Evolutionary Computing Assignment 2
- Robot Neuroevolution -

READ THIS WHOLE DOCUMENT BEFORE STARTING THE ASSIGNMENT!!!

Introduction

The objective of this assignment is to gain experience in Evolutionary Computing and Evolutionary Robotics. To do this, you will develop, apply, and compare evolutionary algorithms for the task of robot neuroevolution (evolving a robot's brain) on the Ariel platform. The assignment is to be done in groups of three or four. **You need to deliver your code and a three-page report on your work and findings.**
Submission deadline is **Monday, September 22, at 9:00 in the morning**, and it is **NOT NEGOTIABLE!!!**

Each day of delay in the submission will result in a 0.6-point deduction from your grade (out of 15). This means that a submission done at 9:01 on September 22nd caps at a grade of 9/10.

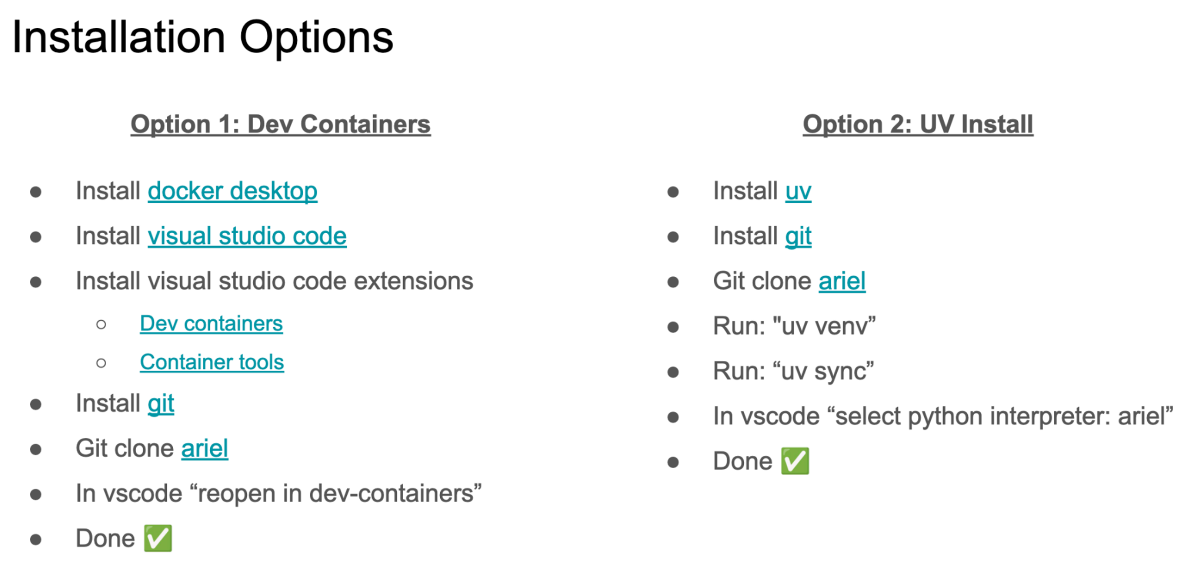
Resources

All resources you need are available in Ariel. For example, many of the features you have gotten used to from the N-Queens assignment are. Additionally, you can utilise the information and tips provided during the weekly practical sessions.

Preparation

- If you haven't done so already, you should install ARIEL (https://github.com/ci-group/ariel).

- We suggest you use UV (https://github.com/astral-sh/uv) for this project. The library was created with it.
- After installing, you should test the framework using the example files we have provided. This is shown in the installation guide as well. The most important example is “***examples/_robogen_build.py***”, which tests all the components that you will need for A2 (other examples may have features you do not yet need).



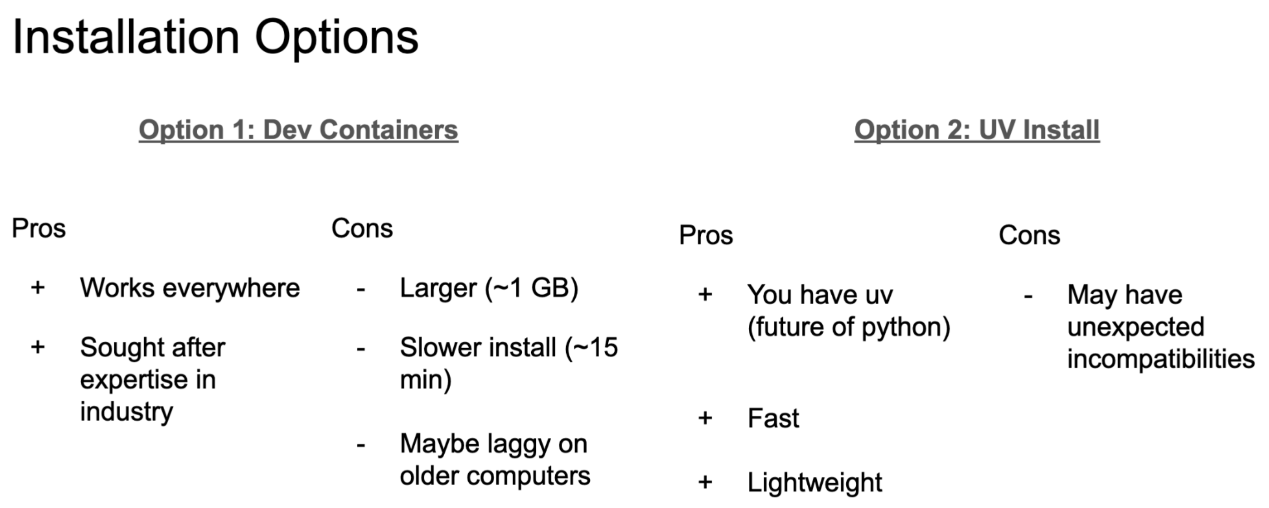
Installation Options

Option 1: Dev Containers

- Install docker desktop
- Install visual studio code
- Install visual studio code extensions
  - Dev containers
  - Container tools
- Install git
- Git clone ariel
- In vscode “reopen in dev-containers”
- Done ✅

Option 2: UV Install

- Install uv
- Install git
- Git clone ariel
- Run: "uv venv”
- Run: “uv sync”
- In vscode “select python interpreter: ariel”
- Done ✅



Installation Options

Option 1: Dev Containers

Pros	Cons
+ Works everywhere	- Larger (~1 GB)
+ Sought after expertise in industry	- Slower install (~15 min)
	- Maybe laggy on older computers

Option 2: UV Install

Pros	Cons
+ You have uv (future of python)	- May have unexpected incompatibilities
+ Fast	
+ Lightweight	

Our suggestion: **start with “Option 2: UV install”**, if that does not work *WithRelativeEase*™, switch to “Option 1: Dev Containers”.
Neither may work *WithRelativeEase*™, if so, please come to the TA sessions, we will get you sorted.

Task

The task of this project is to implement an evolutionary algorithm to evolve the brains of a robot. This can be the network architecture, weights, or both. Ariel provides a variety of environments (such as smooth ground, rugged terrain, or both) and tasks (walking the furthest or spinning around); "***src / ariel / simulation / environments***" and "***src / ariel / simulation / tasks***" respectively.

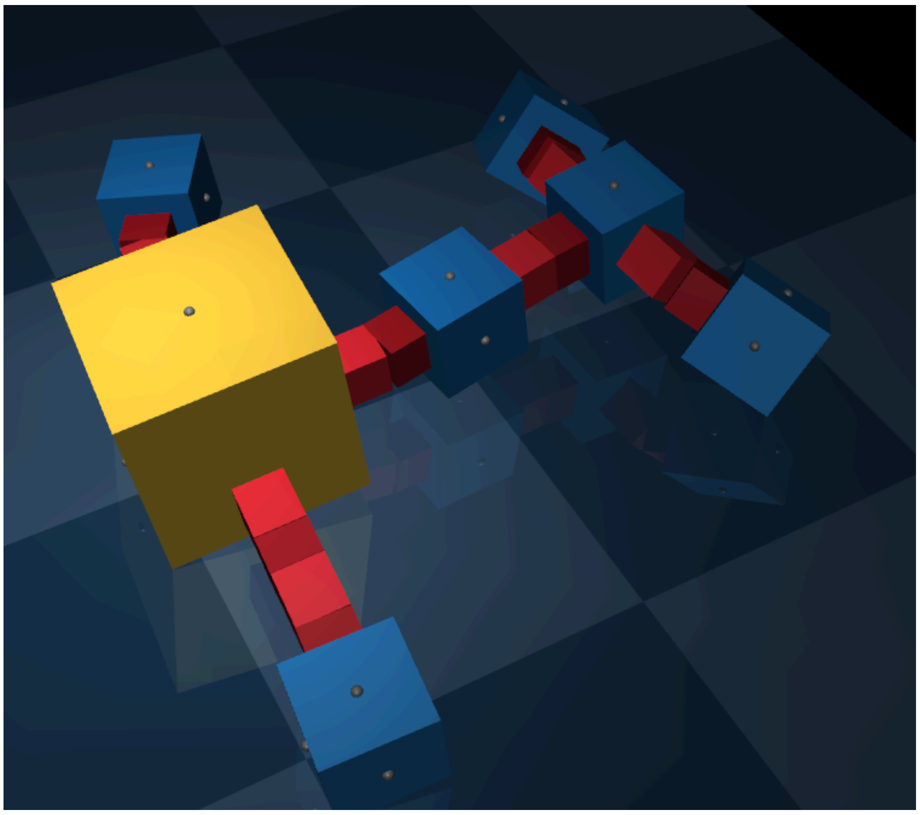


Figure 1. Gecko robot in ARIEL

The goal of this assignment is robot neuroevolution. This means that you will have to **evolve the network** controlling the movements of the Gecko robot provided in ARIEL.
The robot body can be imported from the ARIEL `prebuilt_robots` module under: `ariel.body_phenotypes.robogen_lite.prebuilt_robots.gecko`.

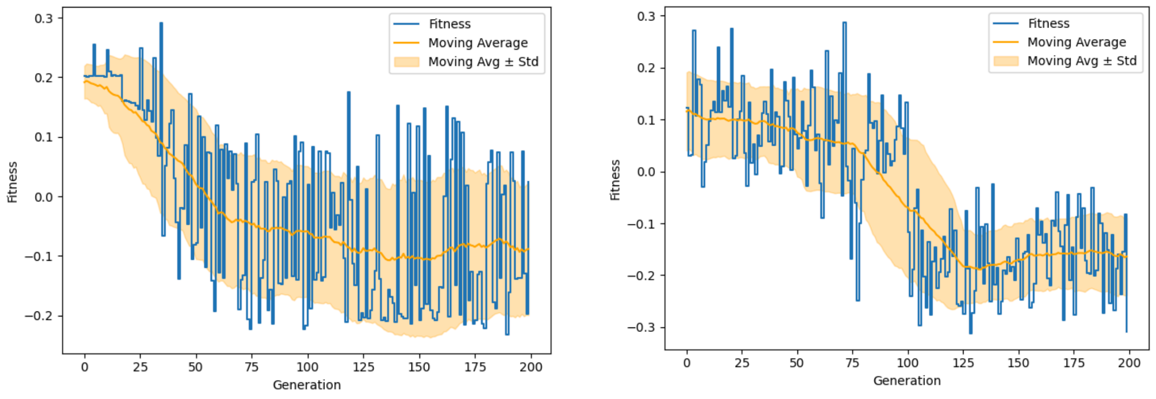


Figure 2. Example plots showing the fitness over time with standard deviation and mean lines for 3 runs.

A **code template is provided on the** Canvas assignment page **to help you get started.**

The code shows an example of starting a simulation and making the gecko do random movements. You can change this template to do neuroevolution. You **are allowed** to import third-party libraries like (DEAP, nevergrad, evotorch, etc) for your EA instead of writing your own from scratch. Just be sure to mention and explain the algorithm used in your report; and make sure it's an EA!

We also expect a plot showing the fitness over time for each experiment, with standard deviation shading and mean lines as shown in *Figure 2*.

This assignment is going to be a comparative analysis, which means that you will have to implement two main experiments (on top of the baseline experiment) and compare them to draw a conclusion.

This can be a comparison of the EA that does the neuroevolution (weight and/or architecture), the fitness function (task), or the environment. We expect you to test the baseline and the main experiments and show the differences/improvements in each. Then repeat each experiment a minimum of 3 times, and plot the results together to show any statistical significance. So, 3 experiments, each repeated 3 times (a total of 9 runs, and a total of 3 line plots).
It's important that the final comparative plots also show the performance of random moves, as this gives us an idea of how well you are actually learning!

At the end of the project, you need to hand in a report as specified below, your code for the project, and a file with the controller of your best-performing experiment for your chosen environment and task (such that we can reproduce your results). The report must be in the form of a PDF, and all files (report and controller weights) should be submitted in a zip.

Rules and guidelines

- You are not allowed to change the ARIEL source code in behaviour-changing ways. This means that you are not allowed to change the gravity setting to make walking easier or anything similar. We recommend that you do not edit the source code at all.
- You are allowed, but not required, to use the default fitness function for each environment and task. But if you choose to make your own fitness function, make sure to explain it and its boundaries in your report.
- As mentioned, this is a comparative assignment. This means that you will need to test your implemented algorithm/fitness function and compare it with another one (and the baseline) to show your work.
- After implementing your EA, you will need to repeat the experiment a minimum of 3 times for statistical significance. This will also allow you to create line plots with std shading and mean lines.
- Some libraries, like nevergrad, contain optimisation methods that are not evolutionary or population-based. **Such methods are not allowed!** If you use such a method, there will be **a point deduction**. This is also specified in the rubric.

Report structure

Your report should be a maximum of 3 pages (**including text, figures, and tables but excluding references**). The report format must follow the GECCO19 template. The word template can be found here:https://gecco-2019.sigevo.org/index.html/dl1241, and the LaTeX template can be found here: https://gecco-2019.sigevo.org/index.html/dl895.
Reports exceeding the page limit will automatically be graded as 0/10. This goes for even a few lines of text, SO NO EXCEPTIONS.

You need to include the following info at the beginning of the assignment.
These are:

- Your team number
- Names and student numbers of all members

The pages containing the actual content should be as follows:

1. Introduction (½ page, 1.5 point):
   a. Define your research question/goal and background information.
2. Methodology (1 page, 5 points):
   a. Explain how your algorithm/s work.
   b. Provide the motivation behind it/them, the parameter settings, experimental setup, fitness function, budget, etc.
   c. Make sure that everything is reproducible with the information presented.
3. Results and discussion (1 page, 5 points):
   a. Discuss the differences between the results of your EAs (or fitness functions). Do they outperform each other? Why? Comment on possible explanations for that.
   b. Discuss the difference between your results.
4. Conclusions (½ page, 3 points):
   a. What was the takeaway from your experiments?
   b. How can the experiments be improved?
   c. Were there any limitations?
5. Bibliography (0.5 points):
   a. Cite all works you are using in your project.

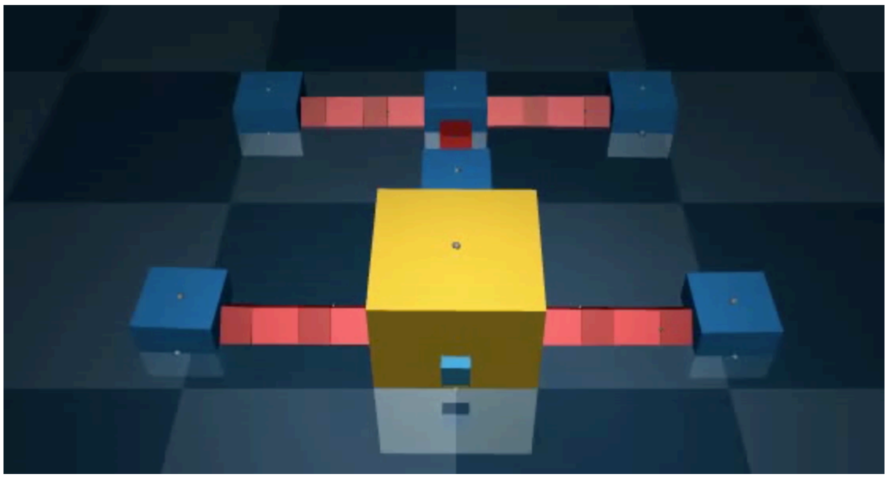
Grading

This assignment is worth 15 points and counts towards 15% of your final grade. A more detailed grading rubric can be found on the Canvas assignment page.

Common Pitfalls

Here are a set of (very) common mistakes that will cause you hours of debugging time if you are not careful (do not ask me how I know 😢)

<u>Robot clips through the ground, as seen here:</u>



Solution: change last element of the spawn point to above [0, 0, 0] eg [0, 0, 0.1]

<u>Robot “breaks physics” aka turbo-boi robots that spin and flies.</u>

Solution: ensure the data.ctrl is clipped to it’s correct bounds, as seen here:

```python
# Bound the control values to be within the hinge limits.
# If a value goes outside the bounds it might result in jittery movement.
data.ctrl = np.clip(data.ctrl, -np.pi/2, np.pi/2)
```

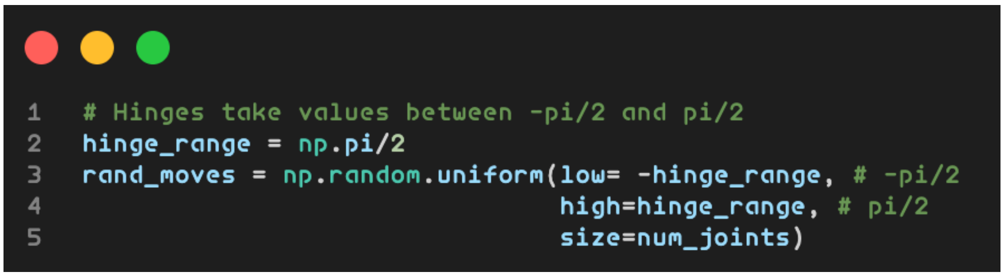
<u>Segmentation fault in MuJoCo.</u>

Solution: you aren’t clearing the control callback at the start of your script, please add:

```python
# Initialise controller to controller to None, always in the beginning.
mujoco.set_mjcb_control(None) # DO NOT REMOVE
```

<u>Robots makes small and slow movements no matter what I do with data.ctrl</u>

(Possible) solution: ensure that whatever is producing the signal for data.ctrl is in the correct range [-pi/2, pi/2], for example:

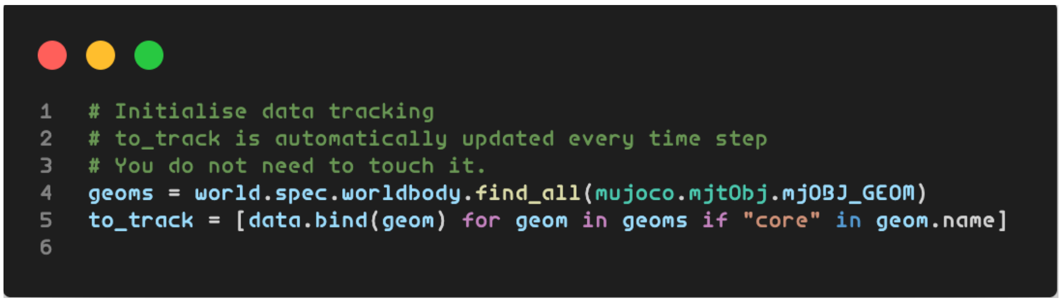


```python
# Hinges take values between -pi/2 and pi/2
hinge_range = np.pi/2
rand_moves = np.random.uniform(low= -hinge_range, # -pi/2
                               high=hinge_range, # pi/2
                               size=num_joints)
```

<u>The "to_track" rabbit hole</u>

If you relate with any of:

How does "to_track" work or
I don't know what's in "to_track" or
Why is "to_track" magic or
I do not know how to update "to_track" or
I am so done with the bugs that "to_track" gives me, I lost hope in robotics, and I want to have a garden with a little pond and ducks and corgis (and sleep); please help



```python
# Initialise data tracking
# to_track is automatically updated every time step
# You do not need to touch it.
geoms = world.spec.worldbody.find_all(mujoco.mjtObj.mjOBJ_GEOM)
to_track = [data.bind(geom) for geom in geoms if "core" in geom.name]

```

Solution: ask John or Jacopo… we mastered the object via ExL.