Oemof – User Meeting

Requirements of small energy systems

BinaryFlow and additional constraints

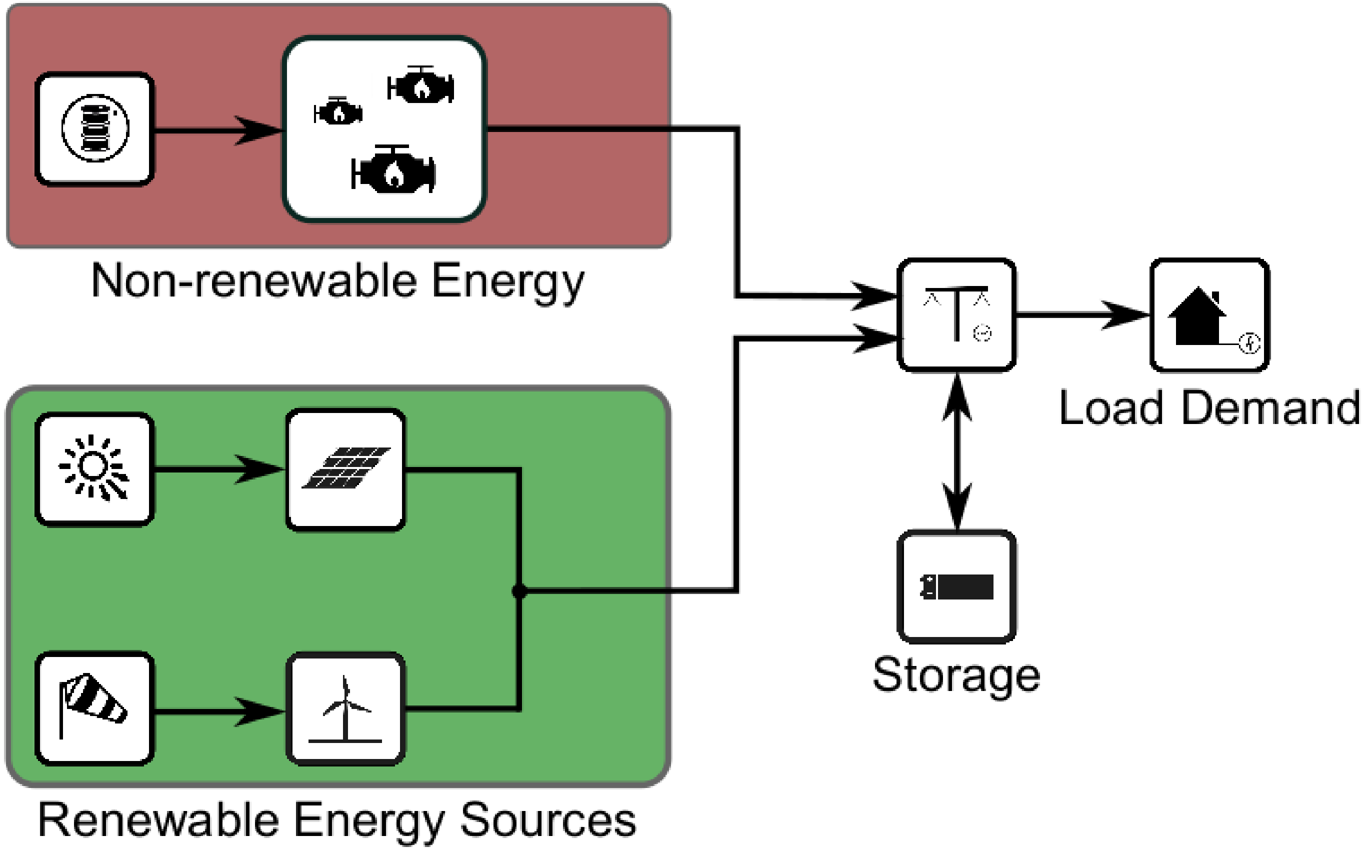
Hendrik Huyskens (with thanks to Uwe and Cord for support)

Contents

- Problem definition / Challenges simulating small island grids
- Implementation in Oemof
- Results
- Conclusion/Lookout

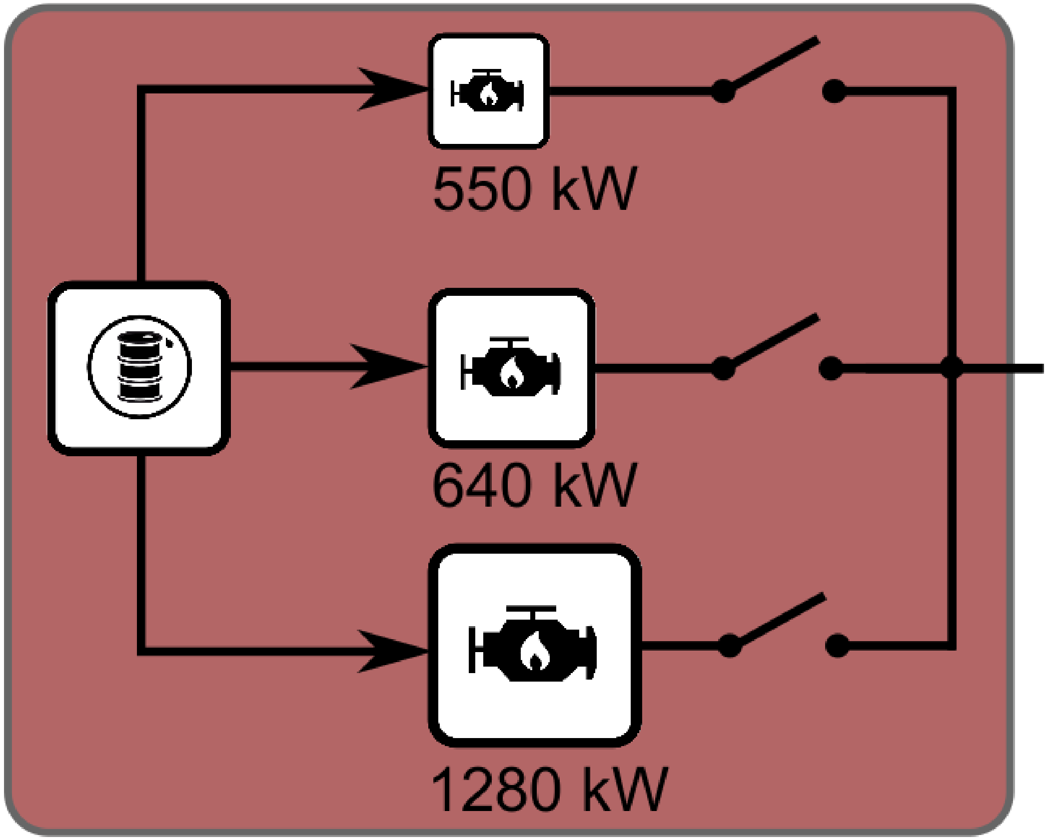
Problem definition – System Overview

Example overview of small island grid
(normally „1-node system“)



Problem definition – Generator Setup

- Multiple (small) diesel generators instead of one generic/combined generator
- Can be switched on and off
- Generators have different efficiencies *(and load curves)*
- Run at least on minimal loading
- *(Should have minimal runtimes)*



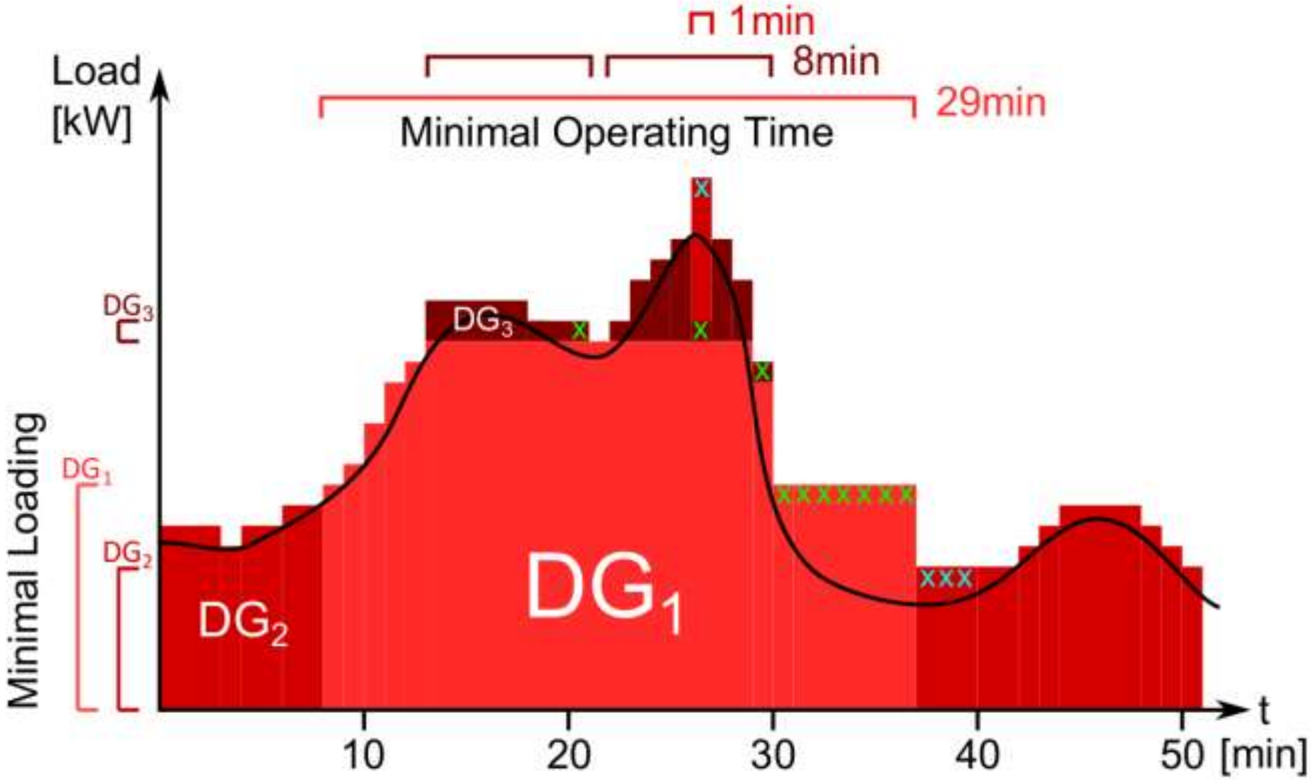
Generators should be implemented as exact as possible, since pv feedin, fuel consumption etc. highly depent on generator set!

Problem definition – Load dispatch

Example dispatch of three generators following load curve

Constraints:

- Minimal loading
- Minimal runtime



Problem definition – Spinning Reserve

Additional constraint for small (hybrid) island grids *(is valid for every energy system)*:

In times of an energy lack by renewables, extra energy must be provided to the system very quickly!
This extra energy on hold is called *Spinning Reserve* and can be provided (in our system) by generators or the battery.

In our model, we introduced a constraint called „Rotating mass“, which is similar to the concept of Spinning Reserve:

Certain percentage of load must be covered by running generators or provided as spinning reserve by the battery

(in the future, correct adaption of Spinning Reserve is planned)

Implementation – Minimal loading, switching on/off

Generators are implemented using a *LinearTransformer* with a *BinaryFlow* as output flow, considering minimal and maximal loading.

BinaryFlow:

- Makes use of variables which are either 0 or 1
- Leads to mixed-integer linear problem (MILP) instead of LIP! (more complex)

```
def _minimum_flow_rule(block, i, o, t):
    """Rule definition for MILP minimum flow constraints.
    """
    expr = (self.status[i, o, t] *
            m.flows[i, o].min[t] * m.flows[i, o].nominal_value <=
            m.flow[i, o, t])
    return expr
self.min = Constraint(self.MIN_FLOWS, m.TIMESTEPS,
                      rule=_minimum_flow_rule)
```

Generator instance:

```
pp_oil = LinearTransformer(
    label='pp_oil',
    inputs={boil: Flow()},
    outputs={
        b_el: Flow(
            nominal_value=53,
            min=0.1,
            max=1,
            binary=BinaryFlow()
        )
    },
    conversion_factors={b_el: 10.6},
)
```

Implementation – Spinning Reserve

```
# CONSTRAINTS:
rmblock = po.Block()

def rotating_mass_rule(m, t):
    return (
        om.flow[pp_oil, b_el, t] +
        om.flow[pp_oil_medium, b_el, t] +
        om.flow[pp_oil_small, b_el, t] >=
        data['load'][t] * SPINNING_RESERVE -
        (
            om.Storage.capacity[storage, t] -
            storage.nominal_capacity *
            storage.capacity_min[t]
        ) * storage.nominal_output_capacity_ratio
    )

rmblock.rm_constr = po.Constraint(
    om.TIMESTEPS, rule=rotating_mass_rule)

# add the sub-model to the oemof OperationalModel instance
om.add_component('Rotating Mass', rmblock)
```

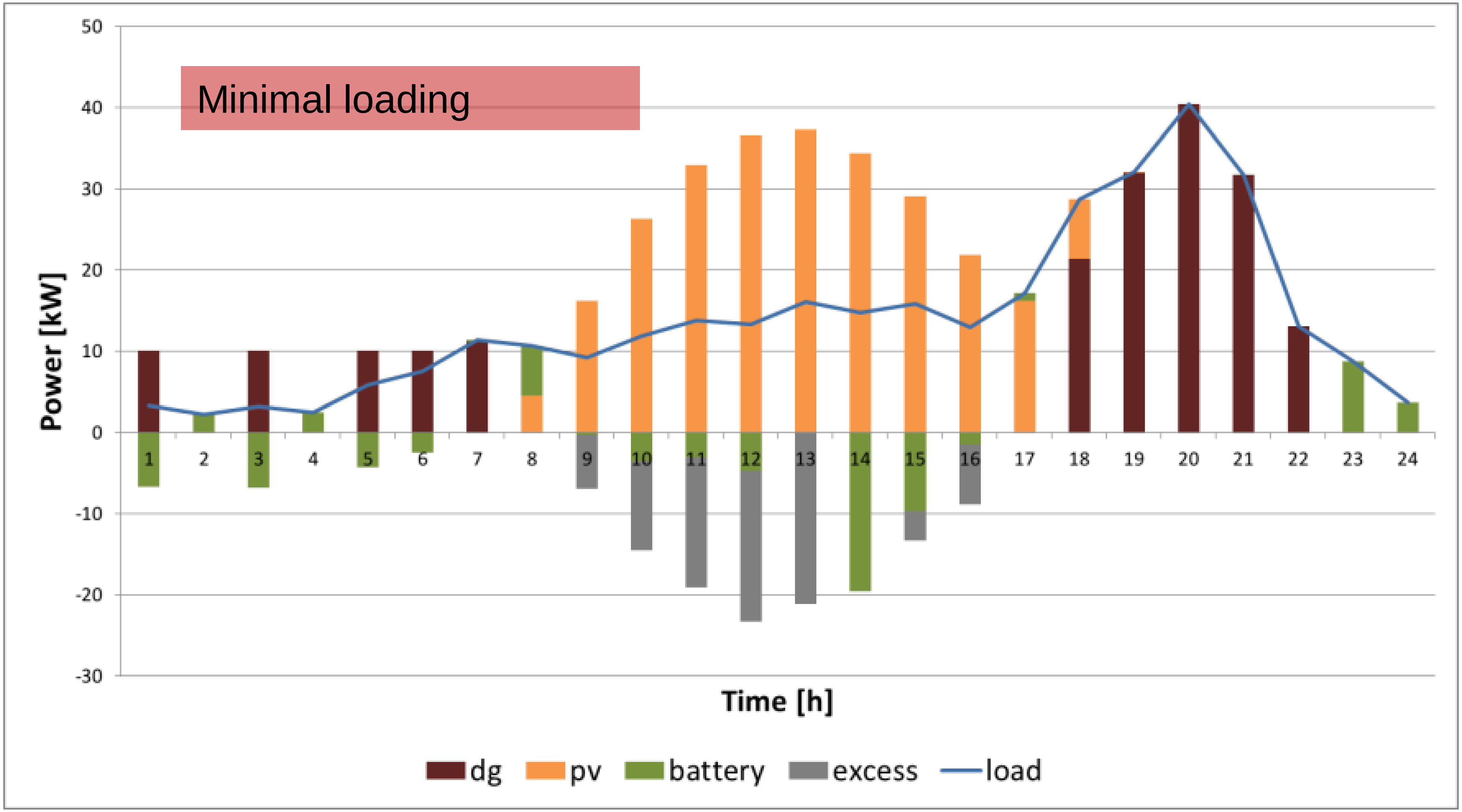
Create pyomo-Block

Define function for rotating mass

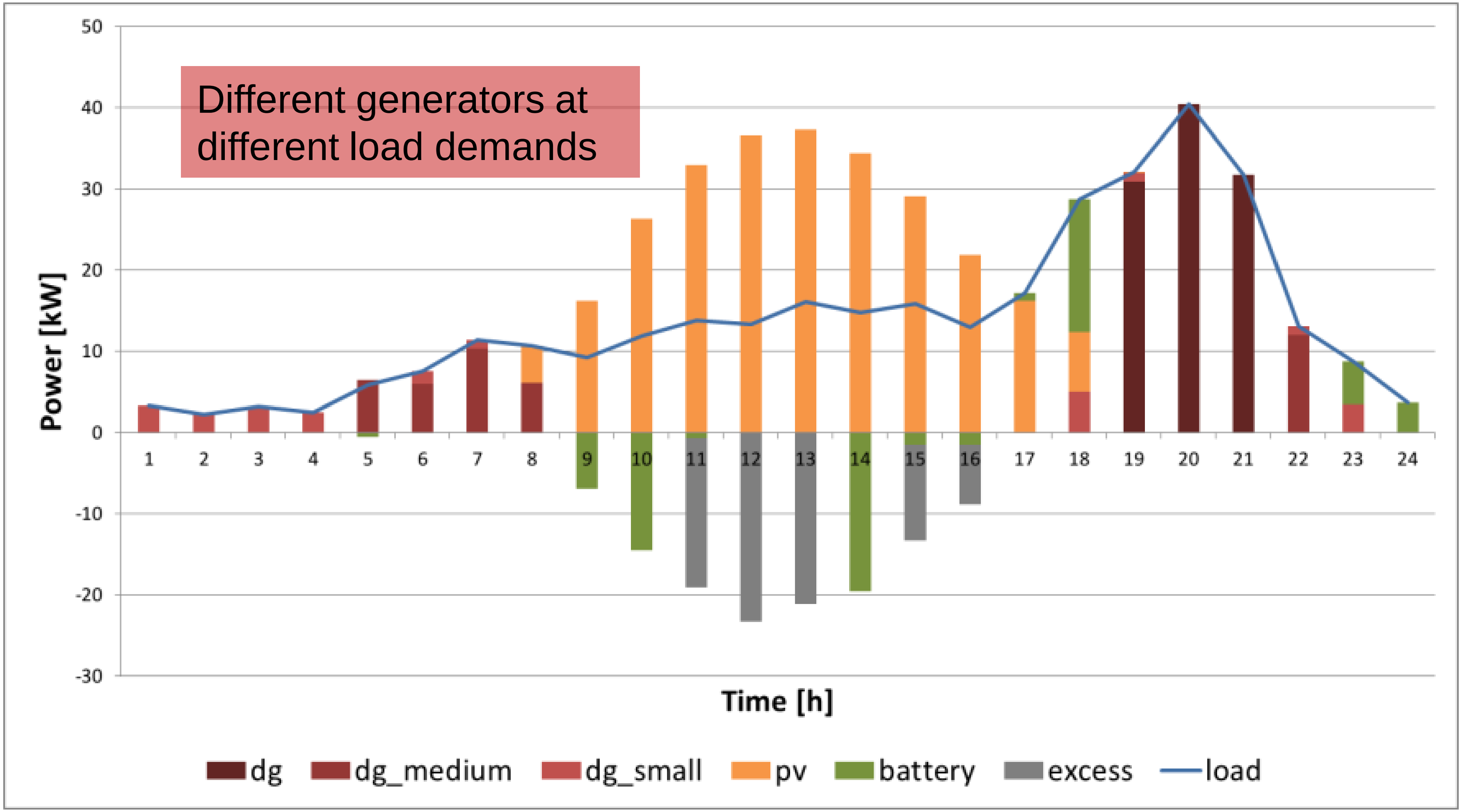
Add constraint to block, using above function as rule

Add block to OperationalModel

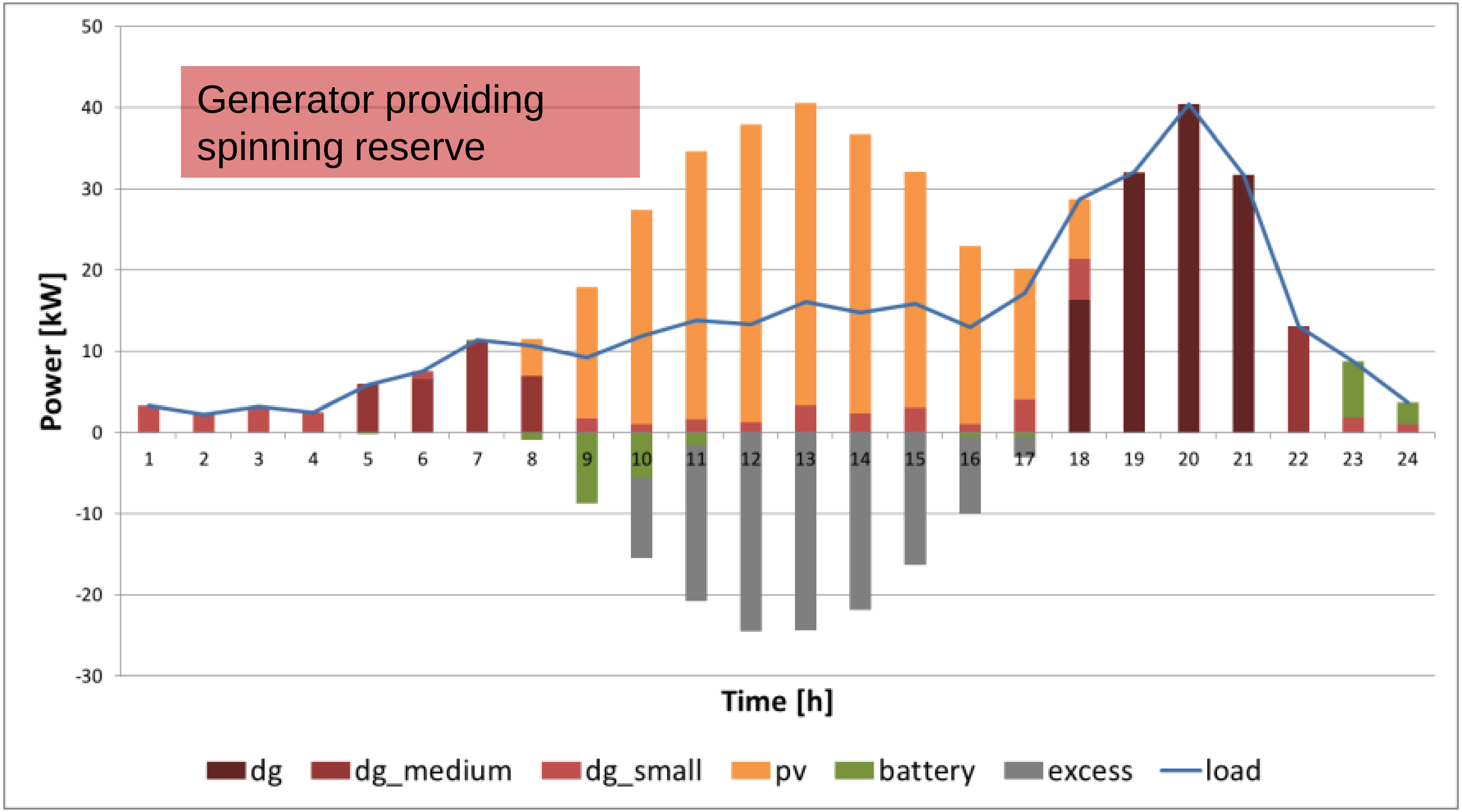
Results – Simulation of one big generator with high minimal loading (20%)



Results – Simulation of three different generators with medium spinning reserve constraint (40%)



Results – Simulation of three different generators with high spinning reserve constraint (70%) and small battery



Conclusion/Lookout

- Minimal loading is already there!
- Easy and fast implementation of additional constraints is possible
- Oemof is capable of mixed-integer problems allowing modelling of more complex models

Lookout (to be done…).

- Load curve
- Implementation of minimal (maximal) runtime [->in BinaryFlow]
- Correct adaption of Spinning Reserve
- Cost-optimization of scenario