Genetic Algorithms — Walking Agent:

For our project, we will parallelize a program that uses genetic algorithms to teach an agent to walk. A sample outline for the project is included below. Our project will use parallel programming to evaluate the performance of the agents in each generation and to select the best performers from each generation. Our project will most likely exploit a distributed memory model and a SIMD model. Running this simulation should be relatively cheap since all we need are the agents (a number decided by us), their parameters, and the relevant packages (i.e. Box2D). We would have to run the simulation multiple times, once for each of the generations. We anticipate a single run of the simulation to take only a couple of minutes. The total amount of memory required for this execution should be in the order of megabytes.

Example implementations:

1. https://rednuht.org/genetic_walkers/
    - Good visualization of the concept but likely more complex than ours
    - Implemented in JS; source code does not seem to be available
2. https://scholarworks.calstate.edu/concern/projects/qf85nj249
    - Goes into more detail about genetic algorithm implementation
    - Interacts with an external website via Java; no physics-simulation

Sample Outline:

1. *Define the Environment*: The first step is to define the environment in which the agent will learn how to walk. For this project, we will use a 2D environment space with a possible slope. The environment should include a starting position where the agent starts.
2. *Define the Genotype*: The next step is to define the genotype, which is a vector of values representing the parameters of an agent. For example, in the case of an agent trying to learn how to walk, the genotype can represent the torques in each of its joints. The length of the genotype would depend on the number of joints in the agent's body. These parameters can be represented as a vector of floating-point values. The first implementation above contains a large number of joints for feet, arms, elbows, and so on, in addition to legs. Our implementation will likely only consist of four joints, two for each leg.
3. *Initialize the Population (for a Generation of Parent Genotypes)*: The initial population of agents should be generated randomly, with each agent having a different genotype. This generation of agents will each pass through a fitness-evaluation algorithm.
4. *Evaluate Fitness*: The fitness function should be defined to evaluate the performance of each agent in the environment.
    a. Carrying out the steps of an agent: In our example, the performance of an agent would be calculated by measuring the distance that the agent travels after applying the torques to its joints. The movement and distance that the agent goes performs can be calculated (and simulated) using the C++ library Box2D, which provides a straightforward interface to handle rigid bodies in 2D
    b. The fitness function can be based on how large the steps of the agent are, how much energy it consumes, and how smoothly it moves. These attributes can collectively factor into a cost function that evaluates performance as a whole.
5. *Select Parents*: The agents with the highest fitness should be selected as parents for the next generation. This can be done using a tournament selection method.
6. *Crossover*: The crossover operator should be used to create new offspring from the chosen parents. This can be done by randomly selecting a subset of parameters from each parent and combining them to create a new genotype. This ensures that following generations explore genetic variation potential that previously led to the highest improvements in performance. The crossover may be done using an implementation similar to those seen in the C++ library *GALib* documentation (pp. 92).
7. *(Optional) Mutation:* This can be done by randomly perturbing some parameters in an agent's genotype.
8. *Repeat:* Steps 4-7 should be repeated for a fixed number of generations, or until the agents have converged to a satisfactory solution, which we can decide as a certain distance to be traveled, etc.