Evolutionary Robotics A/B Testing Final Project Report

Alex Friedrichsen

5/4/2022

Section One: Goals

Throughout the CS 206 Evolutionary Robotics curriculum we worked on homework assignments that gradually set up a basic simulation of a quadruped robot. We gradually implemented components of the simulation in Python using the Pybullet library for a physics engine, including a body, a neural network controller, an evolutionary algorithm, and several other specific functions. During the last portion of the semester we were given the freedom to improve upon the model we had each constructed up until that point and choose from a variety of projects ranging in difficulty from easy to hard. I had at first set my hopes upon the multi-objective optimization algorithm implementation, however ended up foregoing it due to the inherent complexity and setup required. Instead, I chose to implement the “body building” project in these last couple of weeks. I wanted to parametrically change the number of legs of the robot. Moreover, the topology of the robot would have to change in my vision, as the distribution of leg placements would look nicer and radially symmetrical with a spherical torso.

Conveniently, creating variants to test through A/B testing was a relatively straightforward step in the project. Initially, I set out to create a hexapod robot to competitively complement the incumbent quadruped on the fitness metric we had implemented through the homeworks - distance traveled from the spawn point of the robot. I chose to uphold the same fitness metric as distance traveled seemed a fitting measure for robots whose main structural difference was the number of appendages. We commonly think of legs as the primary motor for forward mobility: shouldn’t more legs change this forward mobility in all likelihood?

Section Two: Implementation

As I was coding the hexapod, I realized that the lines I was adding to the body building function in the “solution.py” file of my repository were repetitive in nature. This led me to consider a modular approach to the number of legs. If I could add two legs, why not four? Or six? Two main changes were required, and a smaller detail. The latter involved adding spheres into the pyrosim object constructor. The former were two loop structures: in “solution.py” inside Generate_Body(), a for loop replaced the creation of all upper, lower and corresponding joints of legs that would loop a number of times equal to the number of legs specified in the constants file. The distance between legs was calculated as 360 degrees over the number of legs, thus distributing the legs symmetrically around the now spherical robot torso. The size and relative position of legs and joints were untouched from the quadruped, with the exception of their looped implementation. The second loop structure was added in Generate_Brain(). Three for loops managed the creation of motor neurons, sensor neurons, and the sending of synapses. The fitness function worked pretty much “out of the box” with these changes.

Nevertheless, while the fitness function worked well and good in the search runs, I needed a way to compare the fitness of differently abled robots. In the “parallelHillClimber.py” file a fitness matrix was created and populated in the Evolve() function as the simulation ran with the individual fitness values of the parents for each generation. This matrix was saved at the end of each run to a fitness file (technically two files: one .npy file and a copy that was a .txt file) that corresponded to the fitness of that robot, with nomenclature determined by the number of legs. These results were explored in “plotFitnessValues.py” and will be disseminated in the following section.

Section Three: A/B/(C) results

The modular nature of my project allowed me to easily add a third candidate to the testing, the octopod as C test. In fact, any number of legs (constrained by the computational resources of the system) could've been tested to attempt to find the robot with optimal fitness values given the metric of distance from the start. However, in manual test runs a strong diminishing return was observed with additional legs being added to the robot past ten or so legs. Hence, this A/B/C test pits the familiar (now spherically torsoed) quadruped against new contenders Hexapod and Octopod. The tests were performed with a population of 10 and 10 generations: 10 x 10 is 100 individuals total per simulation run per number of legs. Additionally, the number of simulations was also increased to reduce variability with 2 simulations per number of legs resulting in two final fitness curve plots.

As can be seen in figures one and two, the fitness curves of all three multi-legged robots are plotted next to each other in one fitness plane. In addition, the standard deviations of the fitness values each generation are included in the plot to improve the statistical certainty of the results. After three generations the quadruped clearly outperforms both other robots, in both runs of the simulation. This result was consistent across several trial runs and we have sufficient evidence to claim that the quadruped evolves better to maneuver the furthest distance from the origin and is more fit in regards to our fitness function. However, several nuances are worth investigating in the results. Firstly, The standard deviations show some amount of overlap, especially in figure two, between the octopod and the quadruped. There is a large amount of deviation in general between generations and runs of the simulation, perhaps evidencing that robots can try something that end up moving them backwards at least initially. In places where the standard deviation lined up perfectly with the mean vectors, all the robots in the population

had exactly the same solution and thus the fitness values in that generation did not change and the means perfectly matched. Our other result is inconclusive: whether or not the hexapod has higher fitness than the octopod. While the fitness curve for the octopod is higher in figure two, figure one suggests their fitness is about equal. Over multiple simulations further than what is pictured here, the results were similarly inconclusive. It appears that the hexapod and octopod have similar fitness and the dominating robot depends on the particular optima found in those runs of the simulations.

Section Four: Reflection

While not directly part of my project, my best intentions were on completing the multi-objective optimization project alluded to in "an evolution revolution" final project description. This project proved to be more work than what I had time allocated for. If I were to revisit this project, I would like to implement a different evolutionary algorithm. I find it extremely interesting that there is no optimal evolutionary algorithm, and that one may continuously add more layers - sacrificing computing power for better solutions - ad nauseam. It was surprisingly easy to work change the neural controller to work with more legs, as I expected this part of the project to be the most difficult. It was also easier than expected to maintain the same fitness function. I struggled with understanding the relative position of joints and limbs, and I personally believe this is something that takes a lot of practice as our minds are not good at intuiting how these relative and absolute coordinates should interact and convert to the simulation. Finally, I believe that I would want to change the fitness function to something that emphasized more vertical movement of legs, to see if it in tandem improved distance achieved.

Appendix:

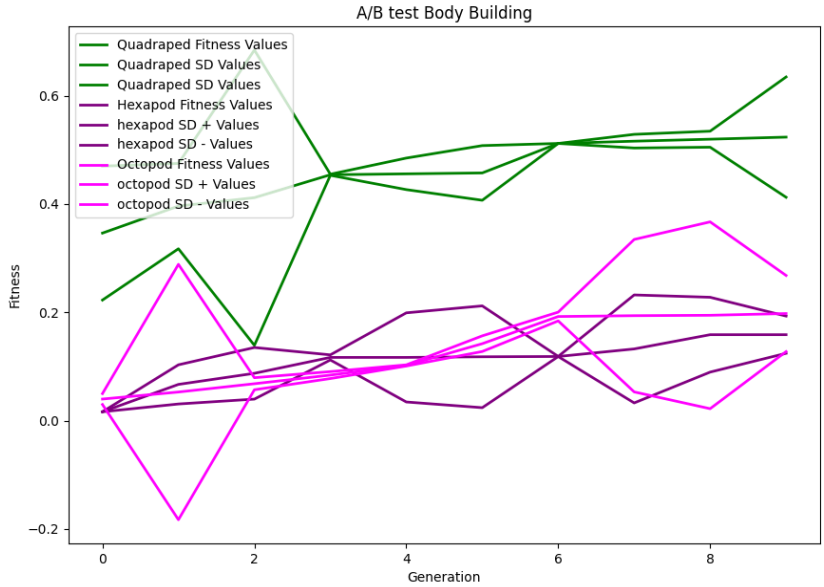

Figure 1: Fitness Curves and associated standard deviations for 10 generations of 10 size populations for quadruped, hexapod, and octopod, simulation run number one

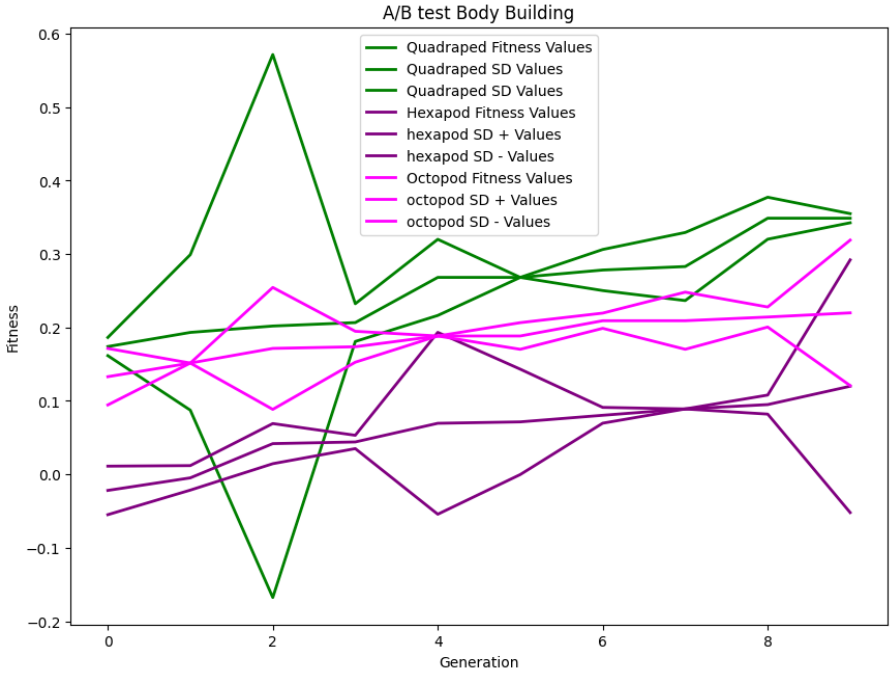

Figure 2: Fitness Curves and associated standard deviations for 10 generations of 10 size populations for quadruped, hexapod, and octopod, simulation run number two