Smart Energy Systems
Winter 2020-2021

Optimization Project Group
Milestone 1

supervised by Ogün Yurdakul

Eric Rockstädt   Theodor Schönfisch   Isabell von Falkenhausen

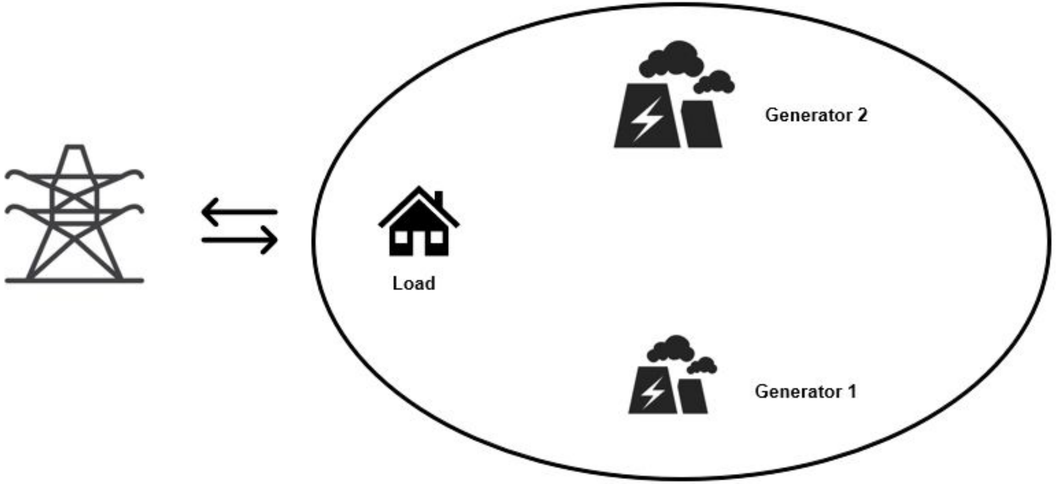
Structure

1. The problem
2. Implementation
3. Results & Interpretation

The problem: Unit Commitment

- Optimization of start-up and shut-down schedule of all power generation units to match the electricity demand and to minimize total cost.
- Difference to economic dispatch: Unit Commitment decision happens before economic dispatch
- Unit commitment decisions are typically taken one hour to one week ahead and are time-coupled problems
- Source: Quantification of the Impact of GHG Emissions on Unit Commitment in Microgrids - Yurdakul, Sivrikaya, Albayrak

The problem: Microgrid



The problem: Objective function

Objective function: minimization of costs of microgrid the operation

$$EUC: \quad \underset{\substack{u^{\mathsf{i}}_{\gamma_g}[h], p^{\mathsf{i}}_{\gamma_g}[h], \\ p^{\mathsf{n}}_{\varphi}[h]}}{\text{minimize}} \quad \sum_{\gamma_g \in \mathscr{G}_{TGR}} \left[\xi^{\dagger}_{\gamma_g} \right] + \xi_{\varphi}$$

with fuel cost of thermal generation resource gamma_g over the study period:

$$\xi^{\dagger}_{\gamma_g} = \sum_{h \in \mathscr{H}} \left[\left(\overline{\overline{c}}_{\gamma_g} (p^{\mathsf{i}}_{\gamma_g}[h])^2 + \overline{c}_{\gamma_g} (p^{\mathsf{i}}_{\gamma_g}[h]) + c_{\gamma_g} \right) u^{\mathsf{i}}_{\gamma_g}[h] (1 \ hr) \right]$$

and total net cost (i.e., total cost minus total benefit) associated with the exchange of power with the distribution company:

$$\xi_{\varphi} = \sum_{h \in \mathscr{H}} \left[\lambda[h] p^{\mathsf{n}}_{\varphi}[h] (1 \ hr) \right]$$

The problem: Constraints

Power balance

$$\sum_{\gamma_g \in \mathscr{G}} p^{\mathsf{i}}_{\gamma_g}[h] + p^{\mathsf{n}}_{\varphi}[h] = p^{\mathsf{w}}_{\delta}[h]$$

Minimum and maximum outputs of the generation resources

$$u^{\mathsf{i}}_{\gamma_g}[h][p^{\mathsf{i}}_{\gamma_g}]^m \leq p^{\mathsf{i}}_{\gamma_g}[h] \leq u^{\mathsf{i}}_{\gamma_g}[h][p^{\mathsf{i}}_{\gamma_g}]^M$$

Implementation: Pyomo basics

```
import pyomo.environ as pyo
from pyomo.opt import SolverFactory
from pyomo.core import Var, value

# create concrete pyomo model
model = pyo.ConcreteModel()

# hourly set for 24 hours, 2 generator set
model.H = pyo.RangeSet(0,23)
model.G = pyo.RangeSet(0,1)

# Variables
model.pn = pyo.Var(model.H)
model.pg = pyo.Var(model.H, model.G,within=pyo.NonNegativeReals)
model.u = pyo.Var(model.H, model.G, within=pyo.Binary)
model.y = pyo.Var(model.H, model.G, within=pyo.NonNegativeReals) # Helper Variable
```

Implementation: Objective Function

```
# Objective Function

model.OBJ = pyo.Objective(
        expr= sum(lamda*model.pn[h] for h in model.H)
            + sum(
                (c2[g]*model.y[h,g] + c1[g]*model.pg[h,g] + c[g])*model.u[h,g]
                for h in model.H for g in model.G
            )
    )
```

Implementation: Constraints

```
def loadc(model, H):
        return sum(model.pg[H,g] for g in model.G) + model.pn[H] == pl[H]
model.loadc = pyo.Constraint(model.H, rule=loadc)

def minc(model, H, G):
        return model.u[H,G]*pmin[G] <= model.pg[H,G]
model.minc = pyo.Constraint(model.H, model.G, rule=minc)

def maxc(model, H, G):
        return model.u[H,G]*pmax[G] >= model.pg[H,G]
model.maxc = pyo.Constraint(model.H,model.G, rule=maxc)

def hvc(model, H, G):
        return model.y[H,G] == model.pg[H,G]**2
model.hvc = pyo.Constraint(model.H, model.G, rule=hvc)
```

Implementation: Solve, Loop

```
opt = pyo.SolverFactory('gurobi')
opt.options['NonConvex'] = 2
results = opt.solve(model)
```

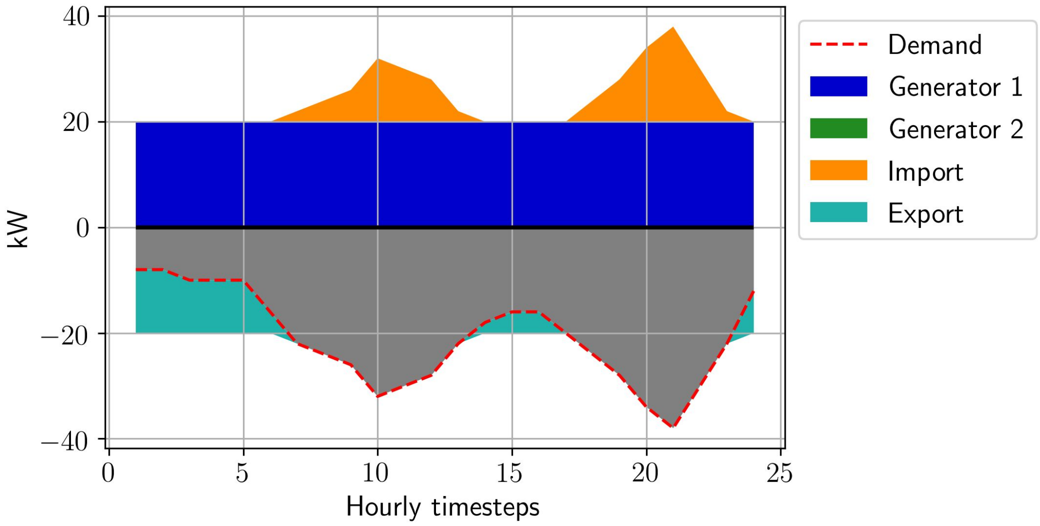
```
# Sensitivity Analysis
lmbdas = np.arange(5,80,10)*0.01

for lamda in lmbdas:
      [...]
```

Implementation: Data

```
#load for 24 hours
pl = [8,8,10,10,10,16,22,24,26,32,30,28,22,18,16,16,20,24,28,34,38,30,22,12]  [kW]

#electricity retail price
lamda = [21.97/100]  [$/kWh]

#max power of generator 1,2
pmax = [20,40]   [kW]


#quadratic, linear, absolute fuel cost parameters of generators 1,2
c2 = np.array([1.2,1.12])*0.001     [$/kW^2h]
c1 = np.array([0.128,0.532])        [$/kWh]
c  = np.array([2.12,1.28])*0.00001   [$/h]
```

Results | Static λ = 0.2197 $/kWh

Energybalance



<u>Objective Value</u>

78.23 $

<u>Marginal Costs</u>

Generator 1

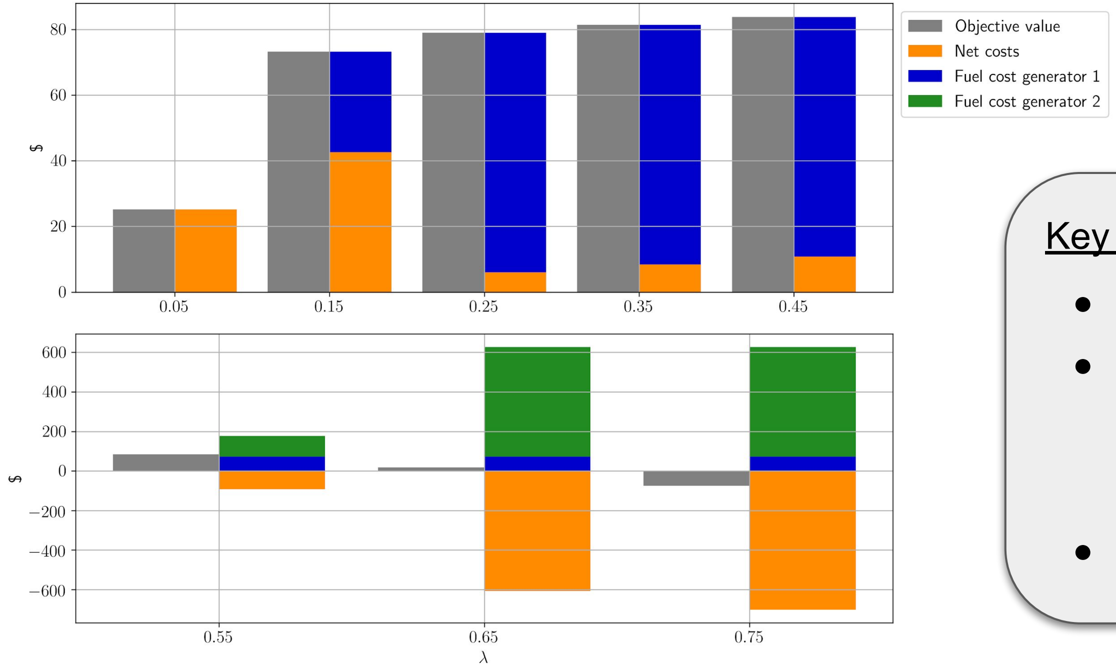
$$0.0024 \frac{\$}{(kWh)^2} * p^i_{\gamma_G}[h] + 0.128 \frac{\$}{kWh}$$

Generator 2

$$0.00224 \frac{\$}{(kWh)^2} * p^i_{\gamma_G}[h] + 0.532 \frac{\$}{kWh}$$

Results | Sensitivity analysis

Total costs

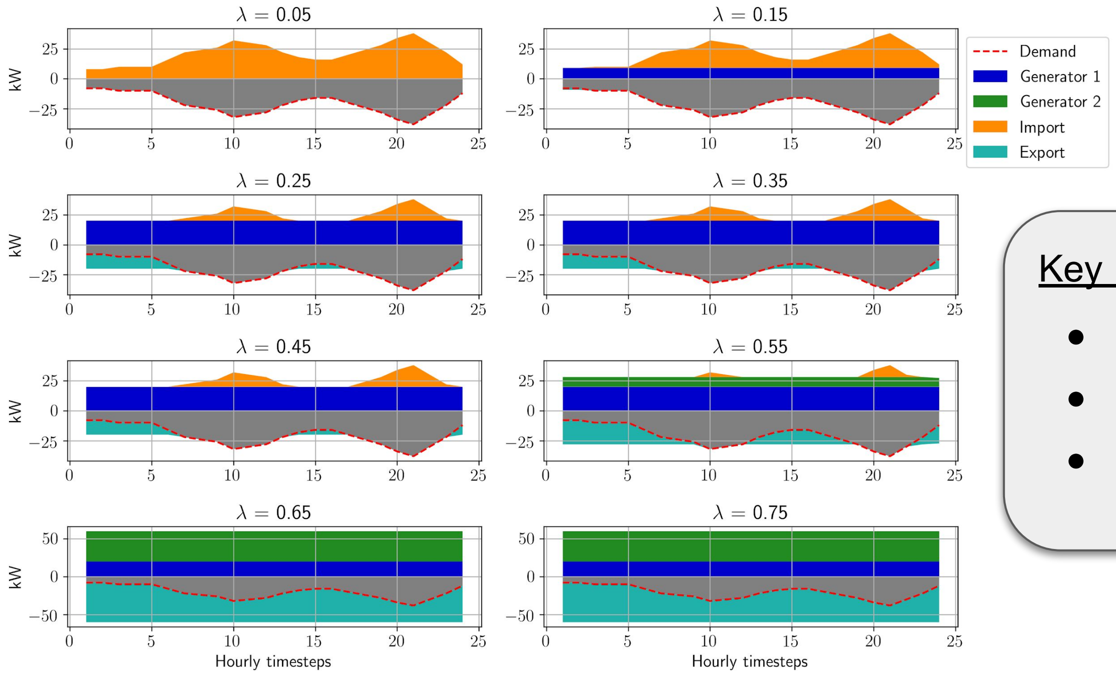


Key Insights:

- Higher lambda ⇒ higher TGR generation
- At λ = 0.55 $/kWh
  - Objective value starts to decrease
  - Exports are higher than imports
- Best objective value: -75.31 $

Results | Sensitivity analysis

Energybalance



<u>Key Insights:</u>

- Full capacity of generator 1 at λ = 0.25
- Full capacity of generator 2 at λ = 0.65
- No curtailment

Thanks for your attention

Eric Rockstädt
Theodor Schönfisch
Isabell von Falkenhausen