Evolving Coordinated Quadruped Gaits with the HyperNEAT Generative Encoding

Jeff Clune, Benjamin E. Beckmann, Charles Ofria, and Robert T. Pennock

Abstract— **Legged robots show promise for complex mobility tasks, such as navigating rough terrain, but the design of their control software is both challenging and laborious. Traditional evolutionary algorithms can produce these controllers, but require manual decomposition or other problem simplification because conventionally-used direct encodings have trouble taking advantage of a problem's regularities and symmetries. Such active intervention is time consuming, limits the range of potential solutions, and requires the user to possess a deep understanding of the problem's structure. This paper demonstrates that HyperNEAT, a new and promising generative encoding for evolving neural networks, can evolve quadruped gaits without an engineer manually decomposing the problem. Analyses suggest that HyperNEAT is successful because it employs a generative encoding that can more easily reuse phenotypic modules. It is also one of the first neuroevolutionary algorithms that exploits a problem's geometric symmetries, which may aid its performance. We compare HyperNEAT to FT-NEAT, a direct encoding control, and find that HyperNEAT is able to evolve impressive quadruped gaits and vastly outperforms FT-NEAT. Comparative analyses reveal that HyperNEAT individuals are more holistically affected by genetic operators, resulting in better leg coordination. Overall, the results suggest that HyperNEAT is a powerful algorithm for evolving control systems for complex, yet regular, devices, such as robots.**

I. Introduction

Legged robots are likely to play an increasingly important role in our lives in generations to come. Legged consumer robots already exist, such as the biped ASIMO and the quadruped AIBO. Both the military and industry have a variety of legged robots under development. One benefit of legged robots over their wheeled counterparts is their mobility on rugged terrain, but a major drawback is the challenge of creating controllers for them. The problem is complicated because of the number of degrees of freedom in each leg and because of the body's changing center of mass and momentum. Human engineers have had to design the majority of controllers for legged robots, which is a difficult and time-consuming process [1], [2]. Furthermore, given how sensitive gait controllers are to slight changes in the configuration of a robot, a new gait must be created each time a robot is changed, which can lead to significant delays in the prototyping stage of robotic development [3].

It is not surprising, therefore, that people have tried to automate the process of gait creation. Evolutionary computation, frequently involving the evolution of neural network controllers, has been successfully used to this end [3]–[10]. Evolved gaits are often better than those produced by human designers; one was even included on the commercial release of Sony's AIBO robotic dog [3], [7]. However, many researchers have found that they cannot hand the entire problem over to evolutionary algorithms, because of the large number of parameters that need to be simultaneously tuned to achieve success [3], [7], [9]–[15]. Many of these scientists report that, while it is possible to evolve a controller to manage the inputs and outputs for a single leg, once evolution is challenged with the inputs and outputs of many legs, it fails to make progress.

One solution that has worked repeatedly is to help the evolutionary algorithm 'see' that there are regularities and symmetries to the problem. This approach involves manually decomposing the problem by, for example, evolving the controller for one leg of a quadruped and then copying that controller to each leg, with some variation in phase. This tactic imposes modularity on the network and stipulates that a single encoding will be used for multiple modules. Many permutations of this strategy of manually decomposing the problem have produced functioning gaits [3], [7], [12]–[15]. Another type of manual decomposition, which is often used in addition to the previous one, is to simplify the high-level problem of locomotion by breaking it into manually-defined subproblems (e.g., producing leg oscillations, not falling over, moving certain legs in synchrony, etc.), and first rewarding the simple problems, then more advanced problems, on upwards until a high-level goal, such as rapid locomotion, can be rewarded directly [9]–[11]. Combinations of these two strategies for simplifying the problem are not unique to quadruped control, and have been used in other attempts to evolve controllers, such as for helicopters [16 and cites therein].

Unfortunately, both of these strategies have drawbacks. It would be better if we could completely automate the process and remove the need for human engineers to spend time decomposing the problem. Furthermore, such manual decomposition potentially introduces constraints and biases that could preclude the attainment of better solutions [17]. Finally, if we can employ algorithms that can automatically

Manuscript received November 14th, 2008 and supported by the Cambridge Templeton Consortium ("Emerging Intelligence: *Contingency, Convergence and Constraints in the Evolution of Intelligent Behavior*"), NSF grants CCF-0643952, CCF-0523449, CCF-0750787, CNS-0751155, and CCF-0820220, U.S. Army Grant W911NF-08-1-0495, the DARPA FunBio program, and a Quality Fund Grant from Michigan State University. Jeff Clune (jclune@msu.edu, 517.214.1060), Benjamin E. Beckmann (beckma24@msu.edu), Charles Ofria (ofria@msu.edu) and Robert T. Pennock (pennock5@msu.edu) are in the Department of Computer Science and Engineering at Michigan State University (MSU) in East Lansing, MI, USA. Robert T. Pennock is also in the Department of Philosophy and the Lyman Briggs College at MSU.

978-1-4244-2959-2/09/$25.00 ©

discover and exploit regularities in a problem, such algorithms will be able to do so for complex problems with respect to regularities of which humans might not be aware.

For these reasons, it is important to investigate algorithms that can automatically exploit regularities in problems. Generative encodings can facilitate this goal by allowing elements in a genome to be reused to produce many parts of a phenotype [19]. For example, the roughly 25,000 genes in human DNA encode for the trillions of cells in a human. This developmental system is in contrast to a direct encoding, where each phenotypic element is specified by a corresponding genotypic element. For example, if one were evolving a four-legged table using a direct encoding, it would take four mutations, one for each leg, to completely change the height of the table. In contrast, with a generative encoding where the length of all legs is specified only once, a single mutation at this site would bring about a coordinated change in the height of the table.

Reuse in generative encodings can produce the desirable properties of symmetry, repetition of modules, and coordinated mutational effects. Previous work has shown that generative encodings produce more modular, complex phenotypes with higher fitnesses and more beneficial mutations on average than direct encoding controls [4]. It has further been shown that modular neural network controllers tend to evolve more effective gaits for legged robots [7]. If generative encodings produce more modular neural network controllers, can they outcompete direct encodings on this task? In this paper, we shed light on this question by testing whether a generative encoding will evolve quadruped gaits without requiring the manual decomposition of the problem. This paper improves our understanding of how to evolve quadruped gaits and also investigates how successful generative encodings are at exploiting problem-regularity, which they have been shown to do on simpler problems [18].

While it has not been the norm, generative encodings have been used previously to evolve the gaits of legged creatures. In two well-known cases, a generative encoding was used to evolve the gaits and the morphologies of creatures [4], [6]. These dual evolutionary goals complicate analysis because the creatures may not be regular, and because it is unclear if any of the demonstrated fitness advantage of generative encodings (e.g., [4]) was due to the ability to evolve modular neural networks or due to the ability to build better morphologies. In one paper with similar goals as the current work, a generative encoding and direct encoding were compared for their ability to evolve a gait for a legged creature in an attempt to see whether the generative encoding could exploit the regularity of the problem without the problem being simplified or manually decomposed [5]. However, this project used a simple model of a six-legged insect that had only two degrees of freedom per leg. Nevertheless, the work showed that a generative encoding could automatically discover the regularity of the problem and decompose it by encoding a neural submodule once and using it repeatedly. The generative encoding also outperformed a direct encoding by solving the problem faster. Unfortunately, computational limits at the time meant that such results were anecdotal and not statistically significant because so few trials could be performed.

To summarize, in this paper we demonstrate a generative encoding that enables the exploitation of regularity in the legged locomotion problem, making its solution feasible for an evolutionary algorithm without simplification via manual decomposition. In addition, the generative encoding we tested is new and shows real promise for a broad range of domains. As such, it is interesting in its own right to determine how it performs on the challenging legged locomotion problem.

II. A Description of the Generative Encoding and Its Direct Encoding Control[1]

The generative encoding utilized in this paper is called HyperNEAT [20] and is freely available (http://eplex.cs.ucf.edu). It was recently introduced as a generative encoding that evolves neural networks with the principles of the widely used NeuroEvolution of Augmenting Topologies (NEAT) algorithm [21]. HyperNEAT evolves Compositional Pattern Producing Networks (CPPNs), each of which is a function that takes an input and produces an output (Fig 1). If the goal is to evolve two-dimensional pictures, the inputs to the CPPN function are the Cartesian coordinates of each of the pixels on the canvas. The output of the function determines the color or shade of the pixel.

Evolution can be used to modify a population of CPPN functions. Each CPPN is itself a directed graph where each node is a math function, such as sine or Gaussian. The nature of the functions used can create a wide variety of desirable properties, such as symmetry (e.g., an absolute value or Gaussian function) and repetition (e.g., a sine or cosine function) that evolution can take advantage of. Because a directed graph of functions is used, nested coordinate frames can develop. For instance, a sine function used early in the network can create a repeating theme that, when passed into the symmetrical Gaussian function, creates a repeating series of symmetrical motifs. This process is similar to how natural organisms develop. For example, many organisms set up a repeating coordinate frame (e.g., body segments) within which are symmetrical coordinate frames (e.g., left-right body symmetry). Asymmetries can be generated by sourcing global coordinate frames, such as $f(x)$. The links between each node in a CPPN have a weight value that can magnify or diminish the values that pass along them. The ability to change these weights enables evolution to, for example, give strong weight to one part of the network generating symmetry while rendering the influence of another aspect of the network more subtle. When CPPNs are evolved artificially with humans doing the selection, the evolved shapes look surprisingly beautiful, complex, and natural [22]. More importantly, they exhibit the desirable features of

[1] This description is adapted from [18].

generative encodings, namely, the repetition of themes, symmetries, and hierarchies, with and without variation.

In addition to images, CPPNs can be used to generate neural networks [20]. In this case, the inputs are a constant bias value and the locations on a Cartesian grid of both a source node (e.g., $<x_1=4, y_1=4>$) and a target node (e.g., $<x_2=5, y_2=5>$). The function takes these five values (bias, x_1, y_1, x_2, y_2) as input and produces two output values [23]. The first value determines the weight of the link between the associated input and hidden layer nodes. The second value determines the weight of the link between the associated hidden and output layer nodes. All pairwise combinations of source nodes and target nodes are iteratively passed as inputs to a given CPPN to determine what the weight value is for each possible link. Thus, the CPPN function is a genome that encodes for a neural network phenotype (also called a *substrate*) [20].

An additional benefit of HyperNEAT is that it is purportedly the first neuroevolutionary algorithm capable of exploiting the geometry of a problem [20]. Because the link values between nodes are a function of the geometric positions of those nodes, if those geometric positions represent aspects of the problem that are relevant to its solution, HyperNEAT can exploit such information. For example, when playing checkers, the concept of adjacency (on the diagonals) is important. Link values between neighboring squares may need to be very different than link values between distant squares. HyperNEAT can use adjacency to create a connectivity motif and repeat it across the board [20], [23]. In the case of quadruped locomotion, HyperNEAT could, for example, implement front-back, left-right, or diagonal symmetries to produce common gaits.

Variation in HyperNEAT occurs when mutations change the CPPN function networks. Mutations can add a node to the graph, which results in the addition of a function to the CPPN network, or change its link weights. The functions used in CPPNs in this paper are sine, sigmoid, cosine, Gaussian, square, absolute root, linear, and one's complement. The evolution of the population of CPPN networks occurs according to the principles of NEAT, which

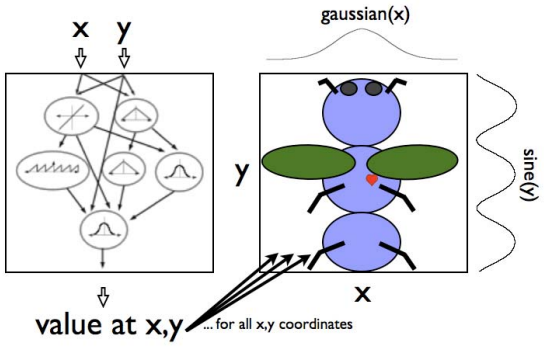

Fig. 1. CPPNs can compose math functions to generate the properties of symmetry and modular repetition, with and without variation. This figure is adapted from [22].

was originally designed to evolve neural networks. NEAT can be fruitfully applied to CPPNs because the population of CPPN networks is similar in structure to a population of neural networks.

The NEAT algorithm is unique in three main ways [21]. Initially, it starts with small genomes that encode simple networks and slowly *complexifies* them via mutations that add nodes and links to the network. This complexification enables the algorithm to evolve the network topology in addition to its weights. Secondly, it uses a fitness-sharing mechanism that preserves diversity in the system and allows new innovations time to be tuned by evolution before forcing them to compete against rivals that have had more time to mature. Finally, it uses historical information to perform crossover in a way that is effective, yet avoids the need for expensive topological analysis. A full explanation of NEAT can be found in [21].

Conveniently, a direct encoding version of NEAT exists that can serve as a control. *Fixed-Topology NEAT* (FT-NEAT, also called Perceptron NEAT or P-NEAT when it does not have hidden nodes), has been previously used to compare the generative encoding of HyperNEAT with a direct encoding that is similar to HyperNEAT in all ways, except its use of generative CPPNs [20]. FT-NEAT directly evolves neural network phenotypes. It is the same as NEAT without the complexification. In other words, FT-NEAT uses evolution to tune the weights of a neural network with a fixed topology. Since in HyperNEAT the complexification is performed on the CPPN, but the resultant neural network topology remains fixed, the topology of the FT-NEAT neural network substrate is also fixed. The end product of HyperNEAT and FT-NEAT are thus neural network substrates with the same topology, whose weights are determined in different ways. The rest of the elements from NEAT (e.g., fitness sharing) remain the same in both HyperNEAT and FT-NEAT, making the latter a good control.

III. The Experimental System and Setup

Following previous work, a neural network substrate configuration is used that separates the inputs and outputs onto separate planes [18], [20], [23]. The specific configuration here features three two-dimensional, 5x4 Cartesian grids forming an input, hidden and output layer (Fig. 2). There are no recurrent connections. All possible connections between adjacent layers exist (although weights can be zero, functionally eliminating the link), meaning that there are $(5x4)^2*2 = 800$ links in the substrate of each organism. Following [20] and [23], to facilitate the elimination of links, any link weight with a specified value less than 0.2 or greater than -0.2 is set to 0. Otherwise, the value is normalized to a range of -3 to 3.

The inputs to the substrate are the current angles of each of the 12 joints of the robot (described below), a touch sensor that provides a 1 if the lower leg is touching the ground and a 0 if it is not, the pitch, roll and yaw of the

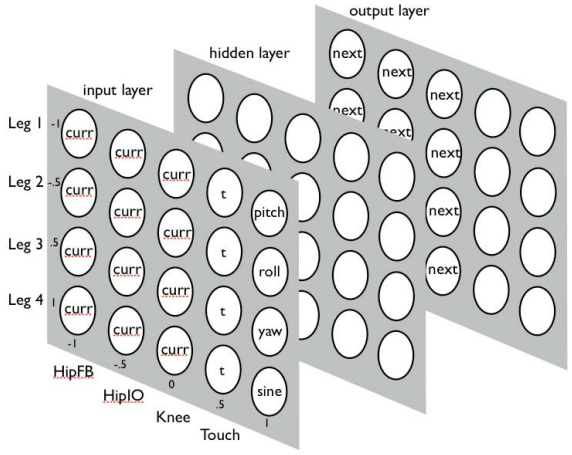

Fig. 2. The substrate configuration for both HyperNEAT and FT-NEAT treatments. The first four columns of each row of the input layer receive information about a single leg (the current angle of each of its three joints, and a 1 or 0 depending on whether the lower leg is touching the ground). The final column provides the pitch, roll and yaw of the torso as well as a sine wave. Evolution determines how to use the hidden layer nodes. The nodes in the first three columns of each of the rows in the output layer specify the desired new joint angle. The joints will move toward that desired angle in the next time step as described in the text. The outputs of the nodes in the rightmost two columns of the output layer are ignored.

torso, and a modified sine wave (which facilitated the production of periodic behaviors). The sine wave function is $\sin(120) \times \pi$. Multiplying by π facilitates the production of numbers that go from $-\pi$ to π, which is the range of the unconstrained joints. The constant 120 was chosen because it was experimentally found to produce fast, yet natural, gaits. While changing this constant can affect the types of gaits produced, doing so was never observed to alter any of the qualitative conclusions of this paper. Preliminary tests determined that the touch, pitch, roll, yaw, and sine inputs all improved the ability to evolve fit gaits (data not shown).

The outputs of the neural network were the desired joint angle for each joint. This was fed into a PID controller that simulated a servo. The controller subtracts the current joint angle from the desired joint angle. This difference was then multiplied by a constant force (2.0), and a force of that magnitude was applied to the joint such that the joint would move toward the desired angle. Such PID-based control systems have been shown to be effective [3], [10], [11].

The parameter configurations used for HyperNEAT and FT-NEAT are similar to those previously used [18], [20], and can be found at http://devolab.msu.edu/SupportDocs/EvolvingCoordinatedGaitsWithHyperNEAT. The experimental results reported here were found to be robust to moderate changes in the parameter settings, and to changes in the number and types of inputs. 50 trials were conducted for each encoding. Trials within a treatment differed only in their random number generator seed, which influenced stochastic events such as mutations. Each trial featured a population of 150 organisms, which is common for HyperNEAT experiments [20], and lasted 1,000 generations (~24-48 hours on 2.33 GHz Intel Xeon Linux machines).

Robots were evaluated in the widely used ODE physics simulator [www.ode.org]. The quadrupeds in this paper look like tables (Fig. 4), which is fitting because seminal work comparing generative encodings to direct encodings was performed on the evolution of static tables [24]. The rectangular torso of the organism is (in arbitrary ODE units) 0.15 wide, 0.3 long, and .05 tall. For a point of reference, the rightmost, shorter side of the robot from the viewer's initial perspective is designated as the robot's front. Each of four legs is comprised of three cylinders (length 0.075, radius 0.02) and three hinge joints. The first cylinder functions as a hip bone. It is parallel to the proximal-distal axis of the torso and barely sticks out from it. The second cylinder is the upper leg and the last cylinder is the lower leg. There are two hip joints and one knee joint. The first hip joint (HipFB) allows the legs to swing forward and backward (anterior-posterior) and is constrained to 180° such that at maximum extension it is parallel with the torso. The second hip joint (HipIO) allows a leg to swing in and out (proximal-distal). Together, the two hip joints approximate a universal joint. The knee joint swings forward and backward. The HipIO and knee joints are unconstrained.

Each organism was simulated for 6,000 time steps. Trials were cut short if any part of the robot save its lower leg touched the ground or if the number of direction changes in joints exceeded 960. The latter condition was an attempt to roughly reflect the physical fact that servo motors cannot be vibrated incessantly without breaking. The fitness of a controller was the following function of the maximum distance traveled in the X and Y dimensions: $2^{(X^2+Y^2)}$. An exponential function was used so that even small increases in the distance traveled would result in a sizable selective advantage.

IV. Results

The first question is whether HyperNEAT and FT-NEAT are able to make any reasonable progress given that they are facing the entire problem without simplification. Of further interest is the difference in performance, if any, between the two encodings. As Fig. 3 reports, while both encodings are able to improve over time, HyperNEAT vastly outperforms FT-NEAT in every generation ($p<.0001$ comparing the fitness of the best organism from each encoding for each generation. This and all future p values come from a non-parametric Wilcoxon rank sum test). While progress is still being made at the end of the run, it is seems unlikely that FT-NEAT would catch up and surpass HyperNEAT. Unfortunately, computational limitations prevented us from running these experiments longer. Even if FT-NEAT were able to close the gap with HyperNEAT, it is still worthwhile to note how much of an advantage HyperNEAT has in early generations.

The difference in fitness in the first generation, which is comprised of randomly generated organisms, is interesting. HyperNEAT begins with an advantage over FT-NEAT because even randomly generated CPPNs are sometimes

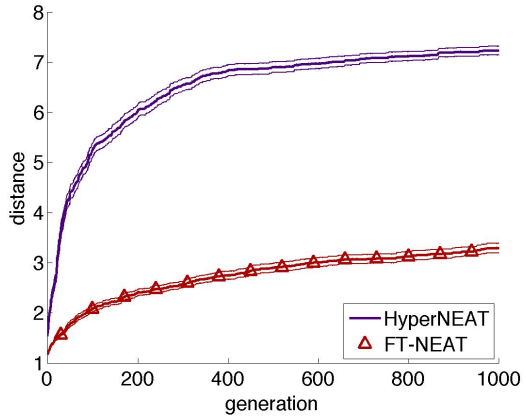

Fig. 3. The performance of HyperNEAT and FT-NEAT on the legged locomotion problem. Plotted for each treatment is the mean across 50 trials of the greatest distance away from the starting place arrived at by the best organism in that generation. Thin lines above and below the mean represent one standard error of the mean.

able to produce the coordination of legs that facilitates movement. It was observed that some of the randomly generated organisms in HyperNEAT displayed impressive amounts of coordination and immediately appeared to be on the road towards rudimentary locomotion. Randomly generated FT-NEAT organisms did not reach the same level of success.

We employ three methods of delving deeper into how HyperNEAT is able to outperform FT-NEAT. The first is to watch the gaits produced by the different encodings and make subjective judgments about what is going on. Videos of the evolved gaits are available at http://devolab.msu.edu/SupportDocs/EvolvingCoordinatedGaitsWithHyperNEAT. This procedure reveals that HyperNEAT quadrupeds tend to have a significant amount of coordination among all of their legs. Some organisms move all four legs in synchrony, which produces a coordinated, natural, repeated bounding across the landscape (Fig 4, top). This implies that HyperNEAT is discovering the four-legged regularity of the problem and is using the same neural module for each leg. Another gait that commonly evolved had three legs bounding nearly in synchrony and the fourth (a front leg) in opposite phase, resembling a horse gallop. This gait demonstrates that HyperNEAT is not constrained to simply repeat one theme for every leg. It makes an exception to the rule for the out-of-phase leg. Moreover, the movement of the out-of-phase leg resembles that of the other three, implying that HyperNEAT is simply using a variation on the theme used in the other legs. The ability to reuse modules with variation in different contexts is a desirable property in encodings [19]. A third gait was particularly interesting. It featured the back two legs bounding, as in the previous two gaits, but involved a running motion of the front two legs, with the front-left leg in opposite phase as the front-right (analogous to a human hopping forward with their hands placed on a jogger's shoulders). In general, the gaits produced by HyperNEAT looked smooth, effective, natural, and successful. Moreover, it appeared that most of these robots could continue running off into the distance because they had evolved a gait that perpetually repeated a basic movement (e.g., bounding).

In contrast, the gaits produced by FT-NEAT were mostly uncoordinated, erratic, and often consisted of a patchwork assembly of tripping, cartwheeling, and stumbling until the robot finally fell. Frequently, each leg acted independently (Fig 4, bottom). That is not to say that there was never any coordination among legs. It was not uncommon for two or even three legs to be synchronized, but usually the actions of another leg would eventually hamper the efforts being made by the coordinated legs. These results demonstrate that it was not impossible for FT-NEAT to coordinate its legs, but suggests that the direct encoding made it much harder to evolve coordinated behaviors. It was rare to see any basic move perpetually repeated. Most FT-NEAT trials ended with the robot falling over, as opposed to HyperNEAT trials, which were usually terminated only because time expired. A test of the reliability of each encoding is to watch the least fit gait of the 50 trial champions for each encoding: the worst HyperNEAT gait is coordinated and effective, whereas the worst FT-NEAT gait is discombobulated. Watching the videos of FT-NEAT organisms, one concludes that FT-NEAT is not really solving the legged locomotion problem

HyperNEAT

FT-NEAT

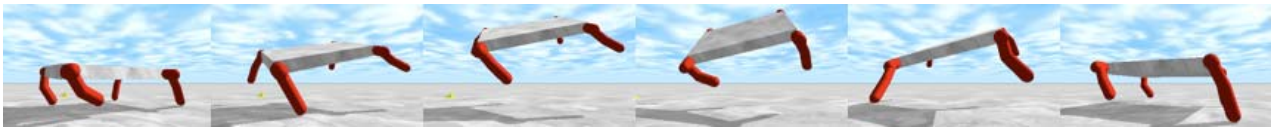

Fig. 4. Comparing gaits produced by HyperNEAT and FT-NEAT. Both organisms move from left to right. The first row shows the HyperNEAT robot with the median fitness. It was typical for HyperNEAT robots to have all of their legs coordinated, whether all legs were in phase (as with this robot) or with one leg in anti-phase. This series of moves would be repeated over and over in a stable, natural gait. The second row shows the FT-NEAT robot with the median fitness. It displays far less coordination among its legs, is less stable and does not end up in the same position as it started.

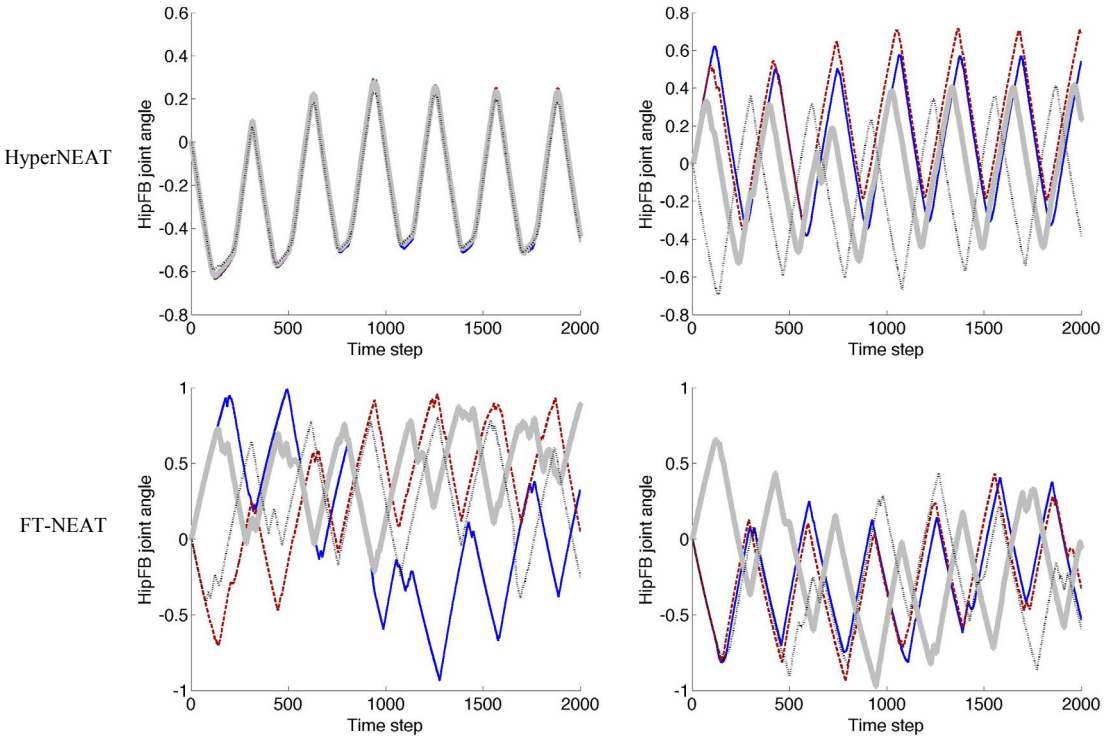

Fig. 5. Comparing HipFB joint angles observed in robots evolved using HyperNEAT (top row) and FT-NEAT (bottom row). The possible range for this joint was -0.5π to .5π, with the y-axis reporting radians from the initial down (0) position. For clarity, only the first 2,000 of 6,000 time steps are depicted. **Top left:** This was the most common gait produced by HyperNEAT and involved the organism bounding with all four legs moving in unison. It is depicted in Fig 4. **Top right:** This gait, which resembles the four-beat 'gallop' gait, has three legs in near synchrony and one of the front legs out of phase. **Bottom:** Two typical FT-NEAT runs. Note that some legs are synchronized but that other legs prevent the coordinated repetition of a pattern.

to an acceptable level. In contrast, HyperNEAT seems to solve the legged locomotion problem in all trials with a small variety of different solutions.

A second method for investigating how HyperNEAT was able to outperform FT-NEAT is to look at the angles of the leg joints during locomotion. This technique is a different way of estimating the coordination, or lack thereof, of the different legs under each encoding. Corroborating the subjective opinions formed by watching videos, the plots of each leg's HipFB joint from four typical runs shows that the legs in HyperNEAT organisms were far more coordinated than those in FT-NEAT organisms (Fig 5). While only the HipFB joint is shown, plots of the other two joints are consistent with these results.

A third way of learning how HyperNEAT was able to outperform FT-NEAT is to look at the effects that mutation and crossover had on organisms in the different encodings. It has been shown that, when simultaneously evolving the morphologies and controllers of creatures, mutations to a generative encoding were, on average, much less damaging than with a direct encoding. Moreover, the generative encoding produced far more beneficial mutations than the direct encoding control [4]. A similar study of the effects of genetic perturbations in this experiment tells a more complicated story. Organisms that were produced solely via mutation are analyzed separately from those produced solely via crossover. While the majority of organisms were both mutated and crossed over, focusing on the cases where only one genetic operator was involved clarifies the analyses. In the entire experiment, over three million organisms were produced via mutation only and nearly two million were produced via crossover only, in nearly equal parts per encoding, providing a substantial sample size.

Overall, the *magnitude* of the effect of mutations is much higher using HyperNEAT than using FT-NEAT (Fig 6). This difference likely reflects the fact that mutations to the CPPN encoding can bring about holistic changes to the entire neural network substrate in HyperNEAT, and thus have larger effects. Such holistic effects are more advantageous on highly regular problems, such as this one. Incredibly, HyperNEAT is consistently able to generate mutant offspring that are dozens of orders of magnitude more fit than their parents. HyperNEAT's ability on this front dwarfs that of FT-NEAT. Of course, many of these large mutational effects could be the result of a high-fitness organism suffering a mutation that makes it immobile one generation and then a compensatory mutation that approximately restores its original fitness in the next generation. Nevertheless, the fitness trajectories of the two encodings (Fig 2.) suggest that such a large variance in offspring fitness seems to greatly enhance evolutionary adaptation.

To quantify the differences, a mutation effect score of $\log_2(\text{child\_fitness}/\text{parent\_fitness})$ was used. A zero means the offspring fitness is identical to the parent fitness. Negative scores signify an offspring that was less fit than its parent and positive scores reflect the opposite. Interestingly,

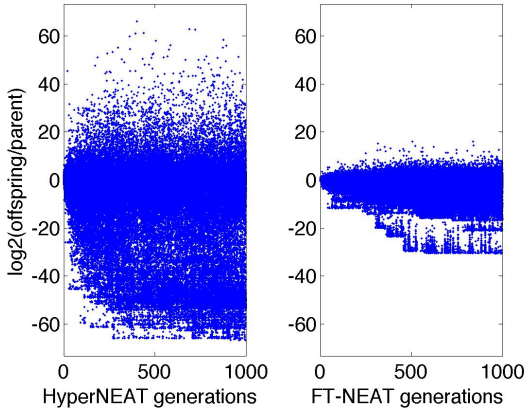

Fig. 6. The effect of mutations in HyperNEAT and FT-NEAT on organisms that were only mutated (i.e. were descendant from only one parent).

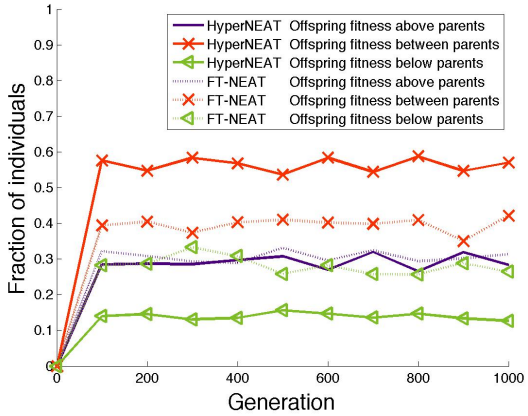

Fig. 7. The effect of crossover in HyperNEAT and FT-NEAT on organisms that resulted from crossover but not mutation. Although standard error bars are not shown for clarity, each plotted group is significantly different from every other (last generation p<.0001), except that the fraction of offspring better than both parents for each encoding was statistically indistinguishable (last generation p>.41).

the median mutation effect score is the same for both encodings, at approximately 0. The mean mutation effect score for HyperNEAT was -1.17, which was lower than the -0.61 for FT-NEAT. While the average mutation for HyperNEAT was thus more deleterious, evolution was capable of exploiting the steady production of extremely beneficial mutations to end up with higher overall fitness values. The distributions of mutations in these experiments were similar to those that arose when evolving static tables with generative and direct encodings [24]. That the distribution of mutations alone contributed to the success of HyperNEAT over FT-NEAT is supported by the fact that in an alternate experiment where only mutations generated offspring, HyperNEAT still vastly outperformed FT-NEAT, and a plot of mutational effect scores looked similar to Fig 6 (data not shown). The magnitude of mutational effects in Fig. 6 does change with time in both treatments because those mutant offspring that barely move from the starting place will count as a smaller fraction of the more fit parents of later generations. Unsurprisingly, the two distributions are significantly different statistically (p<.0001).

An analysis of crossover reveals further differences between the encodings (Fig. 7). The number of offspring that were better than both parents is statistically the same between the two encodings (final generation p>.41). However, HyperNEAT produces far more offspring with a fitness value between those of their parents than FT-NEAT (final generation p <.0001). The reason for this difference is not intuitively obvious. One might have expected the opposite, given that crossover in FT-NEAT could have combined successful leg controllers for different legs, whereas crossover in HyperNEAT is presumably more likely to disrupt a controller design that affects the controllers for multiple legs. This topic remains an open area for future research.

V. Discussion and Future Work

Both objective and subjective analyses found that HyperNEAT is much better at evolving legged locomotion controllers than FT-NEAT. The success of HyperNEAT should be expected for two reasons. Initially, as a generative encoding, HyperNEAT can reuse neural modules and thus more easily coordinate the behaviors of a robot's legs. Mutations to reused modules are also more likely to holistically change the gait controller, allowing evolution to spend more time testing coordinated gaits. These types of mutational effects are similar to what was found when evolving static tables, where mutations produced coordinated effects that frequently led to substantial fitness increases [24]. Here, however, the reused modules are not the body components themselves, but the neural networks driving such components. It will be interesting in future investigations to directly and quantitatively assess the modularity of neural network substrates produced by HyperNEAT in comparison to those produced by FT-NEAT. The findings in this paper, which reveal substantial coordination among the legs in HyperNEAT robots, coupled with previous demonstrations that modular neural networks are better at controlling legged creatures [7], suggests that HyperNEAT networks are far more modular. The results of this paper serve as another demonstration that generative encodings can outperform direct encodings on regular problems [4], [5], [18], [20].

A second reason for HyperNEAT's success may be its unique ability to exploit geometric aspects of the problem, such as symmetry [20], [23]. During this research many symmetries were observed in HyperNEAT gaits, including four-way symmetry (all legs in unison), left-right symmetry, front-back symmetry, and even combinations of these symmetries (e.g., the left and right legs operating in perfectly opposite phase while the back two legs acted in a completely different, but synchronized way). However, such symmetries might simply be more likely to arise when genetic modules are reused and could have cropped up using any generative encoding. In fact, such symmetries have been observed before when evolving moving creatures with generative encodings [4]–[6], raising the question of how

much, if at all, HyperNEAT's ability to exploit geometry aids it in these experiments. In reality, the setup in this paper is not especially conducive to the exploitation of geometry. For example, the joints are represented by approximated two-dimensional coordinates (Fig. 2), when they truly exist in a three-dimensional space. Furthermore, the two-dimensional layout does not accurately reflect all the geometric information it could. The HipFB and HipIO joints, for example, are extremely close together on the actual robot. In the coordinate system fed into the CPPN, however, the HipIO joint is listed as being as far away from the HipFB joint as the knee is from the HipIO joint. These deficiencies were intentional. This investigation focuses more on HyperNEAT's ability as a generative encoding on the locomotion task, instead of HyperNEAT's ability to exploit geometric information. For this reason, a somewhat geometrically naïve, 'out of the box' version of HyperNEAT was used. Separate investigations are already underway to explicitly test the extent to which HyperNEAT's ability to exploit geometries fuels its success, and what the effect of alternate geometric configurations of the nodes might be.

While HyperNEAT's many features make it difficult to isolate the contributions of each one, they also make it a promising encoding to investigate. HyperNEAT has yet to be tested on a wide range of problems. To date, however, it has performed quite well on tasks as diverse as visual recognition [20], controlling simple multi-agent systems [25], and evaluating checkers boards [23]. Additionally, the CPPNs underlying HyperNEAT produce complex, elegant, natural-looking images with symmetry and repetition [22]. All of these accomplishments suggested that HyperNEAT would excel at evolving controllers for legged robots. The results in this paper confirm that ability. HyperNEAT's success at evolving gaits for legged controllers is all the more impressive because it was able do so without the problem being manually decomposed or simplified. The encoding could handle the complexities of the entire problem because it discovered how to exploit the regularities of the problem. It was also observed in preliminary tests (not shown) that HyperNEAT could produce coordinated gaits with different types of input, and with more complicated morphologies (e.g., a six-legged, segmented robot). Based on the success reported here, which comes despite naïve geometric inputs and without a large amount of tuning, it is not unreasonable to predict that HyperNEAT, and variations of it, will contribute significantly to the science of automating the creation of controllers for complex, yet regular, devices, such as legged robots.

ACKNOWLEDGMENT

The authors thank Ken Stanley, Jason Gauci, Kosei Demura, Luis Zaman and reviewers for their contributions.

REFERENCES

[1] C. Ridderstrom, "Legged locomotion control - a literature survey," Tech. Rep. TRITA-MMK 1999:27, Dept. of Machine Design, Royal Inst. of Technology, S-100 44 Stockholm, Sweden, November 1999.

[2] D. Wettergreen and C. Thorpe, "Gait generation for legged robots," Proceedings of lEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 2, pp. 1413–1420, Jul 1992.

[3] G. Hornby, S. Takamura, T. Tamamoto, and M. Fujita, "Autonomous evolution of dynamic gaits with two quadruped robots," IEEE Transactions on Robotics, vol. 21, no. 3, pp. 402–410, 2005.

[4] G. S. Hornby, H. Lipson, and J. B. Pollack, "Generative representations for the automated design of modular physical robots," IEEE Trans. on Robotics and Automation, vol. 19, pp. 703–719, 2003.

[5] F. Gruau, "Automatic definition of modular neural networks," Adaptive Behaviour, vol. 3, no. 2, pp. 151–183, 1995.

[6] K. Sims, "Evolving 3d morphology and behavior by competition," Artif. Life, vol. 1, no. 4, pp. 353–372, 1994.

[7] V. K. Valsalam and R. Miikkulainen, "Modular neuroevolution for multilegged locomotion," in GECCO '08: Proceedings of the 10th annual conference on Genetic and evolutionary computation, (New York, NY, USA), pp. 265–272, ACM, 2008.

[8] J. C. Gallagher, R. D. Beer, K. S. Espenschied, and R. D. Quinn, "Application of evolved locomotion controllers to a hexapod robot.," Robotics and Autonomous Systems, vol. 19, no. 1, pp. 95–103, 1996.

[9] R. A. Téllez, C. Angulo, and D. E. Pardo, "Evolving the walking behaviour of a 12 dof quadruped using a distributed neural architecture.," in BioADIT, pp. 5–19, 2006.

[10] H. Liu and H. Iba, "A hierarchical approach for adaptive humanoid robot control," in Proceedings of the 2004 IEEE Congress on Evolutionary Computation, (Portland, Oregon), pp. 1546–1553, IEEE Press, 20-23 June 2004.

[11] K. Wolff and P. Nordin, "An evolutionary based approach for control programming of humanoids," in Proceedings of the 3rd International Conference on Humanoid Robots (Humanoids'03), (Karlsruhe, Germany), IEEE, VDI/VDE-GMA, 1-2 October 2003.

[12] M. Raibert, M. Chepponis, and H. B. Brown, "Running on four legs as though they were one," IEEE Journal of Robotics and Automation, vol. RA-2, pp. 70–82, June 1986.

[13] R. D. Beer and J. C. Gallagher, "Evolving dynamical neural networks for adaptive behavior," Adapt. Behav., vol. 1, no. 1, pp. 91–122, 1992.

[14] D. Filliat, J. Kodjabachian, and J.-A. Meyer, "Evolution of neural controllers for locomotion and obstacle avoidance in a six-legged robot," Connect. Sci., vol. 11, no. 3-4, pp. 225–242, 1999.

[15] J. Kodjabachian and J-A Meyer, "Evolution and Development of Neural Controllers for Locomotion, Gradient-Followig, and Obstacle-Avoidance in Artificial Insects." IEEE Transactions on Neural Networks. Vol 9:5, pp. 796-812. Sept. 1998.

[16] R. D. Nardi, J. Togelius, O. Holland, and S. M. Lucas, "Evolution of neural networks for helicopter control: Why modularity matters," 2006.

[17] S. Nolfi and D. Floreano, Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines. Bradford Book, 2004.

[18] J. Clune, C. Ofria, and R. T. Pennock, "How a generative encoding fares as problem-regularity decreases," in PPSN (G. Rudolph, T. Jansen, S. M. Lucas, C. Poloni, and N. Beume, eds.), vol. 5199 of Lecture Notes in Computer Science, pp. 358–367, Springer, 2008.

[19] K. O. Stanley and R. Miikkulainen, "A taxonomy for artificial embryogeny," Artif. Life, vol. 9, no. 2, pp. 93–130, 2003.

[20] K. O. Stanley , D. B. D'Ambrosio and J. Gauci, "A Hypercube-Based Indirect Encoding for Evolving Large-Scale Neural Networks." Artificial Life. 2009. To be published.

[21] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," Evol. Comput., vol. 10, no. 2, pp. 99–127, 2002.

[22] K. O. Stanley, "Compositional pattern producing networks: A novel abstraction of development," Genetic Programming and Evolvable Machines, vol. 8, pp. 131 – 162, June 2007.

[23] J. Gauci and K. O. Stanley, "A case study on the critical role of geometric regularity in machine learning," in AAAI (D. Fox and C. P. Gomes, eds.), pp. 628–633, AAAI Press, 2008.

[24] G. S. Hornby, "Functional scalability through generative representations: the evolution of table designs," Environment and Planning B: Planning and Design, vol. 31, pp. 569 – 587, July 2004.

[25] D. B. D'Ambrosio and K. O. Stanley, "Generative encoding for multiagent learning," in GECCO '08: Proceedings of the 10th annual conference on Genetic and evolutionary computation, (New York, NY, USA), pp. 819–826, ACM, 2008.