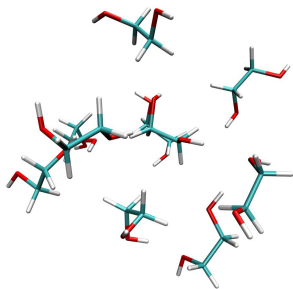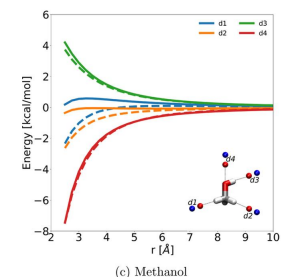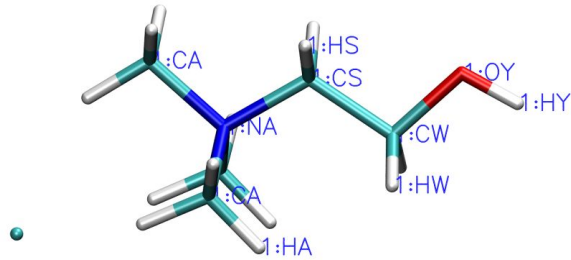# Lab Session 4

Obtain χ and η by reproducing:
1) QM/EE interaction energies
2) QM static polarizability

The goal of this lab will be that of parameterizing the FQ parameters χ and η for a Deep Eutectic Solvent by using a Genetic Algorithm.
Our training set will be *QM/EE interaction energies* and *QM polarizabilities*
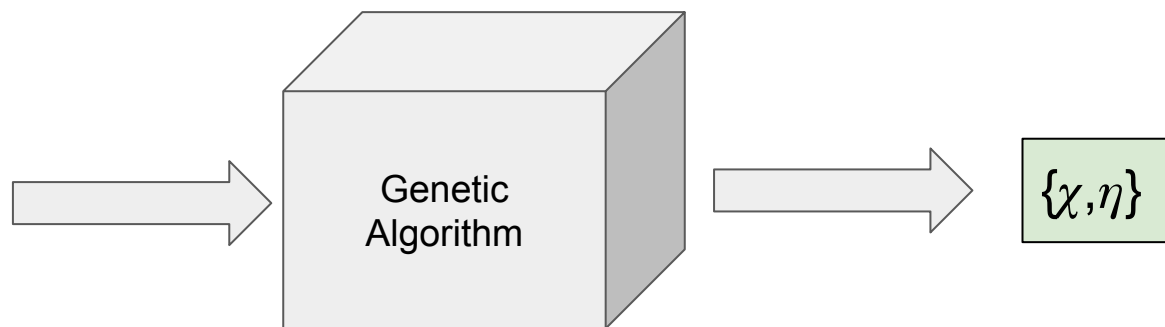
(c) Methanol

Genetic Algorithm

$\{\chi, \eta\}$

genes: χHS, ηHS, χOY, ηOY, …

i-th parent= [χHS, ηHS, χOY, ηOY, …]$_i$

M-th
Generation

parent 1

parent 2

.
.
.

parent N

Generation exploitation

genes: χHS, ηHS, χOY, ηOY, …

i-th parent= [χHS, ηHS, χOY, ηOY, …]$_i$

M-th
Generation
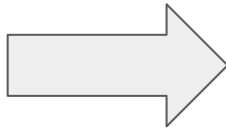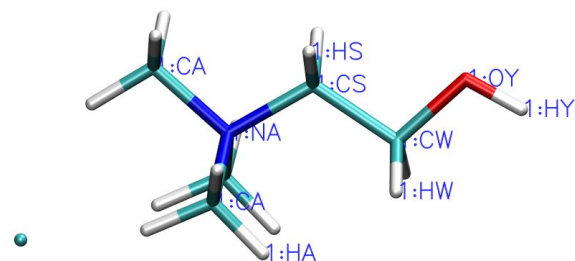
parent 1

parent 2

.
.
.

parent N

1st parent

Create FQ/EE calculation from each translated dipole geometry

Send the FQ/EE calculation

Collect the FQ/EE interaction energy

Compare to QM/EE interaction energy: compute a fitness value

genes: χHS, ηHS, χOY, ηOY, …

i-th parent= [χHS, ηHS, χOY, ηOY, …]$_i$

M-th
Generation

parent 1

parent 2

.
.
.

parent N

2nd parent

Create FQ/EE calculation from each translated dipole geometry

Send the FQ/EE calculation

Collect the FQ/EE interaction energy

Compare to QM/EE interaction energy: compute a fitness value

genes: χHS, ηHS, χOY, ηOY, …

i-th parent= [χHS, ηHS, χOY, ηOY, …]$_i$

M-th
Generation

parent 1

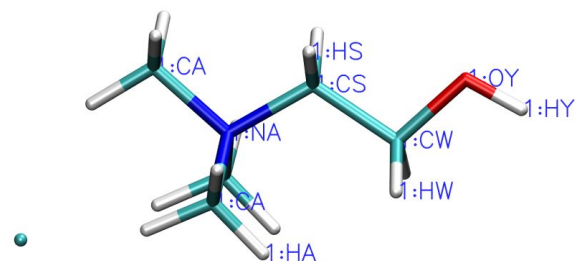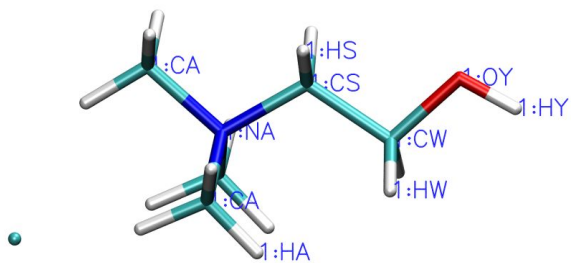parent 2

.
.
.

parent N

N-th parent

Create FQ/EE calculation from each translated dipole geometry

Send the FQ/EE calculation

Collect the FQ/EE interaction energy

Compare to QM/EE interaction energy: compute a fitness value

genes: χHS, ηHS, χOY, ηOY, …

i-th parent= [χHS, ηHS, χOY, ηOY, …]$_i$

M-th
Generation

parent 1

parent 2

.
.
.

parent N

Generation exploitation

genes: χHS, ηHS, χOY, ηOY, …

i-th parent= [χHS, ηHS, χOY, ηOY, …]$_i$

M-th
Generation

parent 1

parent 2

.
.
.

parent N

Generation exploitation

Parents selection (select fittest individuals)

genes: χHS, ηHS, χOY, ηOY, …

i-th parent= [χHS, ηHS, χOY, ηOY, …]$_i$

M-th
Generation

parent 1

parent 2

.
.
.

parent N

Generation exploitation

Parents selection (select fittest individuals)

Cross-over and mutation
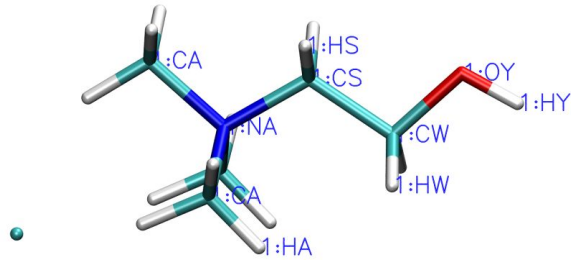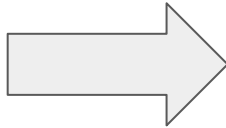
New individuals

How does a cycle of the GA look like?

genes: χHS, ηHS, χOY, ηOY, …
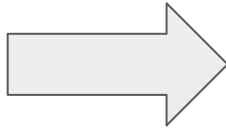
i-th parent= [χHS, ηHS, χOY, ηOY, …]$_i$

(M+1)-th Generation

parent 1

parent 2

.
.
.

parent N

Generation exploitation

Parents selection (select fittest individuals)

Cross-over and mutation

New individuals

genes: χHS, ηHS, χOY, ηOY, …

i-th parent= [χHS, ηHS, χOY, ηOY, …]$_i$
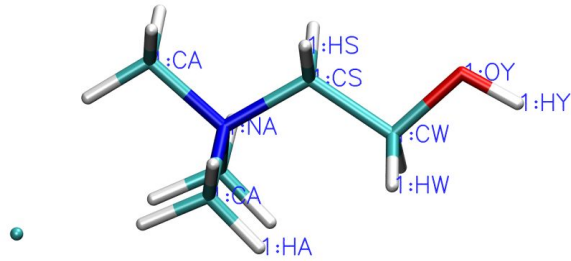
M-th Generation

parent 1

parent 2

.
.
.

parent N

Generation exploitation

genes: χHS, ηHS, χOY, ηOY, …

i-th parent= [χHS, ηHS, χOY, ηOY, …]$_i$

M-th Generation

parent 1

parent 2

.
.
.

parent N

i-th parent

Create FQ/EE calculation from each translated dipole geometry

Send the FQ/EE calculation

Collect the FQ/EE interaction energy

Compare to QM/EE interaction energy: compute a fitness value

How do we perform the FQ calculations?

mamaFQ

Program by Tommaso Giovannini
with contributions by Luca Bonatti,
Piero Lafiosca, and Luca Nicoli
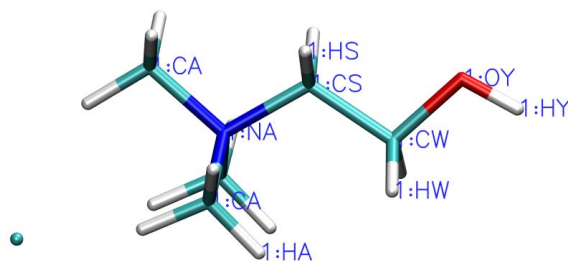
polarizable_embedding_class: gathers information about
1) Type of the force field (FQ in our case)
2) atomtypes of the system
3) electronegativity of each atomtype
4) chemical hardness of each atomtype
5) Some other stuff for other polarizable embedding models different from FQ

Run ***ga_cycle/help_scripts/setup_pe.py*** to see how does an initialized forcefield object look like

```
python3 setup_pe.py
---polarizable force field---
 force_field : fq
 atomtypes   : 'CS'       , 'HS'       , 'NA'       , 'CW'       , 'HW'       , 'OY'       , 'HY'       , 'CA'       , 'HA'       , 'Cl'
 chi         : 0.02900    , 0.47900    , 0.05400    , 0.96100    , 0.84800    , 0.72400    , 0.95400    , 0.34900    , 0.84500    , 0.58900
 eta         : 0.30600    , 0.24500    , 0.52100    , 0.80200    , 0.10000    , 0.70500    , 0.24000    , 0.48200    , 0.50500    , 0.29500
```

genes: χHS, ηHS, χOY, ηOY, …

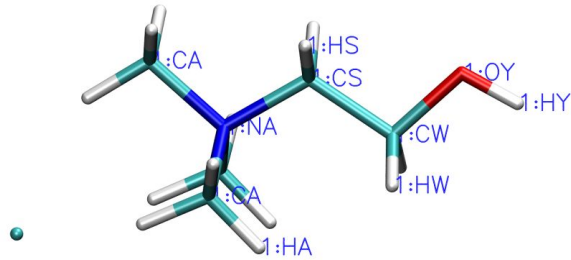i-th parent= [χHS, ηHS, χOY, ηOY, …]$_i$

(M+1)-th Generation

parent 1

parent 2

.
.
.

parent N

Generation exploitation

Parents selection (select fittest individuals)

Cross-over and mutation

New individuals

genes: χHS, ηHS, χOY, ηOY, …

i-th parent= [χHS, ηHS, χOY, ηOY, …]$_i$

https://pygad.readthedocs.io/en/latest/

genes: χHS, ηHS, χOY, ηOY, …

i-th parent= [χHS, ηHS, χOY, ηOY, …]$_i$

Ahmed Gad

https://pygad.readthedocs.io/en/latest/

How does a cycle of the GA look like?

pygad.GA Class

The first module available in PyGAD is named `pygad` and contains a class named `GA` for building the genetic algorithm. The constructor, methods, function, and attributes within the class are discussed in this section.

__init__()

For creating an instance of the `pygad.GA` class, the constructor accepts several parameters that allow the user to customize the genetic algorithm to different types of applications.

Task 1: Using PyGad for a simple optimization

We want to fit a very simple function:

$y = 0.5 + 0.3*x + 0.4*x^2$

To do this we want to use a genetic algorithm.

As a first approximation we will try to use the following function:
$f = w0 + w1*x + w2*x^2$

We thus need to optimize the weights w0,w1,w2

In *task1/optimize.py* you find the code we need to launch.

NOTE:
We need to run by using some packages I have in one of my environments.
Please connect to a remote machine by using **compile**, copy task1 in the scratch (/scratch/atqcXX/)
Move to /scratch/atqcXX/task1 and work from there.
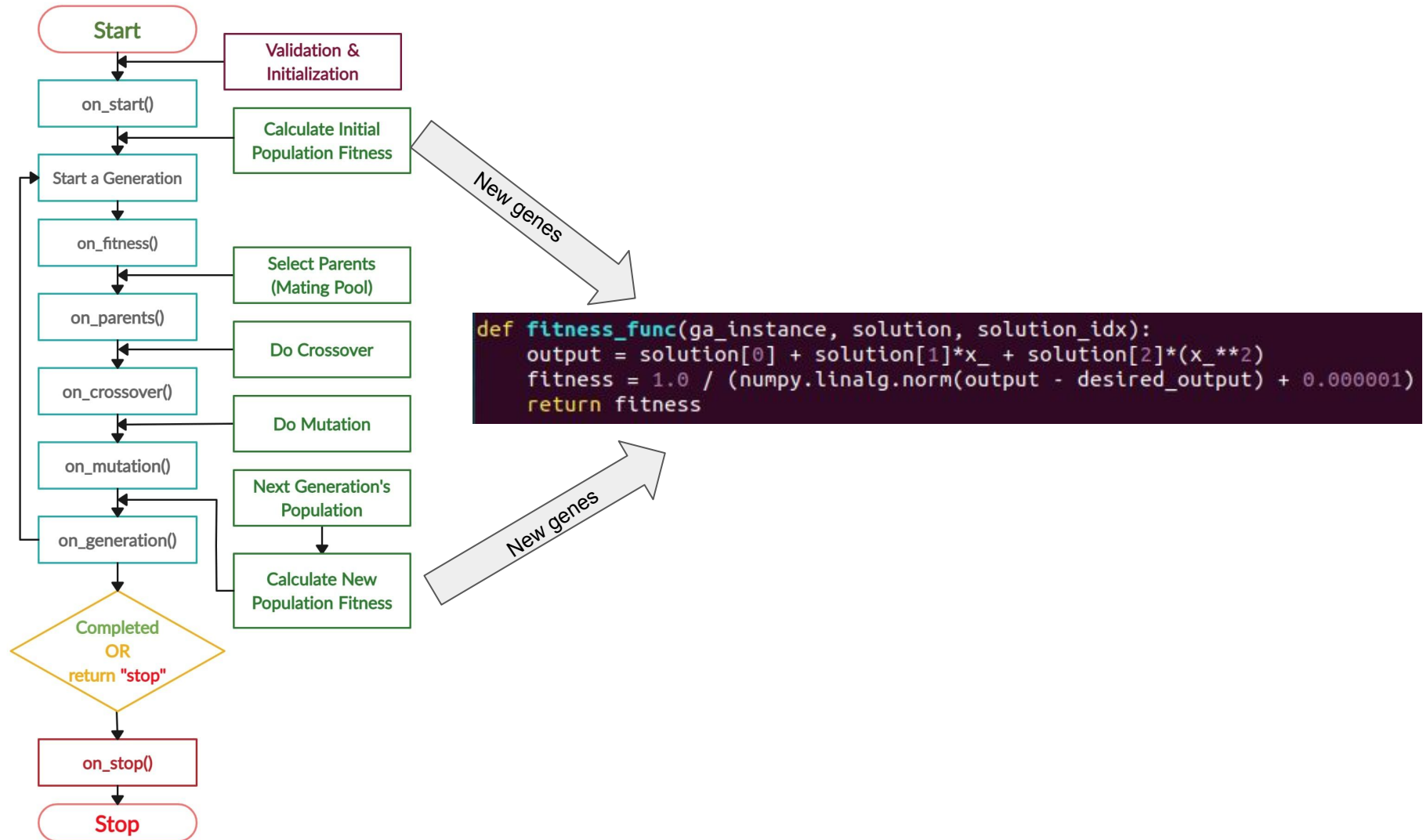Please launch the GA by using the command **python** (I changed in your .bashrc where this command is pointing towards)

To do so, you'll need to set up a fitness function in **fitness_func(ga_instance, solution, solution_idx):**
solution = list of gene values of the solution_idx individual

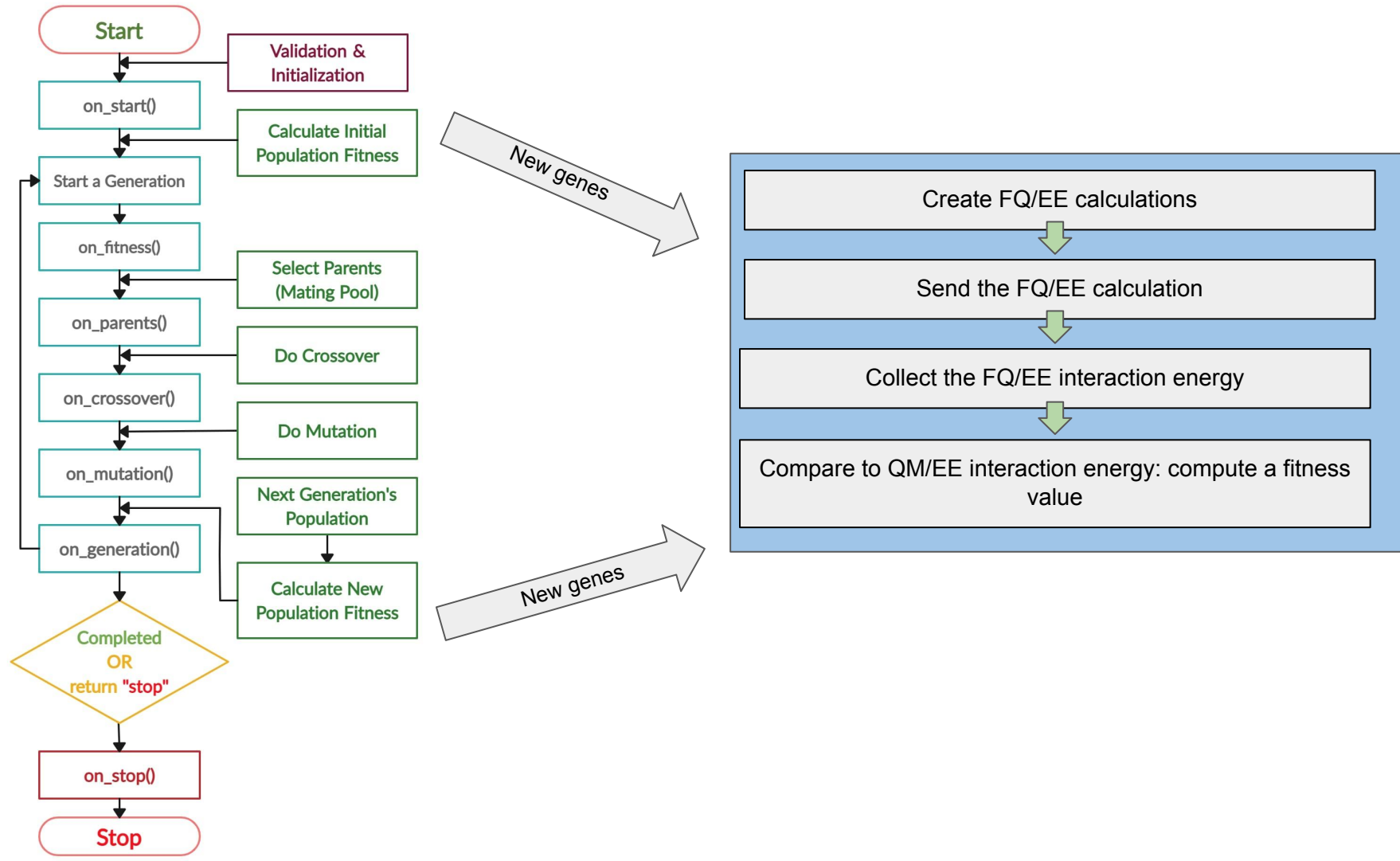**EXTRA:**
Try to see what happens to the convergence by changing the hyperparameters

How does a cycle of the GA look like?


```python
def fitness_func(ga_instance, solution, solution_idx):
    output = solution[0] + solution[1]*x_ + solution[2]*(x_**2)
    fitness = 1.0 / (numpy.linalg.norm(output - desired_output) + 0.000001)
    return fitness
```

How does a cycle of the GA look like?

Task 2: Optimizing FQ for water

We now want to use the machinery of PyGAD to handle the GA and the Generation exploitation we developed, to optimize the FQ forcefield for water.

In **task_2** you find all you need.

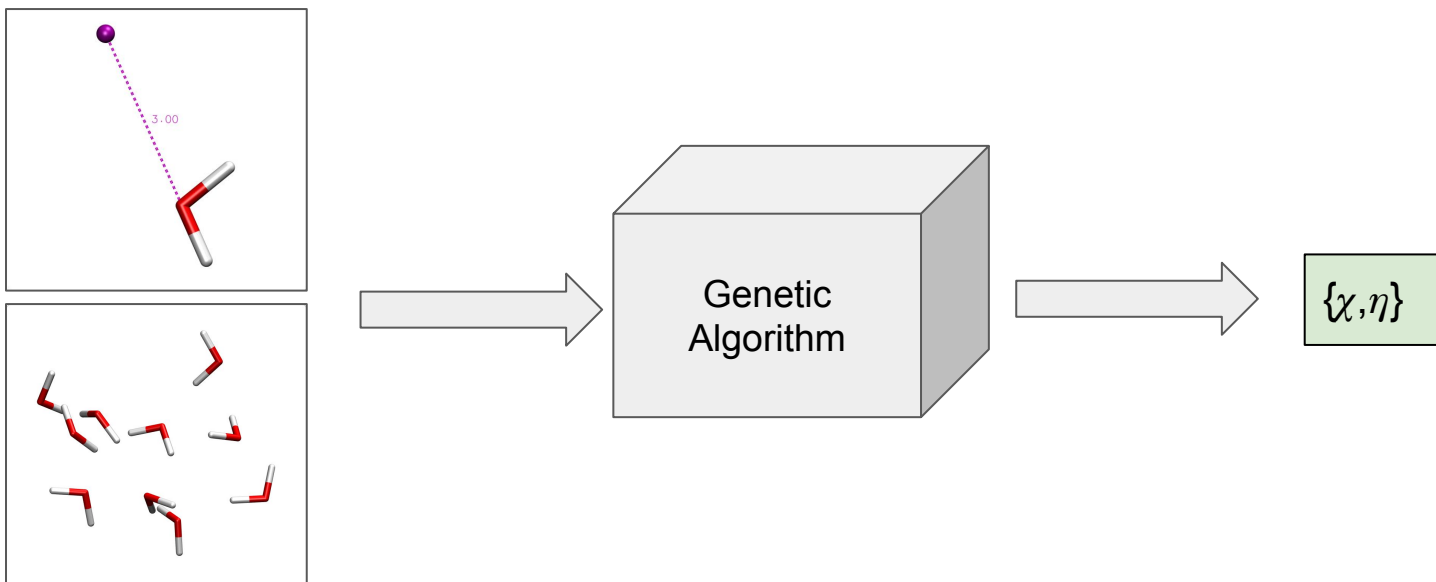**NOTE:**
In **run_ga.py** change the path towards your nanoFQ
This time we won't enter interactively in the machine, but we will send the calculations remotely.
To do this, you can use **qsub send.sh** in **send_scripts/**

To see if the calculation is going you can press **qs** (alias for qstat -n -1 -u ${USER})

Then log in into the machine by using **ssh diamondXX** (with your correct machine number)
Go to **/scratch/atqcXX/task2/** and check the **GA_logfile.txt**

Finally here we are!

In *task_3* you find all you need.

NOTE:
Again we will send the calculations remotely.
To do this, you can use *qsub send.sh*. in run_ga change the path towards your nanoFQ

Before sending the calculations set in ga_setup.txt:
-The population dimension;
-The number of generations;
-The range of electronegativities (select one to be zero and some ranges for the others);
-The range of the chemical hardnesses;
-mutation min and max val;

NOTE:
Keep in mind that for a single individual, using 40 % of the training set, the code will run for 30 seconds, and next time we need to have all the results
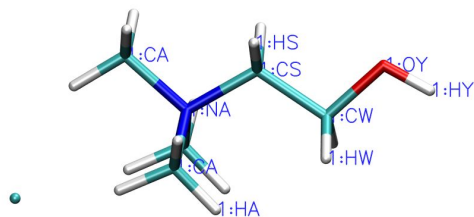
Possible hyperparameters you can set in ga_setup.txt

Possible hyperparameters:
1) generations
2) population_dimension (in PyGAD it is called sol_per_pop)
3) elitism_percentage
4) mutation_num_genes;
5) mutation_probability;
6) num_parents_mating;
7) mutation_min_val;
8) mutation_max_val;
9) The values of the chis and etas;

REMEMBER to set one chi to 0