MICRO-515: Evolutionary Robotics
Prof. Dario Floreano

Exercise 3: RoboGen Body-Brain Co-Evolution 1

Jan Petrs (jan.petrs@epfl.ch)
Alexander Dittrich (alexander.dittrich@epfl.ch)
Juliette Hars (juliette.hars@epfl.ch)

Goal

To perform a body-brain evolution with RoboGen. The body of the robot will be evolved from scratch to locomote as fast as possible. The exercises in this sheet are directly related to the tasks in the final graded project.

Learning objectives

- Understand the computational cost involved with just learning a good controller, let alone co-evolving the brain and body.
- Try co-evolving controllers and morphologies with RoboGen.
- See the effect of simple changes of the simulated environment (e.g. friction) can affect robot evolution.

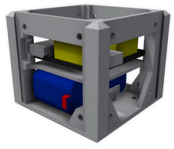
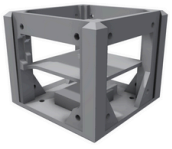
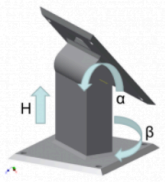
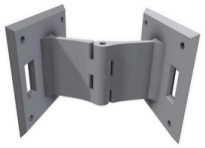
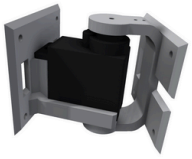
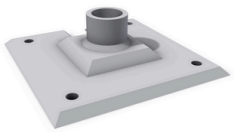
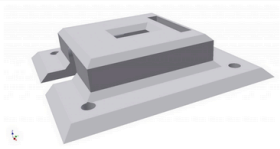
Getting Started

To get started, visit http://robogen.org/app and upload the files provided in Moodle into `Robogen2022/es3/`.

> **Note:** If you receive a "404 Error" ensure you access via "http://..." and not "https://...", as some browsers access URLs by default with secure HTTP.

Important:

- Remember, all data is being saved to a virtual file system within your web browser. If you want to save anything for later use, be sure to download it to your home directory!
- In Exercise 1, you were evolving a wheeled robot. In this and the following exercises, you are not allowed to use wheels. So you have to come up with a legged design for locomotion. The available parts are `CoreComponent`, `FixedBrick`, `ParametricJoint`, `PassiveHinge`, `ActiveHinge`, `LightSensor`, and `IrSensor` (if you specify `addBodyPart=All` when evolving morphologies, these will be the parts that your robots will be composed of).

Image	Part name	Code	Arity	Motors	Sensors	Image	Part name	Code	Arity	Motors	Sensors
	Core Component (CoreComponent)	E	4	0	6		Fixed Brick (FixedBrick)	F	3	0	0
	Parametric Bar Joint (ParametricJoint)	B	1	0	0		Passive Hinge Joint (PassiveHinge)	H	1	0	0
	Active Hinge Joint (ActiveHinge)	I	1	1	0		Light Sensor (LightSensor)	L	0	0	1 (analog)
	Infrared Sensor (IrSensor)	D	0	0	1						

Exercise 3.1

Before we get started co-evolving the brain and body of a robot, let's first go back to just evolution of a brain but this time for a robot without wheels. We will use the starfish robot in `starfish.txt`.

Working Directory	`/localStorage/Robogen2022/es3`
Configuration file	`evolConf.txt`

Set the following parameters in `evolConf.txt`:

```
evolutionMode=brain
referenceRobotFile=starfish.txt
```

Despite the morphology already being developed with symmetrical limbs, you will find that it still takes significant computational resources to evolve anything reminiscent of legged locomotion. Instead, initial evolved solutions often include vibrating across the floor. As the servo motors used to build the real robots may not be capable of such high frequency oscillations, you may want to explore simulation parameters that you set in `simConf.txt` that make this behavior less likely (http://robogen.org/docs/evolution-configuration/#Simulator_settings). For example, one parameter you could explore is the number of times the motor can shift per second: `maxDirectionShiftsPerSecond`

> **Note:** When you see `maxDirectionShiftsPerSecond`, you will often notice that the video of your solution ends early, sometimes in under a second. This is because the simulation is terminated early when the commands sent by the neural network to the motor exceed the max direction changes per second. If you continue the evolution, it may eventually find a solution that does not exceed `maxDirectionShiftsPerSecond` in later generations.

The purpose of this exercise is to give you an appreciable understanding of the computational cost of evolving even just a controller for a more complex robot, where the gait must be evolved first before the robot can be evolved to do anything more useful. Now let's move on to evolving the body as well as the brain.

Exercise 3.2

Now let's move on to explore the basics of evolving morphologies:

Look at the `es3/evolConf.txt`, this shows an example evolutionary configuration for evolving brains + bodies (note: change back to `evolutionMode=full` and remove `referenceRobotFile=starfish.txt`). Your population will start from a random collection of morphologies using the allowed parts.

- `numInitialParts=MIN:MAX` defines the possible sizes of these initial morphologies.
- The `addBodyPart` command in your evolutionary configuration file defines what body parts can be included.
    - In the example `addBodyPart=All`, but why do you think including all body parts may not always be the best idea?
    - Change this to specify just particular body parts (on separate lines), this can take either the Character Code, or name of the Body Part. For example:
        - *addBodyPart=FixedBrick*
        - *addBodyPart=ActiveHinge*
        - *addBodyPart=PassiveHinge*
- Try evolving some basic morphologies with the `evolConf.txt` file! We recommend starting with a small number of initial components (say 4 or 5), and only allowing a small subset of components at first.
- Familiarize yourself with the other parameters controlling the mutation operators for morphological evolution.
    - http://robogen.org/docs/evolution-configuration/#Evolution_client_settings
    - For example, try adjusting the probabilities of adding body parts, swapping subtrees, modifying parameters, etc.

Getting good results with full evolution may take some time.

- You may need to use **larger population sizes**, experiment with the replacement strategy and tournament-size, and run for many generations. **You may not see good results just by running the evolution engine in the limited time you have during this lab. To see good results, consider executing some evolutionary runs, e.g. over the night.**

Exercise 3.3

The evolution of a robot will be strongly affected by the simulated environment. Therefore, if some parameters in the simulator configuration file are changed, the results can be quite different. You should experiment with different ground friction coefficients to see how they influence the evolution of a robot.

> **Note:** The full documentation for defining simulator configuration file is available at *RoboGen Simulator Settings*

`terrainFriction` – this specifies the coefficient of friction between the robot and the terrain.

Try setting the value of `terrainFriction` to `10` (this setting could represent operating in a very muddy field) and run a simulation using `es3/simpleRobot.txt` as the Robot description file and `simConf.txt` as the configuration file. Do you notice differences in the robot's behavior?

Now you can try evolving a robot that moves as fast as possible (with the racing scenario) in a terrain with friction coefficient of `10`. Analyze the performance of the evolution (using the `plot_results.py` file provided in Exercise 1) and try to evolve a robot that does well in this terrain.

Exercise 3.4 (optional)

If your robot can locomote quickly on flat ground with a friction coefficient of `10`, try evolving a robot with the same friction coefficient but with obstacles added to the environment.

Exercise 3.5

Now let's consider that the robot you evolve will be used to explore Antarctica. Evolve a robot that can move as far (not as fast) as possible on flat ground when the friction coefficient is `0.04`. The performance metric is how far the robot moves in 30 seconds.