

EVOLUTIONARY COMPUTING - STANDARD ASSIGNMENT



AGENDA

> Workshop

> Hands-on

GOAL

Use EC to create **video game playing** strategies:

Task I: week 4 of the course, 25-09-2020, Friday 11:59pm

Task II: week 7 of the course, 12-10-2020, Monday 11:59pm

40% (Task I) + 60% (Task II)

* Each day of delay in the submission will result in a deduction of 0.5 points from your Task grade.

EVOMAN: A VIDEO GAME PLAYING FRAMEWORK

8 predator/prey **games** cloned from MegaMan II (only boss stages)

Why using EvoMan?

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

Watch demo: `controller_specialist_demo.py`

We need 3 volunteers!

Use the *arrows* to move,
space to jump and *shift* to shoot.

EVOMAN: SIMULATION MODES

Training objectives

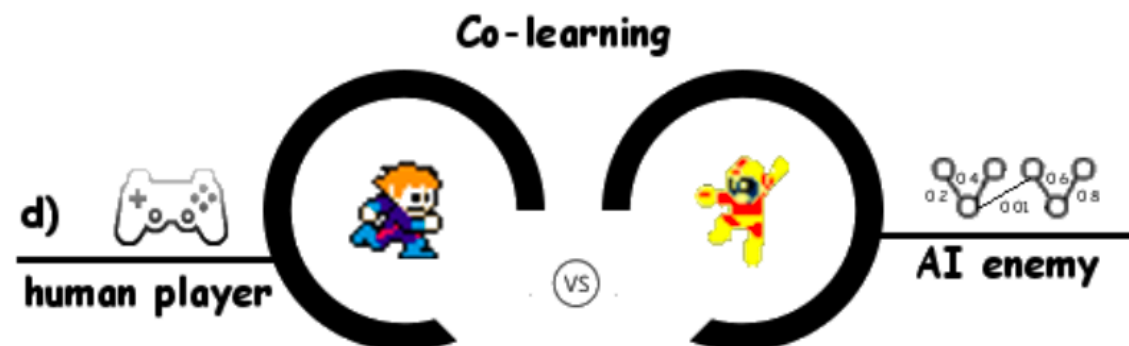
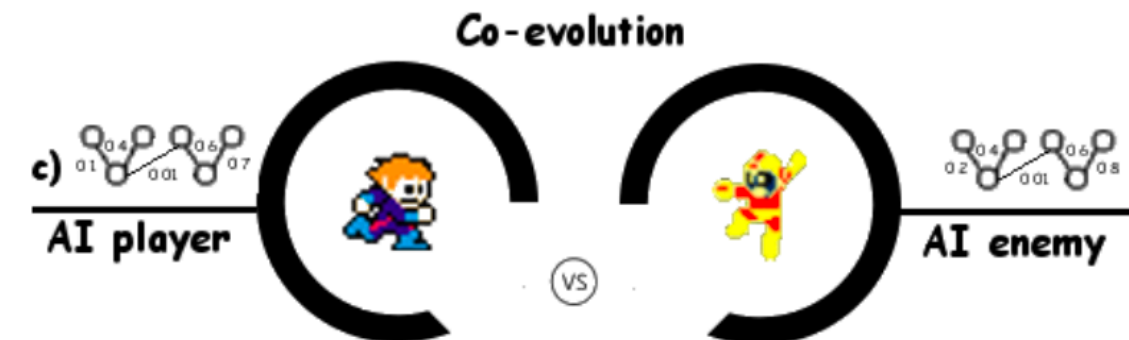
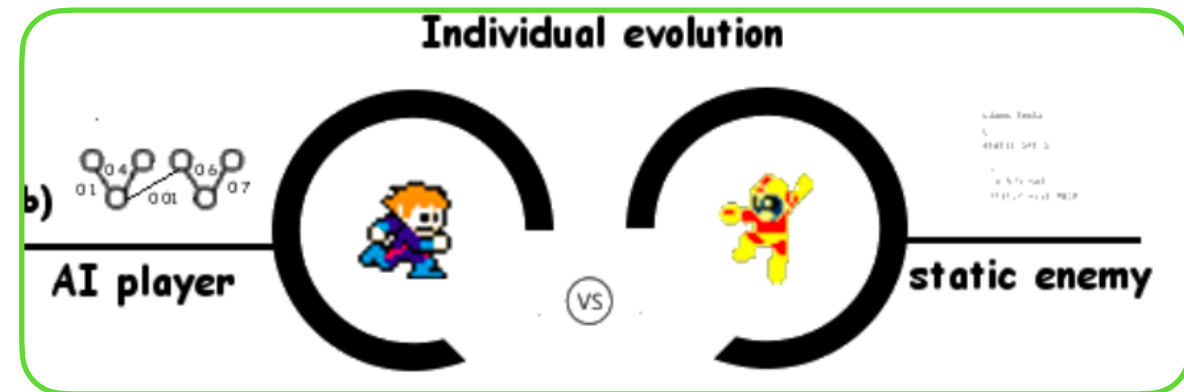
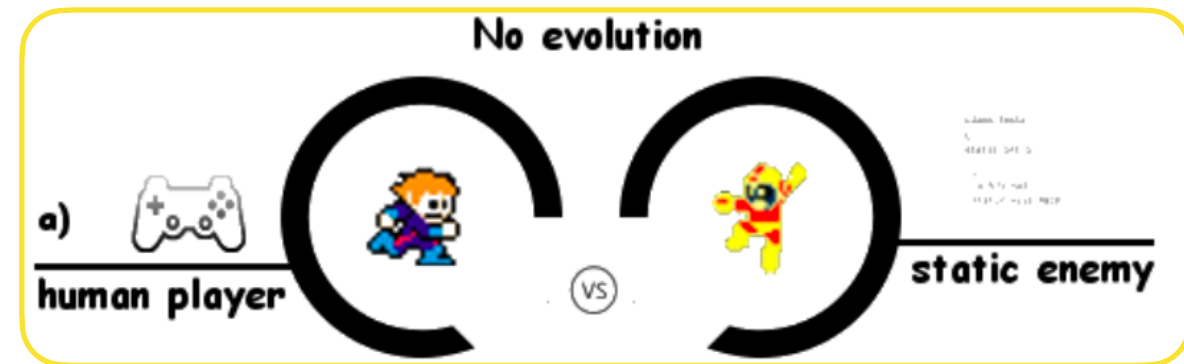
Task I: specialist

single objective

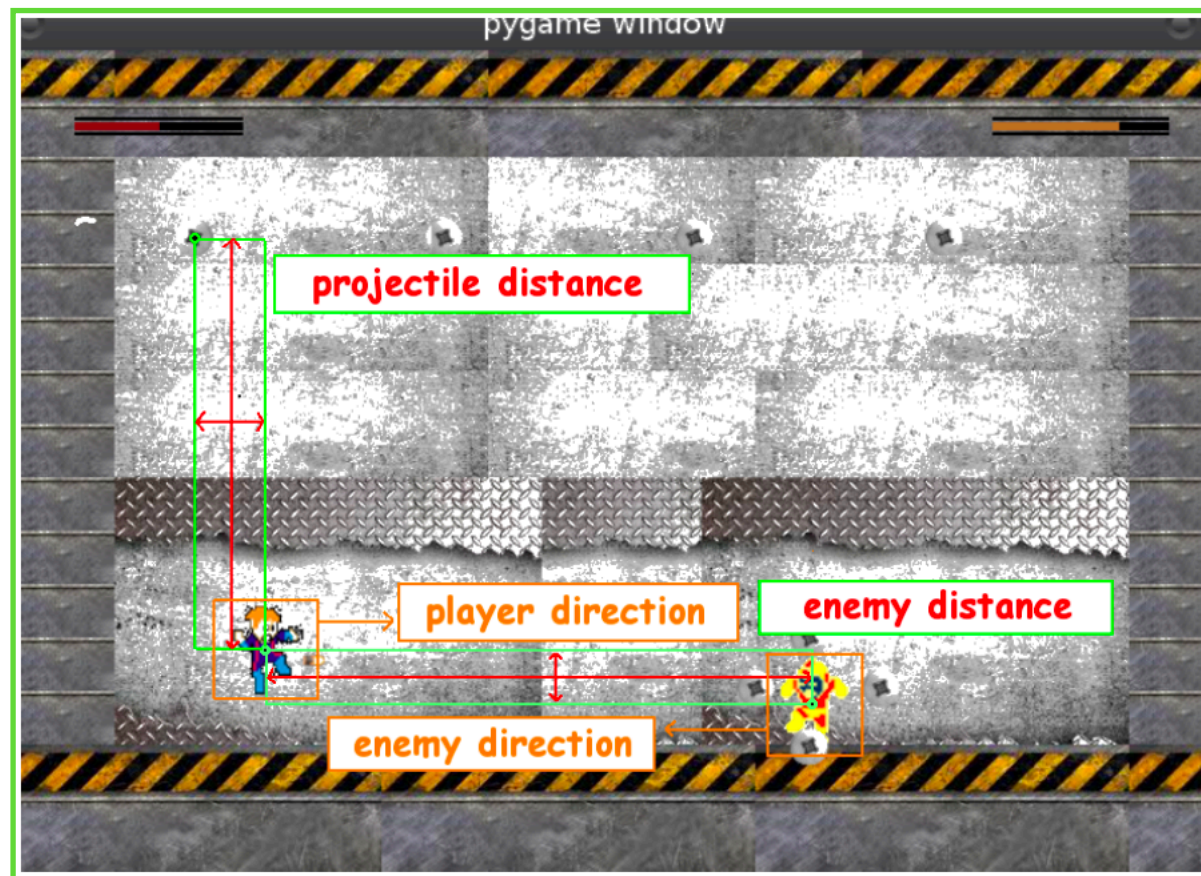


Task II: generalist

multi objective



EVOMAN: SENSORS AND ACTIONS



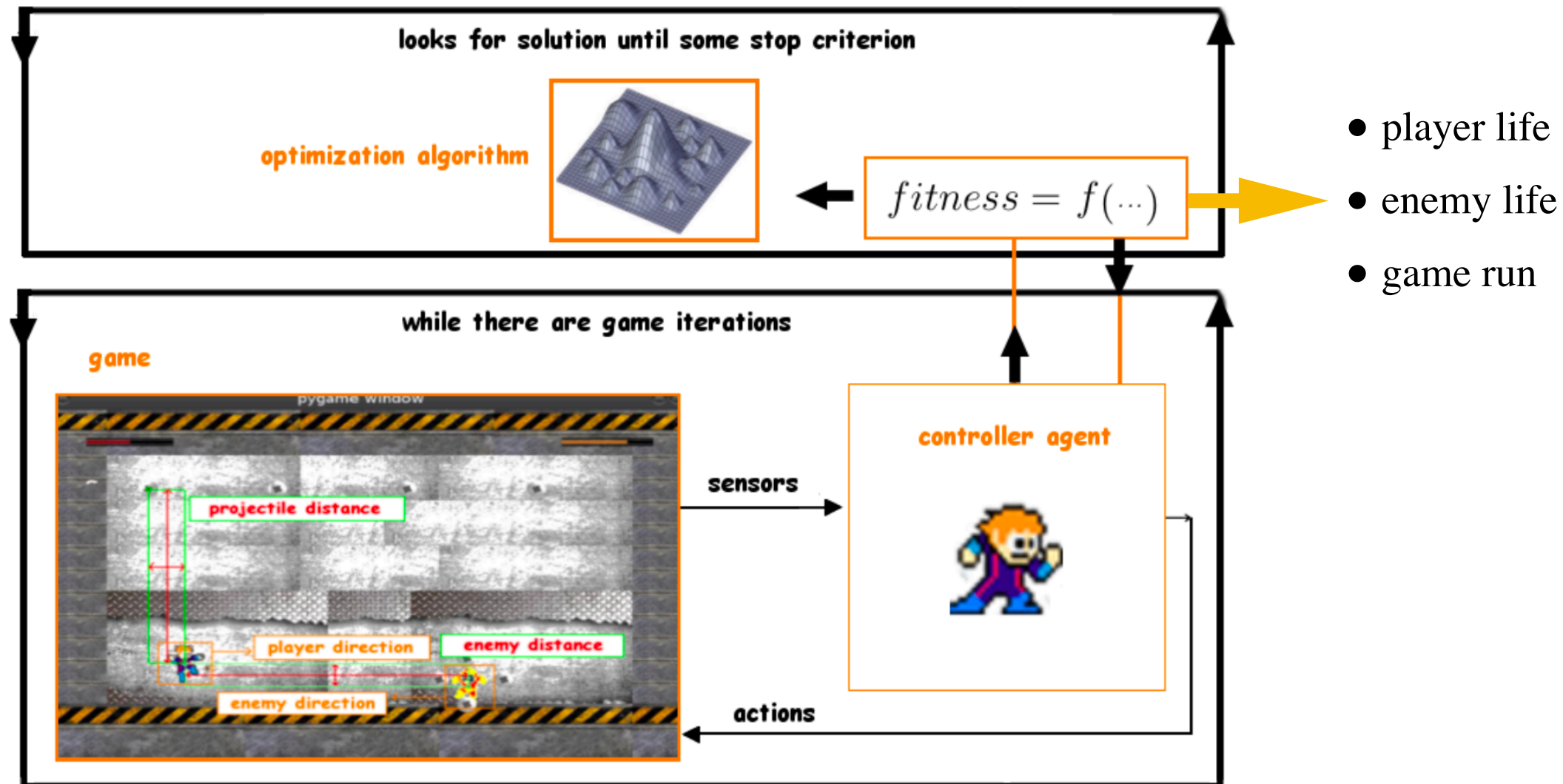
20 sensors:

- 16** for distances (x, y) from the player to the projectiles;
- 2** (x, y) for distances from the player to the enemy;
- 2** for concerning the direction the agents are facing.

Actions:

left, right, shoot, jump, release

EVOMAN: LOOP



RUN: run status: enemy: 6; fitness: 65.80; player life: 0; enemy life: 20; time: 491


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

$$\gamma = 0.9 \quad \alpha = 0.1$$

$$fitness' = \bar{f} - \sigma$$

EXAMPLES

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



fitness, player energy, enemy energy, game run time

```
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
```

- dummy_demo.py
- optimization_specialist_demo.py
- optimization_generalist_demo.py

SUPPLEMENTAR MATERIAL

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

For inspiration, and proof of feasibility:

- Baseline paper I: [multi_evolution.pdf](#) (on GitHub)
 - ~~Baseline paper II: [co_evolution.pdf](#) (on GitHub)~~
-
- Reporting Research (on Canvas)

TASK I: SPECIALIST AGENT

- **WHAT:** two EAs that use the simulation mode “Individual evolution” of the framework to train a specialist agent.
- **HOW:** 3 or more enemies (games) to experiment with your algorithms (using exactly the same algorithm parameters for all enemies). You are allowed, but not required, to use the code of the neural network (demo_controller.py) available in the scripts of demonstration.
- **WHAT TO HAND IN:** The report as specified below and your code.
- **HOW TO HAND IN:** Submit a single compressed file named groupnumber.zip containing: the report named as groupnumber.pdf and your code (include the evoman folder, but no demos).

TASK II: GENERALIST AGENT

- **WHAT:** two EAs that use the simulation mode “Individual evolution” of the framework to train a generalist agent.
- **HOW:** Compare the use of two different groups of enemies for the training. You are free to choose the number of enemies(games) in a group to use for training/test. You are required to use the neural network (demo_controller.py) available in the script of demonstration (optimization_generalist_demo.py), using one hidden layer with 10 hidden neurons.
- **WHAT TO HAND IN:** The report as specified below, your code, and a text file with your best solution containing the weights for the neural network.
- **HOW TO HAND IN:** Submit a single compressed file named groupnumber.zip containing: the report named as groupnumber.pdf, your code (include the evoman folder, but no demos), and the solution named as groupnumber.txt.

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 get-time (duration of the match – lower is better)

- Rank II:

Gain measure

The best three groups of each rank get extra points (no-cumulative):

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

TASK II: GENERALIST AGENT

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.

REPORTS

- 3 pages of content plus 1 for the cover
- GECCO template
- Content:
 - Introduction (make sure to define a clear research question or goal, cf. Reporting Research document).
 - Methods: your algorithm and its motivation, parameter settings, experimental setup, fitness function, budget, etc. Make sure everything is reproducible with the information presented!
 - Results and discussion: compare the results of your EAs and discuss why they outperform (or not) each other; compare your results with the baseline and discuss why your results are better/equal/worse; etc.
 - Conclusions
 - Literature list / bibliography: cite all works you are using in your project.

RULES

- You are not allowed to use the EA code available in the scripts of demonstration. That is, you need to implement some other EA(s), but you may use any EA framework if you wish (Ex.: DEAP).
- You are allowed, but not required, to use the default fitness functions of the framework.
- You are not allowed to make changes to the source code of the framework (anything in the 'evoman' folder). Making such changes would be qualified as fraud, leading you to FAIL the assignment.
- For the “two EAs”, you can use two different EAs, or the same EA with differences such as, the evolutionary mechanisms, parameter tuning, etc.
- For each Task, algorithm, and enemy, you should repeat your final experiments 10 times (independently) and your report should present the statistics based on these 10 runs.

RULES

- For each Task, algorithm, and enemy, plot the average/standard deviation of the mean and maximum fitness of the agents across the generations using a line-plot for the mean and maximum values.
- For each Task, algorithm, and enemy/group, test your final best solution for each of the 10 independent runs 5 times, and plot the averages of these times in a box-plot. Additionally, do a statistical test to verify if the differences in average are significant. Values are energy points for task I, and gain or defeats for task II.
- 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 $[10+10]*2$). This is the solution you will submit to the competition.
- The environmental parameter of level of difficulty of the game should be 2 for all Tasks, as well as the parameter contacthurt should be set as “player” (both these values are set by default).
- As for the other environmental parameters, you are free to choose their values, but required to report any changes to their default values.

- Note that you are not allowed to use the EA code of the demos. They are not part of the framework, but only a demonstration of how one specific user could use the framework. The cleanest example to start from is `dummy_demo.py`. We emphasize that all you need to know to start your own experiments is in the manual (`evoman1.0-doc.pdf`), and it does not depend on the demos.
- Add legends with large letters to the plots, so that you can reduce the plot size while still making it readable, saving a lot a space.
- Test your algorithm(s) at first using a small population for few generations. This also applies for tuning.
- Remember to count the time of some of your experiments and take this into consideration to plan carefully your budget.

- Be organized concerning the output files of your experiments.
- It might be interesting to normalize the inputs for the controller (as done in the default controller).
- Try to make a group with people of diverse scientific background, e.g. an AI student, a CS student, and a biology student
- Use the methods `load_state()` and `save_state()` to allow experiments recovery
- Do a literature review using the references of the baseline papers. One key author to have in mind is Julian Togelius.

LETS CHECK THE CODE QUICKLY...

HANDS-ON

1. Form a **group** with 4 members (3 is less preferable, but also acceptable)
2. **Read** the assignment THOROUGHLY (standard_assignment_taskI.pdf and standard_assignment_taskII.pdf)
3. **Do** the items in **section 3** (Preparation) of the assignment
4. **Ask** us questions/help (if you want)

TAs

Karine Miras: karine.smiras@gmail.com

Arwin Gansekoele: awgansekoele@gmail.com

Justus Huebotter: huebotter@outlook.com