Course: Introduction to Artificial Intelligence

Instructor: Dr. Syed Ali Raza

Spring 2025

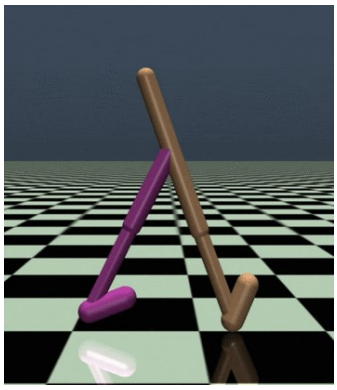
Institute of Business Administration, Karachi

Assignment #2

Evolutionary Optimization of Keyframe-Based Locomotion in a Mujoco Environment

Overview

The goal of this assignment is to develop an evolutionary algorithm (EA) to optimize a keyframe-based walking behavior for the **Walker2D** robot in the **MuJoCo** environment (via Gymnasium). The Walker2D domain is a bipedal locomotion environment in the MuJoCo physics engine, available through Gymnasium. It consists of a 2D robot with six joints (two in each leg and one in each foot) that must learn to walk efficiently. The agent receives torque inputs for its joints and is rewarded for forward movement while being penalized for excessive energy use or falling. This environment is widely used in reinforcement learning and evolutionary optimization for testing locomotion policies. For further information, visit https://gymnasium.farama.org/environments/mujoco/walker2d/.



In this assignment, you will:

- Define keyframes as a sequence of torque values and time durations.
- Implement an evolutionary algorithm to optimize the keyframes.
- Evaluate the performance of the optimized keyframes in the environment.

Keyframes in Walker2D

- **A keyframe** consists of:
  - **6 torque values** (one per joint).
  - **1 duration value** (number of timesteps for which the keyframe is applied).
- **A keyframe sequence** consists of **K keyframes** applied cyclically.
- The EA will **optimize** the torque values and the duration of each keyframe to maximize the **cumulative episode reward**. An episode consists of the steps taken from the start till a termination condition is met.

Gymnasium Installation:

First, install the gymnasium on your computers. There are many tutorials available online to show how to install gymnasium. For example, here is a good one for the Windows users, https://www.youtube.com/watch?v=gMgj4pSHLww&list=PL58zEckBH8fCt_lYkmayZoR9XfDCW9hte&index=1. Note, there are different environment types supported by gymnasium but you don't need to install all of them. You can skip installation steps for the Box2d environment (which is a bit complicated). This assignment needs the Mujoco environments to be installed. However, you are free to install others.

You can test the installation using the script *installation-test.py*, attached with the assignment.

Gymnasium main functions:

Gymnasium is a Python library for developing and benchmarking reinforcement learning (RL) algorithms. It provides a collection of standardized environments (e.g., robotics, control tasks, and games) where agents interact by taking actions and receiving rewards. However, it can be used for an application of the evolutionary algorithm. An example of how a environment from Gymnasium can be adopted for EA is shown in the starter code (*kf-walker2d.py*) whose description is given in the following section.

Following is the description of the Gymnasium methods you will need in this assignment.

reset(seed=None)

- Initializes or reinitializes the environment.
- Returns the **initial observation** and additional info.
- The optional seed ensures reproducibility.

step(action)

- Executes the given **action** in the environment.
- Returns:
  - **Next observation** (state after action).
  - **Reward** for the action.
  - **Done flag** (True if the episode ends).
  - **Truncated flag** (True if the episode ends due to time limits).
  - **Info dictionary** with extra details.

render(mode="human")

- **Visualizes** the environment (e.g., displays the agent's movements).
- Common modes: "human" (real-time rendering, used in this assignment), "rgb_array" (image output).

You can find more information about Gymnasium on this page https://gymnasium.farama.org/introduction/basic_usage/.

Starter Code:

You can find the code available in the file *kf-walker2d.py* helpful in completing this assignment. This code shows:

- How to import libraries needed to solve the problem.
- A simple example of a sequence of keyframes defined as a list of numpy arrays. Note, the first six values in a keyframe (a numpy array) are the torque values and the last value is the duration or the number of steps for which this keyframe is executed.
- How an environment is reset (which provides a fresh start).
- Next, there is a code to execute the keyframes one by one for a number of steps which is given for each keyframe. This enables a cyclic motion which can produce a bipedal walk.
- The quality of a solution (the keyframes) can be found by summing the rewards for all the steps taken from the start till the end (an episode). In the code, the *episode_reward* variable calculates this score.

Other Instructions:

- You can find a simple example of keyframes provided in the *kf-walker2d.py* file (see *keyframes* variable). This solution can serve as an initial solution. You are free to create your own solutions for the initial population of EA.
- Exclude the render_mode="human" argument from the gym.make() function call to stop visualizing the domain. The robot will still run as usual in the background, but it saves time needed to display the moves. Hence, it speeds up the simulation. In the testing, you can reinitialize the environment with the render_mode="human" argument.
- To find better solutions you can change the number of keyframes, the number of generations, the number of chromosomes, mutation type, crossover type, and the selection strategy.

Deliverables:

- Code with the implementation of EA to find the optimized keyframes.
- You need to apply different combinations of the selection methods taught in the class for the parent selection and the survival selection. Compare and report the results.
- The best keyframes found in the experiments. Your solution should achieve more than 600 in episode_reward. Provide this in a text file with the first line containing the number of keyframes and the following lines containing the comma separated values of keyframes (one keyframe per line). For example:

  3

  -0.01, -0.01, 0.00, 0.02, 0.17, 0.00, 25

  0.18, -0.08, 0.22, 0.12, 0.27, 0.10, 45

  1.00, -0.16, -0.87, 0.15, 0.47, -0.39, 8
- Plot the episode_reward over the generations for the best solution found.