Evolutionary Submodular Optimisation Competition – GECCO 2025 https://cs.adelaide.edu.au/~optlog/CompetitionESO2025.php

This is a step-by-step instruction guide for using the Python version of IOHexperimenter for the Evolutionary Submodular Optimisation competition.

You can use **pip** to install the latest version of IOHexperimenter:

```
pip install ioh
```

You can verify the installation by using:

```
pip show ioh
```

You will see some information about the tool, similar to the example shown below:

```
Name: ioh
Version: 0.3.18
Summary: The experimenter for Iterative Optimization Heuristics
Home-page: https://iohprofiler.github.io/IOHexperimenter
Author: Jacob de Nobel, Furong Ye, Diederick Vermetten, Hao Wang, Carola Doerr and Thomas Bäck
Author-email: iohprofiler@liacs.leidenuniv.nl
License: BSD
Location: /Users/saba/PycharmProjects/competition/venv/lib/python3.9/site-packages
Requires: numpy
```

Once *IOHexperimenter* is installed, you can begin using it in your code. Start by importing the **ioh** module.

```
import ioh
```

You can retrieve submodular graph problems by calling the **get_problem** function with their problem ID:

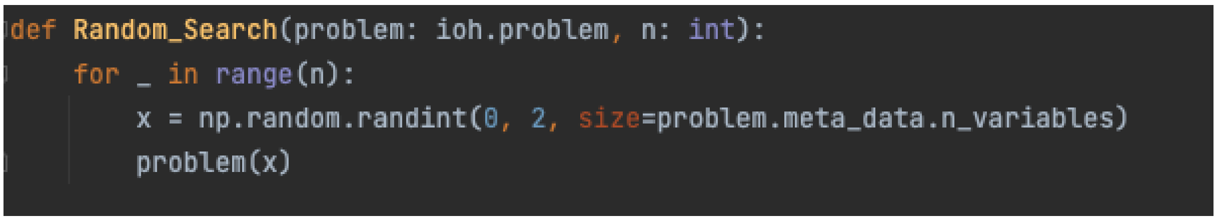
```
f = ioh.get_problem("problem ID", problem_class=ioh.ProblemClass.GRAPH)
```

Here are the problem IDs associated with the available submodular functions:

Maximum Cut Problem: 2000 - 2004
Maximum Coverage Problem: 2100 - 2127

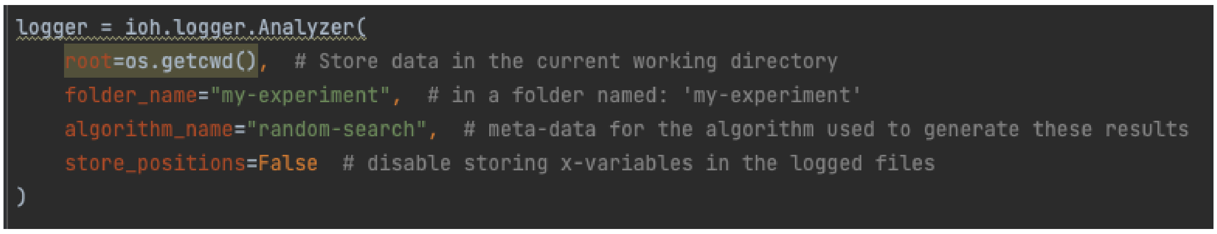
Maximum Influence Problem: 2200 - 2224

You can use these problems directly with your algorithms. As an example, below is a basic implementation of a random search:



```python
def Random_Search(problem: ioh.problem, n: int):
    for _ in range(n):
        x = np.random.randint(0, 2, size=problem.meta_data.n_variables)
        problem(x)
```

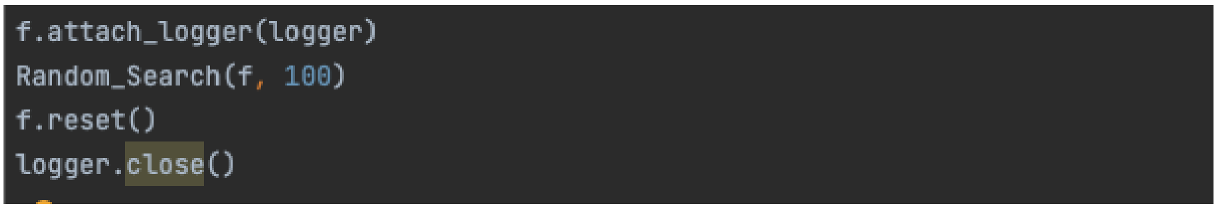
To record the results, you need to initialize a logger and attach it to the problem. Here's how you can set up a logger:



```python
logger = ioh.logger.Analyzer(
    root=os.getcwd(),  # Store data in the current working directory
    folder_name="my-experiment",  # in a folder named: 'my-experiment'
    algorithm_name="random-search",  # meta-data for the algorithm used to generate these results
    store_positions=False  # disable storing x-variables in the logged files
)
```

The logger will automatically record the data for iterations where an improvement occurs.

Finally, to run the algorithm with the problem "f", attach the logger to the problem, call the algorithm function with the problem passed as an argument, and then reset both the problem and the logger:



```python
f.attach_logger(logger)
Random_Search(f, 100)
f.reset()
logger.close()
```

After running the code, a new folder will be created in the same repository, containing a .json file and a .dat file that display the results obtained. You can upload these files to the IOHanalyzer site to get a visualized analysis of the experimental data.

You can find the full code below. This code gets one of the maximum cut problems in IOHprofiler (Id=2000), runs a random search algorithm on the problem, and logs any evaluations that have improvement throughout the optimization process:

```
import ioh
import numpy as np
import os

# help(ioh.problem)
f = ioh.get_problem(2000, problem_class=ioh.ProblemClass.GRAPH)


def Random_Search(problem: ioh.problem, n: int):
    for _ in range(n):
        x = np.random.randint(0, 2, size=problem.meta_data.n_variables)
        problem(x)



logger = ioh.logger.Analyzer(
    root=os.getcwd(),  # Store data in the current working directory
    folder_name="my-experiment",  # in a folder named: 'my-experiment'
    algorithm_name="random-search",  # metadata for the algorithm used to
generate these results
    store_positions=False  # disable storing x-variables in the logged
files
)

f.attach_logger(logger)
Random_Search(f, 100)
f.reset()
logger.close()
```

For more detailed instructions, refer to the following links:
https://github.com/IOHprofiler
https://iohprofiler.github.io

References:

1. Frank Neumann, Aneta Neumann, Chiao Qian, Viet Anh Do, Jacob de Nobel, Diederick Vermetten, Saba Sadeghi Ahouei, Furong Ye, Hao Wang, Thomas Baeck. Benchmarking algorithms for submodular optimization problems using IOHProfiler. *2023 IEEE Congress on Evolutionary Computation (CEC)*. IEEE (2023). https://arxiv.org/abs/2302.01464
2. Jacob de Nobel, Furong Ye, Diederick Vermetten, Hao Wang, Carola Doerr, Thomas Baeck. IOHexperimenter: Benchmarking platform for iterative optimization heuristics. *Evolutionary Computation* 32.3 (2024): 205-210. https://arxiv.org/abs/2111.04077