BEE 4750/5750 Homework 3

Jason Shao (jls647)

2022-10-18

Problem 1

Problem 1.1

The decision variables are the installed capacities of each generator type g, as well as the production from generator type g in period t

Notation:

$$x_g = \text{installed capapcity of generator type g}$$
$$x_1 - Geothermal, x_2 - Coal, x_3 - CCGT, x_4 - CT, x_5 - Wind, x_6 - Solar$$
$$\vec{x} \text{ is a vector of length 6 containing all of the x values}$$

$$t = 1 : 24$$
$$g = 1 : 6$$
$$y_{g,t} = \text{production of generator type g at time period t}$$
$$Y \text{ is a 6 x 24 matrix containing the production of each generator type g at each time period t, } y_{g,t}$$

Problem 1.2

$$\text{MinCost} = \text{Investment Cost} + \text{Operating Cost} + \text{Non-served demand penalty}$$

ic is a vector of length 6 containing the cost per installed MW for each generator type g oc is a vector of length 6 containing the cost per MWh for each generator type g

```
investment_cost = [457000, 268000, 85000, 62580, 92000, 92000];
op_cost = [0, 22, 35, 45, 0, 0];
#note there are only operating costs for coal, CCGT, and CT
```

$$\text{MinCost} = \sum_{g=1}^{6} ic_g * x_g + \sum_{g=1}^{6}\sum_{t=1}^{24} oc_g * y_{g,t} + 1000\sum_{t=1}^{24} nse_t$$

Problem 1.3

Constraints:

Non-negativity

$$x_g \geq 0 \text{ for g=1,...,6}$$
$$y_{g,t} \geq 0 \text{ for g=1,...,6 and t=1,...,24}$$

Cannot produce more than installed capacity allows

CF is a 6x24 matrix containing the capacity factor for generator type g in time period t, $cf_{g,t}$

$$y_{g,t} \leq cf_{g,t} * x_g \text{ for g=1,...,6 and t=1,...,24}$$

Meet demands at each hour including non-served energy

d is a vector of length 24 containing demand values at each time period t

nse is a vector of length 24 containing all of the non-served energy values from each time period t

$$\sum_{g=1}^{6} y_{g,t} + nse_t = d_t \text{ for t=1,...,24}$$

Problem 1.4

```
julia> using JuMP, HiGHS

julia> gencap=Model(HiGHS.Optimizer)
A JuMP Model
Feasibility problem with:
Variables: 0
Model mode: AUTOMATIC
CachingOptimizer state: EMPTY_OPTIMIZER
Solver name: HiGHS

julia> generators=["geothermal", "coal", "CCGT", "CT", "wind", "solar"];

julia> periods=["hour 1","hour 2","hour 3","hour 4","hour 5","hour 6","hour
7","hour 8","hour 9","hour 10","hour 11","hour 12","hour 13","hour 14","hour
15","hour 16","hour 17","hour 18","hour 19","hour 20","hour 21","hour
22","hour 23","hour 24"];

julia> G=1:length(generators)
1:6

julia> T=1:length(periods)
1:24
```

```
julia> @variable(gencap, x[G] >=0);

julia> @variable(gencap, y[G,T]>=0);

julia> @variable(gencap, nse[T]>=0);

julia> @objective(gencap, Min,
sum(investment_cost.*x)+365*sum(y*ones(24,1).*op_cost)+sum(nse)*1000*365);

julia> @constraint(gencap, load[t in T], sum(y[:,t])+nse[t]==demand[t]);

julia> #put all capacity factors in one array
       avail=ones(6,24);

julia> for i=1:4
       avail[i,:]=avail[i,:].*thermal_cf[i];
       end

julia> avail[5,:]=wind_cf;

julia> avail[6,:]=solar_cf;

julia> @constraint(gencap, availability[g in G, t in T],
y[g,t]<=avail[g,t]*x[g]);
```

Problem 1.5

```
julia> using DataFrames

julia> optimize!(gencap)
Presolving model
156 rows, 162 cols, 420 nonzeros
156 rows, 162 cols, 420 nonzeros
Presolve : Reductions: rows 156(-12); columns 162(-12); elements 420(-24)
Solving the presolved LP
Using EKK dual simplex solver - serial
  Iteration        Objective     Infeasibilities num(sum)
          0     0.0000000000e+00 Pr: 24(60321.5) 0s
        120     9.1214221224e+08 Pr: 0(0) 0s
Solving the original LP from the solution after postsolve
Model   status      : Optimal
Simplex   iterations: 120
Objective value     :  9.1214221224e+08
HiGHS run time      :          0.00

julia> objective_value(gencap)
9.12142212241888e8

julia> installed=value.(x).data;

julia> generated=(value.(y).data*ones(24,1))/1000;

julia> generated=vec(generated);

julia> results=DataFrame(
```

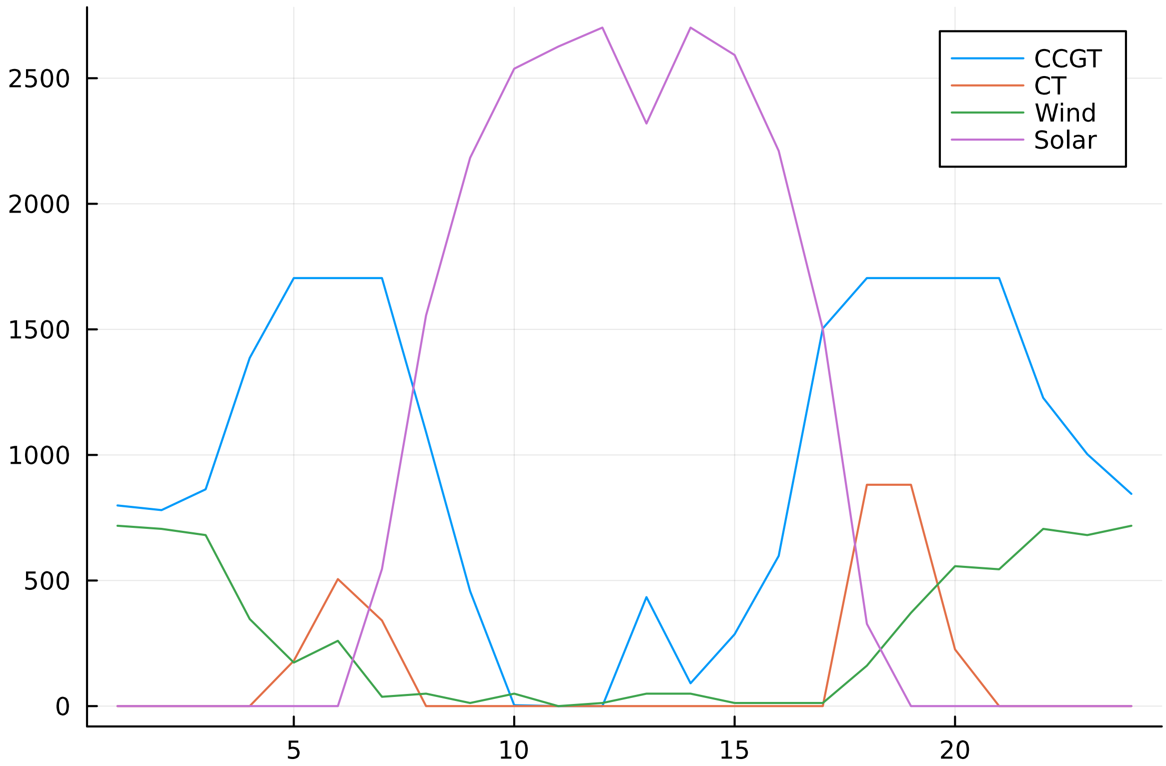
```
        "Resource" => generators,
        "Installed (MW)" => installed,
        "Generated (GWh/day)" => generated,
        )
6×3 DataFrame
 Row │ Resource    Installed (MW)  Generated (GWh/day)
     │ String      Float64         Float64
─────┼──────────────────────────────────────────────
   1 │ geothermal          0.0                 0.0
   2 │ coal                0.0                 0.0
   3 │ CCGT             1704.26               23.2987
   4 │ CT                881.327               3.01526
   5 │ wind             1238.05                6.92072
   6 │ solar            2728.91               23.8023

julia> generatedHourly=ones(1,6)*value.(y).data;

julia> generatedHourly=vec(generatedHourly);

julia> results2=DataFrame(
        "Time Period" => periods,
        "Generated (MWh/day)" =>generatedHourly,
        "Non-served (MWh/day)" =>value.(nse).data,
        "Demand" => demand
        );

julia> show(results2, allrows=true)
24×4 DataFrame
 Row │ Time Period  Generated (MWh/day)  Non-served (MWh/day)  Demand
     │ String       Float64              Float64               Int64
─────┼───────────────────────────────────────────────────────────────
   1 │ hour 1                    1517.0                   0.0    1517
   2 │ hour 2                    1486.0                   0.0    1486
   3 │ hour 3                    1544.0                   0.0    1544
   4 │ hour 4                    1733.0                   0.0    1733
   5 │ hour 5                    2058.0                   0.0    2058
   6 │ hour 6                    2470.0                   0.0    2470
   7 │ hour 7                    2628.0                   0.0    2628
   8 │ hour 8                    2696.0                   0.0    2696
   9 │ hour 9                    2653.0                   0.0    2653
  10 │ hour 10                   2591.0                   0.0    2591
  11 │ hour 11                   2626.0                   0.0    2626
  12 │ hour 12                   2714.0                   0.0    2714
  13 │ hour 13                   2803.0                   0.0    2803
  14 │ hour 14                   2842.0                   0.0    2842
  15 │ hour 15                   2891.0                   0.0    2891
  16 │ hour 16                   2821.0                   0.0    2821
  17 │ hour 17                   3017.0                   0.0    3017
```

18	hour 18	3074.0	0.0	3074
19	hour 19	2957.0	0.0	2957
20	hour 20	2487.0	0.0	2487
21	hour 21	2249.0	0.0	2249
22	hour 22	1933.0	0.0	1933
23	hour 23	1684.0	0.0	1684
24	hour 24	1563.0	0.0	1563

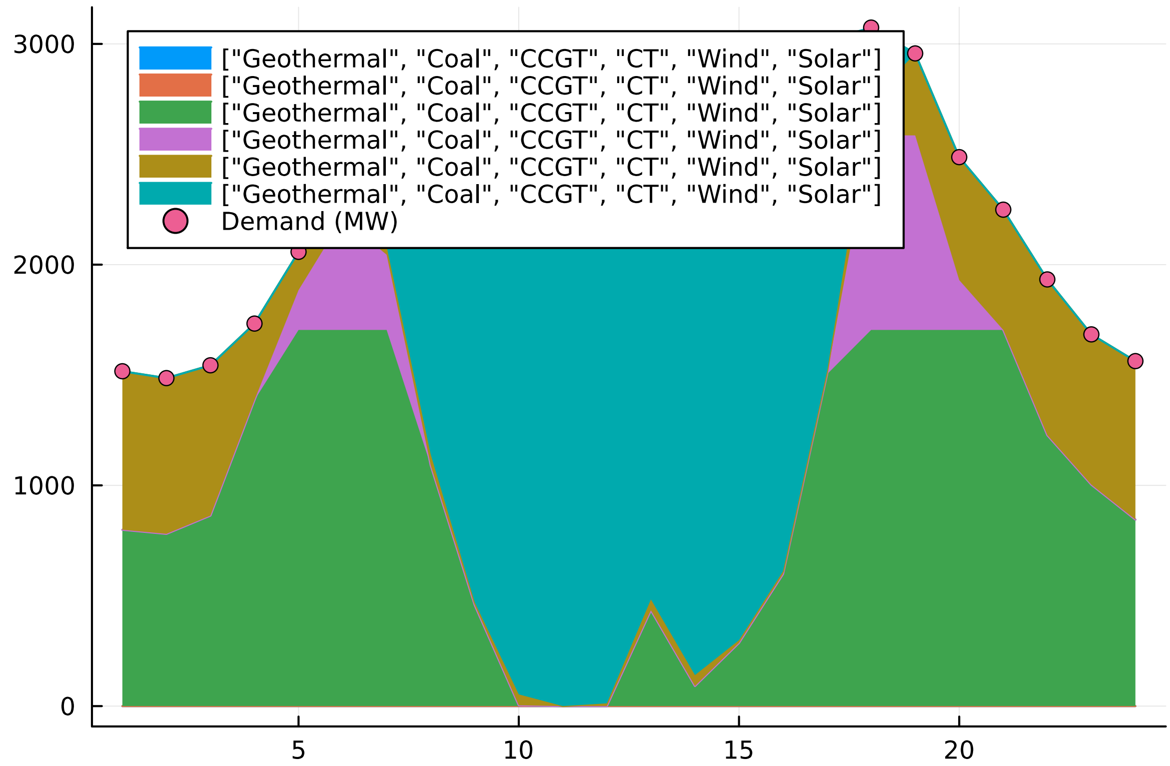
As shown in the dataframes above, in the optimal solution the utility should build 1704.26 MW of CCGT, 881.327 MW of CT, 1238.05 MW of wind, 2728.91 MW of solar, and 0 MW in both geothermal and coal. This will cost approximately $910 million for installation and operation for 1 year. In this solution, there will be no non-served energy for any hour.

Problem 1.6

```julia
julia> using Plots

julia> gen=value.(y).data;

julia> geoHour=gen[1,:];

julia> coalHour=gen[2,:];

julia> CCGTHour=gen[3,:];

julia> CTHOUR=gen[4,:];

julia> windHour=gen[5,:];

julia> solarHour=gen[6,:];

julia> plot(CCGTHour, label="CCGT",legend=:topright);

julia> plot!(CTHOUR, label="CT");

julia> plot!(windHour, label="Wind");

julia> plot!(solarHour, label="Solar")
```




```
julia> #Note there is no generation from Geothermal and Coal


       areaplot(gen', labels=["Geothermal", "Coal", "CCGT", "CT", "Wind",
"Solar"]);

julia> scatter!(demand, label="Demand (MW)", legend=:topleft)
```



Problem 2

Problem 2.1

With this limit, you could still try to minimize cost, therefore the objective function would remain the same. In order to account for the limit, a new constraint could be added to the linear program which puts a max amount of carbon emission. In order to formulate this constraint we need the values of CO2 emissions per MWh for each generator type.

$CO2_g$ is a vector of length 6 containing the CO_2 emission rate associated with each generator type g

New Constraint:

$$365 * \sum_{g=1}^{6} \sum_{t=1}^{24} y_{g,t} * CO2_g \leq 1.5MtCO_2$$

Problem 2.2

```
julia> using JuMP, HiGHS

julia> gencapCO2=Model(HiGHS.Optimizer)
A JuMP Model
```

```
Feasibility problem with:
Variables: 0
Model mode: AUTOMATIC
CachingOptimizer state: EMPTY_OPTIMIZER
Solver name: HiGHS

julia> generators=["geothermal", "coal", "CCGT", "CT", "wind", "solar"];

julia> periods=["hour 1","hour 2","hour 3","hour 4","hour 5","hour 6","hour
7","hour 8","hour 9","hour 10","hour 11","hour 12","hour 13","hour 14","hour
15","hour 16","hour 17","hour 18","hour 19","hour 20","hour 21","hour
22","hour 23","hour24"];

julia> G=1:length(generators)
1:6

julia> T=1:length(periods)
1:24

julia> @variable(gencapCO2, xCO2[G] >=0);

julia> @variable(gencapCO2, yCO2[G,T]>=0);

julia> @variable(gencapCO2, nseCO2[T]>=0);

julia> @objective(gencapCO2, Min,
sum(investment_cost.*xCO2)+365*sum(yCO2*ones(24,1).*op_cost)+sum(nseCO2)*1000*365);

julia> @constraint(gencapCO2, load[t in T],
sum(yCO2[:,t])+nseCO2[t]==demand[t]);

julia> #put all capacity factors in one array
       avail=ones(6,24);

julia> for i=1:4
       avail[i,:]=avail[i,:].*thermal_cf[i];
       end

julia> avail[5,:]=wind_cf;

julia> avail[6,:]=solar_cf;

julia> @constraint(gencapCO2, availability[g in G, t in T],
yCO2[g,t]<=avail[g,t]*xCO2[g]);

julia> #new constraint
       @constraint(gencapCO2, CO2,  365*sum(yCO2*ones(24,1).*co2_emissions) <=
1.5*10^6);
```

Problem 2.3

```
julia> using DataFrames

julia> optimize!(gencapCO2)
Presolving model
157 rows, 162 cols, 492 nonzeros
```

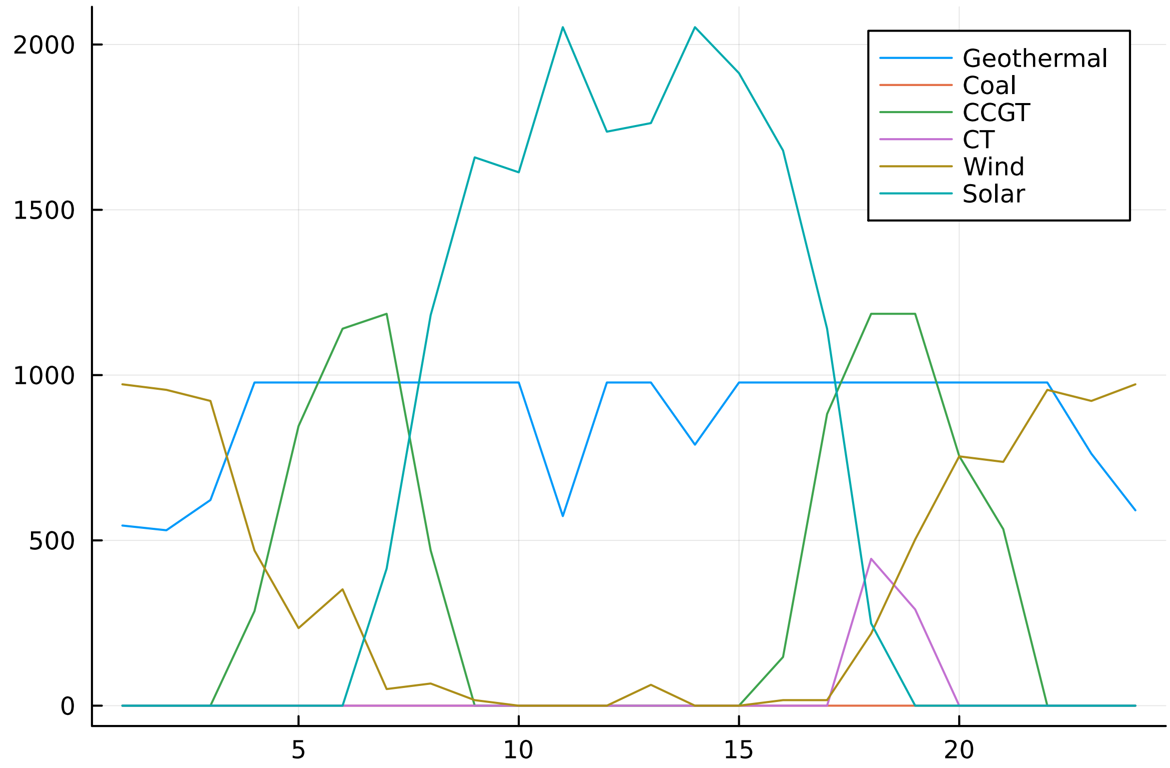
```
157 rows, 162 cols, 492 nonzeros
Presolve : Reductions: rows 157(-12); columns 162(-12); elements 492(-24)
Solving the presolved LP
Using EKK dual simplex solver - serial
  Iteration        Objective     Infeasibilities num(sum)
          0     0.0000000000e+00 Pr: 24(173966) 0s
        105     1.0659518571e+09 Pr: 0(0); Du: 0(1.45519e-11) 0s
Solving the original LP from the solution after postsolve
Model   status      : Optimal
Simplex   iterations: 105
Objective value     :  1.0659518571e+09
HiGHS run time      :          0.00

julia> objective_value(gencapCO2)
1.0659518570585868e9

julia> installedCO2=value.(xCO2).data;

julia> generatedCO2=(value.(yCO2).data*ones(24,1))/1000;

julia> generatedCO2=vec(generatedCO2);

julia> resultsCO2=DataFrame(
       "Resource" => generators,
       "Installed (MW)" => installedCO2,
       "Generated (GWh/day)" => generatedCO2,
       )
6×3 DataFrame
 Row │ Resource    Installed (MW)  Generated (GWh/day)
     │ String      Float64         Float64
─────┼──────────────────────────────────────────────────
   1 │ geothermal        1029.09             21.0336
   2 │ coal                 0.0               0.0
   3 │ CCGT              1185.43              8.6166
   4 │ CT                 444.253             0.735364
   5 │ wind              1676.07              9.19768
   6 │ solar             2073.25             17.4538

julia> generatedHourlyCO2=ones(1,6)*value.(yCO2).data;

julia> generatedHourlyCO2=vec(generatedHourlyCO2);

julia> results2CO2=DataFrame(
       "Time Period" => periods,
       "Generated (MWh/day)" =>generatedHourlyCO2,
       "Non-served (MWh/day)" =>value.(nseCO2).data,
       "Demand" => demand
       );

julia> show(results2CO2, allrows=true)
24×4 DataFrame
 Row │ Time Period  Generated (MWh/day)  Non-served (MWh/day)  Demand
     │ String       Float64              Float64               Int64
─────┼──────────────────────────────────────────────────────────────────────
```

```
 1 │ hour 1       1517.0     0.0   1517
 2 │ hour 2       1486.0     0.0   1486
 3 │ hour 3       1544.0     0.0   1544
 4 │ hour 4       1733.0     0.0   1733
 5 │ hour 5       2058.0     0.0   2058
 6 │ hour 6       2470.0     0.0   2470
 7 │ hour 7       2628.0     0.0   2628
 8 │ hour 8       2696.0     0.0   2696
 9 │ hour 9       2653.0     0.0   2653
10 │ hour 10      2591.0     0.0   2591
11 │ hour 11      2626.0     0.0   2626
12 │ hour 12      2714.0     0.0   2714
13 │ hour 13      2803.0     0.0   2803
14 │ hour 14      2842.0     0.0   2842
15 │ hour 15      2891.0     0.0   2891
16 │ hour 16      2821.0     0.0   2821
17 │ hour 17      3017.0     0.0   3017
18 │ hour 18      3074.0     0.0   3074
19 │ hour 19      2957.0     0.0   2957
20 │ hour 20      2487.0     0.0   2487
21 │ hour 21      2249.0     0.0   2249
22 │ hour 22      1933.0     0.0   1933
23 │ hour 23      1684.0     0.0   1684
24 │ hour24       1563.0     0.0   1563
```

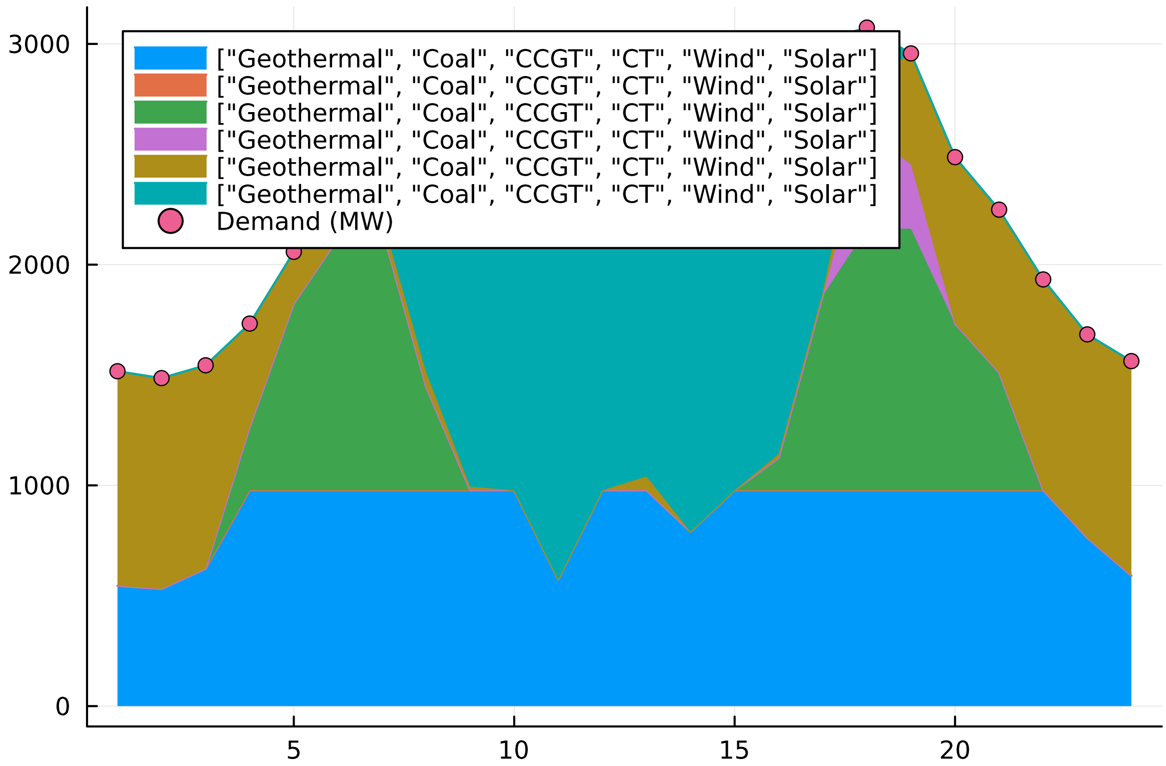
Problem 2.4

```
julia> using Plots

julia> genCO2=value.(yCO2).data;

julia> geoHourCO2=genCO2[1,:];

julia> coalHourCO2=genCO2[2,:];

julia> CCGTHourCO2=genCO2[3,:];

julia> CTHOURCO2=genCO2[4,:];

julia> windHourCO2=genCO2[5,:];

julia> solarHourCO2=genCO2[6,:];

julia> plot(geoHourCO2, label="Geothermal", legend=:topright);

julia> plot!(coalHourCO2, label="Coal");

julia> plot!(CCGTHourCO2, label="CCGT");
```

```
julia> plot!(CTHOURCO2, label="CT");

julia> plot!(windHourCO2, label="Wind");

julia> plot!(solarHourCO2, label="Solar")
```




```
julia> areaplot(genCO2', labels=["Geothermal", "Coal", "CCGT", "CT", "Wind",
"Solar"]);

julia> scatter!(demand, label="Demand (MW)", legend=:topleft)
```



Problem 2.5

```
julia> using JuMP, HiGHS

julia> shadow_price(CO2)
-130.22112610691698
```

The shadow price of CO2 is the marginal cost of increasing CO_2 emissions limit by 1 tCO_2/yr. Therefore the value to the utility of allowing it to emit an additional 1000 tCO_2/yr is $130221.

References