White Rose Research Online

eprints@whiterose.ac.uk
https://eprints.whiterose.ac.uk
Universities of Leeds, Sheffield and York

Deposited via The University of York.

White Rose Research Online URL for this paper:
https://eprints.whiterose.ac.uk/id/eprint/93468/

Version: Published Version

Article:
Turner, Alexander Phillip, Caves, Leo, Stepney, Susan et al. (2017) Artificial Epigenetic Networks: Automatic Decomposition of Dynamical Control Tasks using Topological Self-Modification. IEEE transactions on neural networks and learning systems. 7372471. pp. 218-230. ISSN: 2162-237X

https://doi.org/10.1109/TNNLS.2015.2497142

Reuse
Items deposited in White Rose Research Online are protected by copyright, with all rights reserved unless indicated otherwise. They may be downloaded and/or printed for private study, or other acts as permitted by national copyright laws. The publisher or other rights holders may allow further reproduction and re-use of the full text version. This is indicated by the licence information on the White Rose Research Online record for the item.

Takedown
If you consider content in White Rose Research Online to be in breach of UK law, please notify us by emailing eprints@whiterose.ac.uk including the URL of the record and the reason for the withdrawal request.

Artificial Epigenetic Networks: Automatic Decomposition of Dynamical Control Tasks Using Topological Self-Modification

Alexander P. Turner, *Member, IEEE*, Leo S. D. Caves, Susan Stepney, Andy M. Tyrrell, *Senior Member, IEEE*, and Michael A. Lones, *Senior Member, IEEE*

***Abstract*— This paper describes the artificial epigenetic network, a recurrent connectionist architecture that is able to dynamically modify its topology in order to automatically decompose and solve dynamical problems. The approach is motivated by the behavior of gene regulatory networks, particularly the epigenetic process of chromatin remodeling that leads to topological change and which underlies the differentiation of cells within complex biological organisms. We expected this approach to be useful in situations where there is a need to switch between different dynamical behaviors, and do so in a sensitive and robust manner in the absence of *a priori* information about problem structure. This hypothesis was tested using a series of dynamical control tasks, each requiring solutions that could express different dynamical behaviors at different stages within the task. In each case, the addition of topological self-modification was shown to improve the performance and robustness of controllers. We believe this is due to the ability of topological changes to stabilize attractors, promoting stability within a dynamical regime while allowing rapid switching between different regimes. Post hoc analysis of the controllers also demonstrated how the partitioning of the networks could provide new insights into problem structure.**

***Index Terms*— Epigenetic networks, intelligent control, recurrent neural networks (RNNs), self-modification, task decomposition.**

I. Introduction

COMPLEX real world tasks can often be reduced to multiple interacting subtasks. It has long been realized that there are advantages to capturing the structure of this subtask decomposition within the topology of a neural network architecture, especially when compared with monolithic networks [1]. Conventionally, this is done using various kinds of modular neural network [2], [3], which typically structure solutions as a decision tree whose leaves are subnetworks, each trained to solve a particular subtask. In this respect, modular neural networks resemble the macrostructure of the human brain, which is also known to be structured as a hierarchy of special-purpose neural circuits [4].

The brain is not the only naturally occurring connectionist architecture known to solve complex tasks. Another prominent biological network, which we consider in this paper, is a cell's gene regulatory network. There are many similarities between neural and genetic networks; indeed, artificial neural networks (ANNs) have been used as a modeling tool for capturing the structure and dynamics of genetic networks [5]. However, there are also some prominent differences between the two [6]. One of these is the widespread existence of self-modifying processes within genetic networks, whereby the cellular machinery expressed by the genetic network induces physical changes within the network's topology. In this paper, we focus upon a self-modifying process that is central to task specialization within biological cells: chromatin remodeling [7]. Chromatin remodeling is described in Section II in detail. However, in a nutshell, it is a mechanism that turns genetic subnetworks ON and OFF by regulating their exposure to the cell's gene expression machinery. Significantly, the biochemical components that control chromatin remodeling are expressed by the genetic network; so, in essence, the genetic network regulates changes to its own topology.

The premise of this paper is that the topological self-modification can be used as a novel mechanism for achieving task decomposition within connectionist architectures. In the conventional modular approach to task decomposition, processing is divided into independent subnetworks, which are always turned ON, and whose outputs are integrated by some higher level decision node. In our approach, by comparison, the subnetworks used to solve different subtasks can be overlapping, are only turned ON when in use, and the output is determined by whichever subnetwork is currently expressed. We expect the resulting approach to be useful in situations where there is no *a priori* knowledge of how a task can be decomposed, where there is significant overlap between subtasks, and where highly dynamic solutions are beneficial. We demonstrate this by showing that a self-modifying connectionist architecture is able to solve three difficult control

Manuscript received February 17, 2015; revised October 19, 2015; accepted October 21, 2015. This work was supported by the Engineering and Physical Sciences Research Council through the Project entitled Artificial Biochemical Networks: Computational Models and Architectures under Grant EP/F060041/1 and through the Project entitled Bio-Inspired Adaptive Architectures and Systems under Grant EP/K040820/1.

A. P. Turner, L. S. D. Caves, S. Stepney, and A. M. Tyrrell are with the University of York, York YO10 5DD, U.K. (e-mail: alexander.turner@york.ac.uk; leo.caves@york.ac.uk; susan.stepney@york.ac.uk; amt@ohm.york.ac.uk).

M. A. Lones is with the School of Mathematical and Computer Sciences, Heriot-Watt University, Edinburgh EH14 4AS, U.K. (e-mail: m.lones@hw.ac.uk).

Data created during this research is available at the following DOI: 10.15124/45427dd0-fa53-4e78-b86a-8a9a177541e0.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2015.2497142

This work is licensed under a Creative Commons Attribution 3.0 License. For more information, see http://creativecommons.org/licenses/by/3.0/

tasks. This paper builds upon an earlier model [8], in which task decomposition was prespecified, and upon initial results reported in [9].

This paper is structured as follows. Section II introduces genetic networks and the form of topological self-modification carried out by chromatin remodeling. Section III summarizes the related work on computational models of genetic networks, task decomposition in ANNs, and self-modification. Section IV describes the self-modifying mechanism used in this paper. Section V outlines the dynamical control tasks. Section VI presents the results and analysis. Finally, the conclusion is drawn in Section VII.

II. Genetic Networks and Chromatin Remodeling

A gene is a region of DNA that describes a protein. In order for this protein to be expressed within a cell, the gene must be transcribed (and later translated) by the cell's processing machinery. In higher organisms, such as humans, this involves the binding of a group of around 5–20 interacting proteins, known as a transcription complex. These proteins, in turn, are the products of other genes. Hence, genes are regulated by other genes, and this pattern of regulatory interactions, when extended to all genes, forms the cell's genetic network. Genetic networks have many similarities with recurrent neural networks (RNNs). After abstracting away the detailed biochemical mechanisms, both can be considered to be a set of interacting nodes with connection weights, and both can be viewed as dynamical systems operating on a network. In many cases, gene regulation, such as neurone activation, can be modeled as a sigmoidal function [5].

However, there are also some fundamental differences between neural and genetic networks [6]. Perhaps most significantly, there is no analog of physical wiring (i.e., axons and synapses) in a genetic network. Rather, regulatory pathways emerge from stochastic spatial processes of dynamic molecular association and dissociation. In practice, this means that interactions between biochemical components are relatively unconstrained and, as a consequence, evolution is free to explore interactions between different cellular components. Chromatin modification is a good example of this. Chromatin is an assembly of structural proteins (histones) organized into spindles (nucleosomes) over which DNA is wound [10]. It was originally seen as a spatial compression mechanism that allows very long strands of DNA to fit into the cell's relatively compact nucleus. However, in recent years, it has become clear that the structure of chromatin is closely regulated by the genes it contains, whose products are able to locally wind or unwind the nucleosomes in order to permit or block access to the transcription machinery.

Hence, a different view of chromatin has emerged as a dynamic mechanism for modifying the complement of expressed genes and, hence, the topology of the cell's genetic network. Nowadays, chromatin remodeling is believed to play a significant role in determining cell fate. Exactly, how this is achieved in biological systems remains a topic of contemporary research; however, it has been hypothesized that it is due to the stabilization of the attractor states of the underlying genetic network [11], presumably by removing extraneous genetic pathways. Nevertheless, it is clear that chromatin remodeling plays a key part in this cellular analog of subtask specialization. Given the underlying similarities between genetic and neural networks, it is intriguing to consider whether an analogous mechanism could be used for task decomposition within ANNs.

III. Related Work

A. Artificial Gene Regulatory Networks

Historically, the development of computational models of genetic networks has focused on their role in understanding biological systems, for instance inferring computational models from measurement data [12] or using computational models to understand systems-level properties of genetic networks [13]. Another, less well-known role, involves using these models to carry out computation, in a manner akin to the relationship between ANNs and biological neural networks. These artificial gene regulatory networks take on various forms (see [14] for a recent review). In some cases, representations are borrowed from the wider genetic network modeling community. For instance, Boolean networks, which model genetic networks as the networks of interacting logic functions, have been used to control robots [15]. In other cases, new models have been developed. This includes work on artificial genomes [16] and fractal gene regulatory networks [17].

Given the relative immaturity of the field, it is unclear which model is most suitable for doing a particular kind of computation. In practice, there are likely to be different tradeoffs between expressiveness, efficiency, compactness, and robustness. Since these models are often optimized using evolutionary algorithms, there is also a difficulty discriminating between the influence of expressiveness and evolvability. In this paper, we are interested in understanding the potential benefits of introducing topological modification to connectionist models. Hence, we make use of a relatively simple representation that closely resembles an ANN. Such models have previously been used to model genetic networks [5], and their optimization using evolutionary algorithms has also been well studied [18]. This approach also has the benefit that lessons learnt can be directly applied to the wider field of ANN research.

B. Task Decomposition Using Modular Neural Networks

We are interested in how topological self-modification can be used for achieving automatic task decomposition. As we have already remarked, the mechanism currently most used for achieving task decomposition in ANNs is the partitioning of a network into modules, each of which is used to solve a particular subtask. How this is done varies considerably [19], [20], although all approaches must have some mechanism for identifying modules and then determining which modules to use in a given situation. Modular ANNs are most often applied in domains, such as classification, where *a priori* knowledge of the task domain is available.

When *a priori* knowledge is available, it may be possible to identify the subtasks in advance and train modules accordingly. However, in the more general case, it is also necessary to determine the correct number of modules required

to solve the task. This is arguably easier using neuroevolution approaches, where it is relatively easy to adapt the gross structure of the collective network. An interesting example of this is the use of coevolution [21], where a population of modules is evolved in parallel with a population of module combinators, allowing the algorithm to explore different combinations of modules in a relatively open-ended fashion. It should be noted, however, that coevolutionary algorithms are inherently complex, requiring significant expertise in order to avoid pathological conditions. Where *a priori* knowledge is available, choosing between modules may be as simple as matching input cases. Other approaches include asking modules to vote on their applicability for a particular subtask [1], or the construction of decision trees to determine transitions between modules [22].

Our approach differs from the existing modular ANNs in a number of ways. First, there is no need to explicitly identify the modules. In fact, there is no reason why modules should be completely segregated, since in many cases, it might be advantageous (in terms of size and training cost) for processing to be shared between subtasks. In this paper, this can be achieved using overlapping subnetworks. Second, transitions between subnetworks are handled by the subnetworks themselves, by turning ON or OFF other subnetworks. In effect, the system can transition between subnetworks at all points during execution, suggesting that this might lead to far more dynamic and context-sensitive solutions.

C. Topological Rewriting in Artificial Neural Networks

A number of authors have looked at applying topological rewriting processes during the learning phase of an ANN. This includes learning algorithms that add or remove nodes and links to or from the network. Examples include the early work by Ash [23] on dynamic node creation during backpropagation, and more recent work by Forti and Foresti [24] on dynamic self-organizing maps. Another prominent application of a self-modification process during learning is the work by Schmidhuber *et al.* [25], who looked at self-modification of the learning algorithm itself. There have also been a number of examples of self-modifying and self-organizing processes applied prior to the execution of an ANN, in the form of a developmental mapping. This includes the early work by Gruau [26] on rewriting grammars, and work by Astor and Adami [27] who made use of an artificial chemistry to determine the topology of an ANN. However, in all of these examples, the topology of the network remains fixed during the execution of the ANN.

There are very few examples of ANNs which use self-modifying processes during execution. GasNets [28] are perhaps the best known of these, an ANN model in which diffusive neurotransmitters are able to change the function of nodes within a network in a dynamic fashion. A more recent approach, termed artificial neural tissue [29], is also based around diffusive chemical gradients, but uses them to turn ON and OFF sparsely coded neural circuits in response to external cues. This is arguably the closest related work to our own, though prominent differences in our work include its relative simplicity (e.g., no developmental process), the use of comparatively small networks, and our emphasis on overlapping subnetworks and regulatory interactions between subnetworks.

D. Self-Modification in Artificial Biochemical Networks

The idea of self-modification has also previously been explored within computational models of biochemical networks. In [30], we considered an artificial biochemical network model in which a computational analog of a genetic network both expresses and modifies a computational analog of a metabolic network. This was effective at certain tasks, which benefited from decomposition; however, the complexity and appropriate parameterization of the system was a significant issue. Self-modification was also explored in [31] within the context of a model of mobile DNA applied to Boolean networks, in which the author found it to be beneficial in terms of access and stability of attractors. We have also explored a simpler model of chromatin remodeling in which subtasks are prespecified [8], and published initial results using the current approach [9] which have been significantly extended in this paper.

IV. Topological Self-Modification

A. Architecture

In this section, we describe how an analog of epigenetic remodeling can be implemented within a connectionist architecture. We use a fairly conventional RNN as a baseline architecture, general enough to be considered an abstract model of both a biological neural network and a genetic network.

Formally, this RNN architecture can be defined by the tuple $\langle N, L, \mathsf{In}, \mathsf{Out}\rangle$, where N is a set of nodes $\{n_0 \ldots n_{|N|} : n_i = \langle a_i, I_i, W_i\rangle\}$, where $a_i : \mathbb{R}$ is the activation level of the node, $I_i \subseteq N$ is the set of inputs used by the node, and W_i is a set of weights, where $0 \leq w_i \leq 1$, $|W_i| = |I_i|$. L is a set of initial activation levels, where $|L_N| = |N|$. $\mathsf{In} \subset N$ is the set of nodes used as external inputs. $\mathsf{Out} \subset N$ is the set of nodes used as external outputs.

Chromatin modules can be considered to be context-dependent switches that add or remove network components based on the network's current activation state. This form of topological self-modification can be introduced to the RNN model by adding extra nodes that act as Boolean switches, each adding or removing specified groups of nodes from the network based on the activation levels of one or more nodes.

The resulting artificial epigenetic network (AEN) architecture can be defined by the tuple $\langle N, S, L, \mathsf{In}, \mathsf{Out}\rangle$, where S is a set of switches $\{s_0 \ldots s_{|S|} : s_i = \langle a_i, I_i, W_i, C_i\rangle\}$, where $a_i^s \in \{0, 1\}$ is the activation level of the switch, $I_i^s \subseteq N$ is the set of inputs to the switch, W_i^s is the set of weights, where $0 \leq w_i \leq 1$, $|W_i| = |I_i|$, and $C_i^s \subseteq N$ is the set of nodes controlled by the switch. The other variables are as defined for the RNN. Nodes N and switches S both use a sigmoid function. In the case of nodes, the activation level a_i is the output of the sigmoid function applied to the weighted sum of its input activations. For switches, a threshold of 0.5 is applied.

Algorithm 1 Training AENs With NSGA-II

$P \leftarrow \{\}$
for $popsize$ times **do** ▷ initialize population
  $P \leftarrow P \cup \{\text{new random AEN}\}$
end for
for $maxgen$ times **do**
  for each $p \in P$ **do** ▷ evaluate population
    EVALUATE(p) ▷ see Algorithm 2
  end for
  $P \leftarrow \text{RANK}(P)$ ▷ NSGA-II style ranking [32]
  $P \leftarrow \{p_0, \ldots, p_{popsize/2}\}$ ▷ remove lower ranks
  $P' \leftarrow P$
  repeat ▷ breed child population
    $p1, p2 \leftarrow \text{SELECT}(P)$ ▷ rank-based selection
    $child \leftarrow \text{RECOMBINE}(p1, p2)$
    $child \leftarrow \text{MUTATE}(child)$
    $P' \leftarrow P' \cup \{child\}$
  until $|P'| = popsize$
  $P \leftarrow P'$ ▷ replace with child population
end for

If the output of the sigmoid function is less than this value, the activation level a_i^s of the switch is 1; otherwise, it is 0. If $a_i^s = 1$, then the switch has no effect upon the network. If $a_i^s = 0$, then the activation levels of its controlled nodes are set to 0, i.e., $\forall n_i \in C_j^s$, $a_i = 0$. In effect, these nodes are removed from the network.

Note that the network uses a sparse encoding, i.e., zero-weighted edges are not included in the model. Compared with a fully connected network, this is a more appropriate model of the pattern of connectivity seen within most genetic networks.

B. Training

We train both the RNN and AEN models using an evolutionary algorithm. Traditional neural network training methods, such as backpropagation, do not readily generalize to nonstandard architectures. Evolutionary algorithms, by comparison, are relatively flexible in this respect. They are also less sensitive to local optima, and are able to optimize both the parameters of individual nodes and the topology of the network. Since biological evolution is the mechanism responsible for designing biological genetic networks, it is particularly fitting to use an evolutionary algorithm to train a connectionist architecture that is motivated by genetic networks.

The nondominated sorting genetic algorithm, version II (NSGA-II) [32] is used for the experiments reported in this paper. This is a multiobjective evolutionary algorithm, allowing solutions to be evaluated with respect to more than one objective. Algorithm 1 gives an outline of NSGA-II and describes how it is used to train AENs.

C. Encoding

Given the close relationship between biological evolution and genetic networks, there is value in considering how genetic networks are encoded in biological systems, since this is known to have a significant bearing on their evolvability.

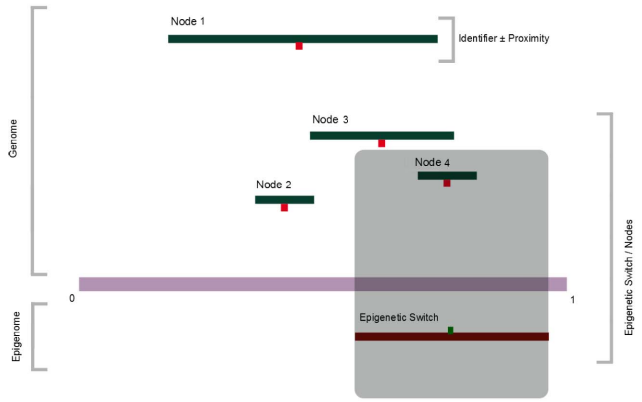

Fig. 1. Illustration of the indirect encoding between the nodes and switches of the network. In this example, the single switch interacts with genes 3 and 4, modulating their functionality.

A notable aspect of this paper is that we use a low-level network encoding in which connections between networks nodes are defined indirectly. Hence, during evolution, the connections, I_i, I_i^s, and C_i^s, are not represented by absolute node identifiers, but by locations within an indirect reference space. Several properties of the biological encoding of genetic networks motivate this approach.

First, gene–gene interactions are positionally independent, meaning that a gene retains its function irrespective of its position within a chromosome. This means that the positional changes due to biological recombination and mutation events preserve the existing structure of the genetic network. By comparison, when positionally sensitive encodings are used in evolutionary algorithms, ordering changes are generally disruptive, leading to child solutions with poor fitness [33], [34].

Second, and related to this, biological components recognize one another based upon their physicochemical properties. In effect, this physicochemical space is used as an indirect reference system in which genes, and other biochemical components, address one another. This observation has motivated a number of positionally independent encodings based upon the use of indirect reference spaces, including our own previous work on implicit context representation [34], and the template matching approach used in some computational models of genetic networks [35], [36].

Third, biological genetic networks display epistatic clustering [37], such that the genes that encode interacting gene products are often found located together within the genome [38]. This means that genetic pathways tend to be encoded in contiguous regions of DNA. From an evolutionary perspective, this leads to compartmentalization, which in turn promotes evolvability [39]. It also means that the winding and unwinding of chromatin modules tends to affect distinct subnetworks, arguably providing a less disruptive means of regulating biological function.

For simplicity, we use a 1-D reference space in which each node and switch has a location in the range [0, 1]. The inputs to a node or switch are defined as a continuous interval within this range. Furthermore, this interval overlaps with the

location of the node or switch. Hence, nodes and switches located proximally within the reference space are encouraged to interact, generating an effect similar to epistatic clustering. In particular, this means that the epigenetic switches will tend to operate upon complete subnetworks. It should be noted that this results in biases in the network landscape, since some patterns of connectivity are more likely to occur than others. Nevertheless, our initial experiments showed that a lack of epistatic clustering results in poor performance, and that this outweighs any issues associated with network shape bias.

Before being executed, an encoded network is mapped into the directly connected form defined in Section IV-A. Consequently, the indirect encoding does not lead to a performance penalty during execution. There is an overhead associated with this mapping, but it is small compared with the execution time.

V. Task Definitions

We hypothesize that topological self-modification will be useful in situations where task decomposition is not apparent *a priori*, where tasks are overlapping, and/or where there is a need to switch often between different behaviors. The control of dynamical systems is a class of problems that exhibits all of these characteristics. In particular, we focus on three interesting problems in dynamical systems control: 1) state-space targeting in a numerical dynamical system; 2) balancing a system of coupled inverted pendulums; and 3) controlling transfer orbits in a gravitational system. These are all challenging to solve, and each has qualitatively different dynamics. Although these kind of systems are traditionally controlled using analytical feedback methods [40], [41], our approach reflects the methodology of previous computational intelligence applications, such as [42] and [43], which do not require *a priori* knowledge about the state space of the system under control.

A. State-Space Targeting in a Numerical Dynamical System

This task involves controlling a trajectory so that it moves back and forth between two boundary points in Chirikov's standard map. This is a numerical dynamical system that models the behavior of a large class of conservative dynamical systems that have coexisting chaotic and ordered dynamics. While the exact definition of the task is to some extent arbitrary, it demonstrates the general concept of trajectory targeting in a complex state space.

Chirikov's standard map [44] is defined within the unit square by the following system of difference equations:

$$\begin{aligned} x_{n+1} &= (x_n + y_{n+1}) \bmod 1 \\ y_{n+1} &= y_n - \frac{k}{2\pi} \sin(2\pi x_n). \end{aligned} \tag{1}$$

For low values of k, the dynamics of the system are ordered, with initial points converging to cyclic orbits which remain bounded to small intervals on the y-axis. As k increases, islands of chaotic dynamics begin to appear. As k increases further, these begin to dominate the upper and lower regions of the map, with a band of ordered dynamics remaining in the central region. This central band prevents trajectories traversing from the top to the bottom of the map (and vice versa) until sufficient chaotic islands have appeared at $k_c \approx 0.972$. After this, it is possible for a trajectory to traverse the y-axis of the map by following the natural dynamics of the system. However, this occurs at a very slow rate. For example, when $k = 1.1$, the median transit time is 64 000 iterations of (1), with 27% of trajectories not reaching the other side within 10^6 iterations [30].

TABLE I
Sensory Inputs for the Standard Map Task. Each Is Mapped to the Range [0, 1]

Input	Variable	Range
0	Distance to target	0,1.25
1	x position	0,1
2	y position	0,1

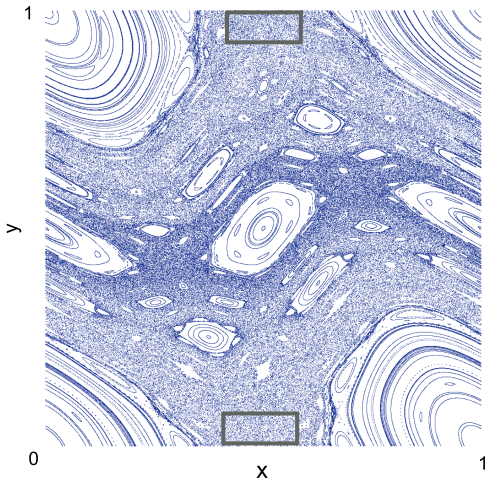

Fig. 2. Chirikov's standard map showing both ordered and chaotic dynamics. The top and bottom regions, which are used as initial and target regions for the control task, are shown as gray boxes.

The aim of this task is to create controllers that can guide a trajectory from a region at the bottom of the map to a region at the top of the map, and then back again, in the least amount of time. These regions are defined as $x[0.475, 0.525]$, $y[0.975, 1]$ for the top region and $x[0.475, 0.525]$, $y[0, 0.025]$ for the bottom region. A traversal between these regions can be seen in Fig. 8. The controller receives the inputs shown in Table I at each time step, and exerts control by modulating the parameter k within the interval [1.0, 1.1]. This results in a small perturbation to the trajectory. In previous work using this map [9], [30], we have observed that different control interventions are required when moving through regions with different dynamical characteristics (e.g., chaotic, ordered, and mixed). In general, it is not obvious when these transitions in behavior should occur. This makes the problem challenging from a control perspective.

In order to generate a fair estimate of a controller's ability to guide trajectories, the task is repeated ten times with different starting positions randomly chosen within the regions shown in Fig. 2. Two objective values are then calculated for each controller: 1) the mean trajectory length when moving from the bottom to the top of the map and 2) the mean trajectory

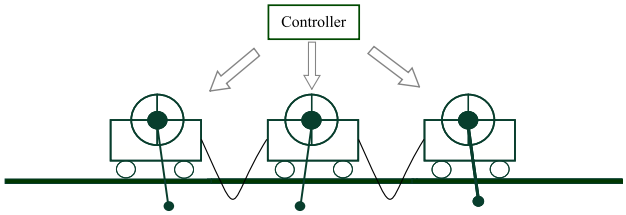

Fig. 3. Coupled inverted pendulums, illustrating a single controller controlling three tethered carts.

length when moving from the top to the bottom of the map. Controllers, which are not able to traverse the map in either direction within a limit of 1000 time steps, are assigned an arbitrary large value of 1000 for the corresponding objective. This penalty will also be applied if the trajectory in a particular direction moves beyond the y-axis bound of [0, 1].

B. Balancing a System of Coupled Inverted Pendulums

Balancing an inverted pendulum is a classic problem in control theory, and a proxy for various real world control problems, such as bipedal locomotion and missile control [45]. In this task, we consider a harder formulation of this problem that involves using multiple pendulums, mounting them on movable carts, and then coupling the carts together (Fig. 3). The aim is to move the carts in such a way that all the pendulums become upright, and then remain upright for a predetermined amount of time. This can be interpreted as a state-space targeting task, in which a trajectory must be guided from a stable equilibrium state (all pendulums pointing downward) to an unstable equilibrium state (all pendulums pointing upward), followed by a stabilization task that involves maintaining the trajectory at the unstable point.

In our formulation, the system has between one and five carts, arranged in a line. When the number of carts is greater than 1, they are connected to their nearest neighbor(s) with inelastic tethers. Each cart is controlled using an actuator with a differential input, allowing it to move toward or away from its neighbors based on the difference of its two inputs. Table II shows the physical parameters of the model. Each cart is controlled independently using the same evolved controller. The controller has access to a number of state variables. These are described in Table III, and their application to the cart can be seen in Fig. 4. The fitness of the controller is defined as an aggregate function over all the carts of the amount of time each pendulum spends in the upright position and scaled between [0, 1]. Hence, if a system contains three pendulums, of which one remains upright throughout simulation with the other two remaining hanging from the carts, a fitness of 0.33 would be assigned. If all pendulums are upright throughout the simulation, a fitness of 1 will be assigned and if all pendulums remain hanging throughout the simulation, a fitness of 0 will be assigned.

C. Controlling Transfer Orbits in a Gravitational System

As a more concrete example of control in a conservative dynamical system, we consider a formulation of the N-body problem in which the aim is to guide a trajectory through

TABLE II
PHYSICAL PARAMETERS OF THE COUPLED INVERTED PENDULUMS TASK

Parameter	Value
Pendulum length	$0.5m$
Max. positive acceleration	7.0 ms^{-2}
Min. positive acceleration	8.5 ms^{-2}
World width	$2m$
Tether length	$0.35m$
Proximity sensor range	$1.0m$
Cart width	$0.1m$
Time steps (t)	4000

TABLE III
SENSORY INPUTS USED FOR THE INVERTED COUPLED PENDULUMS TASK. THE VALUES ARE RESCALED TO [0, 1] BEFORE THEY ARE USED AS INPUTS TO A NETWORK

ID	Sensor Name	System to sensor mapping
S_0	Pendulum Angle 0	$\phi \in [0,0.5\pi] \rightarrow$ [127, 0], 0 else
S_1	Pendulum Angle 1	$\phi \in [1.5\pi,2\pi] \rightarrow$ [0,127], 0 else
S_2	Pendulum Angle 2	$\phi \in [0.5\pi,\pi] \rightarrow$ [127, 0], 0 else
S_3	Pendulum Angle 3	$\phi \in [\pi,1.5\pi] \rightarrow$ [0,127], 0 else
S_4	Proximity 0	Distance left $\rightarrow$ [0,127]
S_5	Proximity 1	Distance right $\rightarrow$ [0,127]
S_6	Cart Velocity 0	$v \in$ [-2,0] $\rightarrow$ [127,0], 0 else
S_7	Cart Velocity 1	$v \in$ [0,2] $\rightarrow$ [0,127], 0 else
S_8	Angular Velocity 0	$w \in [-5\pi,0] \rightarrow$ [127,0], 0 else
S_9	Angular Velocity 1	$w \in [0,5\pi] \rightarrow$ [0,127], 0 else
A_i	Actuators 0	$A_i \in$[0,127], for $i \in$ 0,1
u	Motor Control 0	$2(A_0/127 - A_1/127) \rightarrow$ [0,1]

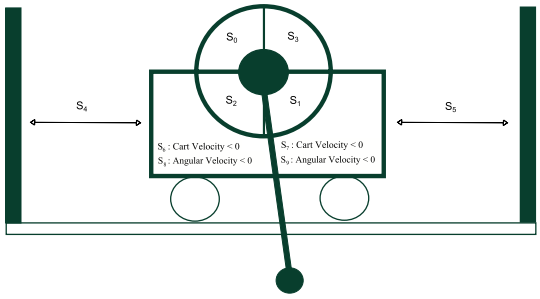

Fig. 4. Sensory inputs for a cart within the coupled inverted pendulums task.

a system of planetary bodies. Gravitational systems with more than two bodies exhibit the kind of mixed chaotic and ordered dynamics seen in Chirikov's standard map. This presents a significant challenge when controlling spacecraft, since efficient orbital transfers require traversal of these complex dynamical regimes. In the example, we consider that there are four bodies: a spacecraft and three planets. The aim is to guide the trajectory of the spacecraft so that it moves repeatedly between two of the three planets. It is required to do this in the least amount of time, and by using the least amount of fuel. It can do this either by taking a direct path (see Fig. 5), or by sling-shotting around the third, more massive, planet. Either way, the spacecraft is under the influence of gravity from all three planets. To make simulation time tractable, the positions of the planets remain fixed. See Tables V and VI for model parameters.

The force exerted on the spacecraft is calculated using (2), where m is the mass of a body and q is a 3-D vector

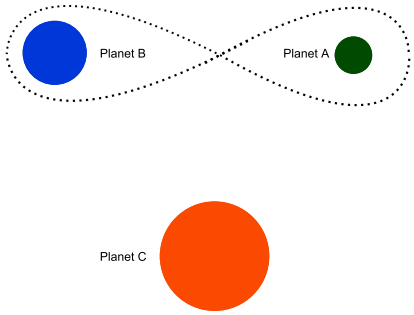

Fig. 5. System of gravitational bodies used in the orbital control task. The aim is to guide an orbit so that it transitions repeatedly between planets A and B, while also under gravitational influence from planet C.

(j specifies an instance of a body, and k represents an instance which is not equal to the first, i.e., force i is the sum of all other forces k which are not force i)

$$m_j q_j = G \sum_{k \neq j} \frac{m_j m_k (q_k - q_j)}{|q_k - q_j|^3}. \quad (2)$$

From this, the acceleration of the spacecraft due to the gravitational forces of the other planets can be calculated using Newton's second law of motion. The equations are simulated using leapfrog integration, which is well suited to the problems of orbital mechanics due to its symplectic nature and time reversibility [46], [47].

The controller has access to the following state variables: 1) distance to target; 2) position of target; 3) spacecraft acceleration; and 4) spacecraft position. These are mapped to nine inputs (see Table IV). The target is determined by the spacecraft's current position, i.e., planet A if it is in orbit around planet B, and vice versa. The controller exerts control by adding thrust in one or more of the three dimensions, subject to an acceleration limit of ± 25 ms^{-2}.

Two objective values are calculated for each controller: 1) the cumulative time taken to move between planets A and B over the course of the simulation and 2) the cumulative thrust used to maneuver the spacecraft. The spacecraft is assumed to be in a valid orbit when it is between 1×10^5 and 2×10^5 m from a planet's center of mass. If the spacecraft moves within 2×10^4 m of the planet's center of mass, it is assumed to have collided and the controller is assigned a fitness value of 0 for planetary hops (corresponding to the lowest possible performance) and positive infinity for the fuel used (corresponding to the worst possible performance). The same penalties are applied if it takes more than 8000 s to transition between the two planets. In initial experiments, it was found that evolution disproportionately favored solutions, which minimize the fuel usage objective by remaining relatively static. To discourage this behavior, we introduced a third objective, the product of the first two objectives. This especially penalizes solutions that do not achieve at least one orbital transition.

VI. Results and Analysis

For each of these tasks, the aim is to evolve a closed-loop controller that can guide the dynamics of the system in the specified manner. At each time step, the state of the controlled system is fed back to the controlling AEN or RNN by setting the activation levels of nodes in the input set (In). See Algorithm 2 for details. There is one input node for each of the sensory inputs given in the task definition. Control is then exerted by copying the activation levels of nodes in the output set (Out) to the governing parameters of the controlled system, scaling as appropriate.

TABLE IV
Sensory Inputs Used for the Orbital Control Task

Input	Variable	Range
0	Distance to target (m)	0, 2 x 10^6
1	Target (x position)	-1.5 x 10^6, 1.5 x 10^6
2	Target (y position)	-1.5 x 10^6, 1.5 x 10^6
3	Target (z position)	-1.5 x 10^6, 1.5 x 10^6
4	Spacecraft acceleration (x plane)	-50, 50
5	Spacecraft acceleration (y plane)	-50, 50
6	Spacecraft acceleration (z plane)	-50, 50
7	Spacecraft position (x plane)	-1.5 x 10^6, 1.5 x 10^6
8	Spacecraft position (y plane)	-1.5 x 10^6, 1.5 x 10^6
9	Spacecraft position (z plane)	-1.5 x 10^6, 1.5 x 10^6

TABLE V
Initial Positions and Masses of the Bodies Within the Orbital Control Task

Body	x, y, z position (m)	Mass (kg)
Planet A	1 x 10^6, 0, -1 x 10^5	5.972 x 10^{22}
Planet B	1 x 10^5, 0, -1 x 10^6	5.972 x 10^{23}
Planet C	0, -2 x 10^6, 0	5.972 x 10^{24}
Spacecraft	0, 0, 0	2000

TABLE VI
Physical Parameters for the Orbital Control Task

Parameter	Variable
Gravitational Constant (G)	6.67384 x 10^{-11} N $(m/kg)^2$
Time Steps	50000s
Integration step	0.02s
Spacecraft max. acc.	$\pm$ 25 ms^{-2}

A. Standard Map

Both AENs and RNNs were evolved to control trajectories within Chirikov's standard map. A population size ($popsize$) of 200 was used, with the evolutionary process allowed to run for 100 generations ($maxgen$). Mutation and crossover rates were 0.05 and 0.5, respectively. In the initial generation, networks were created with lengths of between 10 and 20 nodes. In addition, initial AENs were seeded with 3–5 switches. Solution lengths were otherwise free to vary during evolution.

Since EAs are nondeterministic algorithms, 50 independent runs were carried out for each problem instance to give a fair portrayal of expected performance. Fig. 6 shows the distributions of fitness for the best solutions from these 50 runs, for both AENs and RNNs. This shows that the AEN model leads to better solutions both on average ($p = 2.04 \times 10^{-4}$) and overall. The best performing AEN controller traverses

Algorithm 2 Evaluating an AEN on a Control Task

initialize control task
$a \leftarrow L$ ▷ initialize AEN state
repeat
  $cout \leftarrow$ state variables from controlled system
  In $\leftarrow$ SCALE($cout$) ▷ scale inputs to $[0, 1]$
  for $i \in \{0, \ldots, |S|\}$ **do** ▷ update switches
    $a_i^s \leftarrow$ SIGMOID$(I_i^s \cdot W_i^s)$
    if $a_i^s < 0.5$ **then** ▷ modify topology
      for each $j \in C_i^s$ **do**
        $a_j \leftarrow 0$
      end for
    end if
  end for
  for $i \in \{0, \ldots, |N|\}$ **do** ▷ update nodes
    $a_i \leftarrow$ SIGMOID$(I_i \cdot W_i)$
  end for
  $cin \leftarrow$ SCALE(Out) ▷ scale outputs to range
  modify controlled system according to cin
until control task finished or timed-out
$fitness \leftarrow$ progress on control task objectives

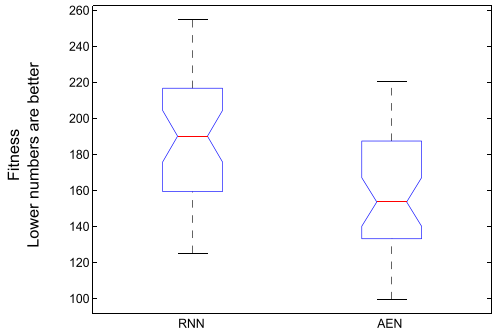

Fig. 6. Fitness distributions for AEN and RNN-based controllers, showing the average number of steps required to traverse the standard map in each direction by the best controllers from 50 sequential runs of NSGA-II.

the map in each direction in ∼99 steps, on average, which is ∼20 steps faster than the best RNN. Fig. 7 shows an example of a controlled trajectory.

Fig. 8 shows the change in mean and maximum controller fitness over time for the AEN and RNN runs. It is evident that the fitness for RNN-based controllers begins to converge considerably earlier than for the AEN-based controllers. It is also notable that the evolution of the best AEN controller is much smoother, in terms of fitness changes, than for the best RNN controller. This may be an indication of better evolvability for the AEN model, allowing controllers to evolve through gradual frequent changes rather than large infrequent changes.

Analysis of the dynamical behavior of evolved controllers gives some insight into these differences in performance. First, it is notable that all but one of the evolved AEN networks used their switches to alter the topology of the network during execution. This suggests that there is strong selective pressure toward using topological modification. Second, significant differences can be seen between the

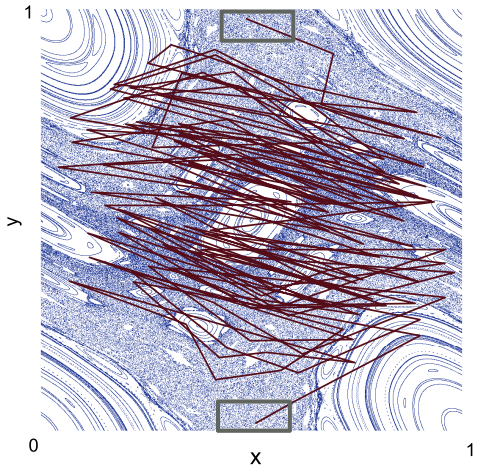

Fig. 7. Example of an AEN controlling a trajectory within Chirikov's standard map, traversing from a region at the bottom to a region at the top in 94 steps.

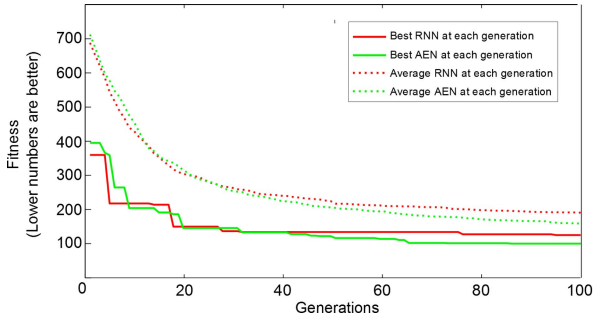

Fig. 8. Best and average performance of both the AEN and RNN controllers at each generation for the standard map task.

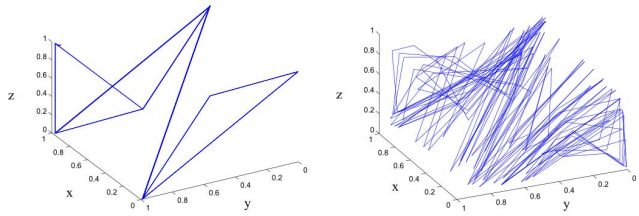

Fig. 9. Phase portraits showing control responses of the AEN (left) from Fig. 10, and a representative RNN (right).

phase spaces of AEN and RNN controllers. For example, Fig. 9 shows the representative phase spaces reconstructed using time-delay embedding [48] from the outputs of an AEN and RNN controller, respectively. It can be seen that the AEN phase space is well conserved, apparently following attractors with well-defined topological characteristics as it navigates the map. The dynamical behavior of the RNN controller, by comparison, is relatively poorly conserved, indicative of a less stable attractor structure. This suggests that the topological modification may play a role in stabilizing different attractors as the controller navigates through the different dynamical regimes exhibited by the map.

To understand how topological self-modification is used by evolved AENs, the smallest working example of an

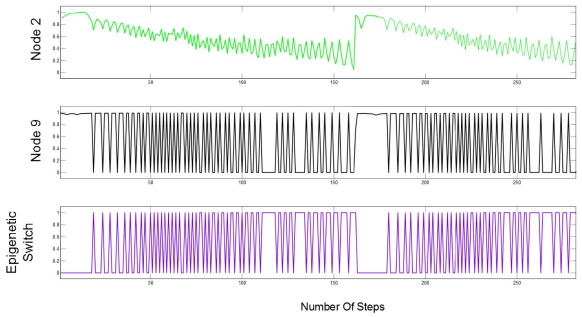

Fig. 10. Time series of expression states of the nodes and switches of the smallest working example of an AEN controlling traversal of Chirikov's map in both directions.

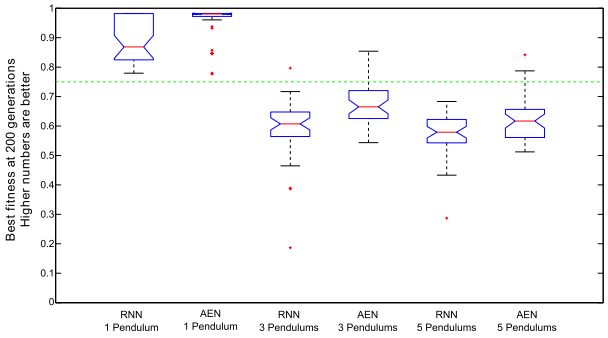

Fig. 11. Fitness distributions over 40 runs for RNN- and AEN-based controllers solving the 1, 3, and 5 coupled inverted pendulum problems. The means are significantly different in all cases ($p = 0.029$, 7.5×10^{-5}, and 0.01, respectively, using the Wilcoxon rank sum test). Dashed horizontal line: approximate fitness required to solve the task. Improvements in stabilization time occur beyond this point.

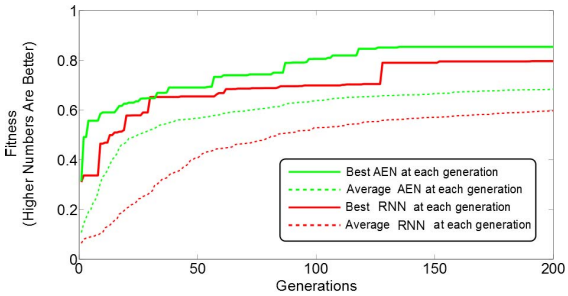

Fig. 12. Best and average performance of both the AEN and RNN controllers at each generation for the three-cart variant of the coupled inverted pendulums task.

AEN controller was first analyzed. Fig. 10 shows the time series of expression values for nodes and switches within this network. In this case, it is apparent that the single switch present in this network is not used simply to transition between subtasks. Rather, during much of the control period, it generates an oscillatory pattern of expression which turns ON and OFF one of the network's other nodes, thereby affecting the network's dynamics. This points to roles for topological modification beyond task decomposition.

However, it was also noticed that the switch elements of many evolved AENs could be used to manually switch between dynamical behaviors. For instance, by forcing a switch to remain ON, it is often the case that the trajectory will then remain within a certain dynamical region of the map. This suggests that the switches are used to move between different attractors during the course of controlling trajectories. It also points to an emergent role of these switches as a means for inferring task decomposition and allowing external control of transitions between subtasks.

It is also notable that the AEN-based controllers are significantly smaller than the RNNs, using an average of seven nodes, compared with an average of ten nodes in the RNN-based controllers. In addition to offering a small benefit in terms of efficiency, this may indicate that smaller overlapping subnetworks are used at different stages of the control task, rather than the single monolithic network used by an RNN.

B. Coupled Inverted Pendulums

A similar approach was used to evolve AEN- and RNN-based controllers for the coupled inverted pendulums problem. Recognizing the greater difficulty of this task, the generation limit was raised to 200, and initial networks were generated with between 15 and 25 nodes.

Fitness distributions for one-, three-, and five-cart variants of the problem are shown in Fig. 11. Unsurprisingly, control of a single cart is significantly easier than multiple carts, and effective control could be achieved using both architectures. Nevertheless, AEN-based controllers were able to balance the pendulum more consistently and, on average, significantly faster than the RNN-based controllers. For the multicart problems, RNN-based controllers were able to solve the three-cart variant only once out of 40 runs, and were unable to solve the five-cart variant. AEN-based controllers were also challenging to evolve; however, substantially more of these solutions were found for the three-cart problem, and several AEN-based controllers were also found for the five-cart variant.

Fig. 12 shows the change in mean and maximum controller fitness over time for the AEN and RNN runs when solving the three-cart problem (although this is also representative of the one- and five-cart versions). It can be seen that the AEN-based controllers evolve much faster and converge toward a higher fitness value. A smoother pattern of evolution is again seen for the best solution, with the RNN exhibiting large step changes. This supports the hypothesis that the AENs are more evolvable.

Analysis of evolved controllers again suggests that AENs solve the problem in a different way to RNNs. First of all, this can be seen in their use of sensory inputs (see Table III), with all AEN solutions using inputs 0 (pendulum angle sensor) or 9 (angular velocity), and the majority of RNNs using inputs 2 (pendulum angle), 3 (pendulum angle), 7 (cart velocity), and 8 (angular velocity) to solve the task. Neither favored inputs 4, 5, or 6. This suggests that the two different architectures are biased toward exploring different input–output mappings, with the AENs presumably more able to express mappings that result in higher fitness. Differences in behavior can also be seen in the reconstructed phase spaces of AEN and RNN controllers (see Figs. 13 and 14).

Figs. 15 and 16, respectively, show the network topology and time series of an AEN evolved to solve the

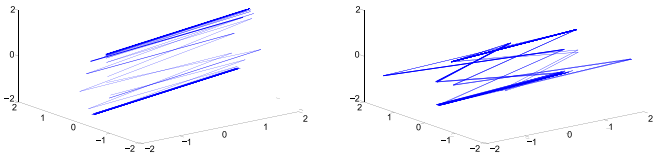

Fig. 13. Phase portraits of representative control responses of an AEN, showing the swinging behavior (left) and the balancing behavior (right).

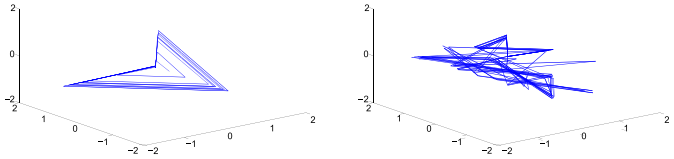

Fig. 14. Phase portraits of representative control responses of an RNN, showing the swinging behavior (left) and the balancing behavior (right).

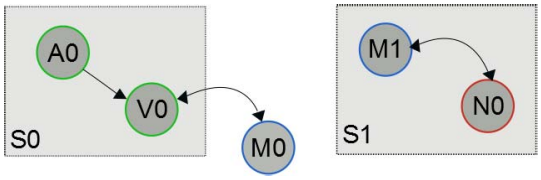

Fig. 15. Typical minimum working example AEN evolved for the three pendulum task. Only the nodes and the switches, which are required to generate the optimal behavior, are shown. S0 and S1 switches, which when active, deactivate the genes within their bounds. V0 refers to the angular velocity in the top quadrants. A0 refers to the angular position in the top-left quadrant. N0 is a node with no external input, but is required to create the oscillatory behavior. M0 and M1 are the motor control for the cart.

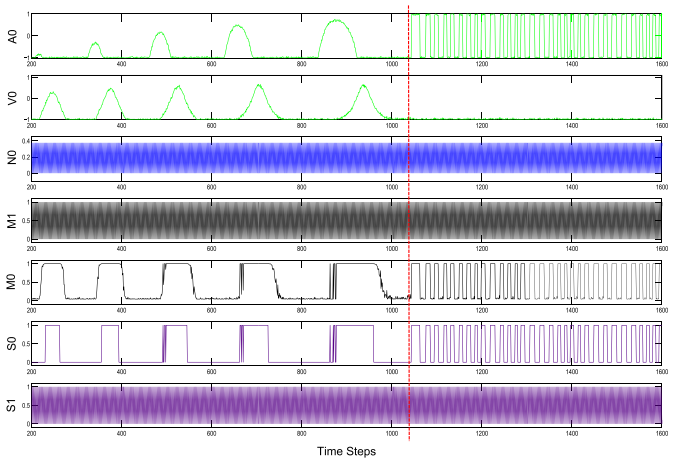

Fig. 16. Time series of expression states of the nodes and switches in the AEN shown in Fig. 15 solving the three-cart coupled inverted pendulums problem. Dashed vertical line indicates where the pendulum stabilizes in the upright state. Nodes that do not contribute to the behavior of the controller have been removed.

three-cart problem. This solution used two switches. One of these is activated when a pendulum is in the upward part of its swing, causing a transition between swinging and stabilizing behaviors. The other functions as an oscillator. Hence, we see both of the behaviors observed in the standard map controllers.

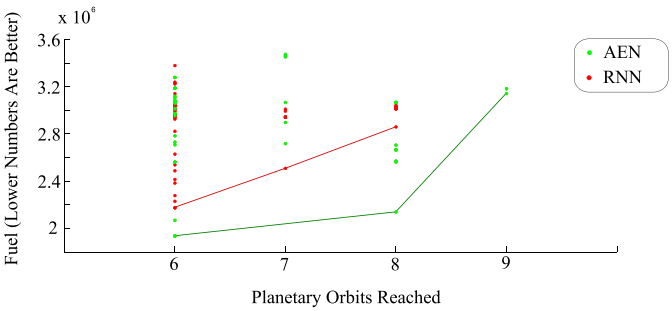

Fig. 17. Performance of AEN and RNN controllers on the orbital transfer problem, showing (red and green lines) the Pareto fronts achieved by each controller type.

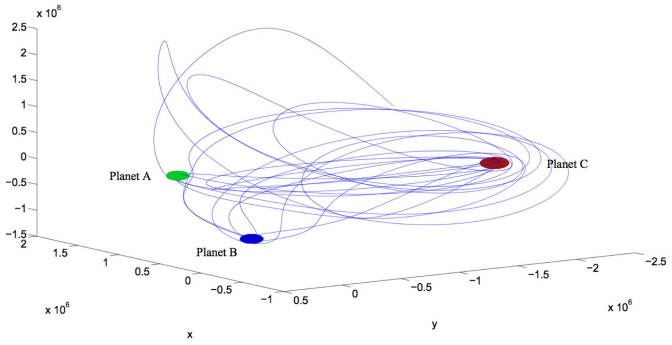

Fig. 18. Trajectory of the spacecraft generated by the fittest AEN, achieving nine planetary hops. It can be seen that this controller utilizes the gravitational slingshot effect.

Again, the first switch can be used to manually transition between the controller's behaviors; for instance, turning this ON during the stabilization phase causes the pendulum to transition to the swinging phase and remain there.

C. Controlling Transfer Orbits in a Gravitational System

Controlling transfer orbits was the hardest of the three problems, requiring a population of 500 and a generation limit of 200. Other parameters were the same as for coupled inverted pendulums problem.

Fig. 17 shows the Pareto front of solutions generated by 40 subsequent evolutionary runs for both AENs and RNNs, showing the tradeoffs between orbital repetitions and fuel used by each controller. The Pareto front for AEN-based controllers is significantly further to the bottom-right, indicating solutions were found that could travel further and with less fuel than the RNNs.

It was observed that the majority of evolved controllers used a gravitational slingshot around planet C to conserve fuel while transitioning between planets A and B. Fig. 18 shows the behavior of one of the best controllers, guiding the trajectory over the course of nine planetary transitions.

Topological self-modification was seen to occur in 36 of the 40 evolved AEN controllers, again suggesting a strong selective pressure toward making use of this behavior. In the remaining four controllers, the switches remained permanently OFF. In all instances where topological modification was applied, its effect included changing the expression state of

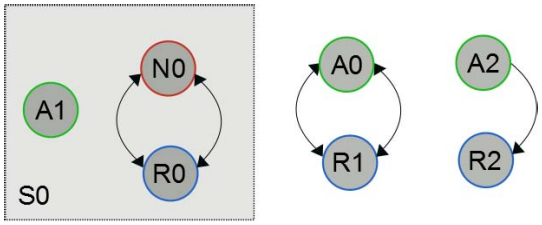

Fig. 19. Structure of an AEN, which was able to perform nine orbital hops in succession. Only the nodes and switches, which are required to generate the behavior, are shown. A0 refers to the distance to the target, and A1 and A2 refer to the absolute position of the spacecraft. N0 is a processing node. R0–R2 are the outputs of the network, which specify the acceleration of the spacecraft.

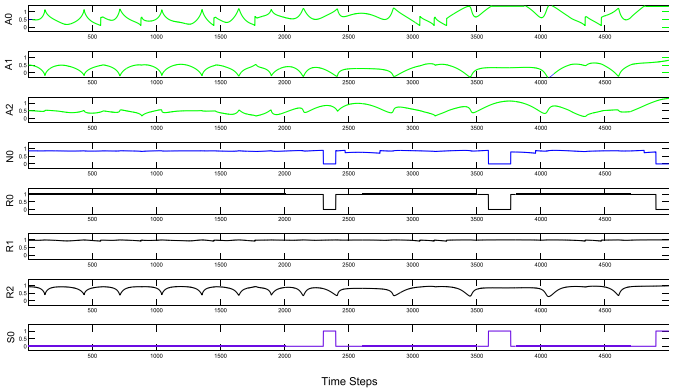

Fig. 20. Expression of each node and switch of the AEN shown in Fig. 19 over 50 000 times steps (sampled at one every ten steps) with nonfunctional nodes and switches removed. It can be seen that the switch is modifying the topology of the network dynamically.

one of the output nodes—in effect, forcing an abrupt direction change within the trajectory.

Fig. 20 shows a time series view of node and switch activations in one of the best evolved AEN controllers. It can be seen that the switch changes state on only three occasions during the period of control. This suggests that the mechanism is being used in a sensitive manner, inducing only small infrequent changes within the dynamics of the network. In this case, the switch is regulated by node A2, which indicates the spacecraft's position in the z plane. It becomes active when the trajectory reaches its extremal position, guiding the spacecraft back into a close orbit.

VII. Conclusion

In this paper, we investigated the potential benefits of introducing topological self-modification to RNN architectures, in the form of an AEN. The AEN approach is motivated by the process of chromatin modification within genetic regulatory networks, particularly the manner in which genes are able to regulate transcriptional access to other genes through chromatin modification. This, in effect, is analogous to adding or removing nodes to or from the network.

The AEN approach was applied to three different dynamical control tasks, using evolutionary algorithms to design the topology and parameters of the networks. A clear pattern seen in these experiments is that the AEN model allows the evolution of better solutions than a conventional RNN model, both in terms of average performance and ability to express more general solutions.

We propose several hypotheses for why this is the case. First, it seems likely that the topological change stabilizes attractors, making it easier for a controller to maintain a stable behavior. This reflects biological understanding of the role of chromatin modification in achieving cell specialization. Second, the topological change can lead to rapid behavioral change, presumably faster than that which can be achieved by following the natural dynamics of a network with fixed topology. This is likely to be beneficial in controllers that require rapid responses to their environment. Third, we see that the evolved AENs express behaviors, which are not seen in evolved RNNs, presumably because the topological change makes these behaviors easier to discover. These behaviors, in turn, appear particularly useful for the kind of control problems we have considered in this paper.

By considering how controllers evolved over time, it also became evident during our experiments that the AEN controllers evolve more smoothly than the conventional RNN architectures. This, in turn, may suggest that the AENs are more evolvable than the RNNs and, therefore, more suitable for use with evolutionary algorithms. It is known that evolvability and robustness are closely related system properties, so this may also be related to an AEN's ability to maintain different stable behaviors and robustly transition between them.

A further benefit of the approach was discovered during post hoc analysis of the evolved controllers, where it was noted that manually changing the activation state of the topological switches led the controllers to transition between different phases of the control task. This gave insights into the natural decomposition of the tasks, and could potentially be used as a general mechanism for exploring and understanding the internal structure of problems. This is something we plan to explore further in the future work.

Although this paper has focused on solving computational problems, it is feasible that computational models such as this could also be used to develop better understanding of epigenetic mechanisms in biology. Epigenetics is a relatively new field of study, and experimental limitations make it difficult to infer general principles from biological data alone. Computational models could help to fill this gap by allowing the exploration of systems-level properties, in much the same way that Boolean network models have helped to understand genetic networks. As a start, we have begun to look at more detailed computational models of epigenetic processes that model the spatiotemporal behavior of chromatin modifying protein complexes.

The results presented in this paper show the potential for using self-modifying processes within connectionist architectures. However, an AEN is only one way of achieving this, and in the future work, we also plan to investigate a broader range of self-modifying connectionist models. These need not be limited to switching ON and OFF different parts of an existing network. They could also, for instance, be used to dynamically create new nodes or subnetworks using processes analogous to development or growth.

REFERENCES

[1] R. A. Jacobs, M. I. Jordan, and A. G. Barto, "Task decomposition through competition in a modular connectionist architecture: The what and where vision tasks," *Cognit. Sci.*, vol. 15, no. 2, pp. 219–250, Apr. 1991.

[2] G. Auda and M. Kamel, "Modular neural networks: A survey," *Int. J. Neural Syst.*, vol. 9, no. 2, pp. 129–151, 1999.

[3] P. Melin and O. Castillo, "Modular neural networks," in *Hybrid Intelligent Systems for Pattern Recognition Using Soft Computing* (Studies in Fuzziness and Soft Computing), vol. 172. Berlin, Germany: Springer-Verlag, 2005, pp. 109–129.

[4] D. Meunier, R. Lambiotte, A. Fornito, K. Ersche, and E. T. Bullmore, "Hierarchical modularity in human brain functional networks," *Frontiers Neuroinformat.*, vol. 3, no. 37, 2009. DOI: 10.3389/neuro.11.037.2009

[5] H. Ling, S. Samarasinghe, and D. Kulasiri, "Novel recurrent neural network for modelling biological networks: Oscillatory p53 interaction dynamics," *BioSystems*, vol. 114, no. 3, pp. 191–205, Dec. 2013.

[6] M. A. Lones, A. P. Turner, L. A. Fuente, S. Stepney, L. S. D. Caves, and A. M. Tyrrell, "Biochemical connectionism," *Natural Comput.*, vol. 12, no. 4, pp. 453–472, Dec. 2013.

[7] T. P. Rasmussen, "Embryonic stem cell differentiation: A chromatin perspective," *Reproductive Biol. Endocrinol.*, vol. 1, no. 1, p. 100, Nov. 2003.

[8] A. P. Turner, M. A. Lones, L. A. Fuente, A. M. Tyrrell, S. Stepney, and L. S. D. Caves, "The incorporation of epigenetics in artificial gene regulatory networks," *BioSystems*, vol. 112, no. 2, pp. 56–62, May 2013.

[9] A. P. Turner, M. A. Lones, L. A. Fuente, S. Stepney, L. S. D. Caves, and A. Tyrrell, "The artificial epigenetic network," in *Proc. IEEE Symp. Ser. Comput. Intell. (SSCI)*, Singapore, Apr. 2013, pp. 66–72.

[10] C. L. Woodcock and R. P. Ghosh, "Chromatin higher-order structure and dynamics," *Cold Spring Harbor Perspect. Biol.*, vol. 2, no. 5, p. a000596, 2010.

[11] S. Huang, "Reprogramming cell fates: Reconciling rarity with robustness," *BioEssays*, vol. 31, no. 5, pp. 546–560, May 2009.

[12] A. R. Chowdhury, M. Chetty, and N. X. Vinh, "Inferring large scale genetic networks with S-system model," in *Proc. 15th Annu. Conf. Genet. Evol. Comput.*, Amsterdam, The Netherlands, 2013, pp. 271–278.

[13] S. A. Kauffman, "Metabolic stability and epigenesis in randomly constructed genetic nets," *J. Theoretical Biol.*, vol. 22, no. 3, pp. 437–467, Mar. 1969.

[14] M. A. Lones, "Computing with artificial gene regulatory networks," in *Evolutionary Computation in Gene Regulatory Network Research* (Series in Bioinformatics), H. Iba and N. Noman, Eds. Hoboken, NJ, USA: Wiley, 2016.

[15] A. Roli, M. Manfroni, C. Pinciroli, and M. Birattari, "On the design of Boolean network robots," in *Applications of Evolutionary Computation* (Lecture Notes in Computer Science), vol. 6624, C. D. Chio *et al.*, Eds. Berlin, Germany: Springer-Verlag, 2011, pp. 43–52.

[16] W. Banzhaf, "Artificial regulatory networks and genetic programming," in *Genetic Programming Theory and Practice* (Genetic Programming Series), vol. 6, R. Riolo and B. Worzel, Eds. Norwell, MA, USA: Kluwer, 2003, pp. 43–61.

[17] P. J. Bentley, "Evolving fractal gene regulatory networks for graceful degradation of software," in *Self-Star Properties in Complex Information Systems* (Lecture Notes in Computer Science), vol. 3460, O. Babaoglu *et al.*, Eds. Berlin, Germany: Springer-Verlag, 2005, pp. 21–35.

[18] D. Floreano, P. Dürr, and C. Mattiussi, "Neuroevolution: From architectures to learning," *Evol. Intell.*, vol. 1, no. 1, pp. 47–62, Mar. 2008.

[19] G. Auda and M. Kamel, "Modular neural network classifiers: A comparative study," *J. Intell. Robot. Syst.*, vol. 21, no. 2, pp. 117–129, Feb. 1998.

[20] L. Medsker and L. C. Jain, *Recurrent Neural Networks: Design and Applications*. Boca Raton, FL, USA: CRC Press, 2010.

[21] V. R. Khare, X. Yao, B. Sendhoff, Y. Jin, and H. Wersing, "Co-evolutionary modular neural networks for automatic problem decomposition," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, vol. 3. Edinburgh, U.K., Sep. 2005, pp. 2691–2698.

[22] S. Whiteson, N. Kohl, R. Miikkulainen, and P. Stone, "Evolving soccer keepaway players through task decomposition," *Mach. Learn.*, vol. 59, no. 1, pp. 5–30, May 2005.

[23] T. Ash, "Dynamic node creation in backpropagation networks," *Connection Sci.*, vol. 1, no. 4, pp. 365–375, 1989.

[24] A. Forti and G. L. Foresti, "Growing hierarchical tree SOM: An unsupervised neural network with dynamic topology," *Neural Netw.*, vol. 19, no. 10, pp. 1568–1580, Dec. 2006.

[25] J. Schmidhuber, J. Zhao, and N. N. Schraudolph, "Reinforcement learning with self-modifying policies," in *Learning to Learn*, S. Thrun and L. Pratt, Eds. Norwell, MA, USA: Kluwer, 1998, pp. 293–309.

[26] F. Gruau, "Genetic synthesis of Boolean neural networks with a cell rewriting developmental process," in *Proc. Int. Workshop Combinations Genet. Algorithms Neural Netw. (COGANN)*, Baltimore, MD, USA, Jun. 1992, pp. 55–74.

[27] J. C. Astor and C. Adami, "A developmental model for the evolution of artificial neural networks," *Artif. Life*, vol. 6, no. 3, pp. 189–218, Jun. 2000.

[28] T. Smith, P. Husbands, A. Philippides, and M. O'Shea, "Neuronal plasticity and temporal adaptivity: GasNet robot control networks," *Adapt. Behavior*, vol. 10, nos. 3–4, pp. 161–183, 2002.

[29] J. Thangavelautham and G. M. T. D'Eleuterio, "Tackling learning intractability through topological organization and regulation of cortical networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 4, pp. 552–564, Apr. 2012.

[30] M. A. Lones *et al.*, "Artificial biochemical networks: Evolving dynamical systems to control dynamical systems," *IEEE Trans. Evol. Comput.*, vol. 18, no. 2, pp. 145–166, Apr. 2014.

[31] L. Bull, "Consideration of mobile DNA: New forms of artificial genetic regulatory networks," *Natural Comput.*, vol. 12, no. 4, pp. 443–452, Dec. 2013.

[32] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.

[33] X. Cai, S. L. Smith, and A. M. Tyrrell, "Positional independence and recombination in Cartesian genetic programming," in *Genetic Programming* (Lecture Notes in Computer Science), vol. 3905, P. Collet, M. Tomassini, M. Ebner, S. Gustafson, and A. Ekárt, Eds. Berlin, Germany: Springer-Verlag, 2006, ch. 32, pp. 351–360.

[34] M. A. Lones and A. M. Tyrrell, "Modelling biological evolvability: Implicit context and variation filtering in enzyme genetic programming," *BioSystems*, vol. 76, nos. 1–3, pp. 229–238, Aug./Sep. 2004.

[35] T. Reil, "Dynamics of gene expression in an artificial genome—Implications for biological and artificial ontogeny," in *Proc. 5th Eur. Conf. Artif. Life (ECAL)*, vol. 1674. 1999, pp. 457–466.

[36] W. Banzhaf, "On the dynamics of an artificial regulatory network," in *Advances in Artificial Life* (Lecture Notes in Computer Science), vol. 2801, W. Banzhaf, J. Ziegler, T. Christaller, P. Dittrich, and J. T. Kim, Eds. Berlin, Germany: Springer-Verlag, 2003, ch. 24, pp. 217–227.

[37] J. W. Pepper, "The evolution of evolvability in genetic linkage patterns," *BioSystems*, vol. 69, nos. 2–3, pp. 115–126, May 2003.

[38] D. Newth and D. G. Green, "The role of translocation and selection in the emergence of genetic clusters and modules," *Artif. Life*, vol. 13, no. 3, pp. 249–258, 2007.

[39] M. Conrad, "The geometry of evolution," *BioSystems*, vol. 24, no. 1, pp. 61–81, 1990.

[40] K. Pyragas, "Continuous control of chaos by self-controlling feedback," *Phys. Lett. A*, vol. 170, no. 6, pp. 421–428, Nov. 1992.

[41] F. J. Romeiras, C. Grebogi, E. Ott, and W. P. Dayawansa, "Controlling chaotic dynamical systems," *Phys. D, Nonlinear Phenomena*, vol. 58, nos. 1–4, pp. 165–192, Sep. 1992.

[42] E. N. Sanchez and L. J. Ricalde, "Chaos control and synchronization, with input saturation, via recurrent neural networks," *Neural Netw.*, vol. 16, nos. 5–6, pp. 711–717, Jun./Jul. 2003.

[43] H. Richter, "An evolutionary algorithm for controlling chaos: The use of multi-objective fitness functions," in *Parallel Problem Solving from Nature—PPSN VII* (Lecture Notes in Computer Science), vol. 2439, J. J. Merelo, P. Adamidis, H.-G. Beyer, H.-P. Schwefel, and J.-L. Fernández-Villacañas, Eds. Berlin, Germany: Springer-Verlag, 2002, pp. 308–317.

[44] B. V. Chirikov, "Research concerning the theory of non-linear resonance and stochasticity," Inst. Nucl. Phys., Novosibirsk, Russia, Tech. Rep., 1969.

[45] H. Hamann, T. Schmickl, and K. Crailsheim, "Coupled inverted pendulums: A benchmark for evolving decentral controllers in modular robotics," in *Proc. 13th Annu. Conf. Genet. Evol. Comput. (GECCO)*, Dublin, Republic of Ireland, Jul. 2011, pp. 195–202.

[46] K. C. B. New, K. Watt, C. W. Misner, and J. M. Centrella, "Stable 3-level leapfrog integration in numerical relativity," *Phys. Rev. D*, vol. 58, no. 6, p. 064022, Aug. 1998.

[47] S. Mikkola, "Efficient symplectic integration of satellite orbits," *Celestial Mech. Dyn. Astron.*, vol. 74, no. 4, pp. 275–285, Aug. 1999.
[48] H. Kantz and T. Schreiber, *Nonlinear Time Series Analysis*, 2nd ed. Cambridge, U.K.: Cambridge Univ. Press, 2004.

Alexander P. Turner (M'13) received the B.Sc. degree in computer science from the University of Hull, Kingston upon Hull, U.K., in 2008, and the M.Sc. degree in natural computing and the Ph.D. degree in electronics from the University of York, York, U.K., in 2010 and 2014, respectively.

He is currently a Research Associate with the Intelligent Systems Research Group, Department of Electronics, University of York. His current research interests include evolutionary computing and biological modeling.

Dr. Turner is a member of the IEEE Computational Intelligence Society. He received the ERCIM Alain Bensoussan Fellowship to carry out research with the Bioinformatics and Gene Regulation Group, Faculty of Medicine, Norwegian University of Science and Technology, Trondheim, Norway, in 2014.

Leo S. D. Caves received the B.Sc. degree in chemistry from the University of Birmingham, Birmingham, U.K., in 1985, and the D.Phil. degree in computational biophysics from the University of York, York, U.K., in 1989.

He was a Research Fellow with the Department of Chemistry, Harvard University, Cambridge, U.K., from 1991 to 1994, with Prof. M. Karplus. From 1994 to 1997, he was a Post-Doctoral Researcher with the Department of Chemistry, University of York. In 1997, he became a Lecturer in Computational Chemistry with the Department of Chemistry, University of York, and became a Lecturer in Structural Bioinformatics with the Department of Biology in 2002. In 2008, he became a Senior Lecturer in Computational Biology. He is currently a Founder Member of the York Centre for Complex Systems Analysis and was the Nominated Spokesperson from 2005 to 2012. His current research interests include computational models of biological systems, such as molecular biophysics, systems biology, and bioinspired computation, and offering biological perspectives of complexity to systems in a wide variety of domains.

Susan Stepney received the M.A. degree in natural sciences (theoretical physics) and the Ph.D. degree in astrophysics from the University of Cambridge, Cambridge, U.K., in 1979 and 1983, respectively.

She was an SERC Post-Doctoral Research Fellow with the Institute of Astronomy, Cambridge, from 1983 to 1984, where she was involved in analytical and computational modeling of relativistically hot plasmas. From 1984 to 2002, she was involved in commercial research and development. From 1984 to 1989, she was a Research Scientist with GEC-Marconi, Farnborough, U.K., and a Consultant with Logica, Reading, U.K., from 1989 to 2002. Her industrial work was mostly in the area of formal methods. She was involved in the Z specification and proof of security and financially critical smart card products, Mondex and Multos, and was a member of the BSI/ISO Z Standardization team. Since 2002, she has been a Professor of Computer Science with the University of York, York, U.K., where she leads the Non-Standard Computation Research Group. Since 2012, she has been the Director of the York Centre for Complex Systems Analysis. Her current research interests include unconventional models of computation, complex systems, emergence, bioinspired computing, and computational simulation of biological systems.

Andy M. Tyrrell (SM'96) received the B.Sc. (Hons.) degree in electrical and electronic engineering and the Ph.D. degree in electrical and electronic engineering from Aston University, Birmingham, U.K., in 1982 and 1985, respectively.

He was promoted to Chair in Digital Electronics in 1998. He has been with the Department of Electronics, University of York, York, U.K., since 1990. He is currently the Head of the Intelligent Systems Research Group with the University of York, where he was the Head of the Department of Electronics from 2000 to 2007. In particular, over the last 18 years, his research group with the University of York has concentrated on bioinspired systems. He has authored over 300 publications in his research areas and attracted funds in excess of £9 million. His current research interests include the design of biologically inspired architectures, artificial immune systems, evolvable hardware, field-programmable gate array system design, parallel systems, fault tolerant design, and real-time systems.

Prof. Tyrrell is a fellow of the Institution of Engineering and Technology.

Michael A. Lones (M'01–SM'10) received the M.Eng. degree in computer systems and software engineering and the Ph.D. degree in electronics from the University of York, York, U.K., in 1999 and 2003, respectively.

He was a Research Fellow with the Department of Electronics, University of York, from 2005 to 2013. Since 2013, he has been an Assistant Professor with the School of Mathematical and Computer Sciences, Heriot-Watt University, Edinburgh, U.K. His current research interests include biologically motivated models of computation and their application to problems in computational biology, medical informatics, and complexity science and robotics. He has authored over 50 publications in his research areas.

Dr. Lones is a member of the IEEE Computational Intelligence Society and an Active Member of the IEEE Technical Committee on Bioinformatics and Bioengineering. He received the ERCIM Fellowship to carry out research with the Bioinformatics and Gene Regulation Group, Faculty of Medicine, Norwegian University of Science and Technology, Trondheim, Norway, in 2004.