# Week 3 Report

The remote repository has been updated. At present, there is a ‘Loop files’ Folder that includes: a) problem loop files b) algorithms loop files.

## For problems loop files

First, chose problem you want for random search. In this case, CH\_problem

Graphical user interface, application

Description automatically generated

Secondly, for all problems files, you need to define two parameters: Bound and number of data points. In this case, upper bound and lower bound random generate by numpy. There are many ways to define your bound. For number of data points, select number. In this case, it is 300.

Graphical user interface, text, application, email

Description automatically generated

Finally, each time running this file, you will get result shown as below (they are automatic stored in the result folder):

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

## Algorithms loop files:

Each algorism loop file is trying to fix the parameters in terms of different problems:

Find files here:

Graphical user interface, text, application

Description automatically generated

For example: I would like to compare how different number of generation times affect the algorithm performance in terms of NSGA-II:

Graphical user interface, text, application, email

Description automatically generated

By select NSGA-II\_generation.py file.

Note: Each algorithm file has multiple loop function inside: you need define some parameter lists in prior:

Problem list is availabel problems can be used here. They are all multi\_problems

Selecting problem you would like to have and pass they into select\_problem\_list

Graphical user interface, text, application

Description automatically generated

And then pass the generation number list you would like to compare below:

Graphical user interface, text, application, email

Description automatically generated

The file working flow:

1. Selecting problem
2. For selected problem, loop different generation number in NSGA-II (in this case), it’s NSGA-II parameter, because example is NSGA-II\_generation.py
3. For each generation number, calculate gradient distance between pf and object vector generated by algorithm. The smaller the distance is, the better result is. Alternatively, we can use other performance indicators such as IGD, hypervolume etc.

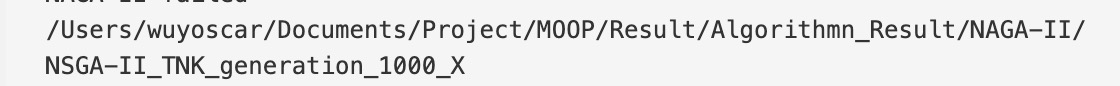
Result show below:

Graphical user interface, text, application, email

Description automatically generated

In this case, we tested NSGA-II with TNK problem. If the generation number is 300, the GD is 0.003575, which is best. We looped generation number list 300, 200, 100, 500, 1000, 80, 900, 800, 20, 10.

Further, result will be stored under result folder like below:



Plus, we can loop problems and parameter (i.e generation) at the same time. Result will be like this. In this case, two problems with generation loop list.

Text

Description automatically generated