Site selection for complementary renewable resources
Discrete Optimization

Gómez Herrera María Andrea Liliana

September 4, 2020

We consider that the matrices in the files represent N locations in M periods of time, such that if an energy source was build in a particular location i, we observe a 1 if in the period of time j the energy source would produce enough energy and 0 otherwise. The objective is decide the location of r energy sources (with $r \leq \frac{N}{10}$) such that the number of time periods is the maximum possible.
We define the matrix $A \in \mathbf{M}_{NxM}(R)$ as

$$A_{ij} = \begin{cases} 1 & \text{if an energy source is built in the location i} \\ & \text{and produces enough energy in the period j} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

We also define

$$x_i = \begin{cases} 1 & \text{if an energy source is build in the location i} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{pmatrix} \tag{3}$$

On the other side, if y_j is the number of sources that provide enough energy in the time period j we have that

$$y_j = \sum_{i=1}^{N} A_{ij} x_i \tag{4}$$

or equivalently

$$\mathbf{y} = A^t \mathbf{x} \tag{5}$$

Finally, we want to provide enough energy in every time period given a number of sources, we define

$$z_j = \begin{cases} 1 & y_j \geq \sum_{i=1}^{N} x_i \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

Given that we want to decide the location of r energy sources such that the number of time periods over would be the maximum possible, the objective function would be

$$max \sum_{j=1}^{M} z_j \tag{7}$$

$$s.t. \sum_{i=1}^{N} x_i \leq \frac{N}{10} \tag{8}$$

Fist, we chose the Gurobi Optimizer to solve this formulation of the problem. Since every file is a matrix of potential locations with time periods and each one represents different weather conditions, we decided to work with the data given by the Small Instance File. We obtained that the maximum number of time periods is 944. The solver explored 3941 nodes and did 26952 iterations in 15.53 seconds.

For the second part of the project, we decided to use local search for find the maximum number of time periods. First, we looked for a feasible solution by iterating over the data set and stopping the iteration once that we get a solution. We say that a set of locations is a solution if at least one of the columns achieves that the sum is greater or equal than a threshold, defined by us (for choose it we tried several values).
As local search is based in move from solution to solution in the space of solutions by applying local changes, until a solution optimal is found.
Given the feasible initial solution, we moved the set of locations, so we can find more possible solutions. Finally we looked for the set of locations which has the maximum solution.
After several trials, we got that the maximum number of periods is less than the number obtained by using the solver, this maximum value depends of the value of the threshold. On the other hand, there are less iterations and the execution time is way faster than the one using the solver.

The results obtained using the Gurobi Solver were the following:

```
Academic license - for non-commercial use only
Academic license - for non-commercial use only
Gurobi Optimizer version 9.0.2 build v9.0.2rc0 (win64)
Optimize a model with 73 rows, 8832 columns and 264646 nonzeros
Model fingerprint: 0x7256cc89
Variable types: 1 continuous, 8831 integer (8831 binary)
Coefficient statistics:
  Matrix range     [1e+00, 1e+03]
  Objective range  [1e+00, 1e+00]
  Bounds range     [0e+00, 0e+00]
  RHS range        [9e+02, 1e+03]
Found heuristic solution: objective 876.0000000
Variable types: 1 continuous, 8831 integer (8831 binary)

Root relaxation: objective 9.468650e+02, 112 iterations, 0.05 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0  946.86500    0   27  876.00000  946.86500  8.09%     -    0s
H    0     0                      883.0000000  946.86500  7.23%     -    0s
H    0     0                      884.0000000  946.86500  7.11%     -    0s
H    0     0                      922.0000000  946.86500  2.70%     -    0s
H    0     0                      940.0000000  946.86500  0.73%     -    0s
H    0     0                      941.0000000  945.89000  0.52%     -    0s
     0     0  945.80500    0   40  941.00000  945.80500  0.51%     -    0s
     0     0  945.50100    0   44  941.00000  945.50100  0.48%     -    1s
H    0     0                      943.0000000  945.50100  0.27%     -    1s
     0     0  945.50100    0   44  943.00000  945.50100  0.27%     -    1s
     0     0  945.47300    0   50  943.00000  945.47300  0.26%     -    1s
H    0     0                      944.0000000  945.47300  0.16%     -    1s
     0     0  945.47050    0   52  944.00000  945.47050  0.16%     -    1s
     0     0  945.47050    0   48  944.00000  945.47050  0.16%     -    1s
     0     2  945.47050    0   48  944.00000  945.47050  0.16%     -    2s
   947   194  944.73537  131   21  944.00000  944.83067  0.09%   3.9    5s
  2134   593  944.83067   16   67  944.00000  944.83067  0.09%   3.4   10s
  3803    25  944.27835  270   31  944.00000  944.27835  0.03%   6.6   15s
```

```
Cutting planes:
  Gomory: 80
  MIR: 2
  Flow cover: 3
  RLT: 1

Explored 3941 nodes (26952 simplex iterations) in 15.53 seconds
Thread count was 8 (of 8 available processors)

Solution count 8: 944 943 941 ... 876

Optimal solution found (tolerance 1.00e-04)
Best objective 9.440000000000e+02, best bound 9.440000000000e+02, gap 0.0000%
Value of sum(x) 876.0
Value of c 874.0
Model with Small Instance file:
944.0
```