Learning Machines Demo 2

Food Collection

Team α - Mick IJzer, Leon Berberich, Justus Huebotter & Nikolay Polyanov

Agenda

- Problem definition
- Methodology
- Experimental setup
- Results
- Conclusions

Task & Problem Definition

- Task:
    - **Foraging** using Robobo
    - Robobo should be able to **detect objects** of the desired color in the arena and change it’s path in order to **collect** them
- Problem(s):
    - **Color sensitive object recognition** using **camera** of the mobile phone
    - (Population) **learning from experience** to increase target **objects** collected **per minute**
    - **Obstacle avoidance**

Problems Faced

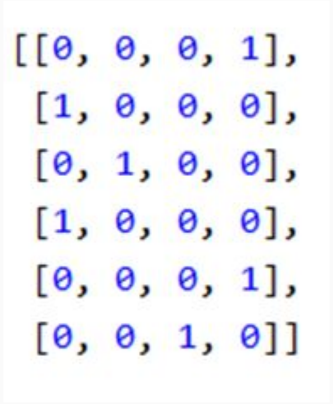
There were several challenges faced while working on the task:

- Python 3 & ROS 1 cannot use the camera
- Hardware Robobo sometimes goes slightly left
- Camera image needed cropping on sides
- Detection threshold needed adaption

Methods Overview

A combination of an Evolutionary Algorithm and Policy Iteration:

- Learning algorithm:
    - Evolutionary Strategy ($\mu + \lambda$)
    - Variation (crossover & mutation)
    - Selection (probabilistic parent & survival selection)
- Controller:
    - Deterministic policy π
    - Each row represents a state
    - Each column represents an action
    - Deterministic → Action *a* taken in some state *s* will always be the same for an individual



```
[[0, 0, 0, 1],
 [1, 0, 0, 0],
 [0, 1, 0, 0],
 [1, 0, 0, 0],
 [0, 0, 0, 1],
 [0, 0, 1, 0]]
```

Image Processing

- Cropped image
- OpenCV 2 - findContours
- Green in range [0, 100, 0] - [90, 255, 90]

Controller Representation

Actions:

- Forward
- Left turn
- Right turn

Bits included in state:

- IR sensor: object detected right or front
- IR sensor: object detected left
- Camera: target detected
- Memory: target seen in recent iterations
- Memory: target last seen on left or right

Deterministic policy π with 3 columns and $2^5 = 32$ rows.

There are $3^{32} = 1.85 \times 10^{15}$ possible combinations

Evolution Parameters & Metrics

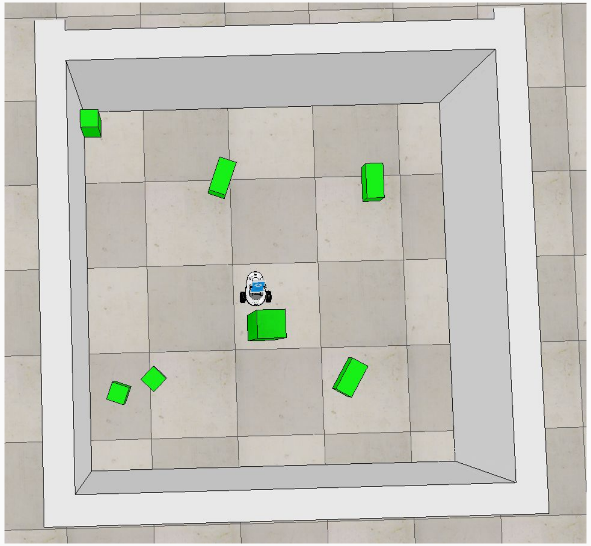
Phenotype	Behavior of the robot when in a state
Genotype	Deterministic policy (where every row is a state with 0 or 1 for each action)
Mapping	Readings & policy table are translated into robot movement by a python function
Fitness	+1 moving forward, +50 for every block collected, -50 and stop simulation on wall hit (100 steps in total → max fitness = ~400)
Crossover	Uniform (Every row is a copy from either P1 or P2)
Mutation	Shuffle a row with probability mu = 0.1
General Settings	Parent selection: prob. rank (2 parents) Survivor selection: prob. rank (10 individuals) Initialization: random Termination: 10 generations

Experimental Setup

Similar scenes were used both in simulation and real arena (size is the only difference):

- Desired food are green blocks
- On the touch, blocks are lifted in the air

At the start of every simulation the blocks are randomly scattered around the arena. Position of the robot and size of the arena remain constant.



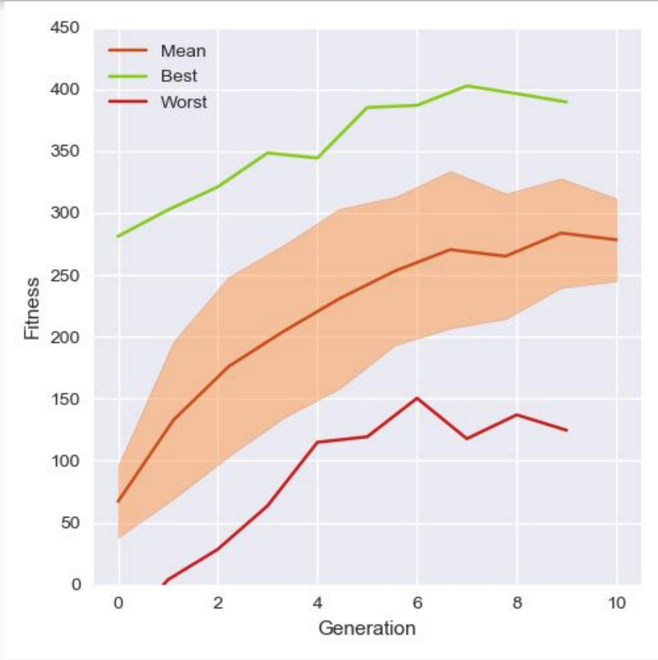
Results - Population

10 generations with 10 individuals per generation

7 reruns

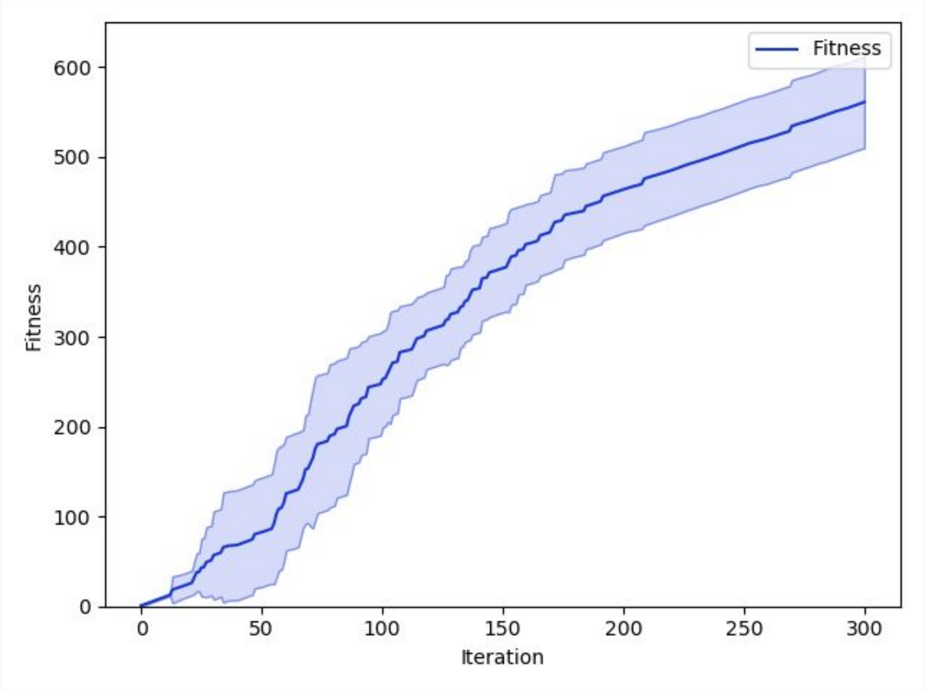
Graph shows:

- Mean of population mean fitness (red)
- Mean of population max fitness (green)
- Mean of population min fitness (darkred)
- Mean variation between the reruns (orange)



Results - Best Individual

Graph shows mean +- std fitness for 10 repeats a 300 steps



Conclusions

- Evolutionary Approach gives a promising learning curve for the population
- Best individual exhibits desired behavior
- Problems & task solved ✓
- Not enough time to experiment with ROS 2

Outlook:

- Expand minimalistic approach & extend policy slightly
- Explore neural net approach again

Thank you for your kind attention!

Questions?
Demonstration!