Evolutionary Computing

standard assignment

Dr. Karine Miras

AGENDA

> Interactive workshop

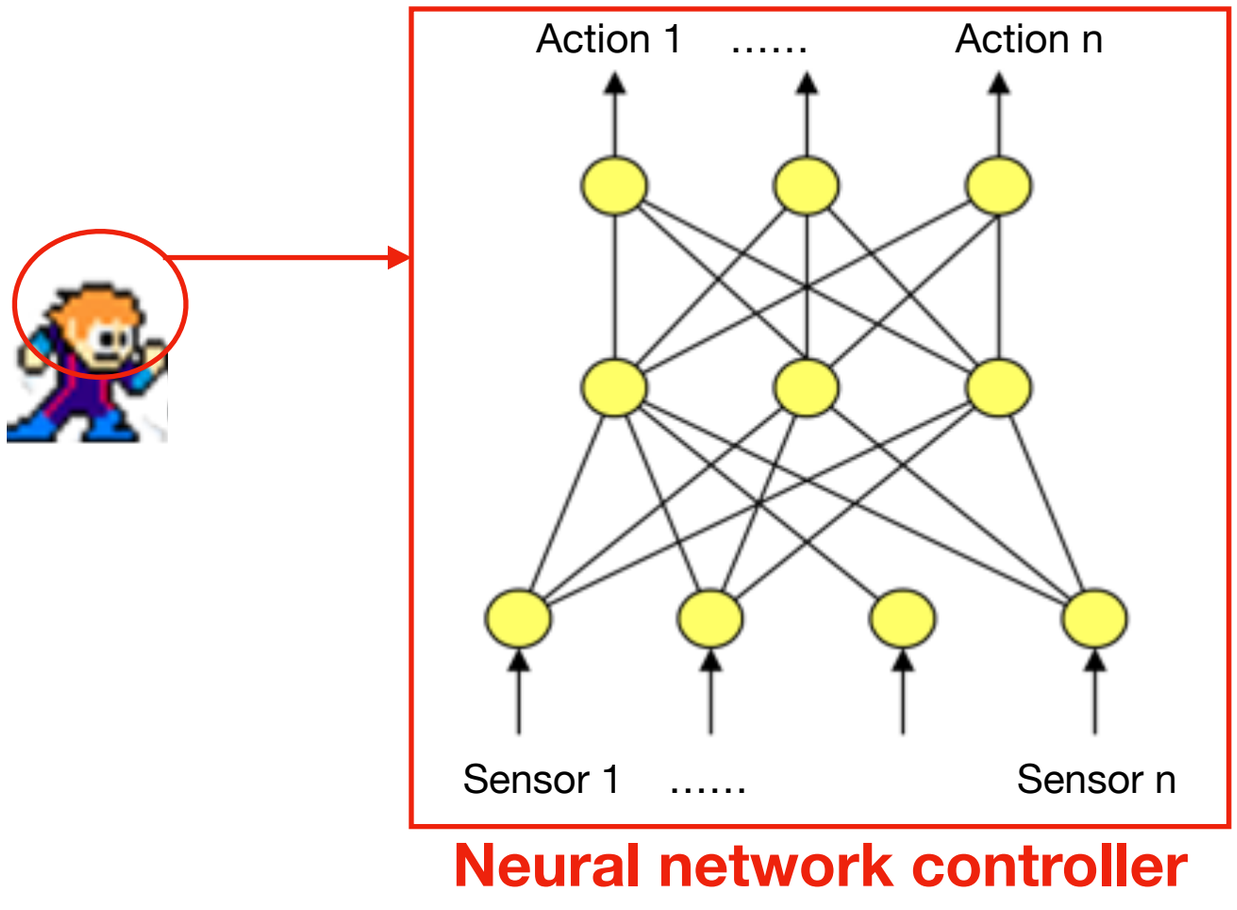
> Ask questions anytime:
raise your hand
or post it on Mentimeter (QR code below)

menti.com 4663 9025

GOAL

Use Evolutionary Computing to design **video game playing** strategies.

NEUROEVOLUTION: EVOLVE A NEURAL NETWORK (WEIGHTS AND/OR ARCHITECTURE)



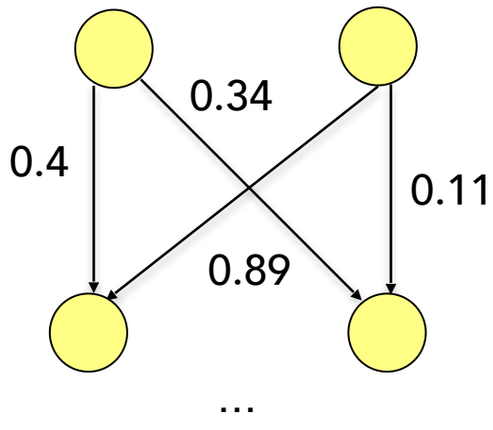
Neural network controller

Solution 1: 0.1, 0.3, 0.9, 0.6…

Solution 2: 0.4, 0.34, 0.89, 0.11…

…

Solution N: 0.23, 0.9, 0.4, 0.12…



EVOMAN: A VIDEO GAME PLAYING FRAMEWORK

- 8 2D-platform predator-prey **games**
- cloned from MegaMan II (only boss stages)
- made with pygame

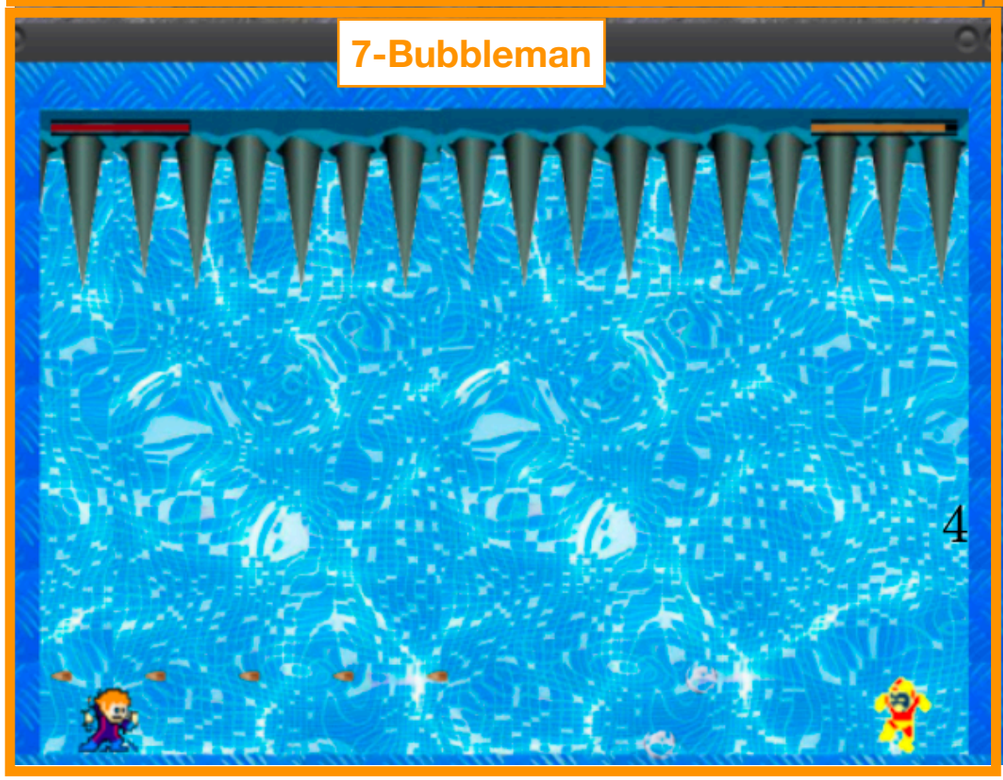
Why using EvoMan?

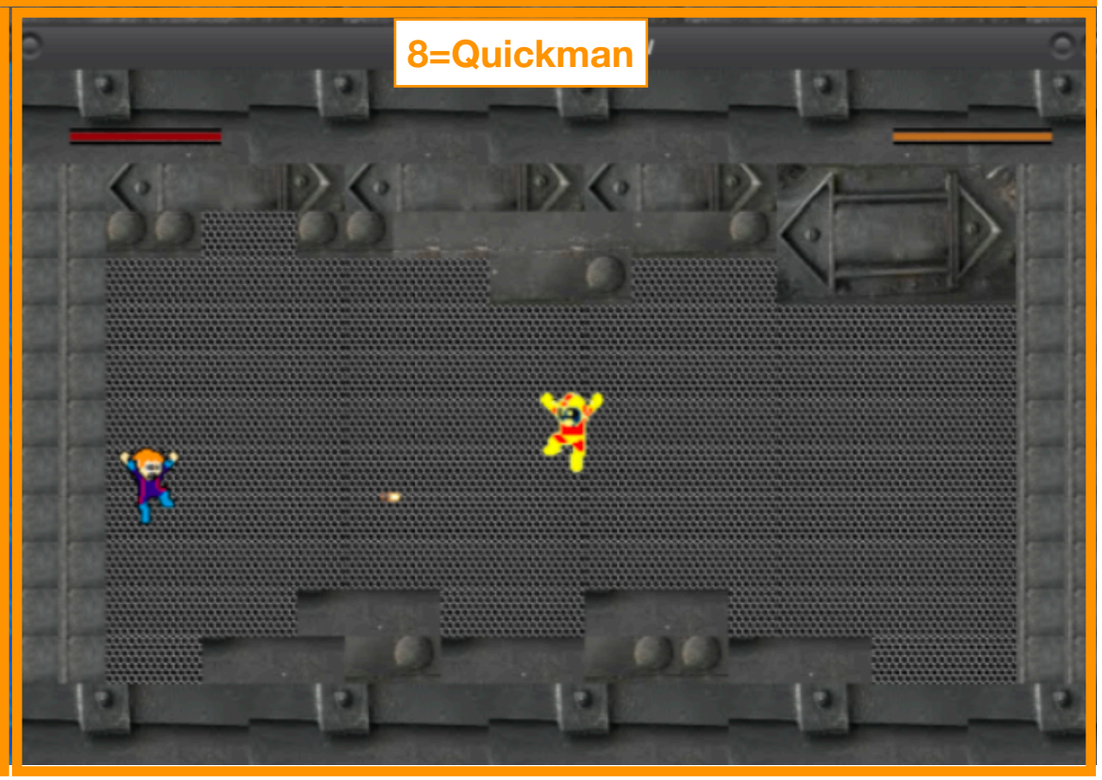
MegaMan is just a game, while EvoMan is a **framework** where we have also *tools* and environmental control.

https://github.com/karinemiras/evoman_framework

ENEMIES (GAMES)

ENEMIES (GAMES)





BUILD UP AN INTUITION

Watch demo: *controller_specialist_demo.py*

We need 3 **volunteers**!

human_demo.py

Use the **arrows left-right** to move,
space+up* to jump and **shift** to shoot.

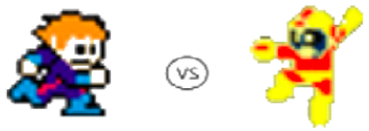
* perhaps only space?

SIMULATION MODES

Training objectives

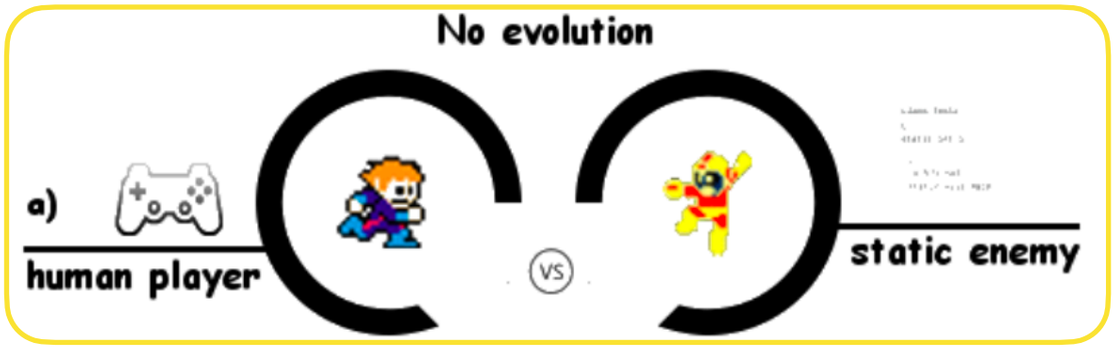
specialist

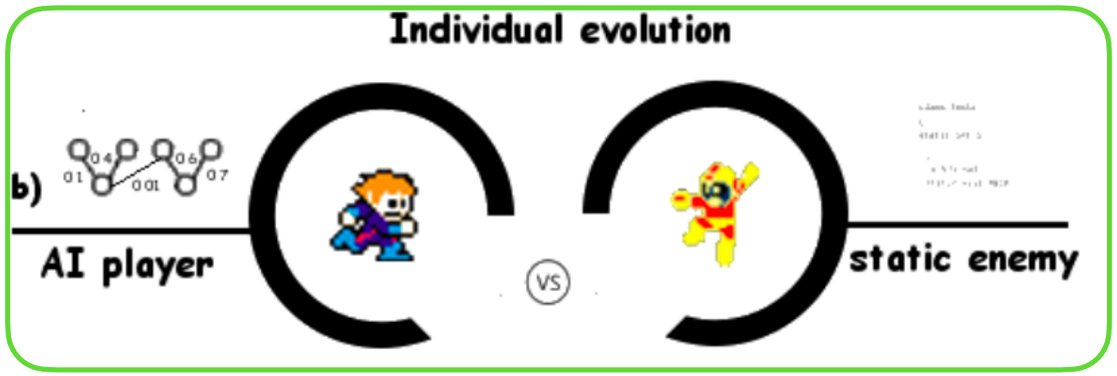
single objective

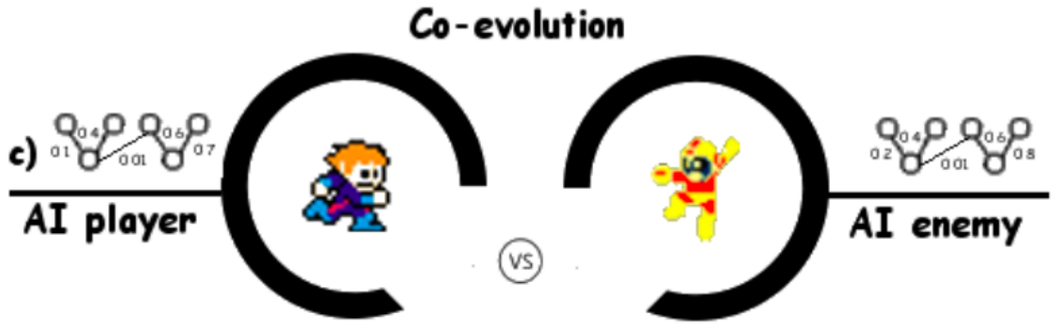


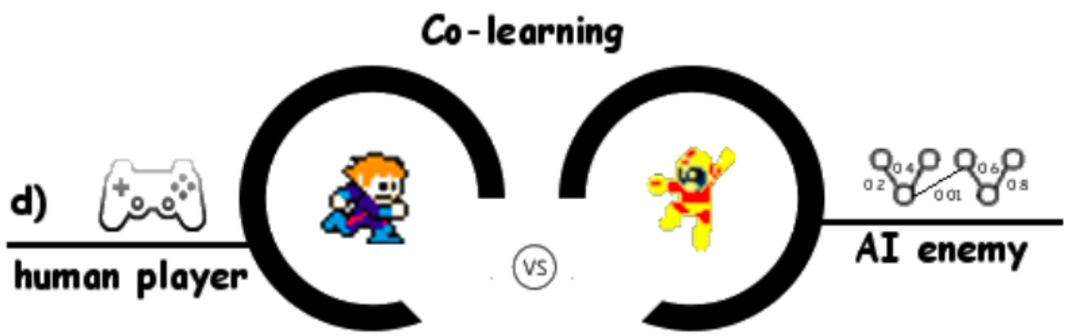
generalist

multi objective

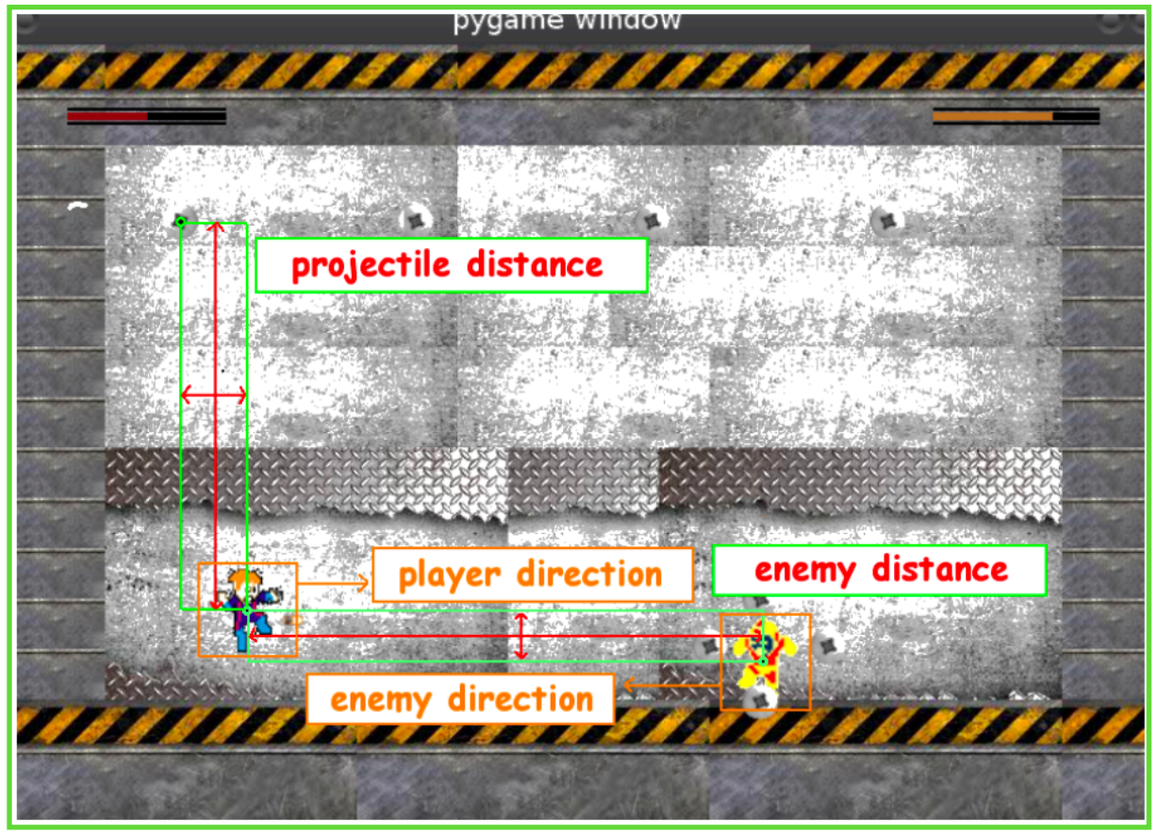








SENSORS AND ACTIONS



<u>20 sensors</u>:

16 for <u>distances</u> (x, y) from the player to the projectiles

2 (x, y) for <u>distances</u> from the player to the enemy

2 for concerning the <u>direction</u> the agents are facing

<u>5 actions:</u>

left

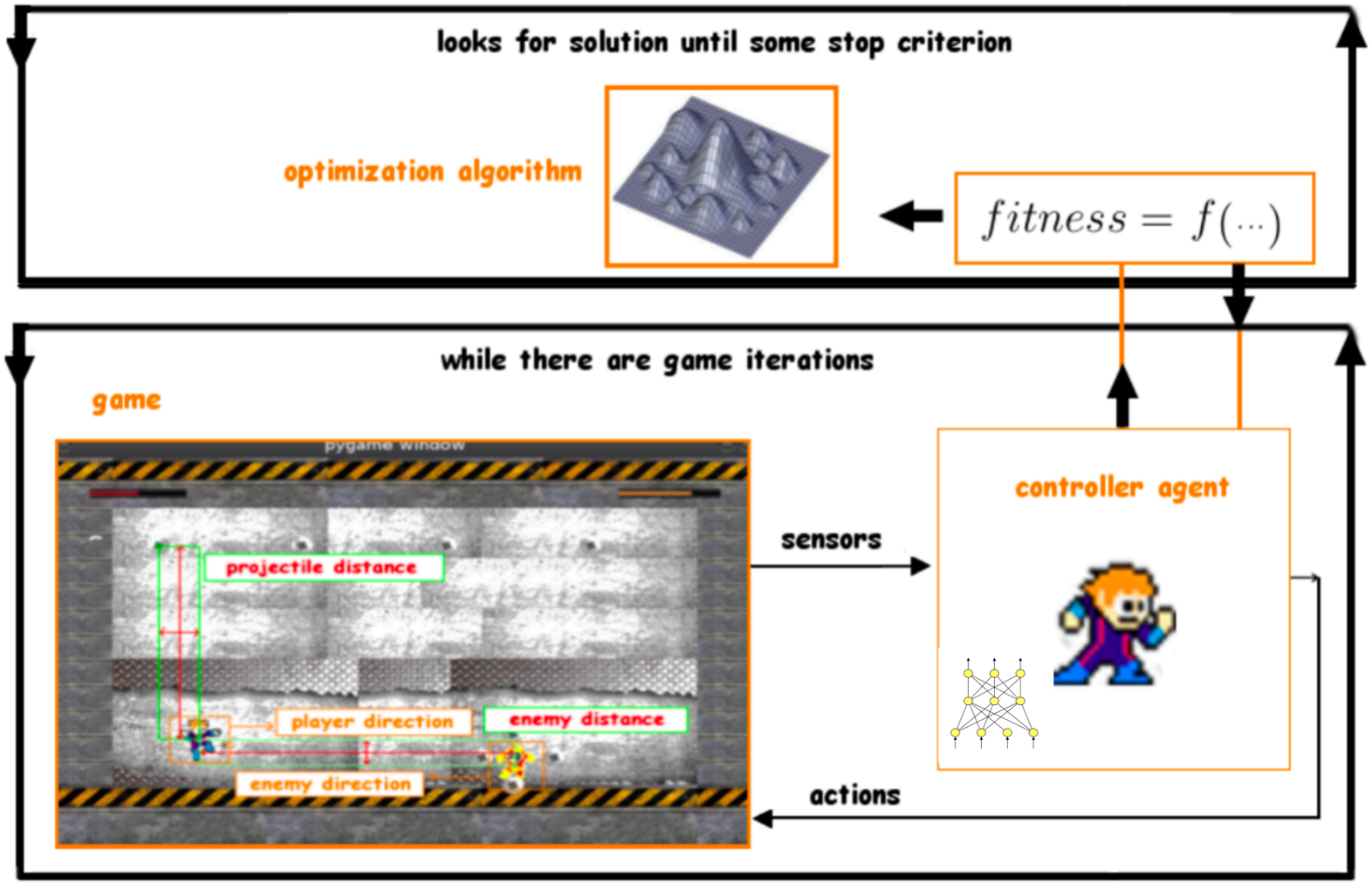
right

shoot

jump

release

SEARCH AND EPISODE



- player life
- enemy life
- game run time

Specialist

$$fitness = \gamma * (100 - e_e) + \alpha * e_p - \log t$$

γ = 0.9 α = 0.1

Generalist

$$f = \mu - \sigma$$

DEMOS

Start from:

- dummy_demo.py
  or
- optimization_dummy.py

```
from environment import Environment
env = Environment()
env.play()
```

fitness, player energy, enemy energy, game run time

Learn from examples:

- optimization_specialist_demo.py
- optimization_generalist_demo.py

FRAMEWORK PARAMETERS

```
experiment_name='test',
multiplemode="no",          # yes or no
enemies=[1],                # array with 1 to 8 items, values from 1 to 8
loadplayer="yes",           # yes or no
loadenemy="yes",            # yes or no
level=2,                    # integer
playermode="ai",            # ai or human
enemymode="static",         # ai or static
speed="fastest",            # normal or fastest
inputscoded="no",           # yes or no
randomini="no",             # yes or no
sound="on",                 # on or off
contacthurt="player",       # player or enemy
logs="on",                  # on or off
savelogs="yes",             # yes or no
clockprec="low",            # low or medium
timeexpire=3000,            # integer
overturetime=100,           # integer
solutions=None,             # any
player_controller=None,     # controller object
enemy_controller=None       # controller object
```

Possibility of non-deterministic environment

```
headless = True
```

SUPPLEMENTARY MATERIAL

- Manual of the framework: evoman1.0-doc.pdf (on GitHub)

  > Installation

  > You do not depend on the demos (manual is self-sufficient)

- Paper: *Evolving a Generalized Strategy for an Action-Platformer Video Game Framework*

  multi_evolution.pdf (on GitHub)

  To demonstrate feasibility and serve as theoretical baseline.

TASKS

Task I: 50%

+

Task II: 50%

+

(competition: potential extra points)

Delayed submission can lead to a score of 0 for the respective assignment!

TASK I: SPECIALIST AGENT

- Compare two EA methods - individual evolution mode with a single objective.

- 3 or more enemies using the *same* parameters.

- You are allowed but not required to use the provided code of the **neural network controller** (demo_controller.py).

- **WHAT TO HAND IN**: Report and code.

TASK II: GENERALIST AGENT

- Compare two EA methods - individual evolution mode using multiple objectives.
- 2 or more different groups of enemies for the training.
- Choose the number of enemies in a group to use for training/test.
- For at least one of the EAs, you are required to use the provided neural network (demo_controller.py) which uses one hidden layer with 10 hidden neurons.
- **WHAT TO HAND IN**: report, code, and a text file with your best solution containing the weights for the neural network. These weights will be used for the competition part.

TASK II: GENERALIST AGENT – COMPETITION

- Rank I:
  - number of defeated enemies
  - sum of player-life (number of energy points kept by the player – larger is better)
  - sum of time (duration of the match – lower is better)

- Rank II:

  Gain measure

The best three groups of each rank get extra points (non-cumulative):

the winner gets 1 point, silver medal gets 0.6 points, bronze medal gets 0.3 points

TASK II: GENERALIST AGENT – COMPETITION

To realize a final generalization test for generic agent controllers, a measure called Gain was used, as shown in Figure 9. The measure goes from -800 to 800, and the greater it is, the better is the generalization power of the agent.

$$g = \sum_{i=1}^{n} p_i - \sum_{i=1}^{n} e_i$$

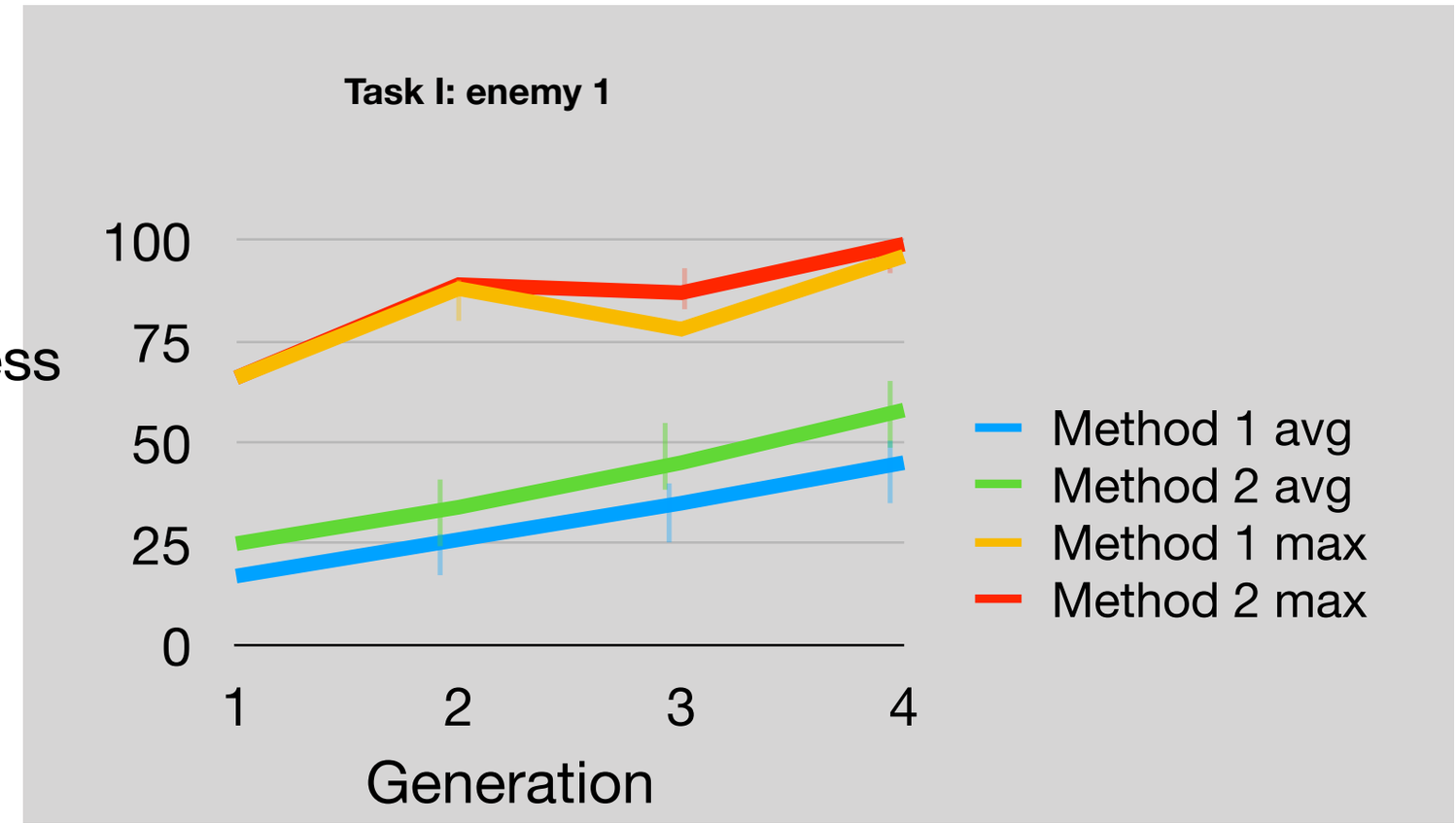
Figure 9: Gain measure

where e is the energy of the enemy, from 0 to 100; p is the energy of the player, from 0 to 100; n=8, is the total number of games in the test.

RESULTS PRESENTATION

- **Repeat** final **experiments** 10 times (independently) and consolidate.

For each task, algorithm, and enemy, plot the average and standard deviation of the mean and maximum **fitness** of the population across the generations using a **line plot.**



RESULTS PRESENTATION

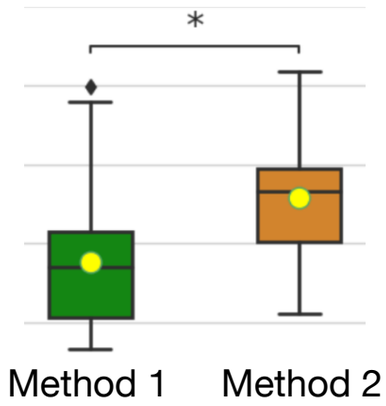
For each task, algorithm, and enemy/group, test your **best** solution for each of the 10 independent runs 5 times.

Calculate the average per run (among the 5). Plot the averages from the runs using a **boxplot.**

Additionally, do a statistical test to verify if the differences in average are significant.

Values to be plotted are the individual **gain** for task I, and general **gain** *or* number of **defeats** for task II.

Task I: enemy1/group1 (gain/defeat)



RESULTS PRESENTATION

For task II: Add a **table** containing the average (of 5 repetitions) energy points of player and enemy (for each of all enemies) for your VERY best solution (only one of the 10+10).

This is the solution you should submit to the **competition**!

Enemy	Player energy	Enemy energy
1	30	20
2	40	0
3	80	20
...		

RULES

- **not allowed** to copy code from the optimisation **demos**, but you may use any EA framework (Ex.: DEAP, Neat-Python).
- **allowed**, but not required, to use the default **fitness** functions.
- **not allowed** to **change** *evoman* folder. Could lead to FAILING.
- parameter **difficulty** of the game should be **2** and **contacthurt** should be set as “player”. Could lead to FAILING.
- You can use two **different** EAs, **or** the **same** EA with different **operators**.
- report any **changes** to other **default parameters**.

TIPS

- Large legends and small plots.

- Test using a small populations for few generations.

- Plan your experiments budget (use tests).

TIPS

- Organise output files.
- Consider normalizing the inputs for the controller.
- Group with people of diverse background.
- Use load_state() and save_state() for experiments recovery.
- Literature review, e.g., Julian Togelius and Kenneth Stanley.

LETS CHECK THE CODE. . .

MENTIMETER QUESTIONS