Prof. Dr. Kathrin Flaßkamp
M.Sc. Matthias Hoffmann
Systems Modeling and Simulation
Saarland University

Optimization – Exercise 4 – WS 21/22
Evolutionary Algorithms

Exercise 4.1 – For Preparation: Optimizing the power outcome of a wind farm

We address the problem of positioning n_{wt} wind turbines within a given area such that the joint power outcome of the wind park is maximized. To this end, you are to compare gradient-based optimization with two evolutionary algorithms, namely Differential Evolution and Particle Swarm Optimization. Within the wind park, certain distances between the wind turbines have to be kept. Also, we have to model how a wind turbine reduces the wind velocity in its wind shadow. In order to keep the constrained optimization problem formulation manageable for you, some assumptions can be made:

1) The wind blows from west to east with a constant wind velocity of u_0.

2) The power outcome function of the i-th wind turbine is a function of the wind velocity at the i-th wind turbine u_i

$$P_i(u_i) = \begin{cases} 0, & u_i \in [0;3) \\ 21401 \cdot u_i^2 - 17154 \cdot u_i - 143481, & u_i \in [3;11.6] \\ 2533000, & u_i \in (11.6;\infty) \end{cases}$$

3) The wind velocity at the position (x, y) is a function of the *intensity factor* $c_{\text{intens}}(x, y, p_{\text{wt,all}})$

$$u(c_{\text{intens}}) = u(x, y, p_{\text{wt,all}}) = u_0 \cdot w^{c_{\text{intens}}}, \quad w \in (0;1), c_{\text{intens}} \in \mathbb{R}_0^+$$

4) The intensity map $c_{\text{intens}}(x, y, p_{\text{wt,all}})$ depends on the position of all wind turbines. Each wind turbine contributes to the map

$$c_{i,\text{intens}}(x, x_i, y, y_i, d) = c_{x,\text{intens}}(x, x_i) \cdot c_{y,\text{intens}}(y, y_i, d)$$
$$c_{\text{intens}}(x, y, p_{\text{wt,all}}, d) = \sum_{i=1}^{n_{\text{wt}}} c_{i,\text{intens}}(x, x_i, y, y_i, d)$$

where (x_i, y_i) is the position of the i-th wind turbine. Plug in a fixed d to obtain $c_{\text{intens}}(x, y, p_{\text{wt,all}})$.
The intensity map tries to describe the impact of all wind turbines to the wind velocity. The bigger c_{intens} is at a position (x, y), the smaller is the wind velocity at that position.

5) $c_{x,\text{intens}}, c_{y,\text{intens}}$ are modelled as

$$c_{x,\text{intens}}(x, x_i) = \sigma(x - x_i) \cdot \exp\left(\frac{-(x - x_i)^2}{2 \cdot 10^4}\right)$$
$$c_{y,\text{intens}}(y, y_i, d) = \exp\left(\frac{-(y - y_i)^2}{2 \cdot d^2}\right)$$

The function σ is a *differentiable switch* $\sigma(x) := \frac{1}{\pi}\left(\arctan(s_{\arctan} x) + \frac{\pi}{2}\right)$, $s_{\arctan} >> 1$.

6) The minimal distance between two wind turbines has to be greater than d_{wt}

$$\left\| \begin{pmatrix} x_i - x_j \\ y_i - y_j \end{pmatrix} \right\|^2 \geq d_{\text{wt}}^2 \qquad \forall i, j \in \{1, \ldots, n_{\text{wt}}\},\ i \neq j$$

7) The wind turbines have to be placed inside a square with side lengths l_{square}.

By plugging the functions into each other one obtains the power outcome of the i-the wind turbine as a function of all wind turbine positions $P_i(p_{\text{wt,all}})$. Summing up over all wind turbines one obtains the total outcome of the wind park.
In the provided templates, we have already implemented these assumptions and function calls with `fmincon` and with Differential Evolution.

Your tasks are the following:

a) Consider general placement problems as described above. Do you expect a single global solution?

b) Download the template consisting of the following files: `distance_constr_de.m`, `distance_constr_fmin.m`, `Task1_de.m`, `Task1_fmin` and `wind_power_cost.m`. The first two files describe the constraints in a way that they can be used in DE and `fmincon`. The `Task1_*.m` files can be executed for computing the optimal placement. `wind_power_cost.m` evaluates the cost function.

c) Read through both `Task1_*.m` files and run them multiple times. Compare their performance and take notes on it (keep them for discussion within the group). In `Task1_de.m`, you can change various parameters of the DE algorithm. Test if you can enhance the average performance. Feel free to make further modification if this enhances the performance!

d) *Optional:* Implement a Particle Swarm Optimization algorithm, similar to the DE algorithm you downloaded, with the following properties:

- Pull towards the personal historical best position
- Pull towards the current global best position
- Pull towards the historical global best position
- Pull towards random direction, `randn(...)` should be useful
- Boolean variables to activate/deactivate these pulls
- Intertia term

Make sure all the dimensions in your algorithm fit with the ones from the DE implementations, as otherwise the cost function and constraint functions might fail.

e) *Optional:* Compare the PSO algorithm with DE and gradient-based optimization.

Remark: The `unifrnd`-function generates evenly distributed values between two bounds. If it does not work for you, you either have to install the "Statistics and Machine Learning Toolbox" or you replace `unifrnd(min_val, max_val, dim_1, dim_2)` with `rand(dim_1,dim_2)*(max_val - min_val)+min_val`.

Exercise 4.2 – Discussion

Discuss in small groups the results you got. Especially, talk about the average performance, what parameters yielded the best results, and the difficulty of implementation and finding fitting hyperparameters. Be ready to present your PSO algorithm!