BEE 4750 Homework 4: Generating Capacity Expansion

Name: Jonathan Marcuse

ID: jrm564

> **Due Date**
>
> Friday, 10/27/23, 9:00pm

Overview

Instructions

- In Problem 1, you will formulate, solve, and analyze a standard generating capacity expansion problem.
- In Problem 2, you will add a CO_2 constraint to the capacity expansion problem and identify changes in the resulting solution.

Load Environment

The following code loads the environment and makes sure all needed packages are installed. This should be at the start of most Julia scripts.

In [ ]:
```
import Pkg
Pkg.activate(@__DIR__)
Pkg.instantiate()
```

```
 Activating project at `~/Desktop/BEE4750-2/hw04-jrmarcuse`
```

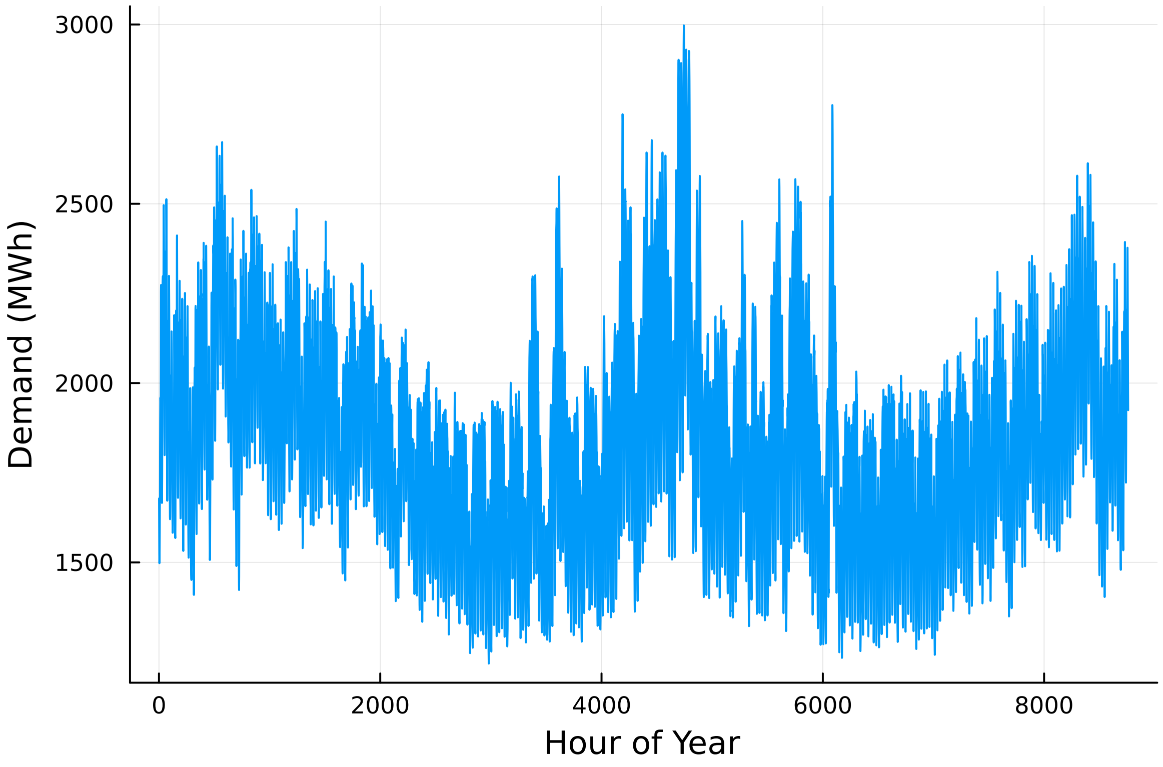
In [ ]:
```
using JuMP
using HiGHS
using DataFrames
using Plots
using Measures
using CSV
using MarkdownTables
```

Problems (Total: 100 Points)

For this problem, we will use hourly load (demand) data from 2013 in New York's Zone C (which includes Ithaca). The load data is loaded and plotted below in Figure 1.

In [ ]:
```julia
# load the data, pull Zone C, and reformat the DataFrame
NY_demand =
DataFrame(CSV.File("data/2013_hourly_load_NY.csv"))
rename!(NY_demand, :"Time Stamp" => :Date)
demand = NY_demand[:, [:Date, :C]]
rename!(demand, :C => :Demand)
demand[:, :Hour] = 1:nrow(demand)

# plot demand
plot(demand.Hour, demand.Demand, xlabel="Hour of Year",
ylabel="Demand (MWh)", label=:false)
```



- this will be MWh/hour of the year in demand in NY Zone C electrical demand

Next, we load the generator data. This data includes fixed costs ($/MWinstalled), variablecosts($/MWh generated), and CO_2 emissions intensity (tCO_2/MWh generated).

In [ ]:
```julia
gens = DataFrame(CSV.File("data/generators.csv"))
```

6×4 DataFrame

Row	Plant	FixedCost	VarCost	Emissions
	String15	Int64	Int64	Float64
1	Geothermal	450000	0	0.0
2	Coal	220000	24	1.0
3	NG CCGT	82000	30	0.43
4	NG CT	65000	40	0.55
5	Wind	91000	0	0.0
6	Solar	70000	0	0.0

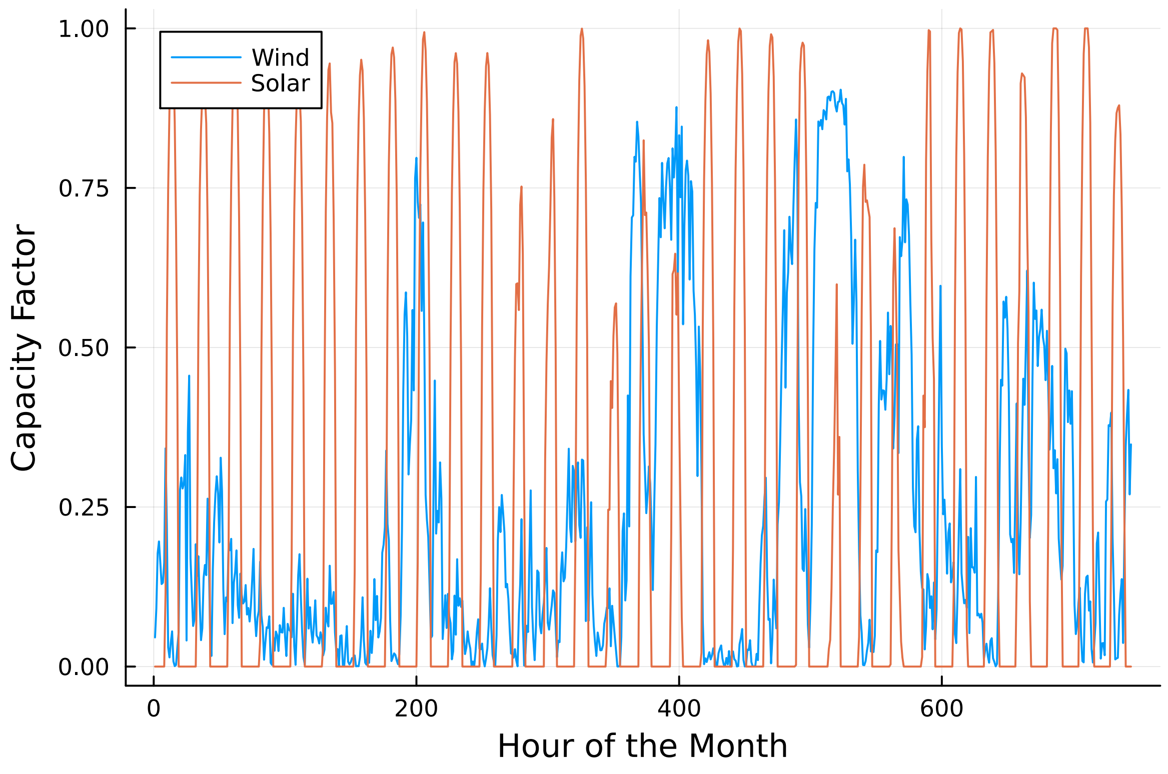
Finally, we load the hourly solar and wind capacity factors, which are plotted in Figure 2. These tell us the fraction of installed capacity which is expected to be available in a given hour for generation (typically based on the average meteorology).

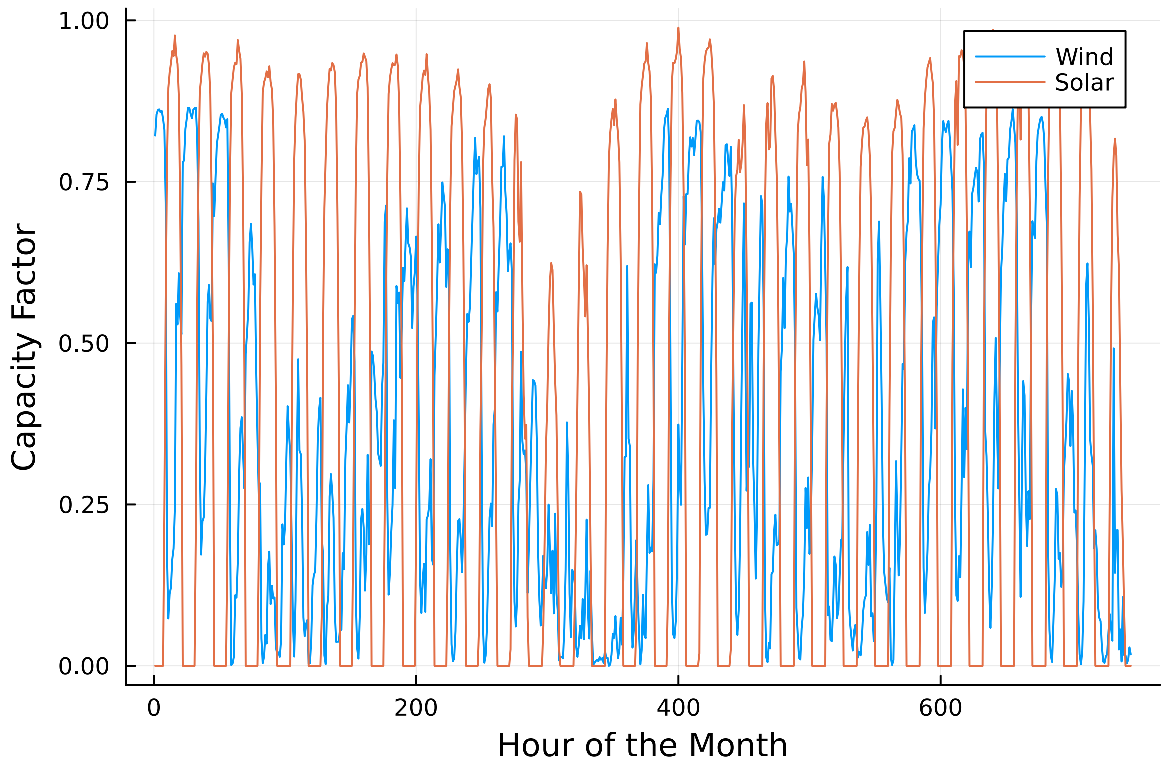
```
In [ ]: # load capacify factors into a DataFrame
cap_factor = DataFrame(CSV.File("data/wind_solar_capacity_factors.csv"))

# plot January capacity factors
p1 = plot(cap_factor.Wind[1:(24*31)], label="Wind")
plot!(cap_factor.Solar[1:(24*31)], label="Solar")
xaxis!("Hour of the Month")
yaxis!("Capacity Factor")

p2 = plot(cap_factor.Wind[4344:4344+(24*31)], label="Wind")
plot!(cap_factor.Solar[4344:4344+(24*31)], label="Solar")
xaxis!("Hour of the Month")
yaxis!("Capacity Factor")

display(p1)
display(p2)
```





So the total amount of energy available at any given moment will be the MW capacity installed multiplied by the corresponding capacity factor for solar and wind here. so it would be total MW times total hours times total capacity factor and then that would have to be equal to demand and you need to meet these requirements knowing the costs of each type of energy source, both up front costs and O&M costs must be included and maybe later we will have to consider the associated emissions as well.

use capacity factor of 0.85 for geothermal.

The added cost if demand is not met of $1000/MWh

You have been asked to develop a generating capacity expansion plan for the utility in Riley, NY, which currently has no existing electrical generation infrastructure. The utility can build any of the following plant types: geothermal, coal, natural gas combined cycle gas turbine (CCGT), natural gas combustion turbine (CT), solar, and wind.

While coal, CCGT, and CT plants can generate at their full installed capacity, geothermal plants operate at maximum 85% capacity, and solar and wind available capacities vary by the hour depend on the expected meteorology. The utility will also penalize any non-served demand at a rate of $1000/MWh.

> **Significant Digits**
>
> Use `round(x; digits=n)` to report values to the appropriate precision!

> **Getting Variable Output Values**
>
> `value.(x)` will report the values of a `JuMP` variable `x`, but it will return a special container which holds other information about `x` that is useful for `JuMP`. This means that you can't use this output directly for further calculations. To just extract the values, use `value.(x).data`.

> **Suppressing Model Command Output**
>
> The output of specifying model components (variable or constraints) can be quite large for this problem because of the number of time periods. If you end a cell with an `@variable` or `@constraint` command, I *highly* recommend suppressing output by adding a semi-colon after the last command, or you might find that your notebook crashes.

Problem 1 (22 points)

Your first task is to find a capacity expansion plan which minimizes total costs of investment and operation.

Problem 1.1 (2 points)

Identify and define the decision variables for the problem. Make sure to include units.

The decision variables will be the amount of capacity of each type of energy plant to install, in MW. This will be a vector of length 6 of which each element contains the correpsonding installed capacity.

The second decision variable will be the amount of energy produced by source for each hour and it will be an array of dimensions 6 x 8760 as there are 6 sources and 8760 hours in a year. Each value will be determined by multiplying the corresponding source capacity by the corresponding source capacity factor for each hour of the year.

The last decision variable will be the amount of energy not serviced. It will be constrained to be the difference between the demand hourly variable and the total energy produced per hour, which is determined by taking the sum of the energy produced decision variable for each hour across the sources.

The capacity installed will be: energy_installed[s in 1:6] >=0

The energy generated will be: energy_produced[s in 1:6, h in 1:8760] >=0

The total energy not serviced will be: energy_not_serviced[h in 1:8760] >=0

Problem 1.2 (3 points)

Formulate the objective function. Make sure to include any needed derivations or justifications for your equation(s) and define any additional required notation beyond that introduced in Problem 1.1.

The objective function will be 1000 times the sum of the energy not serviced decision variable, plus the sum of the total energy produced by source for the year multiplied by the O and M costs per MWh, plus the sum of the energy capacity installed vector multiplied by the fixed cost per MW vector.

This multiplied altogether will give the total cost associated with the optimized development of energy.

The objective function will be:
sum(1000*energy_not_serviced)+sum(energy_produced.*gens.VarCost)+sum(energy_instal

Problem 1.3 (4 points)

Derive all relevant constraints. Make sure to include any needed justifications or derivations.

- The total amount of energy must be a positive number which will be added as a constraint when the variables are inputted
- The energy produced must be less than or equal to the energy installed muliplied by the capacity factor. This must be implemented per source per hour.
- The total demand must be equal to the energy not serviced plus the total energy produced, and this will be implemented per hour.

energy_produced[s,h] <= energy_installed[s].*CF[s,h]

demand.Demand[h]==sum(energy_produced[:,h])+energy_not_serviced[h]

Problem 1.4 (3 points)

Implement your optimization problem in `JuMP`.

In [ ]:
```
#Define the energy model
energy_model = Model(HiGHS.Optimizer)

#Decision variables will be the capacity of each type
#of energy source that must be built
    #index 1 through 6 is Geothermal, Coal,
    #NG CCGT, NG CT, Wind, Solar respectively
#capacity installed
@variable(energy_model,energy_installed[s in 1:6] >=0);
#energy generated
@variable(energy_model,energy_produced[s in 1:6, h in 1:8760] >=0);
#energy not serviced
@variable(energy_model,energy_not_serviced[h in 1:8760] >=0);

#create vector of capacity vectors
CF = zeros(6,8760);
CF[1,:].=0.85;
CF[2:4,:].=1;
CF[5,:].=cap_factor.Wind[:];
CF[6,:].=cap_factor.Solar[:];

#The objective function will be the sum of the build,
#O&M and Demand penalty costs,
#and the goal will be to minimize this cost
@objective(energy_model,Min,sum(1000*energy_not_serviced)
+sum(energy_produced.*gens.VarCost)
+sum(energy_installed.*gens.FixedCost));

#The amount of energy produced must be equal
#to the energy installed times the CF
@constraint(energy_model,capacity[s in 1:6, h in 1:8760],
energy_produced[s,h] <= energy_installed[s].*CF[s,h]);
#The demand must be equal to the total
#energy produced plus what isn't met
    #no reason to produce more energy
    #than is demanded ever
@constraint(energy_model,not_serviced[h in 1:8760],
demand.Demand[h]==sum(energy_produced[:,h])
+energy_not_serviced[h]);

optimize!(energy_model)
```

```
Running HiGHS 1.5.3 [date: 1970-01-01, git hash: 45a127b78]
Copyright (c) 2023 HiGHS under MIT licence terms
Presolving model
56856 rows, 56862 cols, 153048 nonzeros
56856 rows, 56862 cols, 153048 nonzeros
Presolve : Reductions: rows 56856(-4464); columns 56862(-4464); elements 153
048(-8928)
Solving the presolved LP
Using EKK dual simplex solver - serial
  Iteration        Objective     Infeasibilities num(sum)
          0     0.0000000000e+00 Pr: 8760(4.52383e+06) 0s
      39534     6.3804929912e+08 Pr: 3402(819487); Du: 0(1.07067e-20) 5s
      41818     6.4455540055e+08 Pr: 0(0); Du: 0(3.41061e-13) 8s
Solving the original LP from the solution after postsolve
Model   status      : Optimal
Simplex   iterations: 41818
Objective value     :  6.4455540055e+08
HiGHS run time      :          8.22
```

Problem 1.5 (5 points)

Find the optimal solution. How much should the utility build of each type of generating plant? What will the total cost be? How much energy will be non-served?

In [ ]:
```
#Get the capacity values for printing
e = value.(energy_installed);
e = round.(e, digits = 2);
println("The total amount the utility
should build of geothermal is $(e[1])MW,
of coal is $(e[2])MW, of NGCCGT is $(e[3])MW,
of NGCT is $(e[4])MW, of Wind is $(e[5])MW,
and of Solar is $(e[6])MW")

#Get the price objective for printing
p = objective_value(energy_model);
p = round.(p,digits=2);
println("The total cost will be \$$(p)")

#Get the unserviced energy total for printing
u = value.(energy_not_serviced);
u = round.(sum(u),digits=2);
println("The total energy
non-served per year will be $u MWh")
```

```
The total amount the utility should build of geothermal is 0.0MW, of coal is
0.0MW, of NGCCGT is 1644.26MW, of NGCT is 687.09MW, of Wind is 534.51MW, and
of Solar is 1944.55MW
The total cost will be $6.4455540055e8
The total energy non-served per year will be 5966.05 MWh
```

Problem 1.6 (5 points)

What fraction of annual generation does each plant type produce? How does this compare to the breakdown of built capacity that you found in Problem 1.5? Do these results make sense given the generator data?

```
In [ ]: #Get the total amount of energy produced by source
        PES = sum(value.(energy_produced),dims=2);
        Tot = sum(PES); #grand total energy produced
        println("The total energy fraction of geothermal
         is $((PES[1]/Tot)*100)%, of coal is $((PES[2]/Tot)*100)%,
          of NGCCGT is $((PES[3]/Tot)*100)%, of NGCT
          is $((PES[4]/Tot)*100)%, of Wind is $((PES[5]/Tot)*100)%,
          and of Solar is $((PES[6]/Tot)*100)%")

        #compare to capacity percentages
        et = sum(e);
        println("The total capacity fraction of geothermal is
        $((e[1]/et)*100)%, of coal is $((e[2]/et)*100)%, of NGCCGT
         is $((e[3]/et)*100)%, of NGCT is $((e[4]/et)*100)%, of Wind
          is $((e[5]/et)*100)%, and of Solar is $((e[6]/et)*100)%")
```

```
The total energy fraction of geothermal is 0.0%, of coal is 0.0%, of NGCCGT
is 52.23218098555079%, of NGCT is 2.7432644151125642%, of Wind is 9.57610818
2135112%, and of Solar is 35.44844641720154%
The total capacity fraction of geothermal is 0.0%, of coal is 0.0%, of NGCCG
T is 34.18128600264843%, of NGCT is 14.283397880845916%, of Wind is 11.11152
6876087485%, and of Solar is 40.42378924041818%
```

This breakdown is similar to the breakdown of capacities, however the fraction of NGCCGT is considerably larger, and the fraction of NGCT is considerably smaller. These results make sense given the generate data and NGCCGT and NGCT are base load and controllable sources whereas wind and solar rely just on the weather, these can be modified to meet demand when necessary.

Problem 2 (18 points)

The NY state legislature is considering enacting an annual CO_2 limit, which for the utility would limit the emissions in its footprint to 1.5 $MtCO_2$/yr.

Problem 2.1 (3 points)

What changes are needed to your linear program from Problem 1? Re-formulate the problem and report it in standard form.

A new constraint must be added to make the sum of all associated emissions have a maximum of 1.5 MtCO2/yr.

sum(energy_produced.*gens.Emissions) must be less than or equal to 1.5 MtCO2/yr.

Problem 2.2 (3 points)

Implement the new optimization problem in `JuMP`.

```
In [ ]: #Define the energy model
energy_model2 = Model(HiGHS.Optimizer)

#Decision variables will be the capacity of each type
#of energy source that must be built
    #index 1 through 6 is Geothermal, Coal,
    #NG CCGT, NG CT, Wind, Solar respectively
#capacity installed
@variable(energy_model2,energy_installed[s in 1:6] >=0);
#energy generated
@variable(energy_model2,energy_produced[s in 1:6, h in 1:8760] >=0);
#energy not serviced
@variable(energy_model2,energy_not_serviced[h in 1:8760] >=0);

#create vector of capacity vectors
CF = zeros(6,8760);
CF[1,:].=0.85;
CF[2:4,:].=1;
CF[5,:].=cap_factor.Wind[:];
CF[6,:].=cap_factor.Solar[:];

#The objective function will be the sum of the build,
#O&M and Demand penalty costs, and the goal
#will be to minimize this cost
@objective(energy_model2,Min,sum(1000*energy_not_serviced)
+sum(energy_produced.*gens.VarCost)
+sum(energy_installed.*gens.FixedCost));

#The amount of energy produced must be equal
#to the energy installed times the CF
@constraint(energy_model2,capacity[s in 1:6, h in 1:8760],
energy_produced[s,h] <= energy_installed[s].*CF[s,h]);
#The demand must be equal to the total energy produced plus what isn't met
    #no reason to produce more energy than is demanded ever
@constraint(energy_model2,not_serviced[h in 1:8760],
demand.Demand[h]==sum(energy_produced[:,h])+energy_not_serviced[h]);
#Total emissions must not exceed the required level
@constraint(energy_model2,emissions_cap,
sum(energy_produced.*gens.Emissions)<=(1.5*10^6));

optimize!(energy_model2)
```

```
Running HiGHS 1.5.3 [date: 1970-01-01, git hash: 45a127b78]
Copyright (c) 2023 HiGHS under MIT licence terms
Presolving model
56857 rows, 56862 cols, 179328 nonzeros
56857 rows, 56862 cols, 179328 nonzeros
Presolve : Reductions: rows 56857(-4464); columns 56862(-4464); elements 179
328(-8928)
Solving the presolved LP
Using EKK dual simplex solver - serial
  Iteration        Objective     Infeasibilities num(sum)
          0     0.0000000000e+00 Pr: 8760(5.18855e+06) 0s
      18168     2.9105645424e+08 Pr: 7744(1.11056e+06); Du: 0(1.88022e-07) 6
s
      20745     4.9234888319e+08 Pr: 10158(4.72925e+06); Du: 0(1.88022e-07)
11s
      23773     6.5188104370e+08 Pr: 9987(1.06556e+07); Du: 0(1.88022e-07) 1
6s
      27016     7.1719630790e+08 Pr: 9358(6.19016e+06); Du: 0(7.07015e-08) 2
1s
      28816     7.6119532781e+08 Pr: 9953(1.28028e+06); Du: 0(7.07015e-08) 2
7s
      30495     7.7236339374e+08 Pr: 1694(121409); Du: 0(1.48436e-07) 33s
      31567     7.7360023773e+08 Pr: 0(0); Du: 0(5.57066e-12) 36s
Solving the original LP from the solution after postsolve
Model   status      : Optimal
Simplex   iterations: 31567
Objective value     :  7.7360023773e+08
HiGHS run time      :         36.96
```

Problem 2.3 (5 points)

Find the optimal solution. How much should the utility build of each type of generating plant? What is different from your plan from Problem 1? Do these changes make sense?

In [ ]:
```julia
#Get the capacity values for printing
e = value.(energy_installed);
e = round.(e, digits = 2);
println("The total amount the utility should build
of geothermal is $(e[1])MW, of coal is $(e[2])MW,
of NGCCGT is $(e[3])MW, of NGCT is $(e[4])MW, of
Wind is $(e[5])MW, and of Solar is $(e[6])MW")

#Get the price objective for printing
p = objective_value(energy_model2);
p = round.(p,digits=2);
println("The total cost will be \$$(p)")

#Get the unserviced energy total for printing
u = value.(energy_not_serviced);
u = round.(sum(u),digits=2);
println("The total energy non-served per year will be $u MWh")
```

```
The total amount the utility should build of geothermal is 270.99MW, of coal
is 0.0MW, of NGCCGT is 1519.83MW, of NGCT is 452.68MW, of Wind is 2585.84MW,
and of Solar is 2118.58MW
The total cost will be $7.7360023773e8
The total energy non-served per year will be 9144.88 MWh
```

A major difference between the solution in Problem 1 is now there is a greater capacity installed of geothermal, wind and solar which makes sense in order to have more clean energy. Previously the greater reliance on natural gas was sending emissions above the newly instated cap.

Problem 2.4 (5 points)

What fraction of annual generation does each plant type produce? How does this compare to the breakdown of built capacity that you found in Problem 2.3? What are the differences between these results and your plan from Problem 1?

In [ ]:
```
#Get the total amount of energy produced by source
PES = sum(value.(energy_produced),dims=2);
Tot = sum(PES); #grand total energy produced
println("The total energy fraction of geothermal
is $((PES[1]/Tot)*100)%, of coal is $((PES[2]/Tot)*100)%,
of NGCCGT is $((PES[3]/Tot)*100)%, of NGCT is $((PES[4]/Tot)*100)%,
of Wind is $((PES[5]/Tot)*100)%, and of Solar is $((PES[6]/Tot)*100)%")

#compare to capacity percentages
et = sum(e);
println("The total capacity fraction of geothermal
is $((e[1]/et)*100)%, of coal is $((e[2]/et)*100)%, of NGCCGT
is $((e[3]/et)*100)%, of NGCT is $((e[4]/et)*100)%, of Wind
is $((e[5]/et)*100)%, and of Solar is $((e[6]/et)*100)%")
```

```
The total energy fraction of geothermal is 6.301689649200526%, of coal is 0.
0%, of NGCCGT is 20.38248850798364%, of NGCT is 0.7362336652083096%, of Wind
is 39.71447481732701%, and of Solar is 32.86511336028051%
The total capacity fraction of geothermal is 3.9003039758661586%, of coal is
0.0%, of NGCCGT is 21.87460419809094%, of NGCT is 6.515331207037502%, of Win
d is 37.21746940091423%, and of Solar is 30.492291218091168%
```

The percent reliance on each source is quite similar to the breakdown of capacities with a slight increase in the fractions for solar and wind due to the varying capacity factors and solar and the no emissions nature of them. There is a greater percent reliance on geothermal than the percent installed, likely because geothermal is a clean base load source that can run at all hours of the day.

These values differ from problem 1 as the percentage of geothermal, wind and solar is much greater and the percentages of natural gas are lower and the amount of coal remains the same at 0 in order to produce cleaner energy. Consequently, the total energy capacity that must be installed is greater than in Questio 1 due to lower capacity factors of the clean energy sources.

Problem 2.5 (2 points)

What would the value to the utility be of allowing it to emit an additional 1000 tCO_2/yr? An additional 5000?

In [ ]:
```
#Total costs saved from increasing emissions caps
e1 = round.(-1*shadow_price(emissions_cap)*1000, digits=2);
e2 = round.(-1*shadow_price(emissions_cap)*5000, digits=2);

println("From increasing the emission cap by 1000 tCO\u2082/yr
the utility will save \$$(e1) per year and by increasing the
cap by 5000 tCO\u2082/yr the utility will save \$$(e2) per year")
```

```
From increasing the emission cap by 1000 tCO₂/yr the utility will save $1838
61.32 per year and by increasing the cap by 5000 tCO₂/yr the utility will sa
ve $919306.58 per year
```

References

List any external references consulted, including classmates.