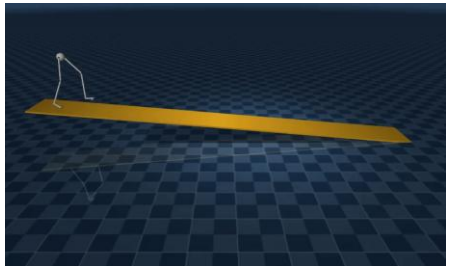
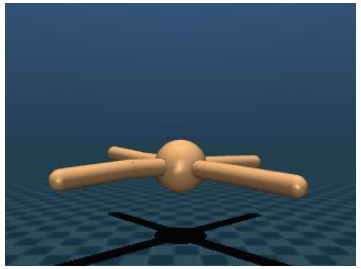
Overview Challenges



Introduction

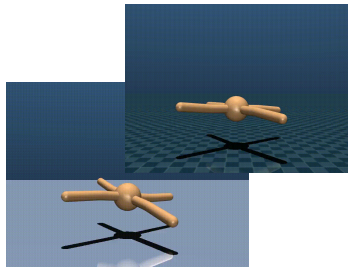
Evolutionary Strategies
Passive Dynamic Walker



Challenge 1

Submission Deadline:
19.03.2026

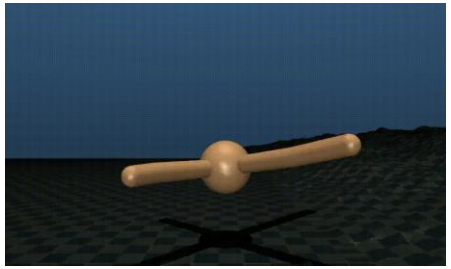
Evolving Neural Controllers
Ant – Flat Terrain



Challenge 2

Submission Deadline:
02.04.2026

Multi-objective Optimization
Ant – Ice and Flat Terrain



Challenge 3

Submission Deadline:
23.04.2026

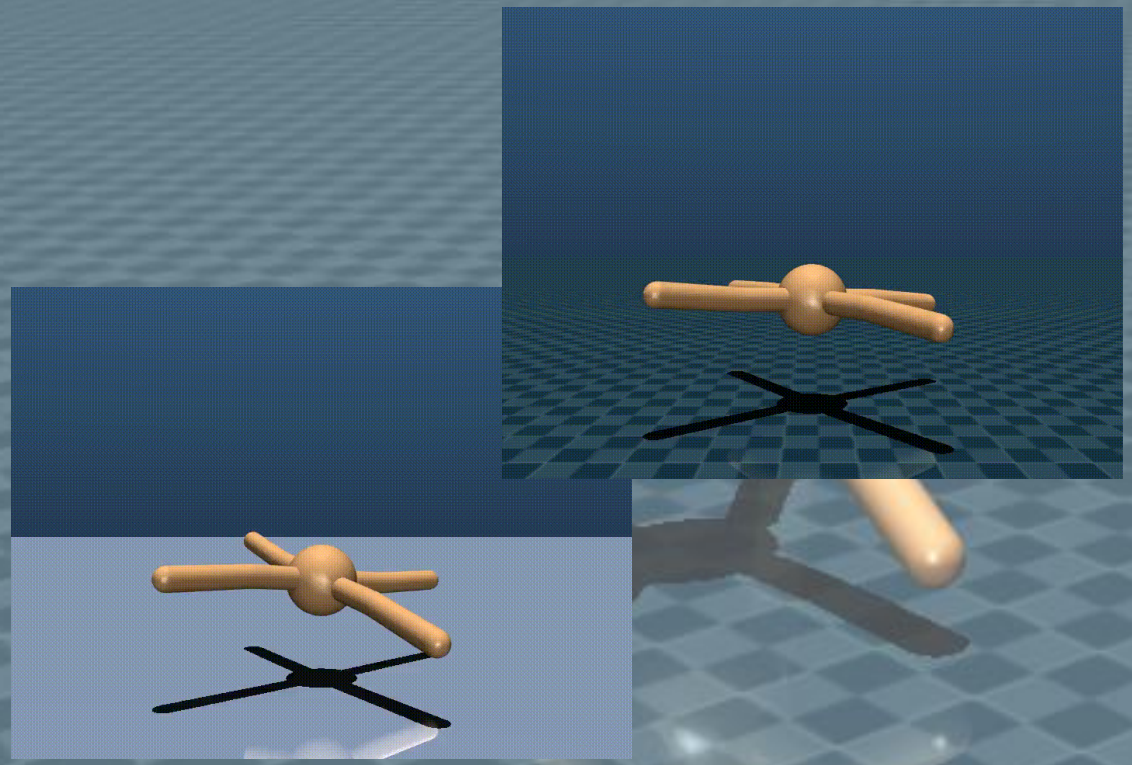
Evolving Robot Body
Ant – Hill Climbing

Final Project

Poster Presentation:
28.05.2026

Your Algorithm ☺
Multi-terrain Obstacles

Challenge 2
Objectives



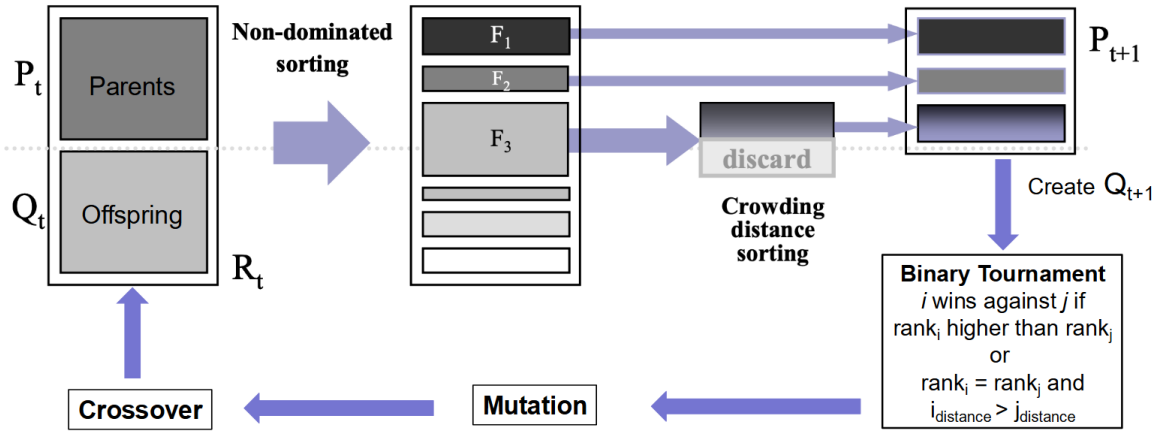
Multi-objective Evolutionary Optimization

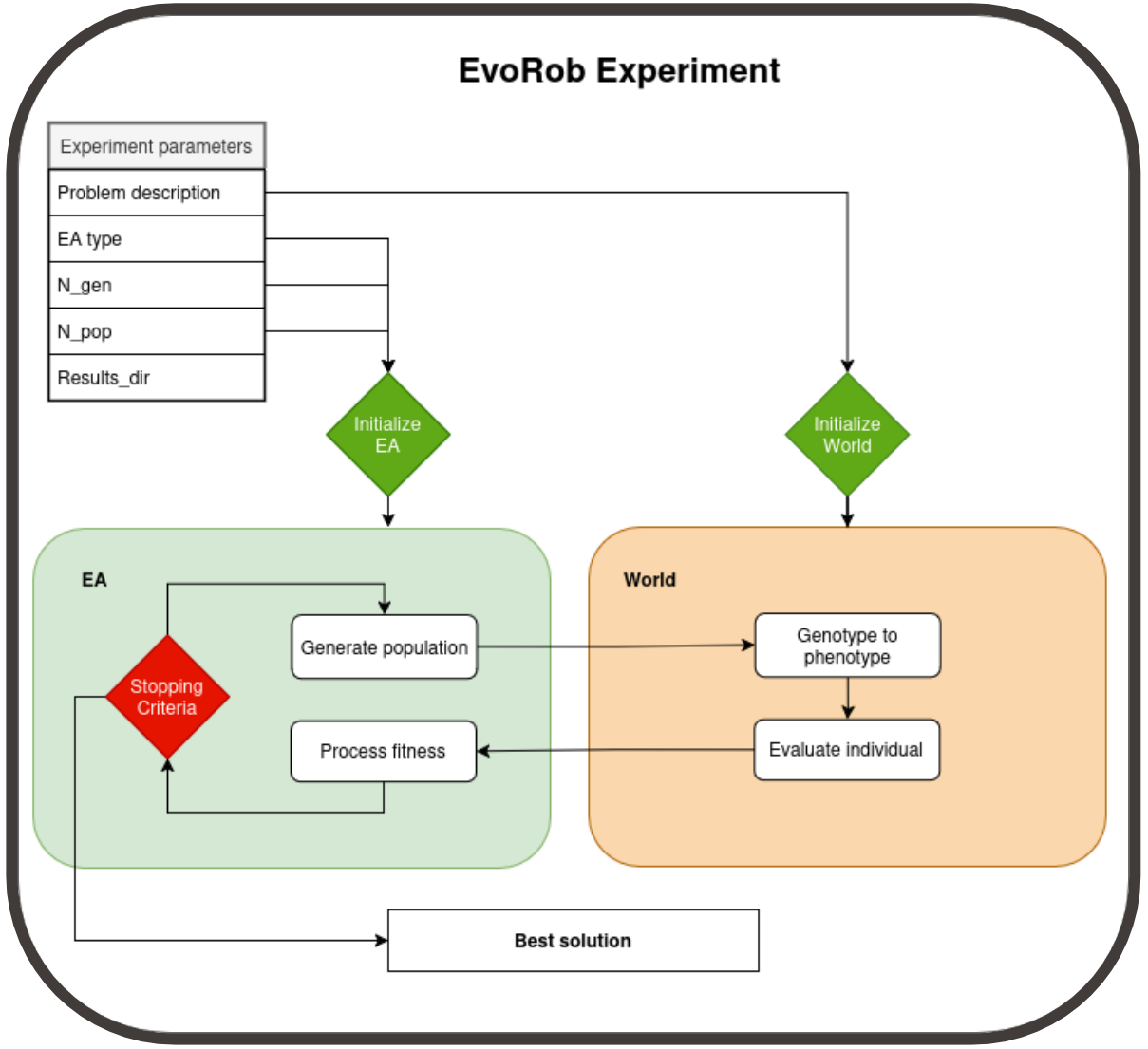
Challenge 2
NSGA-II

In the exercise today, you will write your own NSGA-II implementation as taught in the lecture.

Contents:

- Pareto dominance
- Non-dominated sorting
- Crowding distance for diversity
- Crowding operator





Challenge 2
NSGA-II

What do you need to run `Challenge2.py`:

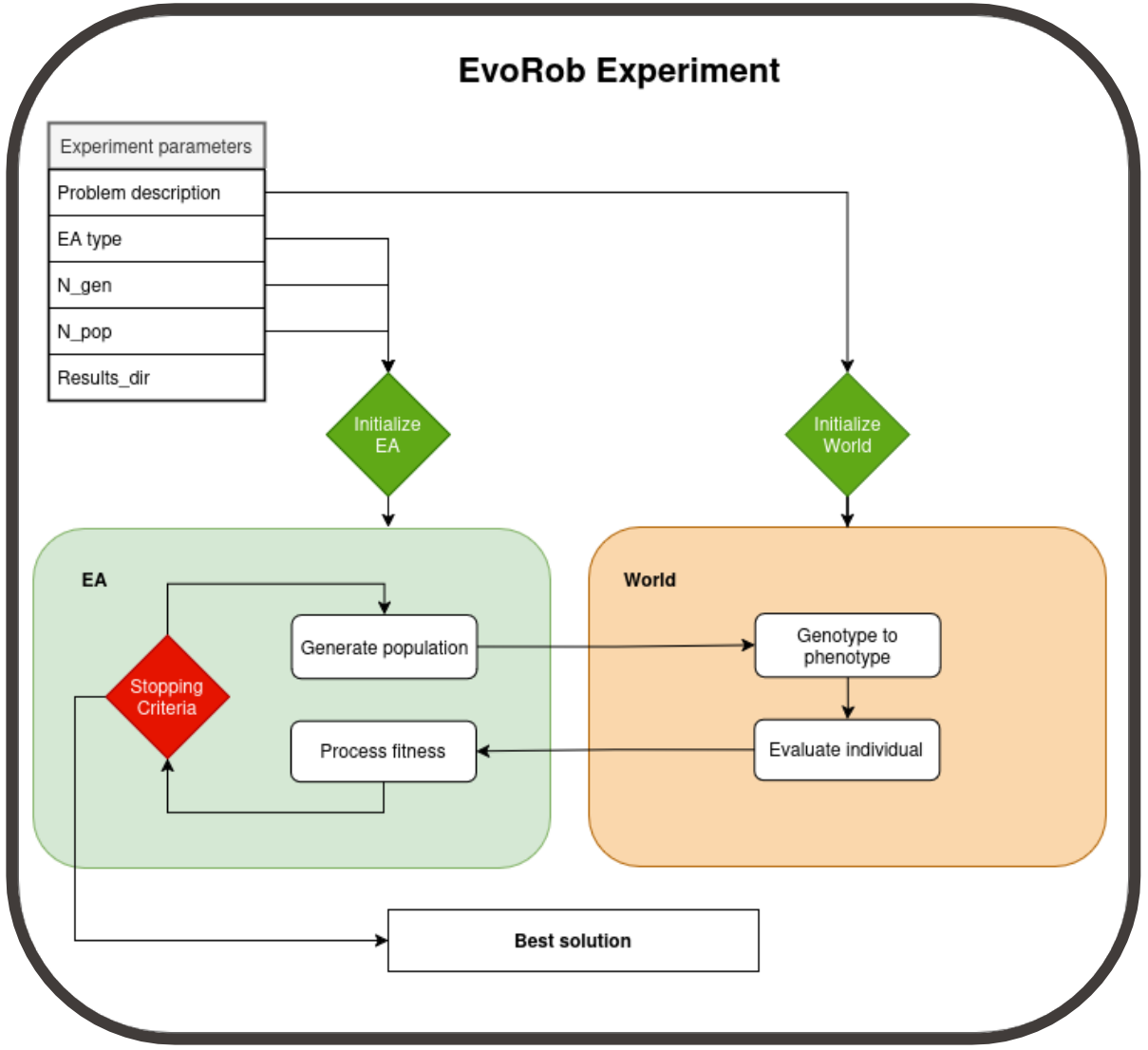
- MLP (`evorob/world/robot/controllers/mlp.py`)
- Ant Environment (`evorob/world/robot/ant_flat.py`)

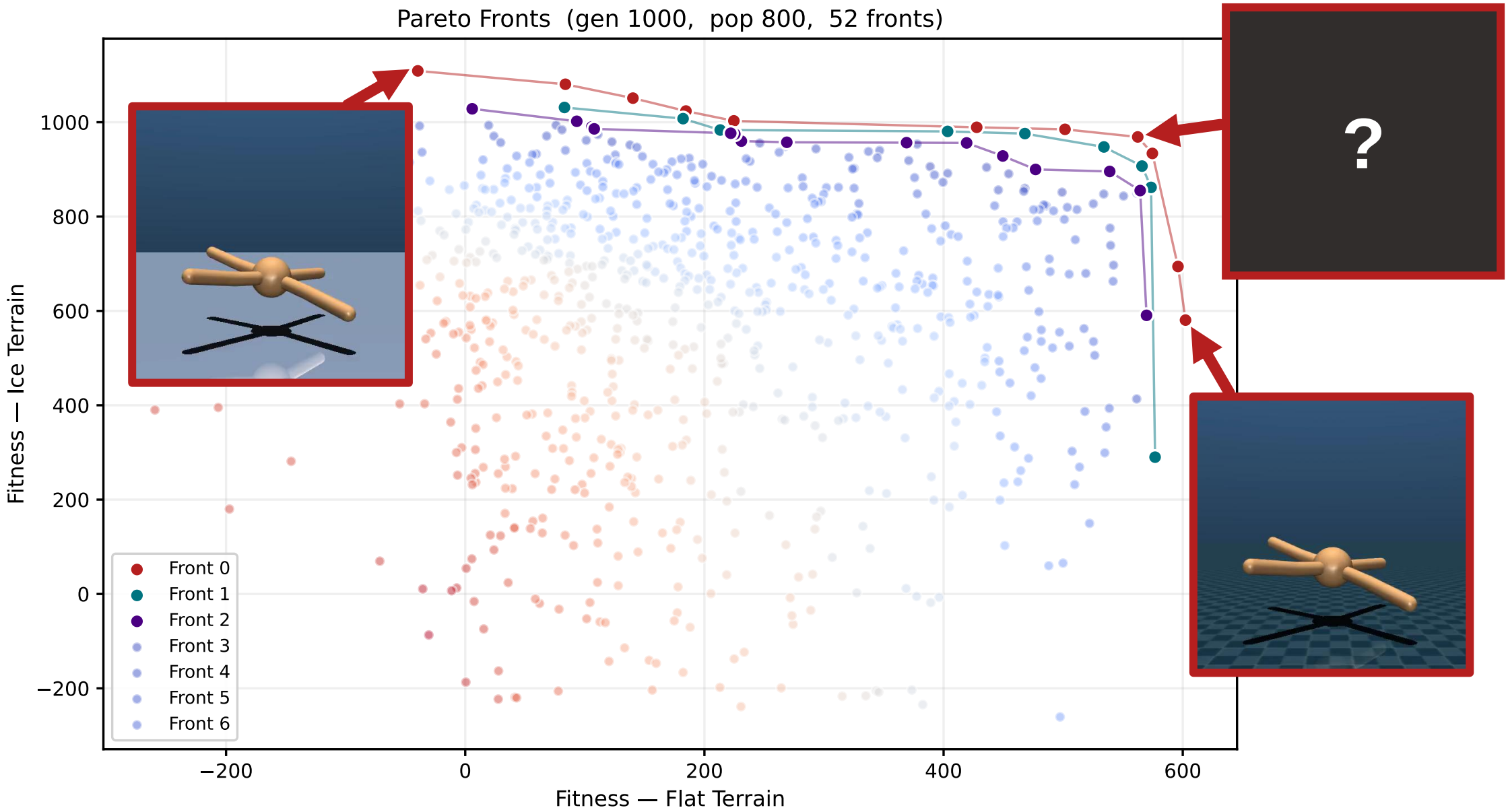
Can be either taken from **Solution** (uploaded now!)
- `evorob/world/robot/controllers/mlp_sol.py`
- `evorob/world/robot/ant_flat_sol.py`

or merge your own implementation from Challenge 1.

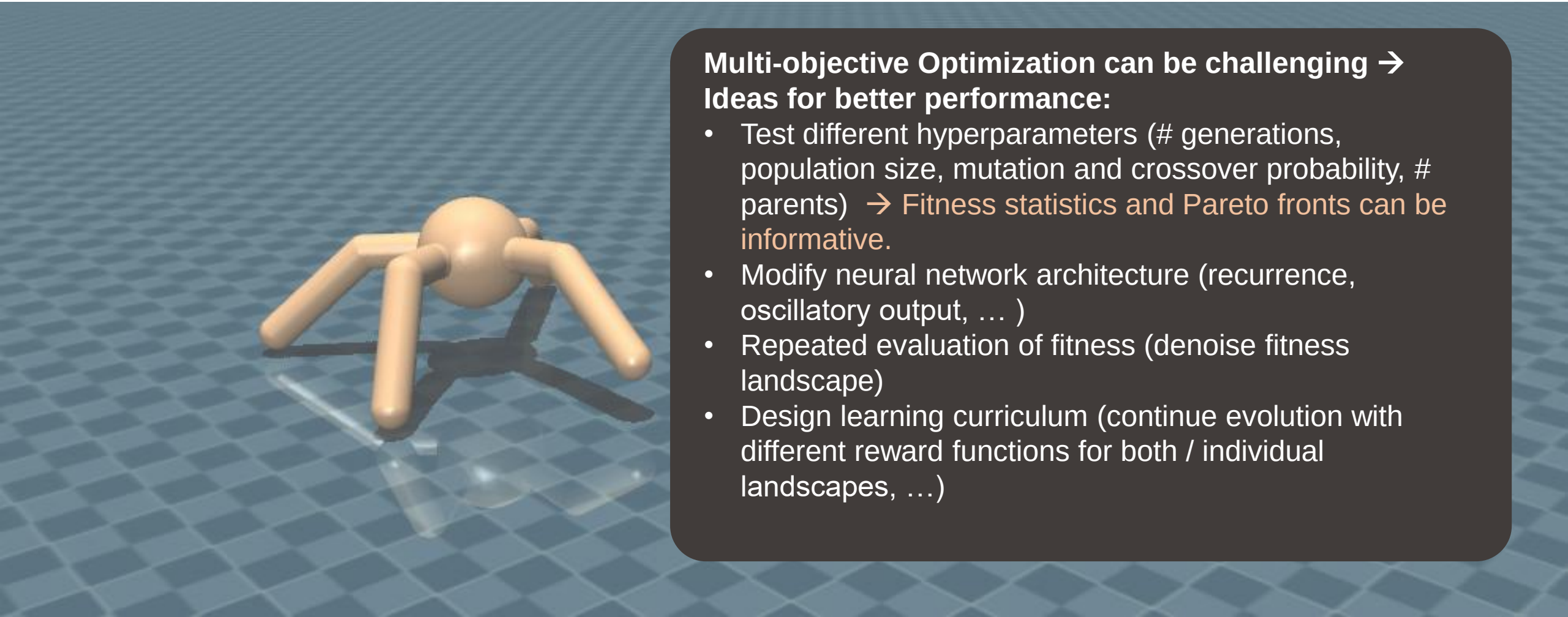
You need to modify this:

`evorob/algorithms/nsga.py`




Pareto Fronts (gen 1000, pop 800, 52 fronts)
Fitness — Ice Terrain
Fitness — Flat Terrain
1000
800
600
400
200
0
−200
−200
0
200
400
600
Front 0
Front 1
Front 2
Front 3
Front 4
Front 5
Front 6
?

Challenge 2
How to get better performance?



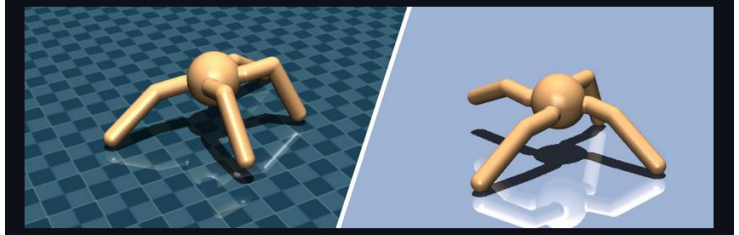
Multi-objective Optimization can be challenging → Ideas for better performance:

- Test different hyperparameters (# generations, population size, mutation and crossover probability, # parents) → Fitness statistics and Pareto fronts can be informative.
- Modify neural network architecture (recurrence, oscillatory output, …)
- Repeated evaluation of fitness (denoise fitness landscape)
- Design learning curriculum (continue evolution with different reward functions for both / individual landscapes, …)

Challenge 2
Where to find help?

README.md in Repository

Challenge 2: Evolving Multi-objective Neural Controllers in Two-Terrain Scenario



In Challenge 2, you will use multi-objective evolutionary algorithms to evolve specialist and generalist neural controller to let an abstract four-legged robot learn to locomote on a flat surface with different friction conditions as fast as possible.

Learning Goals

In this exercise, you will ...

- implement the baseline multi-objective evolutionary algorithm **Non-sorting Genetic Algorithm II** proposed by Deb et al. 2002.
- compare single-objective specialist controllers to controllers on the Pareto front evolved by your multi-objective algorithm.

Good to know!

This exercise builds on the skills you developed in Challenge 1. You will now implement NSGA-II, a multi-objective evolutionary algorithm that can optimize multiple competing objectives simultaneously (e.g., speed on flat terrain vs. speed on rough terrain). The main running script is Exercise2.py, and you'll work with the `evorob` codebase to make it functional.

Exercise 2: Understanding Multi-Objective Optimization

In Challenge 1, we optimized a single objective: moving forward as fast as possible. In real robotics, we often face multiple competing objectives:

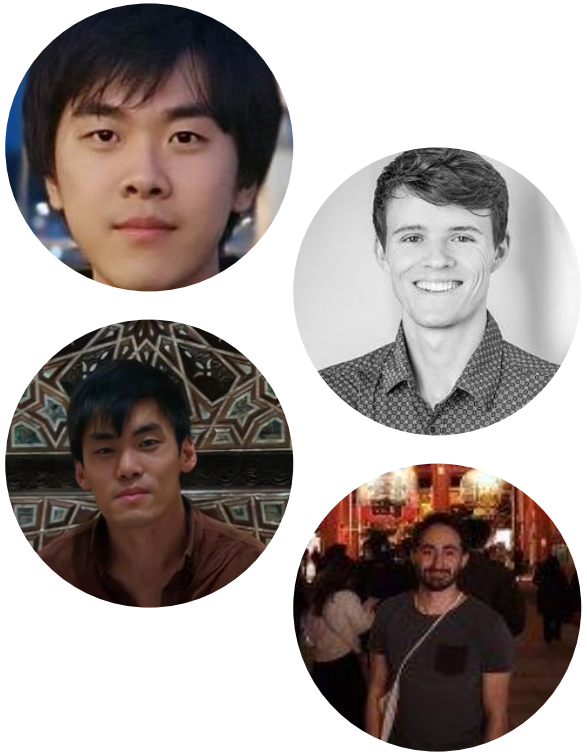
- Speed on flat terrain vs. speed on rough terrain
- Energy efficiency vs. maximum speed
- Robustness vs. performance

NSGA-II helps us find a **Pareto front** of solutions where improving one objective requires sacrificing another. This gives us a diverse set of specialized and generalist controllers.

TODO comments give hints

```
100
101     def _get_obs(self):
102         # TODO: Return observation as concatenation of:
103         # - position EXCLUDING x,y: self.data.qpos[2:].flatten() (13 values)
104         # - velocity: self.data.qvel.flatten() (14 values)
105         # This gives 27 total dimensions, making the task translation-invariant
106         # Hint: Use np.concatenate() to combine both arrays
107         raise NotImplementedError("TODO: Implement observation function")
108
109     def _get_rew(self, x_velocity: float, action):
110         # TODO: Implement reward function with three components:
111         # 1. forward_reward = x_velocity * forward_reward_weight (weight=1.0)
112         # 2. healthy_reward = healthy_reward_weight (weight=1.0)
113         # 3. ctrl_cost = ctrl_cost_weight * sum of squared actions (weight=0.5)
114         # Final reward = forward_reward + healthy_reward - ctrl_cost
115         # Return: (reward, reward_info_dict)
116         raise NotImplementedError("TODO: Implement reward function")
117
118     def _get_termination(self):
119         # TODO: Robot should terminate when:
120         # - Any value in state is not finite (check with np.isfinite(state).all())
121         # - Torso height (state[2]) is below 0.26 or above 1.0
122         # Return True if NOT healthy (i.e., should terminate)
123         # Hint: Use self.state_vector() to get current state
124         raise NotImplementedError("TODO: Implement termination function")
```

MICRO-515 TAs



Challenge 2
Submission Details

- the code of your **final controllers** and the corresponding **evolved weights** (**important**: your controller needs be compatabile with the controller interface),
- the **videos** rendering of your best “specialist” on flat / ice terrain and the best “generalist” on flat / ice terrain,
- the **fitness graph** showing the evolution of fitness over generations and the **pareto-front** over fitness on flat terrain and fitness on ice terrain,
- a textfile **README.md** describing shortly the selected algorithm, environment design and controller (max. 300 words),
- the generated evolution_score.txt file.

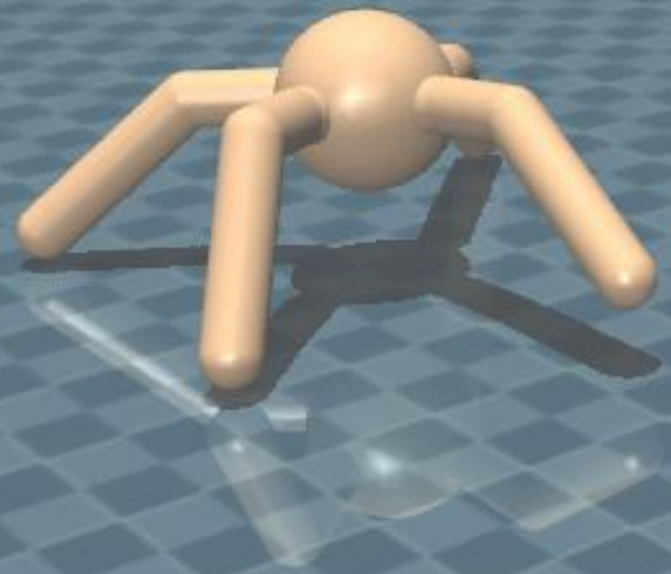
Submit these documents in a single zipped folder to **Moodle** at **Submission - Challenge 2** with the following naming convention:
`2026_micro_515_TEAMNAME_LASTNAME1_LASTNAME2.zip`

One submission per team is sufficient.

Challenge 2
Leaderboard

https://lis-epfl.github.io/micro-515-EvoRob/

will be updated on Monday, 23.03.2023



EPFL

MICRO-515 · EVOLUTIONARY ROBOTICS

Challenge Leaderboard

TEAMS	CHALLENGES	TOP SCORE	AVERAGE	MAX POSSIBLE
10	3	287.0	256.6	300

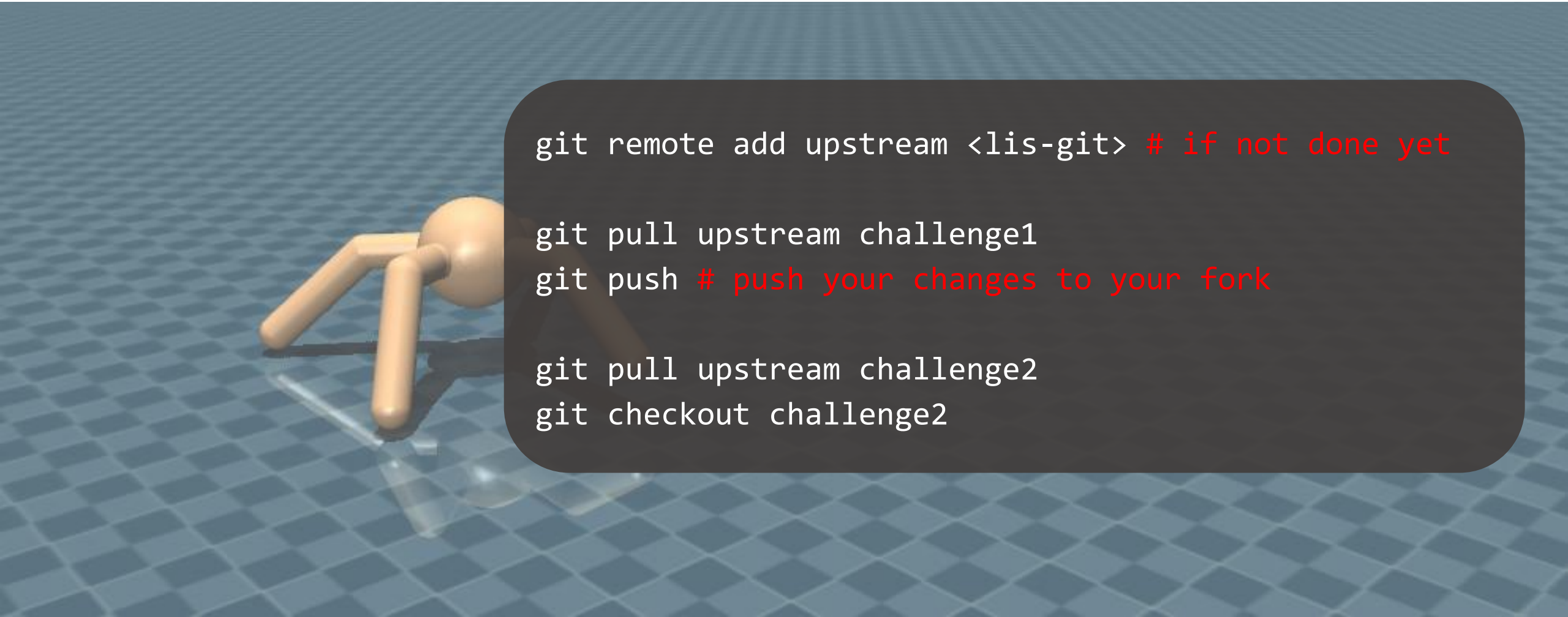
Search team...

Overall | Challenge 1 | Challenge 2 | Challenge 3

#	TEAM	CHALLENGE 1	CHALLENGE 2	CHALLENGE 3	TOTAL (NORM.)
1	NanoBot Crew	4990	330	810	287 / 300
2	Iron Wolves	4823	310	872	284.6 / 300
3	Helix Drive	3880	305	893	266.6 / 300
4	ByteForce	3950	275	901	260 / 300
5	Delta Robotics	3120	340	830	254.6 / 300
6	Sigma Dynamics	3600	290	868	253.8 / 300
7	Quantum Gears	4450	255	790	251.9 / 300

Challenge 2

Update your current repository version



```
git remote add upstream <lis-git> # if not done yet

git pull upstream challenge1
git push # push your changes to your fork

git pull upstream challenge2
git checkout challenge2
```