TP - Machine Learning

LE ROY-NIVOT Mathis

Dependencies installation

```
python3 -m pip install -r requirements.txt
```

TP #1 - Genetic Algorithm: Scorpio 🎯

Execute program

```
cd tp-1-scorpio
python3 main.py
```

The program will ask you the Energy you want to reach

Dataset

The following arguments will have an influence of all algorithm behaviour / result :

```
EARTH_GRAVITY_CONST = 9.81
MOON_GRAVITY_CONST = 1.62
JUPITER_GRAVITY_CONST = 24.80

PROBABILITY_TO_MUTATE = 0.1
PROBABILITY_TO_KEEP_GRATED = 0.2
PROBABILITY_TO_KEEP_NON_GRATED = 0.05
POPULATION_AMOUNT = 500
GENERATION_MAX_AMOUNT = 100
FITNESS_MAX = 100
```

Algorithm info

This algorithm is composed with 4 classes:

- `Arrow` which defines the arrow properties (*material, length, diameter & mass*)
- `Scorpio` which defines scorpio's properties (*arrow, material, arm_length, arm_base, arm_height, string_length, angle...*)
- `Material` which defines material's properties (*mass, young module & poisson coefficient*)
- `Launch` which gives all the information about scorpion shoot (*impact energy*) according to the Scorpio object & the gravity constant given to the (Launch) construcor `def __init__()`

Algorithm logic

The fitness function `get_fitness()` function allow to know, with the returned value, if the fitness of the launch is sufficient by giving the launch, a grade.

This function is based on the lauch object givent to this function. The objective is to reach the energy (given by the user when the program start) at the arrow's impact.

The fitness fluctuate between 0 and 100, with 100 the maximum, depending on the distance to the wanted energy value.

TP #2 - Ant Colony Optimization 🐜

> Execute program

```
cd tp-2-ant-aco
python3 main.py
```

Dataset

The following arguments will have an influence of all algorithm behaviour / result:

```
NODE_MAX_QTY = 50
CENTRALISATION_MAX = (NODE_MAX_QTY / 2)
MAX_LENGTH_AMOUNT = 100

TRY_AMOUNT = 100
ANT_QTY = 100

# Start where all ants start from...
START_NODE = 0
# Finish node where the food is...
FINISH_NODE = 19
```

Algorithm info

This algorithm take place in one file, `main.py` and contains multiple functions (describe bellow) and constant variables:

- `NODE_MAX_QTY` which defines the maximum amount of nodes in the `nx.Graph()`
- `MAX_LENGTH_AMOUNT` which defines the maximum length between two given nodes
- `TRY_AMOUNT` which defines the amount of attempts for each ant of the program (to find the food node)
- `ANT_QTY` which defines the amount of ants
- `START_NODE` which defines the node in the `nx.Graph()` where all ants start
- `FINISH_NODE` which defines the node in the `nx.Graph()` where all ants need to find

Algorithm logic

1rst Step

The graph (with nodes & edges) is initialized in the `init()` function. The graph is initialized by using `networkx` library.

2nd Step

Then, the `launch()` function is called to calculate / find the shortest path of the generated graph (in the `init()` function) with the logic as follow:

```
for i in range(TRY_AMOUNT):
        for j in range(ANT_QTY):
            node = START_NODE
            visited_nodes = [START_NODE]
            while node != FINISH_NODE:
                node = choose_next_node(node, visited_nodes)
                if node == -1:
                    break
                visited_nodes.append(node)
            if node == FINISH_NODE:
                spread_pheromone(visited_nodes)
        evaporate_pheromone()
    get_shortest_path()
```

3rd Step

If the path is founded, then the graph with the shortest path (in red) is drawed, and ‘unused’ paths will be showed in black.