was a downward-opening elliptic paraboloid with its vertex at* $(0, 0, G)$ (finding g). This surface had vertical cross sections that were downward-opening parabolas and horizontal cross sections that were ellipses. The parameters that create a shape with this description must satisfy $A < 0$, $C < 0$, $B^2 - 4AC < 0$, and $D = F = 0$. When fitting this curve to the data, *the height of the vertex,* G*, increased with the mutation rate* (finding h). *The value* $|A|$ *was larger than* $|C|$ *for all data sets* (finding i).

Rapid Acquisition of Responding. Davison and Baum (2000) conducted a test of reinforcer-by-reinforcer changes in behavior rates. They showed that each successive reinforcer affects expressed behavior ratios, or per the title of the paper, "every reinforcer counts."

---

[14] Properties of quadric surfaces vary depending on the parameters. A full discussion is beyond the scope of this text. See Silvio (1995) for more information.

Reinforcement in this experiment used Stubbs and Pliskoff (1969) style scheduling. In these manipulations, a single RI schedule determines when the next reinforcer is available. Once available, the schedule probabilistically assigns the reinforcer to one of the two targets based on a preset ratio. The subject must obtain this reinforcer before the next gets scheduled.[15]

Davison and Baum (2000) conducted two experiments on pigeons using these schedules. The first was to measure how preference changed for each successive reinforcer as a function of the ratio of reinforcement between the two sides. The authors divided time into bins between successively delivered reinforcers and calculated the logarithm of the ratio of target behaviors during each bin. They found that for all subjects, *ratios reached asymptote after no more than eight consecutive reinforcers* (finding j). Additionally, the *overall reinforcement context affected the rate of approach to asymptote; more dense overall reinforcement corresponded to a more rapid approach* (finding k).

A second experiment looked at the relative effect of consecutive reinforcers for the same target behavior ("confirmations") vs. reinforcers that were obtained on different target behaviors ("disconfirmations"). Confirmations had a smaller effect on preference, but disconfirmations always had a large effect. *A graph of log behavior ratios as a function of the sequence of length one, two and three of confirmations/disconfirmations produced a characteristic interleaved pattern* (finding l; see Figure 1).

[15] In independent schedules, the ratio of acquired reinforcers may be quite different from the ratio of scheduled reinforcers. This occurs, for instance, when the subject distributes all their behavior to one target class. Stubbs and Pliskoff (1969) schedules solve this problem and ensure the acquired ratio is close to the scheduled ratio.

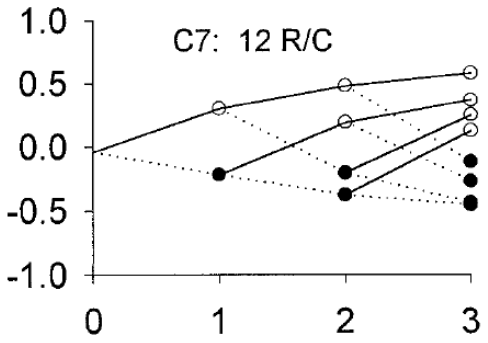


Figure 1. Log behavior ratio of live subject (y-axis) with identifier "C7" as a function of the pattern of the last three reinforcers (x-axis). Black dots show that the last reinforcer was on the right, open dots on the left. From Davison and Baum (2000).

Kulubekova and McDowell (2013) repeated the experiments from Davison and Baum (2000) with the ETBD, and their results showed the same patterns. As each reinforcer arrived, behavior ratios increased and reached an asymptote before eight reinforcers. The rate of approach to asymptote depended on the overall reinforcement density. Additionally, the ETBD reproduced the characteristic interleaved pattern of behavior as a function of sequences of confirmations/disconfirmations (see Figure 2).

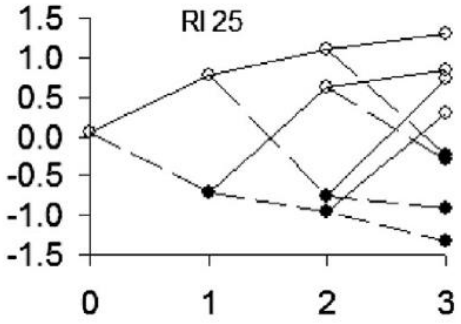


Figure 2. Log behavior ratio of AOs as a function of the pattern of the last three reinforcers. Black dots show that the last reinforcer was on the right, open dots on the left. From Kulubekova and McDowell (2013).

These twelve findings have been converted into hypotheses in Table 1. Any algorithm that claims to replicate the ETBD should satisfy these hypotheses. Networks described in this paper will be evaluated against each of them.

Table 1

Summary of Findings and Hypotheses from ETBD Experiments

Finding/Hypothesis	ETBD source	Live animal source
When the cGML[a] is fit to data from concurrent bivariate RI-RI schedules and mutation rate is between 7.5% and 14%: a. $a_r \approx .83$ b. $a_m \approx .68$ c. Effects of reinforcer rate and magnitude are independent	McDowell, Popa, and Calvin (2012)	Cording, McLean, and Grace (2011), McDowell (2013)
Preference RR-RR schedules: d. Increases with ratio of schedule densities e. Increases with overall reinforcement rate in unequal schedules f. Increases with overall reinforcement rate in equal schedules	McDowell and Klapes (2018)	Horner and Staddon (1987)
For a quadric surface[b] fit to changeover rates from concurrent bivariate RI-RI schedules: g. $A < 0; C < 0; B^2 - 4AC < 0; D = 0; F = 0$; and $G > 0$ h. G increases with mutation rate i. $\lvert A\rvert - \lvert C\rvert > 0$	McDowell, Popa, and Calvin (2012)	None[c]
For Stubbs and Pliskoff (1969) schedules[d]: j. Preference reaches asymptote before eight reinforcers are obtained k. More dense reinforcement causes a more rapid approach to asymptote l. Confirmations and disconfirmations yield a characteristic interleaved pattern	Kulubekova and McDowell (2013)	Davison and Baum (2000)

[a]cGML = concatenated generalized matching law, stated as $\log\left(\frac{B_1}{B_2}\right) = a_r \log\left(\frac{R_1}{R_2}\right) + a_m \log\left(\frac{M_1}{M_2}\right) + \log b$. [b]The equation for this quadric surface is $z = Ax^2 + Bxy + Cy^2 + Dx + Fy + G$, where z is the number of changeovers per 500 generations, $x = \log\left(\frac{R_1}{R_2}\right)$, and $y = \log\left(\frac{M_1}{M_2}\right)$. [c]Hypotheses g, h, and i have not been evaluated on live animals. [d]Stubbs and Pliskoff (1969) schedules use a shared RI schedule with a fixed probability that each reinforcer appears on the left vs. right side.

Gaps Between the ETBD and Live Behavior

The ETBD is a selectionist system that also conforms to quantitative behavioral observations of live organisms under variable schedules of reinforcement. It allows the study of selectionism with the tools developed for the experimental analysis of behavior. However, there are gaps in the ability of the ETBD to model behavior. Three important gaps are generalization, hierarchical organization, and biological plausibility. The following section describes how the ETBD struggles with them. Following each discussion will be an argument why artificial neural networks (ANNs) are well equipped to solve the problem.

Problem One: Generalization of Discriminative Stimuli. A discriminative stimulus elicits behavior that previously led to reinforcement in the presence of that stimulus. In live organisms, the appearance of a discriminative stimulus (e.g., a feeder light changes from red to green) causes changes in behavior (e.g., increased pecking at a key). The behavior is said to be under stimulus control (Urcuioli, 2013). In the current implementation of the ETBD, the only way to model stimulus control is to keep separate populations of behaviors associated with each combination of stimuli. This is a workable solution if there is a finite set of predefined discriminative stimuli and there are adequate computing resources available for training each population. However, with large, unbounded, or continuous stimulus spaces (e.g., all images of a certain size, or all frequencies of visible light), this solution may require more memory and training time than is reasonable. To get around this problem, a user may segment stimuli into equivalence classes and use one population per class. But, segmenting a large stimulus space into predefined bins would not allow for smooth generalization gradients (e.g., small changes in light

frequency produce small changes in behavior rates; Hanson, 1959). Ideally there would be a way for the ETBD to automatically segment or grade discriminative stimuli, though currently there seems to be no way to add this capability without significant changes to the algorithm.

This situation is analogous to the early work on Markov decision processes (MDPs), which are fully observable, time-independent, memoryless reinforcement learning environments. An MDP contains a set states and actions, and each action might produce a reward or cause an agent to transition between states. An early algorithm for optimizing behavior in MDPs was Q-Learning (Watkins, 1989). This algorithm learns an expectation of value for each state-action pair by exploring the world and observing the consequences. This expected value is called a Q-value. Given enough exploration, the highest Q-value in each state will converge to the best action available in that state (Watkins & Dayan, 1992). However, the number of observations required for convergence grows faster than the number of states (Even-Dar & Mansour, 2003). Additionally, Q-learning cannot solve unbounded state spaces in finite time, nor does it generalize learned values to novel states or allow for smooth dependence between similar state/action pairs and Q-values. As a result, the original form of Q-learning only works in small discrete state-action spaces.

Using a neural network to estimate Q-values rather that maintaining a table is an effective way to generalize across large input spaces. Mnih et al. (2015) used a network architecture called "deep Q-learning" to achieve human-level performance across a variety of Atari video games. A deep Q-network takes only the score and raw pixel information from the display as inputs. The space of inputs is extremely large, and to solve this task the network must group together input patterns with similar characteristics. It can then use these groupings to find the expected value of

each action/state pair. No definition of "similar" is available to the network; it spontaneously organizes input patterns according to what is useful for predicting reinforcement. The network automatically segments and grades the space of discriminative stimuli without need for exhaustive sampling.

Problem Two: Hierarchical Behavior Organization. As an organism masters a set of behaviors, its repertoire becomes organized into hierarchical patterns. Limb movements become part of a gait, which allows travel, and travel helps conduct longer term goals. The ETBD has no mechanism to organize behavior patterns in a hierarchy. To do so requires a reliable method of credit assignment (Minsky, 1961), without which an algorithm has no way to divide up a reinforcement signal among a hierarchy of behaviors. Schmidhuber (2000) argues that implementing hierarchical credit assignment in genetic algorithms is achievable by using conditional logic within the genome. Allowing conditional jumps when reading out genetic code might lead to the reuse of sections that are useful, but no such conditional coding is used in the ETBD.

In contrast, ANNs naturally develop a hierarchical organization of information during training. Feedforward ANNs consist of layers of artificial neurons, each of which receives connections from earlier layers and sends messages to later layers. The first level takes raw inputs and the last chooses the output. Later levels learn successively more abstract patterns. For instance, neurons in successive layers of convolutional neural networks (LeCun & Bengio, 1995), which are the first layer in deep Q-networks, form receptive fields that resemble those in the visual hierarchy. Artificial neurons in earlier levels have receptive fields similar to biological neurons in V1 (Ukita, Yoshida, & Ohki, 2019), while neurons in later levels respond like

downstream brain regions, such as the inferior temporal cortex and V4 (Cadieu et al., 2014; Yamins et al., 2014).

Problem Three: Biological Plausibility. The ETBD is not a biologically plausible account of behavior. In contrast, ANNs were originally based on biological networks (McCulloch & Pitts, 1943), though models have since strayed from biological realism. The simplest neuron models in current use are perceptrons, which are the building block of most modern ANNs, and the most realistic are multicompartment simulations based on Hodgkin and Huxley's (1952) equations. Recasting the ETBD within a neural network and gradually increasing the complexity and realism of the neuron models would shed light on why the output patterns of the ETBD resemble those of live animals.

ETBD vs. Other RL Algorithms

Given the inability of the ETBD to perform these expected functions of modern RL algorithms, it may be tempting to dismiss it as a curiosity. Recently developed RL algorithms demonstrate superhuman performance on a wide range of activities (Brown & Sandholm, 2019; Schrittwieser et al., 2020; Vinyals et al., 2019). This prompts the question why one should study the ETBD rather than the current state of the art in RL.

There are two main reasons why standard RL algorithms are not appropriate for the current context. First, the goal of the ETBD is not to maximize reward, but to replicate patterns of animal behavior. The typical paper in RL starts out with a description of a problem, a reward function, and an explicit goal of maximizing that reward function over a problem space. If the problem obeys certain constraints, then typical algorithms in use today, for instance Q-Learning, will converge on the best policy given adequate exposure (Watkins & Dayan, 1992). In contrast,

animals do not maximize reinforcement within the tasks that they are given in laboratory experiments regardless of exposure time. As discussed earlier, animals do not maximize rates of reward per unit effort on variable schedules of reinforcement even after days or weeks of exposure to the task. Using an RL algorithm that always converges to the optimal policy will not provide a good model of animal behavior.

Second, most RL algorithms implicitly or explicitly assume that the problem space is a fully observable Markov decision process (MDP)[16]. These algorithms are based on the idea that only the currently observable state matters, and no information about how the current state came to be is relevant. This assumption does not hold, for instance, under RI schedules of reinforcement. For an MDP, if there is an optimal policy, then it must be deterministic and memoryless (Puterman, 2014).[17] However, on concurrent RI schedules of reinforcement, the optimal policy is mixed, and no deterministic policy can produce the same rate of return (Kubanek, 2017). So, by contradiction, RI schedules are inconsistent with fully observable MDPs. An RI schedule turns a fully observable MDP into a partially observable MDP, and these types of tasks require memory or stochastic policies to solve (Ghosh et al., 2021). The assumptions underlying common RL algorithms are thus inconsistent with studies of animal behavior under RI schedules.

---

[16] An MDP is an environment where an agent takes actions that cause stochastic transitions between discrete world states, and sometimes these actions yield rewards. For the purposes of this discussion, the most important features of an MDP are that the state of the world is always fully observable, and that the reward function only depends on the current state, the last state, and the action taken to transfer between these states. The reward function does not depend, for instance, on the amount of time since the last reward was obtained.

[17] More specifically, if there exists an optimal policy then there must be a deterministic policy within the set of policies that is optimal. There may be optimal stochastic or mixed policies, but they will never dominate the best deterministic policy.

Goals

There are three goals of this thesis. The first is to create an algorithm that approximates the behavioral outputs of the ETBD and live animals on variable schedules of reinforcement. The findings summarized in Table 1 reference the relevant studies on live animals and the ETBD. For each of these findings, studies on the ETBD give more precise predictions; so, matching the results of the ETBD implies being able to match the outputs of live organisms.

The second goal is for the implementation of the new algorithm to be able to increase the generalizability, hierarchical organization capability, and biological plausibility of the ETBD. As discussed previously, neural networks are well suited to improving the ETBD in these ways. There are other benefits to remaking the ETBD as a neural network. For instance, computational neuroscience and AI share an interest in ANNs; a neural ETBD would help to unite the experimental analysis of behavior with these two fields and allow them to speak a common language.

The third goal is to highlight the underlying similarities of the computations performed by sexually reproducing genetic algorithms and stochastic neural networks, thus giving a sound theoretical underpinning to the selectionist metaphor. In particular, the goal is to show that an isomorphism exists between the statistical properties of a certain genetic algorithm and a certain stochastic network. For processes based on these two algorithms, it will be shown that if the distributions of possible outputs P_t are the same at a given time t, and the emitted outputs b_t and consequences c_t are the same, then the expected set of possible outputs at the next time step $E(P_{t+1}(x)|P_t(.), b_t, c_t)$ will be the same.

Stochastic Networks

Stochastic neural networks, which contain units that turn on or off probabilistically based on their inputs, have advantages over real-valued ANNs for these tasks. First, there is a natural correspondence between stochastic hidden neurons and selection of parents during sexual reproduction. Stochastic spiking of hidden neurons, when combined with integration at the output layer, allows for search over hyperplanes much like the recombination step in genetic algorithms (Whitley, 1994). By aligning operations in a stochastic network with those of a genetic algorithm, the expected distribution of outputs becomes identical. This claim will be supported in later sections.

Second, the learning rules available to stochastic neurons have more biological plausibility than those available to real-valued ANNs. The most common way of training ANNs is the backpropagation algorithm (Rumelhart, Hinton, & Williams, 1985). Without adjustments, backpropagation is not a biologically plausible model of learning (Crick, 1989).[18] In contrast, stochastic networks can use learning rules based on spike timing dependent plasticity (STDP; Bi & Poo, 1998) and neuromodulation. STDP occurs when a presynaptic neuron and a postsynaptic neuron both fire within a brief time window, which causes a change of synapse strength. While STDP can perform unsupervised learning, it cannot learn from reinforcement; to do so, there

[18] Surprisingly, the properties of ANNs trained by backpropagation can show striking similarity to biological networks. Depending on the task and the structure of the network, backpropagation can generate neurons that respond like cells at various levels of the ventral visual stream (Yamins & DiCarlo, 2016) or like entorhinal grid cells (Banino et al., 2018). Recently there has been interest in reconciling backpropagation with biology (Lillicrap, Santoro, Marris, Akerman, & Hinton, 2020; Sacramento, Costa, Bengio, & Senn, 2018; Whittington & Bogacz, 2017), though there is still no consensus as to how the brain approximates backpropagation. Whittington and Bogacz (2019) argue that there are multiple mechanisms approximating backpropagation and that the brain may switch among them as development progresses.

would need to be a way for the reinforcement signal to affect the synapse strength. Seung (2003) postulated a three-factor form of Hebbian learning that incorporates a reinforcement signal. Seung called this a "hedonistic synapse." When presynaptic and postsynaptic neurons both fire within a brief window, the synapse creates a long-lasting synaptic tag. Neuromodulators interact with the tag to change the strength of the synapse (Gerstner, Lehmann, Liakoni, Corneil, & Brea, 2018).

This paper presents two stochastic networks that are intended to be the first rungs of a ladder connecting the ETBD to a biologically realistic simulation. Network one is highly constrained with low biological fidelity and implements a close approximation to the ETBD. It has binary synapse weights so that it may faithfully replicate the operations of a genetic algorithm. The second network still runs on a fixed tempo, though it expands the function of the units to include properties that better approximate biological neurons. In particular, the second network solves problems related to generating binary outputs while allowing real-valued synapse weights. Future work may expand this ladder, for instance by creating a network with recurrent connections, continuous-time updates, delayed synaptic conduction, Poisson distributed inputs, or separate inhibitory and excitatory neuron populations.

The functionality of each network is a superset of the functionality of the last one. In this way, these networks are successive "relaxations" of the ETBD toward biological realism. Both networks will be assessed for conformance to the hypotheses in Table 1 using methods identical to those of experiments on the ETBD.

Implementing the ETBD in Stochastic Networks

A reimplementation of the ETBD with stochastic neural networks is sought. An important first step in designing this network is to decide on a mapping between operations of the ETBD and a stochastic network.

How to Map ETBD Functions

To cast the ETBD as a stochastic network, assumptions will be made about which functions correspond between the two algorithms. These assumptions are not the only ones possible, nor does this paper provide any guarantees that they best reframe the ETBD. Given the utility of evolutionary algorithms, it is likely that there are multiple ways that neural networks have co-opted processes akin to evolution (Fernando et al., 2012).

Outputs

The outputs of the ETBD are numbers from 0 to 1023 that have a decimal and a binary representation. A reasonable interpretation that maps between genetic algorithms and binary networks is that an "on" state in an output neuron corresponds to a one in a binary representation, while "off" corresponds to zero. To enforce this assumption, each bit in the ETBD will have a single output neuron as its surrogate. These output neurons receive signals along synapses from a group of hidden neurons and then produce zeros or ones at each time step.[19] The order in which

[19] The approach of using a single neuron to read from a pool of hidden neurons is common in studies of spiking networks (e.g., Fiete & Seung, 2006; Maass, Natschlager, & Markram, 2002; Seung, 2003). However, it is at odds with real biological readouts, which are often not individual neurons, but populations (Pouget, Dayan, & Zemel, 2000). These readout populations can measure a graded response as opposed to a single binary digit, and so have more flexibility and information content (Urbanczik & Senn, 2009). This extension is left to future research.

these ten outputs are read will always be the same. Reading these bits produces a ten-digit binary string that can be translated into a decimal number.

Hidden Neurons and Update Rules

Parents in the ETBD are replaced by hidden neurons whose synapse weights encode genomes. Output neurons integrate signals from these synapses and generate samples from the equivalent mating operations between parents. The selection and mutation operations in the population will not look like their network equivalents, but distributions of outputs will update in ways that are similar. The network selection rules will change outputs to be more like a just-reinforced behavior, and network mutation rules will increase the variability of outputs. When possible, selection and mutation rules will have analytically derived approximations to ETBD operations. Otherwise, hyperparameters will require tuning through pilot testing.

Phenotype Space

In the ETBD, the phenotype range corresponds to the functional distribution of behaviors. The typical interpretation of a target class in the ETBD (e.g., 471 - 511) is that it is a set of behaviors that have the same effect on the environment. Behaviors that are phenotypically similar fall into the same target class. These behaviors can have varying topographies, such as pushing a lever with a right paw or a left paw; what binds them into a target class is that the function of these behaviors is the same. The networks in this paper will do away with the phenotype space for three reasons: lack of support for its importance, difficulty implementing an equivalent space in a network, and concern that its existence implies pre-exposure to an environment.

First, there is some evidence that the phenotype metric has only a marginal effect on behavior dynamics. Popa and McDowell (2010) showed that target classes with the same phenotype distance, but different genotype distances, had markedly different behavior patterns. Calvin (2019) studied a phenotype-only implementation of the ETBD and concluded that it did not replicate the results of the original ETBD. In contrast, unpublished data from Klapes (personal communication, June 12, 2020), who examined the effect of increasing phenotype distance between target classes without changing genotype distance, suggested that this manipulation had a small effect.

Second, there does not seem to be a natural parallel to the phenotype space within a set of binary neurons. To create such a space would require a translation of binary to decimal representations of numbers. This translation is simple in computer code, but it is not clear how one might implement it with binary neurons. Neural nets are universal approximators (Hornik, Stinchcombe, & White, 1989; Maass et al., 2002), so it must be possible to create a structure that mimics the characteristics of the phenotype space. However, a goal of this paper is to create a network instantiation with as few extra parts as possible, so adding an extra unnecessary structure goes against the spirit of these simulations.

Third, for an organism to learn functional similarities between behaviors may require it to have prior exposure to an environment. The topology of the phenotype space makes sense if an organism has had prior training in the environment, but this may not always be the case. There is no mechanism to simulate the learning of functional similarities within the ETBD.

Given these reasons, networks developed in this paper will lack an object that corresponds to the phenotype space. In the ETBD, the phenotype determines the probability that

behaviors will become parents following reinforcement. As a replacement, the selection mechanisms in the described networks will depend only on distance between genotypes (i.e., the Hamming distance). A hypothesis will be that a phenotype analogue produces only small effects on behavior records, and that the networks in this paper will be able to successfully approximate the ETBD without one.

Network One

Properties of Network One

Network one is an attempt to replicate a genetic algorithm as closely as possible. The justification for the similarity between network one and the ETBD comes from an observation by Holland (1992) about the distribution of offspring in a genetic algorithm. The following sections lay out how a stochastic network can approximate the reproduction, selection, and mutation of the ETBD. Encoded patterns in synapse weights are the "population" undergoing these operations.

When the genes in a genetic algorithm are binary strings of length *n*, then a natural representation of the genotype space is an *n*-dimensional analogue of a cube, also known as a hypercube (Holland, 1992; Whitley, 1994). Every instance of a genotype sits at a different vertex in the hypercube, and two genomes share an edge if there is a single bit that differs between them. This organization leads to a fundamental geometric interpretation of reproduction in a genetic algorithm: every unmutated child will lie on the smallest hyperplane[20] that connects its

[20] Hyperplanes are lower dimensional hypercubes contained in a higher dimensional hypercube, such as the square faces of a six-sided cube.

parents (see Figure 3). If each gene assorts independently, as in the ETBD, then the probability distribution of potential children is uniform over this minimal connecting hyperplane (Whitley, 1994). This property gives sexually reproducing populations a unique quality: for every two individuals, there is one encoded hyperplane in the genome space. In contrast, asexually reproducing populations do not share information, so search takes place over a collection of unconnected neighborhoods surrounding the elements of the population. Sexually reproducing populations search hyperplanes, while asexual reproduction searches small neighborhoods.

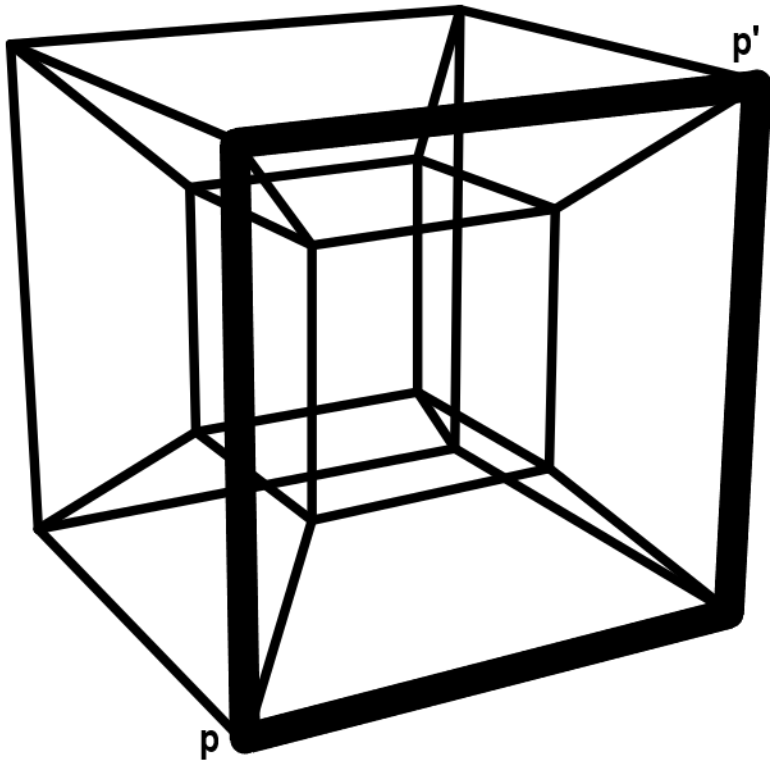


Figure 3. A four-dimensional hypercube represents all possible four-bit genomes. The genomes of two parents, p and p', sit at 0000 and 0011, respectively. Without mutations, any child of p and p' will have a genome somewhere on the heavy black square.

An example will illustrate this principle in action. Consider two bitstrings of 0000000000_2 and 0000000111_2 (phenotypes zero and seven, respectively). When these genomes combine, the offspring obtains bits with equal probability from the two parent genomes. Since

the first seven bits of the parent genotypes agree, it does not matter which supplies the gene, and the offspring gets a zero at these locations either way. For the last three bits, where the parent genomes disagree, there is a 50/50 chance of receiving a one or zero. Representing an indeterminate bit as "*", as in Whitley (1994), the genome of the child is $0000000***_2$. It is now possible to visualize the space of genotypes of all offspring that might result from this pairing. There are eight genotypes that fit the pattern of the child's genome, which are the binary strings that correspond to the integers from zero through seven. These are binary values from 0000000000_2 to 0000000111_2. These genotypes spread across three orthogonal dimensions. Each genotype connects to three other genotypes via single bit-flips (e.g., 0000000000_2 connects to 0000000001_2, 0000000010_2, and 0000000100_2), so this set forms a cube. It is a three-dimensional hyperplane within the ten-dimensional genome.

Network one exploits a parallel between genetic algorithms and stochastic networks. Specifically, it relies on the fact that both structures naturally encode hyperplanes. In genetic algorithms, it is pairs of parents that code for a hyperplane. In contrast, a single hidden neuron in a stochastic network is very flexible and can encode hyperplanes and more complicated objects (see network two). So, network one must be constrained to get the same two-objects-encode-one-hyperplane property of a genetic algorithm.

Intuitively, the synapses of a hidden neuron hold information about whether a set of output neurons should fire. Positive weights increase the probability of an output firing, while negative weights decrease it. The most extreme case is when synapse weights are so strong that the firing of a single hidden neuron causes a specific output pattern with 100% chance. Consider a network architecture that allows this result for synapses with maximal weight. These saturated

synapses correspond to an "always fire" and "never fire" signal, respectively. Without loss of generality, saturated weights are defined to be positive one and negative one.

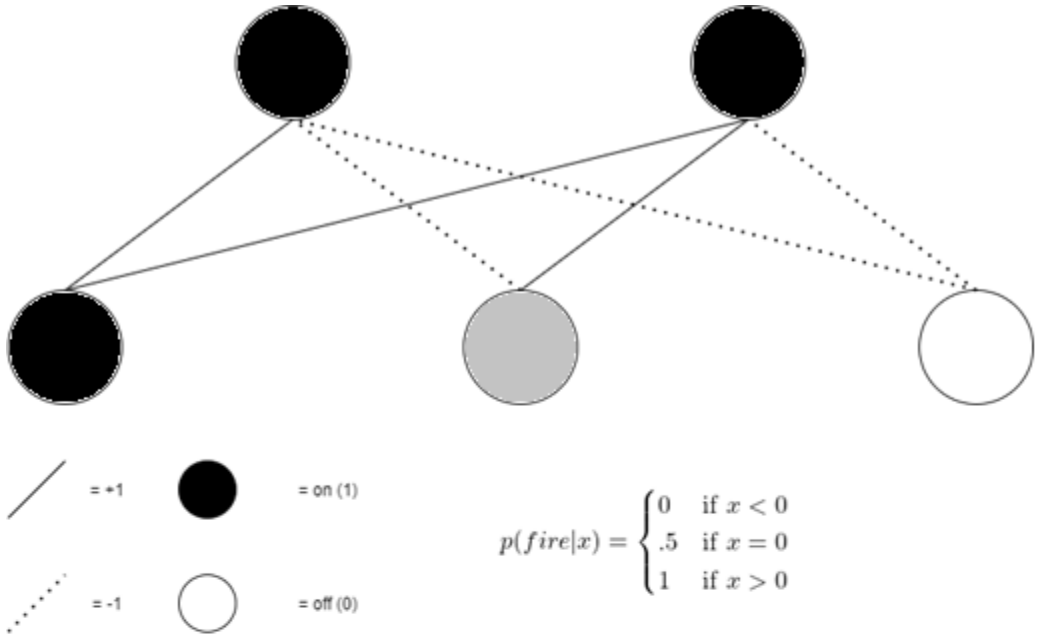
Now, consider what would happen if two such hidden neurons were to fire at the same time. If they agreed as to what a given output neuron should do, then the output neuron would act in a manner consistent with both received messages. But, if the hidden neurons disagree, then the messages contradict. In a genetic algorithm the result of conflicting bits at a locus is to randomly choose the bit. So, network one will give the same answer – there is a 50% chance of a one or a zero output in this case[21]. This answer yields the following rule: if inputs agree, follow orders; if they disagree, behave randomly.

This rule is directly analogous with reproduction in genetic algorithms. When the bits from parents at a given locus agree, the child will contain that bit at that locus; if the parents' bits disagree, the child is equally likely to receive a zero or a one. By constraining network one to only have saturated weights, an isomorphism is established between a hidden neuron in network one and a parent genome in a genetic algorithm. If two such hidden neurons fire at the same time, then the set of "offspring" outputs has the same distribution that would be generated from mating the "parents" that have those encoded bitstrings.

---

[21] Stochastic binary neurons of this type were first proposed by Little (1974). They extend the behavior of the deterministic formal neuron of McCulloch and Pitts (1943). Thompson and Gibson (1981a, 1981b) analyze network behavior of Little's neurons, concluding that they are more computationally realistic than the formal neuron and that they can simulate a wide range of biological network activity patterns.

The bits in a parent genome correspond to the weights of a synapse. A one in the genome corresponds to a synapse weight of positive one, and a zero in the genome corresponds to a synapse weight of negative one. Figure 4 illustrates this correspondence.



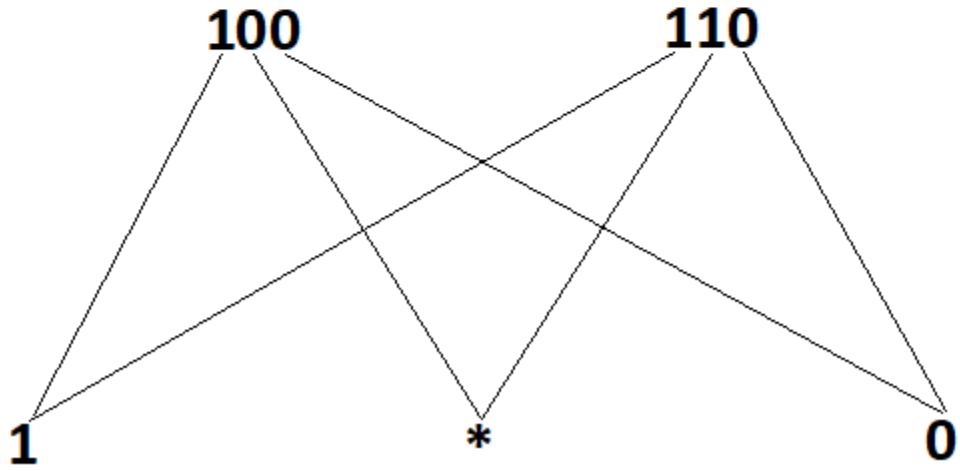


Figure 4: Correspondence between a stochastic network and a genetic algorithm. Both are computing the possible offspring of the binary genomes 100 and 110. Above, the weights of the synapses in the two hidden neurons encode the genomes, where a weight of positive one (solid line) corresponds to a bit of one, and a weight of negative one (dashed line) corresponds to a bit of zero. Output neurons sum their inputs and apply a stochastic transfer function. The probability of firing is 0% if the sum of inputs is negative, 50% if it is zero, and 100% if it is positive. The leftmost output neuron will always fire in this case, the middle is

*indeterminate, and the right will never fire. This output pattern corresponds to the binary pattern 1*0, the same as the output of mating 100 and 110.*

In the ETBD, when an emitted behavior produces reinforcement, the FDF constrains the parents of the next generation to be phenotypically close to the reinforced behavior. Network one has no phenotype space, so this distance metric is not usable. Instead, the selection operation for these networks will move the bits encoded in the synapses closer to those of the reinforced behavior. By probabilistically flipping bits that disagree, the average Hamming distance between each hidden neuron and the output behavior is reduced.

Each hidden neuron encodes the genome of a single parent behavior. After reinforcement, a subset of synapses of every hidden neuron will flip to match the output of the reinforced behavior. Wherever a hidden neuron's encoded output differs from a reinforced output, the synapse weight will change sign with fixed probability ρ. In other words, the reinforced behavior will be one of the parents of every child in the next generation, and ρ controls the proportion of alleles that originate from the reinforced behavior.[22] The hyperparameter ρ governs reinforcement strength like the selection strength hyperparameter does in the ETBD, though these values are inversely related: high ρ acts like a low μ and vice versa. Pilot testing will find the proper mapping between ρ and μ.

Mutation in the ETBD causes random changes from zeros to ones in the population. A corresponding rule for network one would be to randomly flip the sign of synapse weights. The

[22] The reinforced behavior acts like a dominant "alpha" for purposes of mating. Each new element of the population will be a genetic combination between the alpha and an element from the old population. In effect, one parent from each pair will be the alpha, and each element of the current population will be the parent of exactly one child. The hyperparameter ρ equals the penetrance of the alpha's genome into the next generation.

number of bits that mutate in the ETBD is directly comparable to the number of synapse weight changes in network one. Full specification for network one along with the equivalent genetic algorithm can be found in Appendix A.

Discussion of Network One

Network one approximates the operations of the ETBD, though it is not an exact copy. One major difference between the algorithms is the stability of their output distributions over small timescales. In the absence of any reinforcement or mutation, net one's output distribution over the set of behaviors stays steady, but under the same conditions the ETBD's population will change due to the random nature of parent selection. This process is equivalent to genetic drift. Allele frequencies may fluctuate due to this resampling and random noise may accumulate in the system. By design, the two algorithms should have similar output distributions when viewed over long timescales, but in the short term the outputs may have different statistical properties which may affect the outcomes of behavioral experiments.

Network Two

Properties of Network Two

See Appendix A for the full definition of network two. Network two address the problem of reimplementing network one without so many unrealistic restrictions. Network two has the same structure as network one but with a richer set of operating rules. These functional changes are a step toward biological plausibility. There are four major differences between network one and network two, which are discussed below. These differences are real-valued synapses, a smooth transfer function, a multiplicative mutation rule, and an additive selection rule. The first

change, real-valued synapses, necessitates the other three changes, though each difference increases biological plausibility.

The units in this network are stochastic binary neurons with real-valued synapses, which were first proposed and studied by Little (1974). Output units sum the synapse weights from "on" hidden neurons, pass the sum through a smooth stochastic transfer function, and turn on with probability equal to the output of this function. To fit an interpretation consistent with the definition of an excitatory synapse, any stochastic transfer function should be monotonically increasing and bounded between zero and one.

The logit function

$$p(y = 1|x) = \frac{1}{1 + e^{-4x}}$$

, where x is the weighted sum of inputs, will be used as the transfer function in net two. This choice allows for a straightforward mapping of the mutation rule from network one onto network two (see Appendix B). The inverse of this transfer function is

$$x = \frac{1}{4}\ln\left(\frac{p}{1-p}\right)$$

, so synapse weights are proportional to the log odds of an output neuron turning on. When multiple hidden neurons are on, the output neurons sum the weights, thus adding their respective log-odds values of firing.

By allowing real-valued weights, hidden neurons lose their correspondence to points in a hypercube - there are no more saturated synapses that can guarantee a spike in an output neuron. Hidden neurons now correspond to probability distributions over the hypercube. Were synapse weights to approach ∞, $-\infty$, or zero, these output distributions would approach hyperplanes with

coordinates one, zero, or *, respectively. However, in exchange for this loss of correspondence, an individual neuron can now encode what used to require multiple neurons. As a result, fewer hidden neurons may be necessary to achieve the same behavior patterns. The number of hidden neurons, k, becomes a free hyperparameter.

Since weights are no longer binary, network two requires a different mutation rule.[23] With synapses that can take only two weights, the rule was straightforward: change the current weight to the other possible weight, akin to changing between zero and one. But with real-valued weights the same constraint does not hold.

The output statistics of the ETBD are a guide for how to implement mutation. Mutation "flattens" the distribution of behaviors over the genotype space. After repeated mutations, without any other effects present, each bit will become equally likely to be a one or zero. Viewed as a whole, the expected distribution of behaviors approaches the uniform distribution. Network two's mutation operator, whatever its implementation, should therefore have this same limiting action. Since the transfer function of net two has a y-intercept of .5 (i.e., $p(0) = .5$), a uniform distribution on each bit corresponds to a sum of inputs equal to zero. So, moving all synapse weights closer to zero will cause a mutation-like effect.

[23] It might be tempting to equate mutation with flipping the sign of a synapse weight, just like in network one, but this solution would have unacceptable consequences. Mutation would have an insignificant effect if all weight values were close to zero, and it would have a large effect if synapse values were also large. As a result, mutation would cause more changes in behavior when synapse values were far from zero. Large synapse weights cause responding to be consistent and rapid, such as when there has been frequent reinforcement in the recent past. Thus, mutation would have more effect in richer reinforcement contexts. This relation is opposite to what is found in the ETBD (McDowell & Klapes, 2019).

A simple method that approximates the ETBD's mutation operator is therefore to multiply all synapse weights by a constant c between zero and one. Appendix B shows a derivation for a reasonable value of

$$c = 1 - \frac{\varphi}{5}$$

, where φ is the equivalent mutation rate from the ETBD. For instance, the typical mutation rate of 10% corresponds to repeatedly multiplying each synapse weight by .98.[24]

Network two also requires a selection rule. In network one, the rule was to randomly flip a subset of weights of all hidden neurons to align more closely with the output. With real-valued synapse weights a bit more subtlety is allowed, and changes in weights may be smaller and more widespread. Network two's selection rule is a three-factor extension of Hebbian learning, what Seung (2003) referred to as a "hedonistic synapse." In three-factor learning rules, a synapse strengthens if a presynaptic neuron fires, a postsynaptic neuron fires, and a reinforcing neuromodulator enters the synapse. When all three occur, the synapse weight changes. Network two's selection rule equates this neuromodulator release with reinforcement. Following reinforcement, synapse strength will increase by a positive value θ, where θ is a free hyperparameter.[25]

---

[24] The calculation in Appendix B is based on a linear approximation to the transfer function at $p(0)$. There will be deviations from this approximation when the sum of weights is far from zero. Due to the concavity of the transfer function this will cause mutation effects to be weaker in network two than what would be observed in the ETBD or network one. A higher order expansion, or one that worked directly on log-odds values, would improve the fidelity of the mutation operator. This extension is left to future research.

[25] Variations in reinforcer magnitude will alter the amount by which synapse weight changes. The most used value for μ is 40, and reinforcement strength is inversely proportional to μ, so the nominal magnitude of a reinforcer is $m = 40/\mu$. The change to a synapse should be proportional to reinforcer magnitude, so it is $40 * \theta/\mu$.

To make this network function appropriately, there must be a second rule that balances out the first. Without this balancing rule weights would only ever be positive, which would bias the output of the network toward ones. This second rule is a mirror image of the first: when there is a presynaptic spike and reinforcement, but the output neuron has not fired, the synapse weight will decrease by θ. In other words, if the reinforced output contains a zero at bit x, then synapses will be more likely to produce a zero at bit x in the future. This second rule is necessary due to the interpretation of "off" as equivalent to zero in the ETBD.[26] It reinforces the lack of output, but in doing so it runs counter the STDP paradigm.

Discussion of Network Two

Network two includes exponential weight decay (Xiao, Niu, & Wigstrom, 1996) and three-factor Hebbian learning (Gerstner et al., 2018) that are consistent with biology and approximate the mutation and selection rules of the ETBD. Networks one and two can each handle input connections, so they may be capable of solving the stimulus control issue discussed previously. Hidden units in network two are more powerful than the corresponding objects in network one and the ETBD, so network two may need fewer hidden units. With the same number of units, network two can store more information. However, interpreting "off" as "0" in this network necessitates an additional learning rule for balance, and this second rule conflicts with the empirical observations of STDP experiments.[27]

---

[26] A solution to this problem might entail replacing a single output neuron with a pair of mutually inhibitory outputs and labeling one of these outputs 'zero' and the other 'one.' This type of recurrent connection would require either a more complicated update rule or a change from discrete to continuous time. This extension is left to future research.
[27] The learning rule from network one also contained this flaw, though it was less visible due to the binary nature of the network.

Method

Subjects, Apparatus, and Materials

AOs animated by the ETBD, network one, and network two were evaluated in three experiments. The rules for animating these AOs and justifications for design choices can be found above in their respective sections. J. J McDowell originally developed the computer code for the ETBD and the environment module in VB.Net. Other contributors to the ETBD codebase include Olivia Calvin, Bryan Klapes, Cyrus Chi, Andrei Popa, and Saule Kulubekova. The ETBD was transcribed into python and both networks were implemented in python by the current author. The transcription of the ETBD and the implementations of the two networks are publicly available at https://github.com/misterriley/PyETBD. All simulations were run in Python 3.9 on Windows 10.

Procedure

For all experiments, variable schedules were implemented using exponential distributions. That is, if a RI or RR schedule had a nominal value of x, then reinforcers were arranged using an exponential distribution with mean x. See Table 2 for a summary of all experiments.

Pilot Testing: Mapping Hyperparameters Between AOs

Due to unclear relationships between hyperparameter values in the three AOs, hyperparameters were tuned by fitting the outputs of one AO to the outputs of another. The ETBD implementation has two free hyperparameters, which are the mean of the FDF, μ, and the mutation rate, φ.

Phase One. The selection strength parameter in network one, ρ, is inversely related to μ, though the exact relationship is unclear. A function $\rho(\mu)$ that maps between them is required. Phase one of pilot testing generated the data for finding this function. Network one and the ETBD were each run on a series of RI 40/40 schedules of reinforcement, and each used a fixed mutation rate of 10%. For the ETBD, the left alternative always had $\mu = 40$, and the right alternative varied from $\mu = 10$ to $\mu = 80$ by increments of 5. For network one, the value of ρ for the left alternative was fixed at 20%, and the value for the right alternative varied from 5% to 50% by increments of 5%.

Phase Two. This phase of pilot testing generated the data for tuning the hyperparameters of network two, which are θ, the selection strength, and k, the number of hidden neurons. The values chosen to fit on were the sensitivities to rate and magnitude in the bivariate concurrent RI/RI analysis performed by McDowell et al. (2012). These values are $a_r = 0.83, a_m = 0.68$. Since these values are included in the predictions described in the section on empirical findings of the ETBD, there are two fewer actual predictions made for network two. The schedules from McDowell et al. (2012) were used, with RI values of 20/120, 45/95, 70/70, 95/45, and 120/20, varied factorially with μ values of 75/25, 62/38, 50/50, 38/62, and 25/75. There were twenty-five schedules total. Mutation rate was fixed at 10% across all runs. Sixteen runs through the procedure were collected per batch with small random variations in θ and k. Sensitivities to rate and magnitude were calculated for each run, and a second order Newton-Raphson method was used to iteratively minimize the L2-norm distance between outputs and targets.

Experiment One: Bivariate Matching and Changeovers During Concurrent RI-RI Schedules

The first experiment was a replication of selected components of McDowell et al. (2012). Each of the three AOs completed a series of bivariate concurrent RI-RI schedules of reinforcement. RI values were 20/120, 45/95, 70/70, 95/45, and 120/20, μ values were 75/25, 62/38, 50/50, 38/62, and 25/75, and mutation rates were 7.5%, 8%, 10%, 12%, and 14%. RI values, μ values, and mutation rates were varied factorially for a total of 125 schedules per AO. Each schedule in the experiment ran 20,500 generations, and ten repetitions of the experiment were run per AO. Outputs of interest for each schedule were counts of reinforcers obtained per side, behaviors emitted per side, and changeovers per five hundred generations.

Experiment Two: Exclusive Preference on RR-RR Schedules

The second experiment was a replication of McDowell and Klapes (2018). Each of the AOs completed a series of concurrent RR-RR schedules of reinforcement. Mutation rate was fixed at 10% and μ was fixed at 40 for all schedules. For each block of five thousand generations, the output of interest was the percent of target behaviors expressed on the preferred alternative. All schedules were repeated five times.

Phase one examined the distribution of behaviors on unequal schedules of RR reinforcement. Each schedule contained a richer option on the left target class and a leaner option on the right target class. The RR values for the left option were 2, 3, 5, 10, and 15. The RR values for the right alternative were multiplies of the left RR value. The multiples used were 1.1, 1.25, 1.5, 2, 3, 4, 5, 10, and 20 times the left RR value. Schedules were run for 20,500 generations.

Phases two and three examined the distribution of behaviors on equal schedules of concurrent RR reinforcement and how these distributions changed with reinforcement density. In both experiments the AOs were not reset between schedules. Phase two used RR schedules of 160/160, 5/5, and 160/160, and phase three used RR schedules of 2/2, 3/3, 4/4, 5/5, 10/10, 20/20, 40/40, 80/80, 160/160, 320/320, and 640/640. Each schedule in both parts was run for four 5000-generation blocks.

Experiment Three: Preference Development During Stubbs and Pliskoff (1969) Schedules

The third experiment was a replication of phases one and two from Kulubekova and McDowell (2013). Each of the three AOs completed a series of Stubbs and Pliskoff (1969) style concurrent schedules of reinforcement. In each of these schedules there was a fixed probability ratio for where the reinforcer would appear and a combined RI schedule for when the reinforcer would be available. For instance, on a RI 25 schedule with a probability ratio of 3:1, time intervals between reinforcers are drawn from an exponential distribution with mean twenty-five. The location of the reinforcer is random with probabilities of 75% left and 25% right. Only one reinforcer is available or scheduled at a time and a new reinforcer is not scheduled until the current reinforcer is obtained. For all schedules, the mutation rate was fixed at 10% and μ was fixed at 25. Both phases used the same probability ratios, which were 27:1, 9:1, 3:1, 1:1, 1:3, 1:9, and 1:27, and the same RI values, which were a rich schedule with RI 25 and a lean schedule with RI 60. In both phases AOs ran fifty repetitions of each schedule.

For phase one, each schedule was run until ten total reinforcers were obtained. The output of interest in phase one was the ratio of behaviors expressed per side as a function of total

reinforcers obtained. So, each schedule generated ten data points: from beginning to first reinforcer, from first to second, from second to third, etc.

For phase two, each schedule was run until forty total reinforcers were obtained. The output of interest for phase two was the ratio of behaviors as a function of the last three reinforcer locations. For instance, one data point consisted of all behaviors where the last three reinforcer locations formed the pattern L-R-L. There are eight such patterns. Data from all schedules with the same RI value was pooled.

Table 2

Details of Experimental Procedures

Experiment	Source	φ	Schedules and magnitudes	Stop criterion	Reset between schedules?	Repetitions
Pilot testing, phase one	N/A	10%	μ 40/10, 40/15, 40/20, 40/25, 40/30, 40/35, 40/40, 40/45, 40/50, 40/55, 40/60, 40/65, 40/70, 40/75, 40/80 ρ .2/.05, .2/.1, 2/.15, 2/.2, 2/.25, 2/.3, 2/.35, 2/.4, 2/.45, 2/.5 RI 40/40	20,500 generations	Yes	One
Pilot testing, phase two	McDowell et al. (2012)		μ 75/25, 62/38, 50/50, 38/62, 25/75 RI 20/120, 45/95, 70/70, 95/45, 20/120 Varied factorially			Until converged
One	McDowell et al. (2012)	7.5%, 8%, 10%, 12%, 14% Varied factorially	μ 75/25, 62/38, 50/50, 38/62, 25/75 RI 20/120, 45/95, 70/70, 95/45, 20/120 Varied factorially	20,500 generations	Yes	Ten

Experiment	Source	φ	Schedules and magnitudes	Stop criterion	Reset between schedules?	Repetitions
Two, phase one	McDowell and Klapes (2018)	10%	μ 40/40 Left RR 2, 3, 5, 10, 15 Right Left RR x1.1, x1.25, x1.5, x2, x3, x4, x5, x10, x20 Varied factorially	20,500 generations	Yes	Ten
Two, phase two			μ 40/40 RR 160/160, 5/5, 160/160	20,500 generations	No	Five
Two, phase three			μ 40/40 RR 2/2, 3/3, 4/4, 5/5, 10/10, 20/20, 40/40, 80/80, 160/160, 320/320, 640/640	20,500 generations	No	Five

Experiment	Source	φ	Schedules and magnitudes	Stop criterion	Reset between schedules?	Repetitions
Three, phase one	Kulubekova and McDowell (2013)	10%	μ 25/25 Stubbs and Pliskoff (1969) schedules Probability ratios 27:1, 9:1, 3:1, 1:1, 1:3, 1:9, 27:1 RI 25, 60 Varied factorially	Ten reinforcers	Yes	50
Three, phase two				Forty reinforcers		

Results

There were several notable divergences between the behavior records of the ETBD and the two networks. These divergences occurred even though the networks were tuned to produce the same results as the ETBD in other circumstances. The meaning of these divergences and some ways to ameliorate them will be examined in the discussion section.

Pilot Testing: Mapping Hyperparameters

Phase One

The responses of the AOs on each schedule were tallied and response ratios between the left and right alternatives were calculated. For the ETBD, these response ratios were compared against the ratios of μ between the two target classes. For network one, they were compared against the ratios of ρ. *A priori*, equal selection strength on the two sides should produce no bias. The regressions were constrained to meet this assumption. A regression between the log of μ ratios and the log of behavior ratios produced a relationship of

$$\log_{10}\left(\frac{B1}{B2}\right) = -1.044\log_{10}\left(\frac{\mu 1}{\mu 2}\right)$$

. Figure 5 displays this regression.

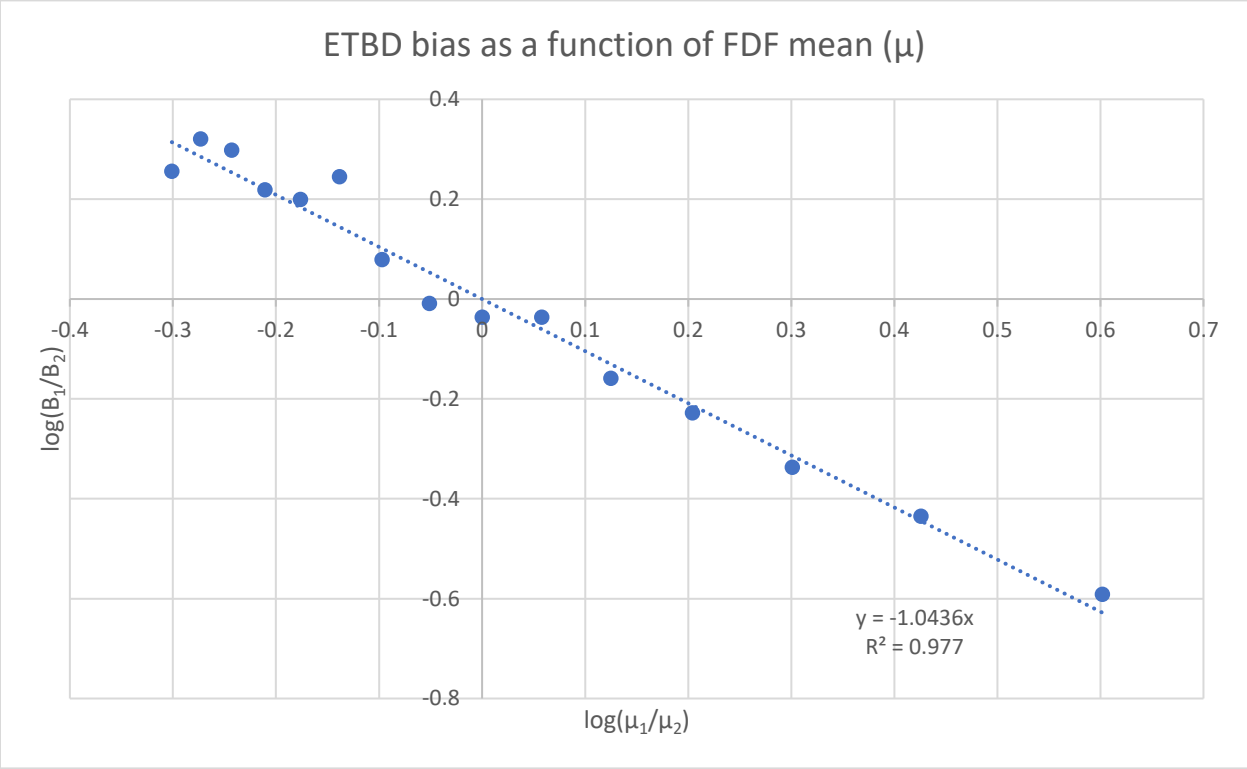


Figure 5. ETBD bias as a function of the ratio between μ values of the fitness density function. The regression was constrained such that it contained the point (0,0), corresponding to no bias when μ values on the two sides are equal. The resulting line is $log_{10}\left(\frac{B1}{B2}\right) = -1.044 log_{10}\left(\frac{\mu 1}{\mu 2}\right)$

.

A similar analysis was performed on the outputs of net one. A highly linear relationship between the ratios of ρ and the logarithm of behavior ratios was found. The function

$$\log_{10}\left(\frac{B1}{B2}\right) = -.552\left(\frac{\rho 1}{\rho 2} - 1\right)$$

fits the desired constraint (see Figure 6).

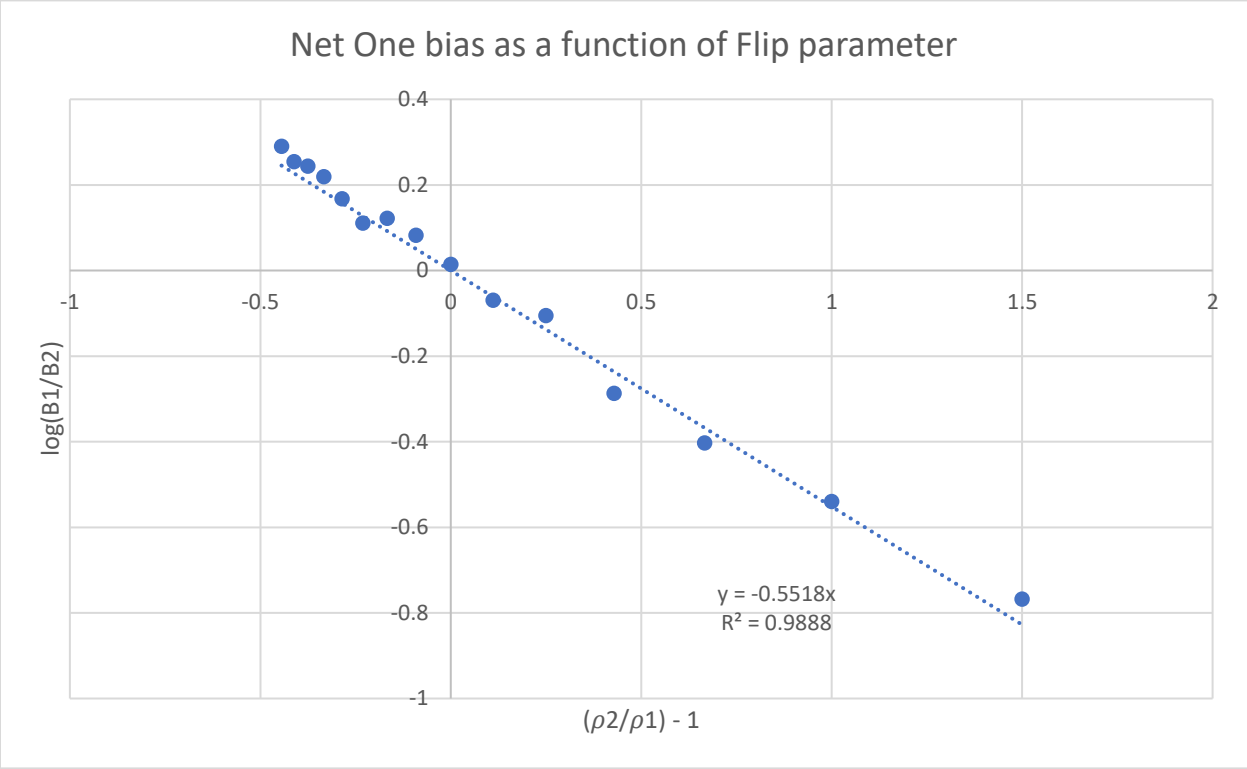


Figure 6. Net one bias as a function of ratio between values of ρ on the two alternatives. The equation is

$$log\left(\frac{B1}{B2}\right) = -.552(\frac{\rho 2}{\rho 1} - 1)$$

.

This data gives an estimate of how the bias relates to both μ and ρ. Combining these two equations yields

$$-1.044 \log_{10}\left(\frac{\mu 1}{\mu 2}\right) = -.552(\frac{\rho 2}{\rho 1} - 1)$$

. Plugging in the fixed values of $FDF2 = 40$ and $\rho 2 = .2$ that were used in the experiments, and solving for $\rho 1$, gives the relationship

$$\rho(\mu) = .2 - .378 \log_{10}\left(\frac{\mu}{40}\right)$$

The following experiments use this function to calculate the value $\rho(\mu)$ that corresponds to a given value .

Phase Two

Net two's best fitting hyperparameters were found to be $(\theta, k) = (1.035, 27)$. This gives a relationship between θ and μ:

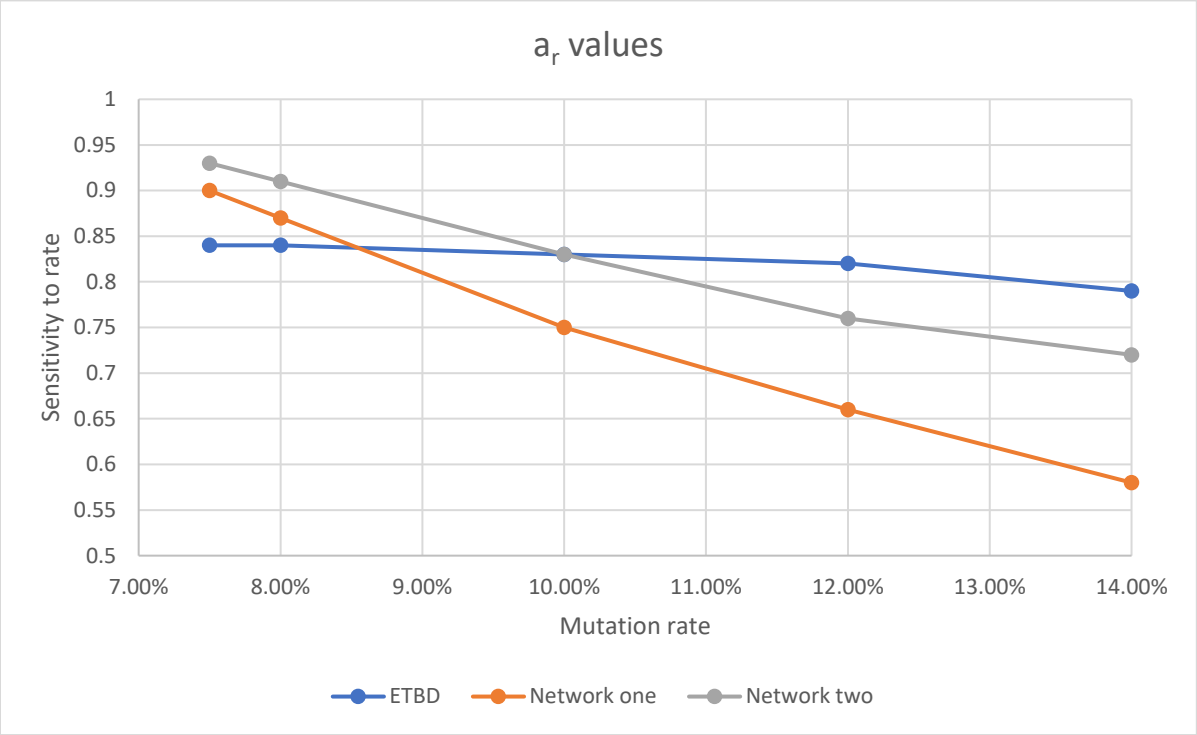
$$\theta(\mu) = 1.035 * (\frac{40}{\mu})$$

These hyperparameters consistently produce the desired coefficients of $(a_r, a_m) = (0.83, 0.68)$ when run on the schedules from McDowell et al. (2012) and a mutation rate of 10%. The following experiments use this function to calculate the value $\theta(\mu)$ that corresponds to a given value of μ, and the implementation of net two in these experiments uses $k = 27$ hidden neurons.

Experiment One: Bivariate Matching and Changeovers During RI-RI Schedules

Expressed behaviors and obtained reinforcers in the two target classes were tallied per schedule variant. For each AO and mutation rate, twenty-five such data points were collected per repetition. The cGML for rate and magnitude was fit to each set of points, and coefficients from these fits were averaged across repetitions. Averaged coefficients are graphed in Figure 7.

At a mutation rate of 10%, the coefficient a_m is close to the hypothesized value of 0.68 for both networks. For net two, the coefficient a_r is close to the hypothesized value of 0.83. However, there is a much wider variation in the sensitivity coefficients than was hypothesized. Mutation rates in net one and net two have a larger effect on sensitivity values than does mutation rate in the ETBD.



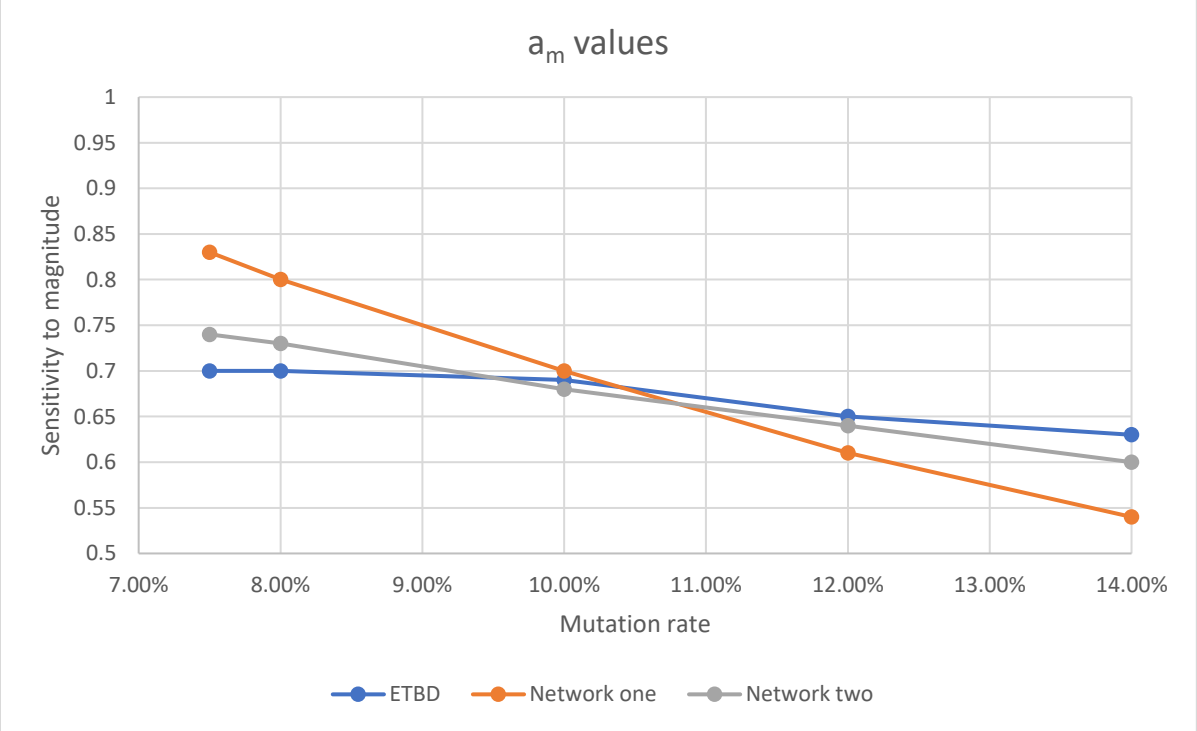


Figure 7. Mean coefficients from fits to cGML from experiment one. Standard errors for data points are all smaller than 0.01 and are omitted.

Changeovers per five hundred generations were calculated per schedule variant, then averaged across all blocks from the same schedule. The quadric surface

$$f(x,y) = Ax^2 + Bxy + Cy^2 + Dx + Fy + G$$

was fit to the 25 points from each combination of AO and mutation rate, where

$$f(x,y)$$

is the number of changeovers,

$$x = \log_{10}\frac{R1}{R2}$$

, and

$$y = \log_{10}\frac{M1}{M2}$$

. Coefficients and values of interest from these fits are shown in Table 3, and values of G are graphed in Figure 8.

Hypotheses concerning changeovers were that for all AOs and mutation rates, $A < 0$, $C < 0$, $G > 0$, $B^2 - 4AC < 0$, and $|A| - |C| > 0$. A surface fitting these constraints will be a downward-opening elliptic paraboloid with a vertex above the xy plane that is wider in the y direction and steeper in the x direction. Additionally, it was hypothesized that as φ increases, so does G.

Net one meets all hypotheses, though the fitted values of coefficients are much larger than those of the ETBD. Net two fails each hypothesis besides $G > 0$ for at least one mutation rate, with 12% mutation causing most hypotheses to fail. The surface with the coefficients reported for net two is a hyperbolic paraboloid with a saddle point at $(0, 0, G)$. Reported coefficients for net two are much more variable than those of the ETBD.

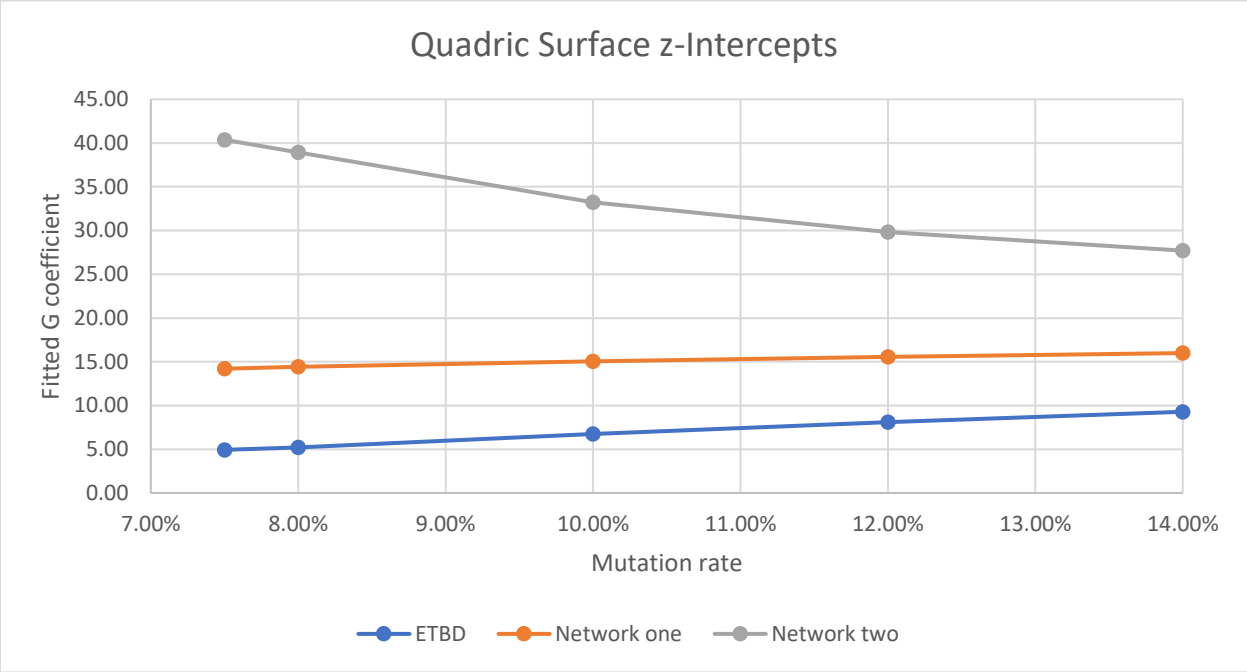


Figure 8. Values of G from fits of changeovers to the quadric surface. These values correspond to the z-intercept of the quadric surface.

Table 3

Quadric Surface Coefficients and Values of Interest

AO	φ	A	B	C	G	$B^2 - 4AC$	$\lvert A\rvert - \lvert C\rvert$
ETBD	7.5%	-1.87	0.11	-0.70	4.93	-5.22	1.17
	8%	-1.80	-0.11	-0.69	5.19	-4.95	1.11
	10%	-2.59	-0.19	-1.31	6.74	-13.54	1.28
	12%	-3.17	-0.56	-1.39	8.08	-17.32	1.78
	14%	-3.84	-0.57	-1.63	9.28	-24.71	2.21
Net One	7.5%	-5.56	-2.63	-2.70	14.20	-53.12	2.86
	8%	.5.57	-2.55	-3.08	14.43	-62.10	2.49
	10%	-5.57	-2.70	-2.98	15.06	-59.12	2.59
	12%	-5.44	-2.51	-2.99	15.56	-58.75	2.45
	14%	-5.50	-2.23	-2.79	15.99	-56.40	2.71
Net Two	7.5%	-5.84	-15.07	**4.61**	**40.37**	**334.79**	1.23
	8%	-5.49	-12.8	**2.70**	**38.92**	**223.13**	2.79
	10%	-1.38	-8.05	**3.05**	**33.21**	**81.64**	**-1.67**
	12%	**1.03**	-5.46	**1.93**	**29.83**	**21.86**	**-0.90**
	14%	**1.72**	-3.35	**1.43**	**27.70**	**1.38**	0.29

Note: Values of *D* and *F* were not significantly different from zero and are not reported. Values that violate hypotheses are displayed in bold.

Experiment Two: Exclusive Preference During RR-RR Schedules

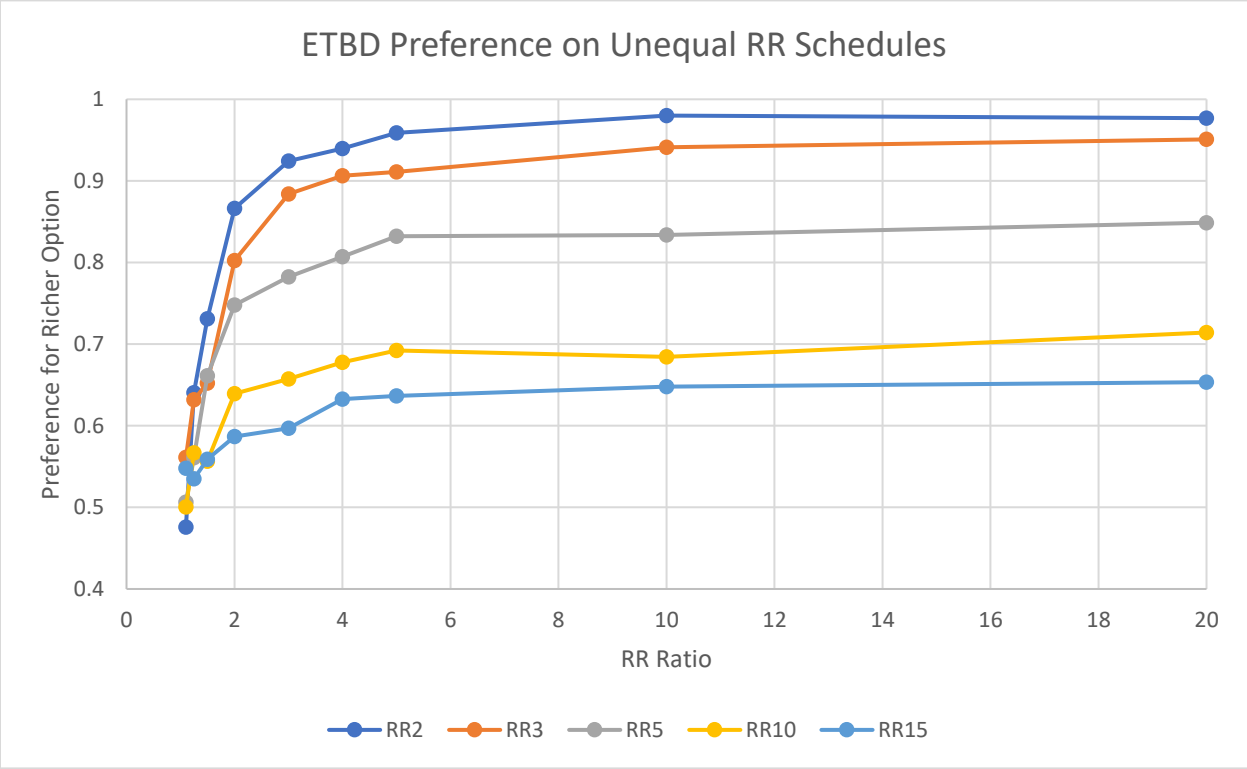
Behaviors expressed on the two target classes were tallied per 500-generation block. The percent of target behaviors on the preferred alternative, which equals

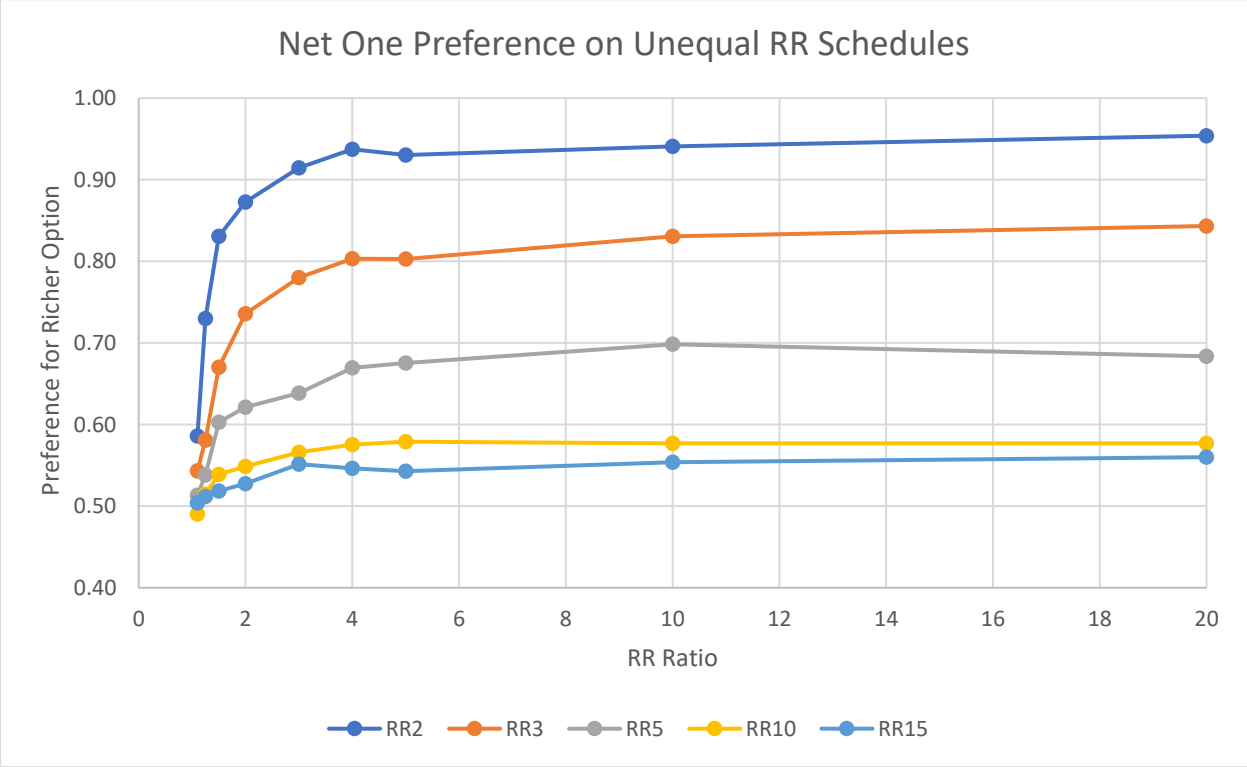
$$\frac{\max(B1, B2)}{B1 + B2}$$

, was calculated. The median of these values per ten consecutive blocks was taken, so each data point in this experiment represents five thousand generations.

Phase One

AOs were evaluated on unequal RR schedules of reinforcement in phase one. Data are graphed in Figure 9. Hypotheses d and e concern how AOs should behave in these settings, and data confirm these hypotheses: as the ratio between RRs increases or the overall rate of reinforcement increases, the preference for the richer alternative steadily increases as well.


ETBD Preference on Unequal RR Schedules
Preference for Richer Option
1
0.9
0.8
0.7
0.6
0.5
0.4
0
2
4
6
8
10
12
14
16
18
20
RR Ratio
RR2
RR3
RR5
RR10
RR15


Net One Preference on Unequal RR Schedules
Preference for Richer Option
1.00
0.90
0.80
0.70
0.60
0.50
0.40
0
2
4
6
8
10
12
14
16
18
20
RR Ratio
RR2
RR3
RR5
RR10
RR15

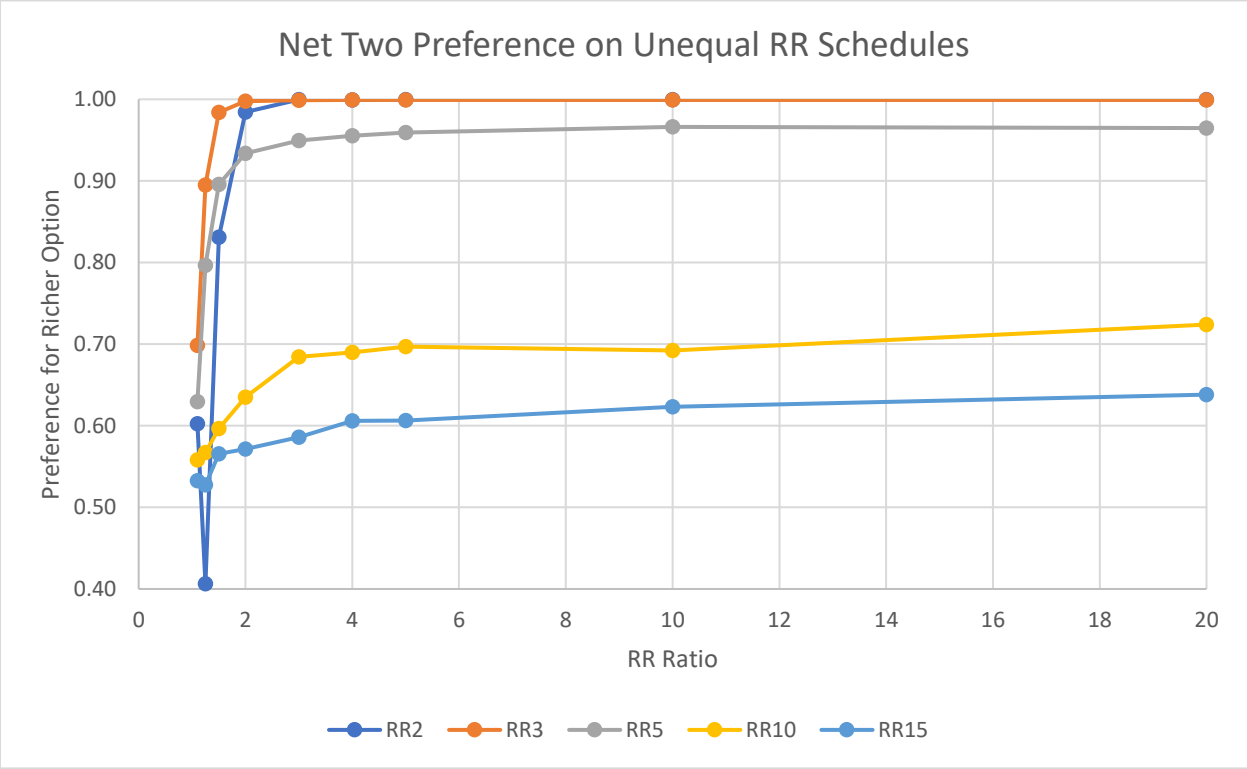


Figure 9. AO Preference for the richer alternative in a pair of unequal concurrent RR-RR schedules of reinforcement. Lines connect values where the left alternative had the RR value shown in the key.

Net one shows less preference for the richer alternative at low overall reinforcement rates. For instance, when the richer alternative is RR 5, net one shows a maximum preference of 0.68 compared to the ETBD's maximum preference of 0.85. In most cases net one's preference is lower than that of the ETBD. In contrast, net two's preference for the richer alternative is unexpectedly high for rich schedules. For the richest schedules net two shows complete preference for the richer alternative even at low ratios of reinforcement density between the two options. The graph for preference in net two is vertical at the left compared to the gradual slopes in the other two graphs, indicating a much higher sensitivity to reinforcement ratios than would be expected.

Phases Two and Three

Phases two and three tested AO behavior on equal schedules of RR reinforcement. Data from both experiments are consistent with hypothesis f. For phase two, median preference as a function of schedule index is graphed in Figure 10. The first and last four points of Figure 10 show preference on RR 160/160 schedules, while the middle four points show preference on RR 5/5 schedules. Net one shows weak preference for either alternative across all schedules. While preference in net two is comparable to the ETBD's for rich RR 5 schedules, its preference for a side on lean RR 160 schedules is also weak.

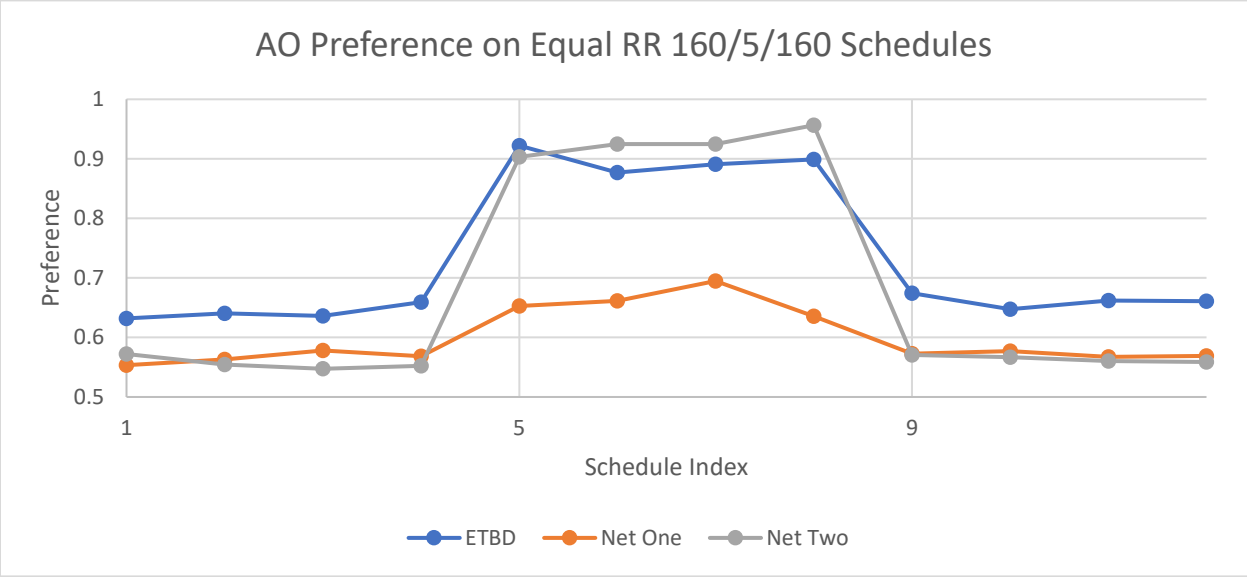


Figure 10. AO preference for an arbitrary alternative during equal concurrent schedules of RR reinforcement. Schedules one through four and nine through twelve are RR 160/160, schedules five through eight are RR 5/5.

For phase three, median preference on 500-generation blocks is graphed in Figure 11. A similar pattern of preference is evident here. For all schedules, preference in net one is much weaker than preference in the ETBD. Net two shows absolute preference for low RR values, but switches to weak preference when the reinforcement rate crosses a threshold between RR 5/5 and RR 10/10.

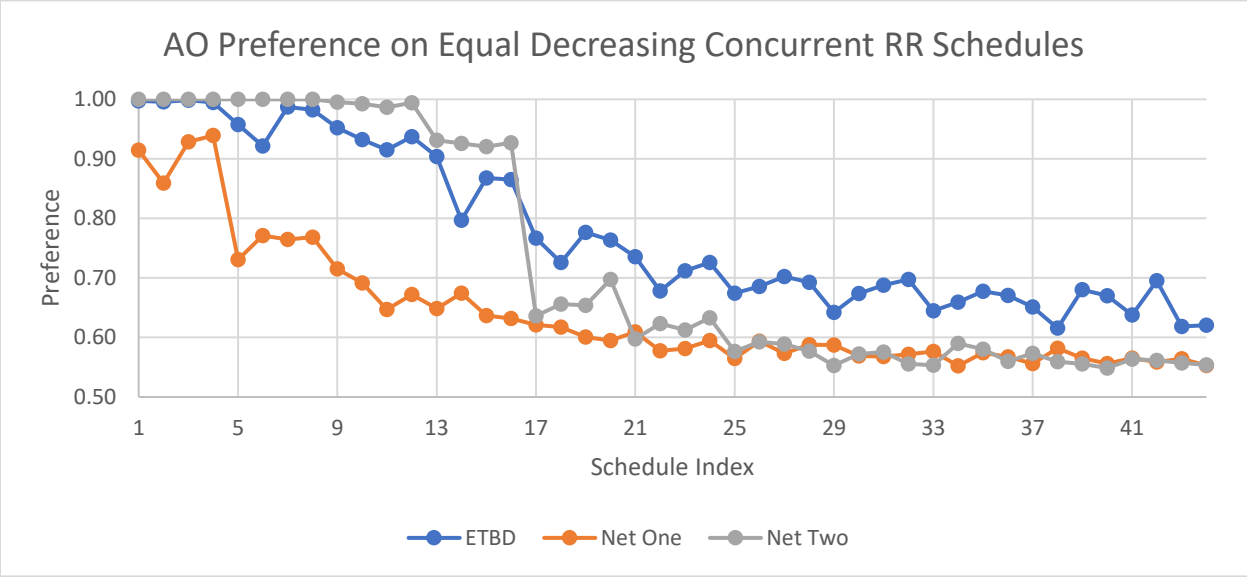


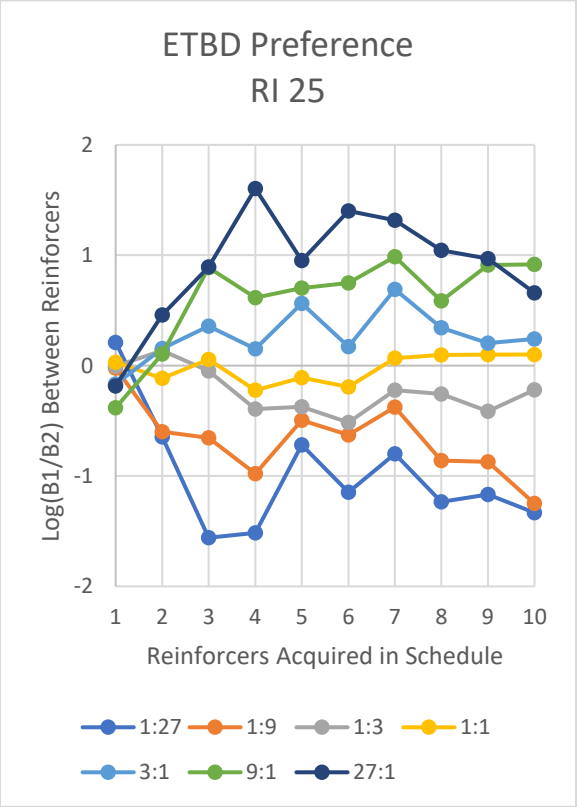
Figure 11. AO preference for an arbitrary alternative during decreasing equal concurrent RR schedules. Schedules start at a rich RR 2/2 and decrease every four points, ending at a lean RR 640/640. The sequence of equal RR values is 2, 3, 4, 5, 10, 20, 40, 80, 160, 320, 640.

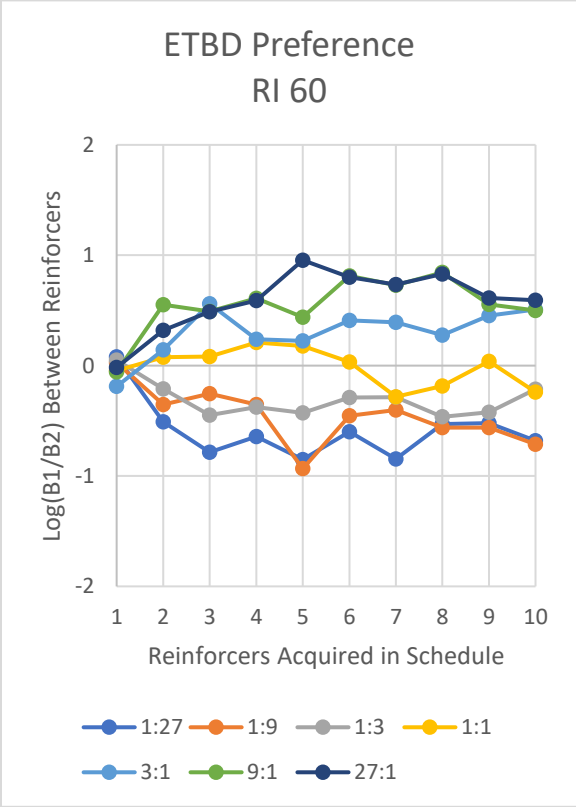
Experiment Three: Preference Development During Stubbs and Pliskoff (1969) Schedules

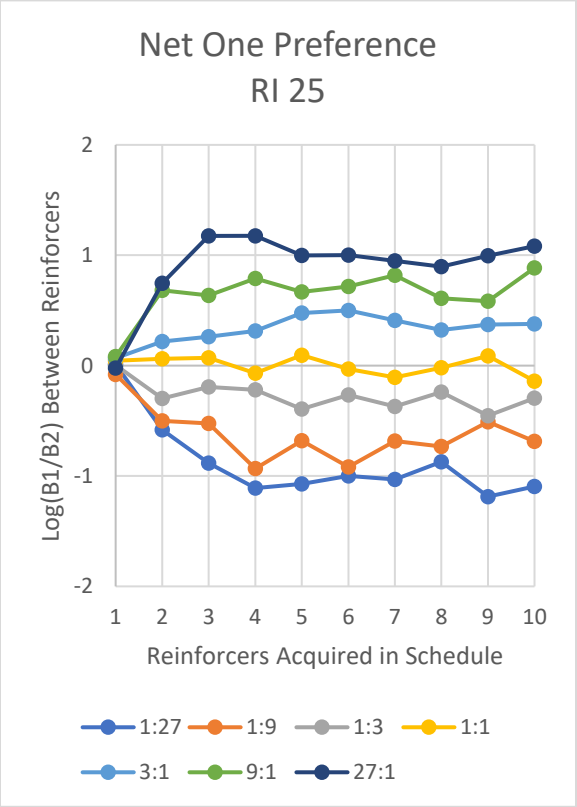
Each data point in experiment three represents an aggregate of behaviors that meet specified criteria across schedules. For each such criterion, the log of the ratio of behaviors between the two alternatives was calculated.

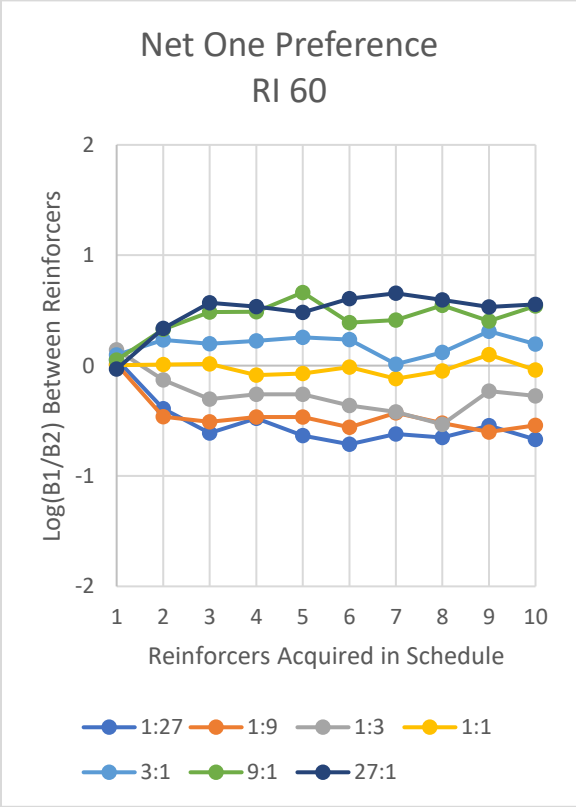
Phase One

In phase one data were aggregated across all schedules with the same reinforcement ratio. Behaviors were placed into bins according to the number of consecutive reinforcers that had already arrived in that schedule. Data from this experiment are graphed in Figure 12. Data are consistent with hypotheses j and k. For all AOs, preference levels off after three to five reinforcers have been acquired, and a richer reinforcement context yields a higher slope of ascent and a higher asymptote.


ETBD Preference
RI 25
Log(B1/B2) Between Reinforcers
2
1
0
-1
-2
1 2 3 4 5 6 7 8 9 10
Reinforcers Acquired in Schedule
1:27 1:9 1:3 1:1
3:1 9:1 27:1


ETBD Preference
RI 60
Log(B1/B2) Between Reinforcers
2
1
0
-1
-2
1 2 3 4 5 6 7 8 9 10
Reinforcers Acquired in Schedule
1:27 1:9 1:3 1:1
3:1 9:1 27:1


Net One Preference
RI 25
Log(B1/B2) Between Reinforcers
2
1
0
-1
-2
1 2 3 4 5 6 7 8 9 10
Reinforcers Acquired in Schedule
1:27 1:9 1:3 1:1
3:1 9:1 27:1


Net One Preference
RI 60
Log(B1/B2) Between Reinforcers
2
1
0
-1
-2
1 2 3 4 5 6 7 8 9 10
Reinforcers Acquired in Schedule
1:27 1:9 1:3 1:1
3:1 9:1 27:1

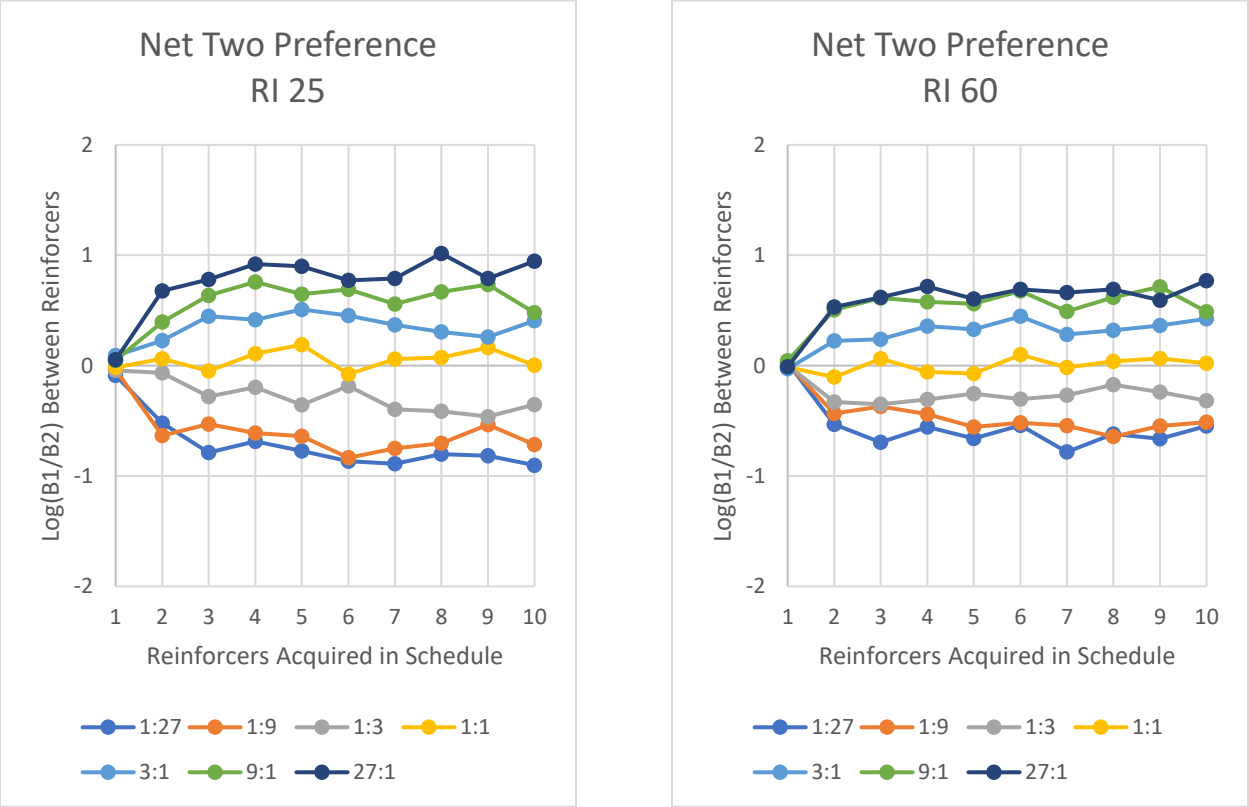


Figure 12. AO development of preference under Stubbs and Pliskoff (1969) schedules of reinforcement.

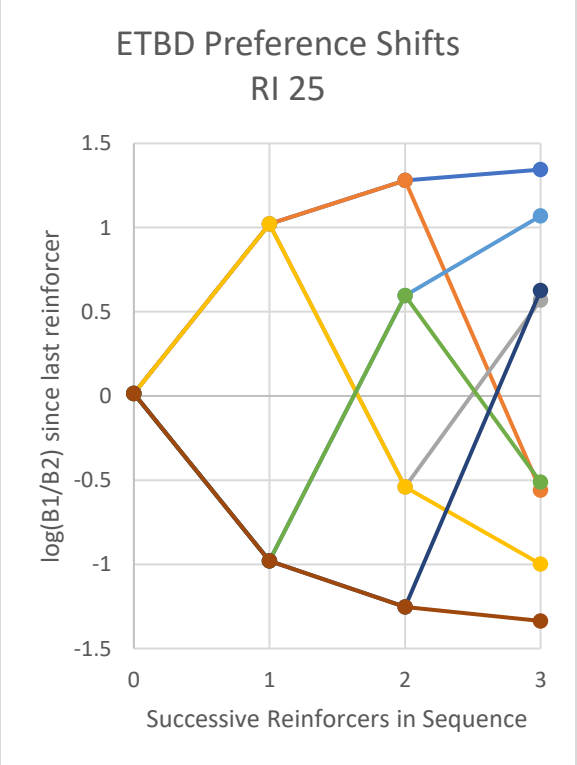
The preference development of the ETBD is more erratic compared with the smoother curves generated by data from net one and net two. The difference between the richer and leaner reinforcement contexts has less of an effect on net two than it does on net one.

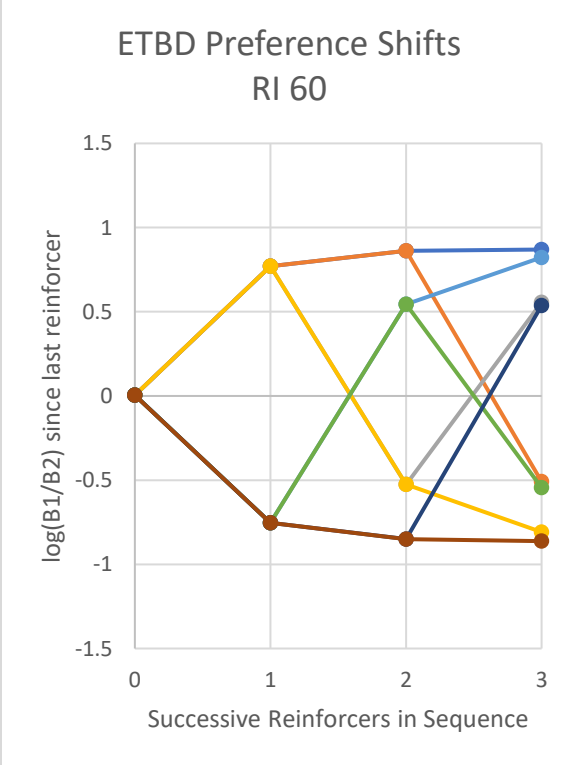
Phase Two

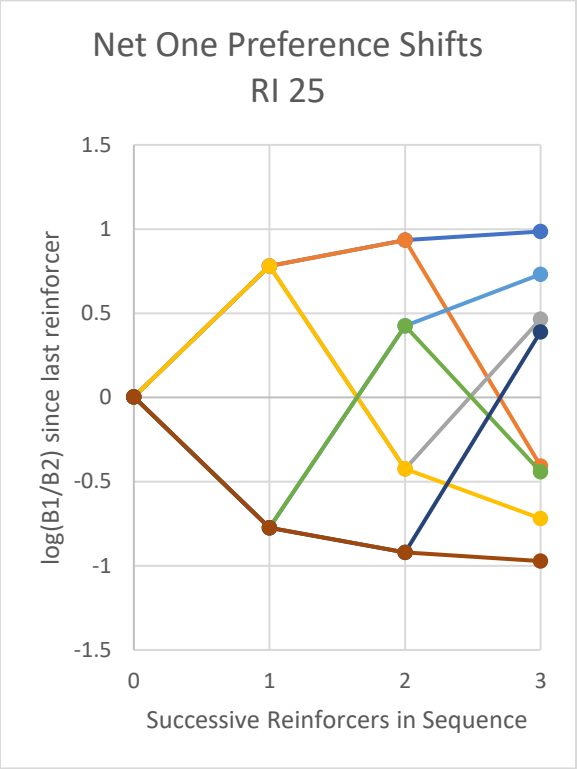
Phase two shows how AO preferences shift when encountering two consecutive reinforcers on the same side (a "confirmation," using the terminology of Davison and Baum, 2000), or on alternating sides (a "disconfirmation"). Data from this experiment are graphed in Figure 13. Each graph contains a series of splits showing the difference between preference following a reinforcer obtained on the left vs. on the right. All graphs start close to the origin, which shows that behaviors are evenly distributed between the two target classes across the

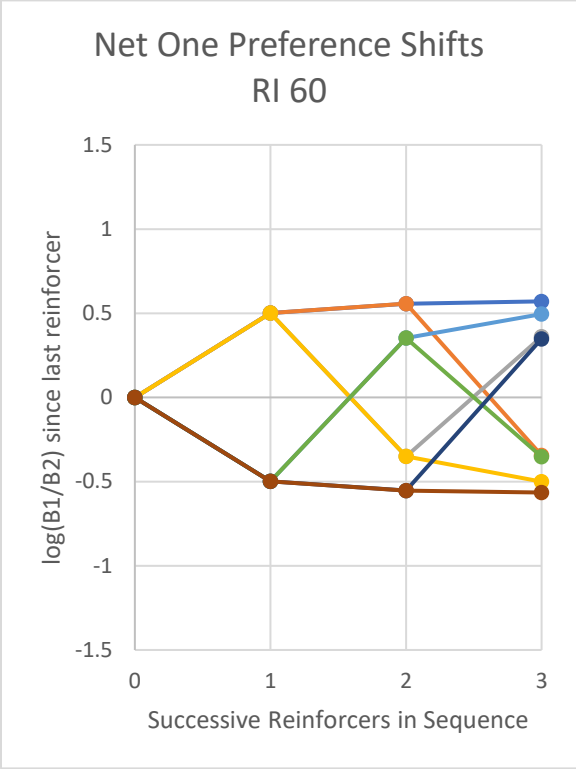
entire experiment. The points with an x-value of one show preferences given that the last obtained reinforcer was on the left or right alternative, with higher y-values indicating more preference for the left. In each case, a confirmation leads to a small shift in preference while a disconfirmation causes a larger shift. Shifts are larger in the richer reinforcement context of RI 25 and smaller in the leaner reinforcement context. This pattern is consistent with hypothesis i.

While the overall pattern is similar between AOs, the ETBD shows a larger maximum preference at both RI values. The highest point on the first graph, which occurs when the ETBD obtains three reinforcers in a row on the left alternative for RI 25, has a y-value of 1.34. The same point for net one has a y-value of .99, and for net two, 1.00. Since this is a logarithmic scale, the ETBD's maximum preference after three reinforcers is more than twice as large as the maximum of the other two AOs.






Net One Preference Shifts
RI 25
log(B1/B2) since last reinforcer
1.5
1
0.5
0
-0.5
-1
-1.5
0
1
2
3
Successive Reinforcers in Sequence


Net One Preference Shifts
RI 60
log(B1/B2) since last reinforcer
1.5
1
0.5
0
-0.5
-1
-1.5
0
1
2
3
Successive Reinforcers in Sequence

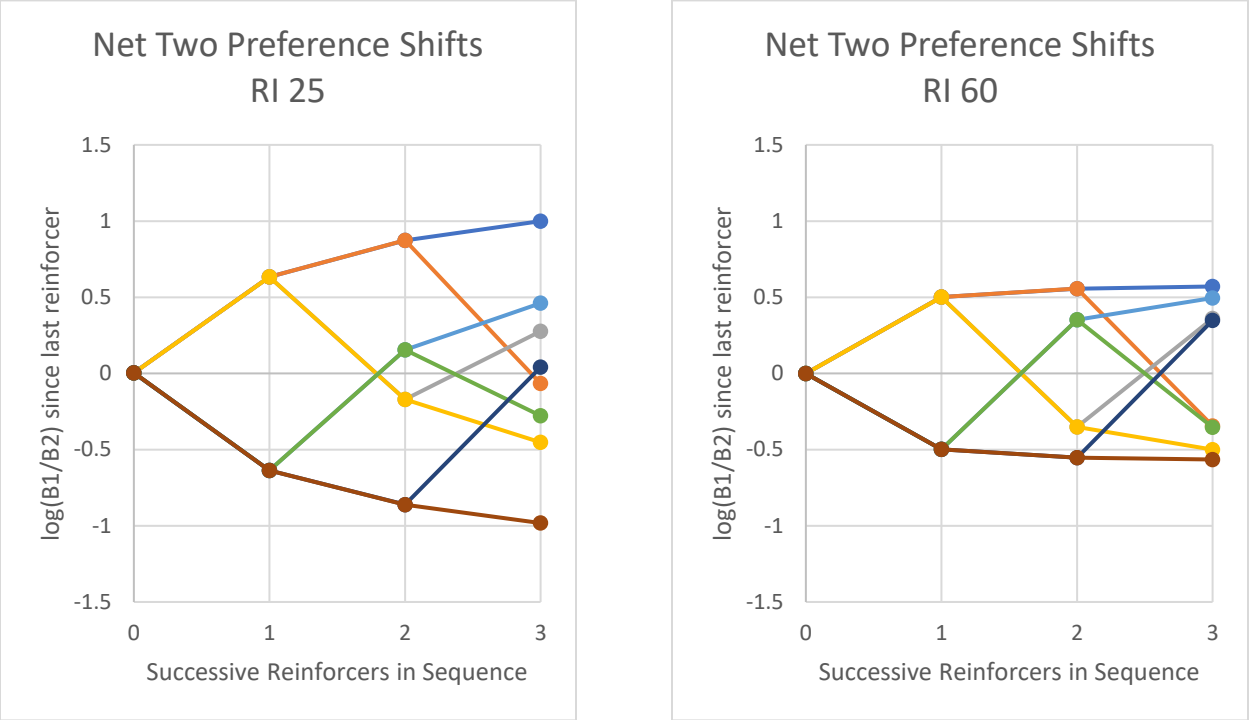


Figure 13. AO preference shifts following confirmations and disconfirmations. Branches of a color connect values of log(B1/B2) between successive reinforcers. For instance, each brown line shows preference development across a sequence of reinforcers obtained on right-right-right, and the yellow path shows the same for the sequence left-right-right.

Results Summary

The results of the two nets when evaluated against the twelve hypotheses is described in Table 4.

Table 4

Summary of Results with Respect to Hypotheses

Hypothesis	Statement	Net One	Net Two
a.	$a_r \approx .83$ for all φ[a]		
b.	$a_m \approx .68$ for all φ[a]		
c.	Effects of rate and magnitude are independent	✓	✓
d.	Preference increases with ratio of schedule densities	✓	✓
e.	Preference increases with overall reinforcement rate	✓	✓
f.	Preference on equal RR-RR increases with reinforcement rate	✓	✓
g.	$A < 0$; $C < 0$; $B^2 - 4AC < 0$; $D = 0$; $F = 0$; and $G > 0$	✓	
h.	G increases with mutation rate	✓	
i.	$\lvert A \rvert - \lvert C \rvert > 0$	✓	
j.	Preference reaches asymptote before eight reinforcers	✓	✓
k.	Denser reinforcement causes faster approach to asymptote	✓	✓
l.	Reinforcer patterns yield a characteristic interleaved pattern	✓	✓

Note: "X" indicates that there was support for the hypothesis in the behavior record of the AO.
[a] For net one and net two the ranges of values of a_r and a_m were much larger than anticipated, though the means were in the correct range.

General Discussion

The pattern of met and missed hypotheses for net one suggests a poor calibration between it and the ETBD. The relationship between mutation rate and the sensitivity exponent from the cGML is tepid in the ETBD, but sensitivity in net one is tightly dependent on mutation rate (see Figure 7). While hypotheses d, e, and f were technically met in net one, the degree of preference for a side in RR-RR schedules is markedly lower in net one that in the ETBD (see Figures 9, 10, and 11). Changeovers are two or three times as common in net one compared with the ETBD baseline (see Figure 8).

Net one and genetic algorithm A (see Appendix A) produce the same results if they begin from the same state, produce the same output, and obtain the same consequence. Genetic algorithm A is much like the ETBD, but on short timescales there may be differences in outputs. The ETBD's sequence of behavior populations is serially dependent on the previous population. These discrepancies may lead to the failed hypotheses and weak preferences found in the previous experiments.

To evaluate this idea, a *post hoc* experiment was run on net one and the ETBD to detect serial correlations in behavior records. A single reinforcer on the left target class was set up at the beginning of a trial. Once that reinforcer was acquired, the pattern of behaviors was recorded for twenty-five generations, and during those generations no other reinforcers were scheduled. This test was run 10,000 times. The output at each tick was labeled as positive one if the behavior was in the left target class, negative one if in the right target class, and zero if it was in neither. The correlation between output at $t = 0$ (i.e., the first behavior after the reinforcer was acquired) and all other time points in the sequence was calculated. Results from this experiment are shown in Figure 14. These results demonstrate a much "sticker" behavioral profile for the ETBD than for net one.

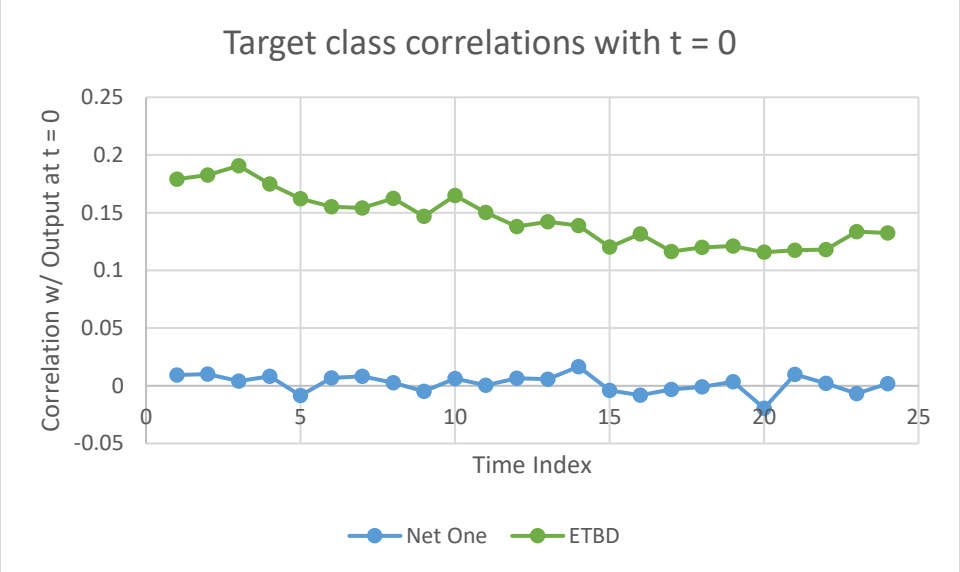


Figure 14. Target class correlations with output at $t = 0$ after a single reinforcer is acquired at time $t = -1$. The ETBD shows significant persistent positive serial correlations in the absence of reinforcement. For instance, if the behavior at time $t = 0$ was in the left target class, the chance of a behavior in that target class is increased at all subsequent times. This effect is independent of reinforcement. If the behavior at time $t = 0$ was in the right target class (which was not previously reinforced), then subsequent behaviors will be more likely to be in the right target class as well.

Genetic Drift in the ETBD

The ETBD shows significant correlations between behaviors that persist through time, even in the absence of reinforcement, while net one shows no such correlations. In other words, random signals in the present significantly affect the future behavior of the ETBD. This statement is not true for net one.

An explanation for this difference can be found by examining both algorithms in the absence of reinforcement and mutation. In net one the distribution of latent behaviors remains constant over time given no reinforcement or mutation, and so successive behaviors are random draws from an unchanging distribution. However, the distribution of behaviors in the ETBD does not stay constant given these constraints. At each time tick the ETBD generates a new population by randomly selecting and mating pairs from the current population. The expected distribution of

children at time $t + 1$ is dependent on the current population at time *t*, and the expected distribution of children at time $t + 2$ is dependent on the population at time $t + 1$, and so on. Crucially, the set of children is a sample from this expected distribution and not the distribution itself. As a result, the statistics describing the population of behaviors can fluctuate between timesteps. The distribution of children at $t + 1$ may therefore be different from the distribution at t, since the random nature of sampling may cause the new set of parents to cover a distinct set of genomes.

As a result, random sampling errors accumulate in the ETBD. While the result may be close to the same as net one for the next step, random deviations gradually move the population of behaviors in one direction or another. These deviations will persist over time, thus leading to correlated outcomes between subsequent samples. This process is analogous to genetic drift in biological populations.

This explanation leads to a testable hypothesis for the original ETBD. If the population size of the ETBD were large, then the difference between the sampled populations at each time step due to genetic drift should tend toward zero. Conversely, a smaller population should increase the difference between populations at successive timesteps. So, larger populations in the ETBD should produce outputs more like net one, with higher changeover rates and lower preference during RR-RR schedules.

Modeling Genetic Drift in Net One

If genetic drift in the ETBD has an observable effect on outputs, then net one as currently formulated is not a reasonable approximation to the ETBD, since net one contains no mechanism for reproducing the effects of genetic drift. So, it is necessary to find a way to model genetic drift

in net one. Ideally this mechanism would be consistent with a mechanism of biological neural networks while at the same time reducing the differences between the behavior records of net one and the ETBD.

A small amount of information about the population of the ETBD exists in the most recent emission. Let b_t refer to the most recently emitted behavior. After emission of b_t, it is possible to conclude at the very least that the population at time t contains b_t. More broadly, the emission of b_t implies that other behaviors like b_t are also likely to exist in the population at time. For example, the parents of b_t should share approximately half their bits with b_t, and any other children of those parents should share about a quarter of their bits with b_t.[28] Any random sampling process that caused b_t to come into existence would probably also create other behaviors with a similar genome as b_t. So, the similarities between b_t and another behavior in the population at time t should on average have more bits in common than would two random points in the genotype space. As a result, the emission of b_t indicates that the population of behaviors in the ETBD is closer to b_t on average than would be expected of a random population. So, there should be more similar behaviors to b_t in the emission sequence than there would be dissimilar behaviors, so there would be correlated results over time.

Recall that the reinforcement operator in network one contracts all behaviors toward the reinforced output. So, a reasonable approximation to the effects of genetic drift would be to

[28] Any non-shared bits would also have a 50% chance of having the same allele on average, so parent genomes would share about 75% of alleles.

unconditionally dispense a small amount of artificial reinforcement at each time tick. This small artificial reinforcer will be referred to as the Reinforce Every Behavior (REB) value.

Reinforce Every Behavior

Adding REB will reduce sensitivity to reinforcement rate.[29] To compensate it is necessary to increase the strength of obtained reinforcement, so adding REB requires recalibrating the outputs of net one. Since adding REB would require changing up to three parameters (i.e., the REB constant and the slope and intercept of the FDF mean conversion equation $\rho(\mu)$), three constraints are needed. If genetic drift in the ETBD is the main cause of differences between behavior records, then hypothetically, adding REB and increasing obtained reinforcement strength should bring the behavior records in line with each other.

Evaluating the REB Hypothesis

To test this idea, net one was tuned to match the outputs of the ETBD on a_r, a_m and changeovers at 10% mutation. The set of schedules is the same as phase one from pilot testing (see Table 2). Fitting on these criteria produces REB = 0.067 and

$$\rho(\mu) = .57 - .70 \log_{10}\left(\frac{\mu}{40}\right)$$

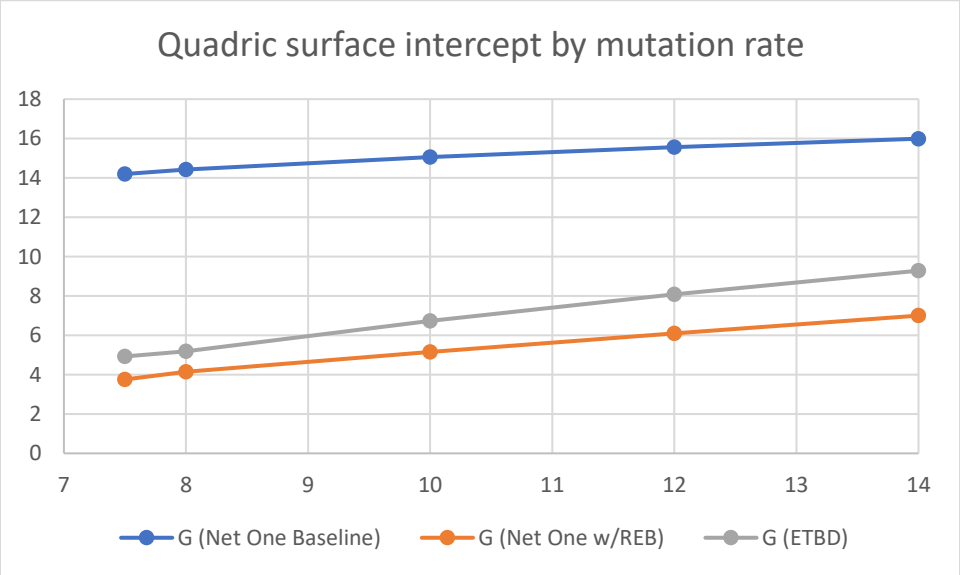
.

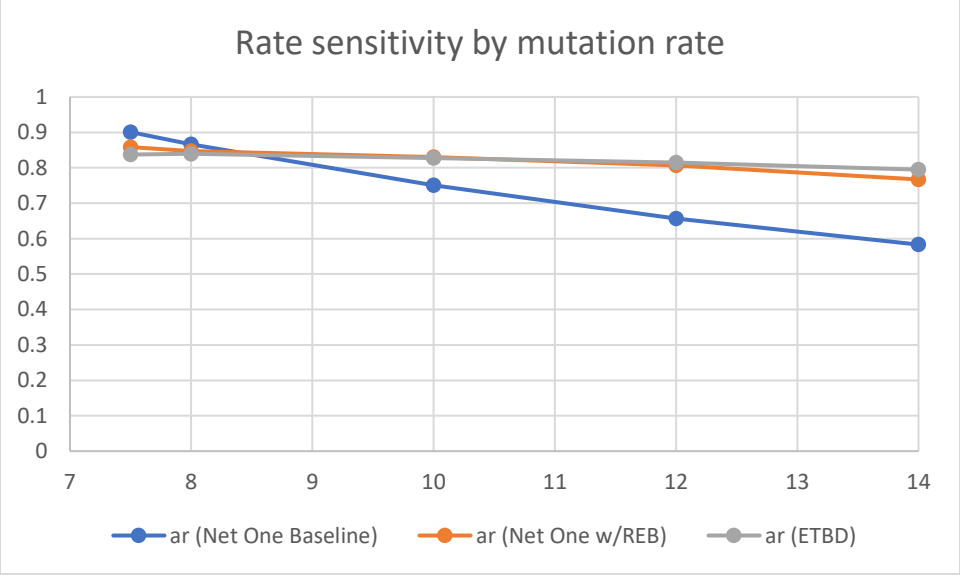
Results from this manipulation show that it causes the outputs of net one to fall crisply in line with those of the ETBD. Experiment one is repeated (see Table 2) using the retuned net one.

---

[29] Consider what would happen in the extreme case where the strength of reinforcement was the same for the REB constant and an obtained reinforcer. In this case there would be no discrimination between reinforced and unreinforced behaviors, so sensitivity to both rate and magnitude would approach zero.

See Figure 15 for the graphs of a_r, a_m, and the intercept of the quadric surface, and for the same values for the ETBD. The graphs of a_r and a_m for net one with REB and the ETBD overlap. The intercept of the quadric changeover surface is not identical to that of the ETBD, though the difference is minor compared to the difference between the ETBD and the baseline implementation of net one.





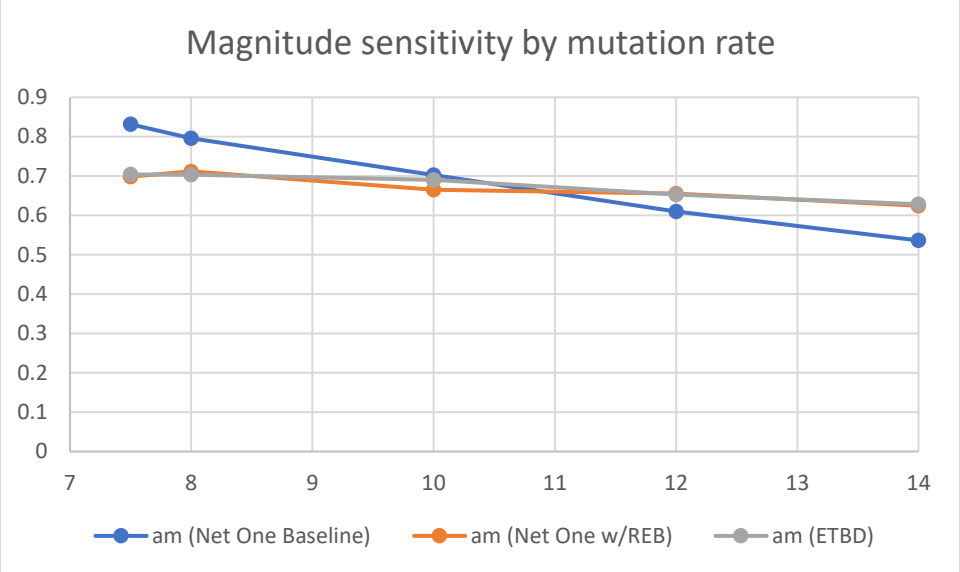


Figure 15. Outcomes from repeating experiment one and adding REB. The outputs after adding REB are much more like the ETBD in sensitivity and in changeover profiles. The REB constant was tuned to replicate the ETBD on the data points on each graph where the mutation rate is 10%.

Experiment two, phases one through three, are repeated using the same settings used for the repetition of experiment one. The outcomes from these experiments support the claim of increased similarity between net one with REB and the ETBD as compared to the baseline implementation.

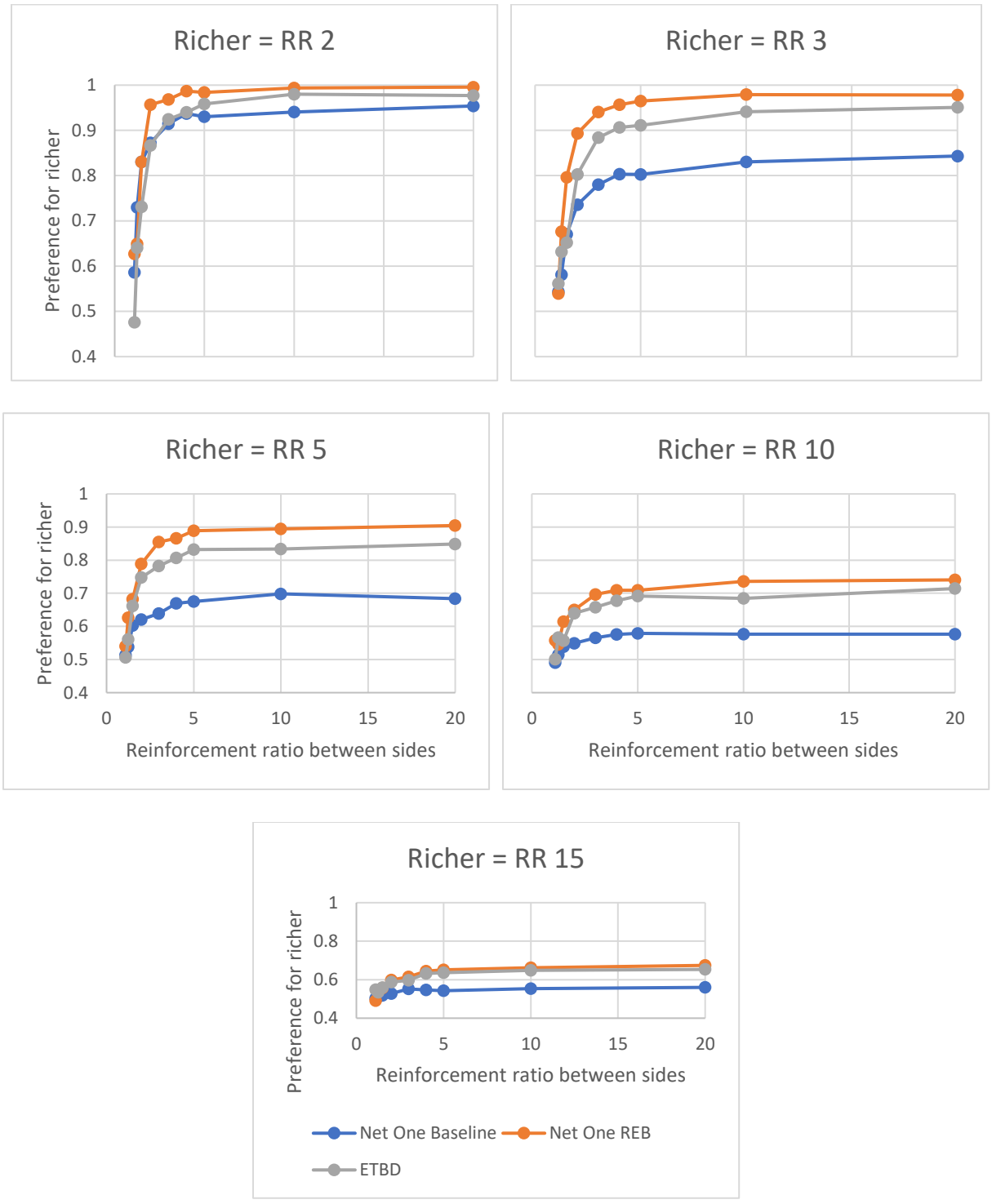


Figure 16. Outputs from repeating experiment two phase one while adding REB to net one. The x-axis on each graph is the reinforcement ratio between two RR schedules, and the y-axis is the preference for the richer schedule expressed as the fraction of target behaviors on the richer alternative. The graphs are split by the richer alternative from RR 2 to RR 15. Each data point represents five thousand generations.

See Figure 16 for graphs of the outputs from repeating experiment two phase one. Each graph shows how the three AOs prefer the richer alternative in a series of unequal concurrent RR-RR schedules. There are five graphs separated by the richer RR schedule to make it easier to compare preference. In all graphs, the orange line (net one with REB) is closer to, and the blue line (net one baseline) is farther away from, the grey line (ETBD).

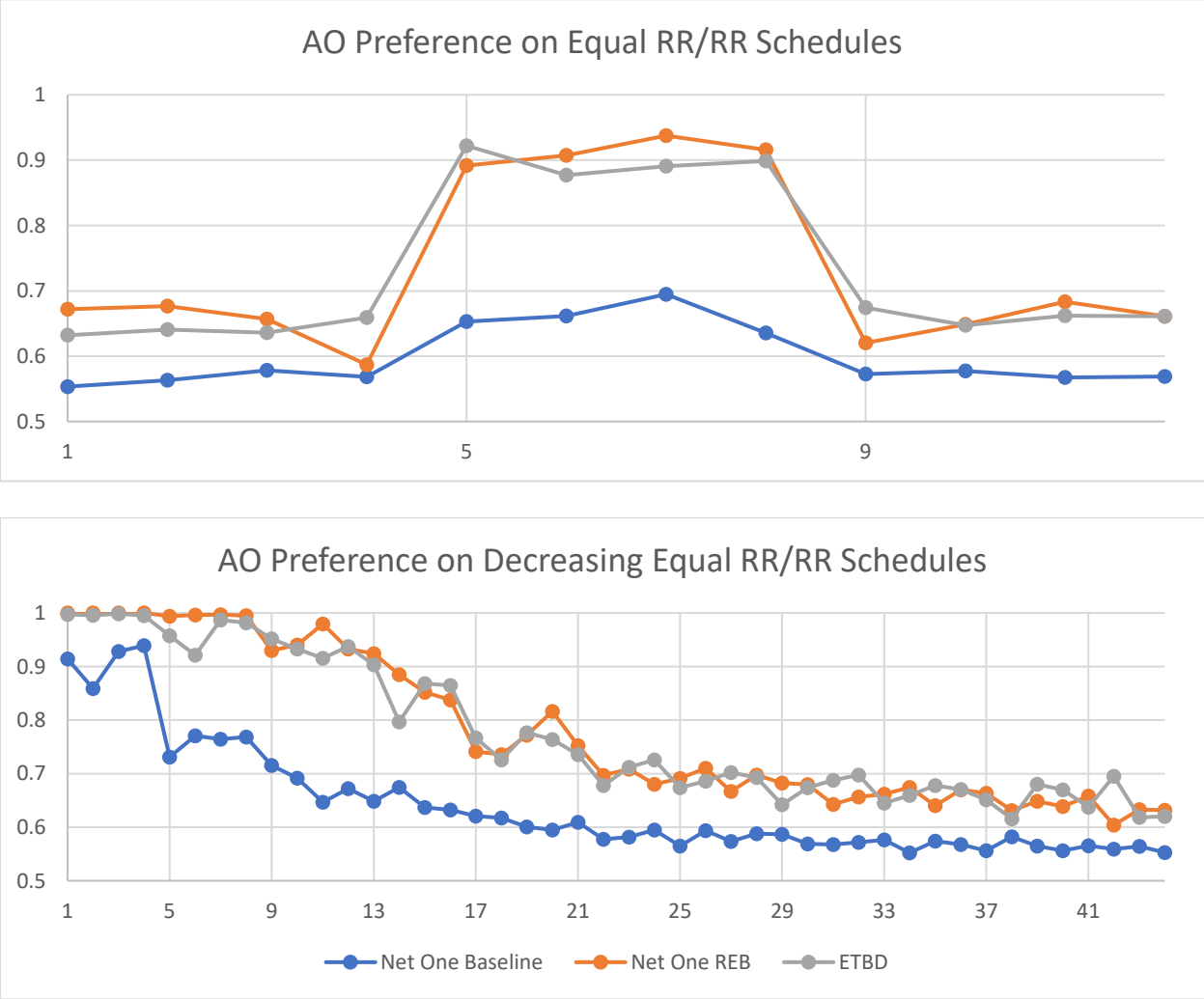


Figure 17. Outputs from repeating phases two and three from experiment two while adding REB to net one. The x-axis on each graph is the reinforcement ratio between two RR schedules, and the y-axis is the preference for the richer schedule expressed as the fraction of target behaviors on the richer alternative. Both graphs show preference for one alternative when an AO is exposed to concurrent equal RR schedules. Each schedule repeats for four data points, and each data point represents five thousand generations. Schedules on the top graph are 160/5/160, and schedules for the bottom graph are 2/3/4/5/10/20/40/80/160/320/640.

See Figure 17 for a graph of the outputs from repeating phases two and three. In both graphs, the similarities between the orange and grey lines are apparent, and the blue line sits far beneath the other two.

Discussion of REB

REB does indeed seem to bring the outputs of net one into close alignment with those of the ETBD. So, to the extent that the ETBD's behavior record replicates that of live animals, REB brings net one closer to what would be expected from live behavior. It is important to address whether this added reinforcement is *ad hoc* or can be justified on theoretical grounds.

There seems to be a biological correlate to REB, which is the "standard" STDP (recall that STDP refers to spike timing dependent plasticity) that occurs in the absence of reinforcement or punishment. The first experiments on STDP (Bi & Poo, 1998) did not incorporate any measures of neuromodulatory transmitters, nor did these experiments try to manipulate those transmitters. STDP experiments started in the unsupervised paradigm. Experiments showed that repeated stimulation of a presynaptic neuron followed by a postsynaptic neuron would lead to increased strength of the synapse connecting them regardless of consequences.

In net one, reinforcement is modeled by switching synapses between weights of positive and negative one, and in net two it is modeled as a graded change of weight. Net two's graded weight change is based on neuromodulated STDP, so it would make sense that unmodulated STDP should have a correlate in net two. This correlate would add a small amount of "reinforcement" after every action, which is exactly what REB does.

There may be a second correlate to REB in dopamine (DA) signaling. DA is a neurotransmitter that is heavily implicated in reward processing (Berridge & Robinson, 1998). In particular, phasic DA signals, which are brief spikes of DA concentration at latencies of 70-100 ms in the midbrain, are associated with presentation of rewarding stimuli (Schultz, 2015). A common interpretation of the role of dopamine is that it signals reward prediction error, which, in operant learning situations, leads to changes in the weights of synapses responsible for the behavior that preceded the reward (Bayer & Glimcher, 2005; Joel, Niv, & Ruppin, 2002; Schultz, 2015; Schultz, Dayan, & Montague, 1997). However, an alternative (but not mutually exclusive) interpretation is that phasic dopamine spikes occur when a non-aversive consequence is *caused* by a behavior; in other words, dopamine helps an agent figure out which events are under its control (Redgrave & Gurney, 2006; Redgrave, Gurney, & Reynolds, 2008). In this interpretation, phasic DA leads to the repetition of behaviors that produce changes in the environment, and successive pairings of behaviors with their caused stimuli confirm that the agent is responsible for those stimuli (Redgrave et al., 2008). This effect of DA would produce a behavioral record that appeared as if some non-reinforced behaviors had actually been reinforced. This the effect that REB provides. This “behavioral stickiness” should crop up as increased preference during equal RR-RR schedules and decreased changeover rates during RI-RI schedules, for instance; REB causes the agent to be more likely to stay where it is. Indeed, these effects emerge when REB is added (see figures 14 and 16).

To sum up, there is theoretical and empirical support to the idea that adding REB to net one would make its outputs resemble animal behavior. The theoretical support is from the similarities with STDP and phasic DA, and the empirical support is from the similarities between

its outputs and the outputs of the ETBD. However, it is unlikely that REB would have the desired effect on net two due.

Conclusions

This paper showed how the ETBD can be translated into a stochastic neural network. A simpler version of the ETBD, genetic algorithm A (see Appendix A), is procedurally identical to network one. The differences between the output patterns of network one and the ETBD were shown to be dependent on the structural differences between the ETBD and genetic algorithm A. Specifically, the resampling of the behavioral population at each tick plays a key role in behavior patterns under RR-RR and RI-RI schedules. This resampling was simulated in network one by adding a REB constant, which is a small amount of reinforcement dispensed unconditionally after every time step.

Applying REB in artificial agents might bring about more similarity between their behavior records and those of biological agents. Adding REB and reinterpreting reward as a measure of control over the environment would allow an agent to discover what it is able to accomplish with its actions. This reinterpretation would fit nicely with the options framework of hierarchical reinforcement learning (Veeriah et al., 2021), where an RL agent attempts to discover sequences of behavior patterns that produce results relevant to the agent's long term goals.

At the same time, this paper showed how a further extension with real-valued synapses and STDP-like learning rules failed to replicate the outputs of the ETBD. This failure indicates that there is a process within the ETBD that is not captured by the dynamics of network two.

Limitations

Computational and Structural Limitations

An unanswered question about nets one and two is where one might look in a nervous system to find a copy of the algorithms. The ETBD has no direct analogue to a neural process, so there is no need to look for one.[30] Networks one and two are mere caricatures of biological networks, so it is unlikely that any biological process has only the functions of the networks in this paper. More work is needed to answer this question.

Indeed, there may be more than one reasonable answer. For instance, a single cortical minicolumn has ~100 neurons (Favorov & Kelly, 1994a, 1994b), which is the same size as network one, so there may be a natural extension of network one that maps its structure and function onto a minicolumn. However, given the abstractness of what behaviors mean in these studies, it is possible that the best match is a brain region somewhere high up in the motor hierarchy. This interpretation would require each binary neuron from net one to correspond to an interconnected cluster of biological neurons, and outputs would be lower-level clusters in the hierarchy.

Another possible answer is that there is nothing in the brain that functions like networks one or two. It may be the case that the output similarities between the ETBD and those of

---

[30] However, there are researchers who describe processes like reproduction and mutation in neural networks. Fernando and colleagues ((Fernando, Goldstein, & Szathmary, 2010; Fernando, Karishma, & Szathmary, 2008; Fernando et al., 2012; Fernando, Vasas, Szathmary, & Husbands, 2011) argue for the neuronal replicator hypothesis, which describes a mechanism for how small groups of neurons copy their patterns onto other groups. Edelman (1987) argues for neural Darwinism, which states that networks compete for access to stimuli, and those that are unable to latch onto a reliable source of stimulation will be cannibalized by other more active networks.

animals exist only on a computational level and not a structural one (McDowell, 2013b). As such, the "ladder" from the ETBD to a biologically plausible network may not exist, or there may be several rungs missing. These gaps would correspond to large deviations between successive models that could not be bridged without theoretically or empirically unsound additions.

Evidentiary Limitations

To the extent that the ETBD's outputs have been shown to match up with those of live organisms, the benchmarks set in Table 1 are reasonable goals for a biological network. However, some of the findings of the ETBD have not been replicated in live animals (e.g., changeover patterns found by McDowell et al., 2012), and some of the predictions of the ETBD are more detailed or precise than what has been established by animal studies (e.g., preference during unequal RR-RR schedules found by McDowell & Klapes, 2018). If future studies refute these findings of the ETBD in live animals, this set of benchmarks will need to be updated.

Future Directions

Improving Net Two

While the gap between net one and the ETBD is narrowed by adding REB, there was no attempt in this paper to address whether a similar alteration to net two would have the same effect. When examining the outcomes of experiment two for RR values below five, net two had increased preference for richer or equal alternatives as compared with the ETBD. Hypothetically, adding REB to net two would further increase this preference, pushing the results of net two farther away from those of the ETBD. Thus, it seems unlikely that solely adding REB to net two would improve the situation. Alternate points of attack would be needed.

One such point may be found by examining how information accumulates across the network as successive reinforcement occurs. In net one, whenever there is a reinforcer, all synapses are eligible for a change, and the probability of change equals the reinforcement magnitude. So, if a reinforcer with magnitude ρ is obtained, then at the next time tick approximately 20ρ of the synapses[31] connected to the next pair of activated hidden nodes will have been altered by the reinforcer. For typical values of ρ this means around four to ten synapses active at the next time step will have been altered. However, following reinforcement in net two, only the synapses that are connected to the two activated hidden nodes are eligible for change. At the next step only an expected $40/k$ of the synapses[32] connected to activated nodes will change, where k is the total number of hidden nodes. For $k = 27$, fewer than two activated synapses at the next time step will change on average. As a result, weight changes due to reinforcement are concentrated in fewer synapses. A solution to bring net two in line with net one might be to increase the number of active hidden nodes at every time step while also reducing the strength of reinforcement at each synapse. This would create more eligible synapses and a more rapidly diffusing and less concentrated change in weights.

Another difference between nets one and two is the variability of reinforcement and mutation effects across eligible synapses. In net one each synapse has a specified probability of changing for each reinforcement or mutation. For mutation, this probability is usually 0.1%, and

---

[31] There will be two active nodes at the next step, each with ten synapses. If the fraction of changed synapses equals ρ, then there will be $2 * 10 * \rho$ changed synapses observed at the next time step on average.

[32] The fraction of all synapses changed will be $2/k$. There are twenty synapses connected to active hidden nodes at the next time step, so the average number of observed changed synapses will be $40/k$.

for reinforcement it is usually between 20% and 50%. Contrastingly, in net two each eligible synapse has a 100% probability of changing. As a result, there is more uncertainty added to the distribution of weights in net one, and it may be the case that this added uncertainty contributes to its behavioral outcomes. A way to increase the variability of weight changes in net two would be to add or multiply by a random variable as opposed to a fixed value when reinforcement or mutation occurs. An exponentially distributed random variable might work for reinforcement, and a beta distribution might work for mutation.

Beyond Nets One and Two

The project started by this paper aims to build connections from a genetic algorithm up to a realistic simulation of a biological network. Net two is a rung on a ladder that hopefully will reach between the ETBD and biologically plausible networks. After figuring out what alterations are necessary to get net two producing the same results as net one, net two will be expanded on to include structures that are more consistent with real neurons and networks. There are many constraints that might be added to net two to move to the next step in the ladder.

For instance, net three might include separate populations of inhibitory and excitatory neurons in accordance with Dale's rule (Catsigeras, 2013; Strata & Harvey, 1999). It might contain recurrent connections and act like a liquid state machine (Maass et al., 2002), making information about timing and delay available for later use. Doing so would require net three to run in continuous time, which might lead to using any of several suitable models of neurons (Izhikevich, 2004). Net three might simulate interactions between multiple networks with reentrant connections, modeling the neural darwinism of Edelman (1987). It might use populations as readouts instead of single neurons, allowing for graded responses (Valente et al.,

2021). Or, it might add storage akin to the prefrontal cortex and learn how to open and close gates to information, replicating some hypothesized functions of the basal ganglia (O'Reilly & Frank, 2006).

Each addition should be studied carefully to make sure that outputs are what might be expected. The pattern of outputs in net one deviated significantly from those of the ETBD for an unexpected reason, and the outputs of net two diverged even more. Any addition to these networks is likely to encounter unexpected deviations due to their complexity. Finding theoretically and empircally sound alterations that overcome these deviations, and doing so without overfitting to outputs, will be a challenge.

References

Adams, P. (1998). Hebb and Darwin. *J Theor Biol, 195*(4), 419-438. doi:10.1006/jtbi.1997.0620

Andersson, C. (2007). Sophisticated selectionism as a general theory of knowledge. *Biology & Philosophy, 23*(2), 229-242. doi:10.1007/s10539-007-9085-7

Aparicio, C. F. (2001). Overmatching in rats: the barrier choice paradigm. *J Exp Anal Behav, 75*(1), 93-106. doi:10.1901/jeab.2001.75-93

Bacon, F. (1878). *Novum organum*: Clarendon press.

Baum, W. M. (1974). On two types of deviation from the matching law: bias and undermatching. *J Exp Anal Behav, 22*(1), 231-242. doi:10.1901/jeab.1974.22-231

Bayer, H. M., & Glimcher, P. W. (2005). Midbrain dopamine neurons encode a quantitative reward prediction error signal. *Neuron, 47*(1), 129-141. doi:10.1016/j.neuron.2005.05.020

Berridge, K. C., & Robinson, T. E. (1998). What is the role of dopamine in reward: hedonic impact, reward learning, or incentive salience? *Brain Research Reviews, 28*(3), 309-369. doi:10.1016/s0165-0173(98)00019-8

Bi, G. Q., & Poo, M. M. (1998). Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type. *J Neurosci, 18*(24), 10464-10472. doi:10.1523/JNEUROSCI.18-24-10464.1998

Börgers, T., & Sarin, R. (1997). Learning Through Reinforcement and Replicator Dynamics. *Journal of Economic Theory, 77*(1), 1-14. doi:10.1006/jeth.1997.2319

Brown, N., & Sandholm, T. (2019). Superhuman AI for multiplayer poker. *Science, 365*(6456), 885-890. doi:10.1126/science.aay2400

Cadieu, C. F., Hong, H., Yamins, D. L., Pinto, N., Ardila, D., Solomon, E. A., . . . DiCarlo, J. J. (2014). Deep neural networks rival the representation of primate IT cortex for core visual object recognition. *PLoS Comput Biol, 10*(12), e1003963. doi:10.1371/journal.pcbi.1003963

Calvin, O. L. (2019). *Clonal Amplification of Behavior: A Simple Interpretation of the Effect of Reinforcement.* (Unpublished doctoral dissertation, Emory University, Atlanta, GA).

Catania, A. C. (1978). The Psychology of Learning: Some Lessons from the Darwinian Revolution. *Annals of the New York Academy of Sciences, 309*(1 Psychology), 18-28. doi:10.1111/j.1749-6632.1978.tb29439.x

Catania, A. C., & Cutts, D. (1963). Experimental control of superstitious responding inhumans. *J Exp Anal Behav, 6*, 203-208. doi:10.1901/jeab.1963.6-203

Catsigeras, E. (2013). Dale's Principle Is Necessary for an Optimal Neuronal Network's Dynamics. *Applied Mathematics, 04*(10), 15-29. doi:10.4236/am.2013.410A2002

Claesen, M., & De Moor, B. (2015). Hyperparameter search in machine learning. *arXiv preprint arXiv:1502.02127*.

Cording, J. R., McLean, A. P., & Grace, R. C. (2011). Testing the linearity and independence assumptions of the generalized matching law for reinforcer magnitude: a residual meta-analysis. *Behav Processes, 87*(1), 64-70. doi:10.1016/j.beproc.2011.02.011

Costa, R. M. (2011). A selectionist account of de novo action learning. *Curr Opin Neurobiol, 21*(4), 579-586. doi:10.1016/j.conb.2011.05.004

Crick, F. (1989). The recent excitement about neural networks. *Nature, 337*(6203), 129-132. doi:10.1038/337129a0

Davison, M., & Baum, W. M. (2000). Choice in a variable environment: every reinforcer counts. *J Exp Anal Behav, 74*(1), 1-24. doi:10.1901/jeab.2000.74-1

Davison, M., & McCarthy, D. (2016). *The matching law: A research review*: Routledge.

Donahoe, J. W., Burgos, J. E., & Palmer, D. C. (1993). A selectionist approach to reinforcement. *J Exp Anal Behav, 60*(1), 17-40. doi:10.1901/jeab.1993.60-17

Edelman, G. M. (1987). *Neural Darwinism: The theory of neuronal group selection*: Basic books.

Even-Dar, E., & Mansour, Y. (2003). Learning rates for Q-learning. *Journal of machine learning Research, 5*(Dec), 1-25. doi:http://dx.doi.org/10.1007/3-540-44581-1_39

Favorov, O. V., & Kelly, D. G. (1994a). Minicolumnar organization within somatosensory cortical segregates: I. Development of afferent connections. *Cereb Cortex, 4*(4), 408-427. doi:10.1093/cercor/4.4.408

Favorov, O. V., & Kelly, D. G. (1994b). Minicolumnar organization within somatosensory cortical segregates: II. Emergent functional properties. *Cereb Cortex, 4*(4), 428-442. doi:10.1093/cercor/4.4.428

Fernando, C., Goldstein, R., & Szathmary, E. (2010). The neuronal replicator hypothesis. *Neural Comput, 22*(11), 2809-2857. doi:10.1162/NECO_a_00031

Fernando, C., Karishma, K. K., & Szathmary, E. (2008). Copying and evolution of neuronal topology. *PLoS One, 3*(11), e3775. doi:10.1371/journal.pone.0003775

Fernando, C., Szathmary, E., & Husbands, P. (2012). Selectionist and evolutionary approaches to brain function: a critical appraisal. *Front Comput Neurosci, 6*, 24. doi:10.3389/fncom.2012.00024

Fernando, C., Vasas, V., Szathmary, E., & Husbands, P. (2011). Evolvable neuronal paths: a novel basis for information and search in the brain. *PLoS One, 6*(8), e23534. doi:10.1371/journal.pone.0023534

Fiete, I. R., & Seung, H. S. (2006). Gradient learning in spiking neural networks by dynamic perturbation of conductances. *Phys Rev Lett, 97*(4), 048104. doi:10.1103/PhysRevLett.97.048104

Gehrz, R. D., Black, D. C., & Solomon, P. M. (1984). The formation of stellar systems from interstellar molecular clouds. *Science, 224*(4651), 823-830. doi:10.1126/science.224.4651.823

Gerstner, W., Lehmann, M., Liakoni, V., Corneil, D., & Brea, J. (2018). Eligibility Traces and Plasticity on Behavioral Time Scales: Experimental Support of NeoHebbian Three-Factor Learning Rules. *Front Neural Circuits, 12*, 53. doi:10.3389/fncir.2018.00053

Ghosh, D., Rahme, J., Kumar, A., Zhang, A., Adams, R. P., & Levine, S. (2021). Why Generalization in RL is Difficult: Epistemic POMDPs and Implicit Partial Observability. *Advances in Neural Information Processing Systems, 34*.

Glenn, S. S., & Field, D. P. (1994). Functions of the environment in behavioral evolution. *Behav Anal, 17*(2), 241-259. doi:10.1007/BF03392674

Glenn, S. S., & Madden, G. J. (1995). Units of interaction, evolution, and replication: organic and behavioral parallels. *Behav Anal, 18*(2), 237-251. doi:10.1007/BF03392711

Hanson, H. M. (1959). Effects of discrimination training on stimulus generalization. *J Exp Psychol, 58*(5), 321-334. doi:10.1037/h0042606

Herrnstein, R. J. (1961). Relative and absolute strength of response as a function of frequency of reinforcement. *J Exp Anal Behav, 4*, 267-272. doi:10.1901/jeab.1961.4-267

Hodgkin, A. L., & Huxley, A. F. (1952). A quantitative description of membrane current and its application to conduction and excitation in nerve. *J Physiol, 117*(4), 500-544. doi:10.1113/jphysiol.1952.sp004764

Holland, J. H. (1992). *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*: MIT press.

Horner, J. M., & Staddon, J. E. (1987). Probabilistic choice: A simple invariance. *Behav Processes, 15*(1), 59-92. doi:10.1016/0376-6357(87)90034-9

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks, 2*(5), 359-366. doi:10.1016/0893-6080(89)90020-8

Hull, D. L., Langman, R. E., & Glenn, S. S. (2001). A general account of selection: biology, immunology, and behavior. *Behav Brain Sci, 24*(3), 511-528; discussion 528-573. doi:10.1017/S0140525X01004162

Izhikevich, E. M. (2004). Which model to use for cortical spiking neurons? *IEEE Trans Neural Netw, 15*(5), 1063-1070. doi:10.1109/TNN.2004.832719

Joel, D., Niv, Y., & Ruppin, E. (2002). Actor-critic models of the basal ganglia: new anatomical and computational perspectives. *Neural Netw, 15*(4-6), 535-547. doi:10.1016/s0893-6080(02)00047-3

Killeen, P. R. (2001). The Four Causes of Behavior. *Curr Dir Psychol Sci, 10*(4), 136-140. doi:10.1111/1467-8721.00134

Knudsen, T. r. (2004). General selection theory and economic evolution: The Price equation and the replicator/interactor distinction. *Journal of Economic Methodology, 11*(2), 147-173. doi:10.1080/1350178041000169410 9

Kollins, S. H., Newland, M. C., & Critchfield, T. S. (1997). Human sensitivity to reinforcement in operant choice: How much do consequences matter? *Psychon Bull Rev, 4*(2), 208-220. doi:10.3758/BF03209395

Kubanek, J. (2017). Optimal decision making and matching are tied through diminishing returns. *Proc Natl Acad Sci U S A, 114*(32), 8499-8504. doi:10.1073/pnas.1703440114

Kulubekova, S., & McDowell, J. J. (2013). Computational model of selection by consequences: patterns of preference change on concurrent schedules. *J Exp Anal Behav, 100*(2), 147-164. doi:10.1002/jeab.40

LeCun, Y., & Bengio, Y. (1995). Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks, 3361*(10), 1995. Retrieved from https://www.researchgate.net/profile/Yann_Lecun/publication/2453996_Convolutional_Networks_for_Images_Speech_and_Time-Series/links/0deec519dfa2325502000000.pdf

Li, D., Elliffe, D., & Hautus, M. J. (2018). A multivariate assessment of the rapidly changing procedure with McDowell's Evolutionary Theory of Behavior Dynamics. *J Exp Anal Behav, 110*(3), 336-365. doi:10.1002/jeab.478

Little, W. (1974). The Existence of Persistent States in the Brain. *Mathematical Biosciences, 19*, 101-120. doi:10.1007/978-1-4613-0411-1_12

Maass, W., Natschlager, T., & Markram, H. (2002). Real-time computing without stable states: a new framework for neural computation based on perturbations. *Neural Comput, 14*(11), 2531-2560. doi:10.1162/089976602760407955

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics, 5*(4), 115-133. doi:10.1007/bf02478259

McDowell, J. J. (2004). A computational model of selection by consequences. *J Exp Anal Behav, 81*(3), 297-317. doi:10.1901/jeab.2004.81-297

McDowell, J. J. (2013a). On the theoretical and empirical status of the matching law and matching theory. *Psychol Bull, 139*(5), 1000-1028. doi:10.1037/a0029924

McDowell, J. J. (2013b). Representations of Complexity: How Nature Appears in Our Theories. *Behav Anal, 36*(2), 345-359. doi:10.1007/BF03392319

McDowell, J. J. (2019). On the current status of the evolutionary theory of behavior dynamics. *J Exp Anal Behav, 111*(1), 130-145. doi:10.1002/jeab.495

McDowell, J. J., & Caron, M. L. (2007). Undermatching is an emergent property of selection by consequences. *Behav Processes, 75*(2), 97-106. doi:10.1016/j.beproc.2007.02.017

McDowell, J. J., & Klapes, B. (2018). An evolutionary theory of behavior dynamics applied to concurrent ratio schedules. *J Exp Anal Behav, 110*(3), 323-335. doi:10.1002/jeab.468

McDowell, J. J., & Klapes, B. (2019). An implementation of punishment in the evolutionary theory of behavior dynamics. *J Exp Anal Behav, 112*(2), 128-143. doi:10.1002/jeab.543

McDowell, J. J., Popa, A., & Calvin, N. T. (2012). Selection dynamics in joint matching to rate and magnitude of reinforcement. *J Exp Anal Behav, 98*(2), 199-212. doi:10.1901/jeab.2012.98-199

Minsky, M. (1961). Steps toward Artificial Intelligence. *Proceedings of the IRE, 49*(1), 8-30. doi:10.1109/jrproc.1961.287775

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., . . . Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature, 518*(7540), 529-533. doi:10.1038/nature14236

O'Reilly, R. C., & Frank, M. J. (2006). Making working memory work: a computational model of learning in the prefrontal cortex and basal ganglia. *Neural Comput, 18*(2), 283-328. doi:10.1162/089976606775093909

Popa, A., & McDowell, J. J. (2010). The effect of Hamming distances in a computational model of selection by consequences. *Behav Processes, 84*(1), 428-434. doi:10.1016/j.beproc.2010.02.002

Pouget, A., Dayan, P., & Zemel, R. (2000). Information processing with population codes. *Nat Rev Neurosci, 1*(2), 125-132. doi:10.1038/35039062

Price, G. R. (1970). Selection and covariance. *Nature, 227*(5257), 520-521. doi:10.1038/227520a0

Price, G. R. (1972). Fisher's 'fundamental theorem' made clear. *Ann Hum Genet, 36*(2), 129-140. doi:10.1111/j.1469-1809.1972.tb00764.x

Pringle, J. W. S. (1951). On the Parallel Between Learning and Evolution. *Behaviour, 3*(1), 174-214. doi:10.1163/156853951x00269

Puterman, M. L. (2014). *Markov decision processes: discrete stochastic dynamic programming*: John Wiley & Sons.

Redgrave, P., & Gurney, K. (2006). The short-latency dopamine signal: a role in discovering novel actions? *Nat Rev Neurosci, 7*(12), 967-975. doi:10.1038/nrn2022

Redgrave, P., Gurney, K., & Reynolds, J. (2008). What is reinforced by phasic dopamine signals? *Brain Res Rev, 58*(2), 322-339. doi:10.1016/j.brainresrev.2007.10.007

Reid, C. R., MacDonald, H., Mann, R. P., Marshall, J. A., Latty, T., & Garnier, S. (2016). Decision-making without a brain: how an amoeboid organism solves the two-armed bandit. *J R Soc Interface, 13*(119), 20160030. doi:10.1098/rsif.2016.0030

Ringen, J. D. (1993). Adaptation, teleology, and selection by consequences. *J Exp Anal Behav, 60*(1), 3-15. doi:10.1901/jeab.1993.60-3

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1985). Learning internal representations by error propagation. Retrieved from https://apps.dtic.mil/dtic/tr/fulltext/u2/a164453.pdf

Schmidhuber, J. (2000). Evolutionary computation versus reinforcement learning. *2000 26th Annual Conference of the IEEE Industrial Electronics Society. IECON 2000. 2000 IEEE International Conference on Industrial Electronics, Control and Instrumentation. 21st Century Technologies and Industrial Opportunities (Cat. No.00CH37141)*, 2992-2997. doi:10.1109/iecon.2000.972474

Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., . . . Silver, D. (2020). Mastering Atari, Go, chess and shogi by planning with a learned model. *Nature, 588*(7839), 604-609. doi:10.1038/s41586-020-03051-4

Schultz, W. (2015). Neuronal Reward and Decision Signals: From Theories to Data. *Physiol Rev, 95*(3), 853-951. doi:10.1152/physrev.00023.2014

Schultz, W., Dayan, P., & Montague, P. R. (1997). A neural substrate of prediction and reward. *Science, 275*(5306), 1593-1599. doi:10.1126/science.275.5306.1593

Seung, H. S. (2003). Learning in spiking neural networks by reinforcement of stochastic synaptic transmission. *Neuron, 40*(6), 1063-1073. doi:10.1016/s0896-6273(03)00761-x

Shah, K., Bradshaw, C. M., & Szabadi, E. (2016). Performance of Humans in Concurrent Variable-Ratio Variable-Ratio Schedules of Monetary Reinforcement. *Psychological Reports, 65*(2), 515-520. doi:10.2466/pr0.1989.65.2.515

Silvio, L. (1995, October 4). The Geometry Center. *Quadrics.* Retrieved from http://www.geom.uiuc.edu/docs/reference/CRC-formulas/node61.html

Skinner, B. F. (1981). Selection by consequences. *Science, 213*(4507), 501-504. doi:10.1126/science.7244649

Smith, T. L. (1983). Skinner's environmentalism: The analogy with natural selection. *Behaviorism, 11*(2), 133-153. doi:10.2307/27759024

Strata, P., & Harvey, R. (1999). Dale's principle. *Brain Res Bull, 50*(5-6), 349-350. doi:10.1016/s0361-9230(99)00100-8

Stubbs, D. A., & Pliskoff, S. S. (1969). Concurrent responding with fixed relative rate of reinforcement. *J Exp Anal Behav, 12*(6), 887-895. doi:10.1901/jeab.1969.12-887

Sutton, N. P., Grace, R. C., McLean, A. P., & Baum, W. M. (2008). Comparing the generalized matching law and contingency discriminability model as accounts of concurrent schedule performance using residual meta-analysis. *Behav Processes, 78*(2), 224-230. doi:10.1016/j.beproc.2008.02.012

Szilagyi, A., Zachar, I., Fedor, A., de Vladar, H. P., & Szathmary, E. (2016). Breeding novel solutions in the brain: a model of Darwinian neurodynamics. *F1000Res, 5*, 2416. doi:10.12688/f1000research.9630.2

Thompson, R. S., & Gibson, W. G. (1981a). Neural model with probabilistic firing behavior. I. general considerations. *Mathematical Biosciences, 56*(3-4), 239-253. doi:10.1016/0025-5564(81)90056-0

Thompson, R. S., & Gibson, W. G. (1981b). Neural model with probabilistic firing behavior. II. One- and two-neuron networks. *Mathematical Biosciences, 56*(3-4), 255-285. doi:10.1016/0025-5564(81)90057-2

Thorndike, E. L. (1998). Animal intelligence: An experimental study of the associate processes in animals. *American Psychologist, 53*(10), 1125-1127. doi:10.1037/0003-066x.53.10.1125

Ukita, J., Yoshida, T., & Ohki, K. (2019). Characterisation of nonlinear receptive fields of visual neurons by convolutional neural network. *Sci Rep, 9*(1), 3791. doi:10.1038/s41598-019-40535-4

Urbanczik, R., & Senn, W. (2009). Reinforcement learning in populations of spiking neurons. *Nat Neurosci, 12*(3), 250-252. doi:10.1038/nn.2264

Urcuioli, P. J. (2013). Stimulus control and stimulus class formation. In G. J. Madden, W. V. Dube, T. D. Hackenberg, G. P. Hanley, & K. A. Lattal (Eds.), *APA handbooks in psychology®. APA handbook of behavior analysis, Vol. 1. Methods and principles*. doi:10.1037/13937-016

Valente, M., Pica, G., Bondanelli, G., Moroni, M., Runyan, C. A., Morcos, A. S., . . . Panzeri, S. (2021). Correlations enhance the behavioral readout of neural population activity in association cortex. *Nat Neurosci, 24*(7), 975-986. doi:10.1038/s41593-021-00845-1

Veeriah, V., Zahavy, T., Hessel, M., Xu, Z., Oh, J., Kemaev, I., . . . Singh, S. (2021). Discovery of Options via Meta-Learned Subgoals. *arXiv preprint arXiv:2102.06741*.

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., . . . Silver, D. (2019). Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature, 575*(7782), 350-354. doi:10.1038/s41586-019-1724-z

Wasserman, E. A. (2012). Species, Tepees, Scotties, and Jockeys: Selected by Consequences. *Journal of the experimental analysis of behavior, 98*(2), 213-226. doi:10.1901/jeab.2012.98-213

Watkins, C. J. C. H. (1989). *Learning from delayed rewards.* (Unpublished doctoral dissertation). University of Cambridge, England. Retrieved from https://www.researchgate.net/profile/Christopher_Watkins2/publication/33784417_Learning_From_Delayed_Rewards/links/53fe12e10cf21edafd142e03/Learning-From-Delayed-Rewards.pdf

Watkins, C. J. C. H., & Dayan, P. (1992). Q-learning. *Machine learning, 8*(3-4), 279-292. doi:10.1007/bf00992698

Whitley, D. (1994). A genetic algorithm tutorial. *Statistics and computing, 4*(2), 65-85. doi:10.1007/bf00175354

Xiao, M. Y., Niu, Y. P., & Wigstrom, H. (1996). Activity-dependent decay of early LTP revealed by dual EPSP recording in hippocampal slices from young rats. *Eur J Neurosci, 8*(9), 1916-1923. doi:10.1111/j.1460-9568.1996.tb01335.x

Yamins, D. L., Hong, H., Cadieu, C. F., Solomon, E. A., Seibert, D., & DiCarlo, J. J. (2014). Performance-optimized hierarchical models predict neural responses in higher visual cortex. *Proc Natl Acad Sci U S A, 111*(23), 8619-8624. doi:10.1073/pnas.1403112111

Appendix A

Definition of Network One

A set of synapses fully connect ten output neurons to k hidden neurons (See Figure 18). Unless otherwise specified, k is one hundred. Output neurons have fixed indices from one to ten that decide the readout order. Synapse weights can only be negative or positive one. Upon initialization of the network, each synapse has a randomly selected weight. There are no other connections between neurons, and there are no inputs to the network.

All neurons have binary states that update at each time step. The value one corresponds to "on" and zero corresponds to "off." Output depends on current input, and the output at the current time does not depend on the outputs or inputs at earlier times.

At each time step, exactly two hidden neurons fire, chosen uniformly without replacement from the pool of hidden neurons. Each output neuron sums the incoming synapse weights from only the fired hidden neurons. If the sum is positive, the output neuron fires. If negative, it does not fire. If zero, it has a 50% chance of firing. The output of network one is the sequence of ones and zeroes corresponding to the ordered sequence of on/off states of output neurons. This sequence is passed to an environment that determines the consequences of the output.

If the emitted sequence obtains a reinforcer from the environment, then *all* synapses connecting hidden and output neurons undergo a reinforcement rule:

- If the output neuron fired, then the synapse weight becomes positive one with probability ρ and is unchanged otherwise.

- If the output neuron did not fire, then the synapse weight becomes negative one with probability ρ and is unchanged otherwise.

A mutation rule is then applied to all synapses. With probability $\varphi/10$, a synapse will change sign, otherwise it stays the same.[33]

In summary, the sequence of operations at each time step is as follows: fire two random hidden neurons → fire output neurons → generate behavior from output bits → query environment → apply reinforcement rule → apply mutation rule.

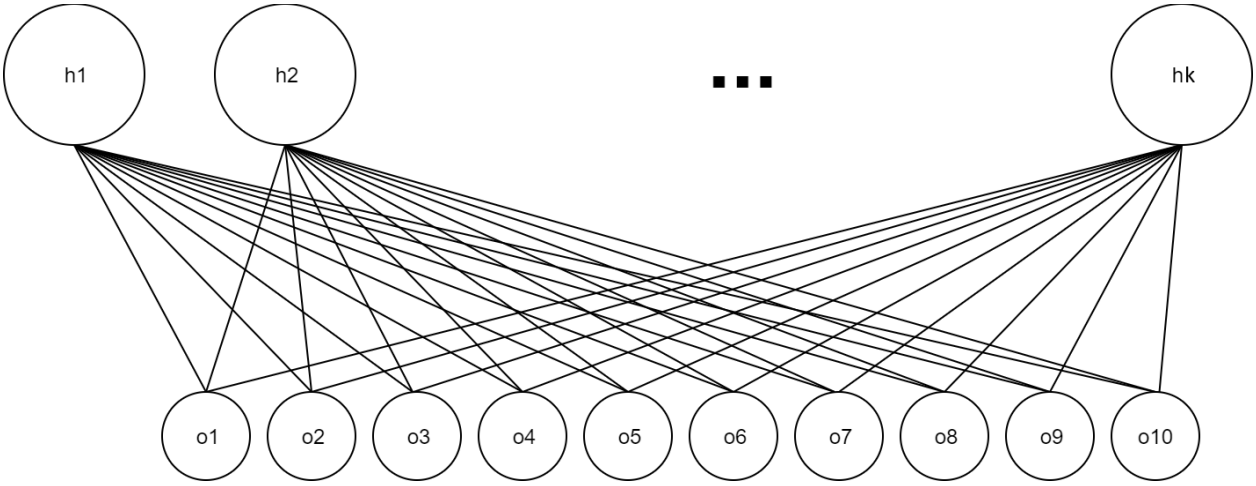


Figure 18. Structure of network one. There are k *hidden neurons fully connected to ten output neurons.*

Genetic Algorithm A

This algorithm is identical to the operations of network one.

1. Randomly initialize a population of one hundred bitstrings, each of length ten.

[33] Division by ten is necessary since in the ETBD the mutation rate is per behavior, not per synapse.

2. Mate two behaviors using universal crossover and emit the offspring.
3. If the emitted behavior *e* obtains a reinforcer of magnitude ρ, apply the following rule to every behavior *b* in the population: for every bit in the genome of *b* that differs from the corresponding bit in *e*, flip the bit in *b* with probability ρ.
4. Mutate each bit of each behavior in the population with probability $\varphi/10$.
5. Repeat steps 2-4 until a prespecified condition is true.

Relevant Differences with the ETBD

Selection rules are different between this genetic algorithm and the ETBD. When there is a reinforcer after a behavior emitted at location e, then the likelihood of behavior b becoming a parent in the ETBD is inversely proportional to phenotype distance, which is $\min(|b - e|, 1024 - |b - e|)$. In genetic algorithm A, the reinforced behavior is one of the parents for every pair. Each behavior in the population becomes a parent of exactly one child, but the percent of bits contributed depends on the magnitude of reinforcement. Stronger reinforcers cause the reinforced behavior to push more of its bits into the next generation.

In the ETBD, when there is no reinforcement, the set of behaviors will still update. Populations are resampled from an implied distribution over a set of hyperplanes. The implied distribution will vary at each tick due to genetic drift. Unexpectedly, this difference caused an observable effect in behavior outputs.

Definition of Network Two

Network two has the same topology as network one (see Figure 18) but k is free to vary. Synapse weights can be any real value and are initially set to zero. At every time step, two neurons chosen uniformly at random from the pool of hidden neurons turn on. Each output

neuron sums the incoming synapse weights from the active hidden neurons. The probability that an output neuron fires is

$$p(y = 1|x) = \frac{1}{1 + e^{-4x}}$$

, where x is the sum of synapse weights from the active hidden neurons. The output neurons are read in their specified order, “on” corresponding to “one” and “off” corresponding to “zero”, which creates a ten-digit binary number. This number is emitted into a user-defined environment.

If the emitted behavior obtains a reinforcer of nominal magnitude m, all synapses from the active hidden neurons undergo the following reinforcement rules: if the output neuron fired, then increase the synapse weight by a positive value $m * \theta$; if the output neuron did not fire, decrease the synapse weight by $m * \theta$. Synapses from silent hidden neurons do not change. For the mutation step, multiply the weight of all synapses by

$$1 - \frac{\varphi}{5}$$

, where φ is the equivalent mutation rate of the ETBD.

Appendix B

Derivation of Mutation Rule for Network Two

Derivation of this rule consists of solving two problems about an ample collection of bits that flip with a fixed probability at each time tick. The first problem is solving for the expected proportion of ones as a function of the starting proportion, the chance of flipping, and the number of elapsed steps. The second problem is how to track this sequence of proportions with a "mutation" rule for input weights. An approximation using multiplicative decay causes outputs to track the sequence of expected proportions.

Problem One: Bits in a Mutating Population

Initialize a population of j bits such that the proportion of ones in this population is equal to k. Bits in the population undergo a sequence of mutations. At each time point, every bit independently flips one → zero or zero → one with probability r. Let the sequence of random variables X_i equal the proportion of ones in the population after i mutations. Let the sequence of values a_i equal the expectations of these random variables, $a_i = E(X_i)$. By assumption, $a_0 = k$. The desired value is a closed-form solution for a_i.

After i mutations, there are $j * X_i$ bits equal to one and $j * (1 - X_i)$ bits equal to zero. Group size multiplied by r gives the expected number of bits that mutate out of the group. Group size times $1 - r$ gives the expected number of bits left untouched. These values are used to calculate the expected number of bits with given values before and after a mutation (see Figure 19).

		Bit Before Mutation	
		One	Zero
Bit After Mutation	One	$j * X_i * (1 - r)$	$j * (1 - X_i) * r$
	Zero	$j * X_i * r$	$j * (1 - X_i) * (1 - r)$

Figure 19. Expected count of bits in the population with a given value before and after the mutation step.

Summing the top row solves for the expected count of ones at time $i + 1$:

$$j * E(X_{i+1}) = j * X_i * (1 - r) + j * (1 - X_i) * r.$$

Rearranging terms and dividing by j gives

$$E(X_{i+1}) = X_i - 2X_i r + r.$$

Apply the expectation operator to both sides of the equation. Since the expectation operator is linear and idempotent (i.e., $E\big(E(X)\big) = E(X)$),

$$E(X_{i+1}) = E(X_i) * (1 - 2r) + r.$$

By the earlier definitions

$$a_{i+1} = a_i * (1 - 2r) + r$$

with initial condition

$$a_0 = k.$$

This gives a linear recurrence relation. The closed-form solution of this recurrence is:

$$a_i = .5 + (k - .5) * (1 - 2r)^i.$$

This equation simplifies after performing a change of variables. Define the excess to be the expected proportion of ones more than .5; the excess z_i at time i is $z_i = a_i - .5$. After the variable change, the recurrence becomes

$$z_{i+1} = z_i * (1 - 2r)$$

with initial condition

$$z_0 = k - .5,$$

and the closed-form solution becomes

$$z_i = z_0 * (1 - 2r)^i.$$

So, mutating the population with probability r will multiply the excess by $1 - 2r$, and the sequence of excesses decays geometrically.

The mutation rate per bit, r, is different from the ETBD's mutation rate φ. Each behavior in the ETBD consists of ten bits and mutation only affects one bit per mutated behavior. So, the bitwise mutation rate is

$$r = \frac{\varphi}{10}$$

. The multiplication factor on the excess is therefore

$$1 - \frac{\varphi}{5}$$

at each time step:

$$z_i = z_0 * \left(1 - \frac{\varphi}{5}\right)^i$$

.

Problem Two: Tracking the Excess in the Synapse Weights

A value drawn from a population after i rounds of mutation will equal one with expected probability a_i, defined previously. Network two requires a weight update rule such that outputs are like this sequence of population draws. For simplicity, start with the case where there is one constantly active hidden neuron and only one output neuron. Refer to the sequence of synapse weights by w_i. So, w_i should update such that "on" probabilities, conditioned on the sum of weights equaling w_i, approximate a_i: $p(y = 1|x = w_i) \approx a_i$.

The transfer function for network two is

$$p(y = 1|x) = \frac{1}{1 + e^{-4x}}.$$

Note that

$$\frac{dp}{dx} = 1$$

when $x = 0$, and $p(y = 1|x = 0) = .5$. So, the first-degree Maclaurin polynomial of p is $x + .5$. In other words, in a sufficiently small neighborhood around $x = 0$, p is well approximated by the function $x + .5$. Approximate p by its Maclaurin expansion and set x to a value w_i that is sufficiently close to zero. These substitutions simplify the problem to finding $w_i + 5 = a_i$. Thus, w_i should equal the excess defined in problem one. This yields the following rule: *if the input to the transfer function is the excess, then the output approximates the expected proportion of ones.*

So, to obtain outputs that approximate the sequence of expected proportions of ones, then inputs should be the sequence of excesses that correspond to those proportions. From problem one, the excess recurrence is:

$$z_{i+1} = z_i * (1 - \frac{\varphi}{5})$$

. So, viewing the last weight w_i as the excess in a population at time i, then multiplying w_i by

$$1 - \frac{\varphi}{5}$$

gives the excess at the next step. Continuing this process, the outputs will approximate the sequence of proportions of ones in the population. Therefore, multiplying the weight by

$$1 - \frac{\varphi}{5}$$

at each time step gives outputs that look like draws from a mutating population.

Note that this equivalence is not dependent upon the number of weights in the summation. Since multiplication distributes over addition, there is no difference in the outcome between a single weight of w or multiple weights that add to w. So, this rule may apply to all weights simultaneously with the same effect.

Therefore, the mutation rule is to multiply all weights by

$$1 - \frac{\varphi}{5}$$

at each time step.

Discussion

This approximation will become less exact the farther the input is from zero. The derivative of p is maximal when $x = 0$, so the tendency will be for outputs to have less sensitivity to changes in input when the input is far from zero. One way to combat this decay in sensitivity is to increase the weight decay when inputs are large. A second would be to create a

piecewise linear transfer function that has constant derivative between y-values of zero and one and is constant outside of this range.