Generating Gaits for Physical Quadruped Robots: Evolved Neural Networks Vs. Local Parameterized Search

Jason Yosinski
Cornell University
239 Upson Hall
Ithaca, NY 14853, USA
yosinski@cs.cornell.edu

Jeff Clune
Cornell University
239 Upson Hall
Ithaca, NY 14853, USA
jeffclune@cornell.edu

Diana Hidalgo
Cornell University
239 Upson Hall
Ithaca, NY 14853, USA
djh283@cornell.edu

Sarah Nguyen
Cornell University
239 Upson Hall
Ithaca, NY 14853, USA
smn64@cornell.edu

Juan Cristobal Zagal
University of Chile
Beauchef 850
Santiago 8370448, Chile
jczagal@ing.uchile.cl

Hod Lipson
Cornell University
242 Upson Hall
Ithaca, NY 14853, USA
hod.lipson@cornell.edu

ABSTRACT

Creating gaits for legged robots is an important task to enable robots to access rugged terrain, yet designing such gaits by hand is a challenging and time-consuming process. In this paper we investigate various algorithms for automating the creation of quadruped gaits. Because many robots do not have accurate simulators, we test gait learning algorithms entirely on a physical robot. We compare the performance of two classes of learning gaits: locally searching parameterized motion models and evolving artificial neural networks with the HyperNEAT generative encoding. All parameter search methods outperform a manually-designed reference gait, but HyperNEAT performs better still, producing gaits nearly 9 times faster than the reference gait.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning

General Terms

Algorithms, Experimentation, Performance

Keywords

Evolving Quadruped Robotic Gaits, HyperNEAT

1. INTRODUCTION AND BACKGROUND

Various machine learning techniques have proved to be effective at generating gaits for legged robots. Kohl and Stone presented a policy gradient reinforcement learning approach for generating a fast walk on legged robots[**?**], which we implemented for comparison. Others have evolved gaits for legged robots, producing competitive results [**?**, **?**, **?**]. In fact, an evolved gait was used in the first commercially-available version of Sony's AIBO robot [**?**].

In this paper we compare the performance of two different methods of learning gaits: parameterized gaits optimized with six different learning methods, and gaits generated by evolving neural networks with the HyperNEAT generative encoding [**?**]. While some of these methods, such as HyperNEAT, have been tested in simulation [**?**], we investigate how they perform when evolving on a physical robot.

GECCO'11, July 12–16, 2011, Dublin, Ireland.
ACM 978-1-4503-0690-4/11/07.

	Average	Std. Dev.
Previous hand-coded gait	5.16	–
Random search	9.40	6.83
Uniform Random Hill Climbing	7.83	4.56
Gaussian Random Hill Climbing	10.03	6.00
Policy Gradient Descent	6.32	7.39
Nelder-Mead simplex	12.32	3.35
Linear Regression	14.01	12.88
Evolved Neural Network (HyperNEAT)	29.26	6.37

Table 1: The average and standard deviation of the best gaits found for each algorithm during each of three runs, in body lengths/minute. Videos of the gaits evolved for this robot can be viewed at http://bit.ly/geccogait

The metric for evaluating gaits was their average speed, which was measured as the Euclidian distance the robot moved during a 12-second run. In order to measure the start and end position in the same pose, and to ensure fair fitness evaluations with as little noise as possible, we linearly interpolated the motion of the robot between the ready position and the commanded gait for the first one second and the last two seconds of the 12 seconds of motion.

2. GAIT GENERATION AND LEARNING

2.1 Parameterized Gaits

By a *parameterized gait*, we mean a gait produced by a parameterized function $g(t;\theta)$. After some experimentation, we settled on one particular family of gaits consisting of a sine wave root pattern and five parameters: amplitude, period, and multipliers for front, left, and inner motors.

We evaluated a total of six different learning algorithms for the parameterized motion models. All methods were started at the same three initial θ vectors in the three runs. The six learning algorithms for the parameterized motion models are as follows:

Random: This baseline method randomly generates parameter vectors in the allowable range for every trial.

Uniform random hill climbing: This method chooses the next θ by randomly choosing one parameter to adjust and replacing it with a new value chosen with uniform probability in the allowable range.

Gaussian random hill climbing: Same as Uniform random hill climbing, except the next θ is generated by adding random Gaussian noise to the current best gait.

N-dimensional policy gradient ascent: We implemented Kohl and Stone's [**?**] method for local gradient ascent for gait learning with noisy fitness evaluations.

Nelder-Mead simplex method: The Nelder-Mead simplex method is a standard learning algorithm; see [**?**] for details.

Linear regression: This method fits a linear model from parameter vector to fitness and predicts the most promising direction for further evaluation.

2.2 HyperNEAT Gait Generation & Learning

The ANN configuration follows previous studies that evolved gaits with HyperNEAT in simulation [**?**]. The inputs to the ANN substrate were the current *commanded* angles of each of the 9 joints of the robot and a sine and cosine wave (to facilitate the production of periodic behaviors). The sine and cosine waves had a period of about half a second.

As with the parameterized methods, three runs of HyperNEAT were performed. The population size for HyperNEAT was 9 and runs lasted 20 generations. These numbers are small, but were necessarily constrained given how much time it took to conduct evolution directly on a real robot.

3. RESULTS AND DISCUSSION

The results for all gaits are shown in Table 1. The best overall gait for the parameterized methods was found by linear regression, which also had the highest average performance. The Nelder-Mead simplex also performed well. The other local search methods did not outperform random search; however, all methods did manage to explore enough of the parameter space to improve on the previous hand-coded gait in at least one of the three runs.

Overall the HyperNEAT gaits were the fastest by far, beating all the parameterized models when comparing either average or best gaits. We believe that this is because HyperNEAT was allowed to explore a much richer space of motions, but did so while still utilizing symmetries when advantageous. The single best gait found during this study had a speed of 45.72 body lengths/minute, 8.9 times faster than the hand-coded gait.

The evaluation of the higher-performing HyperNEAT gaits was more noisy than for the parameterized gaits, which made learning difficult (Figure 1). For example, we tested an example HyperNEAT generation-champion gait 11 times and found that its mean performance was 26 body lengths per minute ($\pm$ 13 SD), but it had a max of 38 and a min of 3. Many effective HyperNEAT gaits were not preserved across generations because if performance in one trial was poor, the genome was unlikely to be selected for.

4. CONCLUSION AND FUTURE WORK

HyperNEAT produced higher-performing gaits than all of the parameterized methods, perhaps because it can explore a much larger space of possibilities than the more restrictive 5-dimensional parameterized space. HyperNEAT gaits tended to produce more complex sequences of motor commands, with different frequencies and degrees of coordination, whereas the parameterized gaits were restricted to scaling single-frequency sine waves and could only produce certain types of motor regularities.

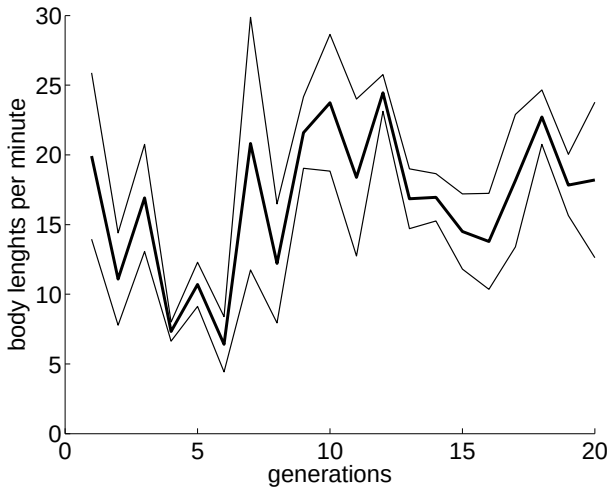

Figure 1: Average fitness ($\pm$ SE) of the highest performing individual in the population for each generation of HyperNEAT runs. The fitness of many high-performing HyperNEAT gaits were halved if the gait overly stressed the motors, so the true performance without this penalty would be much higher.

Because all trials were done in hardware, it was difficult to gather the many trials that would be necessary to properly rank the methods statistically. One direction for future work could be to obtain many more trials. However, a more effective extension might be to combine frequent trials in simulation with infrequent trials in hardware.

5. ACKNOWLEDGMENTS

This work was supported in part by NSF CDI Grant ECCS 0941561, NSF Creative-IT grant 0757478, and NSF Postdoctoral Research Fellowship DBI-1003220.