Assignment 3 **Methods inspired from nature**

Aims:

To solve a problem using Evolutionary Algorithm (EA), and a Hill Climbing Method.

Task:

Specify, design and deploy an application in python that solves your assigned problem using two methods: an Evolutionary Algorithm and a Hill Climbing method. The applications should follow the following conditions:

1. It must have a nice layered architecture with a GUI interface. The graphic interface should not freeze when the numerical algorithm runs. The user should get information about the run (best found solution until then, …) and should be able to stop it at any time.
2. the input data and the specific parameters (probability of mutation and crossover, population size and number of generations) should be introduced by the user in a form.
3. Separate some specific validation tests will be performed and the results will be displayed - the average and standard deviation for the best solutions found by your algorithm after 1000 evaluations of the fitness function in 30 runs, with populations of 40 individuals.
4. A plot depicting the fitness variation during the validations.

Points:

- 40 points - for each algorithm.
- 12 points for the architecture and for the quality of your application (GUI, fork the process).
- 8 points the statistical validation

Time:

The deadline is at the beginning of the fifth laboratory.

General hints:

- Do NOT solve the problems with other methods. You will not be granted points if you do this.
- For the GUI install the package qtpy under anaconda. Here is a tutorial for PyQt5 . In the examples use ***qtpy*** instead of ***PyQt5*** when you import the module.
- For the EA consider a proper representation (for example a n-tuple of permutations).

Problem:

An Euler square:

Consider a natural number $n \in N^*$ and two sets $S = \{s_1, ..., s_n\}$, $T = \{t_1, ..., t_n\}$ each consisting of n symbols.

Generate an Euler square matrix A from $M_{(n,n)}$. In each square is a pair $a_{i,j} = (s_{i,j}, t_{i,j})$, where $s_{i,j} \in S$, $t_{i,j} \in T$ fulfilling the following constraints:

1. $\forall i \in \{1, ..., n\}$, $\{s_{i,k} \in S \mid k = \overline{1,n}\} = \{s_{k,i} \in S \mid k = \overline{1,n}\} = S$,
   $\{t_{i,k} \in T \mid k = \overline{1,n}\} = \{t_{k,i} \in T \mid k = \overline{1,n}\} = T$,
2. $a_{i,j} \neq a_{k,l}$, $\forall i,j,k,l \in \{1, ..., n\}$, with $i \neq k \vee j \neq l$ $((s_a, t_a) = (s_b, t_b)$ *if* $s_a = s_b \wedge t_a = t_b)$.

If for a n such a matrix is not found, print a proper message in a popup form.

Example:

1, 1	2, 3	3, 2
2, 2	3, 1	1, 3
3, 3	1, 2	2, 1

(a)

1, 1	2, 3	3, 4	4, 2
2, 2	1, 4	4, 3	3, 1
3, 3	4, 1	1, 2	2, 4
4, 4	3, 2	2, 1	1, 3

(b)

1, 1	4, 5	2, 4	5, 3	3, 2
5, 4	2, 2	1, 5	3, 1	4, 3
4, 2	5, 1	3, 3	2, 5	1, 4
3, 5	1, 3	5, 2	4, 4	2, 1
2, 3	3, 4	4, 1	1, 2	5, 5

(c)

for the case (a) we could have 6 permutation in an individual (3 for each component):

$$\{(1,2,3),\ (2,3,1),\ (3,1,2),\ (1,3,\ 2),\ (2,1,3),\ (3,\ 2,1)\}$$