Developing Optimisers in Python
Coursework Assignment (2022-23)

Module:	Optimisation.
Module code:	CSC372-M72.
Module coordinator:	Dr Alma Rahat (a.a.m.rahat@swansea.ac.uk).
Allocated marks:	**20**% of the total module marks.
Submission deadline:	**11:00 on 2 December 2022 (Friday)**.
Submission site:	`canvas.swansea.ac.uk`

Tasks

Consider the following specification for a version of the optimisation problem from Section 6.5.2 in [1].

Problem description: Welded Beam Design

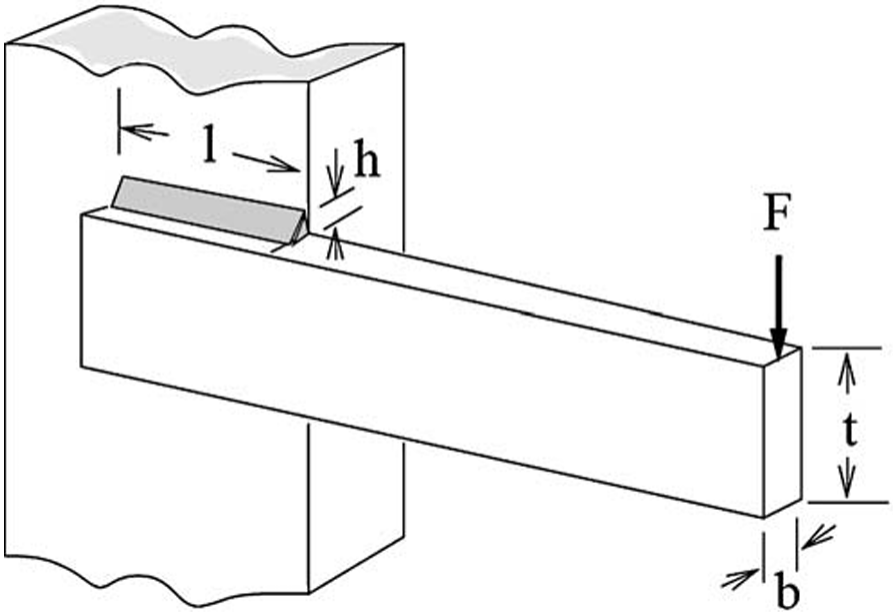

Figure 1: A sketch of a welded beam system (copied from [1]).

In a welded beam design system [2], we often aim to minimise the cost of manufacturing the beam, that has a fixed length sticking out of the welded surface. This *depends* on the four design variables: $\mathbf{x} = (x_1, x_2, x_3, x_4)^\top$, where x_1 is the height h and x_2 is the length l of the welded part, and x_3 is the thickness t and x_4 is the breadth b of the beam.

The objective function can be expressed as:

$$\min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x}) = 1.10471 x_1^2 x_2 + 0.04811 x_3 x_4 (14.0 + x_2). \tag{1}$$

The feasible space $\mathcal{X}$ is defined by the following constraints:

$$g_1(\mathbf{x}) = 13600 - \tau(\mathbf{x}) \geq 0, \tag{2}$$
$$g_2(\mathbf{x}) = 30000 - \sigma(\mathbf{x}) \geq 0, \tag{3}$$
$$g_3(\mathbf{x}) = x_4 - x_1 \geq 0, \tag{4}$$
$$g_4(\mathbf{x}) = P_c(\mathbf{x}) - 6000 \geq 0, \tag{5}$$
$$x_1, x_2 \in [0.125, 5] \subset \mathbb{R}, \text{ and} \tag{6}$$
$$x_3, x_4 \in [0.1, 10] \subset \mathbb{R}. \tag{7}$$

The first constraint ensures that the shear stress developed at the support location of the beam is smaller than the allowable shear strength of the material (13600 psi). The second constraint ensures that normal stress developed at the support location of the beam is smaller than the allowable yield strength of the material (30000 psi). The third constraint makes sure that the breadth of the beam is not smaller than the weld height from a practical standpoint. The fourth constraint makes sure that the allowable buckling load (along t direction) of the beam is more than the applied load $F = 6000$ lbs. A violation of one or more of the above four constraints will make the design unacceptable. The stress and buckling terms are non-linear to design variables and are given as follows:

$$\tau(\mathbf{x}) = \sqrt{(\tau'^2 + \tau''^2) + \frac{x_2 \tau' \tau''}{\sqrt{0.25(x_2^2 + (x_1 + x_3)^2)}}}, \tag{8}$$
$$\tau' = \frac{6000}{\sqrt{2} x_1 x_2}, \tag{9}$$
$$\tau'' = \frac{6000(14 + 0.5x_2)\sqrt{0.25(x_2^2 + (x_1 + x_3)^2)}}{2\left(0.707 x_1 x_2 \left(\frac{x_2^2}{12} + 0.25(x_1 + x_3)^2\right)\right)}, \tag{10}$$
$$\sigma(\mathbf{x}) = \frac{504000}{x_3^2 x_4}, \tag{11}$$
$$P_c(\mathbf{x}) = 64746.022(1 - 0.0282346 x_3) x_3 x_4^3. \tag{12}$$

Now, you have the following tasks.

1. Implement all the functions $f(\mathbf{x})$ and $g_i(\mathbf{x}); \forall i \in [1, 4]$ independently, where each function takes at least a Numpy array $\mathbf{x}$. Each function should have an independent counter that represents how many times a respective function has been called (or in other words evaluated).

2. Implement the Random Search (RS) method discussed in the lectures that can use the functions defined above and return an approximation of the optimum.

3. Implement the simulated annealing (SA) method that can use the functions defined above and return an approximation of the optimum solution $\mathbf{x}^*$.

4. For 21 repetitions of each of the algorithms implemented in 2 and 3, compare and comment on the performances of these optimisers. The number of evaluations for each individual function $f(\mathbf{x})$ or $g_i(\mathbf{x})$ that you are allowed at each instance of an optimisation run is 10000 at most.

In this assignment, you must use Python 3.x to develop your code in a Jupyter notebook. Please note that you are allowed to use basic and advanced Python modules, such as Numpy, Scipy, Matplotlib, etc. However, the core of the algorithmic implementations must be your own: for instance, you cannot just use a module that already implements the algorithms here. If you are in doubt, please feel free to contact the module coordinator for clarification.

Submission

You are required to submit the following on Canvas.

Code A single Jupyter notebook file (with the format `XXXXXXX.ipynb`[1]). You must ensure that all the results are pre-generated and repeatable[2], otherwise you may lose marks.

For validating your function implementations, you must use the following code in your notebook and ensure that your code is producing the same outputs (accurate up to 3 decimal places).

```
x = np.array([1.05, 3.15, 4.43, 7.87])
print("Objective function output: ", f(x))
print("First constraint function output: ", g1(x))
print("Second constraint function output: ", g2(x))
print("Third constraint function output: ", g3(x))
print("Fourth constraint function output: ", g4(x))
```

```
Objective function output:  32.6024179859
First constraint function output:  5308.848564674312
Second constraint function output:  26736.764990548952
Third constraint function output:  6.82
Fourth constraint function output:  122317448.61430933
```

Please note that you can change the function names (`f(x)`, `g1(x)`, `g2(x)`, `g3(x)` and `g4(x)`) in the above code to match your implementations. However, if you do not add the above test in your notebook, you may be penalised up to 10% of the marks.

Assessed Learning Outcomes (ALOs)

The assessed learning outcomes of this assignment are as follows.

ALO_1 Demonstrate systematic understanding of fundamental concepts of optimisation problems and algorithms.

ALO_2 Propose an appropriate method to solve an optimisation problem.

ALO_3 Develop appropriate software for solving an optimisation problem.

ALO_4 Critically evaluate performance of multiple competing optimisers, and communicate analysis.

[1] Please use your student ID to replace XXXXXXX in the name of the document.
[2] You may want to look at setting a seed using `numpy.random.seed` method.

Marking Criteria

Below we provide a table describing the mark allocation. Please note that the following allocation is indicative only, and may change following moderation.

Topic	Expectation	Indicative percentage of marks (%)
Problem implementation (Task 1, ALO_1)	Correctly implement all the functions.	20%
Implementation of RS (Task 2, ALO_3)	A flexible implementation of RS that can be used for constrained problems. It should work with a random seed such that results may be repeated exactly.	15%
Implementation of SA (Task 3, ALO_3)	A flexible implementation of SA that can be used for different problems. It should work with a random seed such that results may be repeated exactly.	30%
Using appropriate constraint handling technique (Tasks 2 and 3; ALO_2)	Use contemporary methods for RS and the optimiser of your choice.	10%
Performance evaluation (Task 4,ALO_4)	A boxplot showing performances of the competing optimisers over 21 repetitions. Statistical comparison of final results, and appropriate comments on significance.	20%
Document structure and code quality (ALO_4)	The document follows a sound logical structure, and the code is written following standard practices.	5%

You will be marked in each category based on the level at which you meet the expectations described in the table above. The level descriptions are given below.

Excellent	Very good	Good	Adequate	Unacceptable
$70-100\%$	$60-69\%$	$50-59\%$	$40-49\%$	$0-39\%$

Please note that we may hold vivas as part of the assessment process. If you are invited for a viva, you will be expected to download your notebook from Canvas, demonstrate functionalities, and answer questions about your submission.

This is an individual coursework, and please adhere to the guidelines about academic misconduct which can be found in the following link:

`myuni.swansea.ac.uk/academic-life/academic-misconduct/`

Please remember that failing to adhere to the guidelines may have severe consequences, including withdrawal from the university.

References

[1] Wenyin Gong, Zhihua Cai, and Li Zhu. An efficient multiobjective differential evolution algorithm for engineering design. *Structural and Multidisciplinary Optimization*, 38(2):137–157, 2009.

[2] Tapabrata Ray and KM Liew. A swarm metaphor for multiobjective design optimization. *Engineering optimization*, 34(2):141–153, 2002.