Evolving Symmetry for Modular System Design

Vinod K. Valsalam and Risto Miikkulainen

***Abstract*—Symmetry is useful as a constraint in designing complex systems such as distributed controllers for multilegged robots. However, it is often difficult to determine which symmetries are appropriate. It is therefore desirable to design such systems automatically, e.g. by utilizing evolutionary algorithms that produce symmetry through developmental mechanisms. The success of these algorithms depends on how well they explore the space of valid symmetries. This paper presents an approach called Evolution of Network Symmetry and mOdularity (ENSO) that utilizes group theory to search the space of symmetries effectively. This approach was evaluated by evolving neural network controllers for a quadruped robot in physically realistic simulations. On flat ground, the resulting controllers are as fast as those having hand-designed symmetry, and significantly faster than those without symmetry. On inclined ground, where the appropriate symmetries are difficult to determine manually, ENSO produced significantly faster gaits that also generalize better than those of other approaches. On robots with a more complicated structure including knee joints, ENSO resulted in more regular gaits than the other approaches. These results suggest that ENSO is a promising approach for evolving complex systems with modularity and symmetry.**

***Index Terms*—Development, Modularity, Symmetry, Group theory, Multilegged robots.**

I. Introduction

Developmental systems can represent complex structures with regularities compactly through gene reuse [52]. They are useful for evolving solutions to large-scale problems for two reasons. First, their compact genotype representation makes the evolutionary search space smaller. The fitness landscapes of such smaller search spaces are likely to be less rugged [22], making them easier to search. Second, known regularities in the phenotype can sometimes be encoded into the genotype representation, into the evolutionary operators, and into the 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 multilegged robot can be decomposed into modules, each controlling a different leg [3, 57, 58]. The constraints specifying which modules and interconnections are identical can be expressed formally in terms of *symmetries*, i.e. permutations of modules that leave the solution invariant. These symmetries are crucial for solving many problems; for instance they determine the type of gaits that a multilegged robot controller can produce [11, 57, 58].

Therefore, the ability to search for symmetries effectively is useful when evolving solutions to such problems. While developmental systems can produce modular phenotypes with symmetries [19, 24, 49], how effectively they can explore different symmetries depends on the abstraction of development they use. Traditionally, these abstractions simulate the developmental growth process, e.g. by modeling cellular interactions or by using grammatical rewrite rules [52]. The resulting complexity in the mapping of genotype to phenotype makes it difficult to evolve the desired phenotypic symmetries through genetic variations. Other abstractions that bypass an explicit developmental process are also possible. For example, *HyperNEAT* uses function composition to capture the essential characteristics of the developmental process [50, 51]. As a result, phenotypic symmetries can be encoded more directly in the genotype as symmetric functions, and therefore can be evolved more easily.

This paper presents an approach that is designed specifically to evolve the symmetry of modular phenotypes. The main insight is that essential characteristics of development such as modularity, repetition of modules, and symmetry can be expressed in a unified manner in terms of *group theory*, which is the mathematical theory of symmetry. This approach, called Evolution of Network Symmetry and mOdularity (ENSO), utilizes a compact genotype, storing the parameters of identical modules only once, while allowing variations between them to evolve. It uses a simple genotype-phenotype mapping that makes the entire space of phenotype symmetries easily accessible to evolutionary search. Thus ENSO leverages the properties of development for designing artificial systems.

ENSO also utilizes domain knowledge by initializing evolution with user-identified modules, making it possible to solve complex problems such as the design of distributed controllers and multiagent systems. For instance in this paper, ENSO is used to evolve neural network controllers for quadruped robots in physically realistic simulations. ENSO evolves robust 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. ENSO also produced more regular gaits than the other approaches on a more challenging robot with knee joints,. Since inclines, multi-jointed legs, and other similar complexities are common in the real world, these results suggest that ENSO is a promising approach for designing distributed controllers in real-world applications.

This paper is organized as follows. The next section presents the biological and mathematical background for ENSO, and reviews related work. Section III describes the ENSO approach, its genotype and phenotype representations, and its evolutionary operators. Section IV discusses the quadruped robot model, the architecture of its modular controller, and the evolutionary experiments designed to evaluate ENSO. Section V presents

V. Valsalam and R. Miikkulainen are with the Department of Computer Sciences, The University of Texas at Austin, Austin, TX 78712 USA (e-mail: vkv@cs.utexas.edu; risto@cs.utexas.edu).

the results of these experiments, demonstrating the benefits of ENSO. Section VI discusses the results and presents directions for future work.

II. Background and Related Work

This section begins by discussing the biological motivation for ENSO and then reviews prior work on developmental systems. This review is followed by a discussion of the group-theoretical concepts that ENSO utilizes to represent symmetry. These concepts are then applied to coupled cell systems, which model controllers for multilegged robots.

A. Biological Motivation

Symmetry in nature expresses constraints between identical subsystems. For example, a bilaterally symmetric animal has identical left and right legs that are constrained to be equidistant from its plane of symmetry, and the symmetric neural circuitry controlling its locomotion has identical modules constrained by the nature of their interconnections [11]. Symmetry in such systems can potentially provide two benefits: (1) it allows encoding identical subsystems compactly using a common set of genes [1, 32, 43] for easier evolutionary search and (2) it can result in patterned oscillations in neural circuits [11, 25, 47], e.g. for effective gaits.

How did organisms with different symmetries evolve in nature? Evidence suggests that more symmetric organisms evolved into less symmetric organisms [36, 43]. For example, primitive, single-celled organisms like protozoans are highly symmetric with spherical shapes. They evolved into less symmetric organisms such as jelly fish, which have radial symmetry. And radially symmetric organisms in turn evolved into even less symmetric organisms, e.g. bilaterally symmetric humans. Simply put, nature evolves the appropriate level of symmetry through symmetry breaking.

Mutations that break symmetries produce novel phenotypic variations [43]. 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, symmetric sides. Breaking this symmetry requires the genome to specify additional information on how one side is different from the other, thus increasing the complexity of the genome.

In fact, symmetry is fundamentally related to complexity, allowing complexity to be characterized as the lack of symmetry [21]. Increase in complexity of organisms during evolution is accompanied by symmetry breaking at different levels of organization [16]. Moreover, *complexification*, i.e. incrementally increasing complexity, makes evolutionary search more effective because it allows evolution to start with low-dimensional genotypes, which are likely to be less rugged and therefore easy to optimize, and gradually add more dimensions while optimizing them further [22, 53]. Building complex systems by incrementally elaborating solutions in this manner is more likely to succeed than evolving solutions in the final high-dimensional space directly. Therefore, evolving complex systems from simple symmetric systems by breaking their symmetry step-by-step might be a good way to design them. This idea motivates the symmetry breaking approach of ENSO discussed in Section III.

Nature utilizes a developmental process to construct the phenotype and express the symmetries encoded in the genotype. ENSO bypasses this developmental process by utilizing a genotype representation that encodes symmetries explicitly, as described in Section III. The contributions of other researchers in devising evolutionary algorithms based on computational abstractions of development, called *indirect encodings*, are discussed next.

B. Indirect Encodings

Most indirect encodings were developed for evolving artificial neural networks, and Kitano was one of its earliest practitioners [28]. He evolved matrix rewrite rules that produce the adjacency matrix of neural networks through a series of rewrite steps. This method is based on *L-systems*, which are grammatical string rewrite rules that were first developed by Lindenmayer to model the biological growth of plants [34]. Such systems yield complex tree-like structures resembling fractals. Kitano's scheme produced similar matrix structures. They are generated from an initial 2×2 matrix, whose symbols are rewritten iteratively with other 2×2 matrices, creating larger and larger matrices. Repeating the symbols in the matrix creates regularities and symmetries. The rewriting stops when the current matrix contains only numerical values, which are produced by terminal symbols. The result is then interpreted as a neural network's adjacency matrix, specifying its connectivity parameters. Kitano evolved such networks to solve encoder/decoder problems. However, these networks are typically very large because their size is exponential in the number of rewrite steps. Evolving detailed connectivity between network units is also difficult.

Boers and Kuiper used a different L-system to evolve the topology of modular neural networks [5]. Their system was based on context-sensitive graph rewriting to describe neural network topologies. Repetition of rule strings and recursive application of the rules result in modular network architectures. They evolved only the architecture of the networks in this manner; the connection weights were later determined using the backpropagation algorithm [9]. Using this method, they evolved solutions for problems such as XOR and shape recognition. Because backpropagation was used for learning connection weights, the networks were limited to feed-forward architectures. Moreover, a different symbol was used for each unit in the network, limiting the scalability of the approach.

Sims used another type of generative graph-based encoding to evolve virtual creatures in simulated physical environments [49]. He used directed graphs as genotypes to encode developmental instructions for constructing the morphology of the creatures. The nodes of the graph contain information on synthesizing body parts, while its edges specify the order in which to synthesize them. Multiple edges to the same child node result in reuse of body parts, which is useful for creating multiple instances of limbs. Recursive edges are also possible, producing repetitive, fractal-like morphologies. The

neural network control circuitry of the creature was embedded in the genotype graph and evolved along with its morphology. Although the developmental mechanism in this method was elementary, the resulting creatures were significantly regular and capable of a variety of interesting locomotive behaviors.

Gruau also used graphs as genotypes in a method called *cellular encoding* (CE), that was inspired by how biological development occurs through cell division [19]. The genotype encodes a program tree for constructing a neural network from a single ancestral cell. These program trees are then evolved using standard techniques for genetic programming [31]. The nodes of the tree contain cellular developmental instructions, such as for splitting a cell into two, deleting a connection between two cells, or changing the weight of a connection. A full neural network is built by executing these instructions in the sequence specified by the edges of the program tree. Gruau showed that networks with repeated structures can be produced by using a recursion instruction that transfers control of development back to the root of the program tree. He evolved such networks to solve problems with regularity such as finding the parity and symmetry of a set of binary digits.

Luke and Specter identified several weaknesses of CE and proposed *edge encoding* (EE) as an alternative to address many of those concerns [35]. For example, crossover in CE can produce drastic changes in the phenotype of an offspring, which may be problematic for evolution in many domains. Moreover, the networks produced by CE tend be highly interconnected because they are grown by splitting cells into two or more interconnected cells. Such networks are a disadvantage in domains where such high connectivity is not required. CE also does not provide a convenient mechanism to tune connection weights because cells, not connections, are the target of its development instructions. EE grows networks by modifying edges rather than cells, thereby avoiding these problems of CE and making it more suitable in some domains. However, although both CE and EE are expressive enough to produce all possible graphs, it is not clear how their particular biases affect their performance in any given problem.

In a domain similar to Sims' simulated 3D virtual creatures, Hornby and Pollack combined the ideas of CE and EE with L-systems to simultaneously evolve the body and brain of such creatures [24]. They used strings of build commands for constructing the neural network brains instead of the trees of build commands that are used in CE. Moreover, these build commands operate on connections in the network as in EE. They defined a different set of commands for building body parts of the creatures. The separate command languages for building the body and brain were then combined using an L-system and evolved. The resulting creatures were more complex, having more parts and regularity, and faster than similar creatures evolved using a non-developmental encoding. They were also more complex than the creatures produced by Sims.

Bongard and Pfeifer also evolved similar virtual creatures, but by using an abstraction of *genetic regulatory networks* (GRNs) for encoding bodies and neural networks [6]. GRNs model gene expression inside biological cells, i.e. the interactions between genes as they regulate each other through the production of proteins [26]. In Bongard and Pfeifer's work, the creature begins development as a single spherical unit. Depending on the concentrations of gene products inside this unit, it grows in size and eventually divides into two child units. These units are attached to each other by a joint. Each unit contains a small neural network, which develops according to a variant of the CE method. In this variant, different gene products trigger different operations that modify the local network. The development continues until a fixed number of time steps is reached. Using this method, the authors produced creatures with hierarchical repeated structures in the task of pushing a block.

Dellaert and Beer had previously used an abstraction of GRNs called *random Boolean networks* (RBNs) to evolve simulated agents capable of following curved lines [12]. In their method, cells representing the body of the agent develop first. A neural network develops on top of the arrangement of these cells when specialized cells send out axons, making connections with other cells within its range. A similar neural network developmental model was used by Cangelosi, Parisi and Nolfi to create organisms that seek out food and water [7]. Their networks grow in a two-dimensional space using processes such as cell division and axon growth. Kodjabachian and Meyer also used connection growth mechanisms in their geometry-oriented version of CE called SGOCE [29]. 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 [38].

In the above methods, small changes in the genotype often produce unpredictable changes in the phenotypes. Steiner et al. proposed to reduce this effect by manipulating the phase space of the dynamic system of the GRN directly [54]. Moreover, GRN-based approaches abstract biological development at levels lower than those of graph-based methods by modeling biological growth processes in varying detail. However, detailed simulation of biological processes are computationally expensive, and may be unnecessary or even counterproductive [12]. Therefore, determining the right level of developmental abstraction for indirect encodings is an important research topic.

Addressing this issue of abstraction, Stanley proposed an indirect encoding called *Compositional Pattern Producing Networks* (CPPNs) that eliminates the traditional local interaction and temporal unfolding mechanisms of developmental systems [50]. Instead, he argued that the effects of such mechanisms can be obtained by composing specific functions in the appropriate order, i.e. by constructing a CPPN. 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 [51]. Clune et al. also used HyperNEAT to evolve controllers for a quadruped robot [10]. However, unlike the controllers evolved by ENSO, they found it useful to utilize external oscillations to produce gaits.

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. In contrast, symmetry plays a central role in the ENSO approach presented in this paper, i.e. other characteristics of development such as modularity and repetition are also expressed in terms of symmetry (Section III). ENSO makes this unified treatment possible by utilizing group theory, which is introduced next.

C. Symmetries and Group Theory

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* [2] 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.

The symmetries of a graph can be represented mathematically as a group [4, 8]. A *group* is a set $\mathcal{G}$ of elements closed under a binary operation $\circ$ satisfying the following axioms:

Associativity: For all $g, h, k \in \mathcal{G}$, $(g \circ h) \circ k = g \circ (h \circ k)$.

Identity element: There exists an element $e \in \mathcal{G}$ such that for all $g \in \mathcal{G}$, $e \circ g = g \circ e = g$.

Inverse element: For each $g \in \mathcal{G}$, there is an element $g^{-1} \in \mathcal{G}$ such that $g \circ g^{-1} = g^{-1} \circ g = e$.

The operation $g \circ h$ is usually written more compactly as gh.

A *subgroup* $\mathcal{H}$ of a group $\mathcal{G}$, denoted $\mathcal{H} \leq \mathcal{G}$, is a subset of the group elements of $\mathcal{G}$ satisfying the group axioms under the same operation. If the subset is a proper subset of $\mathcal{G}$, then the subgroup is called a *proper subgroup* of $\mathcal{G}$. A *maximal subgroup* of $\mathcal{G}$ is any proper subgroup $\mathcal{S}$ such that no other proper subgroup $\mathcal{T}$ contains $\mathcal{S}$ strictly. If $\mathcal{G}$ represents the symmetries of a graph G, then its proper subgroup $\mathcal{H}$ represents the symmetries of a less symmetric graph H. ENSO uses this fact to compare graph symmetries.

Two subgroups $\mathcal{S}$ and $\mathcal{T}$ are said to be *conjugate* if there exists an element $g \in \mathcal{G}$ such that $\mathcal{T} = g\mathcal{S}g^{-1}$, i.e. $\mathcal{T} = \{gsg^{-1} \mid s \in \mathcal{S}\}$. It can be shown that conjugacy is an equivalence relation, partitioning the set of all subgroups of a group $\mathcal{G}$ into equivalence classes called *conjugacy classes*. The subgroups belonging to a given conjugacy class represent graphs with similar symmetries. Therefore, conjugacy classes are useful for characterizing the space of graph symmetries that ENSO has to search.

Fig. 1 illustrates how the above definitions can be used to represent the symmetries of a completely colored graph with four vertices. 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 $\mathrm{Aut}(G)$.

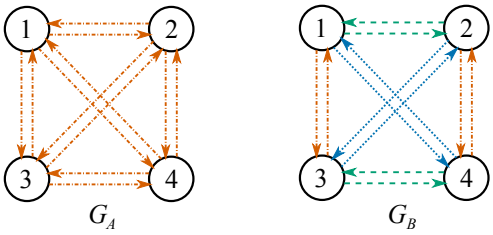


Fig. 1: Representing graph symmetries using groups. Each vertex and edge has a color (indicated by both color and line style) representing a particular combination of parameters. A graph symmetry is any permutation of vertices under which the edge colors remain the same. Both graphs in this figure have 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.

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 coloring of a graph G such that its new automorphism group is a subgroup of its original automorphism group is said to *break the symmetry* of G. In order to implement symmetry breaking, ENSO defines a canonical complete coloring of G for any given automorphism group $\mathcal{G}$ using the concept of group action. Formally, the *action* of $\mathcal{G}$ on the vertex set V is a function $\mathcal{G} \times V \to V$, denoted $(g, v) \mapsto g \cdot v$ for each $g \in \mathcal{G}$ and each $v \in V$, which satisfies the following two conditions:

1) $e \cdot v = v$ for every $v \in V$, where e is the identity element of $\mathcal{G}$, and
2) $(gh) \cdot v = g \cdot (h \cdot v)$ for all $g, h \in \mathcal{G}$ and $v \in V$.

The set of all $w \in V$ to which v is mapped by the elements of $\mathcal{G}$ is called the *orbit* of v. Similarly, the coordinate-wise action of $\mathcal{G}$ on $V \times V$ is defined as $g \cdot (v, w) = (g \cdot v, g \cdot w)$ for any $(v, w) \in V \times V$. The orbits in this action are called *orbitals*, and they form a partition of $V \times V$ called an

orbital partition. Assigning a different color to each part of this partition produces the desired canonical complete coloring of G.

If a graph G' is produced by breaking the symmetry of G, then the orbital partition ρ' under the action of $\mathrm{Aut}(G')$ is a *refinement* of the orbital partition ρ under the action of $\mathrm{Aut}(G)$, i.e. each part of ρ' is a subset of a part of ρ. Therefore, the canonical complete coloring of G' can be obtained from that of G by assigning new colors to the parts of ρ' that are a proper subset of a part of ρ and retaining the colors of parts of ρ' that are also parts of ρ. ENSO represents this hierarchical relationship between the colors of G and G' by organizing the new colors of G' as the children of colors of G that they replace. This organization produces a tree of colors when symmetry is broken repeatedly during evolution.

Breaking symmetry in the above manner induces an ordering of the graphs 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. Fig. 2 illustrates this lattice for the subgroups of $\mathcal{S}_4$. Nodes of this lattice represent conjugacy classes of subgroups. 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 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 [20], making exhaustive symmetry search intractable for graphs with a large number of vertices. Moreover, ENSO must search the parameter space of the modules for each symmetry. Therefore, ENSO uses evolution to focus the search on promising lattice regions and parameter combinations.

In this paper, ENSO is used to evolve locomotion controllers for multilegged robots. These controllers are first modeled as coupled cell systems by applying the above group-theoretic concepts, as described in the following subsection.

D. Coupled Cell Systems

A coupled cell system consists of a set of dynamical systems, called cells, and a specification of how the cells are coupled, i.e. how the state of each cell affects the states of the other cells [18]. Some or all of the cells and couplings may be identical, resulting in symmetries that correspond to permutations of the cells under which the behavior of the system is invariant. Such symmetric, coupled cell systems can exhibit synchronous and phase-related periodic patterns in their state. Collins and Stewart [11] showed that this patterned behavior can be used to model animal locomotion and to explain gait symmetries.

Following their method, the modular controllers in this paper are also modeled as symmetric coupled cell systems. The patterned oscillatory behavior produced by these symmetries is independent of the model parameters, i.e. the details of the internal dynamics of the cells do not matter. Therefore, analyzing the symmetries of a coupled cell system can give insights into the high-level qualitative behavior of the system.

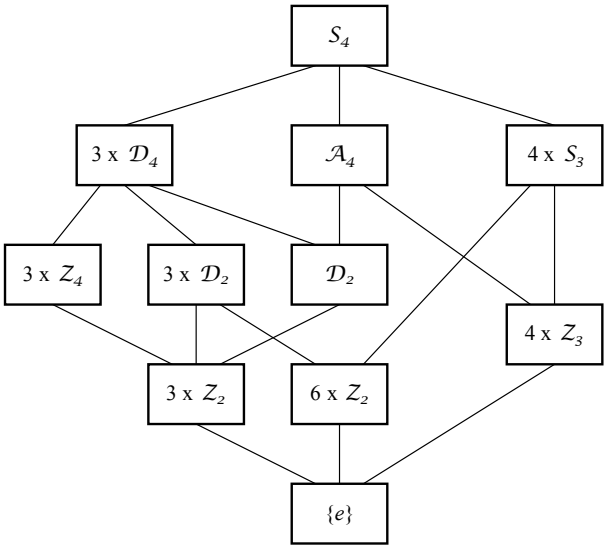


Fig. 2: Lattice of subgroups of $\mathcal{S}_4$. This lattice was computed using the GAP [15] software for computational group theory and shows the subgroups of the group $\mathcal{S}_4$, which is the symmetric group of degree four containing all 4! permutations of the set $V = \{1, 2, 3, 4\}$. Each 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.

This analysis is illustrated below for a coupled cell system due to Pinto and Golubitsky [44]. While they used this system to understand biped locomotion, it is adapted in this review to model quadruped gaits. This system consists of four identical cells, described by the following system of ordinary differential equations (ODEs):

$$\begin{cases} \dot{\mathbf{x}}_1 = F(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4) \\ \dot{\mathbf{x}}_2 = F(\mathbf{x}_2, \mathbf{x}_1, \mathbf{x}_4, \mathbf{x}_3) \\ \dot{\mathbf{x}}_3 = F(\mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_1, \mathbf{x}_2) \\ \dot{\mathbf{x}}_4 = F(\mathbf{x}_4, \mathbf{x}_3, \mathbf{x}_2, \mathbf{x}_1), \end{cases} \tag{1}$$

where $\mathbf{x}_i \in \mathbf{R}^k$ are the k state variables of cell i, and $F : (\mathbf{R}^k)^4 \to \mathbf{R}^k$ encapsulates the internal dynamics of each cell and its coupling with other cells. Thus, this system of ODEs describes how the state variables of each cell change in time as a function of the cell's own state and the state of the other cells.

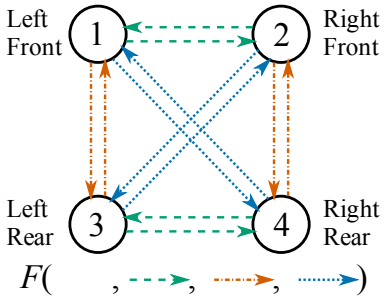


Fig. 3: Graph corresponding to the coupled cell system in equation (1). The vertices, numbered 1 through 4, represent cells and the edges represent coupling between the cells. The different edge colors (also indicated with different line styles) represent different couplings, corresponding to different argument positions in the function F as shown in the legend. This graph helps visualize the symmetries of the coupled cell system and shows how the cells may be assigned to control the legs of a quadruped robot to produce different gaits (Fig. 4). For example, these symmetries can constrain cells 1 and 2 to oscillate synchronously with phase opposite to that of similarly synchronous cells 3 and 4, producing the bound gait.

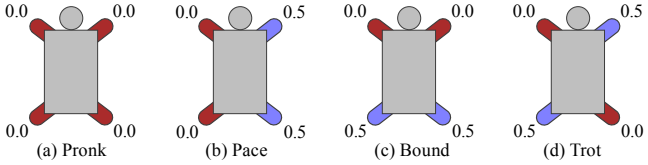


Fig. 4: Phase relations between legs in the pronk, pace, bound and trot gaits of quadrupeds. The numbers as well as the colors indicate phase of leg movement. In the pronk gait, all four legs move synchronously, while in the other gaits pairs of legs are synchronous and a half-period out of phase with the other pair.

TABLE I: Gaits corresponding to different combinations of phase shifts θ_g and θ_h associated with two permutation symmetries g and h of the coupled cell system in Fig. 3. Thus, this system can have solutions modeling a variety of common quadruped gaits.

	Pronk	Pace	Bound	Trot
θ_g	0	$\frac{T}{2}$	0	$\frac{T}{2}$
θ_h	0	0	$\frac{T}{2}$	$\frac{T}{2}$

This system can be represented by the graph in Fig. 3, which helps visualize its symmetries. The vertices of the graph represent cells and the edges represent coupling between the cells. Each edge color represents a different type of coupling, corresponding to a different argument position in the function F. This graph is the same as the graph G_B of Fig. 1 in Section II-C, where its symmetries were analyzed. In particular, its automorphism group is $\mathcal{D}_2$, consisting of the symmetric permutations $g = (1\ 2)(3\ 4)$, $h = (1\ 3)(2\ 4)$, and their composition hg.

Each symmetry of the graph induces a symmetry of the associated system of ODEs, i.e. a transformation γ such that $\gamma\mathbf{x}(t)$ is a solution whenever $\mathbf{x}(t)$ is a solution. For example, suppose $\mathbf{x}(t)$ is a solution to (1). Applying the permutation g to (1) produces an equivalent system of ODEs for which $g\mathbf{x}(t)$ is a solution. Thus, the system of ODEs inherits the symmetry g from the corresponding graph.

In particular, periodic solutions of the system are interesting because they model gaits. Let $\mathbf{x}(t)$ be a T-periodic solution to (1) and γ be a symmetry. Then $\gamma\mathbf{x}(t)$ is also a solution. Because solutions to the same initial conditions are unique, if $\mathbf{x}(t)$ and $\gamma\mathbf{x}(t)$ are the same trajectory, then their phases must be different, i.e. $\gamma\mathbf{x}(t) = \mathbf{x}(t+\theta)$ where $\theta \in [0, T)$ for all t. Since applying either g twice or h twice to a solution is equivalent to applying the identity, $2\theta \equiv 0 \pmod{T}$ for both symmetries. Therefore, the possible values of phase shift θ is either 0 or $\frac{T}{2}$ for both symmetries.

Such phase shifts impose constraints on the components of the solution $\mathbf{x}(t) = (\mathbf{x}_1(t), \mathbf{x}_2(t), \mathbf{x}_3(t), \mathbf{x}_4(t))$, producing specific patterned behavior for the system. For example, the bound gait pattern results from the following constraints. The symmetry g is first applied to $\mathbf{x}(t)$ with a phase shift of $\theta_g = 0$, resulting in the constraints $\mathbf{x}_2(t) = \mathbf{x}_1(t)$ and $\mathbf{x}_4(t) = \mathbf{x}_3(t)$. Consequently, the solution has the form $\mathbf{x}(t) = (\mathbf{x}_1(t), \mathbf{x}_1(t), \mathbf{x}_3(t), \mathbf{x}_3(t))$, implying that cells 1 and 2 are synchronous and cells 3 and 4 are synchronous, but their synchrony is independent, i.e. it does not yet produce an interesting gait. However, applying the symmetry h to this solution with a phase shift of $\theta_h = \frac{T}{2}$ results in a further constraint $\mathbf{x}_3(t) = \mathbf{x}_1(t + \frac{T}{2})$. Now, the solution has the form $\mathbf{x}(t) = (\mathbf{x}_1(t), \mathbf{x}_1(t), \mathbf{x}_1(t + \frac{T}{2}), \mathbf{x}_1(t + \frac{T}{2}))$, implying that cells 1 and 2 are synchronous, while cells 3 and 4 are also synchronous with the same periodic trajectory as cells 1 and 2, but half-period out of phase. Assigning these cells to control the legs of a quadruped robot as illustrated in Fig. 3 produces a bound gait (Fig. 4).

Other common quadruped gaits (such as those depicted in Fig. 4) can be obtained similarly by selecting different combinations of values for θ_g and θ_h as shown in Table I. Although these gaits are possible solutions of the system, whether any particular gait can be obtained in an instance of the system depends on the details of the cell dynamics and the couplings, i.e. on the function F in the ODEs. In prior work [57], we showed that this function F can be designed effectively by utilizing modular neuroevolution, i.e. by representing each cell as a neural network module and evolving its weights. The resulting controllers produced all four gaits listed in Table I.

The above theoretical results make the gait-production capabilities of such modular controller networks easy to understand. Consequently, in contrast to other approaches, these controllers are easy to design and scale well to robots with more legs and more complex legs [57]. Moreover, neuroevolution is an effective alternative to manual design of coupled cell system ODEs for producing desired behaviors (such as [11, 27, 45, 46]). The ENSO approach in this paper extends our prior work on modular neuroevolution [57] by evolving the symmetries of the system simultaneously. As a result, ENSO can evolve controllers that produce effective gaits even when such gaits are unknown and manual design of the required symmetries is difficult. This approach for simultaneous evolution of modules and symmetry is described next.

III. 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 II-C). 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.

A. 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. these graphs have automorphism group $\mathcal{S}_n$. 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 orbital partitions for each subgroup in the lattice at the beginning of evolution using the `GAP` [15] software. During evolution, ENSO utilizes this lattice to mutate the coloring of graphs, thus breaking their symmetry. Each such color mutation creates a new graph coloring from the orbital partition of a randomly chosen 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 is the phenotype that is constructed from the genotype. Each genotype tree of the initial population has two leaf nodes, one representing the color of vertices and the other representing the color of edges (Fig. 5a). These two leaf nodes are the children of a root node that represents a dummy color.

Thus each node in the genotype tree represents a particular color c. Second, it 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 (Fig. 5b). 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. These color changes break the symmetry of the phenotype graph, i.e. it loses its color invariance under a subset of permutations that were its symmetries prior to the mutation.

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 increases 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 [48, 53]. Complexification also means that simpler solutions are preferred over more complex solutions, thus conforming to the principle of Occam's razor, which often results in more robust neural networks [14, 33].

For each phenotype graph that ENSO produces in the above manner, it also 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.

B. 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 (Fig. 5). 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 (for instance, in a legged locomotion controller, such interconnections allow the control module of one leg to receive the state of another leg as input). If modules are not connected, then 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

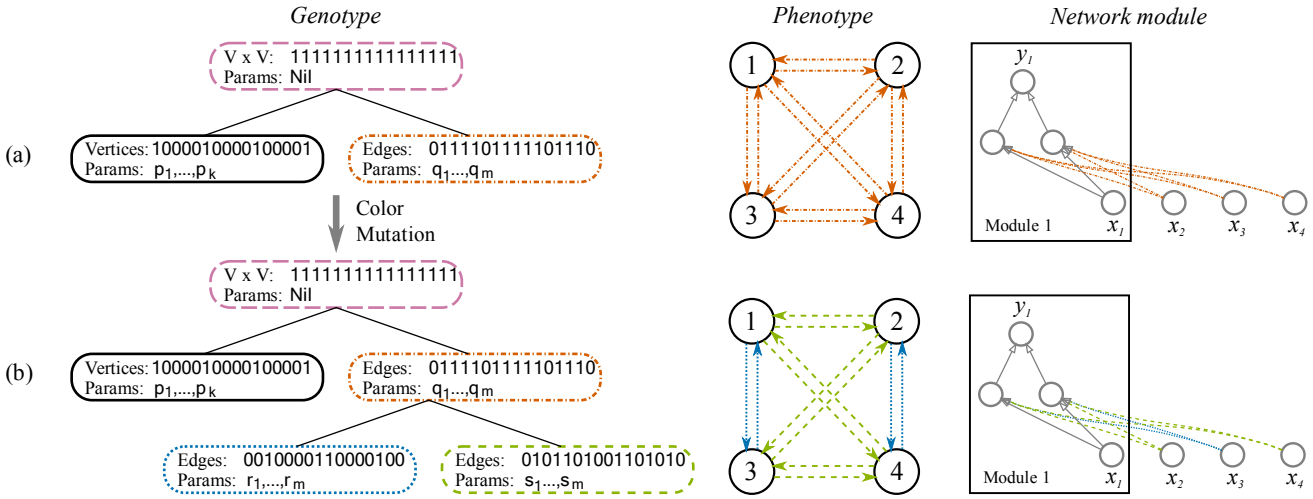


Fig. 5: Examples of genotype, phenotype, network module, and color mutation. ENSO uses a tree of colors as genotype (left). Each leaf of this tree has a unique color, and represents a set of vertices or edges of the phenotype graph (middle) that have the same parameter values. The vertices and edges 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 $\mathcal{S}_4$ (Fig. 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.

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 [53] and to the evolution of structure and parameters at different time scales in the EANT/EANT2 algorithm [48]. 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. As with NEAT and EANT/EANT2, protecting symmetry mutations using this niching mechanism was found empirically to improve 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 multilegged robots.

IV. Evolving Robot Controllers

Multilegged robots are inherently modular, making them a useful real-world application for ENSO. Controllers were evolved using ENSO for two different quadruped models by simulating their locomotion with realistic physics. The base quadruped model, its modular neural network controller, and the experimental methods for evaluating the controllers are described in the following subsections.

A. Robot Model

The robot model resembles a table with a rectangular body supported by legs at the four corners (Fig. 6). 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. 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.

In prior work, experiments with more legs and more complex legs were also run utilizing hand-designed symmetries for the controllers [57]. However, this paper focuses on evolving controller symmetries, first for the above robot model in different environments, and then for an extended model with knee joints as described in Section V-D.

B. Modular Controller

A neural network controller for the above quadruped robot can be constructed using four modules, each controlling a different leg. All modules have the same network architecture shown in Fig. 7a. 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

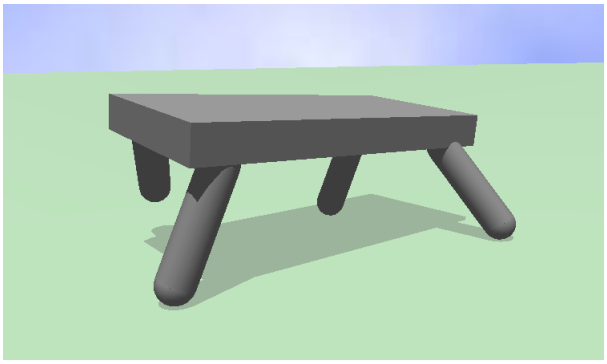
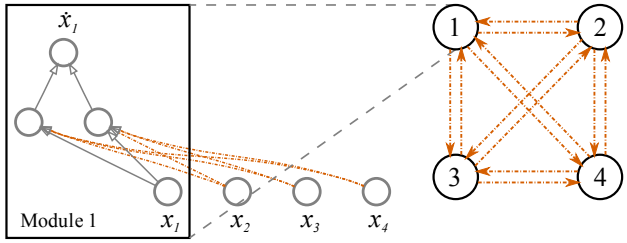
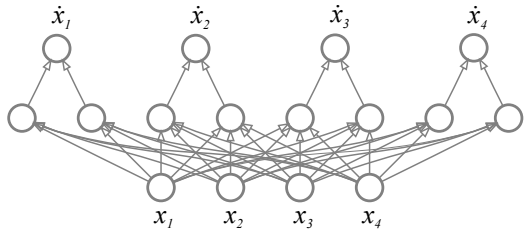


Fig. 6: The quadruped robot model. The legs are attached to the body by hinge joints with axes of rotation tilted sideways, allowing the legs to make full circular rotation. Locomotion is achieved by coordinating the circular movements of the legs. Although more legs and complex legs can be used, this model is a simple and physically realistic platform that has symmetric and modular controllers, and is therefore suitable for evaluating ENSO on challenging environments.



(a) Network module and initial phenotype graph



(b) Full controller network consisting of four modules

Fig. 7: 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), i.e. all vertices and edges have the same combination of network parameters. 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.

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 III-B).

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 (Fig. 7b). The weights of these connections are the edge parameters of the phenotype graph.

Such modular construction, in which the network computes the time derivative of the joint angles as a function of the joint angles, allows the controller to be modeled as a coupled cell system (Section II-D). The modules correspond to cells, and the input connections of modules correspond to couplings. Thus, this coupled cell system has the same graphical representation as the phenotype graph. As illustrated by the analysis in Section II-D, such a graph with appropriate symmetries can be used to produce gaits for robots. In prior work [57], we utilized the graph determined by such an analysis (instead of evolved) to determine the appropriate symmetries of modular controllers so that their network parameters could then be evolved. The resulting controllers produced all the regular, animal-like gaits predicted by theory. In contrast, the ENSO approach makes it possible to evolve the symmetries together with the module parameters, producing effective controllers even when the appropriate symmetry is difficult to determine analytically. The experimental methods used in demonstrating this result are described next.

C. Experimental Methods

In order to demonstrate the benefit of ENSO, five experimental methods for evolving the above modular controller were compared: (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 (i.e. maximal symmetry), (4) using fixed $\mathcal{D}_2$ symmetry during evolution (as was done in our prior work [57]), and (5) using direct encoding without symmetry constraints (which is equivalent to using fixed $\{e\}$ symmetry during evolution). Although these five methods differ in how they determine controller symmetry, they all evolve controller parameters in the same way as ENSO.

The first method (ENSO) initializes evolution with a population of maximally symmetric phenotype graphs (graph G_A in Fig. 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 phenotype 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 those colors. 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 (Fig. 2). Consequently, the resulting symmetry break may not be minimal, producing larger 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 for comparing with the above methods, making it possible to determine whether symmetry evolution produces better controllers.

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 Fig. 1). This symmetry is the same hand-designed symmetry (Fig. 3) utilized in our prior work [57] to evolve effective quadruped controllers. Thus this experiment provides a comparison baseline for determining whether symmetry evolution can find more appropriate symmetries than those found through mathematical analysis.

The fifth method utilizes a direct encoding of controller networks, without constraining them to be symmetric. The resulting networks correspond to phenotype graphs with unique nodes and edges, i.e. with the minimal symmetry group $\{e\}$ (located at the bottom of the subgroup lattice). This method also does not apply symmetry mutations and evolves only parameters. Moreover, by evolving controllers without symmetry, it provides a baseline to determine whether symmetry is in fact useful for evolving effective controllers.

V. Results

The results of the experiments evaluating the above methods are now described. Three experiments were performed, utilizing realistic physical simulation of robot locomotion. The first experiment evolved controllers for the quadruped robot described in Section IV-A on flat ground. The second and third experiments were made more challenging by extending the first experiment in two orthogonal ways: The second experiment included an inclined ground, while the third experiment added knee joints to the robot. The source code for these experiments is available from the website http://nn.cs.utexas.edu/?enso-code.

A. 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 [42] evolutionary computing framework, taking advantage of its generic programming interface. The physics simulation was programmed using OPAL [41], an abstraction library on top of the Open Dynamics Engine (ODE) [39]. The Object-Oriented Graphics Rendering Engine (OGRE) [40] library was used for 3D visualization of the simulation.

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

Each controller network was evaluated in a physically realistic simulation in which the network controlled the locomotion of a robot. When the robot was initially placed in the simulation environment, its longitudinal and lateral axes were aligned with the coordinate directions of the ground plane. The simulation was 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 was calculated as a function of how far the robot traveled normalized by its body length. This function was different in each experiment (as explained later). In addition to calculating performance in terms of fitness, the gaits that champion controllers produce at the end of evolution were also evaluated visually and graphically to determine whether they have properties such as regularity and smoothness that are important in the real world but are difficult to express in terms of fitness.

For all three experiments, evolution was run for 500 generations and repeated ten times, each time with a different random number seed. The following subsections discuss their results in detail.

B. Flat Ground

In the first experiment, the five methods discussed in Section IV-C were used to evolve modular controller networks for the quadruped robot on flat ground. These experiments use the Euclidean distance that the robot travels as the fitness measure. As illustrated in Fig. 8, all four symmetry-based methods perform significantly better than the direct encoding method that does not utilize symmetry (according to the Student's t-test, with $p < 10^{-8}, df = 18$). Therefore, symmetry is indeed useful for evolving effective controllers. Moreover, all four symmetry-based methods produce similar fitness through all generations (their differences at the end of evolution are not statistically significant 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, i.e. 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

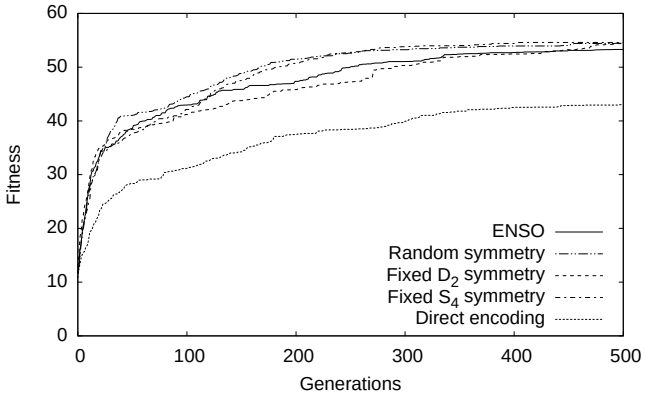
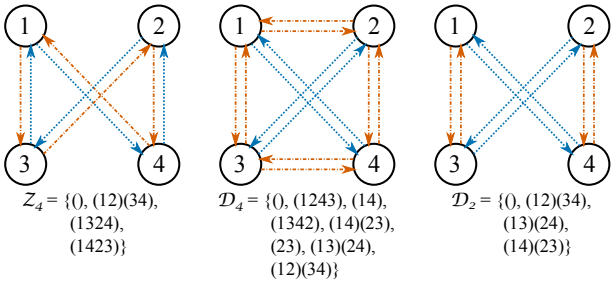
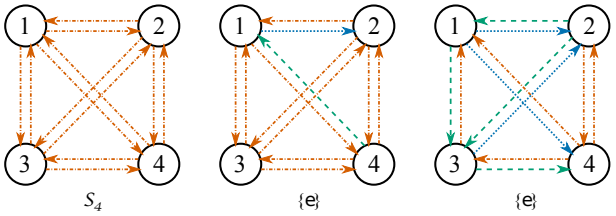


Fig. 8: Performance of quadruped controllers evolved using various methods on flat ground. The curve for each method shows the average fitness of champion controllers in ten trials of evolution. The direct encoding method that does not utilize symmetry performs significantly worse than the other four methods that utilize symmetry, demonstrating that symmetry is indeed useful for evolving good controllers. The four symmetry-based 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. However, in the more challenging experiments discussed in Sections V-C and V-D, ENSO's symmetry evolution approach finds significantly better controllers than the other methods.



(a) ENSO (minimal symmetry breaking)



(b) Evolution with random symmetry breaking

Fig. 9: 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.

symmetry (Fig. 9b). In contrast, ENSO produces graphs with higher symmetry consistently because it uses group theory to break symmetry minimally (Fig. 9a). While both approaches are equally good on flat ground, they differ significantly on more challenging conditions, as will be shown in Sections V-C and V-D.

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 common quadruped gaits such as pronk, bound, and trot seen in animals. In contrast, the lack of symmetry in the direct encoding method makes it difficult to coordinate all four legs, producing ineffective gaits that utilize only a subset of the legs. Visualization videos of such behaviors can be seen at the website http://nn.cs.utexas.edu/?enso-robots.

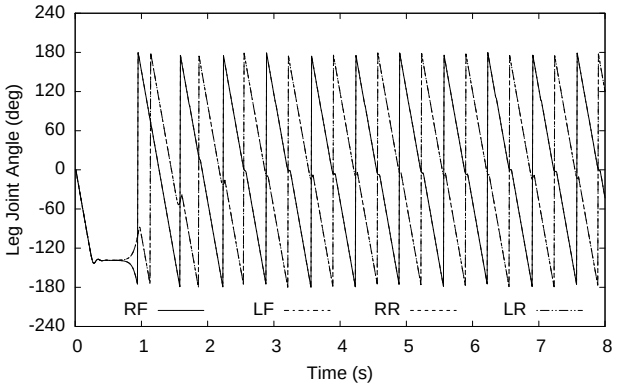
The gaits of champion networks evolved by the different methods can also be assessed by plotting the leg joint angles of the robot as functions of time. Fig. 10 shows typical plots for the first eight seconds of simulated time. Initially, all legs are in the same angular position and they remain synchronous when they start moving. For gaits such as bound and trot that have pairs of legs moving half-period out of phase, this phase difference emerges early on. Thereafter, the controllers maintain synchronicity and phase relations between the legs. Well-coordinated gaits result for ENSO and the two fixed symmetry methods. However, random symmetry evolution typically produce gaits that have the following two flaws: (1) legs are not well synchronized (e.g. the two rear legs in Fig. 10b) and (2) phase difference between legs does not divide the period evenly (e.g. phase difference between the front and rear leg pairs in Fig. 10b). The resulting weak coordination of the legs produces the stumbling effect mentioned above, and seen in the videos.

To sum up, controllers with symmetry produce significantly more effective gaits than controllers without symmetry. Although all symmetry-based methods produce gaits that are equally effective, ENSO's solutions are more symmetric and more smooth. Such a bias is a major advantage in more challenging problems, as demonstrated in the next two experiments.

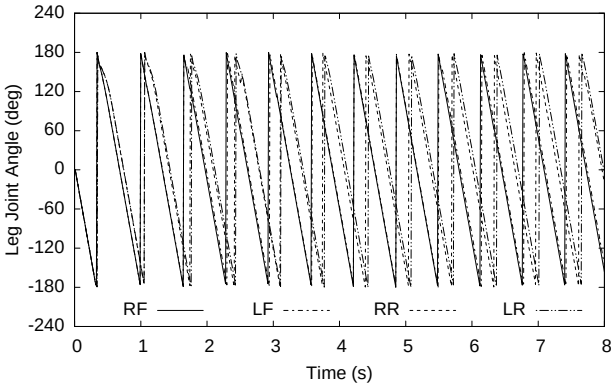
C. Inclined Ground

In the second experiment, 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.

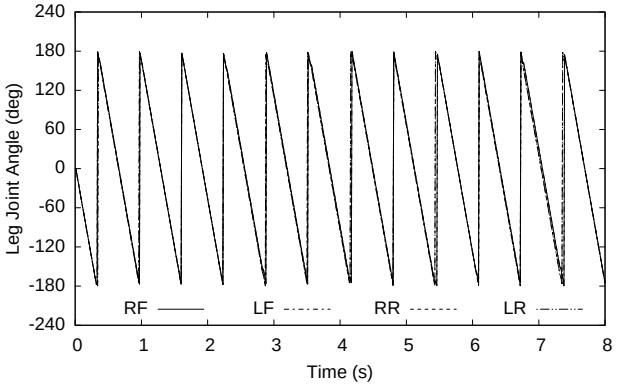
Since the robot is the same as before with no morphological changes, the same hand-designed symmetries should apply. However, when the robot is on inclined ground, the direction of gravity is not aligned with its plane of symmetry, thus breaking the symmetry of its dynamics in a way that is difficult for a human designer to take into account. As a result, the appropriate controller symmetries for this task are expected



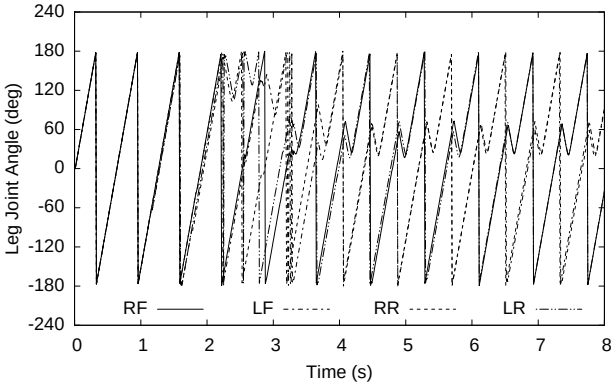
(a) ENSO (bound gait)



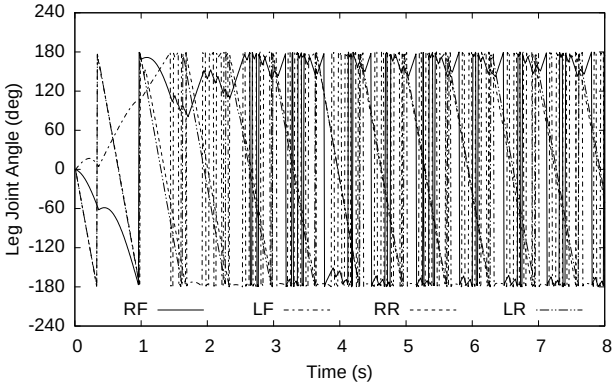
(b) Random symmetry evolution (gait resembling bound)



(c) Evolution with fixed $\mathcal{S}_4$ symmetry (pronk gait)



(d) Evolution with fixed $\mathcal{D}_2$ symmetry (trot gait)



(e) Direct encoding (gait utilizing only rear legs)

Fig. 10: Example gaits of champion networks evolved by the different methods on flat ground. The graphs show the joint angles of the four legs of the robot in the first eight seconds of simulated time. The plot for ENSO was produced using the left phenotype graph in Fig. 9a and that for random symmetry evolution was produced using the middle phenotype graph in Fig. 9b. When the controllers reach a steady state, they maintain synchronicity and phase relations between the legs. The controllers evolved by the direct encoding method utilize only a subset of the legs, producing ineffective gaits. For example, only the rear legs are utilized in the gait plotted in (e); the front legs just vibrate and remain non-functional. In contrast, the controllers evolved by the symmetry-based methods utilize all four legs effectively. Moreover, ENSO and the two fixed symmetry methods evolve controllers that produce the well-defined gaits illustrated in Fig. 4. However, random symmetry evolution typically evolves controllers that produce lesser-quality gaits. For example, in the bound-like gait of (b), the rear legs are poorly synchronized and the phase difference between the front and rear leg pairs is less than half-period. As a result, this gait is not as smooth as the gaits produced by ENSO and the fixed symmetry methods. Such smoothness is a major advantage in more challenging environments, as seen in Fig. 13.

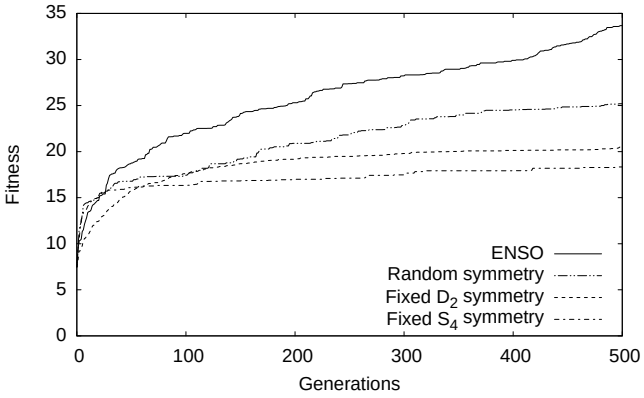
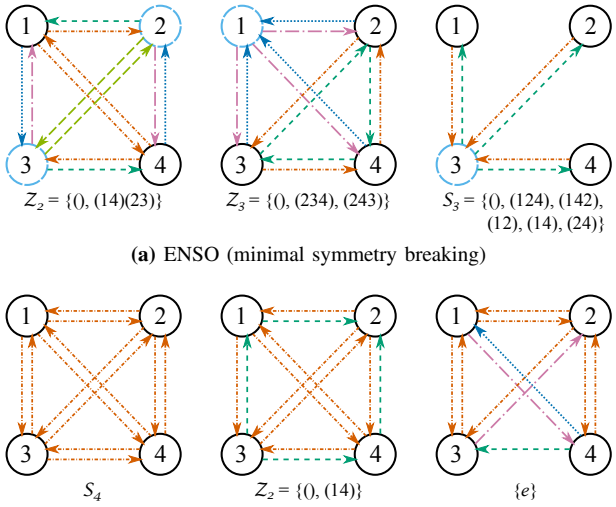


Fig. 11: Performance of quadruped controllers evolved using various methods on inclined ground. The curve for each method shows the average fitness of champion controllers in ten trials of evolution. Both symmetry evolution methods perform better than the hand-designed symmetries because the best symmetries for inclined ground are difficult for humans to conceive. Moreover, the systematic group-theoretic search approach of ENSO finds significantly better symmetries than the random search approach.



(a) ENSO (minimal symmetry breaking)

(b) Evolution with random symmetry breaking

Fig. 12: 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 the graphs on flat ground (Fig. 9a). 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 (Fig. 9b), random symmetry evolution produces many graphs with the initial $\mathcal{S}_4$ symmetry, which produces 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.

to be different from those needed 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.

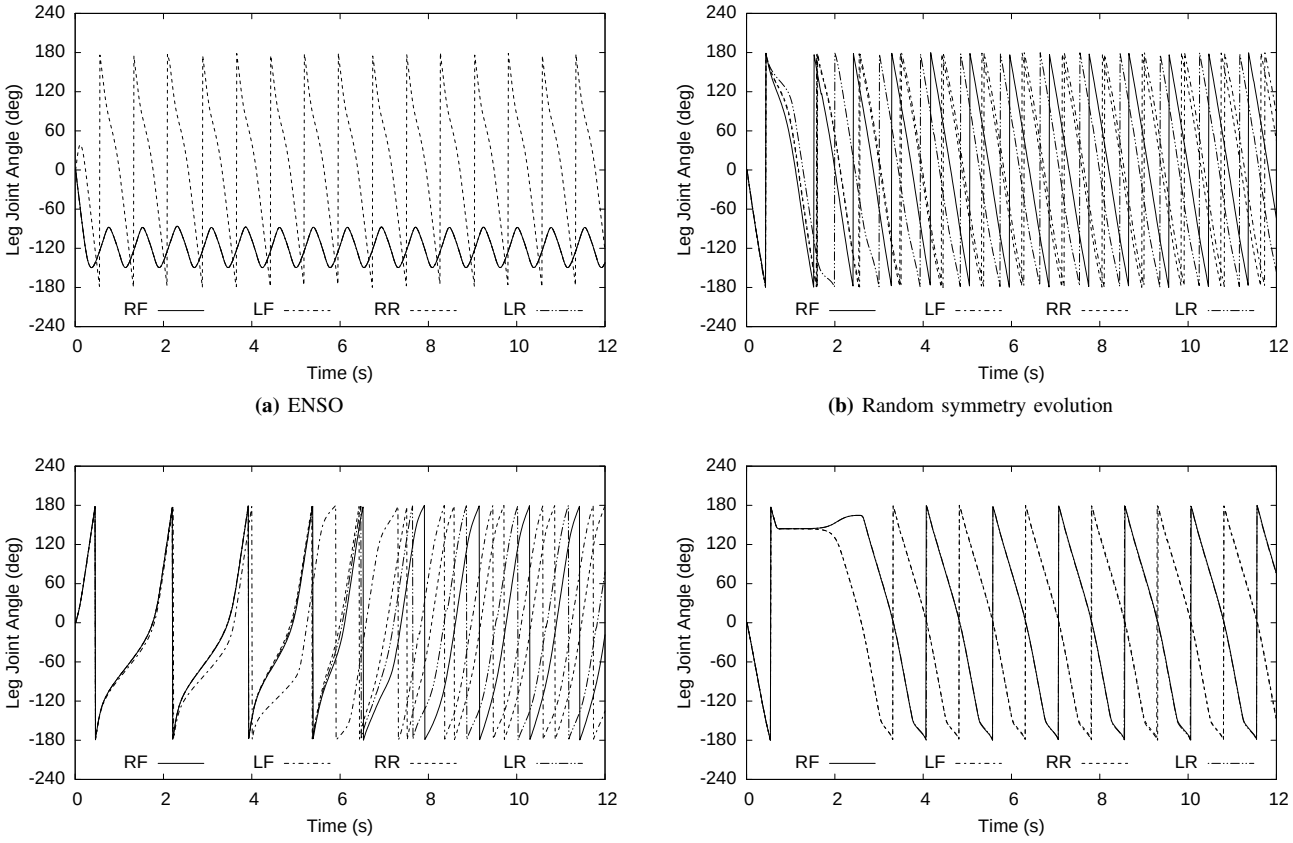
The results of these experiments are shown in Fig. 11. ENSO produces significantly better fitness than random symmetry evolution (according to the Student's t-test, with $p < 0.002, df = 18$), which in turn produces significantly better fitness than evolution of fixed $\mathcal{D}_2$ symmetry ($p < 0.04, df = 18$). 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, the results 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 (Fig. 12a) are often less symmetric than those it evolves on flat ground (Fig. 9a). 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 (Fig. 12b). 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.

Fig. 13 illustrates the above observations by plotting leg angles of typical evolved controllers. The controller evolved by ENSO generates two types of waveforms, each corresponding to a different type of module and representing a different leg behavior. The first module type controls only the right rear leg, which powers the robot's forward motion. The second module type controls all the other legs and helps maintain the robot's orientation. The controllers for the other three methods have only one type of waveform because all legs are controlled by the same type of module. As on flat ground, the gaits evolved by random symmetry mutations are less regular than those evolved by the other methods. The controller evolved with fixed $\mathcal{S}_4$ symmetry implements a gait similar to the walk, i.e. legs are quarter-period separated in phase, while that evolved with fixed $\mathcal{D}_2$ symmetry implements a trot gait. Comparing the periods of the plotted gaits indicates that the gait evolved by ENSO is faster than the other gaits.

Thus ENSO evolves more effective gaits on inclined ground than the other methods by finding symmetries that are better suited to the incline. Since ENSO adapts the gaits better to the environment, they should generalize better to changes in



(a) ENSO

(b) Random symmetry evolution

(c) Evolution with fixed $\mathcal{S}_4$ symmetry

(d) Evolution with fixed $\mathcal{D}_2$ symmetry (trot gait)

Fig. 13: Example gaits of champion networks evolved by the different methods on inclined ground. The graphs show the joint angles of the four legs of the robot in the first twelve seconds of simulated time. (a) The plot for ENSO was produced using the middle phenotype graph in Fig. 12a, and it shows the two waveforms corresponding to the two types of modules in the phenotype graph. (b) The plot for random symmetry evolution was produced using the middle phenotype graph in Fig. 12b, and it shows that this gait is less regular than the gaits produced by the other methods. Plots (c) and (d) show that evolving with fixed $\mathcal{S}_4$ and $\mathcal{D}_2$ produce the same gaits as on flat ground. Comparing the gait periods demonstrates that the gait evolved with ENSO is faster than the other gaits.

the environment as well. Generalization of the champion controllers evolved by the different methods on inclined ground were tested by reducing the friction coefficient of the ground by 25%; all other simulation parameters remained the same. Making the ground slippery in this manner makes it harder for the robots to maintain their balance and orientation as they walk across the incline. However, controllers that evolved effective gaits for the incline should be able to maintain acceptable performance.

Since the controllers evolved with fixed $\mathcal{S}_4$ and $\mathcal{D}_2$ symmetries and random symmetry mutations are ill-suited for inclines, the robot often flipped over or moved downhill instead of walking across the incline, i.e. such controllers generalize poorly to slippery inclines. In contrast, the controllers that ENSO evolved make the robot walk nearly straight across the incline, slipping only a little. This behavior is possible because ENSO optimizes controllers to perform well on inclines, resulting in gaits that also generalize better. Generalizing in this manner is crucial for transferring the controllers evolved in simulation to physical robots since the real world cannot be simulated with perfect accuracy.

Multilegged robots developed for real-world applications are more complex than the quadruped model (Fig. 6) used in the first two experiments. For example, they may have more than one joint per leg to make it possible to produce more flexible gaits. ENSO scales up to such larger problems and evolves better controllers than the other symmetry-based methods, as demonstrated in the next experiment.

D. Quadruped with Knees

In the third experiment, the quadruped robot used in the first two experiments was extended with knee joints (Fig. 14). Each hip joint now allowed the leg to swing only forward and backward, up to 30° in each direction. Similarly, each knee joint allowed the lower part of the leg to swing only forward up to 30° with respect to its upper part. Restricting joint movement in this manner makes it possible to model the locomotion of many quadruped animals with knees. The controllers for this robot consist of eight interconnected modules, each controlling the movement of a different joint to produce gaits. Fitness was measured in the same way as in the previous experiment on inclined ground, thus biasing evolution to produce straight gaits.

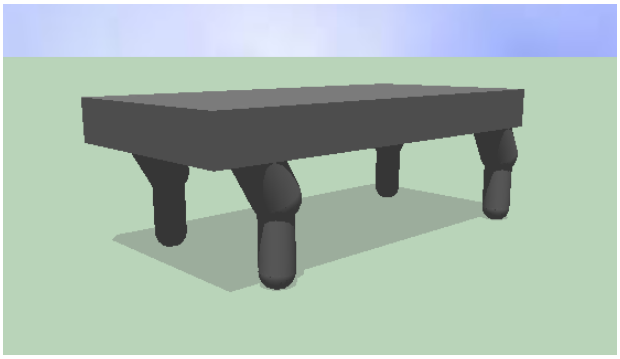
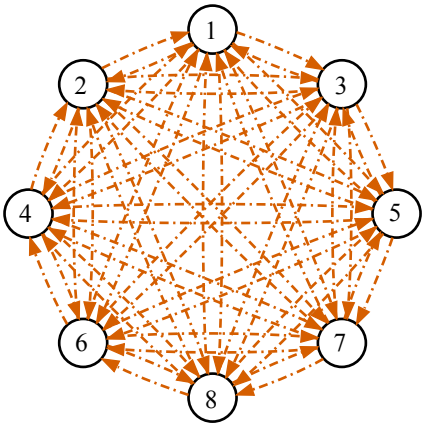


Fig. 14: The quadruped robot model with knees. This model extends the quadruped in Fig. 6 by adding knee joints. The hip joints allow the legs to swing forward and backward up to 30° in each direction. The knee joints allow the lower part of the legs to swing only forward (up to 30°) with respect to the upper part. This model provides a more challenging platform to evaluate how well ENSO scales up to larger problems with more modules.

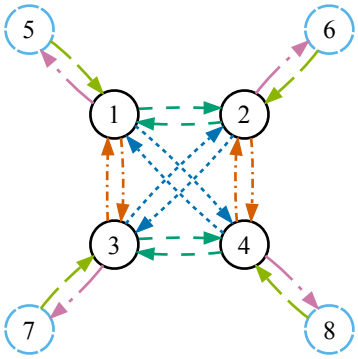
Since the controllers have eight modules, ENSO and random symmetry evolution are both initialized with the symmetry group $\mathcal{S}_8$, corresponding to maximally symmetric phenotype graphs (Fig. 15a). Therefore, the fixed-symmetry method that utilized $\mathcal{S}_4$ symmetry in the first two experiments was modified correspondingly to utilize $\mathcal{S}_8$ symmetry instead. In addition, the hand-designed $\mathcal{D}_2$ symmetry used in those experiments was extended by attaching a knee module to each hip module, thus preserving the original $\mathcal{D}_2$ symmetry (Fig. 15b). The resulting phenotype graph reflects the physical connectivity of corresponding joints. Moreover, the knee and hip modules are of different types, making it possible to evolve different hip and knee control functions if necessary. Thus, this controller symmetry is designed to make it possible to evolve effective quadruped gaits similar to those in Section V-B.

However, as illustrated in Fig. 16, the controllers that ENSO evolved perform significantly better than the controllers evolved with hand-designed $\mathcal{D}_2$ symmetry (according to the Student's t-test, with $p < 0.04, df = 18$). This result demonstrates that ENSO scales well to larger problems. Although $\mathcal{D}_2$ symmetry produces controllers with slightly better average fitness than $\mathcal{D}_8$ symmetry, their differences are not statistically significant ($p > 0.28, df = 18$).

The differences in fitness between ENSO and random symmetry evolution are not statistically significant in this experiment ($p > 0.40, df = 18$). However, as in the previous experiments, ENSO evolves more symmetric phenotype graphs than random symmetry evolution (Fig. 17), resulting in smoother and more regular gaits. These differences in gaits can be observed both in the simulation videos and in the example plots of hip joint angles in Fig. 18. ENSO produces effective trot gaits, which are likely to perform better in the real world than the irregular gaits of random symmetry evolution. Fixed $\mathcal{D}_2$ symmetry can also produce regular gaits, but they are typically less effective and are produced less often. Similarly, evolution with fixed $\mathcal{S}_8$ symmetry produces ineffective pronk gaits that move only a small distance. Thus, ENSO evolves significantly better controllers than the other methods in this challenging experiment.



(a) $\mathcal{S}_8$ symmetry



(b) $\mathcal{D}_2$ symmetry

Fig. 15: Phenotype graphs used in fixed-symmetry evolution of controllers for the quadruped with knees. These controllers have eight modules, each controlling a different leg joint. (a) The maximally symmetric controller has identical connections between eight identical modules, i.e. it has $\mathcal{S}_8$ symmetry. (b) The previous controller with hand-designed $\mathcal{D}_2$ symmetry was extended by attaching a knee module to each hip module, preserving the original $\mathcal{D}_2$ symmetry. Controller evolution with these two fixed symmetries replaces the corresponding control methods for the quadruped without knees, making it possible to evaluate how ENSO scales up relative to hand-design.

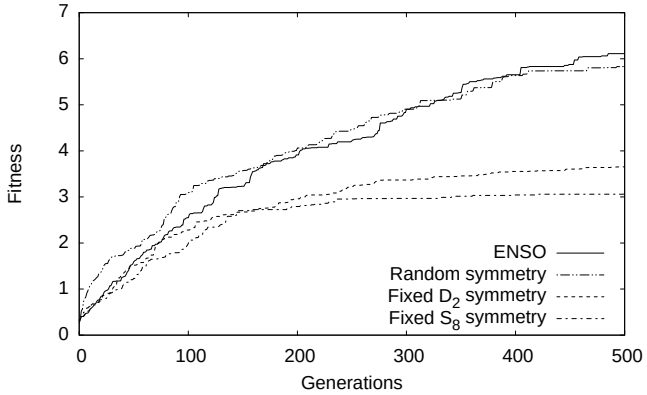
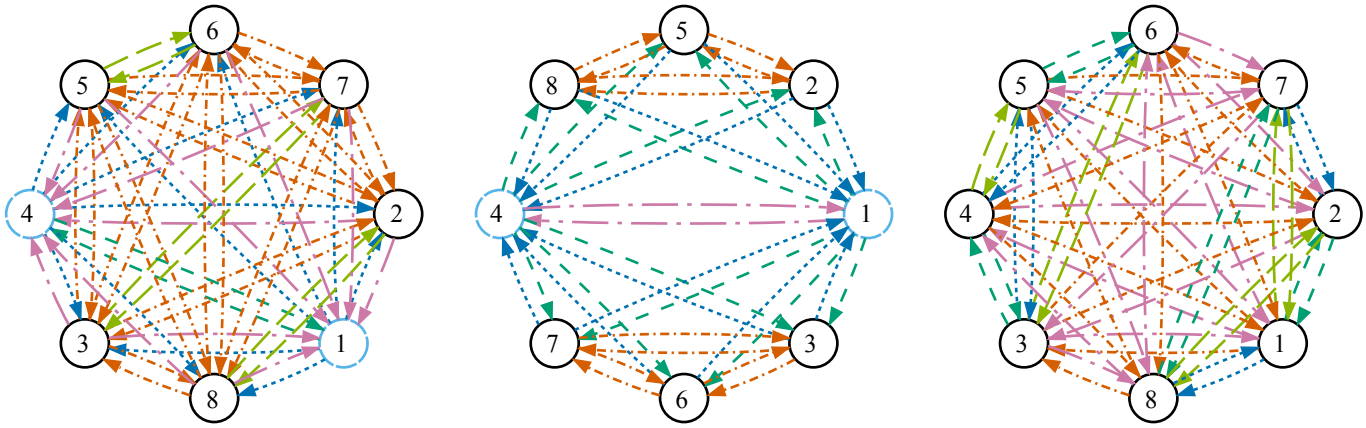
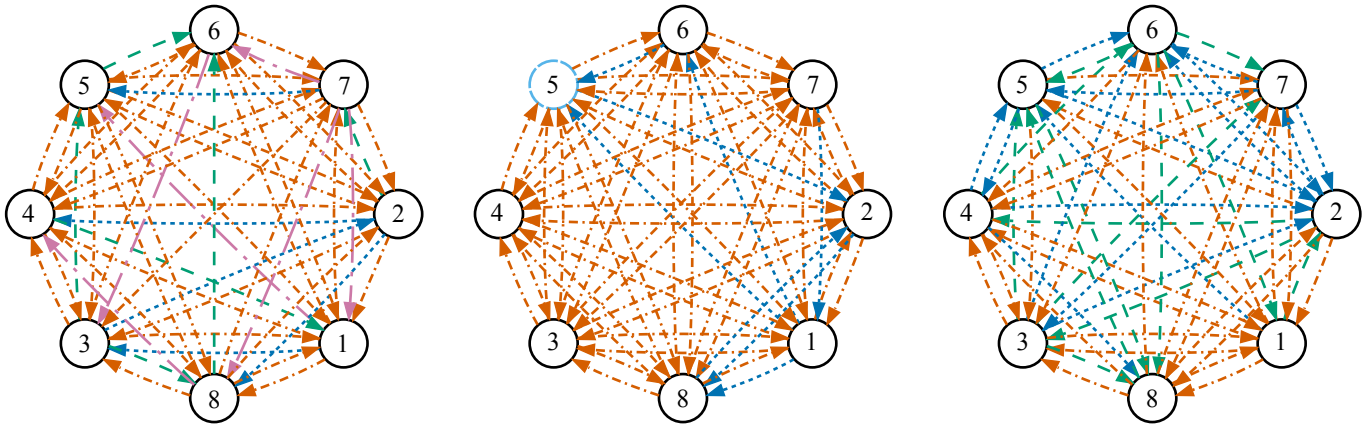


Fig. 16: Performance of quadruped controllers evolved using various methods on flat ground for a quadruped with knees. The curve for each method shows the average fitness of champion controllers in ten trials of evolution. As in the experiment depicted in Fig. 11, the appropriate controller symmetries are difficult to design by hand. As a result, both methods that utilize fixed, hand-designed symmetries perform significantly worse than the symmetry evolution methods that discover symmetries automatically. ENSO and random symmetry evolution perform similarly in terms of average fitness. However, ENSO's gaits are significantly more regular (Fig. 18) and are therefore likely to perform better in the real world.



(a) ENSO (minimal symmetry breaking)



(b) Evolution with random symmetry breaking

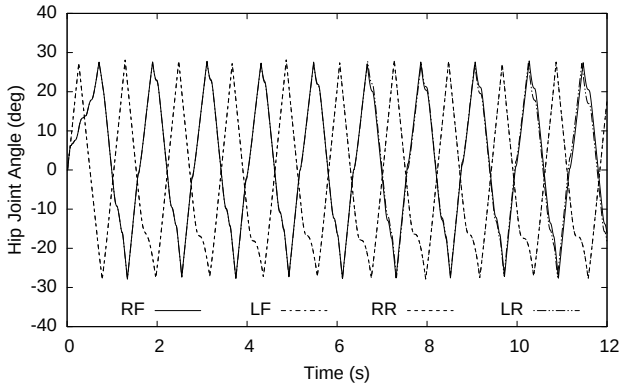
Fig. 17: Phenotype graphs of typical champion networks evolved by ENSO and random symmetry evolution for the quadruped with knees. (a) ENSO's group-theoretic symmetry mutations produce symmetric connections between modules. New types of modules that they create are also often symmetric. For example, the diagonal hips 1 and 4 in both the left and middle graphs have matching modules, producing effective trot gaits as a result. (b) In contrast, random symmetry evolution produces graphs with asymmetric connections between modules. Moreover, it creates new types of modules without symmetric counterparts (e.g. middle graph). The resulting gaits are therefore less regular than those produced by ENSO, as illustrated in Fig. 18.

Together, these results demonstrate that utilizing the systematic symmetry search of ENSO focuses the search on better solutions, making it possible to find significantly more effective controllers than by utilizing random symmetry search or by designing the symmetry by hand. ENSO is therefore a promising approach for evolving distributed controllers for complex tasks, and in general, for designing complex modular systems with symmetries, as will be discussed next.

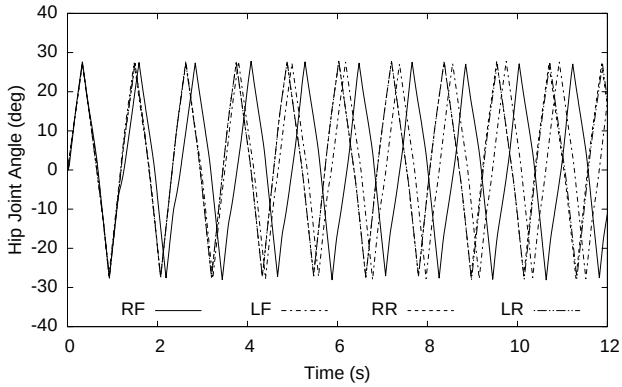
VI. Discussion and Future Work

In the above experiments, controller symmetries constrain the type of module controlling each leg and the type of connection between each pair of modules, producing good leg coordination. It is difficult to evolve similarly good gaits with direct encoding because it does not bias evolutionary search with the appropriate symmetry constraints. Nevertheless, direct encoding can be advantageous in special cases where the appropriate symmetries for the task are close to the bottom of the subgroup lattice and can be approximated easily by evolving controllers without symmetry. For example, in supplementary experiments direct encoding performed better than ENSO for the task in Section V-C of evolving controllers for a quadruped robot with hinge joints to walk on inclined ground. However, ENSO regained its advantage over direct encoding in the same task when the robot was made more complex by replacing its hinge joints (which have only one degree of freedom) with universal joints (which have two degrees of freedom). The resulting controller has more parameters, making it more difficult for direct encoding to optimize in a way that approximates the appropriate symmetries. For the same reason, ENSO also performed significantly better than direct encoding for the task in Section V-D of evolving controllers for a quadruped with knees to walk on flat ground. Determining the appropriate level of symmetry is a main advantage of ENSO that allows it to be applied robustly to a variety of problems.

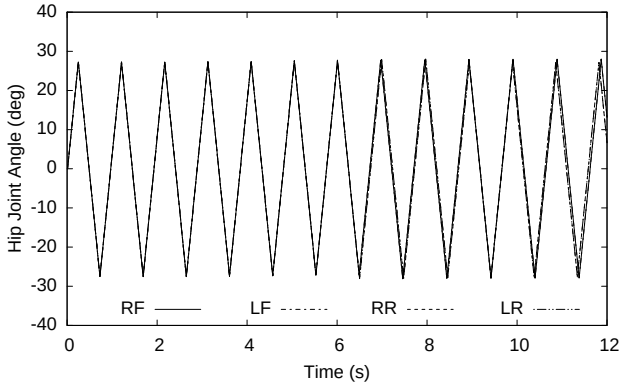
An alternative to evolving the appropriate controller symmetries with ENSO is to design them by hand. A human designer can determine them analytically in simple cases such for as a quadruped robot on flat ground, but cannot (at least not as easily) do so for more complex robots and real-world environments with inclines and other complexities. Given the number of modules, ENSO's group-theory-based systematic symmetry



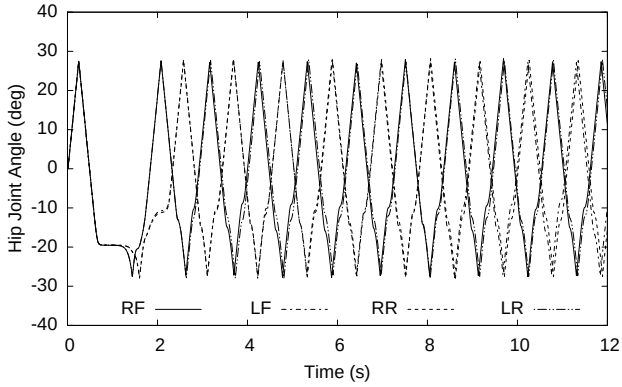
(a) ENSO (trot gait)



(b) Random symmetry evolution (irregular gait)



(c) Evolution with fixed $\mathcal{S}_8$ symmetry (ineffective pronk gait)



(d) Evolution with fixed $\mathcal{D}_2$ symmetry (trot gait)

Fig. 18: Example gaits of champion networks evolved by the different methods for the quadruped with knees. The graphs show the four hip joint angles of the robot in the first twelve seconds of simulated time. (a) The plot for ENSO was produced using the left phenotype graph in Fig. 17a, and it shows a regular trot gait. (b) The plot for random symmetry evolution was produced using the middle phenotype graph in Fig. 17b, and it shows a less regular gait with uneven phase differences between the legs. (c) Evolving with a fixed $\mathcal{S}_8$ symmetry produces pronk gaits that are ineffective and move only a small distance as a result. (d) Evolving with the hand-designed $\mathcal{D}_2$ symmetry also produces regular gaits, but they are typically less effective than ENSO's gaits and are produced less often, thus demonstrating the advantage of evolving symmetries with ENSO.

search makes it possible to evolve the appropriate symmetries automatically. Demonstrating this capability, ENSO evolved gaits similar to those based on hand-designed symmetries on flat ground, and significantly faster gaits on inclined ground. The gaits on the incline also generalized better when friction was reduced to make the ground slippery.

In order to verify that ENSO's symmetry-breaking approach is indeed a useful way to evolve symmetry, it was compared with the less principled, random-symmetry-evolution approach. Random symmetry evolution produces significantly worse gaits than ENSO because its unsystematic symmetry mutations often result in large changes in symmetry. In contrast, the group-theoretic mutations of ENSO result in only minimal changes in symmetry, making complexification possible (Section II-A): Evolution starts with a highly symmetric controller and breaks the symmetry incrementally, producing gradual increases in complexity. As a result, evolution optimizes simpler controllers before elaborating on them by adding more parameters. Thus the complexification resulting from symmetry breaking provides ENSO with a smoother fitness gradient, making evolution easier.

In addition to these evolutionary advantages that group theory provides, the theory of coupled cell systems provides ENSO with theoretical guarantees on the behavior of evolved controllers. For example, if the symmetries of a coupled cell system admit a particular gait, then there exists an instance of the system with an asymptotically stable periodic solution (i.e. limit cycle) implementing that gait [18]. As a result, the gait is robust to small perturbations. ENSO uses neuroevolution to find such an instance of the control system. When the dynamics of this system was perturbed manually (utilizing a visualization interface to the physical simulation), the gait it generates was indeed found to be robust in eight of the ten controllers that we perturbed. Such robustness is useful in controllers for real-world robots because their interactions with environment are frequently perturbed.

As discussed in Section II-D, the theory of coupled cell systems also predicts that the same controller can produce multiple gaits. Each gait is a stable limit cycle of the coupled cell system, and the system can transition from one limit cycle to another when perturbed. Sometimes small perturbations caused by physical interaction with the environment can trigger such a transition. Fig. 10d shows an example of this

phenomenon, where the robot changes from an initial pronk to a trot at about three seconds into the simulation. If the robot is prevented from interacting with the ground, then this change does not occur and the robot continues executing the pronk gait. The ability of controllers to generate such different gaits makes it possible to use the most effective gait for a given terrain, including going over obstacles [57]. Alternatively, desired gaits can be evolved by utilizing an external sinusoidal waveform [10] or by parameterizing the gaits [23, 30].

The above results suggest that ENSO is an effective approach for evolving locomotion controllers for multilegged robots. In the future, ENSO will be tested with more complex robots with more legs, more complex legs, and sensors that receive more varied stimuli from the environment. Their controllers can be modeled as coupled cell systems with more cells and cells receiving additional inputs, as was done for fixed symmetry controllers in prior work [57]. Using such sophisticated 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. ENSO is thus 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 [55]. 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 [37]. 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.

Besides controllers for multilegged robots, coupled cell systems can also model other dynamical systems in nature, producing various types of symmetric networks that ENSO can potentially optimize. For example, they have been used to study the formation of new species in nature [18], the role of structural symmetries of the visual cortex in inducing visual hallucinations [17], and the properties of genetic regulatory networks [13]. ENSO can potentially be used as a tool both for understanding such biological phenomena and for engineering artificial systems based on them.

In addition to using ENSO in various applications, the ENSO approach itself can be extended in several 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 [53]. Fourth, crossover of genotype trees can be implemented by swapping subtrees of parent trees if those subtrees have the same structure and node colors. Fifth, although only the leaf nodes of the genotype tree represent the phenotype graph in the current implementation, an even more compact representation is possible. The child nodes could inherit one or more parameter values from their parent instead of specifying those parameter values in each child node. This feature is useful for representing variations of similar elements compactly, a common theme in complex systems with regularities [50, 52]. Such elements can be constructed from leaf nodes that inherit some parameters from a common parent, but specify different values for other parameters in the leaf nodes. As a result, these elements have the same values for parameters inherited from the parent, while they differ in the values for parameters specified in the leaf nodes. These extensions would improve evolutionary search, potentially making it possible for ENSO to solve more difficult problems and a wider variety of problems.

VII. Conclusion

Based on properties that make development effective, this paper proposes a novel evolutionary algorithm, ENSO, to design complex modular systems. ENSO utilizes group theory to search for symmetry systematically. As a result, evolution progresses from simple, highly symmetric phenotypes to more complex, less symmetric phenotypes. This complexification gradually increases the variety of modules and their interconnections in the phenotype, making evolutionary search effective. In three experiments, symmetry was first shown to be an effective principle in general, and ENSO then shown to result in faster and more elegant solutions than alternative methods of utilizing symmetry. The same approach can potentially be used in other applications as well, suggesting that it is a useful method for solving complex modular problems in the real world.

Acknowledgment

This paper was supported in part by NSF under grants DBI-0939454, IIS-0915038, IIS-0757479, and EIA-0303609, by the Texas Higher Education Coordinating Board grant 003658-0036-2007, and by a Google Research Award.

References

[1] A. Amores, A. Force, Y.-L. Yan, L. Joly, C. Amemiya, A. Fritz, R. K. Ho, J. Langeland, V. Prince, Y.-L. Wang, M. Westerfield, M. Ekker, and J. H. Postlethwait, "Zebrafish HOX clusters and vertebrate genome evolution," *Science*, vol. 282, pp. 1711–1784, 1998.

[2] O. Bastert, "Stabilization procedures and applications," Ph.D. dissertation, Technische Universität Müchen, 2001.

[3] 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*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1989, pp. 577–585.

[4] L. Beineke, R. Wilson, and P. Cameron, "Introduction," in *Topics in Algebraic Graph Theory*, L. W. Beineke and R. J. Wilson, Eds. New York, NY, USA: Cambridge University Press, 2004, pp. 1–29.

[5] E. J. W. Boers and H. Kuiper, "Biological metaphors and the design of modular artificial neural networks," Master's thesis, Departments of Computer Science and Experimental and Theoretical Psychology at Leiden University, The Netherlands, 1992. [Online]. Available: http://citeseer.nj.nec.com/boers92biological.html

[6] J. C. Bongard and R. Pfeifer, "Repeated structure and dissociation of genotypic and phenotypic complexity in artificial ontogeny," in *Proceedings of the Genetic and Evolutionary Computation Conference*, L. Spector, E. D. Goodman, A. Wu, W. B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon, and E. Burke, Eds. San Francisco: Morgan Kaufmann, 2001, pp. 829–836. [Online]. Available: http://www-illigal.ge.uiuc.edu:8080/gecco-2001/

[7] A. Cangelosi, D. Parisi, and S. Nolfi, "Cell division and migration in a 'genotype' for neural networks," *Network: Computation in Neural Systems*, vol. 5, pp. 497–515, 1994. [Online]. Available: http://kant.irmkant.rm.cnr.it/econets/cangelosi.migration.ps.Z

[8] A. Chan and C. Godsil, "Symmetry and eigenvectors," in *Graph Symmetry: Algebraic Methods and Applications*, G. Hahn and G. Sabidussi, Eds. Springer, 1997, pp. 75–106.

[9] Y. Chauvin and D. E. Rumelhart, Eds., *Backpropagation: Theory, Architectures, and Applications*. Hillsdale, NJ: Erlbaum, 1995.

[10] J. Clune, B. E. Beckmann, C. Ofria, and R. T. Pennock, "Evolving coordinated quadruped gaits with the HyperNEAT generative encoding," in *Proceedings of the Eleventh conference on Congress on Evolutionary Computation (CEC'09)*. Piscataway, NJ, USA: IEEE Press, 2009, pp. 2764–2771.

[11] J. J. Collins and I. N. Stewart, "Coupled nonlinear oscillators and the symmetries of animal gaits," *Journal of Nonlinear Science*, vol. 3, no. 1, pp. 349–392, 1993.

[12] F. Dellaert and R. D. Beer, "A developmental model for the evolution of complete autonomous agents," in *From Animals to Animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior*, P. Maes, M. J. Mataric, J.-A. Meyer, J. Pollack, and S. W. Wilson, Eds. Cambridge, MA: MIT Press, 1996. [Online]. Available: http://citeseer.nj.nec.com/dellaert96developmental.html

[13] R. Edwards and L. Glass, "Combinatorial explosion in model gene networks," *Chaos*, vol. 10, pp. 691–704, 2000.

[14] S. E. Fahlman and C. Lebiere, "The cascade-correlation learning architecture," in *Advances in Neural Information Processing Systems 2*, D. S. Touretzky, Ed. San Francisco: Morgan Kaufmann, 1990, pp. 524–532.

[15] "GAP – groups, algorithms, and programming," 2007, http://www.gap-system.org. [Online]. Available: http://www.gap-system.org

[16] A. Garcia-Bellido, "Symmetries throughout organic evolution," *PNAS*, vol. 93, no. 25, pp. 14 229–14 232, December 1996.

[17] M. Golubitsky, L. J. Shiau, and A. Török, "Bifurcation on the visual cortex with weakly anisotropic lateral coupling," *SIAM Journal on Applied Dynamical Systems*, vol. 2, no. 2, pp. 97–143, 2003.

[18] M. Golubitsky and I. Stewart, "Patterns of oscillation in coupled cell systems," in *Geometry, Mechanics, and Dynamics: Volume in Honor of the 60th Birthday of J. E. Marsden*, P. Newton, P. Holmes, and A. Weinstein, Eds. Springer, 2002, ch. 8, pp. 243–286.

[19] F. Gruau, "Neural network synthesis using cellular encoding and the genetic algorithm," Ph.D. dissertation, Ecole Normale Superieure de Lyon, France, 1994. [Online]. Available: http://citeseer.nj.nec.com/frederic94neural.html

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

[21] F. Heylighen, "The growth of structural and functional complexity during evolution," in *The Evolution of Complexity: The Violet Book of 'Einstein Meets Magritte'*, F. Heylighen, J. Bollen, A. Riegler, and A. Riegler, Eds. Springer, 1999, ch. 2, pp. 17–44.

[22] J. H. Holland, "Genetic algorithms," *Scientific American*, vol. 267, no. 1, pp. 66–72, July 1992. [Online]. Available: http://www2.econ.iastate.edu/tesfatsi/holland.gaintro.htm

[23] G. S. Hornby, S. Takamura, J. Yokono, O. Hanagata, T. Yamamoto, and M. Fujita, "Evolving robust gaits with AIBO," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 3, 2000, pp. 3040–3045.

[24] G. S. Hornby and J. B. Pollack, "Creating high-level components with a generative representation for body-brain evolution," *Artificial Life*, vol. 8, no. 3, 2002. [Online]. Available: http://www.demo.cs.brandeis.edu/papers/hornby_alife02.pdf

[25] 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.

[26] S. A. Kauffman, *The Origins of Order*. New York: Oxford University Press, 1993.

[27] H. Kimura, S. Akiyama, and K. Sakurama, "Realization of dynamic walking and running of the quadruped using neural oscillator," *Autonomous Robots*, vol. 7, no. 3, pp. 247–258, 1999.

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

[29] J. Kodjabachian and J.-A. Meyer, "Evolution and development of modular control architectures for 1D locomotion in six-legged animats," *Connection Science*, vol. 10, pp. 211–237, 1998.

[30] N. Kohl and P. Stone, "Machine learning for fast quadrupedal locomotion," in *Nineteenth National Conference on Artificial Intelligence*, 2004.

[31] J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo, Eds., *Genetic Programming 1996*. Cambridge, MA: MIT Press, 1996.

[32] S. Lall and N. Patel, "Conservation and divergence in molecular mechanisms of axis formation," *Annual Review of Genetics*, vol. 35, pp. 407–447, 2001.

[33] Y. le Cun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *Advances in Neural Information Processing Systems 2*, D. S. Touretzky, Ed. San Francisco: Morgan Kaufmann, 1990, pp. 598–605.

[34] A. Lindenmayer, "Mathematical models for cellular interaction in development parts I and II," *Journal of Theoretical Biology*, vol. 18, pp. 280–299 and 300–315, 1968.

[35] S. Luke and L. Spector, "Evolving graphs and networks with edge encoding: Preliminary report," in *Late-Breaking Papers of Genetic Programming 1996*, J. R. Koza, Ed. Stanford Bookstore, 1996. [Online]. Available: http://citeseer.nj.nec.com/128010.html

[36] M. Q. Martindale and J. Q. Henry, "The development of radial and biradial symmetry: The evolution of bilaterality," *American Zoologist*, vol. 38, no. 4, pp. 672–684, September 1998.

[37] D. McFarlane, "Modular distributed manufacturing systems and the implications for integrated control," in *IEE Colloquium on Choosing the Right Control Structure for Your Process (Digest No. 1998/280)*, March 1998.

[38] J. F. Miller, "Evolving a self-repairing, self-regulating, French flag organism," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2004)*. Berlin: Springer Verlag, 2004. [Online]. Available: http://www.elec.york.ac.uk/intsys/users/jfm7/gecco2004.pdf

[39] "ODE: Open dynamics engine," 2007, http://www.ode.org/. [Online]. Available: http://www.ode.org/

[40] "OGRE: Object-oriented graphics rendering engine," 2007, http://www.ogre3d.org/. [Online]. Available: http://www.ogre3d.org/

[41] "OPAL: Open physics abstraction layer," 2007, http://opal.sourceforge.net/. [Online]. Available: http://opal.sourceforge.net/

[42] "Open BEAGLE," 2007, http://beagle.gel.ulaval.ca/. [Online]. Available: http://beagle.gel.ulaval.ca/

[43] A. R. Palmer, "Symmetry breaking and the evolution of development," *Science*, vol. 306, pp. 828–833, 2004.

[44] C. M. A. Pinto and M. Golubitsky, "Central pattern generators for bipedal locomotion," *Journal of Mathematical Biology*, vol. 53, no. 3, pp. 474–489, 2006.

[45] L. Righetti and A. J. Ijspeert, "Pattern generators with sensory feedback for the control of quadruped locomotion," in *Proceedings of the 2008 IEEE International Conference on Robotics and Automation (ICRA 2008)*, May 2008, pp. 819–824.

[46] K. Seo and J. J. E. Slotine, "Models for global synchronization in CPG-based locomotion," in *Proceedings of the 2007 IEEE International Conference on Robotics and Automation*, April 2007, pp. 281–286.

[47] S. V. Shastri, "A biologically consistent model of legged locomotion gaits," *Biological Cybernetics*, vol. 76, no. 6, pp. 429–440, 1997.

[48] N. T. Siebel and G. Sommer, "Evolutionary reinforcement learning of artificial neural networks," *International Journal of Hybrid Intelligent Systems*, vol. 4, no. 3, pp. 171–183, 2007.

[49] K. Sims, "Evolving 3D morphology and behavior by competition," in *Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems (Artificial Life IV)*, R. A. Brooks and P. Maes, Eds. Cambridge, MA: MIT Press, 1994, pp. 28–39. [Online]. Available: http://www.mpi-sb.mpg.de/services/library/proceedings/contents/alife94.html

[50] K. Stanley, "Compositional pattern producing networks: A novel abstraction of development," *Genetic Programming and Evolvable Machines*, vol. 8, no. 2, pp. 131–162, June 2007.

[51] K. O. Stanley, D. B. D'Ambrosio, and J. Gauci, "A Hypercube-Based encoding for evolving Large-Scale neural networks," *Artificial Life*, vol. 15, no. 2, pp. 185–212, Apr. 2009.

[52] K. O. Stanley and R. Miikkulainen, "A taxonomy for artificial embryogeny," *Artificial Life*, vol. 9, no. 2, pp. 93–130, 2003. [Online]. Available: http://nn.cs.utexas.edu/keyword?stanley:alife03

[53] ——, "Competitive coevolution through evolutionary complexification," *Journal of Artificial Intelligence Research*, vol. 21, pp. 63–100, 2004. [Online]. Available: http://nn.cs.utexas.edu/keyword?stanley:jair04

[54] T. Steiner, Y. Jin, and B. Sendhoff, "Vector field embryogeny," *PLoS ONE*, vol. 4, no. 12, p. e8177, 12 2009.

[55] P. Stone and M. Veloso, "Multiagent systems: A survey from a machine learning perspective," *Autonomous Robots*, vol. 8, no. 3, pp. 345–383, 2000.

[56] D. S. Touretzky, Ed., *Advances in Neural Information Processing Systems 2*. San Francisco: Morgan Kaufmann, 1990.

[57] V. K. Valsalam and R. Miikkulainen, "Modular neuroevolution for multilegged locomotion," in *Proceedings of the Genetic and Evolutionary Computation Conference GECCO 2008*. New York, NY, USA: ACM, 2008, pp. 265–272. [Online]. Available: http://nn.cs.utexas.edu/keyword?valsalam:gecco08

[58] ——, "Evolving symmetric and modular neural networks for distributed control," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO) 2009*. New York, NY, USA: ACM, 2009, pp. 731–738. [Online]. Available: http://nn.cs.utexas.edu/keyword?valsalam:gecco09