Tobias Friedrich, Timo Kötzing, Andrew Sutton **Summer Term 2015**

Project 4 – "Heuristic Optimization"

`https://hpi.de/friedrich/teaching/ss15/heuristic-optimization.html`

This last and final project of the course contains two independent sub-tasks.

Sudoku. Sudoku is a popular number-placement game. In our (slightly generalized) setting, the objective is to fill a 16×16 grid with digits so that each column, each row, and each of the sixteen 4×4 sub-grids contains all of the digits from 1 to 16. We provide a very specific partially completed grid, shown below. Your tasks are:

(a) Model the sudoku problem with a SAT formula or as an ILP. Feel free to use Google to help you.

(b) Download and install some SAT or ILP solver of your choice.

(c) Solve the given Sudoku and describe how you find one solution.

(d) **Bonus points** The puzzle has more than one solution. Can you find out how many?

															5
	10				11	15			3	14			8		7
		15			10	3			8	2					6
															15
										11					
	5	12			9								2	7	
	15	11				5							14	4	
								7	5	8	6				
				14	4										
				2	15				9						
										6					
14	6	13	3									15	5	16	4
			10					3							
			7	1	12	2	10	8					15		
			13					15						12	
			1					6							

Designing Wind Farms. Wind farms must be designed by positioning wind turbines optimally so that expected power production is maximized. Each wind turbine in a farm extracts energy from the wind and creates a wake of slower, more turbulent air behind it. The goal is to position the turbines in a farm in such a way to minimize wake effects and maximize expected power production.

For this subtask, you must determine a configuration for 100 wind turbines to be placed on an H-shaped field that is $100\,\mathrm{km}^2$ ($10\,\mathrm{km} \times 10\,\mathrm{km}$). Specifically, let $F \subseteq \mathbb{R}^2$ be a set of points constructed as the union of three bounding boxes (see Figure 2):

$$F_1 = \{(x, y) \colon 0 \leq x < 4000 \wedge 0 \leq y < 10000\}$$
$$F_2 = \{(x, y) \colon 6000 \leq x < 10000 \wedge 0 \leq y < 10000\}$$
$$F_3 = \{(x, y) \colon 0 \leq x < 10000 \wedge 2000 \leq y < 8000\}$$

The objective is to assign coordinates to each of the 100 wind turbines to minimize an energy cost function subject to the following two constraints.

1. *Field Constraint.* Each turbine must have a position $(x, y) \in F = F_1 \cup F_2 \cup F_3$.
2. *Security Distance Constraint.* The distance between each turbine must be at least 308 m.

A candidate solution is a 100×2 array X where $X[i][0]$ (respectively, $X[i][1]$) contains the x-coordinate (respectively, y-coordinate) of the i-th turbine for $i \in \{0, \ldots, 99\}$. We will provide an energy cost function f_{cost} in C++ that, given an X, determines the energy loss from wake effects. The function, which is to be minimized, has been derived from the 2015 Wind Farm Layout Competition (but is not exactly the same, and you should not consider our code equivalent).

To handle constraints, the provided energy function will penalize infeasible solutions by returning a very high energy cost value. In particular, if a configuration X is not feasible because either the Field Constraint or the Security Distance Constraint is violated (or both), $f_{\text{cost}}(X)$ returns `DBL_MAX`.

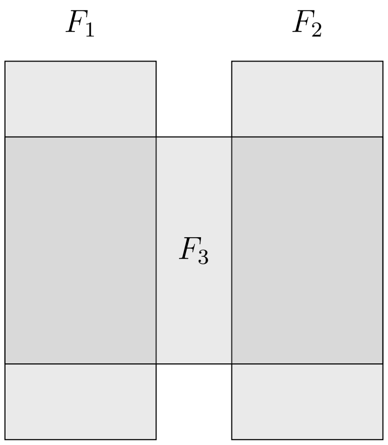
The C++ source code can be found at

`https://hpi.de/fileadmin/user_upload/fachgebiete/friedrich/teaching/SS15/Heuristic_Optimization/windfarm.tgz`

The data structure for representing a candidate solution is in the file `Matrix.h` and the main layout evaluation engine is in `Evaluator.cc`. The evaluation function is in `Evaluator::evaluate`. For example,

```
Matrix<double> layout(100,2);
Evaluator wfle;
// ... initialize layout somehow ...//
double f_cost = wfle.evaluate(layout);
```

(1) A wind farm.



(2) The feasible region for turbine placement is defined by the union of three bounding boxes F_1, F_2 and F_3.

A candidate solution will be represented in machine-readable format as follows:

```
<nTurbines>
<x0> <y0>
<x1> <y1>
...
<x99> <y99>
```

where `<nTurbines>`$= 100$, `<xi>`$= X[i][0]$ is the x-coordinate of the i-th turbine and `<yi>`$= X[i][1]$ is the y-coordinate of the i-th turbine.

In the provided C++ source code, the main compiled program reads a candidate solution in machine-readable format from the standard input and writes a double value corresponding to the calculated f_{cost}. Thus, you can interface to any optimizer using this template.

Your tasks are:

(a) Determine a feasible candidate solution for the problem. Store this in the specified machine-readable format in a file called `initial.dat`. Store the fitness of this candidate solution as a single floating point value in a file called `initial.val`.

(b) Find a better layout! For this, you may use any optimization algorithm. You may download one from the internet or even write your own. You are encouraged to use one that we have discussed in lecture (such as CMA-ES, PSO), but we do not require you to do so. Store the best layout you can find in machine-readable format in a file called `optimized.dat` and the corresponding fitness value as a single floating point value in a file called `optimized.val`. You are welcome to modify any of the code to your liking (for example, writing a better constraint handling technique). Our only requirements are that the layout in `optimized.dat` is feasible and evaluates correctly to the value in `optimized.val` (using a pristine version of the evaluator).

(c) Describe the optimization approach you chose in a short report ($\approx$ one page PDF) and discuss why you chose it and anything interesting that you learned. Submit an archive of the above mentioned data files along with this report.

(d) **Bonus points** for getting a solution that is ranked within the top 10% for the class.