Assignment 4

Implement the following three algorithms described below. Each algorithm is worth up to five points. Solutions must be submitted by 25.5.2025. Use the link on e-ucilnica to submit your work. The report must be in .pdf format with an additional .py file for code submission.

Continuous optimization

This assignment is an introduction to continuous optimization using functions available in *opfunu*[3] Python package. This is also a warm-up for Assignment 5. By completing this, you should have a basis for starting the next assignment.

The assignment consists of finding values at specific points in the 2D space of five functions from Cec2022[1] benchmark, namely F12022, F22022, F32022, F42022 and F52022. Cec challenges are well know and an example of a Cec2022 challenge can be found here[2]. You need to implement three basic search algorithms, namely **grid search**, **random search**, and **first descent local optimization**.

Each algorithm must be implemented as a function in Python. You should submit a PDF report with the required information described later, and your source code as a separate .py file.

This section describes the algorithms you are implementing. The second section has instructions on how to run the selected functions in Python.

Grid search

Implement a grid search algorithm to evaluate the three selected 2D functions on a discrete grid of points with a grid size of 1, within the specified default bounds for each function. The point (0.5, 0.5) should **always** be included in the search, meaning you should align the grid appropriately. See the appendix or the provided Python code to see how to find the default bounds for each function.

For each of the selected functions, the report should include:

a) Number of points tested

b) Coordinates and objective values for the minimum and maximum found

Random search

Implement a random search function that searches the 2D space uniformly randomly within the specified bounds for each function.

For each of the selected functions, the report should include:

a) Mean objective value found over 100.000 calls

b) Coordinates and objective value for the minimum found

Local search

Implement a local search using **first descent**, which means that you move to the next solution as soon as the first neighbor you find is better than the current solution instead of checking all the neighbors and moving to the best one. Let the algorithm run for a maximum of 1000 iterations with a neighborhood size of 100. Define a neighbor of a solution (x,y) as (x±rand(0.1), y±rand(0.1)), where rand(0.1) returns a uniformly random number from 0 to 0.1. The initial solution, from which you start the search, should be generated uniformly randomly inside the bounds of the function. The algorithm should stop after 1000 iterations or if it gets stuck in a local optimum. Meaning that none of the 100 generated neighbors are a better solution.

For each of the selected functions, run the algorithm 10 times and report:

a) Best coordinates and objective value found over the 10 runs

b) Mean objective value found over the 10 runs

c) For each run, report the local minimum found, the number of iterations before reaching the local minimum, and the number of calls to the objective function.

Appendix

Python example

```python
from opfunu.cec_based import cec2022

# Choose the first five benchmark functions
functions = [cec2022.F12022, cec2022.F22022, cec2022.F32022, cec2022.F42022, cec2022.F52022]

for f in functions:
    try:
        # Initialize the 2D function
        func = f(ndim=2)

        # Find the recommended bounds and evaluate function at point [0, 0]
        lower = func.lb
        upper = func.ub
        x = [0, 0]
        # Evaluate the function
        fitness = func.evaluate(x)

        # Print and store results
        print(f"{f.__name__}: fitness = {fitness:.3f}")
    except Exception as e:
        print(f"Error: {f.__name__}: {e}")
```

References

[1] Opfunu Cec2022 based module. `https://opfunu.readthedocs.io/en/latest/pages/cec_based.html#module-opfunu.cec_based.cec2022`. Accessed: 2025-05-05.

[2] Wenjian Luo, Xin Lin, Changhe Li, Shengxiang Yang, and Yuhui Shi. Benchmark functions for cec 2022 competition on seeking multiple optima in dynamic environments. *arXiv preprint arXiv:2201.00523*, 2022.

[3] Nguyen Van Thieu. Opfunu: an open-source python library for optimization benchmark functions. *Journal of Open Research Software*, 12(1), 2024.