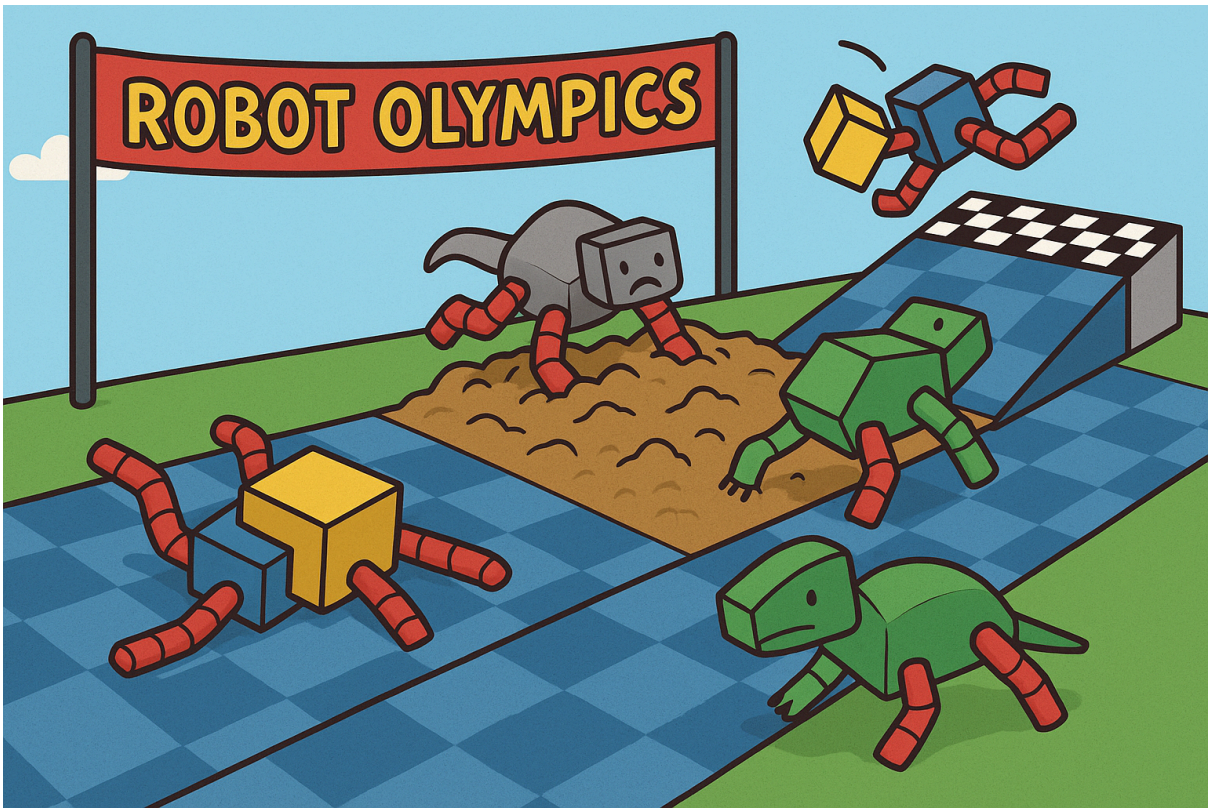
Evolutionary Computing Assignment 3
- Robot Olympics-



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 and apply algorithms for the task of robot evolution (evolving a robot's brain and body) on the Ariel platform. The assignment is to be

done in groups of three or four. **You need to deliver your code and a four-page report on your work and findings.**
Submission deadline is **Monday, October 13, at 9:00 in the morning**, and it is **NOT NEGOTIABLE!!!**

Each day of delay in the submission will result in a 1.5-point deduction from your grade (out of 30). This means that a submission done at 9:01 on October 13th 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. 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 examples are "***examples/A3_template.py***", which tests all the components that you will need for A3.
-

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		Option 2: UV Install	
Pros	Cons	Pros	Cons
+ Works everywhere	- Larger (~1 GB)	+ You have uv (future of python)	- May have unexpected incompatibilities
+ Sought after expertise in industry	- Slower install (~15 min)	+ Fast	
	- Maybe laggy on older computers	+ 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 goal of this assignment is to implement an evolutionary algorithm that evolves both the morphology and controller of a robot. Unlike assignment 2, which focused only on robot evolution, this task requires optimising both aspects together. There are several possible approaches, each with varying levels of effectiveness.
Additionally, unlike assignment 2, this assignment is **not a comparative analysis.** This means that you will not need to implement multiple EAs or fitness functions. The task of the assignment is only to make the best-performing robot you can and report on your research and implementation.

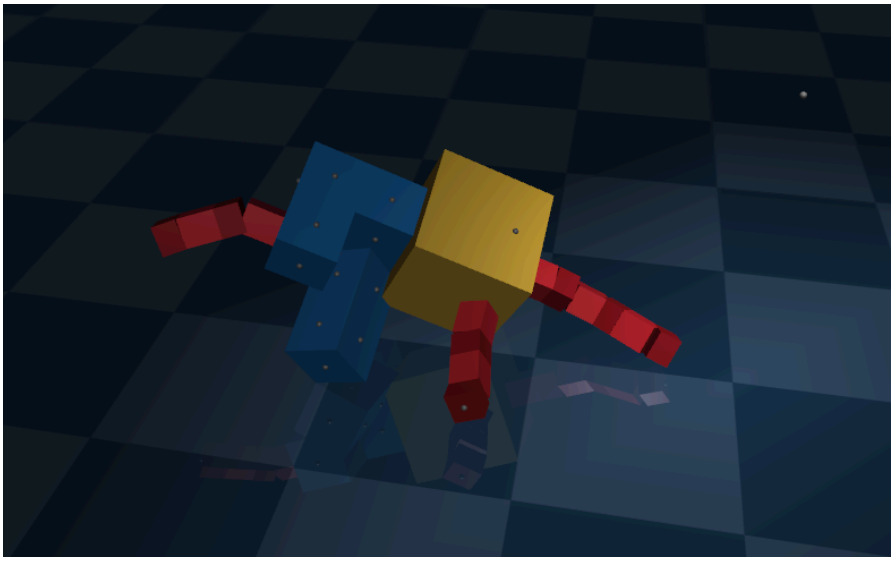


Figure 1. Image of a randomly initialised robot

As this is a robot evolution task, **you are not allowed to use the prebuilt gecko body or to make your own robot by hand.** All runs you do should start with randomly initialised robots,

although, as discussed in the practicals, you are allowed to “kill off” robots that are unable to walk at all.

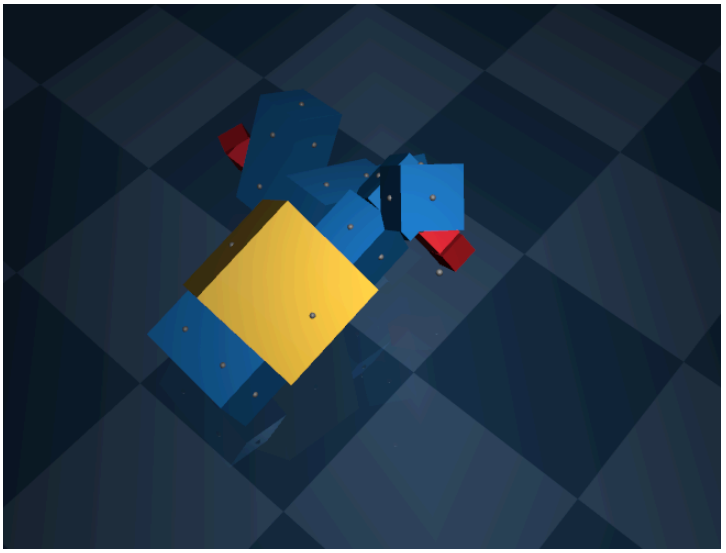
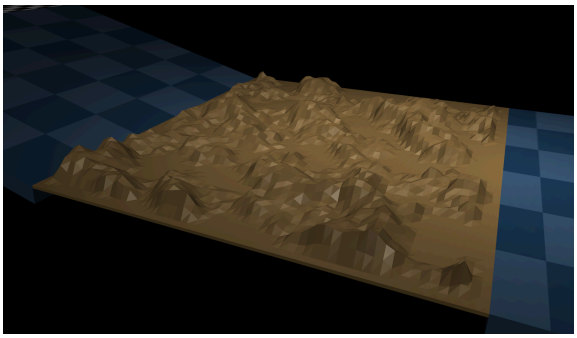
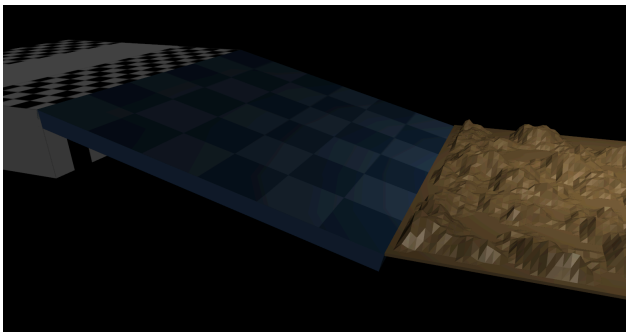


Figure 2. An image of a robot that is physically unable to walk. As you can see, the robot has no way of touching the ground or pushing itself along. This means that it will never move and thus is not worth training at all.





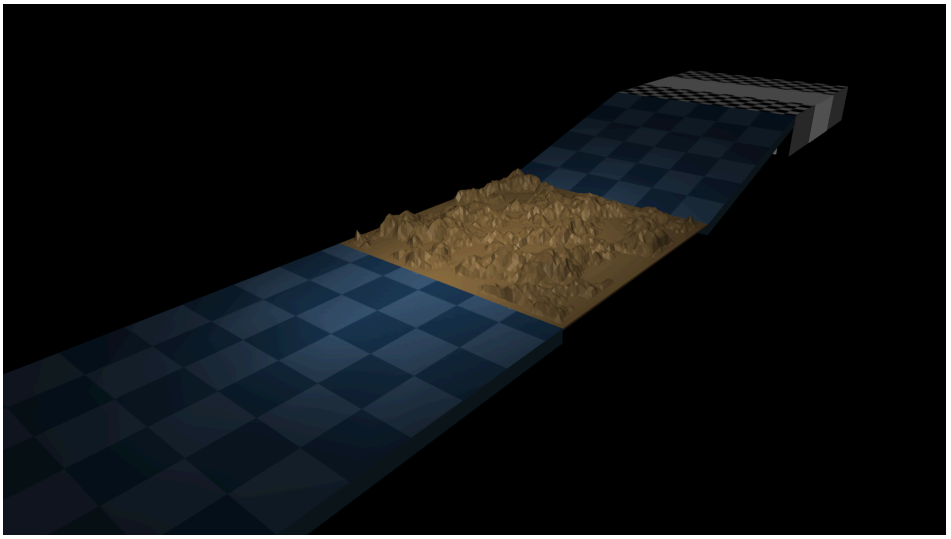


Figure 3. Images of the Olympic Arena environment and its separate parts

The goal of this assignment is to reach the end of the environment in the fastest time possible. You have to use the “OlympicArena” environment (found in `ariel.simulation.environments`), which contains a smooth flat section, a rugged flat section and a flat uphill section. The whole environment and its subparts can be seen in Figure 3.

To achieve the best results, you can use the built-in ARIEL evolutionary operators or implement your own from scratch. Similarly, you can use one of the default fitness functions we provide you with or, if you feel you can improve it, make your own.

Your submission will be tested against the other groups in a race. Your robot will be evaluated on its ability to reach the end of the arena and the speed at which it does the task. The exact evaluation goes like this:

1. If only one robot has reached the finish line, it is the winner. If multiple robots have done so, the robots will be ranked by their speed.
2. If a robot has not reached the finish line, its score is based on the distance it has covered. Again, if multiple robots have covered the same distance, they will be ranked on speed.

At the end of the project, you need to hand in a report as specified below, your code for the assignment and a file with the controller and robot body of the result of your best-performing experiment. The report must be in PDF format, and all code should be submitted in a zip file.

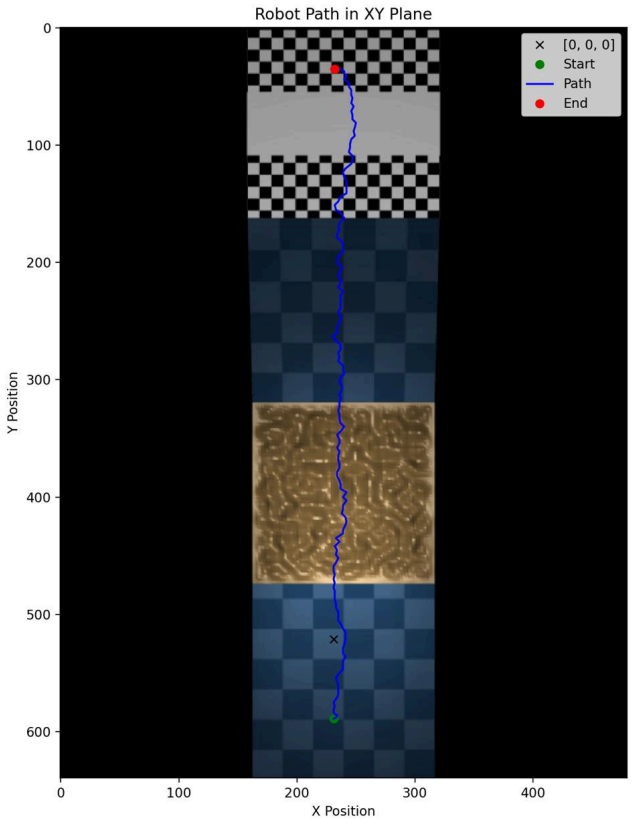


Figure 4. Example of the path taken by the robot plotted on the Olympic Arena environment

Additionally, we will give you a short plotting function that shows the robot's path on the arena. You will have to show the path of the best individual per generation for your experiment in the report. You can see what this plot looks like in Figure 4.

To generate the robot bodies you must use the Neural Developmental Encoder to generate the individuals. The NDE takes in a list of 3 vectors and outputs the probability matrices that then are used to generate the robot. **You are only evolving the vectors, you are not training the NDE network or evolving the matrices. <u>You are only to touch the NDE input vectors!</u>**

The robot olympics submission (Friday October 10th) only needs your robot body and network weights and architecture (**the json file holding your robot, not the genotype!!! since, due to the NDE, we cannot rebuild the robot from the genotype**). The report submission needs a zip of your code, a video of your best-performing robot and your report (the report should not be in the zip), as seen in Figure 6.

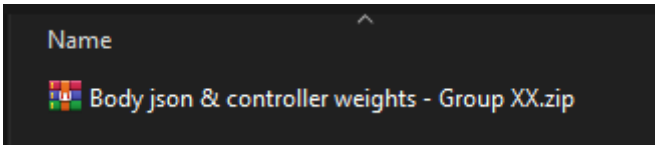


Figure 5. Submission type for the **<u>Robot Olympics</u>**

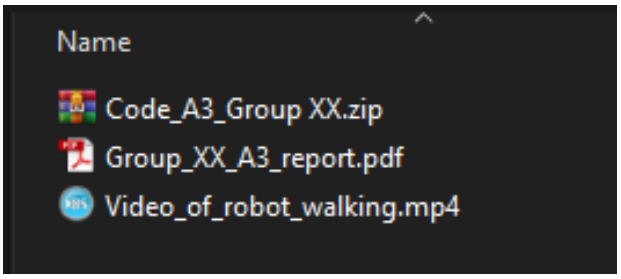


Figure 6. Submission type for assignment 3 **<u>report</u>**

<u>Rules and guidelines</u>

- 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 we give you. But if you choose to make your own fitness function, make sure to explain it and its boundaries in your report.
- Compared to assignment 2, this is **<u>not a comparative analysis</u>** assignment. This means that all you need to do is build the best algorithm you can, and break down your research, methodology and results in your report. You do not need to run this multiple times. One is enough, but if you have the time, running it multiple times is greatly appreciated for statistical significance.
- Similarly to assignment 2, we also ask you to run a **<u>Baseline experiment.</u>** This means that you need to run the same code you have for your normal experiment, but

instead of doing the learning process, you randomly initialise networks and use those in every generation. You do not need to run this multiple times. One is enough.

- Some libraries, like nevergrad, contain optimisation methods that are not evolutionary or population-based (for example, reinforcement learning). **Such methods are allowed for the learning part of the assignment only! Yes, we say they ARE allowed.** But take it from us, reinforcement learning is almost 50 times slower than an EA in this task.
- **You must use evolutionary operators to do the body evolution. If you do not, you will be penalised.**

Report structure

Your report should be a maximum of 4 pages (**including text, figures, and tables but excluding references**). Additionally, you have infinite appendix space for additional figures. **Put all necessary figures in the main text; an Appendix is only used for supplementary material.** The report format must follow is the same template as Assignment 2. The overleaf template can be found here:

- https://www.overleaf.com/project/68d003e461751d28ba22f4af

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, 3 points):
    a. Define your research question/goal and background information.
2. Methodology (1 page, 10 points):
    a. Explain how your algorithm works.
    b. Provide the motivation behind it, the parameter settings, experimental setup, fitness function, budget, etc.
    c. Ensure that everything is reproducible based on the information presented.
3. Results and discussion (1 page, 10 points):
    a. Present your results, including fitness-per-generation plots and other relevant figures.
    b. Discuss why you think your implementation behaved the way it did and its performance..
4. Conclusions (½ page, 6 points):
    a. What was the takeaway from your experiments?
    b. How can the experiments be improved?
    c. Were there any limitations?
5. Bibliography (1 points):
    a. Cite all works you are using in your project.

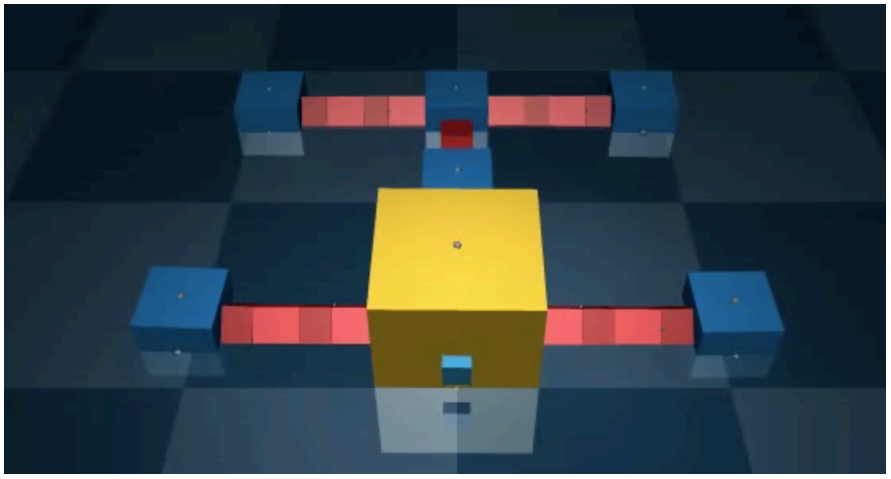
Grading

This assignment is worth 30 points and counts towards 30% of your final grade. A more detailed grading rubric can be found on the Canvas assignment page.

Common Pitfalls

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

Robot clips through the ground, as seen here:



This is much rarer (almost impossible) when initialising random robots, but if it happens, you can change the spawn position.

Solution: Change the last element of the spawn point to above [0, 0, 0]. For example [0, 0, 0.2]

Robot "breaks physics," aka turbo-boi robots that spin and fly.

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

```
# 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)
```

Segmentation fault in MuJoCo.

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

Solution: You aren't clearing the control callback at the start of your script. Please add:

If you do not set the control to None at the very beginning of your simulation, you might get a `Python exception` error, which seemingly has no solution.

Robots make small and slow movements, no matter what I do with data.ctrl

(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)
```

Any other error that seemingly has no solution?

Solution: ask John or Jacopo… we mastered the art of ARIEL errors, via ExL.