Not All Physics Simulators Can Be Wrong in the Same Way

Shane Celis[1], Josh Bongard[2]
University of Vermont

Abstract

Transferring designs in evolutionary robotics from simulation to reality remains problematic. It has been addressed by using quasi-static physics simulations, adding noise to encourage robustness, and evolving primarily in simulation then fine-tuning by evolving on actual hardware. This project explores a different idea: Multiple physics simulations may be used to infer whether a design is transferable. Two physics simulations are used to evolve a quadruped.

Introduction

Evolutionary Robotics is particularly susceptible to errors in physics simulations because evolution will exploit unrealistic physics that provide any robot an advantage. These "cheating" robots may score high fitness values and displace other robots in the population that had a "better" behavior, i.e., a behavior that was more realistic and therefore transferable.

Method

The robot used in this investigation is a quadruped walker as shown in Figure 4. It has a sensor on each leg that detect whether the leg is in contact with the ground. The controller is a feed-forward neural network (NN). The controller is represented by 32 real values that encode the weights of the NN. Three experiments were performed.

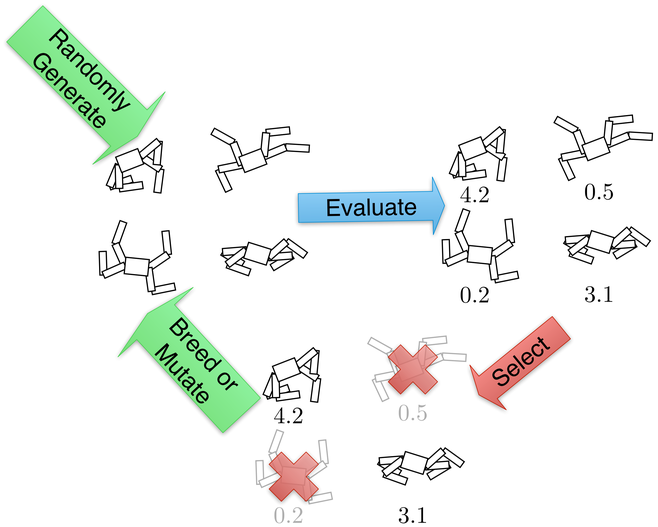

Figure 1: Basic outline of an evolutionary algorithm: Step 1) Randomly generate a population of robots. Step 2) Evaluate performance on a task, e.g., distance traveled. Step 3) Select best performers. Step 4) Breed or mutate selected individuals to create a new population. Repeat steps 2 through 4.

[1] shane.celis@uvm.edu, [2] jbongard@uvm.edu

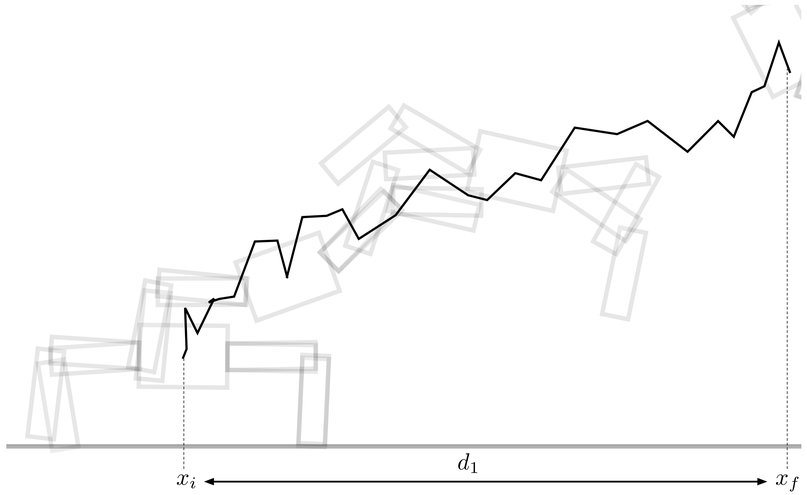

Figure 2: The problem: This robot is not content to crawl. It has decided to fly, physics notwithstanding. This robot is a cheater. How do we detect it?

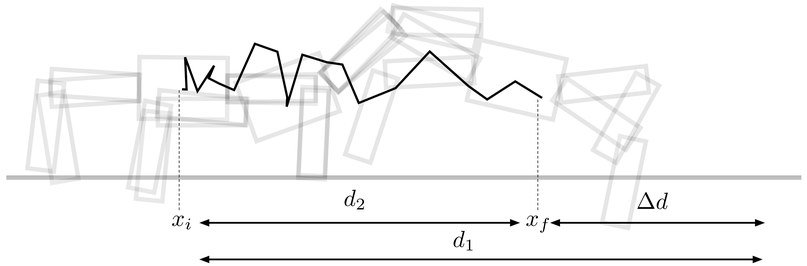

Figure 3: This is the same robot as in fig. 2 but evaluated in a different physics simulation. This physics simulation is not exploited by the robot.

1) **Experiment A** maximizes the distance in the first physics simulator d_1, and minimizes the distance difference from another physics simulation $\Delta d = |d_1 - d_2|$.
2) **Experiment B** maximizes distance d_1 and maximizes the difference Δd.
3) **Experiment C** only maximizes distance d_1 but Δd is shown for comparison.

Results and Discussion

Figure 5 shows the results. Experiment A shows that after a certain distance (10 conservatively) the d_1 and Δd become antagonistic. Experiment B seems to break into two clusters—one on the left, one on the right—that represent two different strategies: stay still in one physics simulation and move as far as possible in the other. Interestingly, 10 appears to be a boundary in both experiment A and B.

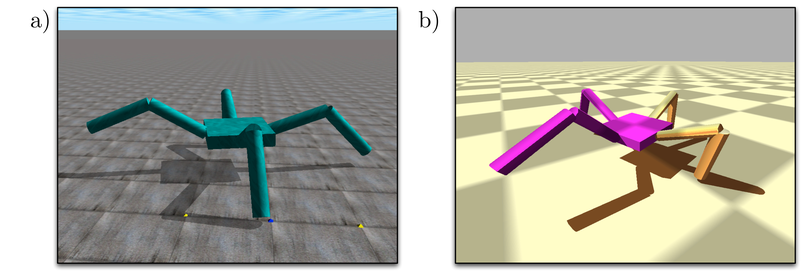

Figure 4: Quadruped in a) Open Dynamics Engine (ODE) and b) Bullet Physics Library.

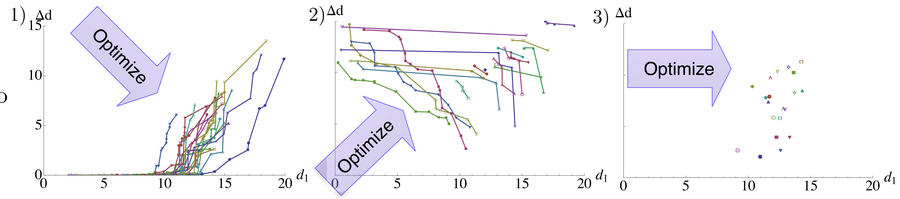

Figure 5: Results of 1) experiment A, 2) experiment B, and 3) experiment C. Twenty independent runs of NSGA-II were done for each experiment.

Future Work

This paper constrained itself to using the fitness value distance and its difference. However, using a more fine-grained time series approach might be more fruitful, perhaps using something similar to behavioral features as in [2].

Selected References

[1] Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm:NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2): 182–197, 2002.

[2] S. Koos, J. Mouret, and S. Doncieux. Crossing the reality gap in evolutionary robotics by promoting transferable controllers. *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pages 119–126, 2010.

[3] J. B. Pollack and H. Lipson. The GOLEM project: evolving hardware bodies and brains. In Evolvable Hardware, 2000. *Proceedings. The Second NASA/DoD Workshop on*, pages 37–42, 2000.