Evolving Symmetric and Modular Neural Networks for Distributed Control

Vinod K. Valsalam
Department of Computer Sciences
The University of Texas at Austin
Austin, TX 78712 USA
vkv@cs.utexas.edu

Risto Miikkulainen
Department of Computer Sciences
The University of Texas at Austin
Austin, TX 78712 USA
risto@cs.utexas.edu

ABSTRACT

Problems such as the design of distributed controllers are characterized by modularity and symmetry. However, the symmetries useful for solving them are often difficult to determine analytically. This paper presents a nature-inspired approach called Evolution of Network Symmetry and mOdularity (ENSO) to solve such problems. It abstracts properties of generative and developmental systems, and utilizes group theory to represent symmetry and search for it systematically, making it more evolvable than randomly mutating symmetry. This approach is evaluated by evolving controllers for a quadruped robot in physically realistic simulations. On flat ground, the resulting controllers are as effective as those having hand-designed symmetries. However, they are significantly faster when evolved on inclined ground, where the appropriate symmetries are difficult to determine manually. The group-theoretic symmetry mutations of ENSO were also significantly more effective at evolving such controllers than random symmetry mutations. Thus, ENSO is a promising approach for evolving modular and symmetric solutions to distributed control problems, as well as multiagent systems in general.

Categories and Subject Descriptors: I.2.6–*Connectionism and Neural Nets*; I.2.9–*Propelling Mechanisms*

General Terms: Design, experimentation, performance

Keywords: Indirect encoding, modularity, symmetry, group theory, multilegged robots, controllers

1. INTRODUCTION

Generative and developmental systems (GDS) can represent complex structures with regularities compactly [22]. They are useful for evolving solutions to large-scale problems for two reasons. First, their compact genotype representation makes the evolutionary search space smaller, and therefore easier to search. Second, known regularities in the phenotype can sometimes be encoded into the genotype representation, evolutionary operators, and genotype-phenotype mapping, making it possible for evolution to find solutions efficiently.

A common type of regularity that appears in the solution to many complex problems is the repetition of interconnected substructures called *modules*. For example, the controller for a legged robot can be decomposed into modules, each module controlling a leg [2, 24]. Regularity constrains some of the modules and interconnections to be identical, resulting in *symmetries*, i.e. permutations of the modules that leave the solution invariant. Symmetries express constraints crucial for solving many problems; for instance they determine the type of gaits that a legged robot controller can produce [4, 24]. Current GDS approaches can sometimes result in symmetric solutions [7, 10, 19–21]. In contrast, the approach developed in this paper evolves symmetry systematically using mathematical abstractions, making it potentially more effective at finding the appropriate symmetries.

This novel approach, called Evolution of Network Symmetry and mOdularity (ENSO), utilizes group theory to represent symmetry mathematically. It uses a compact genotype that stores the parameters of repeated modules only once, while allowing variations between them to evolve. Moreover, it encodes symmetry in a form that evolution can manipulate easily, and maps genotype to phenotype explicitly, making it possible to evolve solutions systematically by breaking symmetry incrementally. The resulting symmetry bias avoids large changes in symmetry (that are otherwise likely with unsystematic mutations), thus enhancing the evolvability of the representation.

The ENSO representation utilizes domain knowledge by initializing evolution with user-identified modules, allowing ENSO to solve complex problems such as the design of distributed controllers and multiagent systems. In this paper, this capability is demonstrated by using ENSO to evolve neural network controllers for a quadruped robot in a physically realistic simulation. ENSO evolves controllers that produce effective locomotive behaviors both on flat and inclined ground. Moreover, on inclined ground these controllers are significantly better than those evolved with hand-designed symmetries, and with random symmetry mutations. Since inclines and other similar complexities are common in the real world, these results suggest that ENSO is a promising approach for designing distributed controllers in the real world.

This paper is organized as follows. The next section presents the biological and mathematical background on ENSO, and reviews related work. Section 3 describes the ENSO approach, its genotype and phenotype representations, and its evolutionary operators. Section 4 discusses the quadruped robot model and experiments designed to evaluate ENSO. Section 5 presents the results of these experiments, demonstrating the benefits of the ENSO approach. Section 6 discusses the results and presents directions for future work.

Visualization videos of the walking behaviors discussed in this paper can be seen at the website `http://nn.cs.utexas.edu/?enso-robots`.

2. BACKGROUND AND RELATED WORK

This section begins by discussing the biological motivation for ENSO, and then reviews prior work on other GDS approaches. Finally, it introduces the group-theoretical concepts utilized by ENSO.

2.1 Biological Motivation

In nature, mutations that break symmetries produce novel phenotypic variations [18]. For example, fiddler crabs, whose males are asymmetric with an oversized claw, evolved from a bilaterally symmetric ancestor. Bilateral symmetry is the default in such organisms, i.e. the same developmental program creates paired, symmetrical sides. Breaking this symmetry requires the genome to specify additional information on how one side is different from the other. Thus symmetry breaking increases the complexity of the genome.

This relation between complexity and symmetry breaking is fundamental, allowing complexity to be characterized as the lack of symmetry [9]. In fact, *complexification*, i.e. the evolution of complexity in organisms, is accompanied by symmetry breaking at different levels of organization [6]. Moreover, complexification makes evolutionary search more effective [23]. It allows evolution to start with low-dimensional genotypes, which are easy to optimize, and gradually add more dimensions while optimizing them further. Building complex systems by incrementally elaborating solutions in this manner is much more likely to succeed than evolving solutions in the final high-dimensional space directly.

The ENSO approach encapsulates these properties of natural evolution and developmental systems using mathematical abstractions as described in Section 3. Other researchers have devised similar evolutionary algorithms based on computational abstractions of development called *indirect encodings*, and they are discussed next.

2.2 Indirect Encodings

Most indirect encodings were developed for evolving artificial neural networks, and Kitano was one of its earliest practitioners [11]. He evolved grammatical matrix rewrite rules that produced the adjacency matrix of neural networks through a series of rewrite steps. Sims used another generative encoding based on graphs to evolve virtual creatures and their neural network controllers in simulated physical environments [19]. Such grammar-based schemes tend to produce repetitive fractal-like regularities, while ENSO is designed to evolve regularities with clearly defined modules.

Gruau's *cellular encoding* (CE) [7] utilizes another abstraction of development inspired by how growth occurs through cell division. In this method, the genotype encodes a program tree for constructing a neural network from a single ancestral cell. Although CE can express any network theoretically, Luke and Specter identified biases of the method that may be problematic for evolution in many domains [12]. Utilizing similar ideas of development, Miller evolved developmental programs that can construct the French flag (i.e. adjoining rectangular regions of blue, white, and red colors) and repair damages in it [13].

In a domain similar to Sims' virtual creatures, Hornby and Pollack combined principles of developmental and grammatical systems to evolve the body and brain of creatures simultaneously [10]. However, since their method is designed to evolve body morphologies and controllers together, it is not well suited for developing controllers for fixed morphology such as in the legged locomotion application explored in this paper. Bongard and Pfeifer also evolved similar virtual creatures, but they used an abstraction of *genetic regulatory networks* (GRNs) for encoding bodies and neural networks [3].

Abstracting the above approaches, Stanley proposed an indirect encoding called *Compositional Pattern Producing Networks* (CPPNs) that eliminates the traditional local interaction and temporal unfolding mechanisms of developmental systems [20]. Instead, he argued that the effects of such mechanisms can be obtained by composing specific functions in the appropriate order. The patterns produced by a CPPN are interpreted as the spatial connectivity patterns of a neural network using a method called *HyperNEAT*. Stanley et al. applied this method to tasks having a large number of inputs and regularities, such as robot food gathering and visual object discrimination [21].

All the above methods provide mechanisms for reuse of genes and repetition of phenotypic substructures, thus encouraging modularity. The developmental process also sometimes produces symmetries in the modular phenotypes, and they can emerge if symmetric and periodic functions are used in the encoding. However, they do not utilize a mathematical theory of symmetry to represent symmetries explicitly, and therefore do not provide effective mechanisms to search for appropriate symmetries systematically. Since symmetry and complexity are closely related (Section 2.1), such mechanisms can be useful for solving complex problems with symmetries efficiently. The ENSO approach presented in this paper addresses this issue by utilizing group theory to represent and evolve the symmetry of modular systems. The group theory concepts it utilizes are introduced in the next subsection.

2.3 Graph Symmetries and Groups

ENSO represents modules as the vertices and the relationships between them as the edges of a complete graph $G = (V, E)$, where V is the vertex set and $E = \{(u, v) \mid u, v \in V;\ u \neq v\}$ is the edge set. Since G cannot have loops, it is possible to represent a vertex v by the pair (v, v), and thus represent both vertices and edges by the elements of the set $V \times V$. In order to represent the symmetries of G, ENSO assigns every element of $V \times V$ a color, producing a *complete coloring* [1] of the vertices and edges of G. In practice, each color represents a particular combination of parameters associated with a vertex or edge. A *symmetry* of G is defined as any permutation of its vertices that preserves the color of edges between them.

Figure 1 illustrates how a group can represent the symmetries of such a completely colored graph. Since all edges of graph G_A have the same color, any permutation of its vertices is a symmetry of the graph. In contrast, graph G_B has fewer symmetries because its edges have different colors. The permutation $g = (1\ 2)(3\ 4)$, which swaps vertices 1 and 2 as well as vertices 3 and 4, is a symmetry of G_B. Similarly, the permutation $h = (1\ 3)(2\ 4)$ is another symmetry, and their composition hg obtained by performing the two permutations in sequence is yet another symmetry. The trivial permutation $e = ()$, which fixes each vertex of the graph, is also a symmetry. The set of all such symmetries of a graph G is closed under composition and inverses, i.e. it forms a *group* with composition as the group operation. This group is called the *automorphism group* of G, denoted as $\mathrm{Aut}(G)$.

The automorphism group of graph G_A, consisting of all 4! permutations of its vertex set $V = \{1, 2, 3, 4\}$, is called the *symmetric group* of degree four, denoted as $\mathcal{S}_4$. The automorphism group of the less symmetric graph G_B is a subgroup of $\mathcal{S}_4$ called the *dihedral group* $\mathcal{D}_2$ (and is isomorphic to the symmetries of a regular polygon with two sides, i.e. a line segment). More generally, the automorphism group of any graph G with vertex set $V = \{1, 2, 3, 4\}$ is a subgroup of $\mathcal{S}_4$, and is fully determined by the complete coloring of G. ENSO utilizes this observation to manipulate the symmetries of graphs by changing their coloring.

Changing the color of a graph G to produce another graph G' such that $\mathrm{Aut}(G')$ is a subgroup of $\mathrm{Aut}(G)$ is said to *break the symmetry* of G. The graphs produced by symmetry breaking can be ordered based on the subgroup relation between their automorphism groups. More precisely, with subgroup as the partial order relation, the set of all subgroups of a group form a lattice. Figure 2 illustrates this lattice for the subgroups of $\mathcal{S}_4$. A group $\mathcal{G}_i$ is placed above another group $\mathcal{G}_j$ and connected by a line if and only if $\mathcal{G}_j$ is a maximal subgroup of $\mathcal{G}_i$. This lattice contains the automorphism groups of all completely colored graphs with vertex set $V = \{1, 2, 3, 4\}$. The most symmetric graphs with automorphism group $\mathcal{S}_4$ are at the top

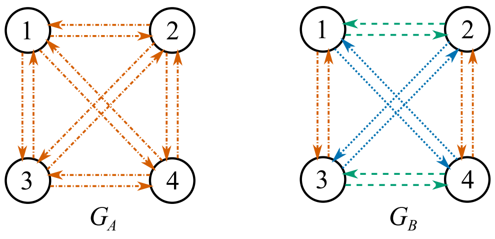


Figure 1: Representing graph symmetries using groups. Each vertex and edge has a color (indicated by both color and line style) representing a particular combination of parameters. A graph symmetry is any permutation of vertices under which the edge colors remain the same. Both graphs in this figure have vertices of the same color. All edges of graph G_A have the same color, while edges of graph G_B have different colors. Therefore, any permutation of the vertices of graph G_A is a symmetry. In contrast, only the permutations $g = (1\ 2)(3\ 4)$ and $h = (1\ 3)(2\ 4)$, and their compositions are symmetries of graph G_B. The set of all symmetries of a graph form a group, with composition as the group operation. Thus group theory is a natural way to represent symmetries.

of the lattice, while the least symmetric graphs with the trivial automorphism group $\{e\}$ are at the bottom.

Therefore, traversing the subgroup lattice from top to bottom visits progressively less symmetric graphs, making it possible for ENSO to search the space of graph symmetries systematically by breaking symmetry incrementally. However, the number of subgroups of $\mathcal{S}_{|V|}$ grows combinatorially as $|V|$ increases [8], making exhaustive symmetry search intractable for graphs with a large number of vertices. Moreover, ENSO must search the parameter space of the modular neural network corresponding to each symmetry. Therefore, ENSO uses evolution to focus the search on promising lattice regions and parameter combinations, as described next.

3. APPROACH

ENSO evolves neural network solutions for problems where domain regularities make it possible to partition the inputs and outputs of the network into modules. It represents the modules and the connections between them as the vertices and edges of a completely colored graph (Section 2.3). Evolving this representation consists of two components: (1) evolving the symmetry of the module interconnection graph and (2) evolving the functionality of the modules and connections. These components are described below.

3.1 Symmetry Evolution

In order to evolve a network with n modules, ENSO initializes a population of maximally symmetric, completely colored graphs with vertex set $V = \{1, 2, \ldots, n\}$, i.e. the automorphism group of these graphs is $\mathcal{S}_n$. These graphs have only two colors: All vertices are of one color while all edges are of the other color. Vertices or edges with the same color have the same set of neural network parameters and are therefore considered identical. Therefore, each graph in the initial population represents a modular neural network having identical modules and identical connections between the modules.

ENSO computes the subgroup lattice of $\mathcal{S}_n$ and the complete coloring corresponding to each subgroup in the lattice at the beginning of evolution using the GAP [5] software. During evolution, ENSO utilizes this lattice to mutate the coloring of graphs, thus breaking their symmetry. Each such color mutation randomly chooses a successor in the subgroup lattice; that is, the automorphism group of the mutated graph is a random maximal subgroup of the automorphism group of the original graph.

ENSO organizes the colors created by successive color mutations as a tree. Each tree is a genotype for evolution. The leaf colors of the tree specify the complete coloring of a graph, which forms the phenotype that is constructed from the genotype. Each genotype

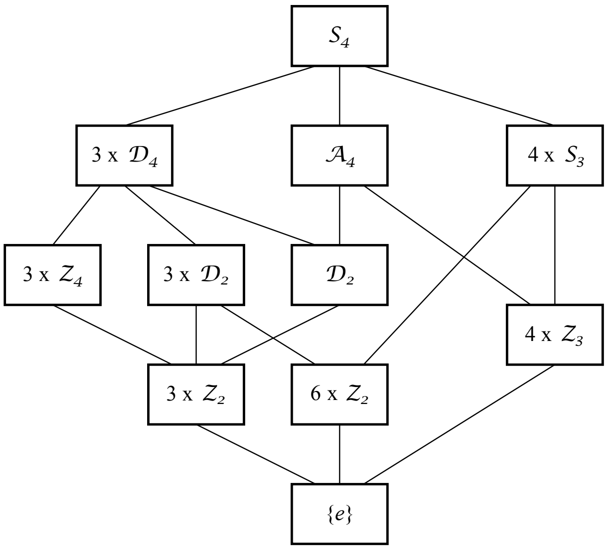


Figure 2: Lattice of subgroups of $\mathcal{S}_4$. This lattice was computed using the GAP [5] software for computational group theory and shows the subgroups of the group $\mathcal{S}_4$, which is the symmetric group of degree four containing all 4! permutations of the set $V = \{1, 2, 3, 4\}$. Each node of the lattice represents an equivalence class of conjugate subgroups. For example, the node labeled $4 \times \mathcal{S}_3$ represents the four symmetric groups of degree three obtained by fixing each of the four elements of V and permuting only the other three elements. The lattice also contains the alternating group $\mathcal{A}_4$ of degree four, formed by the permutations of V that can be expressed as the composition of an even number of transpositions; the dihedral groups $\mathcal{D}_n$, formed by the permutations of V that are isomorphic to the symmetries of a regular polygon with n sides; and the cyclic groups $\mathcal{Z}_n$, formed by the permutations of V that are isomorphic to the group of integers under addition modulo n. The automorphism groups of all graphs with vertex set V appear in this lattice, inducing a partial order of the corresponding graphs. Thus, the most symmetric graphs with automorphism group $\mathcal{S}_4$ appear at the top of the lattice, while the least symmetric graphs with the trivial automorphism group $\{e\}$ appear at the bottom of the lattice. This order makes it possible for ENSO to search the space of graph symmetries systematically by traversing the lattice from top to bottom.

tree of the initial population has two leaf nodes, one representing the color of vertices and the other representing the color of edges (Figure 3a). These two leaf nodes are the children of a root node that represents a dummy color.

A node representing a particular color c also represents the set Q of elements of $V \times V$ (i.e. the set of vertices or edges of the phenotype graph) that have the color c. This representation is a bit string of length n^2, where the bit position $(i-1)n + j$ is set to "1" if and only if the pair (i, j) is in Q. Third, the node stores the neural network parameters of these elements, i.e. the biases and connection weights of the module network (for each vertex) or the connection weights between modules (for each edge).

The effect of a color mutation on the genotype tree is to partition the set of vertex or edge elements associated with one or more leaf nodes and create a new child color for each part of the partition (Figure 3b). As a result, the colors of these elements change correspondingly in the phenotype graph, i.e. some of the elements that were identical before the mutation are no longer identical: A new (initially random) set of neural network parameters is associated with each new color.

Since the automorphism group of the mutated graph is a maximal subgroup of the automorphism group of the original graph, color mutations break symmetry in minimal increments. As a result, evolution searches the space of symmetries systematically by exploring more symmetric graphs before less symmetric ones. Creating new colors and parameters in the genotype tree during this process in-

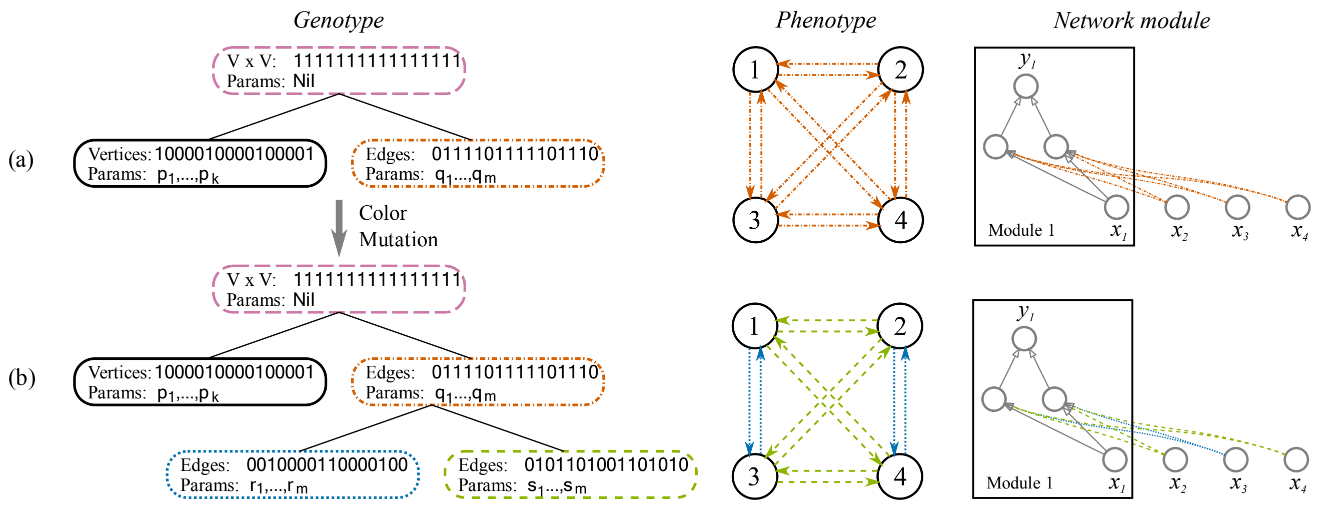


Figure 3: Examples of genotype, phenotype, network module, and color mutation. ENSO uses a tree of colors as genotype (left). Each leaf of this tree has a unique color, and represents a set of vertices or edges of the phenotype graph (middle) that have the same parameter values. The vertices and edges of the phenotype graph represent the modules of a neural network and the connections between them (right). Their parameters (stored in the genotype) consist of node biases and connection weights for each module network (vertex) and weights for each connection between modules (edge). Each module has a fixed architecture with a layer of hidden nodes fully connected to its inputs and outputs. A connection from another module (not shown) is implemented by fully connecting its input layer to the hidden layer of the target module. (a) At the beginning of evolution, each genotype in the population represents a maximally symmetric phenotype graph with automorphism group $\mathcal{S}_4$. All vertices of this graph have the same color (solid black, represented by the leaf on the left) and all its edges have the same color (orange with alternating dots and dashes, represented by the leaf on the right), implying that all modules are identical and all connections between them are also identical. (b) A color mutation breaks the phenotype graph symmetry to $\mathcal{D}_4$, which is a maximal subgroup of $\mathcal{S}_4$ (Figure 2). As a result, two child nodes are created for the node representing the set of edges, i.e. the set of edges is partitioned into two and each part is colored differently (dotted blue and dashed green). Since each color is associated with a different combination of parameter values, the mutated phenotype graph represents two types of connections between network modules. Such color mutations, when combined with parameter mutations, make it possible to evolve symmetric and modular neural networks efficiently.

creases the complexity of the genotype, i.e. evolution searches in a low-dimensional space before it complexifies to a higher dimensional space. This approach allows evolution to optimize solutions in a small search space, and elaborate on them by adding more dimensions. Such complexification has been demonstrated to be useful in other methods for evolving neural networks [23].

For each phenotype graph that ENSO produces in the above manner, it evolves the neural network parameters associated with its colors to optimize the functionality of the network modules and their interconnections. The structure of these modules and the evolution of network parameters is described next.

3.2 Module Evolution

A fixed architecture is used for the neural network modules, each module consisting of a layer of hidden nodes that are fully connected to inputs and outputs (Figure 3). Evolution optimizes two kinds of network parameters: (1) the scalar parameters of these modules, i.e. the connection weights and node biases, which form the vertex parameters of the phenotype graph, and (2) the weights of connections between modules, which form the edge parameters of the phenotype graph. Module interconnections are implemented by fully connecting the input layer of one module to the hidden layer of the other module. If modules are not connected, the corresponding graph edges are disabled using special binary edge parameters.

The vertex and edge parameters of the phenotype graph are stored in the genotype leaf nodes. In the initial population, these parameters are initialized with random values in parameter-specific ranges specified by the experimenter. During evolution, ENSO mutates each of these parameters probabilistically by perturbing them with Gaussian noise. When a parameter in a particular genotype node is mutated, it affects all vertices and edges with that color. Thus, representing identical elements by a single node in the genotype tree allows evolution to search the parameter space efficiently by making coordinated changes to the phenotype.

Symmetry and modules must be evolved together to find solution networks, making it necessary to mix parameter and color mutations. However, color mutations produce severe changes in the phenotype, resulting in sudden changes in fitness that may cause the phenotype to be removed from the population. It is possible to determine how effective such structural changes are only after enough parameter mutations have accumulated over evolutionary time. Therefore, color mutations are given the opportunity to optimize by creating population niches, similar in spirit to speciation in the NEAT algorithm [23]. Individuals occupying a niche have the same phenotype symmetry and remain in the niche for a certain number of generations before they compete with the rest of the population. This number is a linear function of the size of the genotype, allowing individuals with more parameters to stay in their niches longer. Protecting symmetry mutations using this niching mechanism improves evolutionary performance significantly.

Evolving the symmetry of module interconnections while optimizing module functionality in the above manner allows ENSO to find solutions to modular problems effectively, as demonstrated next by evolving controllers for legged robots.

4. EVOLVING ROBOT CONTROLLERS

Legged robots are modular, making them a useful real-world application for ENSO. The quadruped robot model, its modular neural network controller, and the experimental methods for evaluating the controllers are described in the following subsections.

4.1 Robot Model

The robot model resembles a table with a rectangular body supported by legs at the four corners (Figure 4). The legs are cylindrical with capped ends, and attached to the body by a hinge joint having full 360° freedom of rotation. The axis of rotation of the joint is tilted to the side, causing the rotating leg to trace a cone. The leg makes contact with the ground when it is at one edge of the cone.

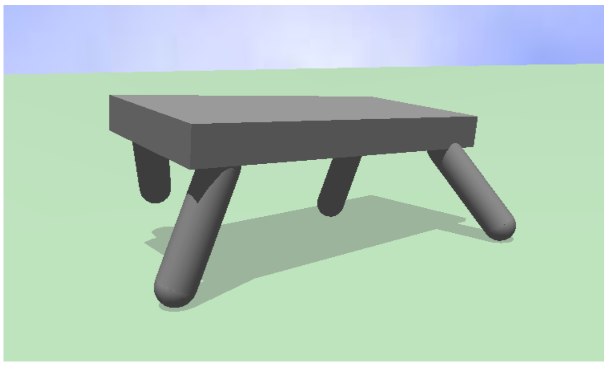


Figure 4: The quadruped robot model. The legs are attached to the body by hinge joints with axes of rotation tilted sideways, allowing the legs to make full circular rotation. Locomotion is achieved by coordinating the circular movements of the legs. This model is a simple but physically realistic platform that has symmetric and modular controllers suitable for evaluating ENSO.

Forward and backward locomotion is achieved by coordinating the circular movements of the leg. The robot controller activates the simulated servo motor attached to each joint by specifying the desired joint angular velocity.

4.2 Modular Controller

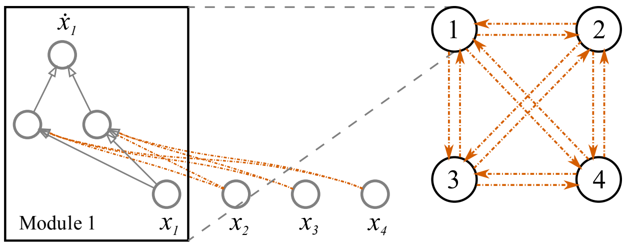
A neural network controller for this quadruped robot can be constructed using four modules, each controlling a different leg. All modules have the same network architecture shown in Figure 5a. Although a variety of architectures are possible, this simple two-layered architecture with two hidden nodes was found to be sufficient for evolving effective controllers. Each module's input is the joint angle of the leg it controls. It can be represented by the angle itself, or by the sine and cosine of the angle; the sine and cosine are actually more robust (because they are continuous), and will be used in the experiments on inclined ground. The module's output is the desired angular velocity of that leg. The hidden and output units have sigmoidal activation functions with a bias and slope as parameters; The input units do not perform any computation. These parameters and the weights of the internal connections of the module are the mutable vertex parameters of the phenotype graph (Section 3.2).

The phenotype graph represents the full controller network. It is obtained by connecting the four modules to each other, such that each module receives input from all the other modules (Figure 5b). The weights of these connections are the edge parameters of the phenotype graph.

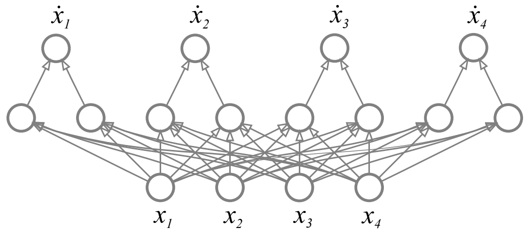
The modular nature of the controller allows it to be modeled as a coupled cell system (i.e. a set of interconnected dynamical systems) having the same graphical representation as the phenotype graph. Collins and Stewart [4] showed such systems with appropriate symmetries can produce synchronous and phase-related oscillations suitable for modelling the gaits of legged animals and robots. Such symmetries, determined through mathematical analysis (instead of evolved), were utilized by Valsalam and Miikkulainen [24] to fix the symmetries of modular controllers so that their module parameters could then be evolved. The resulting controllers produced regular gaits similar to those of quadruped animals. In contrast, ENSO makes it possible to evolve the symmetries together with the module parameters, producing effective controllers even when the appropriate symmetry is difficult to determine manually. The experimental methods used in demonstrating this result are described next.

4.3 Experimental Methods

In order to demonstrate the benefit of ENSO, four experimental methods for evolving the above modular controller were compared:



(a) Network module and initial phenotype graph



(b) Full controller network consisting of four modules

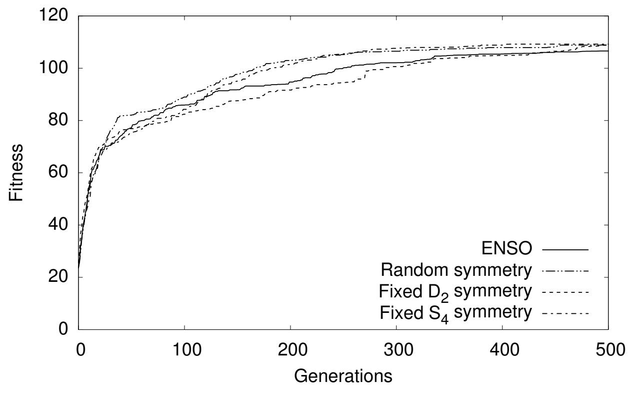
Figure 5: Modular controller network for the quadruped robot model. The input to each module is the leg angle (or its sine and cosine) that it controls, and the output is the desired angular velocity of that leg. The full controller network consists of four such modules, each module receiving input from all the other modules. The phenotype graph represents these modules and their connectivity. At the beginning of evolution, this graph has identical vertices (modules) and edges (interconnections). Evolution discovers effective controllers by breaking symmetry to create new types of vertices and edges, and by optimizing the initially random vertex and edge parameters.

(1) Evolving its symmetry systematically using ENSO, (2) evolving its symmetry randomly without using the group-theory mechanisms of ENSO, (3) using fixed $\mathcal{S}_4$ symmetry during evolution, and (4) using fixed $\mathcal{D}_2$ symmetry during evolution (as was done by Valsalam and Miikkulainen [24]). Although the four methods differ in how they evolve symmetry, they all evolve parameters of the controller in the same way as ENSO.

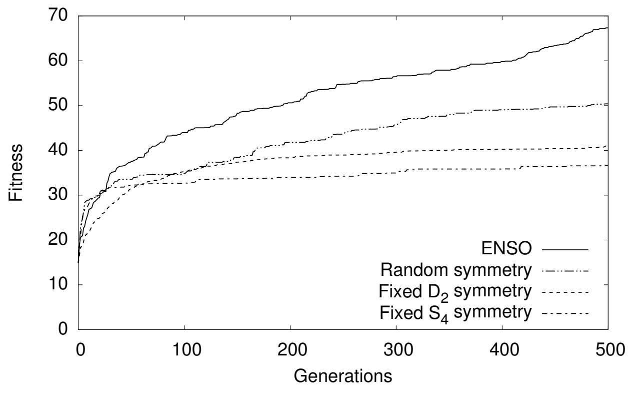
The first method (ENSO) initializes evolution with a population of fully symmetric phenotype graphs (graph G_A in Figure 1, having $\mathcal{S}_4$ symmetry). Since they have identical vertices and edges, their genotype trees have only two leaf colors, one representing vertex parameters and the other representing edge parameters. During evolution, color mutations break the initial graph symmetry minimally to create new types of vertices and edges, and parameter mutations optimize the initially random vertex and edge parameters.

The second method (random symmetry) initializes evolution in the same way as above, but color mutations change the color of vertices and edges of the graph randomly. Each such mutation chooses a random number of genotype leaf colors with probability proportional to the size of the set of vertex or edge elements associated with them. Each of these colors is then split into a random number of child colors corresponding to the subsets of elements produced by recursively partitioning the original set of elements. Like ENSO, these color mutations break graph symmetry, but unlike ENSO, they do not use group theory and therefore do not explore the subgroup lattice systematically (Figure 2). Consequently, the resulting symmetry break may not be minimal, producing large changes in symmetry than ENSO. Therefore, this method is likely to be less evolvable and is likely to perform worse than ENSO.

The third method initializes evolution in the same way as the above two methods, i.e. with graphs of $\mathcal{S}_4$ symmetry. However, it does not break this initial symmetry during evolution, and applies only parameter mutations to the phenotype graphs. Therefore, it is a good baseline to compare with the above methods, and thus to identify performance improvements due to symmetry evolution.



(a) Quadruped robot on flat ground



(b) Quadruped robot on inclined ground

Figure 6: Performance of controllers evolved using ENSO, random symmetry breaking, fixed $\mathcal{S}_4$ symmetry, and fixed $\mathcal{D}_2$ symmetry methods on flat and inclined ground. The plots are averages over ten trials of evolution. (a) On flat ground, all four methods perform similarly and achieve the same high level of fitness because many symmetries (including the hand-designed ones) can produce effective gaits in this case. (b) On inclined ground, however, both symmetry evolution methods perform better than the hand-designed symmetries because the best symmetries are difficult for humans to conceive. Moreover, the systematic group-theoretic search approach of ENSO finds significantly better symmetries than the random search approach.

The fourth method also evolves only parameters, keeping the symmetry of the phenotype graphs fixed, but these graphs have $\mathcal{D}_2$ symmetry (graph G_B in Figure 1) instead of $\mathcal{S}_4$. This symmetry is the same manually-determined symmetry that Valsalam and Miikkulainen [24] utilized to evolve effective quadruped controllers. Thus, this experiment forms a second comparison baseline for determining whether symmetry evolution can find more appropriate symmetries than those found through mathematical analysis.

5. RESULTS

Results of experiments comparing the ENSO method with the other methods for evolving modular controllers are now described. The experiments are run in a realistic physical simulation of robot locomotion on flat ground and on inclined ground.

5.1 Experimental Setup

The experiments were implemented utilizing a number of open source tools. The ENSO code was implemented as a library layer on top of the Open BEAGLE evolutionary computing framework [17], taking advantage of its generic programming interface. The experiments also utilized the customizable methods for logging and statistics collection in Open BEAGLE, as well as its XML-based configuration mechanism for managing parameters and specifying operators. The physics simulation was programmed using OPAL [16], an abstraction library on top of the Open Dynamics Engine (ODE) [14]. The Object-Oriented Graphics Rendering Engine (OGRE) [15] library was used for 3D visualization of the simulation.

In each experiment, the initial population of controllers has connection weights chosen randomly from the range $[-2, 2)$, neuron biases set to 0, and neuron sigmoid slopes set to 1. Parameter mutations are implemented as Gaussian perturbations (with $\sigma = 0.2$) acting with a specified probability (0.5) on each of the parameters. All edges are enabled in the phenotype graphs of the initial controllers, and mutations toggle them with a specified probability (0.1). In each generation, an offspring is created by first selecting a parent in a two-way tournament, and then applying either a parameter mutation, an edge-toggle mutation, or a color mutation. Parameter mutations are 100 times more likely, and edge-toggle mutations are ten times more likely, than color mutations. Each color mutation creates five offspring, all having the same symmetry, and the parameters in their newly created child colors are initialized randomly. In addition to the offspring created by mutations, the network with the best fitness is copied without change to the next generation. A population size of 200 is used in all experiments.

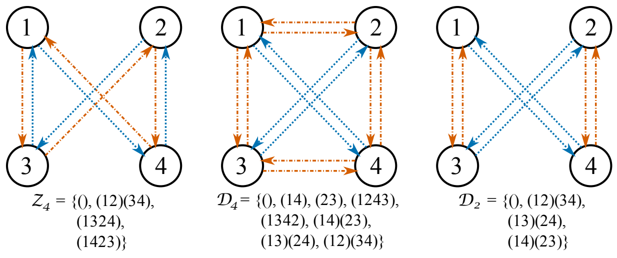
Each controller network is evaluated in a physically realistic simulation in which the network controls the locomotion of a robot. When the robot is initially placed in the simulation environment, its longitudinal and lateral axes are aligned with the coordinate directions of the ground plane. The simulation is then carried out for one minute of simulated time with step size 0.01s. At the end of the simulation, the fitness of the controller network is calculated as a function of how far the robot travels. This function is different on flat ground and on inclined ground (as explained later). Although appropriate as a quantitative measure of performance, this fitness measure does not capture how good the controllers are qualitatively. Therefore, the resulting gaits are also visualized and evaluated manually at the end of evolution to confirm that the champion controller networks have good locomotive properties.

For all experiments, evolution was run for 500 generations and repeated ten times, each time with a different random number seed. The average fitness of champion networks for all experiments is shown in Figure 6. The following subsections discuss the results of each experiment in detail.

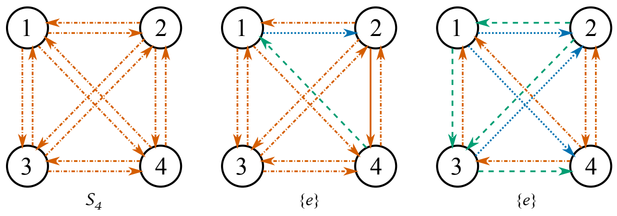
5.2 Flat Ground

In the first set of experiments, the four methods reviewed in Section 4.3 were used to evolve modular controller networks for the quadruped robot on flat ground. These experiments are based on the Euclidean distance that the robot travels as the fitness measure. All four methods produce similar fitness through all generations, as illustrated in Figure 6a (their differences at the end of evolution are not statistically significant according to the Student's t-test, with $p > 0.23, df = 18$). This result implies that $\mathcal{S}_4$ symmetry is sufficient for controllers to produce fast gaits on flat ground, that is, breaking that symmetry manually or through evolution does not improve performance.

However, differences between ENSO and random symmetry evolution are evident in the symmetries of champion phenotype graphs they evolve. Although both methods mutate symmetry at the same rate, champions in many runs of random symmetry evolution have the same $\mathcal{S}_4$ symmetry with which they were initialized, while ENSO evolved a variety of effective symmetries. This result implies that the unsystematic and large breaks in symmetry resulting from random symmetry mutations often produce graphs with low fitness that do not survive, and those that do survive have low symmetry (Figure 7b). In contrast, ENSO produces graphs with higher symmetry


$\mathcal{Z}_4$ = {(), (12)(34), (1324), (1423)}
$\mathcal{D}_4$= {(), (14), (23), (1243), (1342), (14)(23), (13)(24), (12)(34)}
$\mathcal{D}_2$ = {(), (12)(34), (13)(24), (14)(23)}

(a) ENSO (minimal symmetry breaking)



(b) Random symmetry breaking evolution

Figure 7: Phenotype graphs of typical champion networks evolved by ENSO and random symmetry evolution on flat ground. (a) The group theory mechanisms used in ENSO for minimal symmetry breaking biases evolution to produce phenotype graphs with high symmetry. Consequently, the robots they control have well-coordinated legs and smooth gaits. (b) In contrast, breaking symmetry randomly often produces large changes in symmetry that are deleterious and therefore do not survive. As a result, the champions of many evolutionary runs retain their initial $\mathcal{S}_4$ symmetry (left graph). Other champions such as the middle and right graphs have low symmetry that produce less coordinated, stumbling gaits.

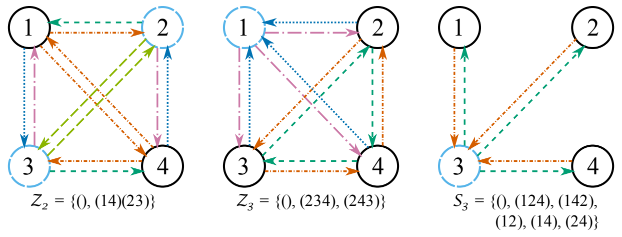
consistently because it breaks symmetry minimally using group theory (Figure 7a).

The evolved symmetries also impact the quality of gaits the controllers produce, as observed in visualizations of the locomotion of champion networks. The more symmetric champions evolved by ENSO produce smooth gaits with well-coordinated legs, while the less symmetric champions from random symmetry evolution produce stumbling gaits because legs are less coordinated. Both fixed symmetry methods also produce smooth and well-coordinated gaits, resembling quadruped gaits such as pronk, bound, and trot seen in animals. Visualization videos of such behaviors can be seen at the website `http://nn.cs.utexas.edu/?enso-robots`.

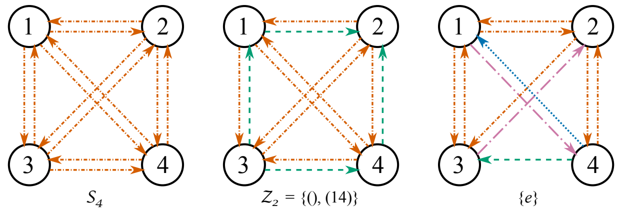
5.3 Inclined Ground

In the second set of experiments, the ground was rotated about the longitudinal coordinate direction of the robot by 20° to make the task of the controller more difficult. The fitness measure is the distance the robot travels along the longitudinal coordinate minus the distance it travels along the lateral coordinate. This measure encourages evolution to find controllers that move the robot forward in a straight line. Thus, the robot must walk across the incline without climbing up or down, while avoiding the risk of tipping over or slipping down the incline.

Since the robot is the same as before, with no morphological changes, the same manually designed symmetries should apply. However, when the robot is on inclined ground, the direction of gravity is not aligned with its plane of symmetry, thus breaking the symmetry of its dynamics in a way that is difficult for a human designer to take into account. As a result, the appropriate controller symmetries for this task are expected to be different from those utilized for walking on flat ground. Therefore, this task is a good test case to determine whether the fixed symmetry methods can evolve effective controllers when appropriate symmetries are difficult to design by hand. Moreover, the task will evaluate whether ENSO is more effective than random symmetry evolution at finding those symmetries.


$\mathcal{Z}_2$ = {(), (14)(23)}
$\mathcal{Z}_3$ = {(), (234), (243)}
$\mathcal{S}_3$ = {(), (124), (142), (12), (14), (24)}

(a) ENSO (minimal symmetry breaking)



(b) Random symmetry breaking evolution

Figure 8: Phenotype graphs of typical champion networks evolved by ENSO and random symmetry evolution on inclined ground. (a) The graphs that produce effective gaits on inclined ground are often less symmetric than those on flat ground (Figure 7a). They typically have two vertex colors, representing two types of controller modules that produce the different leg behaviors of such gaits. (b) As on flat ground (Figure 7b), random symmetry evolution produces many graphs with the initial $\mathcal{S}_4$ symmetry, which however results in less effective gaits on inclined ground. Other graphs it produces can generate faster gaits, but they are often slower than the gaits produced by ENSO. Thus, the systematic symmetry search of ENSO is more effective when finding the right symmetry is more important.

The results of these experiments are shown in Figure 6b. ENSO produces significantly better fitness than random symmetry evolution (according to the Student's t-test, with $p < 0.002, df = 18$), which in turn produces significantly better fitness than evolution of fixed $\mathcal{D}_2$ symmetry ($p < 0.04, df = 18$). The differences between the two fixed-symmetry methods are not statistically significant ($p > 0.13, df = 18$). Since the only algorithmic difference between ENSO and random symmetry evolution is the way symmetries are broken, these results demonstrate that the group-theoretic symmetry mutations of ENSO are significantly better at evolving the appropriate symmetries than random symmetry mutations. In addition, they demonstrate that finding these symmetries is crucial for evolving effective controllers, since fixed symmetry evolution utilizing hand-designed symmetries performs significantly worse.

In this more challenging task, the phenotype graphs that ENSO evolves on inclined ground (Figure 8a) are often less symmetric than those it evolves on flat ground (Figure 7a). In particular, it evolves graphs that have two vertex colors, and therefore the corresponding controllers have two types of modules, making it possible for evolution to implement a different control function in each module. Different modules can implement different leg behaviors useful for walking effectively on inclined ground. Typically, two (or three) legs of the same module type remain nearly stationary to provide the support necessary for maintaining the robot's forward orientation, while the other legs make a full circle, propelling the robot forward without slipping.

The unsystematic symmetry mutations of random symmetry evolution are typically detrimental on inclined ground as well, and as a result many of the champion phenotype graphs retain their original $\mathcal{S}_4$ symmetry (Figure 8b). However, occasionally it manages to discover symmetries that generate faster gaits than the fixed symmetry methods. The gaits the fixed symmetry methods produce on inclined ground are similar to those they produce on flat ground because the possible gaits are constrained by symmetry. However, these gaits are not as effective on inclined ground, and the gaits discovered by

ENSO are faster. These results demonstrate that evolving modular controllers by utilizing the systematic symmetry search of ENSO finds significantly faster gaits than by utilizing random symmetry search or by designing the symmetry by hand.

6. DISCUSSION AND FUTURE WORK

In the experiments, the regularities of the robot morphology and the environment were expressed in the form of controller symmetries, i.e. the type of control module for each leg, and the type of connection between each pair of modules. A human designer can determine these symmetries analytically in simple cases such as a quadruped robot on flat ground, but cannot (at least not as easily) for real-world environments with inclines and other such complexities. Given the number of modules, the group-theory based systematic symmetry search of ENSO makes it possible to find such symmetries automatically for all environments. Demonstrating this capability, ENSO evolved gaits similar to those based on hand-designed symmetries on flat ground, and significantly faster gaits on inclined ground. In contrast, random symmetry search results in significantly inferior gaits compared to ENSO. These results suggest that ENSO is an effective abstraction of generative and developmental systems for evolving symmetric modular controllers.

In the future, the approach will be tested with more complex robots with more legs, more complex legs, and sensors that receive more varied stimuli from the environment. Using such sophisticated robot models will allow ENSO to evolve controllers that produce high-level behaviors such as path-following and foraging, in addition to generating regular gaits. If ENSO can successfully evolve controllers for a sufficiently detailed model of a physical robot, then they can eventually be evaluated on physical robots. Thus, ENSO is a promising approach for developing efficient, robust, and flexible controllers for multilegged robots in the real world.

ENSO can also be used to evolve solutions for other control problems that are characterized by symmetry and modularity. For example, it can be used to design multiagent systems consisting of agents that interact with each other and with an environment, like those in online auctions and robotic soccer. The behavior of these agents will be represented as neural network modules and their interactions as (symmetric) connections between the modules. Another application is in designing distributed control systems for automating manufacturing processes. Such systems consist of controller modules interconnected by communication networks, which can be implemented as modular neural networks. In both cases, identical modules and connections between them produce symmetries that ENSO can exploit to design effective control systems.

In addition to using ENSO in various applications, the ENSO approach itself can be extended in four ways to improve its capabilities. First, the computationally hard group theory computations can be approximated with fast graph computations to improve the scalability of the approach. Second, the current manual decomposition of a given problem into modules can be automated using hierarchical clustering algorithms. Third, instead of using a fixed architecture for the modules, the architectures can be evolved using techniques such as NEAT [23]. Fourth, crossover of genotype trees can be implemented by swapping subtrees of parent trees if those subtrees have the same structure and node colors. These extensions would enhance evolutionary search so that ENSO can potentially solve more difficult problems and a wider variety of problems.

7. CONCLUSION

This paper proposes a novel abstraction of generative and developmental systems called Evolution of Network Symmetry and mOdularity (ENSO) that is useful for solving distributed control problems. ENSO utilizes group theory to search for symmetry systematically, making it possible to evolve the modules and the relationships between them effectively. This approach was evaluated by evolving neural network controllers for a quadruped robot simulated with physical realism. On flat ground, it evolved effective controllers similar in performance to those with hand-designed symmetries and randomly evolved symmetries. On inclined ground, ENSO discovered symmetries that produce significantly faster gaits than the other two methods, suggesting that it is useful for complex control problems in the real world.

8. REFERENCES

[1] O. Bastert. *Stabilization Procedures and Applications*. PhD thesis, Technische Universität Müchen, 2001.

[2] R. D. Beer, H. J. Chiel, and L. S. Sterling. Heterogeneous neural networks for adaptive behavior in dynamic environments. In *Advances in Neural Information Processing Systems 1*, pages 577–585, 1989.

[3] J. C. Bongard and R. Pfeifer. Repeated structure and dissociation of genotypic and phenotypic complexity in artificial ontogeny. In *Proceedings of GECCO*, pages 829–836, 2001.

[4] J. J. Collins and I. N. Stewart. Coupled nonlinear oscillators and the symmetries of animal gaits. *Journal of Nonlinear Science*, 3(1):349–392, 1993.

[5] GAP – groups, algorithms, and programming, 2007. `http://www.gap-system.org`.

[6] A. Garcia-Bellido. Symmetries throughout organic evolution. *PNAS*, 93(25):14229–14232, December 1996.

[7] F. Gruau. *Neural Network Synthesis Using Cellular Encoding and the Genetic Algorithm*. PhD thesis, Ecole Normale Superieure de Lyon, France, 1994.

[8] M. Herzog and O. Manz. On the number of subgroups in finite solvable groups. *Journal of the Australian Mathematical Society (Series A)*, 58:134–141, 1995.

[9] F. Heylighen. The growth of structural and functional complexity during evolution. In *The Evolution of Complexity: The Violet Book of 'Einstein Meets Magritte'*, chapter 2, pages 17–44. Springer, 1999.

[10] G. S. Hornby and J. B. Pollack. Creating high-level components with a generative representation for body-brain evolution. *Artificial Life*, 8(3), 2002.

[11] H. Kitano. Designing neural networks using genetic algorithms with graph generation system. *Complex Systems*, 4:461–476, 1990.

[12] S. Luke and L. Spector. Evolving graphs and networks with edge encoding: Preliminary report. In *Late-Breaking Papers of Genetic Programming*, 1996.

[13] J. F. Miller. Evolving a self-repairing, self-regulating, French flag organism. In *Proceedings of GECCO*, 2004.

[14] ODE: Open dynamics engine, 2007. `http://www.ode.org/`.

[15] OGRE: Object-oriented graphics rendering engine, 2007. `http://www.ogre3d.org/`.

[16] OPAL: Open physics abstraction layer, 2007. `http://opal.sourceforge.net/`.

[17] Open BEAGLE, 2007. `http://beagle.gel.ulaval.ca/`.

[18] A. R. Palmer. Symmetry breaking and the evolution of development. *Science*, 306:828–833, 2004.

[19] K. Sims. Evolving 3D morphology and behavior by competition. In *Proceedings of Artificial Life IV*, pages 28–39, 1994.

[20] K. Stanley. Compositional pattern producing networks: A novel abstraction of development. *Genetic Programming and Evolvable Machines*, 8(2):131–162, June 2007.

[21] K. O. Stanley, D. B. D'Ambrosio, and J. Gauci. A Hypercube-Based encoding for evolving Large-Scale neural networks. *Artificial Life*, 15(2):185–212, Apr. 2009.

[22] K. O. Stanley and R. Miikkulainen. A taxonomy for artificial embryogeny. *Artificial Life*, 9(2):93–130, 2003.

[23] K. O. Stanley and R. Miikkulainen. Competitive coevolution through evolutionary complexification. *Journal of Artificial Intelligence Research*, 21:63–100, 2004.

[24] V. K. Valsalam and R. Miikkulainen. Modular neuroevolution for multilegged locomotion. In *Proceedings of GECCO*, pages 265–272, 2008.