MICRO-515: Evolutionary Robotics
Prof. Dario Floreano

Exercise 4: RoboGen Body-Brain Co-Evolution 2

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

Goal

Use RoboGen to evolve robots able to solve more complex tasks. The evolved robot must follow a signal, in this case a moving light, in the environment.

Learning objectives

- How to co-evolve controllers and morphologies for more complex tasks that include sensor readings.
- How to use penalization in fitness functions.
- How to evolve robots capable of making the jump from simulation to reality, i.e. how to avoid evolving robots that exploit some feature of the simulation and therefore don't work in the real world.

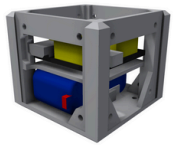
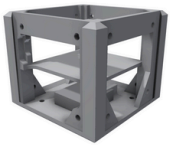
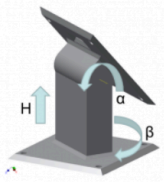
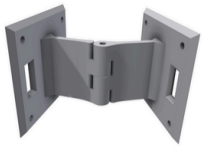
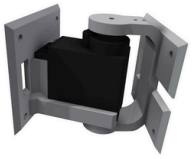
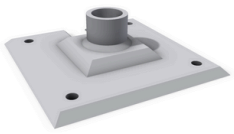
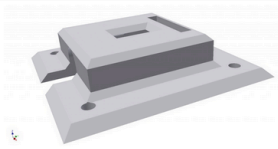
Getting Started

To get started, visit http://robogen.org/app and upload the files provided in Moodle into `Robogen2022/es4/`.

> **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 4.1

You will now perform a body-brain evolution of a robot that can both locomote and follow a light source.

First, if you try "Start a simulation" using `es4/simConf.txt` and `es4/robot.txt`. You will see an example of an arena where a fixed light is placed. In `simConf.txt` observe the two parameters:

- `lightSourcesConfigFile=lights.txt`: used to add information on where the light is placed. See the documentation on the RoboGen website for more details.
- `scenario=chasing`: this fitness function will try to minimize the distance between the robot and the light.

If the light is always in the same position during evolution, it is possible that the robot will just learn to move along a specific path rather than use its light sensors to detect where the light source is and navigate towards it (overfitting). To avoid overfitting, you should set up the evolution so that:

a. The robot begins from different starting positions by making a `startPos.txt` file (remember to add `startPositionConfigFile=startPos.txt` to `simConf.txt`).
b. Alternatively, change the scenario in `simConf.txt` to `scenario=chasing-rand-light.js`. In this scenario, the light moves in a different direction each time the simulator is started, so only robots that utilize their light sensors will be able to chase it.

We aren't quite ready to evolve a robot yet though, as it is actually very difficult to evolve a robot for two different tasks simultaneously, e.g. the locomotion and light following tasks in this example. We therefore suggest that you perform a **multistep evolution**. Multistep evolution is usually performed when multiple distinct behaviors should be evolved in a robot. As the name implies, multiple evolutionary runs are performed in series, each step uses a different fitness function to evolve the robot for one of the desired behaviors. At the end of each step, the best robot from an evolutionary step is given as a `.json` file, which needs to be converted using `json_converter.py` to a `.txt` file so it can be input as the starting point in the next step of the evolution.

As an example, to perform two-step evolution:

1. **Step:** evolve the body and brain of a robot able to locomote and turn.
2. **Step:** now we have a robot body and brain from the Step 1 in `GenerationBest-X.json` that can locomote. Before starting Step 2:
    a. Convert `GenerationBest-X.json` to `GenerationBest-X.txt` using `json_converter.py`.
    b. Set `referenceRobotFile=GenerationBest-X.txt` and `useBrainSeed=True` (see http://robogen.org/docs/evolution-configuration) in `evolConf.txt` to seed the Step 2 population with the body and brain from Step 1.
    c. Set `addBodyPart=LightSensor` in `evolConf.txt` to limit the body evolution to adding light sensors only in Step 2.

    Now evolve a robot able to follow a light thanks to the usage of light sensors in the body and the relative adjustment of the NN controller.

You may see the fitness plateau after a certain number of generations. If this happens, try the following:

- Keep the evolution running for more generations. Big improvements can sometimes happen after 50–100 generations, even after the fitness seems to have plateaued already.
- Explore the fitness landscape more, i.e. increase the population size, mutation rate, crossover rate, or make the tournament size smaller.
- Improve the fitness function. For instance, modify the equation by changing some of the terms or give weights to different terms in the existing fitness function to change their relative importance.

Exercise 4.2

Real sensor data usually includes some noise. In addition, physical phenomena that are not modelled in the simulator may influence the real system, e.g. reflections, different lights in the environment, or inaccurate sensors. Therefore, to make the evolved robot generalizable when transferring the model from simulation to reality, noise should be added to sensor readings in the simulator. Try to add noise of `0.1` to the sensors and check if the performance is still good. If not, try an additional step of the multistep evolution, where you perform a brain only evolution in the presence of sensor noise.

NOTE: See the documentation to add sensor noise.

Exercise 4.3

If your robot can follow the light when the terrain is flat, you could try to evolve a new one using an arena with a movable light and a more challenging terrain of your choosing.