Modular Neuroevolution for Multilegged Locomotion

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

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

ABSTRACT

Legged robots are useful in tasks such as search and rescue because they can effectively navigate on rugged terrain. However, it is difficult to design controllers for them that would be stable and robust. Learning the control behavior is difficult because optimal behavior is not known, and the search space is too large for reinforcement learning and for straightforward evolution. As a solution, this paper proposes a modular approach for evolving neural network controllers for such robots. The search space is effectively reduced by exploiting symmetry in the robot morphology, and encoding it into network modules. Experiments involving physically realistic simulations of a quadruped robot produce the same symmetric gaits, such as pronk, pace, bound and trot, that are seen in quadruped animals. Moreover, the robot can transition dynamically to more effective gaits when faced with obstacles. The modular approach also scales well when the number of legs or their degrees of freedom are increased. Evolved non-modular controllers, in contrast, produce gaits resembling crippled animals that are much less effective and do not scale up as a result. Hand-designed controllers are also less effective, especially on an obstacle terrain. These results suggest that the modular approach is effective for designing robust locomotion controllers for multilegged robots.

Categories and Subject Descriptors

I.2.6–*Connectionism and Neural Nets*; I.2.9–*Propelling Mechanisms*

General Terms

Design, experimentation, performance

Keywords

Modular neuroevolution, coupled cell systems, multilegged robots, controllers, locomotion

1. INTRODUCTION

Legged robots have better mobility than wheeled robots on rugged terrain. They can step over obstacles and use footholds to navigate on broken ground. They can even be designed to walk on vertical walls or climb trees. A robot with multiple legs may also be able to tolerate failure of one or more legs and still complete its mission. For these reasons, legged robots are ideal for tasks such as search and rescue, and remote exploration.

GECCO'08, July 12–16, 2008, Atlanta, Georgia, USA.

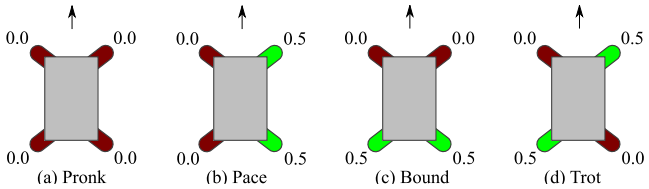


Figure 1: Phase relations between legs in the pronk, pace, bound and trot gaits of quadrupeds. The arrows show the forward direction, and the numbers as well as the colors indicate phase of leg movement. In the pronk gait, all four legs move synchronously, while in the other gaits pairs of legs are synchronous and a half-period out of phase with the other pair.

Controllers for legged robots are usually designed manually, to ensure that the legs are properly coordinated and the robot is stable. This process typically requires analyzing the sensor-motor systems and body-limb dynamics of the robot [5, 14, 16, 24], which is generally difficult and brittle because it is hard to anticipate all possible terrains. Therefore, it is desirable to automate the controller design using learning techniques. One particularly promising such technique is neuroevolution, which has been shown to perform well in various control domains such as pole balancing [11], rocket control [10], robot control [8], and agent control in games [29]. However, the control outputs in such applications have been relatively simple, and it turns out hard to scale them up to multilegged locomotion, where there are many outputs.

However, legged robots typically have a symmetric morphology that can be utilized in controller design and learning. Symmetry reduces the search space for the learning system, and results in more regular and robust controllers. Taking such symmetry into consideration, researchers have evolved modular networks of leaky integrator neurons for control of legged robots [2, 7, 12, 30].

In this paper, a potentially more powerful modular neuroevolution approach is developed based on a systematic, group-theoretical analysis of symmetry. This analysis was pioneered by Collins and Stewart [6] to explain gait symmetries in animals. In the proposed approach, it is utilized as a constraint to make controller evolution effective. Experiments in a physically realistic simulation validate the approach, resulting in modular controller networks that generate various quadruped gaits seen in nature, such as pronk, pace, bound and trot (Figure 1). Moreover, these controllers can change to a more suitable gait in response to changes in the terrain such as unexpected obstacles. The approach also generalises well to more complex robots with more legs and legs with higher degrees of freedom.

Modular evolution performs well because it can leverage the smaller search space and the symmetry in the problem: The resulting gaits

are more efficient and more symmetric than those of non-modular controllers. The modular controllers also outperform a hand-designed, open-loop controller generating a trot gait, especially on a terrain with obstacles. Altogether, these results demonstrate that the modular neuroevolution approach is effective in designing controllers for legged robots.

The paper is organized as follows. Section 2 reviews prior research on developing controllers for legged robots and on modular neuroevolution. It also reviews coupled cell systems that are used to model the modular controllers presented in this paper. A quadruped robot model and its different controller designs are discussed in Section 3. Experimental results demonstrating the advantages of the modular controllers compared to the non-modular and hand-designed controllers are presented in Section 4. Discussions of experimental observations and possible directions for future work are presented in Section 5.

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

2. BACKGROUND

Researchers have used a variety of techniques in the past to model and build controllers for legged robots. The first subsection briefly reviews some of those techniques, including manual and evolutionary approaches. Coupled cell systems having symmetry, which form the theoretical basis for the modular controller networks in this paper, are reviewed in more detail in the second subsection.

2.1 Related Work

Efforts to build legged machines began more than a century ago. Early designs required humans to control the machines, but in the 1970s computer control became a viable alternative to human control [24]. In the 1980s Marc Raibert and his coworkers built legged hopping and running machines [25]. They started with a single-legged algorithm that alternates between a support phase and a flight phase. This algorithm was then generalized to control biped running by alternating support and flight between the two legs. The same approach was then extended to quadruped gaits in which pairs of legs move in unison (in pace, bound and trot), by applying the biped algorithm to the paired legs. Thus, the symmetry of robot morphology was encapsulated as algorithmic modules even in such early work.

Rodney Brooks, another pioneer in robotics, also used modularity in his incrementally constructed controllers for a six-legged robot [4]. These controllers were completely decentralized networks of augmented finite state machines (AFSMs), some of which were repeated in the network to replicate the functionality for each leg. Each step of the incremental construction produces viable controllers for increasingly complex behaviors such as standing up, walking and following moving objects. His work also showed that robust walking behaviors can be produced by distributed sensorimotor control units with limited central coordination. The modular controllers in this paper implement the same idea: Each module produces control signals for a leg through proprioceptive sensing of joint angles without central coordination.

The distributed nature of legged locomotion has also been observed in insects. This observation inspired the distributed neural network hexapod controller hand-designed by Randall Beer and his research group [1]. The network uses leaky integrator neurons each with a different functionality such as sensing and producing rhythmic signals. The controllers produced stable gaits that are resistant to damage, such as the loss of a sensor or some connections. In another approach, they used a genetic algorithm to find parameter values for the controller network [2]. As in the modular approach presented in this paper, the evolutionary search space was shrunk by organizing the controller into subnetworks, resulting in a reduced set of parameters for evolution.

Other approaches to controller design for legged robots typically have a similar flavor, i.e. implementing controllers as continuous-time recurrent neural networks (CTRNNs) organized into distributed modules. For example, Billard and Ijspeert hand-designed CTRNN networks for controlling Aibo robot dogs [3]. Their networks, consisting of oscillator modules for each joint, were able to walk, trot and gallop. More recently, Tellez et al. evolved CTRNNs in the same task [30]. Because of the difficulty of evolving walking behaviors, network modules were evolved in stages, using more complex fitness evaluations in each successive stage. Each stage represented a different abstraction of the task, resulting in a distributed and hierarchical architecture for the controller.

Another approach to evolving modular controllers is based on indirect encoding, a method that encodes the controller as a developmental or generative program. During development of the controller, the same parts of the program may be read multiple times, once for each module instantiation. When modularity is represented intrinsically in the genotype by such means, evolution can discover modularity automatically. Gruau demonstrated this capability by evolving CTRNN controllers for hexapod locomotion using his cellular encoding (CE) method [12, 13]. Subsequently, a version of CE called simple geometry-oriented cellular encoding (SGOCE) was used by Filliat et al. to evolve CTRNN controllers incrementally for a hexapod, although their scheme requires that the precursor cells for module subnetworks are specified explicitly [7]. Such indirect encoding techniques may improve the capabilities of the modular approach presented in this paper as well, as will be discussed in Section 5.

Recent progress in building sophisticated physical robots was summarized by Holmes et al. [14]. They also discussed the role of mathematical models of body-limb and environment dynamics, central pattern generators, and proprioceptive and environmental sensing in the design of a very agile six-legged robot called RHex. The sprawled posture and compliant legs of RHex, inspired by characteristics found in insects, allows the robot to be stable and operate dynamically even on rocky and uneven terrain [16]. The stability is achieved through open-loop control utilizing only proprioceptive feedback. Similarly, the Sprawl hexapedal robot uses open-loop control for stable running [5]. Along the same lines, the evolved modular controllers in this paper perform well utilizing only proprioceptive sensing of joint angles.

In nature, the control systems of animals evolved together with their body morphology, resulting in tightly integrated efficient agents. Inspired by this observation, some researchers have evolved both the controller and the robot morphology concurrently. Examples of such body-brain evolution include the virtual block creatures of Karl Sims [28], and the generative representations used by Hornby and Pollack [15]. Although not necessarily legged creatures, the agents produced by such methods may also be symmetric and regular, and may be able to walk in synchronized manner.

Most of the above approaches are motivated by the biological central pattern generators (CPGs), i.e. groups of neurons that produce oscillatory signals for locomotion [27]. CPGs are typically implemented as CTRNNs, using leaky integrator neurons. The modular controller networks described in this paper also function as CPGs, but they are based on simpler, sigmoidal neurons. Patterned oscillations are still possible in these networks because they are in essence symmetrical, coupled cell systems. Theoretically, such systems are CPGs, and in practice, they generate various gaits for quadrupeds and hexapods [6]. The approach in this paper implements such systems as modular neural networks and uses evolution to find the appropriate system parameters. The theoretical behavior of these coupled cell systems is reviewed in detail next.

2.2 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 [9]. 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 symmetrical, coupled cell systems can exhibit synchronous and phase-related periodic patterns in their state. Collins and Stewart [6] showed that this patterned behavior can be used to model CPGs and to explain symmetries in animal gaits.

Following their method, the modular controllers in this paper are abstracted as coupled cell systems having symmetries. The patterned oscillatory behavior that follows from these symmetries is independent of the model, 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.

The particular coupled cell system utilized in this paper is due to Pinto and Golubitsky [23]. While they used it to understand biped locomotion, the system is extended in this review to quadrupeds in order to use it to develop modular controllers for multilegged robots. 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 $F : (\mathbf{R}^k)^4 \rightarrow \mathbf{R}^k$ 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.

This system corresponds to the graph shown in Figure 2, which helps visualize its symmetries. The cells are numbered 1 through 4, and the arrows show the coupling between them. Each arrow type represents a different type of coupling, corresponding to a different argument position in function F. A symmetry of the graph is a permutation of the cells that preserves the coupling between the cells. For example, the permutation $\rho = (1\ 2)(3\ 4)$, which swaps cells 1 and 2 as well as cells 3 and 4, is a symmetry of the graph in Figure 2. The permutation $\tau = (1\ 3)(2\ 4)$ is another symmetry, and the composition $\tau\rho$ obtained by performing the two permutations in sequence is yet another symmetry.

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 ρ to (1) produces an equivalent system of ODEs for which $\rho\mathbf{x}(t)$ is a solution. Thus, the system of ODEs inherits the symmetry ρ from the corresponding graph.

Periodic solutions 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 ρ twice or τ 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 $\mathbf{x}(t) = (\mathbf{x}_1(t), \mathbf{x}_2(t), \mathbf{x}_3(t), \mathbf{x}_4(t))$, resulting in specific patterned behavior for the system. For example, the bound gait

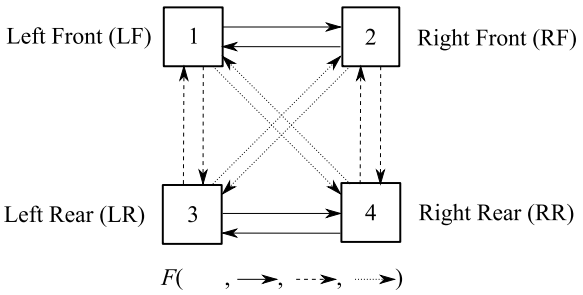


Figure 2: Graph corresponding to the coupled cell system in equation (1). The cells are numbered 1 through 4, and the arrows indicate that a coupling exists between them. The different arrow types represent different couplings, corresponding to different argument positions in the function F, as shown in the legend below. This graph helps visualize the symmetries of the underlying coupled cell system, and shows how the cells may be assigned to control the legs of a quadruped robot for producing different gaits.

Table 1: Gaits corresponding to different combinations of phase shifts θ_ρ and θ_τ associated with two permutation symmetries ρ and τ of the coupled cell system in Figure 2. Thus, this system can have solutions modeling a variety of common quadruped gaits.

	Pronk	Pace	Bound	Trot
θ_ρ	0	$\frac{T}{2}$	0	$\frac{T}{2}$
θ_τ	0	0	$\frac{T}{2}$	$\frac{T}{2}$

pattern results from the following constraints. The symmetry ρ is first applied to $\mathbf{x}(t)$ with a phase shift of $\theta_\rho = 0$, 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 τ to this solution with a phase shift of $\theta_\tau = \frac{T}{2}$ 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 produces a bound gait (as discussed in Section 3.4).

Other common gaits can be obtained similarly by selecting different combinations of values for θ_ρ and θ_τ, as shown in Table 1. Although these gaits are possible solutions of the system, whether any particular gait can be obtained in practice depends on the details of the cell dynamics and the couplings.

For further analysis of the patterned oscillatory behavior of coupled cell systems, see Golubitsky and Stewart [9]. In particular, they showed that a system with prescribed symmetries exists whose periodic solutions are asymptotically stable, meaning that the patterns are unchanged by small perturbations. However, other methods must be used to find such a system with the appropriate internal cell dynamics and coupling functions. In this paper, modular neuroevolution is shown to serve as such a method (Section 3.4).

The theoretical results make the high-level behavior of modular controller networks easy to understand. Consequently, in contrast to other approaches (such as those reviewed in Section 2.1), these controllers are easy to design and scale well to robots with more legs and more complex legs. Moreover, neuroevolution is an effective alternative to manual design of coupled cell system ODEs with desired characteristics (such as [6]). Thus, modular neuroevolution is a promising approach to developing multilegged robot controllers.

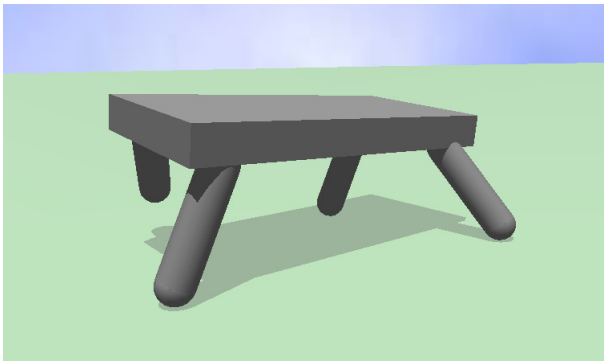


Figure 3: The quadruped robot model. The legs are attached to the body by hinge joints with axes of rotation tilted sideways, allowing the legs to make full circular rotation. Locomotion is achieved by coordinating the circular movements of the legs. This model is a simple but physically realistic platform that also allows scaling up to more complex robots by adding more legs or by increasing the legs' degrees of freedom.

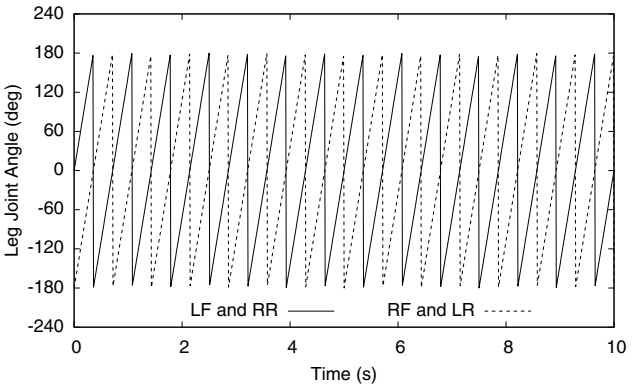
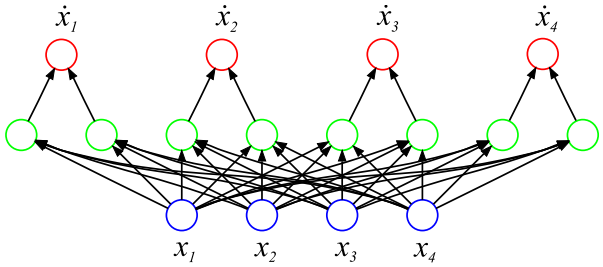
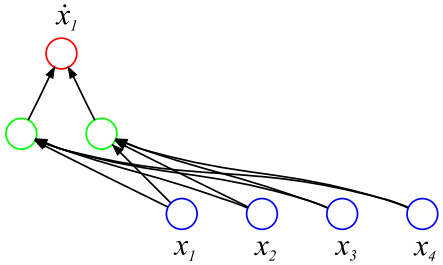


Figure 4: Leg angles specified by the hand-designed controller. Plotted over time, these leg angles produce sawtooth waveforms, i.e. the legs rotate with constant angular velocity. The diagonal pairs of legs move in unison, but each pair moves a half-period out of phase with the other pair, producing a trot gait.



(a) Non-modular controller network



(b) First module of the modular controller network

Figure 5: Genotypes of the non-modular and modular controller networks for the quadruped robot model. Inputs consist of the leg angles and outputs of the desired angular velocities. Both the non-modular and modular networks have the same architecture; however, the genotype of the non-modular network contains the entire network while that of the modular network contains only the first module. The full modular network is constructed by replicating the first module with different permutations of the inputs.

3. METHOD

The approach was evaluated experimentally by evolving effective walking behavior for a simulated quadruped robot in a terrain with and without obstacles. Three types of controllers were tested: (1) hand-designed controllers that serve as a baseline for performance comparisons, (2) non-modular network controllers in which the parameters of the entire network are evolved at once, and (3) modular network controllers in which parameters of only one module are evolved and the full network is formed by combining modules. These methods are each described in detail below.

3.1 Robot Model

The robot model resembles a table with a rectangular body supported by legs at the four corners (Figure 3). The legs are cylindrical with capped ends, and attached to the body by a hinge joint having full 360° degrees 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 or backward locomotion is achieved by coordinating the circular movements of the leg. The controller activates the simulated servo motor attached to each joint by specifying either the desired joint angle or the angular velocity.

This model can be generalized and made more complex by adding more legs to the table or by using joints, such as a universal joint, that have more degrees of freedom. Such more complex robots will be used to test the scale-up properties of the approach in Section 4.

3.2 Hand-designed Controller

The hand-designed controller specifies the desired angular positions of the legs as functions of time, coordinated so as to obtain a trot gait (Figure 1d). Plotted over time, the leg angles for the quadruped robot model described above produce sawtooth waveforms that all have the same period (Figure 4). The waveforms of the diagonal leg pairs are synchronous, and a half-period out of phase with the other pair. This gait is an effective base gait that occurs a lot in nature, and the evolved gaits can be compared with it.

3.3 Non-modular Controller

The non-modular controller network for the quadruped robot is shown in Figure 5a. Although a variety of architectures are possible, this simple two-layered architecture was chosen to facilitate a fair comparison to the modular approach. The inputs are the angular positions of the leg joints, and the outputs are the desired angular velocities of the legs. Each input unit is connected to all hidden units, but each output unit is connected only to two hidden units, which send their activation exclusively to that output unit. The hidden and output units have sigmoidal activation functions with a bias and slope as parameters; the input units do not perform any computation. Since the genotype of the non-modular controller represents the entire network, all parameters of the network will have to be optimized through evolution.

A more general robot model having additional joint angles requires a network with additional inputs and outputs representing those angles. The number of hidden nodes will also need to be increased to achieve better performance on more complex robots, thus increasing the search space.

3.4 Modular Controller

The non-modular network can be decomposed structurally into four subnetworks (modules), each containing one of the output units, the two hidden units from which it receives activation, and all the input units. The first such subnetwork is used as the genotype of the modular controller (Figure 5b), making the evolutionary search space one-fourth of that for non-modular evolution. The full modular network is obtained by instantiating the other subnetworks with copies of the first subnetwork, with appropriate permutations of the inputs.

Such construction, combined with the fact that the network computes the time derivative of the joint angles as a function of each other, allows the modular network to be modeled as a coupled cell system. The modules correspond to cells, and the input connections of modules correspond to couplings. Since the modules are identical, the cells are also identical. The permutations of module inputs determine how cells are coupled to each other. These permutations can be chosen such that the system of ODEs corresponding to the resulting coupled cell system has the same form as equation (1). In this formalism, $\mathbf{x}_i$ represents the joint angles for leg i, and F represents the functional equivalent of each neural network module.

Analysis of the symmetries of equation (1) and its corresponding graph (Figure 2) in Section 2.2 showed that this coupled cell system can have periodic solutions that signify synchronous and phase-related oscillatory behavior of the cells. When these cells (i.e. module outputs) are assigned to control robot legs, symmetric and regular gaits are obtained. With such a setup, neuroevolution can exploit symmetry in the robot morphology, and discover an appropriate F that produces effective gaits.

This coupled cell system models the behavior of the controller only; it does not take into account the dynamical effects of the robot and the environment in which the robot operates. Such effects may be thought of as perturbations to the state variables of the system, and the evolved controllers must be robust against them. Evolution can also discover a controller that uses such perturbations as feedback for switching to more suitable gaits on difficult terrains, as seen in experiments described in the next section.

4. RESULTS

Experiments comparing the controllers obtained by the three different approaches are now described. Evaluations are performed on flat ground and on terrain with obstacles, and on robots with different number of legs and different number of degrees of freedom for each joint. The experiments are run in a realistic physical simulation of locomotion, as will be described next.

4.1 Experimental Setup

The experiments were implemented utilizing a number of open source tools. The neuroevolution code was implemented as a library layer on top of the Open BEAGLE evolutionary computing framework [22], taking advantage of its generic programming interface. The experiments also utilized the customizable methods for logging and statistics collection in Open BEAGLE, as well as its XML-based configuration mechanism for managing parameters and specifying operators. The physics simulation was programmed using OPAL [21], an abstraction library on top of the Open Dynamics Engine (ODE) [19]. The Object-Oriented Graphics Rendering Engine (OGRE) [20] library was used for 3D visualization of the simulation.

The initial population of networks is created with connection weights chosen randomly from the range $[-2, 2)$, neuron biases set to 0, and neuron sigmoid slopes set to 1. Three types of mutations are used, one for each of the above parameter types: (1) weight mutations, (2) bias mutations, and (3) slope mutations. All three types are implemented as Gaussian perturbations (with $\sigma = 0.2$) acting with a specified probability (0.5) on each of the parameters belonging to that type. In each generation, an offspring is created by first selecting a parent in a two-way tournament, and then applying exactly one of the three mutation types, chosen with equal probability. In addition, the network with the best fitness is copied without change to the next generation. A population size of 200 is used in all experiments.

Each network is evaluated in a physically realistic simulation in which the network controls the locomotion of a robot. When the robot is initially placed in the simulation environment, its longitudinal and lateral axes are aligned with the coordinate directions of the ground plane. The simulation is then carried out for one minute of simulated time with step size 0.01s, after which the fitness of the controller network is calculated as the maximum distance travelled by the robot along either of the two coordinate dimensions (i.e., the Chebyshev distance [18]). This fitness measure ensures that the evaluation of controller networks is fair even on the terrain with bumps in the form of concentric squares (Section 4.3). Although appropriate as a quantitative measure of performance, this measure does not capture how good the controllers are qualitatively. Therefore, the resulting gaits were also visualized and evaluated manually at the end of evolution to confirm that the champion controller networks had good locomotive properties.

For all experiments, evolution was run for 500 generations and repeated 10 times, each time with a different random number seed. The average and standard deviation of champion network fitness for all experiments are shown in Figure 6. The best fitness of the hand-designed controller is also plotted in Figure 6 for comparison. This controller was obtained by manually implementing domain knowledge as well as by experimenting with gait periods similar to those produced by the modular controllers. The following subsections discuss the results of each experiment in detail.

4.2 Flat Terrain

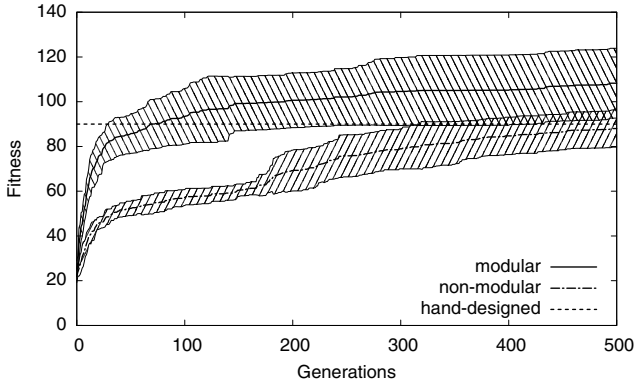
In the first experiment, networks were evolved to control the quadruped robot on flat terrain. The modular networks performed significantly better than the non-modular networks through all generations, as illustrated in Figure 6a. This result implies that the robots with modular controllers are able to travel farther than those with non-modular controllers. Figure 6a also shows that the modular controllers have higher fitness than the hand-designed controller. This result means that through evolution it was possible to discover more efficient or better tuned gait than could be designed by hand.

When the locomotion of champion networks were visualized, another important benefit of modular evolution was revealed. The modular networks produce regular and symmetric gaits, such as pronk, pace, bound and trot, similar to those found in animals (Section 1). In contrast, the non-modular networks typically produce asymmetric gaits in which one or two legs have limited mobility, resembling the gaits of crippled animals.

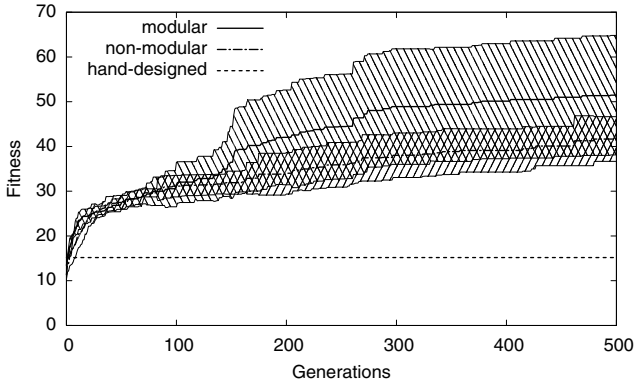
These results, establish a baseline for comparisons involving more difficult terrain and more complex robots, which are described below.

4.3 Obstacle Terrain

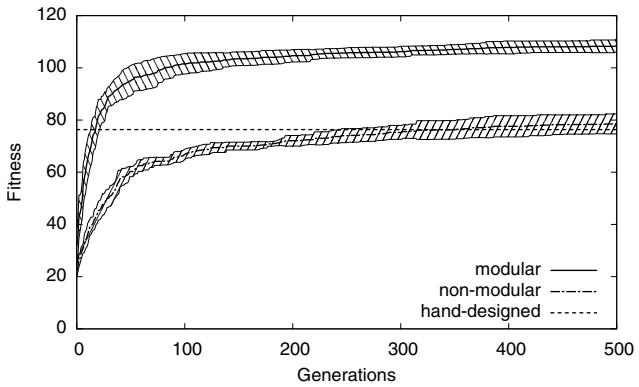
In the next experiment, obstacles in the form of bumps (or walls, or fences) were placed on the ground at regular intervals to make the task of the controller more difficult (Figure 7). Five bumps were used in each coordinate direction, together forming concentric squares aligned with the coordinate directions of the ground plane. The first bump was at 10 units from the center, and the remaining bumps were at every 5 units outwards. The robot was initially placed at the center. Note that although moving in a skewed direction can



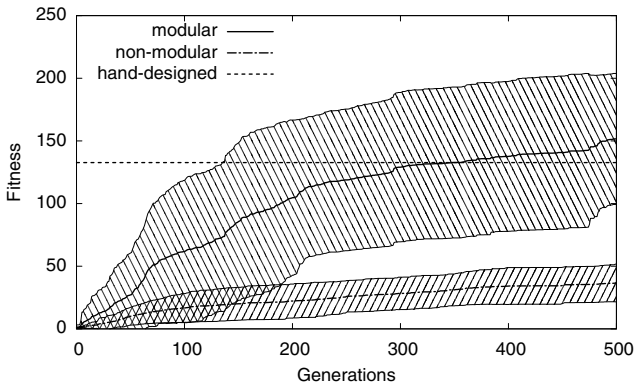
(a) Quadruped robot with hinge joints on flat terrain



(b) Quadruped robot with hinge joints on obstacle terrain



(c) Hexapod robot with hinge joints on flat terrain



(d) Quadruped robot with universal joints on flat terrain

Figure 6: Performance of hand-designed, modular, and non-modular neuroevolution controllers on different terrains and robot models. Bold lines are averages and the shaded regions on either side are standard deviations obtained over 10 trials of evolution. Fitness of the hand-designed controller is also shown for reference. (a) Modular controllers perform significantly better than both non-modular and hand-designed controllers in the baseline experiment. (b) Performance gap between modular controllers and hand-designed controllers increases significantly when obstacles are added to the environment. (c) Similarly, the performance gap increases significantly when number of legs is increased to six. (d) Similarly, performance gap increases significantly when an angular degree of freedom is added to each leg joint. These results demonstrate the advantage of modular evolution over non-modular evolution and hand-design in discovering controllers for multilegged robots.

cover more distance without encountering the bumps, it does not increase the Chebyshev distance measure that is used as fitness.

As in the experiment with the flat terrain, the modular controllers have a clear advantage over non-modular and hand-designed controllers (Figure 6b). The hand-designed controller has a particularly hard time: It is unable to get past the first or second bump, depending on how the legs initially hit the bumps. This result further demonstrates how evolution can discover more effective behavior than can be achieved through hand-design.

The bumps perturb the dynamics of the controller more than flat ground does. As a result, evolution often discovers controllers that change to a more favorable gait when bumps are encountered. An example of this phenomenon is shown in Figure 8, where the robot changes from bound to trot when it hits the first bump, allowing it to get over it more easily. The evolved modular controllers are thus more robust and flexible than the hand-designed controller.

4.4 Number of Legs

Although the genotype of the modular controller encodes only one module, it receives input from all legs. Therefore, when more legs are added to the robot model, only a few more parameters need to be added to the modular genotype. In contrast, the non-modular genotype needs significantly more parameters because it encodes all modules separately. Consequently, evolutionary search is harder for the non-modular method than for the modular method when the number of legs is increased.

This hypothesis was tested by evolving controllers for a hexapod robot having equally spaced rows of legs along its longitudinal axis. A hand-designed controller was also built by generalizing the trot gait of the quadruped into a tripod gait, in which the front and rear legs of one side are in phase with the middle leg of the other side. The coupled cell system associated with the hexapod modular controller generalizes the quadruped system of Figure 2, and is obtained by adding a third row of cells and connecting them with the other cells using bidirectional links. The resulting cell system has two types of symmetries: (1) swapping of the left and right columns of cells, and (2) cycling of the rows of cells either up or down.

Fitness of the modular and non-modular controllers in this experiment are plotted in Figure 6c. The modular controllers now outperform the non-modular and hand-designed controllers with a wider performance gap than in the quadruped case in Figure 6a. The non-modular controllers perform at the same level as the hand-designed controllers, possibly because they only have to get a few legs moving in a coordinated fashion to make reasonable progress. On the other hand, the modular controllers produce gaits with symmetric leg movements, typically producing a longitudinal wave pattern with same-row legs in phase or a half-period out of phase, which is much more effective and similar to the hexapod gaits in nature. Thus,

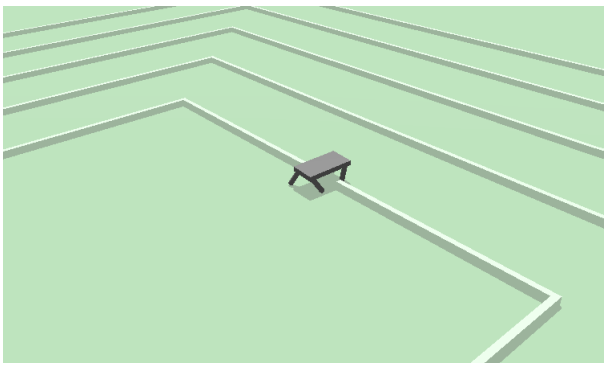


Figure 7: Robot navigating a terrain with obstacles. The obstacles consist of five equally spaced bumps forming concentric squares around the robot. Moving on this terrain by stepping over the bumps is a difficult task: The modular controller performs this task effectively, whereas the non-modular and hand-designed controllers struggle. Visualization videos of such behaviors can be seen at the website `http://nn.cs.utexas.edu/keyword?modularne`.

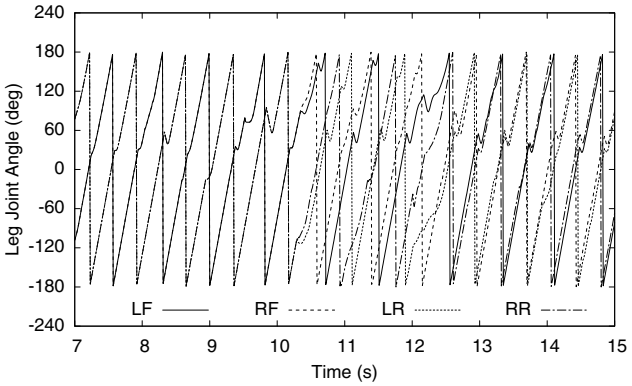


Figure 8: Gait changes produced by an evolved modular controller on a terrain with obstacles. As in Figure 4, the plot shows angular positions of the four legs of the robot over time. Initially, the front pair of legs (LF and RF) move out of phase from the rear pair (LR and RR), i.e. the robot has a bound gait (Figure 1c). The robot encounters the bumps at about 10 seconds. As it tries to move forward, its legs hit the bumps repeatedly, perturbing the dynamics of the controller. These perturbations cause the controller to transition to a trot gait in which the diagonal pairs of legs move out of phase (Figure 1d). This change of gait allows the robot to step over the bumps. Thus, modular controllers have the flexibility and robustness necessary for navigating difficult terrain.

modular evolution scales up better than non-modular evolution when more legs are added to the robot.

4.5 Joint Degrees of Freedom

The robot can also be made more complex and difficult to control by increasing the number of angles that has to be controlled in each leg. That is, while the legs trace a cone in the previous robot models, a more challenging model would require controlling the forward-backward (longitudinal) and sideways (lateral) rotations of the legs separately. Since the controller inputs consist of the joint angles, the number of input connection weights increases correspondingly. This increase is limited to the single genotype module for modular networks, while it gets multiplied by the number of joints for non-modular networks. Again, because of the large number of extra parameters non-modular evolution has to search, finding a good controller will be harder for non-modular than for modular evolution.

The final experiment tested this hypothesis by replacing each hinge joint of the quadruped robot with a universal (hip) joint. This change in the robot model doubles the number of degrees of freedom of each joint and, consequently, the number of controller inputs. The forward-backward rotation is limited to 30° in both directions and the sideways rotation from vertical to 45° outwards. The hand-designed controller is generalized to this setup by adding signals for the lateral angle of each leg such that it is a quarter-period out of phase with its longitudinal angle, producing a trot gait.

The above hypothesis is confirmed by Figure 6d. The performance gap between modular and non-modular evolution has widened significantly: Modular evolution now performs nearly four times better than non-modular evolution, and slightly better than the hand-designed controller. Visualizing the gaits reveals that non-modular controllers typically produce gaits resembling severely crippled animals, with nearly all their legs appearing disabled and poorly coordinated. These robots often have sideways moving gaits and may have certain joint angles locked in a fixed position. The modular controllers, on the other hand, typically produce regular trot gaits that achieve high fitness. These results demonstrate that modular evolution scales up well when the robot legs are made more complex, in contrast to the poor performance of non-modular evolution.

5. DISCUSSION AND FUTURE WORK

In all the experiments described above, modular evolution produced controllers that were better than non-modular evolution in two respects: (1) better fitness by travelling farther, and (2) better gaits that are symmetric and resemble those found in nature. Furthermore, this performance gap widens significantly in both respects when the number of legs or their angular degrees of freedom is increased. The modular controllers also demonstrated the capability to change gaits in response to changes in the environment. The modular controllers perform better than hand-designed open-loop controllers as well, particularly on an obstacle terrain where the modular controllers navigate bumps much more effectively. These results suggest that modular evolution is successful in taking advantage of the symmetry and regularity in the robot morphology to discover efficient controllers.

The symmetry of the robot was manually specified in this paper, using the theory of coupled cell systems. This approach can be extended in three ways to produce more general controllers. First, a single module was replicated for each leg because the legs were identical; in a more complex robot model, more than one type of module may be needed, for example to represent different front and rear legs. The resulting coupled cell system will have more than one type of cell and a different symmetry. Second, the cells were coupled and the cells assigned to legs manually. Systems with other symmetries could be explored by changing these couplings and assignments. Third, the four-cell system of Figure 2 was designed by hand, based on domain knowledge. Larger systems with more cells can be built to obtain more sophisticated behavior.

In future work, these design choices can be optimized by making them parameters for evolution to discover automatically. Such automation frees the human from the design loop, and creates the potential for discovering better solutions that may not appear straightforward to a designer. One potential approach is similar to hierarchical SANE [17], where one population of neurons and another population of network blueprints (that specify how the neurons are combined to form networks) are evolved simultaneously. In a similar manner, a module population could evolve specialized partial solutions such as controlling a single leg, while a blueprint population would evolve combinations of modules representing full solutions,

specifying the couplings between modules and the assignment of modules to legs.

Another approach is to utilize indirect encoding such as cellular encoding (CE) [13] or genetic regulatory networks (GRNs) [26] that allow encoding modular neural networks as developmental programs. Such encodings can represent networks with repetitive structures (i.e. modules) using recursion, and can therefore potentially discover the modules and evolve their structure and parameters automatically.

In the future, the modular neuroevolution approach developed in this paper will be applied to more complex terrains and robot morphologies. Higher-level behaviors such as path-following and foraging will also be investigated. Such behaviors require sensing the environment to provide additional inputs to the controller. In the coupled cell network model, these inputs may be treated as parameters of the system, and adjusting these parameters can result in different robot behaviors.

Complex high-level behaviors may require controllers to have a hierarchical structure. Lower levels of such a controller implement patterned behavior such as gaits, while higher levels implement decision-making suitable for the task. For tasks that require memory, recurrency in the controller networks is expected to be useful, and it also makes it possible to evolve more interesting and complex behavior. Because the controllers for such systems are extremely complex, the automation approach outlined above may prove essential. Finally, if these approaches are successful in evolving controllers for a detailed model of a physical robot, they can eventually be tested on real physical robots. Thus, the modular neuroevolution approach outlined in this paper constitutes a promising starting point for developing efficient, robust, and flexible controllers for multilegged robots in the real world.

6. CONCLUSION

An approach for evolving modular neural networks to control legged locomotion was presented in this paper. As demonstrated in the experiments, modular networks have two main advantages compared to non-modular networks with the same structure: (1) they have smaller genomes, which makes it easier for evolution to find solutions with good fitness, and (2) symmetries in the problem can be expressed in the modular structure, allowing evolution to discover effective symmetric gaits. Furthermore, modular controllers were shown to evolve the ability to change gaits in response to changes in the environment, and to scale well to more complex robot morphologies. The controller fitness was evaluated in a physically realistic simulation of dynamics, which suggests that the approach should be useful for building physical multilegged robots in the future.

7. REFERENCES

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

[2] R. D. Beer and J. C. Gallagher. Evolving dynamical neural networks for adaptive behavior. *Adaptive Behavior*, 1(1):91–122, 1992.

[3] A. Billard and A. J. Ijspeert. Biologically inspired neural controllers for motor control in a quadruped robot. In *Proceedings of IJCNN-2000*, pages 637–641, 2000.

[4] R. A. Brooks. A robot that walks; emergent behaviors from a carefully evolved network. Technical Report AIM-1091, Massachusetts Institute of Technology, 1989.

[5] J. E. Clark. *Design, Simulation, and Stability of a Hexapedal Running Robot*. PhD thesis, Stanford University, 2004.

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

[7] D. Filliat, J. Kodjabachian, and J.-A. Meyer. Evolution of neural controllers for locomotion and obstacle avoidance in a six-legged robot. *Connection Science*, 11(3/4):225–242, 1999.

[8] D. Floreano and F. Mondada. Evolutionary neurocontrollers for autonomous mobile robots. *Neural Networks*, 11:1461–1478, 1998.

[9] 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*, chapter 8, pages 243–286, 2002.

[10] F. Gomez and R. Miikkulainen. Active guidance for a finless rocket using neuroevolution. In *Proceedings of GECCO-2003*, pages 2084–2095, 2003.

[11] F. J. Gomez, J. Schmidhuber, and R. Miikkulainen. Efficient non-linear control through neuroevolution. In *Proceedings of ECML-2006*, pages 654–662, 2006.

[12] F. Gruau. Automatic definition of modular neural networks. *Adaptive Behavior*, 3(2):151–183, 1994.

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

[14] P. Holmes, R. J. Full, D. Koditschek, and J. Guckenheimer. The dynamics of legged locomotion: Models, analyses, and challenges. *SIAM Review*, 48(2):207–304, 2006.

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

[16] D. E. Koditschek, R. J. Full, and M. Buehler. Mechanical aspects of legged locomotion control. *Arthropod Structure and Development*, 33(3):251–272, July 2004.

[17] D. E. Moriarty and R. Miikkulainen. Hierarchical evolution of neural networks. In *Proceedings of the 1998 IEEE Conference on Evolutionary Computation*, pages 428–433, 1998.

[18] F. Murtagh. Clustering in massive data sets. In *Handbook of Massive Data Sets*, chapter 14, pages 501–543, 2002.

[19] ODE, 2007. `http://www.ode.org`.

[20] OGRE, 2007. `http://www.ogre3d.org`.

[21] OPAL, 2007. `http://opal.sourceforge.net`.

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

[23] C. M. A. Pinto and M. Golubitsky. Central pattern generators for bipedal locomotion. *Journal of Mathematical Biology*, 53(3):474–489, 2006.

[24] M. H. Raibert. Legged robots. *Communications of the ACM*, 29(6):499–514, 1986.

[25] M. H. Raibert, M. Chepponis, and J. H. Benjamin Brown. Running on four legs as though they were one. *IEEE Journal of Robotics and Automation*, 2(2):70–82, 1986.

[26] J. Reisinger and R. Miikkulainen. Acquiring evolvability through adaptive representations. In *Proceeedings of GECCO-2007*, 2007.

[27] S. V. Shastri. A biologically consistent model of legged locomotion gaits. *Biol. Cyber.*, 76(6):429–440, 1997.

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

[29] K. O. Stanley, B. D. Bryant, and R. Miikkulainen. Real-time neuroevolution in the NERO video game. *IEEE Transactions on Evolutionary Computation*, 9(6):653–668, 2005.

[30] R. A. Téllez, C. Angulo, and D. E. Pardo. Evolving the walking behaviour of a 12 dof quadruped using a distributed neural architecture. In *Biologically Inspired Approaches to Advanced Information Technology*, pages 5–19, 2006.