Artificial Biochemical Networks: Evolving Dynamical Systems to Control Dynamical Systems

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

***Abstract*—Biological organisms exist within environments in which complex, non-linear dynamics are ubiquitous. They are coupled to these environments via their own complex, dynamical networks of enzyme-mediated reactions, known as biochemical networks. These networks, in turn, control the growth and behaviour of an organism within its environment. In this paper, we consider computational models whose structure and function are motivated by the organisation of biochemical networks. We refer to these as artificial biochemical networks, and show how they can be evolved to control trajectories within three behaviourally diverse complex dynamical systems: the Lorenz system, Chirikov's standard map, and legged robot locomotion. More generally, we consider the notion of evolving dynamical systems to control dynamical systems, and discuss the advantages and disadvantages of using higher order coupling and configurable dynamical modules (in the form of discrete maps) within artificial biochemical networks. We find both approaches to be advantageous in certain situations, though note that the relative trade-offs between different models of artificial biochemical network strongly depend on the type of dynamical systems being controlled.**

***Index Terms*—Genetic programming, dynamical systems, biochemical networks, chaos control, evolutionary robotics.**

I. Introduction

THE REAL WORLD is hard to control. It is complex and non-linear, many aspects are not well understood, and it can be difficult or impossible to predict the behaviour of those which are. Furthermore, the high-level dynamics, through which we observe real world systems, usually emerge through a myriad of low-level interactions between components which cannot be readily observed or measured.

Given the difficulty of real world control, it is no surprise that many people have turned to evolutionary algorithms as a means of generating novel control strategies. Situations where this is particularly appropriate include large parameter search spaces, no known conventional (e.g. mathematical) methods of control, the presence of non-linearity and, more generally, when the target system in poorly understood. A number of different evolutionary algorithms have been used to design controllers. Genetic algorithms (GAs) [1], for instance, are particularly suitable for where there is a need to optimise parameters for an existing controller architecture; for example, tuning the parameters of a PID controller [2]. Another approach is to use model induction algorithms such as genetic programming [3], learning classifier systems [4] and neuro-evolutionary algorithms [5] to induce the entire controller architecture. This is particularly appropriate for non-linear controllers, where manual design is hard, and for which there is an increasing demand in many application areas.

Biological organisms are adept at controlling and responding to complex non-linear dynamics, making them a useful source of information about complex control techniques. From a computational perspective, a biological system that has attracted much attention is the animal brain, leading to diverse research on neurocomputing-based approaches to control [6], [7]. However, control behaviours occur at multiple scales within biological organisms, and arguably the most prevalent of these is at the level of biochemical networks, the protein-mediated networks of biochemical reactions that implement and regulate the behaviour of biological cells. At some level, biochemical networks are responsible for almost all behaviour carried out by biological organisms. The essential role that biochemical networks play within biological organisms, and the complexity they engender, makes them computationally interesting. This computational interest, in turn, has led to a range of computational models [8]–[18], which we refer to collectively as artificial biochemical networks (ABNs).

ABNs are an example of a larger group of computational models, devices and architectures, which we refer to as computational dynamical systems (CDSs) [19]–[21]. Other examples of these are recurrent neural networks [22], [23], cellular automata [24] and reaction-diffusion computers [25]. Like ABNs, many of these are models of processes that occur in biological and other naturally-occurring systems. Rather than carrying out computation in the precise state-based manner of conventional computers, their behaviour can best be described using concepts from dynamical systems theory: such as trajectories, attractors, basins and bifurcations.

CDSs have been widely used for computation [20], [26]–[33], and approaches such as reservoir computing [34] show that any dynamical system with sufficiently rich dynamics, and some mechanism for introducing inputs and extracting outputs, can be used for computation (and in many cases can

The authors are members of the York Centre for Complex Systems Analysis (YCSSA), University of York, UK. Michael Lones, Luis Fuente, Alex Turner, Stephen Smith and Andy Tyrrell are also members of the Intelligent Systems Research Group, Department of Electronics; Susan Stepney is in the Non-Standard Computation Research Group, Department of Computer Science; and Leo Caves is in the Department of Biology.

This research is funded by the EPSRC grant "Artificial Biochemical Networks: Computational Models and Architectures," ref. EP/F060041/1.

be shown to be Turing complete, e.g. [35]–[37]). However, for use with evolutionary algorithms, expressiveness is not sufficient: we also need evolvability. From this perspective, ABNs are particularly appealing, since their biological analogues display many properties that contribute towards evolvability, such as redundancy, modularity, weak linkage and scale-free topologies [38]. It could also be argued that, as a product of evolution, biochemical networks were selected for representing and controlling complex behaviours within biological systems.

In this paper, we demonstrate how ABN-based controllers can be evolved to control a variety of dynamical systems. The paper is organised as follows: Section II introduces the theory of dynamical systems, and discusses how they may be used to carry out computation and how they may be designed using evolutionary algorithms; Section III introduces artificial biochemical networks, outlining prior work in this area; Section IV introduces the ABN models used in this research; Section V describes the control tasks to which we have applied these models; Section VI presents results, which are discussed in Section VII. Section VIII concludes.

II. Dynamical Systems

A. Terminology

A dynamical system [21] is any system whose subsequent state is determined by a function, or *evolution rule*[1], of the system's current state. Starting at a particular point within the system's state space, known as its *initial condition*, the path that the system follows through its state space, its *trajectory*, is determined by iterating the evolution rule over a period of time. Many dynamical systems have no analytical solution, meaning that this iterative process is often the only way of determining the system's state at a particular time.

Following initial periods of wandering, termed *transients*, trajectories may converge to limited parts of the state space known as *attractors*. An attractor has a *basin of attraction*, and any trajectory in this region will be drawn towards the attractor. In a point attractor, all trajectories are drawn to a single point in state space. In a cyclic attractor, they are drawn to an endlessly repeating series of states. Dynamical systems in which all trajectories eventually converge to one or more attractors are termed *dissipative*. Those which do not converge in this fashion are *conservative*.

Dynamical systems can have ordered and chaotic regions of state space. In an ordered region, a trajectory becomes predictable once it enters a point or cyclic attractor. In a chaotic region, by comparison, a trajectory cannot be predicted, despite the deterministic evolution rule. This is due to the phenomenon of exponential sensitivity to initial conditions (popularly known as the *butterfly effect*), whereby neighbouring trajectories move apart at an exponential rate, causing even the smallest prediction error to be amplified exponentially. Chaotic dynamical systems typically display complex, fractal attractors known as *strange attractors*.

[1]Note that evolution here refers to change over time, not to a selection-driven process as in biological evolution and evolutionary computation.

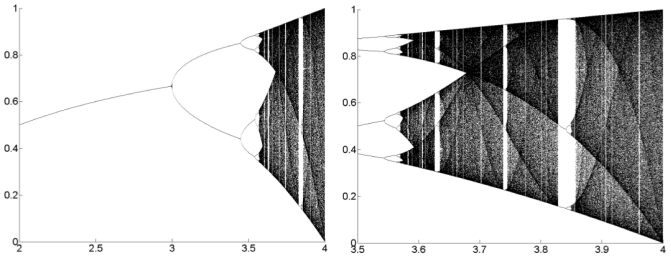


Fig. 1. Bifurcation diagram for the logistic map, showing period doubling behaviour for $r \lesssim 3.57$, and close-up of the fractal structure thereafter [21].

Dynamical systems are said to be *continuous*, *discrete* or *hybrid*, depending upon whether their evolution rule is continuous-time, discrete-time or a mixture of both.

B. Computational Dynamical Systems

There are a number of reasons why dynamical systems are computationally appealing. Perhaps most significantly, they are capable of generating complex behaviours in an efficient manner, using only a compact system definition. The archetypal example of this is the logistic map, defined:

$$x_{n+1} = r x_n (1 - x_n), \qquad x \in [0, 1]; r \in [0, 4] \tag{1}$$

This simple iterative parameterised non-linear map exhibits a wide range of behaviours depending upon the value of parameter r. When $r < 3$, all trajectories converge to a single value of x, irrespective of initial conditions. When $r = 3$, this point attractor bifurcates, forming a cyclic attractor of period 2. The attractor period then doubles repeatedly until, at $r \approx 3.57$, a chaotic attractor appears. Thereafter, bifurcation space takes on a fractal form, with the system displaying various periodic and aperiodic behaviours as r increases (see Fig. 1).

The dynamics of non-linear maps, such as the logistic map, makes them computationally interesting in their own right. Their complex, parameterisable, behaviour also makes them interesting as elements in larger dynamical systems. An example of this is the coupled map lattice (CML) [39], in which an array of non-linear maps is coupled together to form something akin to a continuous-valued cellular automaton. These map lattices have been used to model the complex dynamics that occur in a diverse range of natural systems, including neural and genetic networks [40]. Furthermore, through suitable encoding of inputs and outputs, they can also be used for overtly computational tasks, such as density classification [26].

Various kinds of dynamical system have been used for computation. Table I gives examples of these, organised according to whether they have discrete or continuous evolution rules and state spaces. In addition to relatively well known *in silico* algorithms, such as cellular automata (CAs) and recurrent neural networks (RNNs), these include *in vitro* approaches such as reaction-diffusion computers, and *in vivo* methods, especially if biological organisms are considered as dynamical systems [41]. An example of the latter is the use of the slime mould *physarum polycephalum* for robotic control [31].

TABLE I
EXAMPLES OF COMPUTATIONAL DYNAMICAL SYSTEMS

	Discrete space	Continuous space
Discrete time	Cellular automata P Systems Boolean networks	Coupled map lattices Recurrent neural networks
Continuous time	Gillespie algorithm	Reaction-diffusion computers Continuous-time RNNs *Physarum polycephalum*

C. Evolving Computational Dynamical Systems

The use of compact representations has been posited as a means of addressing the scalability issue in evolutionary algorithms [42], in which solution size (and hence computational effort) tends to grow exponentially with problem size [43]. Hence, the compact nature of dynamical systems makes them particularly relevant from an evolutionary computation perspective, and the use of computational dynamical systems may offer the potential to evolve complex behaviours using relatively modest computing resources.

There are numerous existing examples of computational dynamical systems being evolved using evolutionary algorithms. The most commonplace of these is the design and optimisation of RNNs. Early work by Angeline et al. [44] demonstrated how an evolutionary algorithm can be used to induce both the weights and topology of an RNN, removing the need to design application-specific architectures in advance of conventional weight training. More recent work includes the optimisation of neural modules using variants of GP [45], and the development of specialised neuroevolution algorithms such as NEAT [5], [46], [47]. Work has also been done on evolving cellular automata [48], much of it in the context of exploring the benefits of computation at the 'edge of chaos'. Other examples of evolved computational dynamical systems include reaction-diffusion based controllers [29] and echo state networks [49].

Given the difficulty of designing dynamical systems by hand, the use of evolutionary algorithms provides a convenient technique for programming such systems. However, not all dynamical systems are equivalent from the perspective of evolutionary computation. If we are to use evolved dynamical systems to represent computation, it is sensible that we should look for systems that are *evolvable*, i.e. those which possess robustness in the face of genetic perturbation, whilst still encouraging useful forms of phenotypic exploration. Quite a lot is known about the kinds of systems that possess evolvability [50]–[54], much of which can be summarised by Michael Conrad's triplet: redundancy, compartmentalisation, and weak linkage [50]. Redundancy provides both a buffer against genetic change and a source of evolutionary capacitance to drive future change. Compartmentalisation limits the scope of genetic perturbations, whilst promoting modular reuse. Weak linkage dampens perturbations, whilst easing the recruitment of regulatory signals in novel contexts.

Biological systems are rich in mechanisms which confer evolvability. Whilst these mechanisms operate at higher levels of biological organisation (e.g. populations, ecologies), it is particularly apparent at the organismal level, which is directly exposed to the genetic variation associated with evolution. This is the argument that drives our interest in artificial biochemical networks: biochemical networks are the dynamical systems responsible for encoding low-level biological behaviour—and, in a sense, are the selected means of representing complex behaviour in biological evolution.

III. ARTIFICIAL BIOCHEMICAL NETWORKS

A. Biochemical Networks

In biological systems, biochemical networks emerge from protein-mediated molecular interactions taking place within cells. These complex dynamical networks underlie both the structure and function of biological organisms. We consider three kinds of biochemical network: metabolic, genetic and signalling.

A *metabolic network* results from self-organising interactions between the enzyme-mediated reactions that take place within a cell. It emerges when the products of certain reactions become the substrates of others, forming chains of reactions known as metabolic pathways. Product-substrate sharing between pathways results in the metabolic network. A *genetic network* emerges from the regulatory interactions between genes. It captures how they regulate one another's protein expression levels over time through the production of transcription factors. A *signalling network* comprises the protein-mediated reaction pathways through which chemical messages are delivered to the cell's internal environment. The metabolic, genetic and signalling networks have been described, respectively, as the self-organising, self-modifying, and self-reshaping components of a cell's biochemical network [55].

These three networks do not work in isolation, but are coupled. By regulating protein production, the genetic network modifies the behaviour of both the metabolic and signalling networks. By delivering chemical signals to different subcellular locations, the signalling network modulates the behaviour of both genetic and metabolic networks. In single-celled organisms, these interactions allow the cell's metabolism to be reconfigured for different nutrient environments; in multicellular organisms, they are the basis of cellular differentiation and morphogenesis. The interactions between the three kinds of biochemical network are depicted in Fig. 2.

B. Artificial Metabolic Networks

We use the term artificial metabolic networks (AMNs) to refer to computational architectures which are modelled on the self-organising behaviour of cellular chemistries. In much the same way that complex metabolic processing emerges from interactions between biochemicals, these architectures aim to achieve complex computational behaviour through the interactions of simple computational elements.

P systems (also known as membrane systems) [8] are perhaps the most widely studied of these approaches. In a P system, chemicals are modelled as symbols, and chemical reactions are modelled as symbolic rewriting rules. Symbols

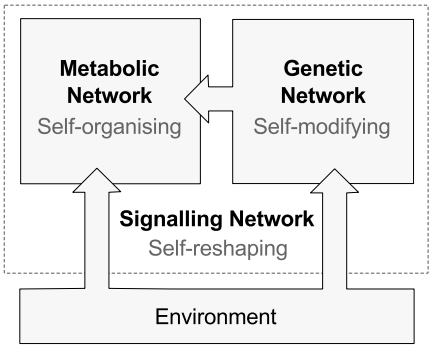


Fig. 2. Interactions that take place between the three classes of biochemical network present within a biological cell.

and rules are organised into compartments. During execution, rules are iteratively applied to the symbolic state of each compartment, and compartments may break open, causing the composition of symbolic states. Whilst P systems are most known for their use in biological modelling, they can also be used computationally; in which case inputs are encoded in the initial symbolic state, and outputs are read from the final symbolic state. Such P systems are usually designed by hand. See [56] for examples.

P systems can be considered a special case of a wider group of algorithms known as artificial chemistries [9], [10] and algorithmic chemistries [11], [12]. These comprise three elements: a set of *chemicals*, a set of *reactions*, and an algorithm that determines how chemicals move about and when reactions can take place. Chemicals may be symbols to which some computational meaning can be associated [27], they may directly encode data structures, they may be overtly computational in nature (e.g. lambda-expressions [11], Prolog terms [12]), or they may even be other ABNs [57]. Likewise, reactions vary from simple symbolic transformations to functional composition and complex structural modifications. By encoding inputs and outputs in the concentrations, internal structures or positioning of chemicals, these artificial chemistries can be used for computation. Artificial chemistries have been evolved to carry out a number of computational tasks, including robot navigation [27], classification [58], and Boolean decision making [59].

C. Artificial Genetic Networks

An artificial genetic network (AGN) is a computational architecture modelled on the regulatory interactions between genes. The simplest, and best known, example of an AGN is the Boolean network (often referred to as a *random* Boolean network, or RBN). An RBN is a closed system comprising a set of interconnected genes, each of which has a Boolean state and a Boolean regulatory function—and whose state is calculated by applying the regulatory function to the states of those genes to which it is connected. In Kauffman's [13] original model, the states are updated synchronously.

RBNs can be seen as a generalisation of binary cellular automata in which update rules can reference non-neighbouring cells and functions are heterogenous. In practice, computation can be achieved in the same way as cellular automata: by providing input via the initial activity state of the genes, running a network for a certain number of time steps, and then reading the output from the final activity states of the genes. The computational properties of RBNs have been discussed at length in [60], and also in [61] from a more implementation-focussed perspective. In [30], the authors showed how RBNs can be evolved to carry out multiplexing tasks.

The RBN model captures the process of gene regulation as a deterministic, synchronous, discrete on/off event: all of which are abstractions of the true behaviour which involves elements of stochasticity, asynchronous timing and continuous behaviour. Despite this, RBNs have been used to successfully model the dynamics of real genetic networks [62], suggesting that greater bio-realism is not a prerequisite for complex behaviour. However, this does not imply that AGN models do not differ in their ease of programmability, evolvability, and (from an applied perspective) how readily they can be coupled to an external system. From this perspective, the Boolean nature of RBNs causes some problems. Most significantly, an RBN comprising N genes has a state space of 2^N possible states. This means that small networks have a limited ability to express complex attractor structures. Furthermore, inputs and outputs must be binary encoded. Assuming that I/O is encoded in initial and final expression states, this means there must be at least as many genes as there are bits in the inputs or outputs. Synchronicity and determinism also affect the ease with which certain dynamics can be expressed, an issue which is explored in [63].

Much of the work done on RBNs has been motivated by the desire to model and understand biological systems. However, other researchers have approached AGNs from the perspective of evolutionary computation [14]–[16], [28], [32], [33], [64], seeing them as a means for representing computational behaviour. There is considerable variation amongst the resulting models. One prominent distinction is between template matching [14], [15], [28] and connection-orientated [16] approaches. The former captures the indirect means by which biological genes regulate one another (i.e. transcription factors binding probabilistically to upstream regulatory sites), whereas the latter represents interactions explicitly. Other distinctions can be made between use of continuous time [15], [32] and discrete time [14], [16], [28], [33], [64] updates; continuous [15], [16], [28], [32], [64] and discrete-valued [14], [33] expression levels; and the use of spatial diffusion of gene products [16], [28]. These AGNs have been applied to a variety of tasks, including robotic control [28], pole balancing [32] and image compression [33].

D. Artificial Signalling Networks

Signalling networks are yet to receive the same level of computational interest as genetic and metabolic networks. Nevertheless, they carry out a number of computationally interesting behaviours [65], [66]. For instance, in [65], the authors discuss the manner in which signalling pathways integrate and pre-process diverse incoming signals, likening their behaviour to that of a fuzzy classifier system. In [66], the author draws parallels between the adaptive behaviours

of various signalling pathways and those of engineered controllers. Early work in computational modelling of signalling pathways led to a feed-forward architecture modelled upon the behaviour of signalling proteins [17]. More recent work in this area has focussed on signalling-based learning classifier systems [18].

IV. ABN Models

The goal of this paper is to provide insight into the computational potential of ABNs, particularly when used within the context of evolutionary algorithms. It should be noted that our aim is *not* to compare the many different ways in which genetic and metabolic networks can be modelled computationally. Rather, we aim to demonstrate how representative ABN models can be practically applied to a range of computational tasks. Also, since our focus is on practicality, our choice of models reflects a desire for usability, simplicity and efficiency, in addition to the more general goals of expressiveness and evolvability. For this reason, all our models are continuous-valued and discrete-time. In addition to allowing relatively small networks to have complex dynamics, the former means that the networks can be readily coupled to their environment. The latter means that the networks are relatively efficient, since there is no need to solve differential equations.

We are also interested in whether there are any benefits to coupling together different types of ABN. As we mentioned in Section III-A, biological networks do not work independently. Rather, the metabolic network is repeatedly reconfigured by the genetic network in order to meet the varying demands placed upon a cell by its environment. We capture this idea by coupling an artificial genetic network to an artificial metabolic network, meaning that changes in the dynamics of the AGN will lead to changes in the dynamics of the AMN.

In this section, we first introduce the stand-alone AGN and AMN models used in this work, and then describe how these are combined to form a coupled artificial biochemical network (CABN).

A. Artificial genetic network (AGN)

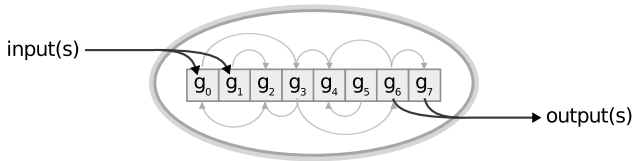


Fig. 3. Artificial genetic network, where I_G={g_0, g_1}; O_G={g_6, g_7}.

The artificial genetic network consists of an indexed set of genes, each of which has an expression level, regulatory inputs, and a regulatory function which maps the expression levels of its regulatory inputs to its own expression level. As with Boolean networks, interactions between genes are defined explicitly through indices, rather than indirectly through analogues of transcription factors. Formally: AGN $= \langle G, L_G, I_G, O_G, t_G \rangle$, where:

G is the set of genes $\{g_0, ..., g_{n_G} : g_i = \langle \lambda_i, R_i, f_i \rangle\}$, where:
$\lambda_i : \mathbb{R}$ is the expression level of a gene.
$R_i \subseteq G$ is the set of regulatory inputs used by a gene.
$f_i : R_i \rightarrow \lambda_i$ is a gene's regulatory function.

L_G is an indexed set of initial expression levels, where $|L_G| = |G|$.
$I_G \subset G$ is the set of genes used as external inputs.
$O_G \subset G$ is the set of genes used as external outputs.
t_G is the number of time steps per execution.

The first time the AGN is executed, its expression levels are initialised from L_G. External inputs can be delivered to the network either by explicitly setting the expression levels of certain genes (specified in I_G), or by introducing new regulatory inputs with fixed values. After applying the regulatory functions and propagating values between genes a specified number of times, t_G, outputs are captured from the final expression levels of genes specified in O_G.

This AGN model is a continuous-valued generalisation of a Boolean network. However, it is also closely related to other computational dynamical systems, including continuous-valued cellular automata (when R is constrained to direct neighbours), coupled map lattices (when f is a discrete map), and recurrent neural networks (when f is a sigmoid). Because of this, we can use a sigmoidal AGN as a proxy for an RNN, giving an indication of how other ABN models compare to this more established form of computational dynamical system. Most RNN architectures are tailored towards particular application areas, e.g. Elman networks [23], whose topology promotes short-term memory acquisition during sequential processing. In order to promote the generality of any comparisons we make (and following other evolutionary approaches to RNN induction [44]), we do not restrict the topology of evolved sigmoidal AGNs, allowing any appropriate pattern of connectivity to be evolved.

B. Artificial metabolic network (AMN)

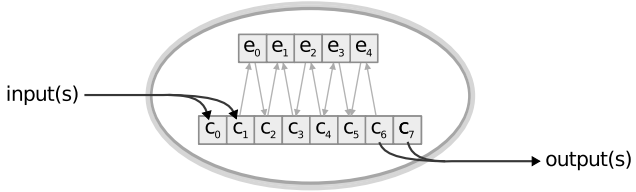


Fig. 4. Artificial metabolic network, where I_C={c_0, c_1}; O_C={c_6, c_7}.

The artificial metabolic network is a minimal implementation of an artificial chemistry, capturing the key idea that a set of computational elements manipulate a set of chemicals over a period of time, but abstracting away the elements found in more complicated chemistries such as non-determinism, internal chemical structure, and spatial distribution. It comprises an indexed set of enzyme-analogous elements which transform the concentrations of an indexed set of real-valued chemicals. Each enzyme has a set of substrates, a set of products, and a mapping which calculates the concentrations of its products based upon the concentrations of its substrates. Formally: AMN $= \langle C, E, L_C, I_C, O_C, t_M \rangle$, where:

C is the set of chemical concentrations $\{c_0, ..., c_{n_C} : \mathbb{R}\}$.
E is the set of enzymes $\{e_0, ..., e_{n_E} : e_i = \langle S_i, P_i, m_i \rangle\}$, where:

$S_i \subseteq C$ is the set of chemicals used by the enzyme (*substrates*).
$P_i \subseteq C$ is the set of chemicals produced by the enzyme (*products*).
$m_i : \mathbb{R}^n \rightarrow \mathbb{R}^n$ is the enzyme's substrate-product mapping.

L_C is an indexed set of initial chemical concentrations, where $|L_C| = |C|$.
$I_C \subset C$ is the set of chemicals used as external inputs.
$O_C \subset C$ is the set of chemicals used as external outputs.
t_M is the number of time steps per execution.

The first time the AMN is executed, its chemical concentrations are initialised from L_C. External inputs are delivered to the network by explicitly setting the concentrations of chemicals whose indexes are specified in I_C. At each time step, each enzyme e_i applies its reaction m_i to the current concentrations of its substrates S_i in order to determine the new concentrations of its products P_i. Where the same chemical is produced by multiple enzymes, i.e. when $\exists j, k : j \neq k \wedge c_i \in P_j \cap P_k$, the new concentration is the mean output value of all contributing enzymes:

$$c_i = \sum_{e_j \in E_{c_i}} \frac{c_{i,e_j}}{|E_{c_i}|} \tag{2}$$

where E_{c_i} are enzymes for which $c_i \in P_i$ and c_{i,e_j} is the output value of e_j for c_i. After iterating the network t_M times, outputs are captured from the final concentrations of chemicals specified in O_C.

We also consider the effect of applying a *mass conservation law*, such that the sum of chemical concentrations remains constant over time. This more closely reflects biological systems, where mass balance results in indirect regulatory interactions between chemical reactions. It is implemented by uniformly scaling concentrations so that:

$$\sum_{c_i \in C} c_i = 0.5|C| \tag{3}$$

However, chemicals which have reached saturation ($c = 1$) and those which are not present in the chemistry ($c = 0$) remain unchanged, preserving these special states.

C. Coupled Artificial Biochemical Network (CABN)

In the coupled artificial biochemical network (CABN) model, we capture the idea of a genetic network controlling the expression of a metabolic network.

Formally: $\text{CABN} = \langle \text{AGN}, \text{AMN}, \chi \rangle$, where $\chi : G_C \rightarrow E$ is an injective coupling function in which $G_C \subseteq G$ is the set of enzyme coding genes. Each enzyme is coupled to a single gene, and some genes may not be enzyme coding (yet are still involved in regulating other genes). Coupling is carried out by giving each enzyme an expression level, ξ_i, and setting this to the expression level of the gene to which it is coupled, i.e. $\forall (g_i, e_j) \in \chi : \xi_j := \lambda_i$. This expression level then determines the relative influence of each enzyme when calculating the new concentration of a chemical. We consider three methods for updating concentrations:

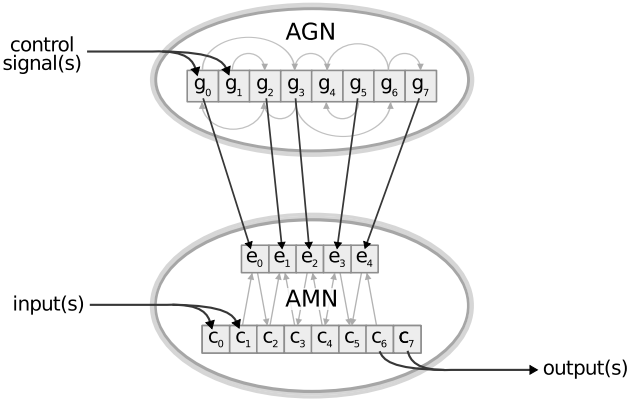


Fig. 5. CABN, where χ=$\{\{g_0, e_0\}, \{g_2, e_1\}, \{g_3, e_2\}, \{g_5, e_3\}, \{g_7, e_4\}\}$. Inputs may be delivered to either $G \in$ AGN or $C \in$ AMN.

1) Proportional updates: The new concentration is the mean of each enzyme's output value weighted by its relative expression level:

$$c_i = \sum_{e_j \in E_{c_i}} \frac{\xi_i c_{i,e_j}}{\sum_{e_j \in E_{c_i}} \xi_i} \tag{4}$$

2) Scaled updates: The new concentration is the sum of the outputs of each contributing enzyme, each scaled by their enzyme's expression level:

$$c_i = \min \left\{ 1, \sum_{e_j \in E_{c_i}} \xi_i c_{i,e_j} \right\} \tag{5}$$

3) Thresholded updates: An enzyme is only active if its expression level is above a threshold value, τ. New concentrations are calculated using the standard AMN update rule (Equ. 2), ignoring any inactive enzymes. This update rule models gene expression as a Boolean process.

In biological cells, the dynamics of the genetic network occur over a longer time-scale than those of the metabolic network. To capture this relationship, the timing of the CABN model is biased so that, on average, AMNs execute at a faster rate than AGNs. This is done by reinterpreting the AGN's t_G variable to be the interval between updates, in terms of the AMN's time frame. For example, if t_M is 10 and t_G is 5, the AGN will be iterated twice in the time it takes the AMN to be iterated 10 times.

Inputs can be delivered to either the AGN or the AMN components of the CABN. In the control tasks, we make a distinction between control inputs and other state inputs. Control inputs are those which direct the desired behaviour of the controller. These are delivered to the AGN, with the expectation that changes in the control input will lead to changes in gene expression, causing the AMN to switch between different input-output mapping behaviours.

The idea of coupling an artificial genetic network to a metabolic model has also recently been explored in [67] and

TABLE II
MATHEMATICAL FUNCTIONS USED WITHIN ABNS.

Sigmoid (logistic function):
$f(x) = \frac{1}{1+e^{-sx-b}}$, where $s \in [0, 20], b \in [-1, 1]$

Logistic map:
$x_{n+1} = rx_n(1 - x_n)$, where $r \in [0, 4]$

Baker's map:
$$(x_{n+1}, y_{n+1}) = \begin{cases} (2x_n, y_n/2) & 0 \leq x_n \leq \frac{1}{2} \\ (2 - 2x_n, 1 - y_n/2) & \frac{1}{2} \leq x_n < 1 \end{cases}$$

Arnold's cat map:
$(x_{n+1}, y_{n+1}) = ([2x_n + y_n] \bmod 1, [x_n + y_n] \bmod 1)$

Chirikov's standard map:
$$x_{n+1} = (x_n + y_{n+1}) \bmod 1$$
$$y_{n+1} = (y_n - \frac{k}{2\pi} \sin(2\pi x_n)) \bmod 1, \quad k \in [0, 10]$$

[68]. In the former, the author investigates the attractor structure of a computational cell model in which two Boolean networks are coupled together, showing an advantage to coupling networks with heterogenous topologies. The authors of [68] also developed a cellular model, in which an ABN controls a simple model of a cell's energy metabolism, showing how it can be used to control foraging behaviours in an artificial life system.

D. Regulatory functions and enzyme mappings

There is a complex relationship between the overall dynamics of a network and the dynamics of the processes taking place at individual nodes within the network. For example, results from work on coupled map lattices suggest that there is no simple relationship between the dynamics of individual maps and the overall dynamics of the coupled map, a phenomenon known as non-trivial collective behaviour [69]. With this in mind, we have used a variety of different mathematical functions to implement the regulatory functions and enzyme mappings which take place within the nodes of ABNs. These are listed in Table II.

Sigmoids model the switching behaviour of non-linear biological systems, making them a good choice for approximating the behaviours of genetic and metabolic pathways. In addition to being a realistic model of processes that take place within the nodes of biochemical neworks, sigmoidal functions have also been used as effective low-level elements in other computational dynamical systems such as recurrent neural networks. The sigmoid function can be implemented in various ways. In our case, we use the logistic function, where (see Table II) s determines the slope and b the slope offset (or *bias*). For multiple inputs, $x = \sum_{j=0}^{n} i_j w_j$, where $i_0 ... i_n$ are inputs and $w_0 ... w_n \in [-1, 1]$ are corresponding input weights, with negative values indicating repression.

Whilst biological regulatory functions and enzyme reactions can often be approximated by sigmoidal and Boolean functions, this is not always the case. These biological functions are the result of complex physical and chemical interactions, often involving multiple diverse biomolecules [70], [71]. Consequently, they tend to be highly non-linear in nature. We have not attempted to model these functions explicitly—although this could be an interesting direction for future work. We have, however, attempted to capture the idea of complex, non-linear functions by using discrete non-linear maps to implement regulatory functions and enzyme mappings. As discussed in Section II-B, discrete maps display a range of complex dynamics and have generated considerable computational interest in their own right.

In this work, we use four discrete maps that capture the natural dynamics present in a range of biological and physical systems. *The logistic map*, which we have already discussed, is a model of biological population growth. Depending on the value of parameter r, the system is attracted to either a fixed-point, cyclic or chaotic orbit [72]. *The baker's map* [73] is an archetypal model of deterministic chaos, capturing the irregular, unpredictable fractal structured behaviour that results from a process of repeated stretching and contraction—as seen when kneading bread, hence the map's name. *Arnold's cat map* [74] is another model of deterministic chaos which results from a geometric transformation of the unit square, and leads to interesting periodic behaviour. *Chirikov's standard map* [75] captures the behaviour of dynamical systems with co-existing ordered and chaotic regimes. Its properties are discussed in detail in Section V-A2. The parameterised maps (the logistic map and Chirikov's map) can be used either with an evolved fixed parameter value or with an extra input, whose current value is used to set the parameter. The latter is referred to as a tunable map, since its dynamics can be modified by the ABN during execution.

V. CONTROLLING DYNAMICAL SYSTEMS

We have applied ABN-based controllers to three different tasks, chosen to be representative of the kinds of complex dynamics that occur in natural systems. The first two involve state space targeting in numerical dynamical systems that display both ordered and chaotic dynamics, namely the Lorenz system and Chirikov's standard map. The final task involves controlling the locomotion of a simulated legged robot. For each of the tasks, the aim is to evolve a closed-loop controller which uses an ABN to map current system state into appropriate control signals—which, in turn, determine future system state.

A. State Space Targetting in Mixed Chaotic/Ordered Systems

Chaotic dynamics occur across a wide spectrum of man-made and naturally occurring systems. Control of chaotic dynamics is therefore important in many domains: including spacecraft steering [76], control of chemical plants, prevention of heart arrhythmia [77], and even control of the weather [78]. For this reason, a number of chaos control methods have been developed [79]. Of these, two approaches are

prominent: those which analyse the system's return map, and those which use time-delayed feedback. The former derive from the seminal method of Ott, Grebogi and Yorke (OGY) [80], which waits for the system to approach an unstable fixed point or orbit and then generates small perturbations to push the trajectory in the stable direction of the fixed point or orbit. The magnitude of the perturbation is determined through analysis of the local Eigenstructure around the control point, and more recent variants of this method consider multiple control points to reduce the time required to reach stabilisation [81]. The delayed feedback approach is typified by Pyragas' method [82], which involves applying a control signal whose magnitude is proportional to the difference between the system's current state and its state at a previous time step. Other than determination of an appropriate time delay, this method requires no explicit knowledge of the local dynamics. More recent adaptations make use of multiple time delays to handle higher degrees of instability [82].

These chaos control techniques are concerned with maintaining a system at a fixed operating point. A more general approach, sometimes termed *chaos targeting*, or even more generally *state space targeting*, involves guiding a trajectory between points or regions within a dynamical system's state space. This can be done by mapping the system's state space and using analytical techniques to determine patterns of perturbations that will minimise transport times from one point to another. Such techniques have been successfully applied in the domain of spacecraft steering [76]. However, they require a detailed understanding of the underlying state space.

In this work, we are interested in whether evolved ABNs can be used to carry out state space targeting without requiring explicit knowledge of the underlying system's state space. Many real world systems display both chaotic and ordered behaviour. In reflection of this, we consider two numerical dynamical systems that exhibit both chaotic and ordered dynamics: the Lorenz system and Chirikov's standard map. The Lorenz system is a continuous dissipative dynamical system, and is representative of the complex flows that occur in many biological and physical systems, such as blood flow, heart rhythms, and the atmosphere. Its behaviour is either ordered or chaotic depending upon the settings of its governing parameters. Chirikov's standard map is a discrete conservative system, representative of many physical systems that display complex and varied behaviour, such as *n*-body gravitational systems and other Hamiltonian systems. Its state space contains co-existing ordered and chaotic regions, with the balance between the two determined by a governing parameter. In both the Lorenz system and the standard map, control is carried out by modulating a governing parameter rather than by directly manipulating trajectories, allowing evolved controllers to explore different dynamical regimes.

There have been previous applications of evolutionary algorithms [83]–[85] and neural networks [86], [87] within chaos control. In general, these were concerned with maintaining a system at a fixed operating point, aiming to improve upon the accuracy of local control techniques such as OGY, and often with explicit knowledge of dynamical information such as local Lyapunov exponents. Our approach is quite different:

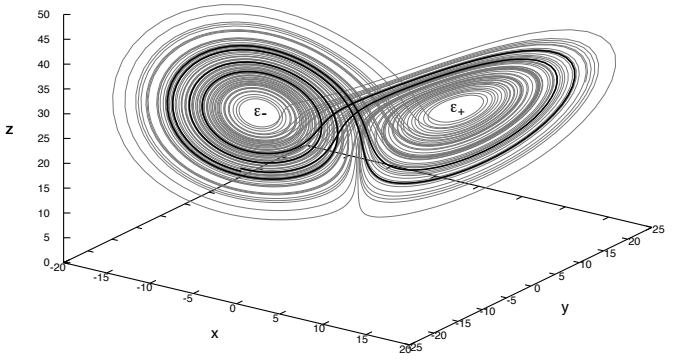


Fig. 6. A trajectory within the Lorenz attractor ($\sigma = 10$, $\rho = 28$, $\beta = \frac{8}{3}$), showing the location of the unstable equilibrium points ϵ_- and ϵ_+. The heavy line shows one of the unstable periodic orbits followed by the trajectory.

the controller must guide a trajectory across a state space, and it is given no information about the structure or dynamical characteristics of the state space, other than the current location of the trajectory being guided and the distance to the target region.

1) Lorenz System: The Lorenz system [88] is a continuous-time dynamical system which models oscillatory behaviours caused by atmospheric convection. It is defined by the following set of ordinary differential equations:

$$\dot{x} = \sigma\,(y - x) \qquad \dot{y} = x(\rho - z) - y \qquad \dot{z} = xy - \beta z \quad (6)$$

For $\rho \gtrsim 24.74$, the Lorenz system displays chaotic behaviour, with all initial points attracted to a single two-lobed strange attractor (see Fig. 6) which orbits two unstable equilibrium points, which we term $\epsilon_+$and ϵ_-, located at:

$$\epsilon_+ = (\sqrt{\beta(\rho-1)}, \sqrt{\beta(\rho-1)}, \rho - 1) \quad (7)$$

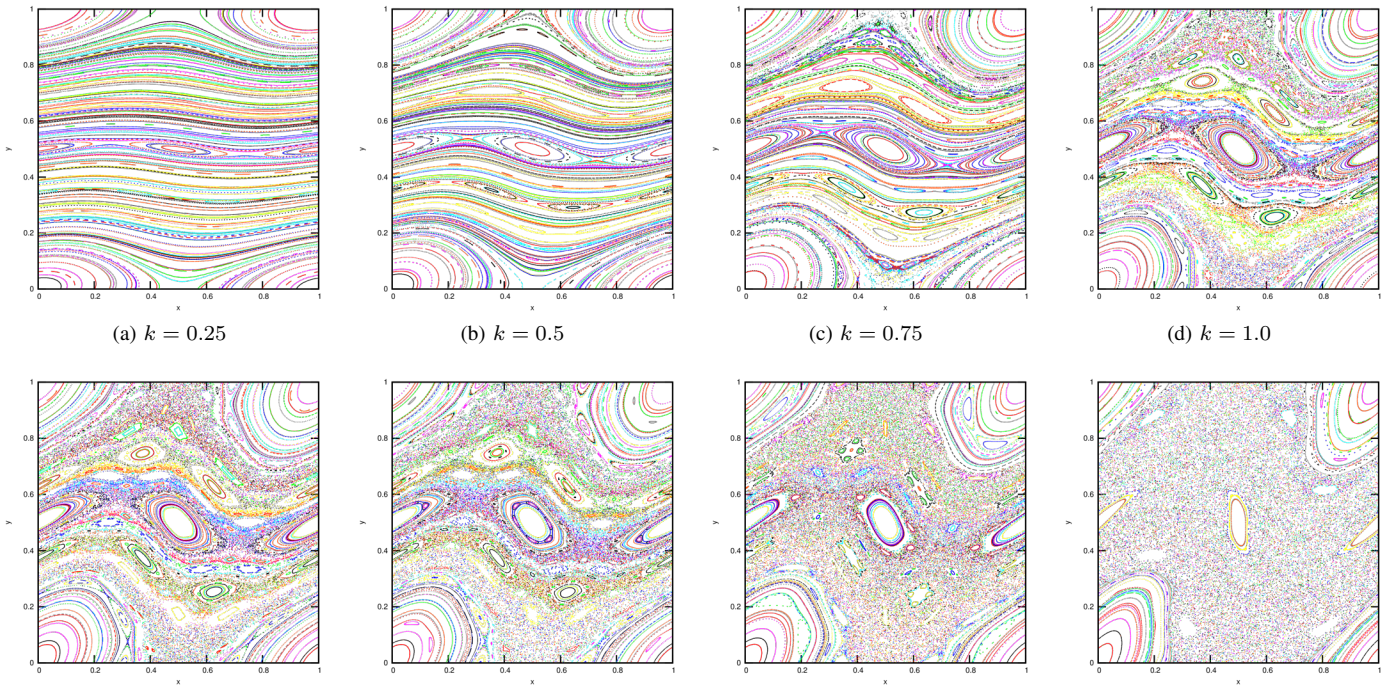
$$\epsilon_- = (-\sqrt{\beta(\rho-1)}, -\sqrt{\beta(\rho-1)}, \rho - 1) \quad (8)$$

The attractor consists of an infinite number of *unstable periodic orbits*. These are periodic in the sense that they orbit one or both of the fixed points a certain number of times before returning to roughly the same location. The orbits are unstable in the sense that trajectories will follow them for only a limited period of time before moving to another orbit. The dynamics of the system lead to trajectories which appear to flip unpredictably between the two lobes of the attractor.

The goal is to find an ABN-based controller which can (i) guide a trajectory between the two equilibrium points and, (ii) once it has reached a target equilibrium point, stabilise it at the fixed point for a defined period of time. This is tested by requiring it to move from ϵ_- to ϵ_+ and remain there until $t = 50$, then return to ϵ_-, remaining there until $t = 100$. In order to do this, the ABN is allowed to modulate the Rayleigh parameter[2], ρ, within the range $[0, 100]$.

For inputs, the ABN is given the current location, (x, y, z) (values scaled from $[-50, 50]$ to $[0, 1]$), and the distance to the target, d, with the latter defined as the control signal when a

[2]Whilst ρ is not normally a physically accessible parameter of the Lorenz system, here we use it to explore the more general concept of control through modulation of a system's governing parameters.



(a) $k = 0.25$ (b) $k = 0.5$ (c) $k = 0.75$ (d) $k = 1.0$

(e) $k = 1.05$ (f) $k = 1.1$ (g) $k = 1.25$ (h) $k = 1.75$

Fig. 7. Sampled orbits of the standard map for various values of k, showing the transition from ordered to chaotic behaviour as k increases. Each plot shows 200 trajectories of length 500, with the same set of initial conditions used for each plot.

CABN is used. To make the problem more challenging, this distance is provided only when the Euclidean distance to the target $E < 2.0$. Above this, it is set to the maximum input value, i.e. $d = \min\{1, \frac{E}{2}\}$.

The ABN generates a single output, the new value of ρ. The Lorenz equations are numerically integrated using the fourth-order Runge-Kutta method with a step size of $\Delta_t = 0.01$. The ABN is executed every 10 steps (i.e. $\Delta_t = 0.1$) to calculate a new value of ρ. The objective function f_L is defined:

$$f_L = \frac{\sum_s 1 - d}{s} \tag{9}$$

where s is the number of time steps, i.e. a measure of the mean distance from the current target for every time step. A population size of 500 and a generation limit of 50 are used.

2) Chirikov's Standard Map: Chirikov's standard map [75] describes a conservative discrete-time dynamical system which iteratively maps points within the unit square. It models the dynamical behaviour of a *kicked rotator*: a rotating bar which is periodically kicked with a frequency corresponding to k. The word 'standard' in the map's name follows from the observation that it locally captures the behaviours of many systems which have co-existing chaotic and ordered dynamics. Consequently, it provides a reductionist model of dynamics that occur in a range of physical systems, such as particle accelerators, cometary systems, and diodes.

The aim is to guide a trajectory from a region at the bottom of the map to a region at the top of the map. Using the normal definition of the map (see Table II), trajectories are able to move directly from y=0 to y=1, making this a trivial task. To prevent this behaviour, we do not take the modulus of the y co-ordinate, forcing trajectories to navigate across the unit square in order to reach the target region (and, as a consequence, the unit square as drawn recurs periodically along the y-axis):

$$\begin{aligned} x_{n+1} &= (x_n + y_{n+1}) \bmod 1 \\ y_{n+1} &= y_n - \frac{k}{2\pi} \sin(2\pi x_n) \end{aligned} \tag{10}$$

For low values of k, the dynamics of the system are ordered, with initial points converging to cyclic orbits which remain bounded on the y axis (see Fig. 7a–b). As k increases, islands of chaotic dynamics begin to appear (see Fig. 7c–h). The map has a critical point at $k_c \approx 0.972$. For $k > k_c$, the chaotic islands are fully connected along the y axis; meaning that, in principle, it is possible to follow a chaotic orbit from $y = 0$ to $y = 1$. However, the permeability of the central region increases only slowly as k moves past k_c [89] (see Fig. 7d–f). As an example of this, when $k = 1.1$, using 1000 randomly chosen initial points and an upper limit of 10^6 iterations, we measured a median transit time of 64000 iterations of Equ. 10 to move from the bottom to the top of the map, with 27% of trajectories not reaching the target within the upper limit.

Following the examples of [89] and [90], the goal is to find a controller which can navigate from the bottom to the top of the standard map in the shortest number of steps. In order to do this, the ABN is allowed to modulate parameter k in Equ. 10 within the range $[1.0, 1.1]$. We use the same initial ([0.45,0]→[0.55,0.05]) and target ([0.45,0.95]→[0.55,1]) regions as used in [90]. Inputs to the ABN are the current position (x, y) and the Euclidean distance from the top-centre of the map, and the single output is the new value of k. Distance is defined to be the control signal when a CABN

is used. The evolved ABNs are evaluated on 20 random points within the initial region. Fitness is the mean number of steps required for these trajectories to reach the target region. Trajectories which do not reach the target region within 1000 steps are assigned an arbitrary figure of 2000 steps, biasing search towards controllers effective over all initial conditions. A population size of 200 and a generation limit of 50 are used.

B. Controlling Hybrid Dynamics in a Legged Robot

Many real world systems do not have purely continuous or discrete dynamics, but rather a hybrid of the two [91]. These often occur on different time scales, such that continuous state flow is occasionally interrupted by jump discontinuities caused by the occurrence of discrete events. Two common examples of this are physical systems with impact, such as a bouncing ball, and switched systems, where a signal change causes a discrete change in behaviour. The robotic locomotion task described below features elements of both of these.

Generating legged locomotion gaits is a challenging problem which involves simultaneously solving the two tightly coupled problems of support and progression [20]. For practical purposes, the problem can be made more tractable through the use of high-level primitives (e.g. sinusoidal functions [92]), morphological features such as rotary legs [93], and servo-motors [94]. However, in this work we are interested in how legged locomotion gaits can be generated through direct control of the actuators. In this respect, the task is similar to that addressed in [20], though a notable difference is that we evolve a single controller to control all actuators at once, with no explicit knowledge given regarding their morphological relationships. Furthermore, in order to transform the problem into a switched system, the controller is required to be able to reverse the robot's direction of movement in response to an external signal.

1) Legged robot: The robot (see Fig. 8) is purposely very simple in design, comprising a square top section with four legs connected by actuators at the corners. The actuators are limited to movement in the x-axis plane, with a maximum elevation of 60° from vertical, a maximum angular velocity of 3m/s, and a maximum torque of 150Nm. The robot is simulated using the Open Dynamics Engine (ODE) physics engine, with a step size of $\Delta_t = 0.05$s, friction of 200N, CFM (an ODE parameter) of 10^{-5}, and standard gravity. Motor characteristics are chosen so that all actuators have to work in co-ordination to move the robot, forcing the controller to use a quadrupedal gait rather than, for instance, a dragging motion.

ABNs have five inputs, corresponding to the actuator angles and the direction signal, and four outputs, which are used to set the torques of the actuators during the next simulation period. The requirement to map angles to torques adds an extra degree of difficulty to the task. For CABNs, the direction signal is used as the AGN's control input. The population size is 500, with a generation limit of 100. We use a total evaluation period of 2000 time steps and a between-update simulation period of 10 steps.

2) Objective function: The ABN controllers are required to generate a quadrupedal gait that moves the robot as fast as possible in a given direction. Upon receiving a signal, the robot is required to change direction by 180°, and then move as fast as possible along this new heading. Controller fitness is measured over a sequence of epochs $< e_0, ..., e_{N-1} >$, each with a random duration between 300 and 600 time steps, with the required direction of movement reversing during subsequent epochs. The objective function f_R is defined:

$$f_R = \frac{t_{\max} - t_{\min}}{N} \min \left\{ \sum_{\substack{n \in \mathbb{N}_{\text{even}} \\ n<N}} p(n), \sum_{\substack{n \in \mathbb{N}_{\text{odd}} \\ n<N}} p(n) \right\} \quad (11)$$

where $t_{\max}$ and $t_{\min}$ are the maximum and minimum bounds on epoch duration and $p(n)$ is the progress made during epoch n, defined:

$$p(n) = \frac{d_n}{t_n} \left(2 \frac{\eta_b(e_n, e_{n+1})}{\pi} - 1 \right) \left(1 - \frac{\eta_w(e_n)}{\pi} \right) \sigma_n \quad (12)$$

where d_n is the distance travelled during epoch n, t_n is the duration of epoch n, η_b is the difference in mean heading between two epochs, η_w is the difference in heading within an epoch (as measured during the first and last 50 time-steps of the epoch), and σ_n is a penalty for non-movement: equal to 1 if the robot has not moved for 100 subsequent ABN updates in epoch n, and 0 otherwise.

In effect, progress is the mean velocity in the required direction, with penalties for turning during an epoch and for non-movement. Assuming movement in a straight-line and no stopping, fitness is equivalent to the expected distance covered during an epoch in the forward or backward direction, whichever is shortest.

C. Evolutionary Algorithm

ABNs are evolved using a fairly standard generational evolutionary algorithm. Special-purpose techniques, such as those developed for neuro-evolution [5], graph induction [95] and multi-objective problems [96], may be able to find better solutions to these tasks. However, we intentionally chose to keep the evolutionary algorithms simple in order to avoid complicating the comparison between different ABN models.

The ABNs are linearly encoded as shown in Fig. 9. This represents the ABN as an array of genetic units, followed by initial gene expression and chemical concentrations (where

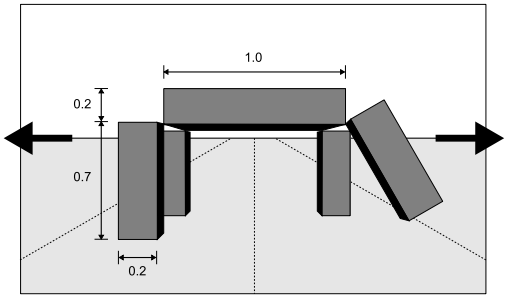


Fig. 8. Quadruped robot simulated in Open Dynamics Environment. Arrows indicate the direction of movement along the x-axis plane.

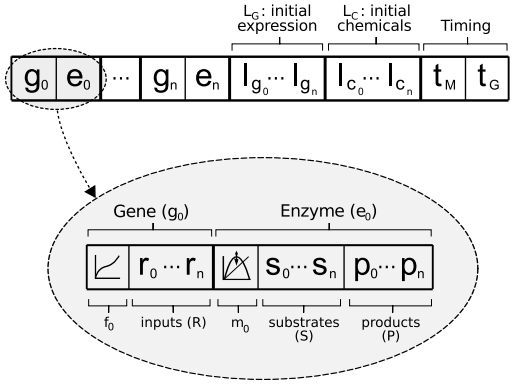


Fig. 9. Linear encoding of a CABN used by the evolutionary algorithm, also showing (inset) how individual genes and enzymes are represented.

applicable) and timing information (t_G and/or t_M). Each genetic unit has an optional regulatory region and an optional coding region. In a coupled network, the regulatory region encodes the gene (g_i) and the coding region encodes the enzyme (e_i) which it expresses. Where a gene does not express an enzyme (such as in an AGN), the coding region is empty. For an AMN, where there are no genes, the regulatory region is empty. Inputs and outputs (R_i, S_i and P_i) are represented by absolute references to indices. Function parameters (e.g. slopes, input weights) and initial values are represented as floating-point values.

The number of genes and enzymes is fixed, except in the case of CABNs, where individual genes may or may not express an enzyme. Again, this is to avoid complicating the interpretation of the results, since variable-length evolutionary algorithms can display unexpected solution size pathologies (as we recently observed when evolving ABNs for time series classification [97]). Based on the results of initial experiments, for all tasks the solution lengths are chosen to be more than sufficient to express a valid solution.

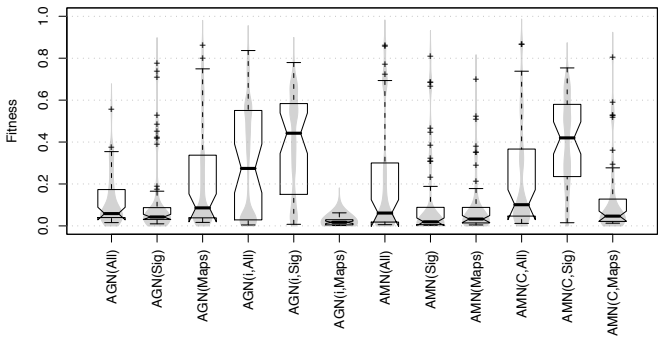
The timing variables, t_G and t_M, are bounded within the interval $[1, 100]$. Again, this upper limit is more than sufficient to express valid solutions, and is intentionally permissive. There is no evolutionary pressure towards time-efficient solutions.

The evolutionary algorithm uses tournament selection (tournament size 4) and elitism (size 1). Child solutions are generated using either uniform crossover or mutation in the ratio 1:4. Crossover points (p=0.15) always fall between genetic units. Real-valued elements are mutated (p=0.015/element) using a Gaussian distribution centred around the current value. Gene regulatory inputs and enzyme substrates may be added or removed (p=0.015/element), with new indices chosen either randomly (p=0.5) or by duplicating an existing index (p=0.5). Operator probabilities were chosen based on experience, and have not been optimised for individual problems.

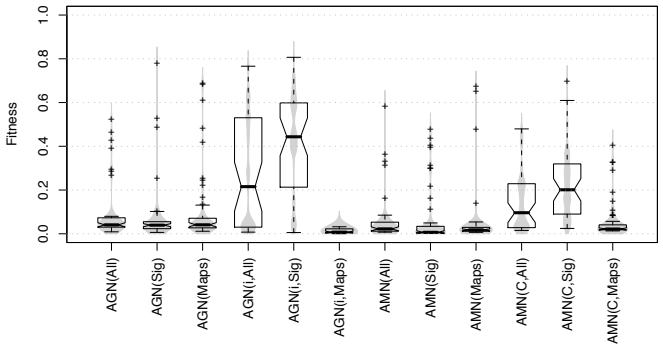
VI. Results

A. Controlling the Lorenz System

Figs. 10 and 11 show fitness distributions of controllers evolved to carry out state space targeting in the Lorenz system. Fig. 10 summarises the performance of uncoupled



(a) Solution length 10



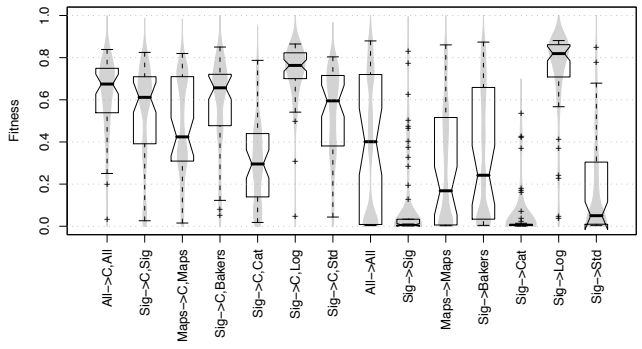
(b) Solution length 20

Fig. 10. Fitness distributions for uncoupled ABNs carrying out state space targeting in the Lorenz system. High numbers are better. Notched box plots show summary statistics over 50 runs. Overlapping notches indicate when median values (thick horizontal bars) are not significantly different at the 95% confidence level. Kernel density estimates of underlying distributions are also shown (in grey). Abbreviations: Sig=sigmoid, All=all functions (maps and sigmoids), C=conservation of mass, i=distance signal delivered to inputs.

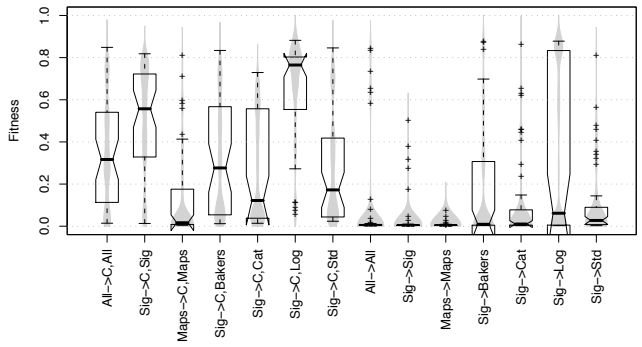
AGN and AMN-based controllers, illustrating the effect that solution size, function choice, mass conservation (for AMNs), and signal destination (for AGNs) have upon fitness. Fig. 11 summarises the performance of coupled ABNs. In addition to function choice, mass conservation and signal destination, this also shows the effect that the coupling function (Section IV-C) has upon fitness.

Overall, coupling has a positive effect upon controller fitness. Comparing CABNs (each composed of an AGN and AMN of length 10) with standalone AGNs and AMNs (of length 10 and 20 for a fair comparison), it is clear that the fitness distributions are significantly shifted towards the optimum for most CABN parameterisations. For the CABNs, the coupling function (Figs. 11a–c) also has a considerable impact upon fitness distributions, with proportional concentration updates leading to the best controllers on average and thresholded updates leading to lower maximum fitness.

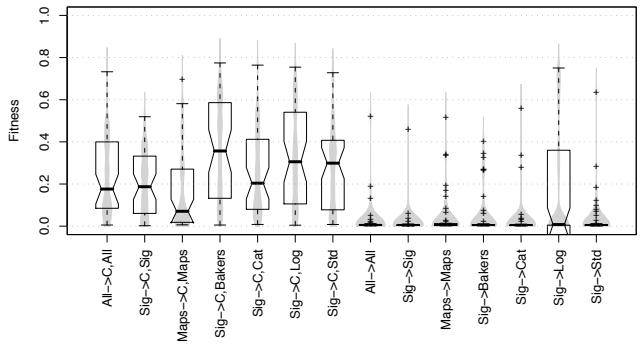
Mass conservation appears to be important when evolving both standalone AMNs and CABNs. For the AMNs, fit controllers were very hard to find when the conservation rule was not applied. For CABNs, conservation both improves mean fitness and makes evolution less sensitive to sub-optimal update rules and function choices. However, it should be noted that conservation does not lead to overall fitter solutions—in fact, for CABNs with proportional updates (Fig. 11a), the fittest solutions were found when conservation was not used.



(a) Proportional updates



(b) Scaled updates



(c) Thresholded updates ($\tau = 0.5$)

Fig. 11. Fitness distributions for coupled ABNs carrying out state space targeting in the Lorenz system. CABNs are defined according to the notation AGN→AMN. Abbreviations: Log=logistic map, Std=standard map.

Delivery of signals to regulatory inputs has a similar effect upon AGNs. For uncoupled AGNs, much fitter solutions were found, on average, when the distance signal was delivered as a regulatory input to each gene, rather than via initial expression state. A similar pattern was found for CABNs, with few fit solutions found when the distance input was delivered via gene expression (results not shown).

Choice of regulatory and enzyme functions also has a large impact upon fitness distributions. For the standalone ABNs, sigmoids performed fairly well on average. With the notable exception of the logistic map, good discrete map-based AGNs and AMNs were hard to find. Near-optimal solutions could only be found when using logistic maps. For CABNs, on average the best controllers were found when using a sigmoidal AGN coupled to a discrete map-based AMN. In Fig. 11 we show the effect that choice of discrete map, in this

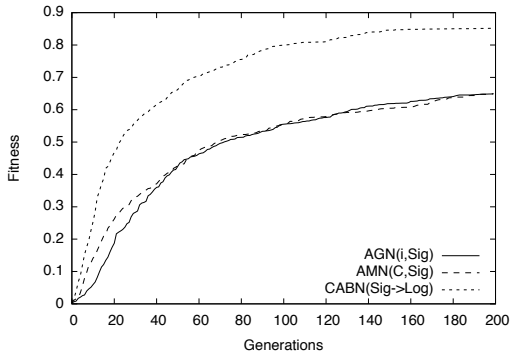


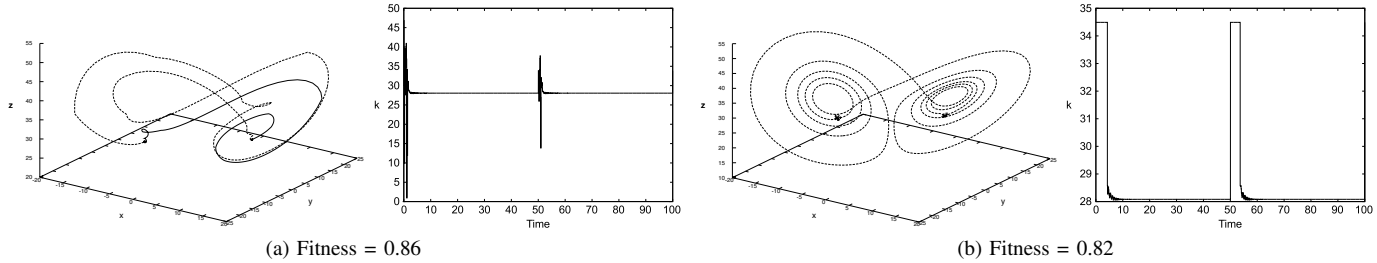
Fig. 12. Fitness evolution of solutions to the Lorenz task, averaged over 50 runs, for the best AGN, AMN and CABN parameterisations.

case, has upon the fitness distributions. As with standalone ABNs, it is clear that CABNs with logistic map-based AMNs are the easiest to find. However, good solutions can still be found when using each kind of map individually.

Fig. 12 compares the fitness evolution of the best AGN, AMN and CABN parameterisations over a longer period of time, showing that, on average, Sig→Log CABN runs converge much faster than the standalone ABN runs, and also appear to approach a higher fitness level. For this task, there is no significant difference in evaluation times for the different ABN models—all runs take ~ 2 minutes to complete on a 25 core cluster, with simulation of the Lorenz equations being the dominant factor. Given the lack of parsimony pressure, there is considerable variance in the evolved timing variables, t_G and t_M. For the CABN runs, for instance, the AMN is iterated 55 times (s.d. 26) for each control step, with the AMN iterated once every 30 iterations of the AMN (s.d. 22). Although the standard deviations are large, this does suggest that the genetic components of evolved CABNs are operating over considerably slower time-scales than the metabolic components.

Fig. 13 shows examples of how trajectories are controlled by evolved ABNs. The fittest controllers follow behaviours similar to those shown in Figs. 13a and b. The former uses a continuously-varying control signal that modulates the dynamics through both ordered and chaotic regimes, guiding the trajectory towards the control point. The latter uses a fixed control signal to push the trajectory into a chaotic orbit, which then naturally approaches the control point. In both cases, as the trajectory approaches the control point, the control signal becomes continuously-varying, with a gradually decreasing magnitude that finally disappears when $\rho = 28$. In this respect, the strategy carried out near the fixed points resembles Pyragas' method [82]. In Figs. 13c–d we show two other control strategies used by evolved ABNs. These involve significantly longer transient periods, but both are valid, and illustrate the diversity of the evolved strategies.

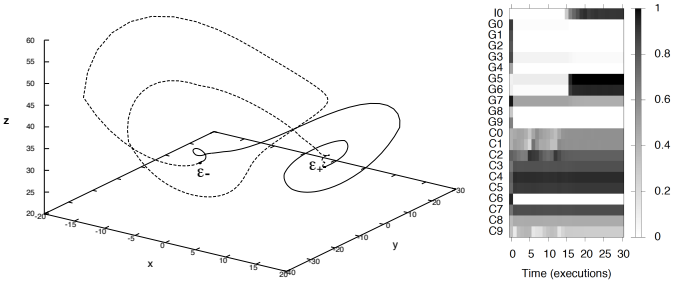
We expected that coupled networks would improve the performance of controllers by allowing rapid switching between different behaviours. Analysis of the behaviour of evolved controllers suggests that rapid switching does take place within the coupled networks. An example is shown in Fig. 14b where the turning on of the distance signal prompts a discrete change



(a) Fitness = 0.86 (b) Fitness = 0.82

(c) Fitness = 0.36 (d) Fitness = 0.18

Fig. 13. Examples of evolved controllers moving between unstable points in the Lorenz system via control of the Rayleigh parameter (ρ), showing (left) the controlled trajectory, and (right) how the Rayleigh parameter is modulated. (a) and (b) show typical control strategies used by high-fitness discrete map and sigmoidal ABNs, respectively; (c) and (d) show examples of more exotic control strategies used by evolved controllers.



(a) Controlled trajectory (b) Time series

Fig. 14. Example of a coupled ABN guiding a trajectory, showing (a) the trajectory being guided from ϵ_- to ϵ_+(broken line) then back to ϵ_-(unbroken line); (b) as the trajectory approaches ϵ_+, (at $t \simeq 15$), the distance signal (I0) starts to turn on, causing a change in gene expression ($G0 - G9$), which leads to a change in the metabolic state ($C0 - C9$) and the control output (C9).

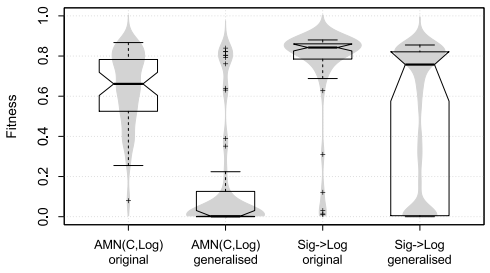
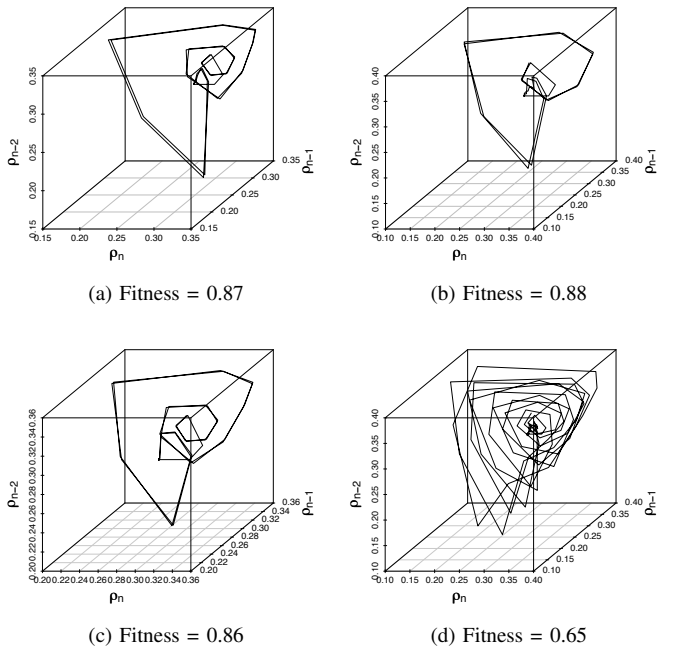


Fig. 15. Fitness of evolved controllers on original and generalised tasks (left, fittest uncoupled; right, fittest coupled).



(a) Fitness = 0.87 (b) Fitness = 0.88

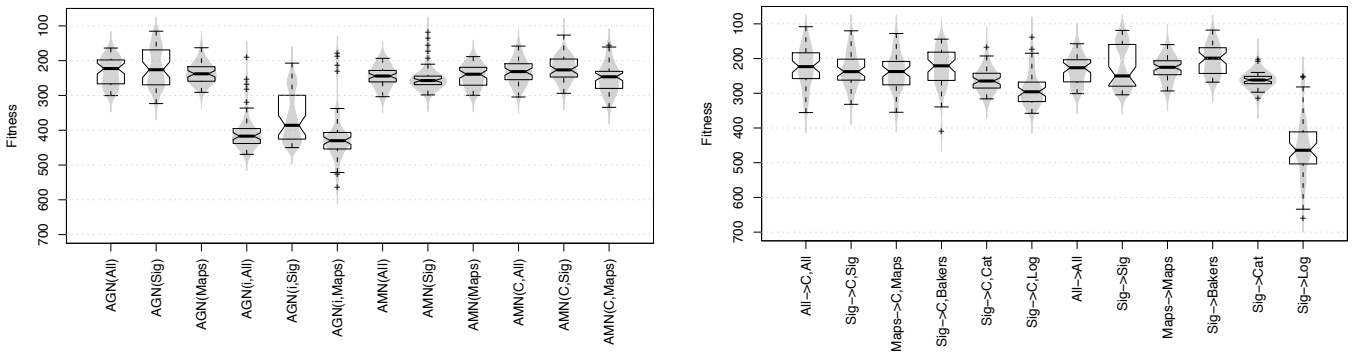
(c) Fitness = 0.86 (d) Fitness = 0.65

Fig. 16. Attractor structure of four different evolved ABNs controlling trajectories in the Lorenz system. These phase portraits were reconstructed using delay embeddings of each network's output (ρ) time series. According to Taken's embedding theorem [98], the resulting phase portrait is topologically equivalent to the system's true dynamical behaviour.

in gene expression state, leading to a change in the network's metabolic behaviour. This change in expression takes place when the trajectory approaches an equilibrium point, causing the controller to flip from a steering behaviour into a stabilising behaviour where it attempts to keep the trajectory at the equilibrium point.

We also looked at the generality of evolved controllers, by requiring them to repeat the original task twice in succession (i.e. $\epsilon_- \rightarrow \epsilon_+ \rightarrow \epsilon_- \rightarrow \epsilon_+ \rightarrow \epsilon_-$). Fig. 15 compares the performance of $\mathsf{AMN}_{\mathsf{C,Log}}$ and $\mathsf{Sig} \rightarrow \mathsf{Log}$ controllers, the fittest uncoupled and coupled ABNs, upon this task. It is evident that the coupled solutions are far more likely to solve the general problem of moving repeatedly between the equilibrium points.

Complex dynamical systems are inherently difficult to analyse, and computational dynamical systems are no exception. However, techniques from the field of non-linear time series



(a) Uncoupled ABNs (b) Coupled ABNs

Fig. 17. Fitness distributions for uncoupled and coupled ABNs carrying out state space targeting in Chirikov's standard map. All networks are of length 10. Distributions towards the top of the figures are better. See Figs. 10 and 11 for abbreviations.

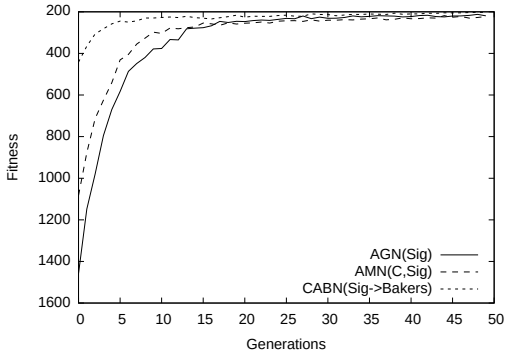


Fig. 18. Fitness evolution of solutions to the standard map task, averaged over 50 runs, for the best AGN, AMN and CABN parameterisations.

analysis [98] can be used to provide some understanding of the dynamical behaviour of these kinds of systems. Phase space reconstruction is one such technique, providing a means of visualising the system's attractor structure. Fig. 16 shows representative examples of the reconstructed phase spaces[3] of evolved controllers. In general, we found that the better solutions have very similar attractor structure. High fitness controllers also appear to be largely symmetrical, carrying out similar behaviours whether moving from ϵ_- to ϵ_+ or vice versa. Solutions with lower fitness displayed less well-defined attractors, but with a similar topology to the fitter controllers (e.g. Fig. 16d). This may explain the differences in generality seen in Fig. 15, with coupling leading to higher fitness solutions whose well-defined attractors are more likely to generalise.

B. Controlling Chirikov's Standard Map

Fig. 17 shows the fitness distributions of both coupled and uncoupled ABNs evolved to carry out state space targeting in Chirikov's standard map, summarising the effect that coupling, function choice, mass conservation (for AMNs and CABNs), and signal destination (for AGNs) have upon fitness. We do not show the effect of the coupling function, since this had minimal impact upon performance. The best controllers had a median path length of $\sim$110 steps (an example path is shown in Fig. 19). This is broadly similar to the trajectory mapping method described in [90], where the authors calculated orbits which traverse the standard map in $\sim$125 steps when $k = 1.25$.

Unlike for the Lorenz system, there appears to be no significant advantage to using coupling when solving this task. Function choice also has a much smaller impact: both sigmoid and map-based ABNs could solve the task well. Interestingly, logistic map-based ABNs perform very poorly when used to solve this problem, in stark contrast to the Lorenz task. The exception to this is when mass conservation is used—again, we find that mass conservation tends to reduce the impact of poor function choice. Signal delivery in AGNs also appears to have the opposite effect to the one seen in Lorenz controllers, with regulatory input delivery leading to considerably lower fitness on average. However, it is perhaps no surprise that state space targeting in Chirikov's map requires very different controllers to state space targeting in the Lorenz system, since the two arguably lie at opposite ends of the dynamical systems spectrum.

Fig. 18 compares the fitness evolution of the best AGN, AMN and CABN parameterisations. As in the Lorenz task, CABN runs converge significantly more rapidly than standalone ABN runs. However, CABN runs take approximately twice as long ($\sim$ 4 minutes) to complete as standalone ABN runs ($\sim$ 2 minutes), since the CABNs have more functional elements and the simulation time for the standard map is trivial. Again, we find that the metabolic components ($t_M = 51$ s.d. 25) of the evolved CABNs operate at a considerably faster rate than the genetic ($t_G = 30$ s.d. 20).

The uniformity of fitnesses on the standard map task indicates that the dynamics required to solve this task are readily expressed using a range of network types and low-level functions. Reconstructed phase portraits (see Fig. 20) tend to support this assertion, showing that the topological structure of the evolved control strategies is relatively simple, and that (unlike the Lorenz task) it does not require well-defined attractors to solve the task.

In general, we have noticed that evolved controllers transition between different dynamical behaviours as they traverse the standard map. Fig. 21 illustrates this phenomenon, showing

[3]See [99] for a discussion of methods for phase space reconstruction, including the use of delay embeddings, and the projection of principal components used later in this article.

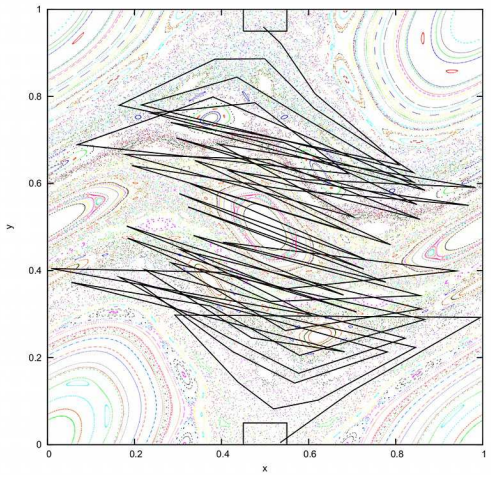
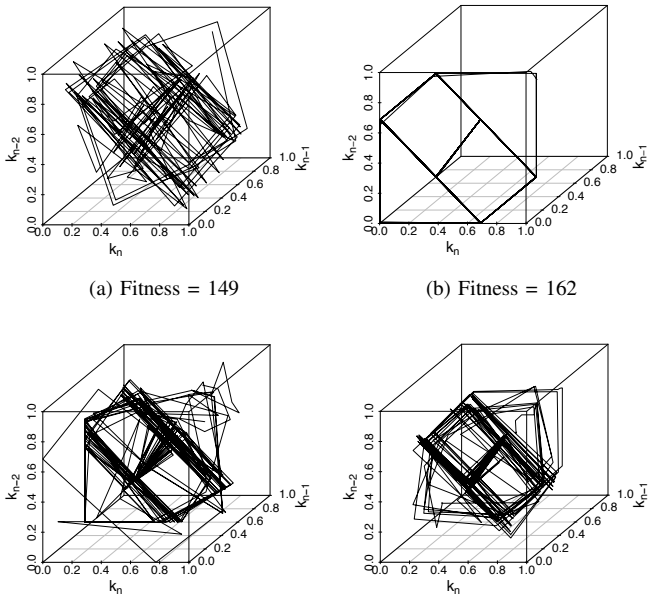


Fig. 19. Example behaviour of an evolved controller guiding a trajectory from a region at the bottom of the standard map to a region at the top in 83 steps. Chirikov's standard map is plotted for $k = 1.1$.

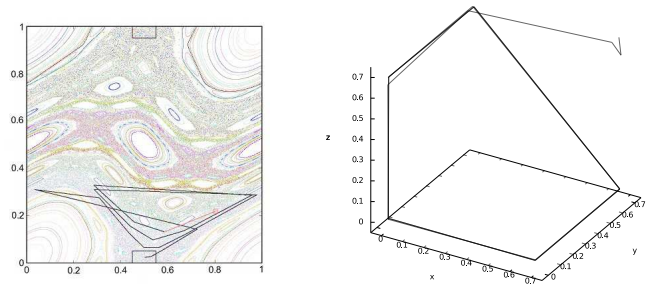


(a) Fitness = 149 (b) Fitness = 162

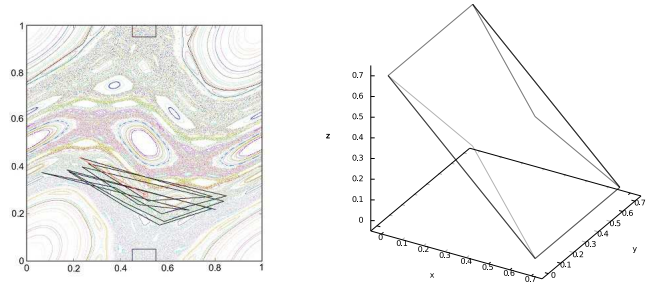
(c) Fitness = 166 (d) Fitness = 166

Fig. 20. Phase portraits produced from delay embeddings of output k, showing four different evolved ABNs controlling trajectories in Chirikov's standard map.

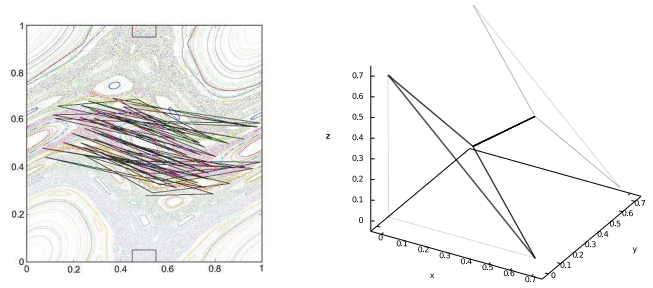
the sequence of movements within the controller's phase space as it moves from the lower chaotic region, through the mixed dynamics of the central region, and finally through the upper chaotic region. In this case, it can be seen that movement through the different regions of the map corresponds to movement through particular components of the controller's trajectory. This example illustrates that, using an uncoupled ABN, changes in dynamical behaviour can be achieved without requiring changes in the network structure, and hence that coupling is not necessary in all situations where there is a requirement to move between different sub-tasks.



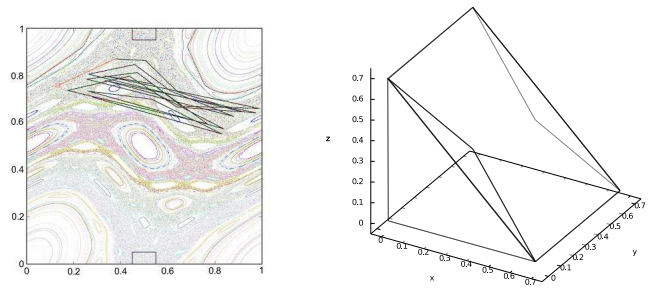
(a) Steps 0–20



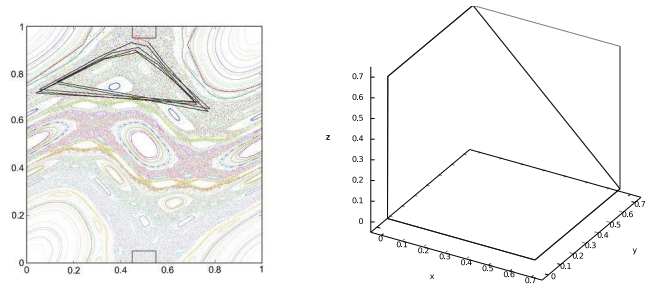
(b) Steps 20–40



(c) Steps 40–114

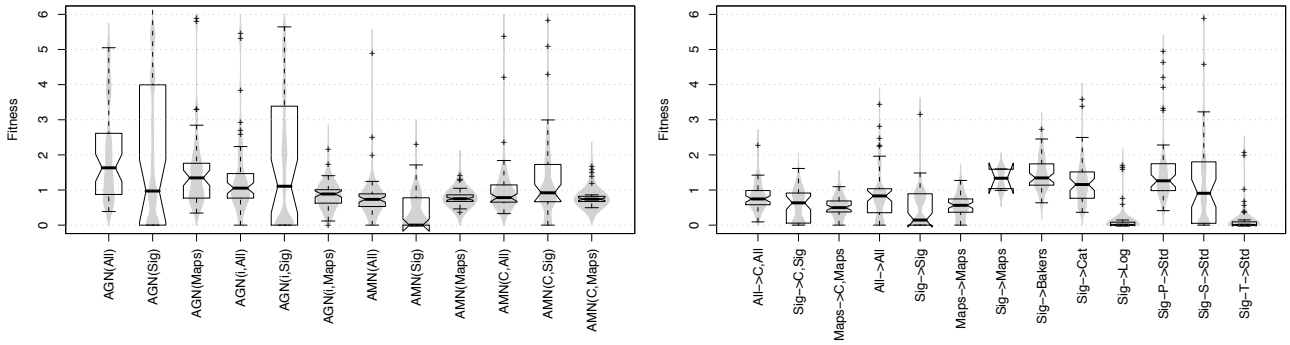


(d) Steps 114–140



(e) Steps 140–165

Fig. 21. Correspondence between the trajectories followed in Chirikov's standard map and the controller's reconstructed phase space, showing three dynamical regimes which roughly correspond to the top and bottom chaotic regions (a and e), the central region (c) and the interface regions (b and d).



(a) Uncoupled ABNs (b) Coupled ABNs

Fig. 22. Fitness distributions for uncoupled and coupled ABNs controlling the gait and direction of movement of legged robots. High numbers are better. The effect of the different coupling functions are illustrated for Sig→Std CABNs, where P=proportional, S=scaled, and T=thresholded. See Figs. 10 and 11 for other abbreviations.

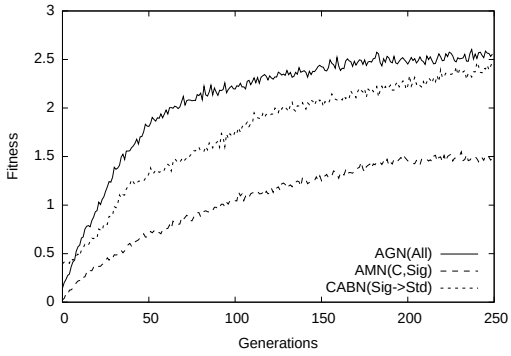


Fig. 23. Fitness evolution of solutions to the robotic locomotion task, averaged over 50 runs, for the best AGN, AMN and CABN parameterisations.

C. Controlling Legged Robots

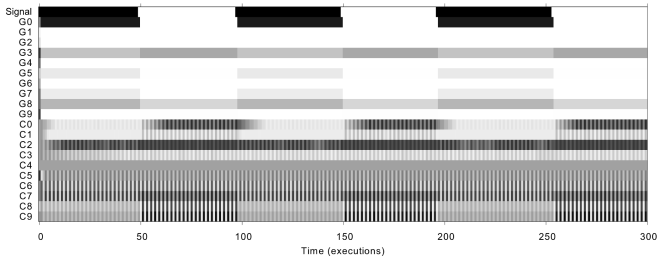
Fig. 22 shows the fitness distributions for controllers evolved to solve the robotic control task. Whilst there is little significant difference between the average fitnesses for different models and functions, the fitness distributions vary markedly. Uncoupled sigmoidal ABNs lead to the largest number of high-velocity gaits, but also produce a relatively large proportion of failed runs. These failed runs have fitnesses of zero, corresponding to motion in only a single direction. For the coupled networks, we found that sigmoidal AGNs coupled to discrete map AMNs lead to the best performance. These generate relatively few failed runs, but also produce fewer high-velocity solutions than the sigmoidal AGNs.

Fig. 23 shows the fitness evolution of the best AGN, AMN and CABN parameterisations over a longer period of time. Although CABNs have a higher average fitness initially, the AGN runs converge faster, with both AGNs and CABNs approaching the same average fitness after 250 generations. Standalone AMN runs plateau at a much lower average fitness. Run times are dominated by the simulation time of the physics engine, taking an average of 9 minutes per run for all ABN models. Timing distributions for the CABNs are similar to the other two tasks ($t_M = 56$ s.d. 26, $t_G = 27$ s.d. 20).

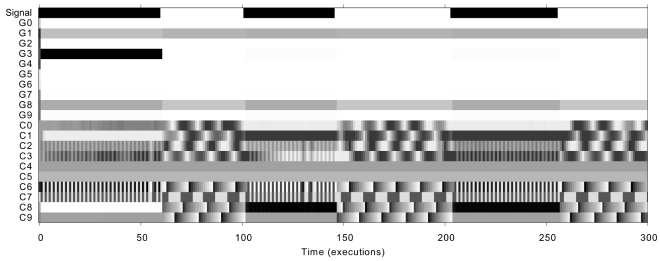
In general, we would argue that discrete map solutions are more likely to solve the problem, but sigmoidal networks are more likely to find a good solution to the problem. This, in turn, suggests a trade-off between expressiveness and evolvability. Randomly-generated discrete map ABNs often exhibit complex behaviours, but tend to be less evolvable than sigmoidal ABNs. Conversely, randomly-generated sigmoidal ABNs are far more likely to display simple ordered behaviours, but tend to be more evolvable than discrete map ABNs. Consequently, it is perhaps unsurprising that the highest average fitness is found when evolving AGNs with mixed sigmoids and discrete maps: the discrete maps presumably increase the expressiveness of initial solutions, whereas sigmoids allow the behaviours of the network to be more effectively tuned by evolution.

As mentioned above, for the coupled networks a combination of sigmoidal AGNs and discrete map AMNs were found to be most effective. Furthermore, the best solutions were found when using a single type of discrete map in the AMN, rather than a mixture of all the maps. Mirroring the results from the standard map control task, the logistic map was found to be the least effective, leading to very few correct solutions. The baker's map, Arnold's cat map, and Chirikov's standard map all performed comparably, with the latter generating the highest number of high-velocity solutions. It is particularly interesting to note that the purely chaotic maps are able to generate effective gaits for what appears to be, at least on the surface, an ordered task. One explanation for this is that chaotic dynamics are inherently more responsive to control signals. Another explanation is that, by exploring a large number of unstable periodic orbits, chaotic controllers are more able to escape from environmental or internal conformations in which an ordered controller would become stuck. Similar arguments have been posited by the authors of [100].

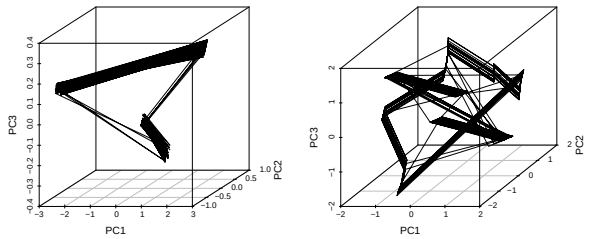
Fig. 24 shows two examples of Sig→Maps CABNs controlling gait and responding to the direction signal. In both cases, the AMN appears to be responsible for generating appropriate patterns of actuator movements and the AGN appears responsible for switching between different patterns by regulating the influence of different enzymes. Hence the genetic network responds to slow changes in the governing dynamics (i.e. the switching direction signal), and the metabolic network responds to fast changes (i.e. leg position).



(a) Time series of a CABN's internal state. The Signal input specifies the required direction of movement, GO–G9 are the gene expression levels, and C0–C9 are the chemical concentrations. In this example, the AMN generates a single cyclic pattern (C5) which is then scaled and propagated to the outputs (C6–C9). The scaling for each output (and hence the direction of the resulting gait) is determined by the current gene expression pattern.

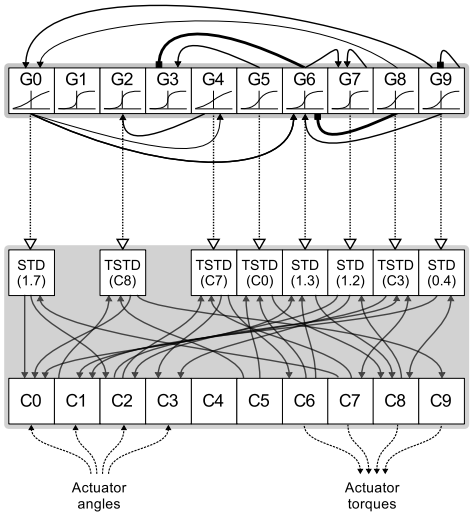


(b) In this second example, a different AMN generates two different cyclic patterns (bunny hopping and a four-legged wading movement), which the AGN switches between in response to changes in the direction signal.

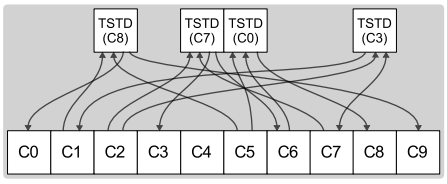


(c) Projections of the first three principal components of the time series data, showing phase space portraits corresponding to (a) and (b). Each show two attractors, corresponding to the two directions of movement, with transient paths showing where direction changes have taken place.

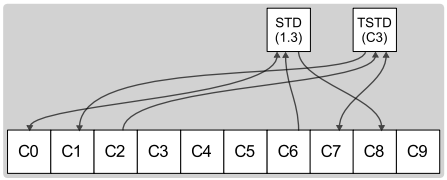
Fig. 24. Examples of the internal states, and corresponding phase portraits, of evolved CABNs controlling the direction and gait of a legged robot.



(a) In the AGN (top), arrows indicate up-regulation, blocks indicate down-regulation, and edge thickness shows interaction strength. In the AMN (bottom), STD indicates a standard map (with k in parentheses), TSTD indicates a tunable standard map (with tuning input in parentheses).



(b) Forwards



(c) Backwards

Fig. 25. Example of a Sig→Std CABN evolved to solve the bidirectional robot control task, showing (a) the overall architecture, and (b–c) the enzymes which are actively involved in controlling the dynamics during the forwards and backwards phases. For each execution, the AGN is iterated once and the AMN is iterated 66 times.

Fig. 24c shows that the evolved networks both display two attractors, with trajectories switching between the attractors following changes in the direction signal. The phase space of the second controller is somewhat atypical, displaying a relatively long cyclic attractor for one direction of movement and a shorter, but two-lobed, attractor for the other. It can be seen that there are a number of paths between the two attractors, corresponding to changes in direction occurring whilst the robot is in a number of different conformations. In part this underscores the difficulty of the problem, since the network must correctly handle transitions between the attractors regardless of the current conformation of the robot in order to maximise fitness. As a consequence of this, evolution tends to favour solutions with short-period attractors, such as the leftmost example in 24c.

In Fig. 25, we show the architecture of an evolved Sig→Std CABN. By analysing the internal state of the network during execution, we were able to identify which subsets of the enzymes are used to control movement in each direction (25b–25c). We focus our analysis on the backwards direction, which is, architecturally, the simpler of the two. It uses two enzymes: a $k \approx 1.25$ standard map which maps $\{\mathsf{C0}, \mathsf{C6}\}$ to $\{\mathsf{C8}, \mathsf{C0}\}$, and a tunable standard map, where k is the current value of C3, and which maps $\{\mathsf{C7}, \mathsf{C2}\}$ to $\{\mathsf{C7}, \mathsf{C1}\}$. Both maps are recursive, with a single output feeding back to the input on the next iteration. The first map receives the angle of the first leg as input, and controls the torque of the third leg. Its second output is fed back to its first input, and the map operates in a majority-chaotic phase (see Fig. 7g). The second map receives inputs from the angles of the third and fourth legs, with the latter controlling the map's dynamics and placing it in a majority-ordered phase. It controls the torque of the second leg, and this value is fed back to the map's inputs. Hence, it can be seen that control is achieved using a combination of ordered and chaotic dynamics.

VII. Discussion

The results show that ABNs are capable of controlling a diverse range of complex dynamical systems. However, they also show that different types of ABN model are appropriate for different types of problem—and, more specifically, that no particular model parameterisation is always better than all other model parameterisations. This is perhaps unsurprising, since we might expect different ABN architectures to more readily produce different kinds of dynamics which, in turn, are more appropriate for different kinds of problem. Nevertheless, it highlights a more general guidance that, when faced with a particular problem, we should not assume that a particular computational dynamical system will be able to solve it well.

Perhaps the most significant observation produced by our experiments is that non-linear discrete maps are generally beneficial to performance. ABNs with discrete maps were the most effective on both the Lorenz and robot control tasks, and no worse than sigmoidal ABNs on the easier to solve standard map task. We can speculate that they provide benefits in two ways: (i) by producing complex, tunable behaviours without requiring evolution of sub-networks; and (ii) by providing a ready source of chaotic dynamics, which may increase the responsiveness and behavioural diversity of controllers. It is less clear which maps are the most beneficial. The logistic map is certainly beneficial for controlling the Lorenz system, but logistic map ABNs performed relatively poorly on the other tasks. The standard map is perhaps the most beneficial in general, since standard map ABNs performed well on both the Lorenz and robotic task. The standard map also seems a particularly relevant choice from a theoretical perspective, since it possesses a form of behavioural universality. The benefits of discrete maps in these experiments support the (admittedly few) previous observations regarding the use of coupled map lattices for computation. It also hints at the potential benefits of using discrete maps within other computational dynamical systems.

In terms of model architecture, it is difficult to draw any firm conclusions. AGNs, in general, perform better than uncoupled AMNs[4]. However, AMNs often perform a lot better when a mass conservation rule is applied, in which case their fitness distributions are more similar to AGNs. We can hypothesise that by forcing covariance between the chemical concentrations, mass conservation causes a reduction in the number of effective variables in the system, reducing the search effort required to find viable solutions. It also serves to spread input signals throughout the network, which may be beneficial when the dynamics need to be sensitive to external perturbations. The importance of signal delivery is also evident in the performance of AGNs, where the delivery of inputs via gene expression or regulatory inputs has a significant effect upon the evolutionary algorithm's ability to find solutions (see [103]). In response to this, we are currently looking at whether artificial signalling networks might provide a means of automatically delivering signals to appropriate locations within AGNs and AMNs, mirroring their role in biological systems [104].

Coupling led to significantly better control in the Lorenz task, faster convergence in the standard map task, but had no overall benefit for the robotic task. Consequently, the results suggest that a higher-order architecture, such as this, can be advantageous in certain situations. We have speculated that such a coupled architecture could have benefits when a controller needs to switch rapidly between different tasks, and an analysis of coupled controllers solving the Lorenz and robot tasks tends to support this. In other work, we are looking at an alternative approach to handling this kind of task switching which uses an analogue of epigenetic chromatin remodelling [105].

As we noted in Section IV-A, sigmoidal AGNs generalise a number of RNN architectures, including the commonly used Elman and Jordan networks. It is therefore notable that, although uncoupled sigmoidal AGNs were generally competitive across the tasks we looked at, they were beaten by other ABN models on both the Lorenz and robot control tasks (the harder problems in terms of computational effort required to solve them). This indicates that there is an advantage to using connectionist architectures motivated by the structure and organisation of biochemical networks, as opposed to those motivated by neural networks. However, we should be careful not to read too much into this, since (as noted above) different architectures are suitable for different problems. In robotic control, for instance, work on RNNs has shown that fixed topologies can be beneficial for certain constrained tasks, such as oscillator networks in the case of regular motion [7] or topologies motivated by certain brain regions in the case of maze solving [6]. The same is likely true for ABNs, and we could speculate that fixed topologies, motivated by actual biochemical pathways, would be beneficial in certain circumstances. A more progressive interpretation of our results, therefore, is that research on ABNs could complement neural computation, by inferring sources of connectionist knowledge not found in brains, but rather in the biochemical networks of cells and tissues. For example, our findings on the beneficial roles of discrete maps and higher-order coupling could readily be applied to RNN models.

As the title suggests, a more general focus of this paper is on the use of evolved computational dynamical systems to control dynamical systems. Previous work in this area has suggested that a wide variety of computational dynamical systems can be used to perform control [5], [7], [27], [28], [31], [32]. Our results build upon this by demonstrating that a certain class of computational dynamical systems can be evolved to control dynamical systems that exhibit a wide range of properties, notably discrete, continuous and hybrid-time dynamics; dissipative and conservative dynamics; and ordered and chaotic dynamics. As such, they suggest that this kind of approach is applicable across a large spectrum of system types. Moreover, they show that controllers can be induced without any explicit knowledge of the underlying system which is being controlled (a theme which has also previously been explored, e.g. [106]). A common criticism of this kind of approach is that the

[4]An interesting exception, explored in [101] is when AMNs are used with the more biologically-plausible Michaelis Menten function, in which case they outperform AGNs when controlling dynamics in the Lorenz system. We have also found that AMNs, with mass conservation, outperform AGNs when used for time series classification [102].

induced controllers behave as black boxes whose behaviour can not be readily understood. However, we have shown how relatively simple analytical techniques, such as phase space reconstruction, can give insight into the behaviour of evolved ABNs; in future work, we aim to strengthen this approach with other analytical tools, such as network analysis techniques [107], and methods of finding equivalent (or approximately equivalent) computational automata [108], [109].

The dynamical systems addressed in this paper are representative of the kind of dynamics that occur in the real world. However, true real world systems are typically composed of multiple, diverse, dynamical components. For practical reasons, it is generally not feasible to directly evolve controllers for these kind of systems. Nevertheless, a potentially fruitful direction for future work would be to evolve controllers for executable models of real world systems. Executable models (such as agent-based systems) have recently become the focus of modelling communities in diverse fields, including biology [110], medicine [111], economics [112] and sociology [113]. It would be feasible to evolve controllers for these kind of models, and it seems plausible that the resulting controllers would give significant insight into the behaviour and control of their real world analogues.

VIII. Conclusions

In this paper, we have shown that artificial biochemical networks (ABNs), a class of computational dynamical systems, can be evolved to carry out control tasks in various kinds of complex dynamical systems. We have considered three ABN architectures: an artificial genetic network (AGN), an artificial metabolic network (AMN) and a coupled network (CABN) in which an AGN controls the expression of an AMN. We have also looked at the role that factors such as function choice, mass conservation, external signal delivery, and coupling functions play in the evolution of ABNs.

In general, we have found that different types of ABN are appropriate for the control of different types of dynamical system. AGNs often, but not always, perform better than AMNs on the control tasks we have looked at. In both cases, good performance depends upon suitable means of delivering external signals. For AMNs, mass conservation plays an important role, reducing the impact of sub-optimal parameter choices. CABNs outperform uncoupled ABNs on certain tasks, but offer no added benefit on others. We have speculated that coupling enables rapid switching of behaviours in response to control signals, and analysis of evolved controllers appears to support this notion.

Our most significant observation is the important role that non-linear discrete maps play when used as functional elements within ABNs. Discrete map-based ABNs led to the best controllers in the two hardest control tasks. Discrete maps are complex dynamical systems in their own right, and we have hypothesised that they provide a ready source of diverse, configurable dynamics. We also observed that chaotic dynamics play a significant role, particularly in the legged robot control tasks, and have suggested that this is due to the inherent controllability of chaotic dynamics.

More generally, we have shown that computational dynamical systems can be evolved to control systems which are representative of the kind of dynamics found in real world systems. In future work, we hope to extend this approach to controlling executable models of real world systems, and show how evolutionary computational dynamical systems may be used as a general approach to exploring and understanding the dynamics of complex systems.

Acknowledgment

The authors would like to acknowledge the support of the White Rose Grid in providing computational resources.

References

[1] J. H. Holland, *Adaptation in Natural and Artificial Systems*. MIT Press, 1975.

[2] R. Krohling and J. Rey, "Design of optimal disturbance rejection PID controllers using genetic algorithms," *Evolutionary Computation, IEEE Transactions on*, vol. 5, no. 1, pp. 78–82, February 2001.

[3] J. M. Dolsma, "Nonlinear controller design based on genetic programming," Master's thesis, Technische Universiteit Eindhoven, 2007.

[4] J. Hurst, L. Bull, and C. Melhuish, "TCS learning classifier system controller on a real robot," in *Parallel Problem Solving from Nature PPSN VII*, ser. Lecture Notes in Computer Science, J. Guervós, P. Adamidis, H.-G. Beyer, H.-P. Schwefel, and J.-L. Fernández-Villacaṇas, Eds. Springer Berlin / Heidelberg, 2002, vol. 2439, pp. 588–597.

[5] J. E. Auerbach and J. C. Bongard, "Evolving complete robots with CPPN-NEAT: the utility of recurrent connections," in *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, ser. GECCO '11. New York, NY, USA: ACM, 2011, pp. 1475–1482.

[6] M. Mokhtar, D. Halliday, and A. Tyrrell, "Hippocampus neurons and place cells/place field representation to provide path navigation," in *Neural Networks, 2007. IJCNN 2007. International Joint Conference on*, aug. 2007, pp. 795 –800.

[7] A. J. Ijspeert, "Central pattern generators for locomotion control in animals and robots: A review," *Neural Networks*, vol. 21, no. 4, pp. 642 – 653, 2008.

[8] Gh. Păun, "Computing with membranes," *Journal of Computer and System Sciences*, vol. 61, no. 1, pp. 108–143, 2000.

[9] P. Dittrich, J. Ziegler, and W. Banzhaf, "Artificial chemistries—a review," *Artificial Life*, vol. 7, pp. 225–275, 2001.

[10] W. Banzhaf, "Artificial chemistries—towards constructive dynamical systems," *Solid State Phenomena*, vol. 97/98, pp. 43–50, 2004.

[11] W. Fontana, "Algorithmic chemistry," in *Artificial Life II*, C. G. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen, Eds. Addison-Wesley, 1992, pp. 159–210.

[12] T. Szuba and R. Stras, "Parallel evolutionary computing with the random PROLOG processor," *Journal on Parallel and Distributed Computing*, vol. 47, pp. 78–85, 1997.

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

[14] T. Reil, "Dynamics of gene expression in an artificial genome—implications for biological and artificial ontogeny," in *Proceedings of the 5th European Conference on Artificial Life (ECAL'99)*, ser. Lecture Notes in Artificial Intelligence, no. 1674. Springer-Verlag, 1999, pp. 457–466.

[15] W. Banzhaf, "Artificial regulatory networks and genetic programming," in *Genetic Programming Theory and Practice*, R. L. Riolo and B. Worzel, Eds. Kluwer, 2003, ch. 4, pp. 43–62.

[16] S. Kumar, "The evolution of genetic regulatory networks for single and multicellular development," in *GECCO 2004 Late Breaking Papers*, M. Keijzer, Ed., 2004.

[17] D. Bray, "Protein molecules as computational elements in living cells," *Nature*, vol. 376, pp. 307–312, 1995.

[18] J. Decraene, G. G. Mitchell, and B. McMullin, "Evolving artificial cell signaling networks: Perspectives and methods," in *Advances in Biologically Inspired Information Systems*, F. Dressler and I. Carreras, Eds. Springer, 2007, pp. 167–186.

[19] F. C. Richards, T. P. Meyer, and N. H. Packard, "Extracting cellular automaton rules directly from experimental data," *Physica D: Nonlinear Phenomena*, vol. 45, no. 1 - 3, pp. 189 – 202, 1990.

[20] R. Beer and J. Gallagher, "Evolving dynamical neural networks for adaptive behavior," *Adaptive Behavior*, vol. 1, no. 1, pp. 91–122, 1992.

[21] S. Stepney, "Nonclassical computation: a dynamical systems perspective," in *Handbook of Natural Computing*, G. Rozenberg, T. Bäck, and J. N. Kok, Eds. Springer, 2009, vol. 2, ch. 52.

[22] M. I. Jordan, "Artificial neural networks," J. Diederich, Ed. Piscataway, NJ, USA: IEEE Press, 1990, ch. Attractor dynamics and parallelism in a connectionist sequential machine, pp. 112–127.

[23] J. L. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, no. 2, pp. 179 – 211, 1990.

[24] S. Ulam, "Random processes and transformations," in *Proc. Int. Congress of Mathematicians*. American Mathematical Society, 1952, vol. 2, pp. 264 – 275.

[25] A. Adamatzky, B. D. L. Costello, and T. Asai, *Reaction-Diffusion Computers*. Elsevier, 2005.

[26] C. Andersson and M. Nordahl, "Evolving coupled map lattices for computation," in *Genetic Programming, Proc. 1st European Workshop on Genetic Programming*, ser. Lecture Notes in Computer Science, W. Banzhaf *et al.*, Eds. Springer, 1998, vol. 1391, pp. 151–162.

[27] J. Ziegler and W. Banzhaf, "Evolving control metabolisms for a robot," *Artificial Life*, vol. 7, pp. 171–190, 2001.

[28] T. Taylor, "A genetic regulatory network-inspired real-time controller for a group of underwater robots," in *Intelligent Autonomous Systems 8 (Proceedings of IAS8)*, F. Groen *et al.*, Eds. Amsterdam: IOS Press, 2004, pp. 403–412.

[29] K. Dale, "Evolving reaction-diffusion controllers for minimally cognitive animats," in *From Animals to Animats 9*, ser. Lecture Notes in Computer Science, S. Nolfi *et al.*, Eds. Springer, 2006, vol. 4095, pp. 498–509.

[30] L. Bull and R. Preen, "On dynamical genetic programming: Random Boolean networks in learning classifier systems," in *Genetic Programming*, ser. Lecture Notes in Computer Science, L. Vanneschi *et al.*, Eds. Springer Berlin / Heidelberg, 2009, vol. 5481, pp. 37–48.

[31] S. Tsuda, S. Artmann, and K.-P. Zauner, "The phi-bot: A robot controlled by a slime mould," in *Artificial Life Models in Hardware*. Springer London, 2009, pp. 213–232.

[32] M. Nicolau, M. Schoenauer, and W. Banzhaf, "Evolving genes to balance a pole," in *Genetic Programming*, ser. Lecture Notes in Computer Science, Esparcia-Alczar *et al.*, Eds. Springer Berlin / Heidelberg, 2010, vol. 6021, pp. 196–207.

[33] M. A. Trefzer, T. Kuyucu, J. F. Miller, and A. M. Tyrrell, "Image compression of natural images using artificial gene regulatory networks," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, Portland, Oregon, July 2010.

[34] W. Maass, T. Natschläger, and H. Markram, "Real-time computing without stable states: A new framework for neural computation based on perturbations," *Neural Computation*, vol. 14, no. 11, pp. 2531–2560, 2002.

[35] M. O. Magnasco, "Chemical kinetics is Turing universal," *Phys. Rev. Lett.*, vol. 78, pp. 1190–1193, February 1997.

[36] P. Rendell, "Turing universality in the game of life," in *Collision-Based Computing*, A. Adamatzky, Ed. Springer, 2002, pp. 513–539.

[37] M. Ionescu, G. Păun, and T. Yokomori, "Spiking neural P systems," *Fundamenta Informaticae*, vol. 71, no. 2, pp. 279–308, 2006.

[38] H. Kitano, "Biological robustness," *Nat Rev Genet*, vol. 5, no. 11, pp. 826–837, 2004.

[39] K. Kaneko, "Overview of coupled map lattices," *Chaos*, vol. 2, no. 3, pp. 279–282, 1992.

[40] F. H. Willeboordse, "Hints for universality in coupled map lattices," *Phys. Rev. E*, vol. 65, no. 2, Jan 2002.

[41] P. T. Saunders, "The organism as a dynamical system," in *SFI Studies in the Sciences of Complexity, Lecture Notes*, F. Varela and W. Stein, Eds. Reading: Addison Wesley, 1993, vol. 3, pp. 41 – 63.

[42] G. Tufte, "Phenotypic, developmental and computational resources: scaling in artificial development," in *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, ser. GECCO '08. New York, NY, USA: ACM, 2008, pp. 859–866.

[43] M. Lozano, D. Molina, and F. Herrera, "Editorial: Scalability of evolutionary algorithms and other metaheuristics for large-scale continuous optimization problems," *Soft Computing*, vol. 15, pp. 2085–2087, 2011.

[44] P. Angeline, G. Saunders, and J. Pollack, "An evolutionary algorithm that constructs recurrent neural networks," *Neural Networks, IEEE Transactions on*, vol. 5, no. 1, pp. 54 –65, jan 1994.

[45] B. Talko, L. Stern, and L. Kitchen, "Evolving neural networks for decomposable problems using genetic programming," in *PRICAI 2000 Topics in Artificial Intelligence*, ser. Lecture Notes in Computer Science, R. Mizoguchi and J. Slaney, Eds. Springer Berlin / Heidelberg, 2000, vol. 1886, pp. 446–456.

[46] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary Computation*, vol. 10, no. 2, pp. 99–127, 2002.

[47] J. Drchal, J. Koutnik, and M. Snorek, "HyperNEAT controlled robots learn how to drive on roads in simulated environment," in *Evolutionary Computation, 2009. CEC '09. IEEE Congress on*, may 2009, pp. 1087 –1092.

[48] M. Mitchell, J. P. Crutcheld, and R. Das, "Evolving cellular automata with genetic algorithms: A review of recent work," in *Proc. 1st Int. Conf. on Evolutionary Computation and Its Applications (EvCA'96)*. Russian Academy of Sciences, 1996.

[49] A. Devert, N. Bredeche, and M. Schoenauer, "Unsupervised learning of echo state networks: A case study in artificial embryogeny," in *Artificial Evolution*, ser. Lecture Notes in Computer Science, N. Monmarché *et al.*, Eds. Springer, 2008, vol. 4926, pp. 278–290.

[50] M. Conrad, "The geometry of evolution," *BioSystems*, vol. 24, pp. 61–81, 1990.

[51] M. Kirschner and J. Gerhart, "Evolvability," *Proceedings of the National Academy of Science (USA)*, vol. 95, pp. 8420–8427, July 1998.

[52] A. M. Poole, M. J. Philips, and D. Penny, "Prokaryote and eukaryote evolvability," *BioSystems*, vol. 69, pp. 163–186, 2003.

[53] M. A. Lones, "Enzyme genetic programming: Modelling biological evolvability in genetic programming," Ph.D. dissertation, Department of Electronics, University of York, 2003.

[54] T. Hu and W. Banzhaf, "Evolvability and speed of evolutionary algorithms in light of recent developments in biology," *Journal of Articial Evolution and Applications*, no. 568375, 2010.

[55] P. C. Marijuán, "Enzymes, artificial cells and the nature of biological information," *BioSystems*, vol. 35, pp. 167–170, 1995.

[56] G. Ciobanu, M. J. Pérez-Jiménez, and Gh. Păun, *Applications of Membrane Computing*. Springer, 2006.

[57] A. Faulconbridge, S. Stepney, J. F. Miller, and L. S. D. Caves, "RBN-World: A sub-symbolic artificial chemistry," in *Advances in Artificial Life. Darwin Meets von Neumann, Part 1*, ser. Lecture Notes in Computer Science, G. Kampis, I. Karsai, and E. Szathmry, Eds. Springer, 2012, vol. 5777, pp. 377–384.

[58] W. Banzhaf and C. Lasarczyk, "Genetic programming of an algorithmic chemistry," in *Genetic Programming Theory and Practice II*, ser. Genetic Programming, J. Koza, U.-M. O'Reilly, T. Yu, R. Riolo, and B. Worzel, Eds. Springer US, 2005, vol. 8, pp. 175–190.

[59] C. W. Lasarczyk and W. Banzhaf, "An algorithmic chemistry for genetic programming," in *Genetic Programming*, ser. Lecture Notes in Computer Science, M. Keijzer *et al.*, Eds. Springer, 2005, vol. 3447, pp. 141–141.

[60] E. V. Koonin, Y. I. Wolf, G. P. Karev, P. Fernández, and R. V. Solé, "The role of computation in complex regulatory networks," in *Power Laws, Scale-Free Networks and Genome Biology*, ser. Molecular Biology Intelligence Unit. Springer US, 2006, pp. 206–225.

[61] E. Dubrova, M. Teslenko, and H. Tenhunen, "A computational scheme based on random Boolean networks," in *Transactions on Computational Systems Biology X*, ser. Lecture Notes in Computer Science, C. Priami *et al.*, Eds. Springer Berlin / Heidelberg, 2008, vol. 5410, pp. 41–58.

[62] R. Albert and H. G. Othmer, "The topology of the regulatory interactions predicts the expression pattern of the segment polarity genes in drosophila melanogaster," *Journal of Theoretical Biology*, vol. 223, no. 1, pp. 1–18, 2003.

[63] C. Gershenson, "Classification of random Boolean networks," in *Proceedings of the eighth international conference on Artificial life*. Cambridge, MA, USA: MIT Press, 2003, pp. 1–8. [Online]. Available: http://portal.acm.org/citation.cfm?id=860295.860297

[64] S. Zhan, J. F. Miller, and A. M. Tyrrell, "An evolutionary system using development and artificial genetic regulatory networks," in *Proc. 2008 IEEE Congress on Evolutionary Computation (CEC)*, J. Wang, Ed. IEEE Press, 2008.

[65] M. J. Fisher, R. C. Paton, and K. Matsuno, "Intracellular signalling proteins as 'smart' agents in parallel distributed processes," *BioSystems*, vol. 50, pp. 159–171, 1999.

[66] D. A. Lauffenburger, "Cell signaling pathways as control modules: Complexity for simplicity?" *PNAS*, vol. 97, no. 10, pp. 5031–5033, 2000.

[67] L. Bull, "A simple computational cell: coupling Boolean gene and protein networks," *Artificial Life*, vol. 18, no. 2, pp. 223 – 236, 2012.

[68] B. Wróbel, M. Joachimczak, A. Montebelli, and R. Lowe, "The search for beauty: Evolution of minimal cognition in an animat controlled by a gene regulatory network and powered by a metabolic system," in *From Animals to Animats 12*, ser. Lecture Notes in Computer Science, T. Ziemke, C. Balkenius, and J. Hallam, Eds. Springer Berlin / Heidelberg, 2012, vol. 7426, pp. 198–208.

[69] H. Chaté and J. Losson, "Non-trivial collective behavior in coupled map lattices: A transfer operator perspective," *Physica D: Nonlinear Phenomena*, vol. 103, no. 1-4, pp. 51 – 72, 1997.

[70] R. D. Kornberg, "The molecular basis of eukaryotic transcription," *Proceedings of the National Academy of Sciences*, vol. 104, no. 32, pp. 12 955–12 961, 2007.

[71] R. J. Conrado, J. D. Varner, and M. P. DeLisa, "Engineering the spatial organization of metabolic enzymes: mimicking nature's synergy," *Current Opinion in Biotechnology*, vol. 19, no. 5, pp. 492 – 499, 2008.

[72] R. M. May, "Simple mathematical models with very complicated dynamics," *Nature*, vol. 261, pp. 459–467, 1976.

[73] T. Tél and M. Gruiz, *Chaotic Dynamics: An Introduction Based on Classical Mechanics*. Cambridge Press, 2006.

[74] V. Arnold and A. Avez, *Ergodic problems in classical mechanics*. New York: Benjamin, 1968.

[75] B. V. Chirikov, "Research concerning the theory of nonlinear resonance and stochasticity," Institute of Nuclear Physics, Novosibirsk, Tech. Rep., 1969.

[76] E. E. Macau and C. Grebogi, "Control of chaos and its relevancy to spacecraft steering," *Philosophical Transactions of the Royal Society A: Mathematical,Physical and Engineering Sciences*, vol. 364, no. 1846, pp. 2463–2481, 2006.

[77] A. Garfinkel, J. N. Weiss, W. L. Ditto, and M. L. Spano, "Chaos control of cardiac arrhythmias." *Trends Cardiovasc Med*, vol. 5, no. 2, pp. 76–80, 1995.

[78] D. G. MacMynowski, "Controlling chaos in El Niño," in *American Control Conference (ACC)*, July 2010.

[79] A. L. Fradkov, R. J. Evans, and B. R. Andrievsky, "Control of chaos: methods and applications in mechanics," *Philosophical Transactions of the Royal Society A: Mathematical,Physical and Engineering Sciences*, vol. 364, no. 1846, pp. 2279–2307, 2006.

[80] E. Ott, C. Grebogi, and J. A. Yorke, "Controlling chaos," *Phys. Rev. Lett.*, vol. 64, no. 11, pp. 1196–1199, Mar 1990.

[81] A. S. de Paula and M. A. Savi, "Comparative analysis of chaos control methods: A mechanical system case study," *International Journal of Non-Linear Mechanics*, vol. 46, no. 8, pp. 1076 – 1089, 2011.

[82] K. Pyragas, "Delayed feedback control of chaos," *Philosophical Transactions of the Royal Society A: Mathematical,Physical and Engineering Sciences*, vol. 364, no. 1846, pp. 2309–2334, 2006.

[83] H. Richter, "An evolutionary algorithm for controlling chaos: The use of multi-objective fitness functions," in *Proceedings of the 7th International Conference on Parallel Problem Solving from Nature*, ser. PPSN VII. London, UK, UK: Springer-Verlag, 2002, pp. 308–320.

[84] C. Y. Soong, W. T. Huang, F. P. Lin, and P. Y. Tzeng, "Controlling chaos with weak periodic signals optimized by a genetic algorithm," *Phys. Rev. E*, vol. 70, p. 016211, 2004.

[85] I. Zelinka, "Real-time deterministic chaos control by means of selected evolutionary techniques," *Engineering Applications of Artificial Intelligence*, vol. 22, pp. 283 – 297, 2009.

[86] E. N. Sanchez and L. J. Ricalde, "Chaos control and synchronization, with input saturation, via recurrent neural networks," *Neural Networks*, vol. 16, pp. 711 – 717, 2003.

[87] X. Xu, Y. Liang, H. Lee, W. Lin, S. Lim, and X. Shi, "A stable adaptive neural-network-based scheme for dynamical system control," *Journal of Sound and Vibration*, vol. 285, no. 3, pp. 653 – 667, 2005.

[88] E. N. Lorenz, "Deterministic nonperiodic flow," *Journal of the Atmospheric Sciences*, vol. 20, no. 2, pp. 130–141, March 1963.

[89] E. M. Bollt and J. D. Meiss, "Controlling chaotic transport through recurrence," *Physica D: Nonlinear Phenomena*, vol. 81, no. 3, pp. 280–294, 1995.

[90] C. G. Schroer and E. Ott, "Targeting in Hamiltonian systems that have mixed regular/chaotic phase spaces," *Chaos*, vol. 7, pp. 512–519, December 1997.

[91] M. S. Branicky, "Introduction to hybrid systems," in *Handbook of Networked and Embedded Control Systems*, D. Hristu-Varsakelis and W. Levine, Eds. Birkhauser, 2005.

[92] K. Seo and S. Hyun, "Genetic programming based automatic gait generation for quadruped robots," in *Proc. 2008 Genetic and Evolutionary Computation Conference (GECCO'08)*, M. Keijzer *et al.*, Eds. Atlanta, GA, USA: ACM, 12-16 Jul. 2008, pp. 293–294.

[93] J. Clune, B. E. Beckmann, C. Ofria, and R. T. Pennock, "Evolving coordinated quadruped gaits with the HyperNEAT generative encoding," in *Proc. 2009 Congress on Evolutionary Computation (CEC 2009)*, A. Tyrrell *et al.*, Eds. IEEE, 2009.

[94] I. Macinnes and E. D. Paolo, "Crawling out of the simulation: evolving real robot morphologies using cheap, reusable modules," in *Proceedings of the International Conference on Artificial Life (ALIFE9)*. MIT Press, 2004, pp. 94–99.

[95] C. Mattiussi and D. Floreano, "Analog genetic encoding for the evolution of circuits and networks," *Evolutionary Computation, IEEE Transactions on*, vol. 11, no. 5, pp. 596 –607, oct. 2007.

[96] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *Evolutionary Computation, IEEE Transactions on*, vol. 6, no. 2, pp. 182 –197, apr 2002.

[97] M. A. Lones, S. L. Smith, A. M. Tyrrell, J. E. Alty, and D. R. S. Jamieson, "Characterising neurological time series data using biologically-motivated networks of coupled discrete maps," *BioSystems*, to appear.

[98] H. Kantz and T. Schreiber, *Nonlinear Time Series Analysis*, 2nd ed. Cambridge University Press, 2004.

[99] L. S. D. Caves and C. S. Verma, "Congruent qualitative behavior of complete and reconstructed phase space trajectories from biomolecular dynamics simulation." *Proteins*, vol. 47, no. 1, pp. 25–30, Apr 2002.

[100] S. Steingrube, M. Timme, F. Wörgötter, and P. Manoonpong, "Self-organized adaptation of a simple neural circuit enables complex robot behaviour," *Nature Physics*, vol. 6, pp. 224–230, 2010.

[101] M. A. Lones, A. M. Tyrrell, S. Stepney, and L. S. Caves, "Controlling complex dynamics with artificial biochemical networks," in *Proc. 2010 European Conference on Genetic Programming (EuroGP 2010)*, ser. Lecture Notes in Computer Science, A. I. Esparcia-Alczar *et al.*, Eds., vol. 6021. Springer Berlin / Heidelberg, 2010, pp. 159–170.

[102] M. A. Lones, S. L. Smith, A. M. Tyrrell, J. E. Alty, and D. R. S. Jamieson, "Evolving computational dynamical systems to recognise abnormal human motor function," in *Information Processing in Cells and Tissues, Proc. 9th Int. Conf.*, ser. Lecture Notes in Computer Science, M. A. Lones *et al.*, Eds., vol. 7223. Springer, March 2012, pp. 177–182.

[103] M. A. Lones, A. M. Tyrrell, S. Stepney, and L. S. D. Caves, "Controlling legged robots with coupled artificial biochemical networks," in *Advances in Artificial Life, ECAL 2011: Proc. 11th European Conference on the Synthesis and Simulation of Living Systems*, T. Lenaerts *et al.*, Eds. MIT Press, August 2011, pp. 465–472.

[104] L. A. Fuente, M. A. Lones, A. P. Turner, A. M. Tyrrell, S. Stepney, and L. S. D. Caves, "Evolved artificial signalling networks for the control of a conservative complex dynamical system," in *Information Processing in Cells and Tissues, Proc. 9th Int. Conf.*, ser. Lecture Notes in Copmuter Science, M. A. Lones *et al.*, Eds., vol. 7223. Springer, March 2012, pp. 38–49.

[105] A. P. Turner, M. A. Lones, L. A. Fuente, A. M. Tyrrell, S. Stepney, and L. S. D. Caves, "Using artificial epigenetic regulatory networks to control complex tasks within chaotic systems," in *Information Processing in Cells and Tissues, Proc. 9th Int. Conf.*, ser. Lecture Notes in Computer Science, M. A. Lones *et al.*, Eds., vol. 7223. springer, March 2012, pp. 1–11.

[106] J.-S. Wang and Y.-P. Chen, "A fully automated recurrent neural network for unknown dynamic system identification and control," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 53, no. 6, pp. 1363 –1372, june 2006.

[107] M. Newman, *Networks: An Introduction*. Oxford University Press, 2010.

[108] J. P. Crutchfield, "The calculi of emergence: computation, dynamics and induction," *Physica D: Nonlinear Phenomena*, vol. 75, no. 1-3, pp. 11 – 54, 1994.

[109] H. Jacobsson, "Rule extraction from recurrent neural networks: A taxonomy and review," *Neural Computation*, vol. 17, pp. 1223–1263, 2005.

[110] J. Fisher and T. A. Henzinger, "Executable cell biology," *Nature Biotechnology*, vol. 25, pp. 1087–0156, 2007.

[111] M. Read, "Statistical and modelling techniques to build confidence in the investigation of immunology through agent-based simulation," Ph.D. dissertation, University of York, 2011.

[112] L. Tesfatsion and K. L. Judd, Eds., *Handbook of Computational Economics: Agent-Based Computational Economics*. North Holland, 2006.

[113] M. W. Macy and R. Willer, "From factors to actors: Computational sociology and agent-based modeling," *Annual Review of Sociology*, pp. 143–166, 2002.