1 2 9 0 FACULDADE DE CIÊNCIAS E TECNOLOGIA UNIVERSIDADE D COIMBRA

dei departamento de engenharia informática

# Evolutionary Computation

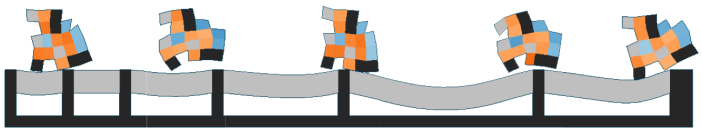
**2024/2025**

## Survival of the Softest: Evolving Adaptive Soft Robots

**Nuno Lourenço**

**João Correia**

March 5, 2025



python 3.10

# 1 Introduction

We present here the practical project, part of the students' evaluation process of the Evolutionary Computation course of the University of Coimbra. This work is to be done autonomously by a group of **two** students. The deadline for delivering the work is **11 of May** via Inforestudante.

The quality of your work will be judged as a function of the value of the technical work, the written description, and the **public defence**. All sources used to perform the work (including the code) must be clearly identified. The document may be written in Portuguese or in English, using a word processor of your choice[1]. The written report is limited to **10** pages long, but in special, justified, cases (e.g., the need of presenting many images and/or tables), that number may be increased accordingly. The document should be well structured, including a general introduction, a description of the problem, the experimental setup, the analysis of the results, and a conclusion. The report should follow the Springer LNCS format. The Latex and Word templates are available in the Support Material of the course. The final mark will be given to each member of the group individually.

To do the work the student may consult any source he/she wants. Nevertheless, plagiarism will not be allowed and, if detected, it will imply failing the course. While doing the work and when submitting it, you should pay particular attention to the following aspects (whose relative importance depends on the type of work done):

relatório é essencial
explorar coisas no projeto (não há uma única solução)
max 10 pags

# 2 Problem Statement

Soft robotics is an exciting field that explores the design of robots with flexible and adaptive structures. Unlike traditional rigid-body robots, soft robots use deformable materials, allowing them to exhibit more versatile and adaptive behaviors in dynamic environments. EvoGym[2] is a physics-based simulation environment specifically designed for evolving and testing soft robots.

EvoGym provides a voxel-based framework where robots are composed of individual unit cells (voxels) arranged in a 2D grid. These voxels define the physical structure and behavior of the robot, with different types influencing the robot's movement and interaction with the environment. The primary voxel types in EvoGym include (Fig. 1):

- **Solid Voxel:** A non-deformable voxel that serves as a rigid structure, providing stability and support (Rigid).

[1] We strongly suggest the use of LaTeX.

[2] https://evolutiongym.github.io

spft robots simples mas quando agregados sao complexos à base de 4 blocos:

por um agente a andar, galopar, saltar e passar por entre obstaculos

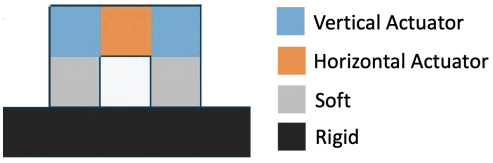



Figure 1: Types of Voxels used to built the robot

- **Soft Voxel:** A deformable voxel that compresses and expands under force, adding flexibility to the robot (Soft).
- **Vertical Actuators:** These actuators expand and contract along the vertical axis (up and down). They are crucial for tasks involving jumping, hopping, or lifting parts of the robot off the ground.
- **Horizontal Actuators:** These actuators expand and contract along the horizontal axis (left and right). They play a significant role in generating forward or lateral movement, as seen in crawling, slithering, or undulating locomotion strategies.

The interactions between these voxel types, combined with appropriate control strategies, allow for the evolution of a wide range of locomotive behaviors. EvoGym enables users to define and evolve both the structural morphology and control strategies of soft robots through evolutionary algorithms. By leveraging mutation, selection, and crossover mechanisms, students can explore how different designs emerge through iterative optimization, leading to novel and efficient robotic behaviors.

The goal of this assignment is to implement an evolutionary algorithm to design soft robots using EvoGym. Students will follow a structured approach, first evolving the structure of a robot for a fixed controller, then evolving a controller for a fixed structure, and finally co-evolving both the structure and controller simultaneously. This exercise will deepen the understanding of evolutionary computation, optimization techniques, and their applications.

evogyn -> heuristicos,...

# 3 Objective

In general terms, the main objective is to develop an Evolutionary Algorithm that is able to discover a structure and policy that allows the agent to move from the starting point to the goal using the minimal number of steps. To fulfill this task, your task entails the following aspects:



trabalho tem 3 partes:
algaritmos evolucionarios
estruturas e politicas de controladores

1. fixar o controlador, evoluir estrutura
2. etsrutura fixa, evoluir controlador
3. evoluir ambos

- Develop an Evolutionary Algorithm, which requires you to tackle the following design issues:
  - Representation
  - Initialization Procedure
  - Variation Operators (crossover and mutation)
  - Parent Selection Mechanisms
  - Survivor Selection Mechanisms
  - Fitness Function

As we discussed in the class the fitness function is crucial for the success of the algorithm. You should experiment several fitness functions, taking into account the main goal of the project.

The project is divided in three stages:

1. Evolving the Structure for a Pre-Specified Controller
2. Evolving the Controller for a Pre-Specified Structure
3. Evolving Both Structure and Controller

cenarios diferentes a ser testados

## 3.1 Evolving the Structure for a Pre-Specified Controller

Implement an evolutionary algorithm to optimize the structure of a soft robot while keeping the controller fixed. Remember that you should design a fitness function that evaluates the performance of different structures in a given environment. Use mutation and crossover to explore different structural configurations. Run the evolution process for a set number of generations and analyze the results.

### 3.1.1 Tasks

The evolutionary process should be carried out for the **Walker-v0**, **BridgeWalker-v0** tasks in EvoGym. These tasks represent different locomotion challenges that require adaptive structures and controllers for successful performance. You should design robots with a size of (5x5).

- **Walker-v0** focuses on forward locomotion on flat terrain, requiring an efficient gait for movement.
- **BridgeWalker-v0** consists of soft ground that deforms under the robot, necessitating stability and adaptive movement strategies to prevent sinking.

### 3.1.2 Approach

For the tasks requiring evolution of the robot structure while keeping the controller fixed, predefined actuation patterns will be used. The pre-specified controls represent different locomotion strategies commonly observed in biological systems. You can use **one** of the following three control schemes:

- **Alternating Gait (Legged Locomotion):** This control pattern alternates the expansion and contraction of active voxels in a cyclic manner, mimicking the gait of legged animals such as quadrupeds. The left and right sides of the robot actuate in opposite phases to generate forward movement.
- **Sinusoidal Wave (Snake-like Motion):** In this control scheme, the actuation follows a sinusoidal function along the body of the robot. This results in a smooth, undulating motion similar to how snakes or eel-like creatures propel themselves.
- **Hopping Motion (Bipedal or Kangaroo-like Locomotion):** This actuation strategy synchronizes the expansion and contraction of certain active voxels to generate periodic jumping motions. This resembles the hopping behavior of bipeds or kangaroo-like movement patterns.

The following Python implementations define these control schemes (we have also provided a script with their implementation):

Listing 1: Predefined Control Functions

```
import numpy as np

def alternating_gait(action_size, t):
    """Alternates actuation to
    mimic a walking gait."""
    action = np.zeros(action_size)

    # Alternate expansion & contraction every 10 timesteps
    if t % 20 < 10:
        action[:action_size // 2] = 1   # Front half expands
        action[action_size // 2:] = -1  # Back half contracts
    else:
        action[:action_size // 2] = -1  # Front half contracts
        action[action_size // 2:] = 1   # Back half expands
```

```
    return action

def sinusoidal_wave(action_size, t):
    """Generates a wave-like
    motion pattern for snake-like robots."""
    action = np.zeros(action_size)
    for i in range(action_size):
        action[i] = np.sin(2 * np.pi * (t / 20 + i / action_size))
# Sin wave pattern
    return action

def hopping_motion(action_size, t):
    """Makes the robot jump forward
    using periodic full-body contraction and expansion."""
    action = np.zeros(action_size)
    if t % 20 < 10:
        action[:] = 1  # Expand all active voxels
    else:
        action[:] = -1  # Contract all active voxels
    return action
```

Each of these control schemes provides a different way for the robot to generate movement. By evolving only the structure while keeping **one** of these actuation patterns fixed, the algorithm will explore structural designs best suited to the specific locomotion strategy.

## 3.2 Evolving the Controller for a Pre-Specified Structure

Implement an evolutionary algorithm to optimize the control strategy of a soft robot while keeping its structure fixed. The goal is to evolve a controller that maximizes performance in a given environment. The controller will dictate how the active voxels actuate over time, impacting the robots ability to move effectively. A well-designed fitness function should evaluate how well the controller enables the robot to complete its task. Use mutation and crossover to explore different control configurations. Run the evolution process for a set number of generations and analyze the results.

The structure that you should use is presented in the list bellow, and corresponds to the robot portrayed in figure :

Listing 2: Robot Structure Definition

```
robot_structure = np.array([
```

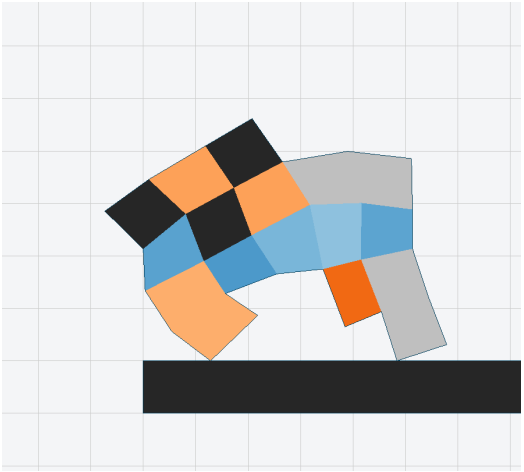


Figure 2: Robot Fixed Structure

```
        [1,3,1,0,0],
        [4,1,3,2,2],
        [3,4,4,4,4],
        [3,0,0,3,2],
        [0,0,0,0,2]
])
```

### 3.2.1 Tasks

The evolutionary process should be carried out for the **DownStepper-v0**, and **ObstacleTraverser-v0** tasks in EvoGym. These tasks represent different locomotion challenges that require adaptive controllers for successful performance.

- **DownStepper-v0** challenges the robot to descend a series of steps while maintaining stability and preventing excessive tumbling.
- **ObstacleTraverser-v0** involves navigating uneven terrain with obstacles, requiring adaptive control strategies to handle varying surface conditions and avoid getting stuck.

### 3.2.2 Approach

The controller will be represented as a neural network that maps sensory inputs to actuation outputs. The evolutionary algorithm will optimize the neural networks weights and biases to improve performance in the given tasks. The following steps outline the process:

- **Neural Network Architecture:** The controller should be implemented as a feedforward neural network with an input layer (sensory data), one or more hidden layers, and an output layer (actuation values for active voxels). You can use the PyTorch example provided.
- **Mutation and Crossover:** Evolutionary operators will be applied to the neural network weights and biases. Mutations will introduce small random changes, while crossover will combine successful controllers to explore new strategies.
- **Fitness Evaluation:** Each evolved controller will be tested in simulation, where its effectiveness will be measured based on task-specific criteria such as distance traveled, stability, and energy efficiency.
- **Selection:** The best-performing controllers will be selected to form the next generation, ensuring gradual improvement in locomotion capabilities.

## 3.3 Evolving Both Structure and Controller

In this final stage of the assignment, the goal is to evolve both the structure and controller of a soft robot simultaneously. By co-evolving morphology and control, the evolutionary algorithm can explore a broader space of possible robot designs, leading to the emergence of more optimized and adaptable solutions for various tasks.

Unlike the previous sections where either the structure or controller was fixed, this section requires the joint optimization of both. The structure will determine the physical form of the robot, while the controller will govern actuation patterns to achieve locomotion. The evolutionary algorithm will explore and optimize these two components together, allowing the evolution of robots that are well-adapted to their respective environments.

### 3.3.1 Tasks

The evolutionary process should be carried out for the **GapJumper-v0**, and **CaveCrawler-v0** tasks in EvoGym. These tasks represent challenges that require coordinated evolution of structure and control for successful performance.

- **GapJumper-v0** challenges the robot to clear gaps between platforms, necessitating an effective jumping mechanism and precise actuation timing.

- **CaveCrawler-v0** requires the robot to navigate through a confined and irregular cave-like environment. The task demands a structure capable of maneuvering through tight spaces while maintaining forward progress.

### 3.3.2 Approach

The co-evolution of structure and controller will be implemented using an evolutionary algorithm that encodes both aspects in a single genotype. The following steps outline the process:

- **Genetic Representation:** The genotype will consist of two components: one encoding the robot's structural morphology and the other defining the neural network-based controller.
- **Mutation and Crossover:** Evolutionary operators will modify both structure and controller. Structural mutations may add or remove voxels, while controller mutations adjust neural network weights and biases. Crossover can be applied to exchange information between successful individuals.

The results of this evolution should provide insights into the interactions between structure and control, demonstrating how co-evolutionary approaches can lead to highly specialized robotic designs.

# 4 Experimentation

cada experiencia:
conseguir ezcrever qual é o setup, que operadores que funcao de avaliacao; várias inicializacoes

Regarding each experiment you should consider:

- An bio-inspired algorithm that encodes both structure and control parameters.
- Initialize a diverse population of robot structures and controllers, allowing for broad exploration of potential solutions.
- Apply evolutionary operators to morphology and/or control, ensuring variation and innovation in the search process.
- Conduct multiple evolution runs to analyze convergence trends and diversity of evolved strategies.

By following these steps, students will ensure that their environment is correctly set up for running the evolutionary algorithm and EvoGym simulations.

When performing the experiments, you should take into account the following points:

- description of the general architecture of the algorithm used;
- description of the representation, variation operators and fitness function used;
- description of the experiment, including a table with the parameters used;
- description of the measures used for the statistical work: quality of the final result, efficacy, efficiency, diversity, or any other most appropriate;
- the minimum number of runs (different random seeds) is 5;
- the comparisons must be fair, i.e., you should give to all variants the same resources (number of evaluations);.

populacoes com numeros semelhantes, iteracoes, wuantas vezes fitness corre

Do not forget, in addition to what was just said, that it is fundamental: (1) to do a correct statistical analysis, including the reasons for choosing a particular method; (2) to conduct an informed discussion about the results obtained; (3) to show the advantages of the chosen alternative.

# 5 Software Installation Requirements

To ensure that all necessary dependencies are installed, students should run the provided installation script before starting the assignment. This script will automatically install all required Python packages and libraries needed for running EvoGym simulations.

## 5.1 Installation Instructions

Students should execute the following command in their terminal or command prompt to install the required dependencies:

Listing 3: Running the installation script

```
python install_requirements.py
```

This script will check for missing dependencies and install them automatically. If any issues arise during installation, students should verify that they have Python installed and that they have the necessary permissions to install packages.

# 6 Conclusion

A few short comments. First, the control of the progression of your work will be done during the classes (T and PL). Moreover, you can discuss eventual problems by presenting yourself during office hours. Second, the projects reflect for the most part your actual knowledge. Third, we try to balance the difficulty of all the work, but we are aware that this is not an easy task and it is somehow a subjective matter. Fourth, we try to ask a workload compatible with the value of the work for the final mark.

Methodological issues, like the statistical background, were elucidated during the previous lectures. You may use the statistical tool you feel at ease with, including the Python code that was provided. Finally, even if this is a work that asks you to do simulations and analyze the results, i.e., it has a practical flavor, there is however a theory behind the work, and you are advised to consult the necessary literature.

Good luck!