Developing Optimisers in Python

Tasks

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

Problem description: Welded Beam Design

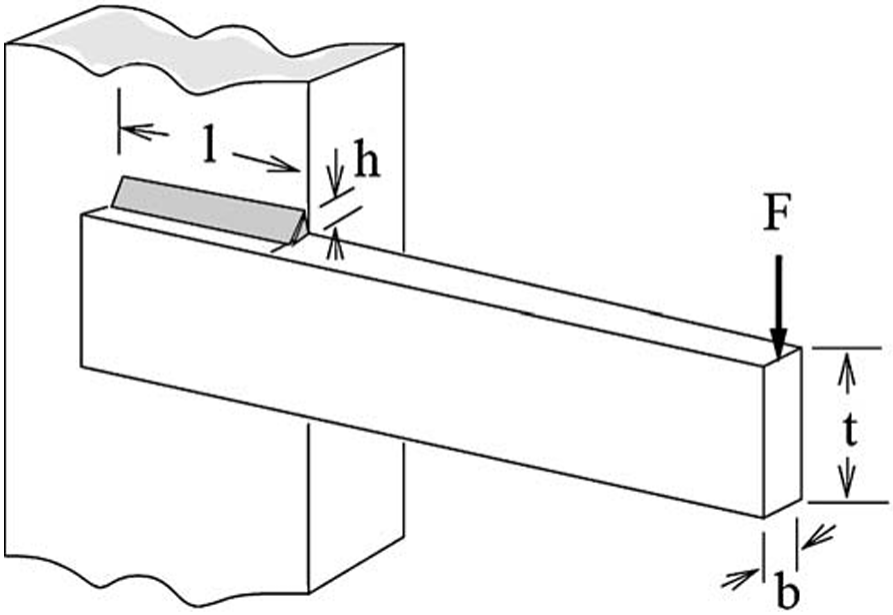

Figure 1: A sketch of a welded beam system (copied from [1]).

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.