

Evolving Spiking Networks with Variable Memristors

Gerard Howard
Dept. of Computer Science
University of the West of
England
Bristol, UK
david4.howard@uwe.ac.uk

Ella Gale
Dept. of Chemistry
University of the West of
England
Bristol, UK
ella.gale@uwe.ac.uk

Larry Bull
Dept. of Computer Science
University of the West of
England
Bristol, UK
larry.bull@uwe.ac.uk

Ben de Lacy Costello
Dept. of Chemistry
University of the West of
England
Bristol, UK

Andy Adamatzky
Dept. of Computer Science
University of the West of
England
Bristol, UK

ABSTRACT

This paper presents a spiking neuro-evolutionary system which implements memristors as neuromodulatory connections, i.e. whose weights can vary during a trial. The evolutionary design process exploits parameter self-adaptation and a constructionist approach, allowing the number of neurons, connection weights, and inter-neural connectivity pattern to be evolved for each network. Additionally, each memristor has its own conductance profile, which alters the neuromodulatory behaviour of the memristor and may be altered during the application of the GA. We demonstrate that this approach allows the evolutionary process to discover beneficial memristive behaviours at specific points in the networks. We evaluate our approach against two phenomenological real-world memristive implementations, a theoretical “linear memristor”, and a system containing standard connections only. Performance is evaluated on a simulated robotic navigation task.

Categories and Subject Descriptors

I.2.6 [Artificial Intelligence]: Learning—*connectionism and neural nets, parameter learning*

General Terms

Experimentation

Keywords

STDP, Neuro-Evolution, Memristors, Self-Adaptation, Constructivism

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

GECCO'11, July 12–16, 2011, Dublin, Ireland.

Copyright 2011 ACM 978-1-4503-0557-0/11/07 ...\$10.00.

1. INTRODUCTION

The memory-resistor, first implemented as a “memistor” by Widrow [39], then theoretically characterized by Chua [3] and renamed a “memristor”, has recently enjoyed a resurgence of interest from the research community after being manufactured *in silico* by HP labs [36]. A memristor is a fundamental passive two-terminal device whose state (memristance) is both nonvolatile and dependent on past activity. Nonvolatile memory [13] is perfect for low-power storage, and the device’s dynamic internal state facilitates information processing. These properties make the memristor an ideal candidate for use in nanoscale neural architectures [22], where the memristor can function as a synapse between - for example - Complementary Metal-Oxide Semiconductor (CMOS) neurons [27].

In this paper, we introduce the notion of a *variable memristor* e.g. a memristor whose conductance profile can vary as a result of the evolutionary process. As the conductance profile of the memristor is responsible for its behaviour, variable memristors can potentially impart a variety of adaptive behaviours to the networks. We analyse the computational properties of variable memristors when cast as synaptic connections in evolutionary Spiking Neural Networks (SNNs) [10].

Neural architectures require some form of learning mechanism in order to harness their computational power. A typical approach involves utilising a form of Hebbian learning [12] to realise Spike Time Dependant Plasticity (STDP) [18], so that memristors between a presynaptic and postsynaptic neuron can alter their efficiency dependant on the spike timings of those neurons. Here, the memristive element of the network allows the weight of the connections to vary during a trial and provides a neuromodulatory learning architecture which is shown to be beneficial to the evolutionary design process. We couple this neuromodulatory process with a constructive model of neuro-evolution, whereby the evolutionary process can add or remove both neurons and connections (which may be memristors) during the application of the Genetic Algorithm (GA) [14].

Our hypothesis is that the additional degrees of functional freedom afforded to the variable memristors can be beneficially harnessed by the evolutionary process. To test

this hypothesis we compare the variable memristor networks to a number of alternates, (i) PEO-PANI networks [8], (ii) HP networks, [36], (iii) idealised“linear memristor” networks, (iv) networks comprised of static (e.g. non-memristive) connections. A simulated robotics navigation task is selected for this purpose. To our knowledge, this is the first approach that allows for the self-adaptation of the characteristic performance of the memristors alongside neuroevolution of both neurons and connection structure.

The remainder of the article is ordered as follows: Section 2 introduces background research, Section 3 introduces the system, Section 4 details SNN implementation, and Section 5 outlines the static memristors. Section 6 details the discovery component and Section 7 details the topology mechanisms used. Following this, Section 8 details the variable memristor implementation, Section 9 gives the environment and Section 10 shows the experimental setup and analyses the results of the experiments that were carried out. Section 11 concludes with a discussion and future research directions.

2. BACKGROUND

2.1 Spiking Networks

Spiking Neural Networks (SNNs) are a relatively recent phenomenological model of neural activity in the brain. In an SNN, a number of neurons are linked via unidirectional, weighted connections that provide a method of intra-network communication. The medium of communication is the action potential, or spike, which is emitted from a neuron and received by all connected neurons that the given neuron is presynaptic to. Each neuron has a measure of excitation, known as “membrane potential”. A spike is emitted after an arbitrary neuron surpasses a certain level of excitation within a given window of time. This build-up of membrane potentials and release of postsynaptic current within a network is able to produce dynamic activation patterns through time, providing increased computing power [20] [31] when compared to other network models, such as the Multi Layer Perceptron (MLP) [30]. Two well-known formal SNN implementations are the Leaky Integrate and Fire (LIF) model [10] and the Spike Response Model (SRM) [10]. Neuroevolution involves the use of evolutionary techniques to alter the topology or weights of neural networks. A survey of various methods for evolving both weights and architectures is presented in [9], similarly the evolution of networks for robotics tasks is covered by Nolfi and Floreano [24].

2.2 Memristors

Memristors (memory-resistors) are the fourth fundamental circuit element, joining the capacitor, inductor and resistor. A memristor can be defined as a resistor whose current resistance value (a) depends on the previous charge that has passed through it (b) is nonvolatile. Formally, a memristor is a passive two-terminal electronic device that is described by the non-linear relation between the device terminal voltage, v , terminal current, i (which is related to the charge q transferred onto the device), and magnetic flux, φ , as (1) shows. Resistance increases or decreases depending on the direction of the current.

$$v = M(q)i \text{ or } i = W(\varphi)v \quad (1)$$

The memristance (M) and memductance (W) properties are both nonlinear functions, defined in (2) as:

$$M(q) = d\varphi(q)/dq \text{ and } W(\varphi) = dq(\varphi)/d\varphi \quad (2)$$

Previous applications of memristors within neural paradigms include HP memristors [36], which have been used in the manufacture of nanoscale neural crossbars [32], and AgSi memristors, which have been shown to function in neural architectures [17].

2.3 Synaptic Plasticity

Hebbian learning [12] is thought to account for adaptation and learning in the brain. Briefly, Hebbian learning states that “neurons that fire together, wire together” - in other words in the event that a presynaptic neuron causes a postsynaptic neuron to fire, the synaptic strength between those two neurons is increased so that such an event is more likely to happen in the future. Such a mechanism allows for self-organising, correlated activation patterns.

Integration of neuroevolution with heterogeneous neuro-modulation rules (which are similar to the different memristor types used here) is investigated by Soltoggio (e.g., [34]), and has been extended to robot controllers [7]. Probabilistic spike emission, governed by modulatory Hebbian rules, have also been investigated [21]. Urzelai and Floreano [38] present a nodes-only encoding scheme where synapses are affected by four versions of the Hebb rule.

Previous studies have presented methods to implement STDP learning using memristive synapses, notably [2][17] [19][33]. Consistent between the papers is the use of a “two-part spike”. The temporal coincidence between a spike sent backwards from a postsynaptic neuron and one sent forwards from a presynaptic neuron is used to alter the voltage across the memristor between those neurons; if it surpasses a threshold voltage the weight of the synapse is altered. The main difference between [17][33] and [2][19] is that in [17][33], the two-part spike is implemented as a discrete-time step-wise waveform approximation, whereas [2][19] use values calculated from continuous waveform equations, allowing them to operate in continuous time.

3. THE SYSTEM

The system consists of a population of SNNs which are evaluated on a robotics test problem, and altered via GA operation which is detailed in Section 6. To introduce the terminology to be used throughout this paper: each experiment lasts for 1000 evolutionary *generations*; each generation involves new networks in the population being evaluated on the test problem (a *trial*). Each trial consists of a number of *timesteps*, which begin with the reading of sensory information and calculation of action, and end with the agent performing that action. Every timestep consists of 21 *steps* of SNN processing, at the end of which the action is calculated.

4. SNN IMPLEMENTATION

We base our spiking implementation on the LIF model, although our model is heavily simplified in terms of the number of simulation steps used per action calculation. Neu-

rons can be stimulated either by an external current or by connections from presynaptic neurons. Each neuron has a membrane potential, y , where $y > 0$, which slowly degrades over time. As spikes are received by the neuron, the value of y is increased in the case of an excitory spike, or decreased if the spike is inhibitory. If y surpasses a positive threshold, y_{thresh} , the neuron spikes and transmits an action potential to every neuron to which it is presynaptic, with strength relative to the connection weight between those two neurons. The neuron then resets its membrane potential to some low number, given in Section 10. At time t , the membrane potential of a neuron is given as (3):

$$y(t+1) = y(t) + (I + a - by(t)) \quad (3)$$

$$If(y(t+1) > y_{thresh})y(t+1) = c \quad (4)$$

Equation (4) shows the reset formula. Here, $y(t)$ is the membrane potential at time t , I is the input current to the neuron, a is a positive constant, b is the degradation (leak) constant and c is the reset membrane potential of the neuron. A model of temporal delays is used so that, in the single hidden layer only, a spike sent to a neuron not immediately neighbouring the sending neuron is received x steps after it is sent, where x is the number of neurons between the sending neuron and receiving neuron.

4.1 Action Calculation

Action calculation involves the current input state being repeatedly processed 21 times by each network (an experimentally determined number of steps). Each network was initialised with six input neurons, nine hidden neurons, and two output neurons (used to calculate the action). Neurons are arranged into layers based on this classification. Each output neuron had an activation window that recorded the number of spikes emitted by that neuron. To calculate the three possible actions that a network could advocate, we classified the spike trains at the two output neurons as being either *low* or *high* activated. A neuron was said to be *highly* activated if it had spikes in over half (>11) of the positions in the sampling window after 21 steps; otherwise it was said to have *low* activation. The combined spike trains translated to a discrete movement; (high, high) or (low, low) = forwards, (high, low) = left turn, (low, high) = right turn, which was calculated once at the end of each timestep. See Section 9.1 for precise details of sensory state generation.

5. MEMRISTIVE CONNECTIONS

From the description of SNNs in Section 2.1 and that of memristors in Section 2.2, the strength of a connection weight in a neural network can be intuitively seen as the inverse memristance of that connection. The impact of the memristor on the functionality of the network depends on the memristive equations used. Next, we describe the equations governing the three comparative memristors; details on creating variable memristors are given in Section 8.

5.1 HP Memristor

The HP memristor is comprised of thin-film TiO_2 and TiO_{2-x} , which have different resistances. The boundary between the two compounds moves in response to the charge on the memristor, which in turn alters the resistance of the de-

vice. To allow for future self-adaptation, memristance equations are refactored from the original, as given in [36]. The static memristor profiles are recreated using $\beta = 1$.

In the following equations, W is the scaled weight (conductance) of the connection, G is the unscaled weight of the connection, M is the memristance, $sf1$ and $sf2$ are scale factors, R_{off} is the resistance of the TiO_2 , R_{on} is the resistance of the TiO_{2-x} , q is the charge on the device, q_{min} is the minimum allowed charge, and β encompasses physical properties of the device. Equations are presented in order of calculation.

$$sf1 = 0.99/1 - \left(\frac{1}{-R_{off}R_{on}\beta q_{min} + R_{off}} \right) \quad (5)$$

$$sf2 = 1/ \left(\frac{-R_{off}R_{on}\beta(R_{on} - R_{off})}{-R_{on}R_{off}\beta + R_{off}} sf1 \right) - 1 \quad (6)$$

$$q = \left(\frac{1}{-R_{off}R_{on}\beta} \right) \left(\frac{sf1}{(W+sf2)} - R_{off} \right) \quad (7)$$

$$M = -R_{off}R_{on}\beta q + R_{off} \quad (8)$$

$$G = \frac{1}{M} \quad (9)$$

$$W = Gsf1 - sf2 \quad (10)$$

5.2 PEO-PANI Memristor

The PEO-PANI memristor consists of layers of PANI, onto which Li^+ -doped PEO is added [8]. We have phenomenologically recreated the performance characteristics of the PEO-PANI memristor in terms of the HP memristor, creating a memristance curve similar to that seen in [8]. Two additional parameters, $G_{q_{min}}$ and $G_{q_{max}}$, are the values of G when q is at its minimum (q_{min}) and maximum (q_{max}) values respectively. As with the HP equations, $\beta = 1$ will produce the static PEO-PANI profile.

$$q_{max} = R_{on} - R_{off} / -R_{on}R_{off}\beta \quad (11)$$

$$G_{q_{min}} = 1/(-R_{off}R_{on}\beta q_{min} - R_{on}) + R_{on} \quad (12)$$

$$G_{q_{max}} = 1/(-R_{off}R_{on}\beta q_{max} - R_{on}) + R_{on} \quad (13)$$

The two scale factors are recalculated as

$$sf1 = 0.99/(G_{q_{max}} - G_{q_{min}}) \quad (14)$$

$$sf2 = (G_{q_{min}} sf1) - 0.01 \quad (15)$$

$$q = \left(\frac{1}{((W + sf2)/sf1) - R_{on}} + R_{on} \right) \left(\frac{1}{-R_{off}R_{on}\beta} \right) \quad (16)$$

$$M = (-R_{off}R_{on}\beta q - R_{on}) + (1/R_{on}) \quad (17)$$

G is calculated as in (9), and W as in (10).

5.3 Linear Memristor

The variable memristor alters W by $1/mem_lifetime$, therefore it takes $mem_lifetime$ memristance events to linearly increase W from R_{off} to R_{on} .

5.4 STDP

In Section 2.3 a number of STDP implementations were reviewed. We have elected to follow [33][17] in using discrete-time stepwise waveforms, as our SNNs operate in discrete

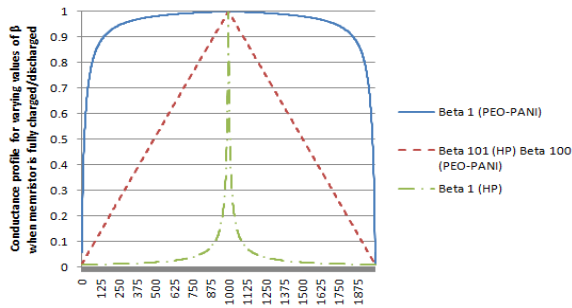


Figure 1: Displaying memristive profiles attained with different values of β . The x axis shows 1000 positive STDP events, followed by 1000 negative STDP events, assuming $mem_lifetime=1000$. Static HP and PEO-PANI memristors have $\beta=1$

time. We augment each neuron with a variable, $last_spike$, which is initially 0. When a neuron spikes, this value is set to some positive number. At the end of each of the 21 steps that make up a single timestep, each memristive connection is analysed by checking the $last_spike$ values of its presynaptic and postsynaptic neurons. If the calculated value exceeds some positive $spike_threshold$, memristance of the synapse occurs. Whether the connection increases or decreases depends on which neuron has the highest $last_spike$ value, providing pre- to postsynaptic temporal coincidence. If the $last_spike$ values are identical, memristance does not occur. At the end of each step, each $last_spike$ value is decreased by 1 to a minimum of 0, creating a discrete stepwise waveform through time. Each STDP event either increases or decreases q , by Δq , as detailed in (18), which is then used to calculate the weight as detailed in Sections 5.1 - 5.3.

$$\Delta q = (q_{max} - q_{min})/mem_lifetime \quad (18)$$

From Fig. 1 it can be seen that the amount of change in connection weight depends heavily on the memristors value of β and current connection weight. Both HP and PEO-PANI memristor types display increased sensitivity (larger ΔW per STDP event) when β is a low number and either $W > 0.1$ for HP networks, or $W < 0.9$ for PEO-PANI networks. Linear memristors display constant sensitivity.

6. DISCOVERY COMPONENT

Having described the component parts of our networks, we now detail the implementation of the GA that acts on them. In our GA, two parents are selected fitness-proportionately, mutated, and used to create two offspring. We use only mutation to explore weight space; crossover is omitted as sufficient solution space exploration can be obtained via a combination of self-adaptive weight and topology mutations; a view that is reinforced in the literature, e.g. [29]. The offspring are inserted into the population and two networks with the lowest fitness deleted. Parents stay in the population competing with their offspring.

6.1 Self-adaptive Mutation

We utilise self-adaptive mutation rates as in Evolutionary Strategies (ES) [28], to dynamically control the frequency and magnitude of mutation events taking place in each net-

work. This allows increased structural stability in highly fit networks whilst allowing less fit networks to vary more strongly per GA application. Here, the μ ($0 < \mu \leq 1$) value (rate of mutation per allele) of each network is initialized uniformly randomly in the range $[0,0.25]$. During a GA cycle, a parent's μ value is modified as in (19), the offspring then adopts this new μ , and mutates itself by this value, before being inserted into the population.

$$\mu \leftarrow \mu \exp^{N(0,1)} \quad (19)$$

Only non-memristive networks can alter their connection weights via the GA. Connection weights in this case are initially set during network creation, node addition, and connection addition randomly uniformly in the range $[0,1]$. Memristive network connections are always set to 0.5, and cannot be mutated from this value. This aims to force the memristive networks to harness the plasticity of their connections during a trial to successfully solve the problem.

7. TOPOLOGY MECHANISMS

In addition to self-adaptive mutation, we apply two evolutionary topology morphology schemes to allow the modification of the spiking networks in two ways; by adding/removing hidden layer nodes, and adding/removing inter-neural connections. The effect of this self-adaptive, constructivist framework is to tailor the evolution of the network to the complexity of the environment explicitly. This allows each network to control its own architecture autonomously in terms of (i) amount of mutation that takes place in a given network at a given time (ii) adapting the hidden layer topology of the neural networks to reflect the complexity of the problem considered by the network. Both memristive and non-memristive networks use these topology mechanisms.

7.1 Constructivism

Constructivist learning postulates that neural structures are initially small and sparsely connected [26]. Learning acts to add emergent neural structure as a result of the learner's interaction with its environment. Implementations include Synapsing Variable Length Crossover (SVLC) [16], and its inspiration, the Species Adaptive Genetic Algorithm (SAGA) [11]. Also relevant is Neuro Evolution of Augmenting Topologies (NEAT) [35], which combines neurons from a predetermined number of subpopulations to encourage diversity and enforce niche-based evolutionary pressure. The implementation of constructivism in this system is based on that used to evolve constructive SNNs in neural Learning Classifier Systems (LCS) [15], due to the demonstrated utility of this approach when using spiking neurons. Each network has a varying number of hidden layer neurons; additional neurons can be added or removed from the single hidden layer. Constructivism takes place during a GA cycle, after mutation. Two new self-adaptive parameters, ψ ($0 < \psi \leq 1$) and ω ($0 < \omega \leq 1$), are incorporated into the model. Here, ψ is the probability of performing a constructivism event and ω is the probability of adding a neuron; removal occurs with probability $1 - \omega$. Both have initial values taken from a random uniform distribution, with ranges $[0,0.5]$ for ψ and $[0,1]$ for ω . Offspring networks have their parents ψ and ω values modified using (19) as with μ . Nodes created during constructivism are initially excitatory with 50% probability, otherwise they are inhibitory.

7.2 Connection Selection

Automatic feature selection is a method of reducing the dimensionality of the data input to a process by using computational techniques to select and operate exclusively on a subset of inputs. In the context of neural networks, the connection structure - as opposed to the connection weights - of artificial neural networks was first evolved by Dolan and Dyer [5]. In this paper we allow each connection to be individually enabled/disabled, a mechanism termed ‘‘Connection Selection’’. During a GA cycle a connection can be enabled or disabled based on a new self-adaptive parameter τ , which is initialised as with μ with range [0,0.5]. If a connection is enabled for a non-memristive network, its connection weight is randomly initialised uniformly in the range [0,1], memristive connections are always set to 0.5. During a node addition event, new connections are set probabilistically, with $P(\text{connection enabled}) = 0.5$. Connection selection is particularly important to the memristive networks. As they cannot alter connection weights via the GA, variance induced in network connectivity patterns plays a large role in the generation of STDP in the networks.

8. VARIABLE MEMRISTORS

Despite being a field in its infancy, a number of different memristors have been manufactured from a variety of different materials, including TiO_2 [36], conductive polymer [8], AgSi [17], and crystalline oxides [6]. Unique memristive profiles are being discovered regularly; for this reason it is assumed that any evolved memristors will have an approximate physical analogue and thus any results will be (eventually) physically replicable. The notion that varied memristive behaviours could be combined in a single network is an attractive one from a computing perspective, as more functional degrees of freedom are afforded to the synapses. Additionally, certain memristive behaviours may be more beneficial than others in certain positions within the network. Mixing different types of synaptic plasticity has been previously investigated by Soltoggio [34], Maass and Zador [21], and Urzelai and Floreano [38].

To allow the memristive profiles of the connections to change, we self-adapt β , which is derived from the physical characteristics of the device. The self-adaptive memristor profiles are allowed to range from HP-like to PEO-PANI-like profiles, each of which are governed by different equations, outlined in Sections 5.1 and 5.2. Because of this, we augment each memristor with a *type*, which is set to either 0 and 1 on memristor initialisation, with $P=0.5$ of each *type* being selected based on a uniform distribution. β is then initialised randomly uniformly in the range $[\beta_{min}, \beta_{max}]$. If *type* = 0, the refactored HP equations are used to calculate the profile of the device; otherwise the PEO-PANI equations are used. During GA activity, on satisfaction of a new self-adaptive parameter ι , which is self-adapted as with μ , a memristors β changes by $\pm 10\%$ of the total range of β . If a memristors new value of β surpasses a threshold β_{max} , the *type* of the memristor is switched and a new β calculated as $\Delta\beta - \beta_{max}$. In this way, a smooth transition between the different profile types is provided.

9. TEST ENVIRONMENT

Our chosen robotics simulator was the Webots platform [23], a test bed that is popular amongst the research com-

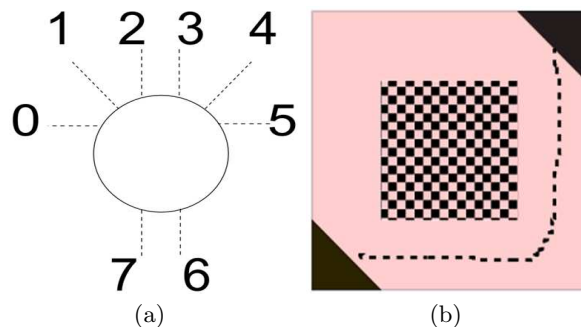


Figure 2: (a)The sensory configuration of the simulated Khepera agent (b)The test environment; the agent starts in the lower left triangle and must navigate to the upper-right triangle, avoiding the central obstacle.

munity; alternatives are summarised [4]. Previous applications of Webots in the field of evolutionary robotics include the application of incremental neuro-evolution to generate complex behaviours [37], and investigations into hierarchical neural control [25].

9.1 The Agent

The agent was a simulated Khepera II robot. At each timestep, the agent sampled its light sensors and IR distance sensors. All sensor readings were scaled to the range [0,1] for computational reasons. Six sensors were used to comprise the input state for the SNN, three IR and three light sensors at positions 0, 2 and 5 as shown in figure 2(a). Additionally, two bump sensors were added to the front-left and front-right of the agent to prevent it from becoming stuck against an object. If either bump sensor was activated, an interrupt was sent causing the agent to reverse 10cm and the agent to be penalised by 10 timesteps. Movement values and sensory update delays were constrained by accurate modelling of physical Khepera agent. Three actions were possible: forward, (maximum movement on both left and right wheels based on physical Khepera data) and continuous turns to both the left and right (caused by halving the left/right motor outputs respectively).

9.2 Environment

The agent was randomly located within a walled arena which it could not leave, with coordinates ranging from [-1,1] in both x and y directions. We constrained the agents initial starting position to a triangle in the lower left-hand corner of the environment, where $x + y < -1.5$. Adding to the complexity of the environment, a three-dimensional box was placed centrally in the arena, with vertices on ‘‘ground level’’ at $(x = -0.4, y = -0.4)$, $(-0.4, 0.4)$, $(0.4, 0.4)$, and $(0.4, -0.4)$, and raised to a height of $z = 0.15$. A light source, modelled on a 15 Watt bulb, was placed at the top-right hand corner of the arena $(x = 1, y = 1)$, which the agent must approach. The environment is shown in figure 2(b). When the agent reached the goal state (where $x + y > 1.6$), the responsible network received a constant fitness bonus of 2500, which was added to the fitness function f ($f > 0$) outlined in (20), where pos_x and pos_y is the agents current x and y position, and st is the number of timesteps taken

Table 1: Detailing performance t-test results (p values) for all systems in the experiment

	Performance	High fitness	Neurons	Connectivity	ψ	ω	τ
SA-HP	0.036	0.146	0.198	0.316	0.08	0.002	0.09
SA-PEO	0.128	0.421	0.232	0.79	0.058	0.005	0.061
SA-LIN	0.132	0.048	0.520	0.227	0.865	0.008	0.915
SA-GA	0.077	0.532	0.046	0.019	0.478	0.005	0.017

to solve. The fitness of an agent is calculated at the end of every timestep, with the highest attained value of f during the trial kept as the fitness value for that network. Optimal performance gives $f = 11800$, which corresponds to 700 timesteps from start to goal state with no collisions.

$$f = (1/(1.6 - (|posx + posy|))) \times 1000 - st \quad (20)$$

10. EXPERIMENTAL SETUP

In the following experiments we gauged the impact of variable memristor connections, comparing to benchmark systems comprised of homogeneous static HP, PEO-PANI and linear memristors, and constant connections. We refer to the various network types as follows: variable memristor = SA, static HP memristor = HP, static PEO-PANI memristor = PEO, static linear memristor = LIN, non-memristive network = GA. All experiments had a population size of 100 networks and were evolved for 1000 generations, with a maximum of 4000 timesteps per trial. SNN parameters were *initial hidden layer nodes* = 9, $a = 0.3$, $b = 0.05$, $c = 0.0$, $c_{ini} = 0.5$, $ythresh = 1.0$, *output window size*=21, *last_spike* = 3, *spike_threshold* = 4. In memristive networks, all connections were memristive. Memristive parameters were $R_{on} = 0.01$, $R_{off} = 1$, static $\beta=1$, $\beta_{min} = 1$, $\beta_{maxHP} = 100$, $\beta_{maxPEO-PANI} = 100$ $q_{min} = 0.0098$, $mem_lifetime = 1000$.

The experiment was repeated ten times, the statistics recorded were the averages of these ten runs. Every 20 trials, the current state of the system was stored and used to create the results that follow. To facilitate useful comparisons, we defined a notion of “performance”. As the start location was tightly constrained, we say the performance of the system is equal to the first trial in which the goal state is found, so that a lower value indicated higher performance. This measure allowed us to perform t-tests to compare the respective performances of the four systems. In the following tables, “Performance” was the average performance as outlined above. “High fitness” refers to the average fitness of the highest-fitness network in each run. “Neurons” were the average final connected neurons per network in the population and “Connectivity” was the average percentage of enabled connections in the population. During a trial, some of the memristive connections in the networks may experience STDP, altering their weights. After every trial, memristive connections were reset to their original values of 0.5.

10.1 Results

In solving the test problem, two general high-fitness strategies were employed by the SA networks. In either case, STDP was harnessed to turn the agent. HP-governed SA beta profiles were found to quickly reduce synaptic efficacy to the left (right) motor, causing perturbation of calculated action during turn by bringing that motor below the “high

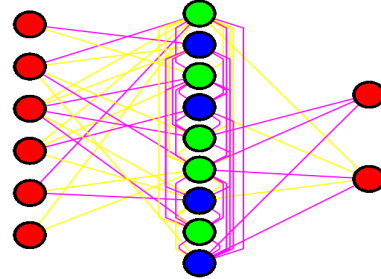


Figure 3: An example evolved network. Light coloured neurons are excitory, dark neurons are inhibitory

activated” threshold. PEO-PANI-governed SA profiles to the same motor were used to swiftly increase the level of spiking activity (usually in response to a light sensor surpassing some threshold) until a “forwards” action was calculated after the turn was completed. A typical evolved network is shown in Figure 3.

10.1.1 Performance

T-tests (data columns 1-4, Table 1) show a number of promising results. SA networks achieved statistically higher “performance” than HP networks ($p=0.036$), and higher “high fitness” than networks LIN networks ($p=0.048$). Figure 4(a) shows SA networks trail slightly behind PEO and LIN networks in terms of performance, but are better than HP and GA. This is an encouraging result considering the vastly increased search/behaviour space the SA discovery component has to deal with due to β , as it indicates that the variable memristor induces no significant performance overhead. Similar results can be seen for average fitness, Figure 4(b).

Figure 4(c) reveals that SA networks finish with the fewest number of required neurons, significantly less than GA networks ($p=0.046$). However, Figure 4(d) shows that GA networks have statistically sparser connectivity ($p=0.019$) than SA networks. Connections were expected to be more prolific in SA networks as they are computationally more powerful than static connections, a notion echoed in recent literature [1]. Considering possible hardware implementations, CMOS neurons are larger and more complex than the synapses that connect them. As neuron numbers are more likely to be a constraint, a reduction in neurons despite increased connectivity can be said to be beneficial. In the case of both Figure 4(c) and 4(d), the profile of the HP memristor did not infer statistical significance.

10.1.2 Self-adaptive parameters

Self-adaptive parameter results can be seen in the final 3 columns of Table 1 (figures not shown). μ was not compared as it was only used in GA networks (final average value

0.022). Similarly ι was only used in SA networks (final average value 0.026). Two main results were of interest (i) SA networks had a universally lower ω , which governed the rate of neuron addition. This result indicated more parsimonious SA network evolution, allowing significantly less neurons than GA networks (ii) GA networks had lower τ than SA networks, allowing sparser connectivity (Figure 4(d)).

10.1.3 Evolution of β

As β varied between 1-101 in the case of SA HP-governed profiles and 1-100 in the case of SA PEO-PANI-governed profiles, the total range of β is 199, where a value between 1-101 is considered to be a HP-governed profile and anything >101 is a PEO-PANI-governed profile.

Analysis of the SA networks revealed that the connections to the motor on the side that made the turn evolved less linear profiles, allowing for quicker action switching behaviour. In addition, connections between the input and hidden layer had a lower average maximum (145.98 vs. 150.11) and higher average minimum (50.945 vs. 27.533) than those between the hidden and output layer. This suggests that connections to motors in general were evolutionarily preferred to have more nonlinear conductance profiles. Connections in both of these areas had higher average maximum and lower average minimum β values than connections within the hidden layer (116.16 and 84.1 respectively), suggesting that more steady memristance profiles were preferred there.

11. CONCLUSIONS

In this paper we have introduced the notion of a variable memristor and analysed its synaptic performance when compared to three static memristor types and non-memristive connections in a simulated robotics environment. The main benefit of memristive STDP over other STDP implementations lies in hardware implementations, as the efficiency and past history of the synapse is stored in the nonvolatile physical state of the device and thus does not require simulation.

Our hypothesis was that the additional degrees of functional freedom afforded to the variable memristors allowed them to outperform these other networks. Numerous findings support this hypothesis, including higher performance than HP networks, higher quality solutions than linear memristor networks, and fewer required neurons than GA networks. These findings suggest that self-adaptation of β is harnessed by the evolutionary process to provide flexible plastic networks with more implicit degrees of freedom than the other network types. Variable plasticity was harnessed via STDP to achieve more expedient goal-finding behaviour with reduced topological complexity when compared to certain other network types. Importantly, the self-adaptation process itself was found to be non-disruptive with respect to network performance. Possible future research directions include hardware and mixed-media implementations.

12. REFERENCES

- [1] L. Abbott and W. Regehr. Synaptic computation. *Nature*, (431):796–803, 2004.
- [2] A. Afifi, A. Ayatollahi, and F. Raissi. Stdp implementation using memristive nanodevice in cmos-nano neuromorphic networks. *IEICE Electronics Express*, 6(3):148–153, 2009.

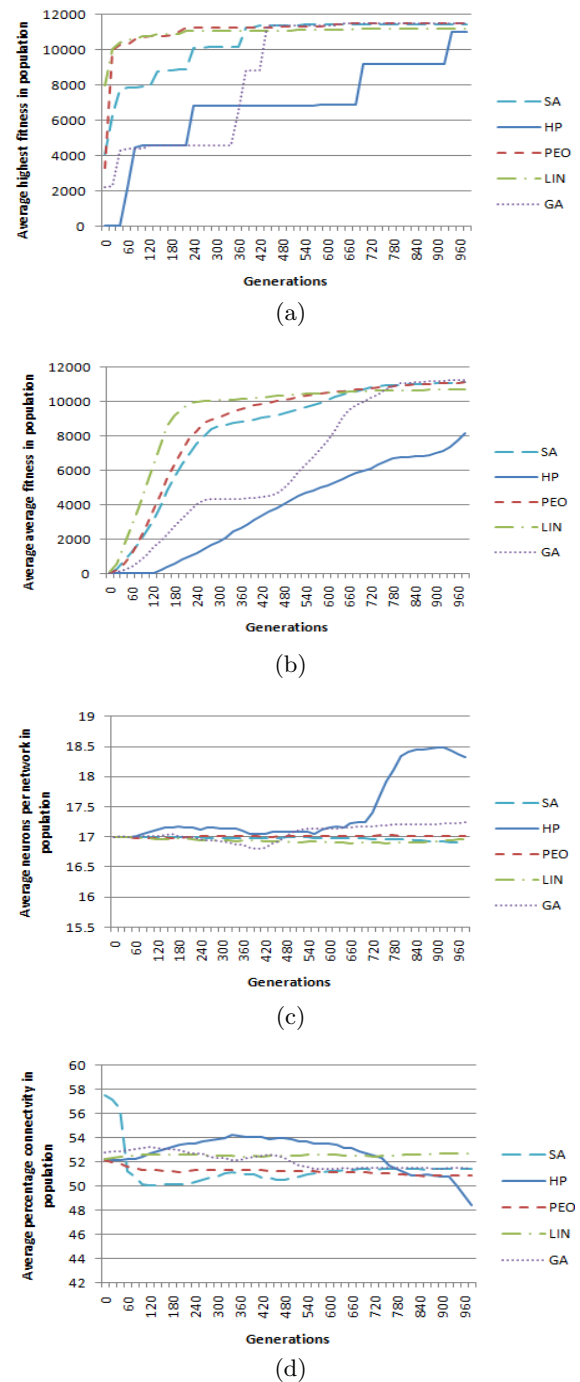


Figure 4: (a) average highest fitness (b) average mean fitness (c) average connected hidden layer nodes, (d) average % enabled connections for the experiment

- [3] L. Chua. Memristor-the missing circuit element. *Circuit Theory, IEEE Transactions on*, 18(5):507 – 519, Sept. 1971.
- [4] J. Craighead, R. Murphy, J. Burke, and B. Goldiez. A survey of commercial open source unmanned vehicle simulators. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 852 –857, apr 2007.

- [5] C. P. Dolan and M. G. Dyer. Toward the evolution of symbols. In J. J. Grefenstette, editor, *Genetic Algorithms and their Applications (ICGA '87)*, pages 123–131, Hillsdale, New Jersey, 1987. Lawrence Erlbaum Associates.
- [6] W. Doolittle, W. Calley, and W. Henderson. Complementary oxide memristor technology facilitating both inhibitory and excitatory synapses for potential neuromorphic computing applications. In *Semiconductor Device Research Symposium, 2009. ISDRS '09. International*, pages 1–2, 2009.
- [7] P. Durr, C. Mattiussi, A. Soltoggio, and D. Floreano. Evolvability of Neuromodulated Learning for Robots. In *Proceedings of the 2008 ECSIS Symposium on Learning and Adaptive Behavior in Robotic Systems*, pages 41–46, Los Alamitos, CA, 2008. IEEE Computer Society.
- [8] V. Erokhin and M. P. Fontana. Electrochemically controlled polymeric device: a memristor (and more) found two years ago. *ArXiv e-prints*, July 2008.
- [9] D. Floreano, P. Durr, and C. Mattiussi. Neuroevolution: from architectures to learning. 2008.
- [10] W. Gerstner and W. Kistler. *Spiking Neuron Models - Single Neurons, Populations, Plasticity*. Cambridge University Press, 2002.
- [11] I. Harvey, P. Husbands, and D. Cliff. Seeing the light: artificial evolution, real vision. In *Proceedings of the third international conference on Simulation of adaptive behavior : from animals to animats 3: from animals to animats 3*, pages 392–401, Cambridge, MA, USA, 1994. MIT Press.
- [12] D. O. Hebb. *The organisation of behavior*. Wiley, New York, 1949.
- [13] Y. Ho, G. M. Huang, and P. Li. Nonvolatile memristor memory: Device characteristics and design implications. In *ICCAD*, pages 485–490. IEEE, 2009.
- [14] J. H. Holland. Adaptation. In R. Rosen and F. M. Snell, editors, *Progress in theoretical biology IV*, pages 263–293. Academic Press, New York, 1976.
- [15] G. Howard, L. Bull, and P.-L. Lanzi. A spiking neural representation for xcsf. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1–8, jul 2010.
- [16] B. Hutt and K. Warwick. Synapsing variable-length crossover: Meaningful crossover for variable-length genomes. *Evolutionary Computation, IEEE Transactions on*, 11(1):118–131, 2007.
- [17] S. H. Jo, T. Chang, I. Ebong, B. B. Bhadviya, P. Mazumder, and W. Lu. Nanoscale memristor device as synapse in neuromorphic systems. *Nano Letters*, 10(4):1297–1301, 2010. PMID: 20192230.
- [18] W. M. Kistler. Spike-timing dependent synaptic plasticity: a phenomenological framework. *Biological Cybernetics*, 87(5-6):416–427, 2002.
- [19] B. Linares-barranco and T. Serrano-gotarredona. Memristance can explain spike-time-dependent-plasticity in neural synapses, 2009.
- [20] W. Maass. Networks of spiking neurons: the third generation of neural network models. *Neural networks*, 1997.
- [21] W. Maass and A. M. Zador. Dynamic stochastic synapses as computational units. *Neural Computation*, 11(4):903–917, 1999.
- [22] C. Mead. Neuromorphic electronic systems. *Proceedings of the IEEE*, 78(10):1629–1636, 1990.
- [23] O. Michel. WebotsTM: Professional mobile robot simulation. *International Journal of Advanced Robotic Systems*, 1(1):39–42, 2004.
- [24] S. Nolfi and D. Floriano. *Evolutionary Robotics*. The MIT Press, Cambridge, Massachusetts, 2000.
- [25] R. W. Paine, R. W. Paine, J. Tani, and J. Tani. How hierarchical control self-organizes in artificial adaptive systems. *Adaptive Behavior*, 13:211–225, 2005.
- [26] S. R. Quartz and T. J. Sejnowski. The neural basis of cognitive development: A constructivist manifesto. *Behavioral and Brain Sciences*, Jan. 01 1997.
- [27] J. M. Rabaey. *Digital integrated circuits: a design perspective*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1996.
- [28] I. Rechenberg. *Evolutionsstrategie: optimierung technischer systeme nach prinzipien der biologischen evolution*. Frommann-Holzboog, 1973.
- [29] M. Rocha, P. Cortez, and J. Neves. Evolutionary neural network learning. In *Progress in Artificial Intelligence*, volume 2902 of *Lecture Notes in Computer Science*, pages 24–28. Springer Berlin / Heidelberg, 2003.
- [30] D. Rumelhart and J. McClelland. *Parallel Distributed Processing*, volume 1 & 2. MIT Press, Cambridge, MA, 1986.
- [31] K. Saggie-Wexler, A. Keinan, and E. Ruppin. Neural processing of counting in evolved spiking and mcculloch-pitts agents. *Artificial Life*, 12(1):1–16, 2006.
- [32] G. Snider. Computing with hysteretic resistor crossbars. *Appl. Phys. A.*, 80:1165–1172, 2005.
- [33] G. Snider. Spike-timing-dependent learning in memristive nanodevices. In *Nanoscale Architectures, 2008. NANOARCH 2008. IEEE International Symposium on*, pages 85–92, jun 2008.
- [34] A. Soltoggio. Neural plasticity and minimal topologies for reward-based learning. In *Proceedings of the 2008 8th International Conference on Hybrid Intelligent Systems*, pages 637–642, Washington, DC, USA, 2008. IEEE Computer Society.
- [35] K. O. Stanley and R. Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10:99–127, 2002.
- [36] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams. The missing memristor found. *Nature*, 453:80–83, 2008.
- [37] R. A. T  llez and C. Angulo. Progressive design through staged evolution. *Frontiers in Evolutionary Robotics*, 2008.
- [38] J. Urzelai and D. Floreano. Evolution of adaptive synapses: Robots with fast adaptive behavior in new environments. *Evol. Comput.*, 9:495–524, December 2001.
- [39] B. Widrow. An adaptive 'adaline' neuron using chemical 'memristors'. Technical Report 1533-2, Stanford Electronics Laboratories, Stanford, CA, oct 1960.