Assignment 5

For this assignment form groups of two. This assignment is worth 40 points. Solutions must be submitted by 26.5.2024. Use the link on e-ucilnica to turn in your work. The **report** must be in .pdf format. You should also submit **source code** of your implementation and **results** in .txt format.

Continuous optimization of pymoo DF functions

The aim of this assignment is to find the best result for 14 optimization (minimization) functions that are available in *pymoo.problems.dynamic.df* named from DF1 to DF14. The functions are meant for pareto optimization and the functions change trough time. For all the functions we will set the **time = 50**, so that the functions will remain static. Also since this is pareto optimization each problem has 2 or 3 objective functions. To convert it to one value you need to **minimize the sum of all the objective functions** for each of the 14 problems.

Instructions

To test the results use all 14 functions with 75 dimensions with bounds in default ranges. Note that some functions use different bounds for the first two dimensions. You can see the lower and upper bounds of each function in variables *xl* and *xu*. Check appendix and supplementary code for example. Use time = 50.

As a team you need to implement at two or more optimization programs. One of the optimization programs must be local search (best descent local search, tabu search, guided local search, variable neighborhood search, simulated annealing, etc...) and the second one can be any optimization approach of your choice (genetic algorithm, differential evolution, whale algorithm, ant colony optimization, etc...). The second approach can also be local search.

You need to implement the approaches yourself and not use already build optimizers in *pymoo* or other python modules. There is no time limit on how long you can let the algorithms running.

Reporting

Write your results into the Google Spreadsheet available here. Each group is encouraged to fill their results as soon as it has them available so that peers can see what are currently the best obtained results. The groups should be written in spreadsheet before 19.5.2024.

Write a report with your results and description of used approaches. Report should include results for each tested method separately and a short description of each method (1-2 pages for each method). Submit your code and report on e-ucilnica.

You should also submit coordinates of the found minimums for each method in a .txt file. The file should contain 14 lines, each line representing its own function. The values of coordinates should be separated by tabulator. See example on e-ucilnica.

Grading

Final grade will be based on quality of results, quality of report, oral presentation, number of methods tested and code quality.

Appendix

Python example

```python
import numpy as np
import matplotlib.pyplot as plt

import pymoo.problems.dynamic.df as dfs

n = 75
time = 50
problems = [dfs.DF1(time = time, n_var = n),
            dfs.DF2(time = time, n_var = n),
            dfs.DF3(time = time, n_var = n),
            dfs.DF4(time = time, n_var = n),
            dfs.DF5(time = time, n_var = n),
            dfs.DF6(time = time, n_var = n),
            dfs.DF7(time = time, n_var = n),
            dfs.DF8(time = time, n_var = n),
            dfs.DF9(time = time, n_var = n),
            dfs.DF10(time = time, n_var = n),
            dfs.DF11(time = time, n_var = n),
            dfs.DF12(time = time, n_var = n),
            dfs.DF13(time = time, n_var = n),
            dfs.DF14(time = time, n_var = n)]

#Choose a sample test point (Note that this point is
    outside of bounds for some functions!)
test_point = np.array([0.5] * n)
for p in problems:
    print(p.name)
    print("Bounds from ", p.xl, " to ", p.xu, ".")
    print(p.evaluate(test_point))
    print(sum(p.evaluate(test_point)))

#Visualization
    ----------------------------------------------------


#Calculates a 2d slice of a n-dimensional space
def sum_of_paretno_functions(DF, x):
    if len(DF.xl) == 2:
        return [sum(z) for z in DF.evaluate(np.array(x
            ))]
    else:
        xm = list((DF.xl+DF.xu)/2)
        x = [[a,b, *xm[2:]] for a, b in x]
        return [sum(z) for z in DF.evaluate(np.array(x
            ))]

#Plots a 2d graph of a function (slice)
```

```
42 def plot_function(DF):
43     d = 400
44     x = np.linspace(DF.xl[0], DF.xu[0], d)
45     y = np.linspace(DF.xl[1], DF.xu[1], d)
46     X, Y = np.meshgrid(x, y)
47     points = [[x, y] for x, y in zip(X.flatten(), Y.
           flatten())]
48     Z = sum_of_paretno_functions(DF, points)
49     Z = np.array(Z).reshape((d,d))
50     print(Z)
51 
52     # Plotting the functions
53     fig, axs = plt.subplots(1, 1, figsize=(15, 15))
54     axs.contourf(X, Y, Z, levels=50, cmap='viridis')
55     axs.set_title(DF.name)
56     axs.set_xlabel('x')
57     axs.set_ylabel('y')
58     plt.show()
59 
60 for p in problems:
61     plot_function(p)
```