power

October 24, 2022

0.1 Power Capacity Planning

- A regional energy company is planning for its energy generation mix for the next decade. The generation mix consists of nuclear, coal, and solar energy. The energy company spans two counties, A and B.
- The projected maximum demand for the entire region beyond the current grid's capacity is 5.0×10^3 MW. We assume that there is no power lost in transmission between the counties (that is, we can satisfy this capacity requirement with any combination of power from county A and B).
- The development costs of power capacity are given in the table below in dollars per MW.

Type/County	Nuclear	Coal	Solar
A	5.2×10^6	2.5×10^6	8.0×10^6
B	4.8×10^6	2.2×10^6	8.5×10^6

- The greenhouse gas emission rates of the power sources are 1.5×10^3 tons/MW, 5.3×10^3 tons/MW, and 0.1×10^3 tons/MW for nuclear, coal, and solar respectively. There is a region-wide ceiling for the total greenhouse gas emissions at 7.2×10^6 tons.
- County B is particular about power generation and wants no nuclear power generated in county B, no more than 1.5×10^3 MW capacity generated by coal in county B, and no less than 2×10^3 MW capacity generated by solar in county B.
- How much nuclear, coal, and solar energy should we plan to develop to minimize the cost while satisfying all constraints?

- Define Types = {Nuclear, Coal, Solar} and Counties = {A, B}.
- Define parameters:
  - $c_{i,j}$: The cost of power for each $i \in$ Types and $j \in$ Counties.
  - e_i: The emissions for each $i \in$ Types.
  - $e_{\max}$: The emissions ceiling.
  - $d_{\max}$: The maximum demand.
  - $\ell_{i,j},\ u_{i,j}$: Lower and upper bounds on the added capacity for each $i \in$ Types and $j \in$ Counties (may be $\pm\infty$)

- The optimization problem is:

$$
\begin{aligned}
\min_{x} \quad & \sum_{i \in \text{Types}} \sum_{j \in \text{Counties}} c_{i,j} x_{i,j} \\
\text{s.t.} \quad & \sum_{i \in \text{Types}} \sum_{j \in \text{Counties}} x_{i,j} \geq d_{\max} \\
& \sum_{i \in \text{Types}} e_i \sum_{j \in \text{Counties}} x_{i,j} \leq e_{\max} \\
& x_{i,j} \in [\ell_{i,j}, u_{i,j}].
\end{aligned}
$$

Let's define the data in code.

[ ]:
```
types = [:Nuclear, :Coal, :Solar]
counties = [:A, :B]

dmax = 5
emax = 7.2*10^3

using NamedArrays
#Remember that the x variables will be in units of 10^3 MW
cmat = [5.2*10^3 4.8*10^3; 2.5*10^3 2.25*10^3; 8*10^3 8.5*10^3]
c = NamedArray( cmat, (types,counties), ("type","county") )

e = Dict(zip(types,[1.5,5.3,0.1]))

umat = [Inf 0; Inf 1.5; Inf Inf]
u = NamedArray( umat, (types,counties), ("type","county") )

lmat = [0 0; 0 0; 0 2]
l = NamedArray( lmat, (types,counties), ("type","county") )
```

[ ]:
```
3×2 Named Matrix{Int64}
type   county   :A   :B

:Nuclear        0    0
:Coal           0    0
:Solar          0    2
```

Now, let's define the model.

[ ]:
```
using JuMP
using HiGHS

power = Model(HiGHS.Optimizer)

@variable(power, l[i,j] <= x[i in types, j in counties] <= u[i,j])

@constraint(power, demand, sum(sum(x[i,j] for j in counties) for i in types) >=
  dmax)
```

```julia
@constraint(power, emissions, sum(e[i]*sum(x[i,j] for j in counties) for i in types) <= emax)

@objective(power, Min, sum(sum(c[i,j]*x[i,j] for j in counties) for i in types))

print(power)
```

$$
\begin{aligned}
\min\quad & 5200x_{Nuclear,A} + 4800x_{Nuclear,B} + 2500x_{Coal,A} + 2250x_{Coal,B} + 8000x_{Solar,A} + 8500x_{Solar,B}\\
\text{Subject to}\quad & x_{Nuclear,A} + x_{Coal,A} + x_{Solar,A} + x_{Nuclear,B} + x_{Coal,B} + x_{Solar,B} \geq 5.0\\
& 1.5x_{Nuclear,A} + 5.3x_{Coal,A} + 0.1x_{Solar,A} + 1.5x_{Nuclear,B} + 5.3x_{Coal,B} + 0.1x_{Solar,B} \leq 7200.0\\
& x_{Nuclear,A} \geq 0.0\\
& x_{Coal,A} \geq 0.0\\
& x_{Solar,A} \geq 0.0\\
& x_{Nuclear,B} \geq 0.0\\
& x_{Coal,B} \geq 0.0\\
& x_{Solar,B} \geq 2.0\\
& x_{Nuclear,B} \leq 0.0\\
& x_{Coal,B} \leq 1.5
\end{aligned}
$$

[ ]:
```julia
optimize!(power)
```

```
Presolving model
2 rows, 5 cols, 10 nonzeros
2 rows, 4 cols, 8 nonzeros
Presolve : Reductions: rows 2(-0); columns 4(-2); elements 8(-4)
Solving the presolved LP
Using EKK dual simplex solver - serial
  Iteration        Objective     Infeasibilities num(sum)
          0     1.7000000000e+04 Pr: 1(3) 0s
          1     2.4125000000e+04 Pr: 0(0) 0s
Solving the original LP from the solution after postsolve
Model   status      : Optimal
Simplex   iterations: 1
Objective value     :  2.4125000000e+04
HiGHS run time      :          0.00
```

[ ]:
```julia
@show objective_value(power);
@show value.(x)
```

```
objective_value(power) = 24125.0
value.(x) = 2-dimensional DenseAxisArray{Float64,2,…} with index sets:
    Dimension 1, [:Nuclear, :Coal, :Solar]
    Dimension 2, [:A, :B]
And data, a 3×2 Matrix{Float64}:
```

```
 0.0  0.0
 1.5  1.5
 0.0  2.0
```

[ ]:
```
2-dimensional DenseAxisArray{Float64,2,…} with index sets:
    Dimension 1, [:Nuclear, :Coal, :Solar]
    Dimension 2, [:A, :B]
And data, a 3×2 Matrix{Float64}:
 0.0  0.0
 1.5  1.5
 0.0  2.0
```

Let's look at how sensitive our solution is to the data.

[ ]:
```
report = lp_sensitivity_report(power)
```

[ ]:
```
SensitivityReport(Dict{ConstraintRef, Tuple{Float64, Float64}}(x[Coal,B] >= 0.0
=> (-Inf, 1.5), x[Coal,A] >= 0.0 => (-Inf, 1.5), x[Nuclear,A] >= 0.0 =>
(-1890.5, 1.5), x[Solar,B] >= 2.0 => (-1381.5192307692307, 1.5), x[Nuclear,B] <=
0.0 => (0.0, Inf), x[Coal,B] <= 1.5 => (-1.5, 1.5), x[Nuclear,B] >= 0.0 =>
(-1890.5, 0.0), demand : x[Nuclear,A] + x[Coal,A] + x[Solar,A] + x[Nuclear,B] +
x[Coal,B] + x[Solar,B] >= 5.0 => (-1.5, 1355.4528301886792), emissions : 1.5
x[Nuclear,A] + 5.3 x[Coal,A] + 0.1 x[Solar,A] + 1.5 x[Nuclear,B] + 5.3 x[Coal,B]
+ 0.1 x[Solar,B] <= 7200.0 => (-7183.9, Inf), x[Solar,A] >= 0.0 =>
(-1381.5192307692307, 1.5)…), Dict{VariableRef, Tuple{Float64,
Float64}}(x[Coal,A] => (-250.0, 2700.0), x[Nuclear,A] => (-2700.0, Inf),
x[Solar,A] => (-5500.0, Inf), x[Solar,B] => (-6000.0, Inf), x[Nuclear,B] =>
(-Inf, Inf), x[Coal,B] => (-Inf, 250.0)))
```

Let's see how much the coefficient for cost of adding capacity with nuclear for county A can change before the solution changes.

[ ]:
```
NArange = report[x[:Nuclear,:A]]
println("c[:Nuclear,:A] can stay between ", c[:Nuclear,:A]+NArange[1], " and ",␣
↪c[:Nuclear,:A]+NArange[2])
```

```
c[:Nuclear,:A] can stay between 2500.0 and Inf
```

What about for county B?

[ ]:
```
NBrange = report[x[:Nuclear,:B]]
println("c[:Nuclear,:B] can stay between ", c[:Nuclear,:B]+NBrange[1], " and ",␣
↪c[:Nuclear,:B]+NBrange[2])
```

```
c[:Nuclear,:B] can stay between -Inf and Inf
```

Remember, county B does not want any nuclear power capacity added, so the value of c does not change the solution!

Let's try looking at what happens when we change the coefficient of the emissions constraint.

[ ]:
```
Emissionsrange = report[emissions]
println("emax can stay between ", emax+Emissionsrange[1], " and ",␣
  ↪emax+Emissionsrange[2])
```

```
emax can stay between 16.100000000000364 and Inf
```

To get the dual solution, we call:

[ ]:
```
@show dual(demand)
@show dual(emissions)
y1 = dual(demand)
```

```
dual(demand) = 2500.0
dual(emissions) = 0.0
```

[ ]: 2500.0

Remember, the upper and lower bounds on x also have dual variables. These are given by:

[ ]:
```
@show dual(UpperBoundRef(x[:Nuclear,:B]))
@show dual(UpperBoundRef(x[:Coal,:B]))
@show dual(LowerBoundRef(x[:Solar,:B]))
```

```
dual(UpperBoundRef(x[:Nuclear, :B])) = 0.0
dual(UpperBoundRef(x[:Coal, :B])) = -250.0
dual(LowerBoundRef(x[:Solar, :B])) = 6000.0
```

[ ]: 6000.0