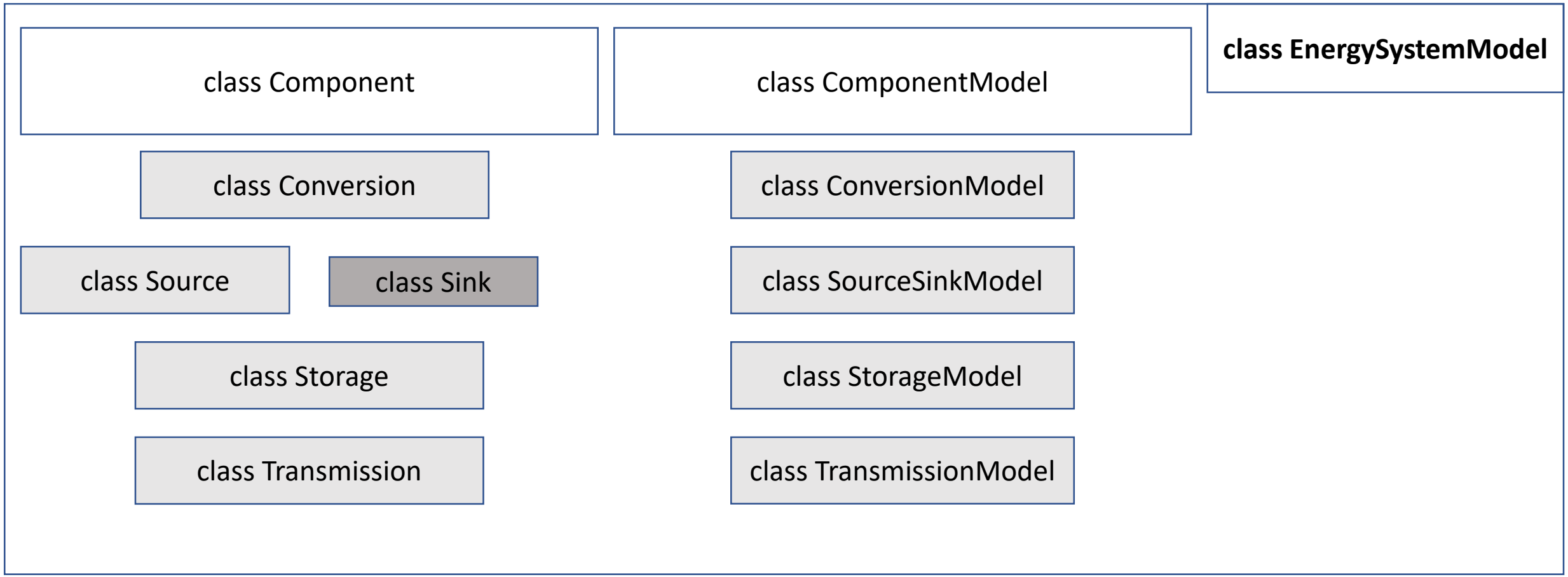
FINE

Framework for Integrated Energy System Assessment

Structure


class Component
class ComponentModel
class EnergySystemModel
class Conversion
class ConversionModel
class Source
class Sink
class SourceSinkModel
class Storage
class StorageModel
class Transmission
class TransmissionModel

Structure

class Sink

class DemandSideManagementBETA

class Conversion

class ConversionDynamic

class ConversionPartLoad

class Transmission

class LinearOptimalPowerFlow

class Storage

class StorageExtBETA

Operation

1. Required packages are imported and the input data path is set
2. An energy system model instance is created
3. Commodity sources are added to the energy system model
4. Commodity conversion components are added to the energy system model
5. Commodity storages are added to the energy system model
6. Commodity transmission components are added to the energy system model
7. Commodity sinks are added to the energy system model
8. The energy system model is optimized
9. Selected optimization results are presented

Operation

1. Required packages are imported and the input data path is set

Data is imported by hand, one parameter by one parameter, in another script.

```
50      # Biogas data
51      operationRateMax = pd.read_excel(os.path.join(inputDataPath, 'SpatialData', 'Biogas',
52                                                    'biogasPotential_GWh_biogas.xlsx'),
53                                                    header = 0, index_col = 0)
54
55      data.update({'Biogas, operationRateMax': operationRateMax})
```

```
135      # Electricity demand data
136      operationRateFix = pd.read_excel(os.path.join(inputDataPath, 'SpatialData', 'Demands',
137                                                     'electricityDemand_GWh_el.xlsx'),
138                                                     header = 0, index_col = 0)
139
140      data.update({'Electricity demand, operationRateFix': operationRateFix})
```

Operation

2. An energy system model instance is created

Locations and timestep are defined, as well as commodities and their unit.

Creation of the model.

In [2]:
```
locations = {'cluster_0', 'cluster_1', 'cluster_2', 'cluster_3', 'cluster_4', 'cluster_5', 'cluster_6', 'cluster_7'}
commodityUnitDict = {'electricity': r'GW$_{el}$', 'methane': r'GW$_{CH_{4},LHV}$', 'biogas': r'GW$_{biogas,LHV}$',
                     'CO2': r'Mio. t$_{CO_2}$/h', 'hydrogen': r'GW$_{H_{2},LHV}$'}
commodities = {'electricity', 'hydrogen', 'methane', 'biogas', 'CO2'}
numberOfTimeSteps=8760
hoursPerTimeStep=1
```

In [3]:
```
esM = fn.EnergySystemModel(locations=locations, commodities=commodities, numberOfTimeSteps=8760,
                           commodityUnitsDict=commodityUnitDict,
                           hoursPerTimeStep=1, costUnit='1e9 Euro', lengthUnit='km', verboseLogLevel=0)
```

Operation

3. Commodity sources are added to the energy system model

Sources added by hand, some with the help of the data file created.

PV

In [9]:
```
esM.add(fn.Source(esM=esM, name='PV', commodity='electricity', hasCapacityVariable=True,
                  operationRateMax=data['PV, operationRateMax'], capacityMax=data['PV, capacityMax'],
                  investPerCapacity=0.65, opexPerCapacity=0.65*0.02, interestRate=0.08,
                  economicLifetime=25))
```

Operation

4. Commodity conversion components are added to the energy system model

Every conversion component is added by hand.

Combined cycle gas turbine plants

In [14]:
```
esM.add(fn.Conversion(esM=esM, name='CCGT plants (methane)', physicalUnit=r'GW$_{el}$',
                      commodityConversionFactors={'electricity':1, 'methane':-1/0.6, 'CO2':201*1e-6/0.6},
                      hasCapacityVariable=True,
                      investPerCapacity=0.65, opexPerCapacity=0.021, interestRate=0.08,
                      economicLifetime=33))
```

Electrolyzers

In [17]:
```
esM.add(fn.Conversion(esM=esM, name='Electrolyzer', physicalUnit=r'GW$_{el}$',
                      commodityConversionFactors={'electricity':-1, 'hydrogen':0.7},
                      hasCapacityVariable=True,
                      investPerCapacity=0.5, opexPerCapacity=0.5*0.025, interestRate=0.08,
                      economicLifetime=10))
```

Operation

5. Commodity storages are added to the energy system model

Stoarges added by hand, some with the help of the data file created.

Lithium ion batteries

The self discharge of a lithium ion battery is here described as 3% per month. The self discharge per hours is obtained using the equation $(1-\text{selfDischarge}_{\text{hour}})^{30*24h} = 1-\text{selfDischarge}_{\text{month}}$.

In [19]:
```
esM.add(fn.Storage(esM=esM, name='Li-ion batteries', commodity='electricity',
                   hasCapacityVariable=True, chargeEfficiency=0.95,
                   cyclicLifetime=10000, dischargeEfficiency=0.95, selfDischarge=1-(1-0.03)**(1/(30*24)),
                   chargeRate=1, dischargeRate=1, doPreciseTsaModeling=False,
                   investPerCapacity=0.151, opexPerCapacity=0.002, interestRate=0.08,
                   economicLifetime=22))
```

Pumped hydro storage

In [22]:
```
esM.add(fn.Storage(esM=esM, name='Pumped hydro storage', commodity='electricity',
                   chargeEfficiency=0.88, dischargeEfficiency=0.88,
                   hasCapacityVariable=True, selfDischarge=1-(1-0.00375)**(1/(30*24)),
                   chargeRate=0.16, dischargeRate=0.12, capacityFix=data['Pumped hydro storage, capacityFix'],
                   investPerCapacity=0, opexPerCapacity=0.000153))
```

Operation

6. Commodity transmission components are added to the energy system model

Transmission components added by hand, some with the help of the data file created.

DC cables

In [24]:
```
esM.add(fn.Transmission(esM=esM, name='DC cables', commodity='electricity', losses=data['DC cables, losses'],
                        distances=data['DC cables, distances'],
                        hasCapacityVariable=True, capacityFix=data['DC cables, capacityFix']))
```

Operation

7. Commodity sinks are added to the energy system model

Sinks added by hand with the help of the data file.

Electricity demand

In [27]:
```
esM.add(fn.Sink(esM=esM, name='Electricity demand', commodity='electricity',
                hasCapacityVariable=False, operationRateFix=data['Electricity demand, operationRateFix']))
```

CO2 exiting the system's boundary

In [29]:
```
esM.add(fn.Sink(esM=esM, name='CO2 to enviroment', commodity='CO2',
                hasCapacityVariable=False, commodityLimitID='CO2 limit', yearlyLimit=366*(1-CO2_reductionTarget)))
```

Operation

8. The energy system model is optimized

```
In [31]: esM.optimize(timeSeriesAggregation=True, optimizationSpecs='OptimalityTol=1e-3 method=2 cuts=0')
```

Operation

9. Selected optimization results are presented

Models summed up in tables, installed capacities on maps, operation on graphics.

```
In [40]: esM.getOptimizationSummary("ConversionModel", outputLevel=2)
```

Out[40]:

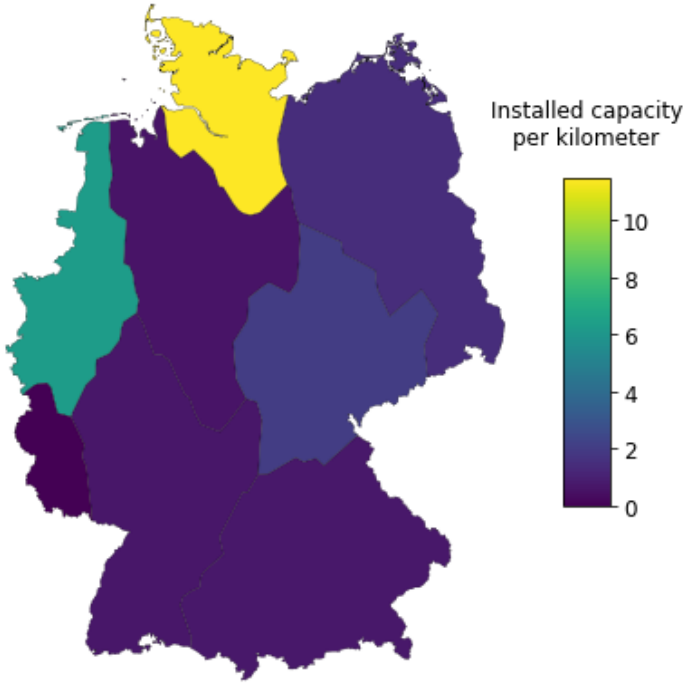
			cluster_0	cluster_1	cluster_2	cluster_3	cluster_4	cluster_5	cluster_6	cluster_7
Component	**Property**	**Unit**								
Electrolyzer	**TAC**	**[1e9 Euro/a]**	0.0614655	0.0509074	0.545151	0.128468	0.064736	0.178004	0.996726	0.00210291
	capacity	**[GW $_{el}$]**	0.70638	0.585044	6.26505	1.4764	0.743966	2.04567	11.4547	0.0241673
	capexCap	**[1e9 Euro/a]**	0.0526358	0.0435944	0.466838	0.110013	0.0554364	0.152433	0.853542	0.00180082
	invest	**[1e9 Euro]**	0.35319	0.292522	3.13252	0.738199	0.371983	1.02284	5.72734	0.0120837
	operation	**[GW $_{el}$ *h/a]**	2430.74	1835.28	25503.2	5978.44	2507.62	8210.61	52811.6	105.27
	opexCap	**[1e9 Euro/a]**	0.00882975	0.00731305	0.0783131	0.018455	0.00929957	0.0255709	0.143183	0.000302092
New CCGT plants (biogas)	**TAC**	**[1e9 Euro/a]**	0.0574716	0.232827	0.314601	0.135068	0.076445	0.086687	0	0.00920039
	capacity	**[GW $_{el}$]**	0.70262	2.84643	3.84617	1.65128	0.934579	1.05979	0	0.112479
	capexCap	**[1e9 Euro/a]**	0.0427166	0.173052	0.233832	0.100392	0.0568188	0.0644313	0	0.00683832
	invest	**[1e9 Euro]**	0.491834	1.9925	2.69232	1.1559	0.654205	0.741855	0	0.0787356
	operation	**[GW $_{el}$ *h/a]**	1236.16	4875.83	6863.73	2843.82	1684.95	1812.25	0	213.263
	opexCap	**[1e9 Euro/a]**	0.014755	0.059775	0.0807695	0.0346769	0.0196262	0.0222556	0	0.00236207

Operation

9. Selected optimization results are presented

Models summed up in tables, installed capacities on maps, operation on graphics.

```
In [41]: fig, ax = fn.plotLocationalColorMap(esM, 'Electrolyzer', locFilePath, 'index', perArea=False)
```

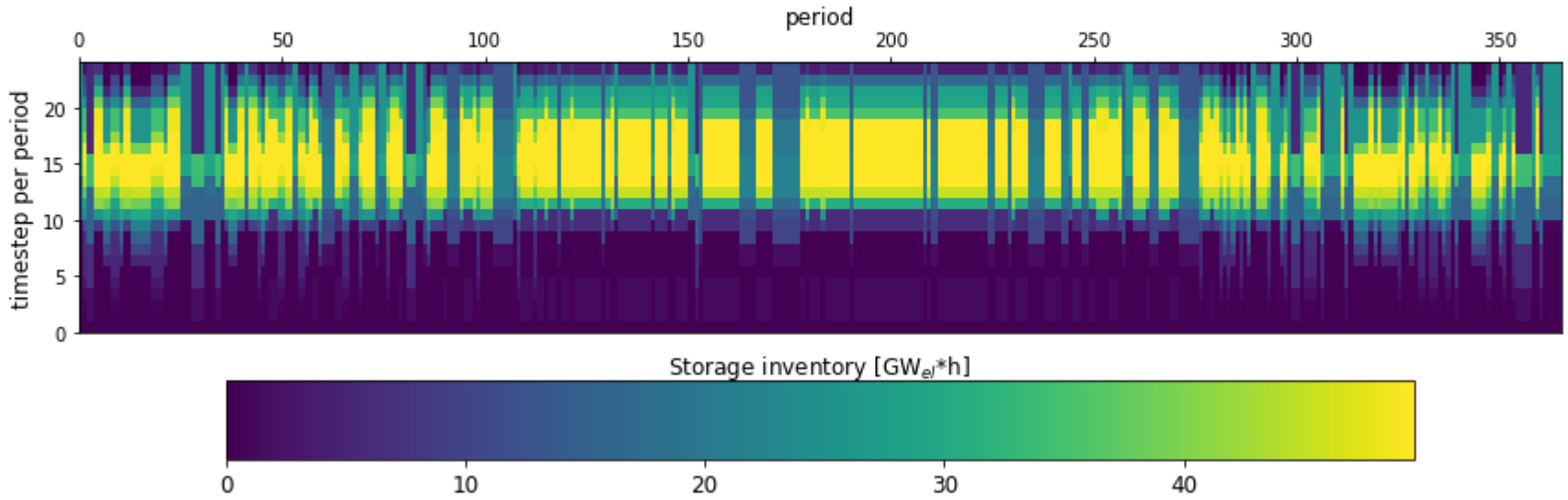


Operation

9. Selected optimization results are presented

Models summed up in tables, installed capacities on maps, operation on graphics.

```
In [44]: fig, ax = fn.plotOperationColorMap(esM, 'Li-ion batteries', 'cluster_2',
                                            variableName='stateOfChargeOperationVariablesOptimum')
```



Limits - Remarks

- Platform still being updated
- 13 minutes to run their basic example (8 locations, 7 typical days for a whole year)
- Optimisation of a district simple, but specific origin unknown
- Issue with the introduction of a function calculating environmental impacts
  - Every xModel class would have to be updated
  - Currently: simple limit of yearly emissions
- Technologies models quite simple
  - Define new classes ?

Limits

- Issue switching to non-linear
  - Would have to change the solver inside the main class

- No choice possible between procudcing space heat or domestic hot water

- Cluster of the horizon with typical periods imposed

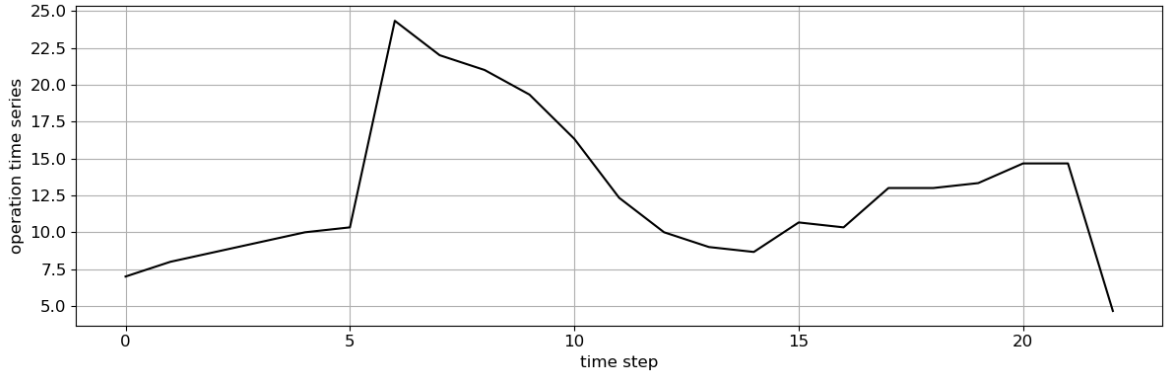
- Operation color maps only for a one year horizon, no title on the graphs
  - Easy to change, but defined inside a specific file

Adaptations + outputs

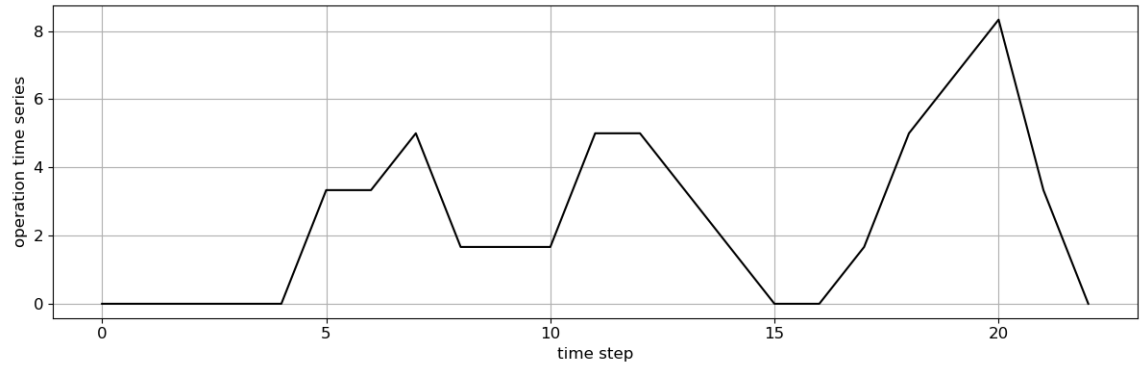
CHP -> only produces space heat

HP -> only produces domestic hot water

No thermal storage yet



CHP1 on building 1 production



HP3 on building 1 production

Outputs

Component	Property	Unit	building_1	building_2
CO2 to enviroment	operation	[kg$_{CO_2}$/h*h/a]	115563,8667	95426,93833
		[kg$_{CO_2}$/h*h]	316,6133333	261,4436667
Domestic hot water demand	operation	[kW$_{dhw}$*h/a]	62050	83585
		[kW$_{dhw}$*h]	170	229
Electricity demand	operation	[kW$_{el}$*h/a]	145270	176660
		[kW$_{el}$*h]	398	484
Electricity grid	TAC	[Euro/a]	0	312,9539591
	capacity	[kW$_{el}$]	10	10
	commodCosts	[Euro/a]	0	564,5246658
	commodRevenues	[Euro/a]	0	251,5707067
	operation	[kW$_{el}$*h/a]	0	2515,707067
		[kW$_{el}$*h]	0	6,892348129
Natural gas	TAC	[Euro/a]	30535,9	24428,72
	commodCosts	[Euro/a]	30535,9	24428,72
	operation	[kW$_{CH4}$*h/a]	324850	259880
		[kW$_{CH4}$*h]	890	712
Space heat demand	operation	[kW$_{sh}$*h/a]	324850	259880
		[kW$_{sh}$*h]	890	712

Component	Property	Unit	building_1	building_2
CHP1	TAC	[Euro/a]	11143,97599	0
	capacity	[kW$_{CH4}$]	24,33333333	0
	capexCap	[Euro/a]	3009,898907	0
	capexIfBuilt	[Euro/a]	3084,910416	0
	invest	[Euro]	40896,66667	0
	isBuilt	[-]	1	0
	operation	[kW$_{CH4}$*h/a]	108283,3333	0
		[kW$_{CH4}$*h]	296,6666667	0
	opexCap	[Euro/a]	5049,166667	0

Component	Property	Unit	building_1	building_2
Battery3	TAC	[Euro/a]	22180,77657	10734,81009
	capacity	[kW$_{el}$*h]	111,3186984	53,78590078
	capexCap	[Euro/a]	9456,168164	4568,940617
	capexIfBuilt	[Euro/a]	34,2767824	34,2767824
	invest	[Euro]	63681,6581	30887,96345
	isBuilt	[-]	1	1
	operationCharge	[kW$_{el}$*h/a]	703082,5959	256079,8248
		[kW$_{el}$*h]	1926,253687	701,588561
	operationDischarge	[kW$_{el}$*h/a]	544185,9292	198205,7844
		[kW$_{el}$*h]	1490,920354	543,0295462
	opexCap	[Euro/a]	12690,33162	6131,592689

Outputs

Component	Property	Unit	LocationIn	building_1	building_2
Cables	TAC	[Euro/a]	building_1	0	1,490294887
			building_2	1,490294887	0
	capacity	[kW$_{el}$]	building_1		20
			building_2	20	
	capexIfBuilt	[Euro/a]	building_1		1,490294887
			building_2	1,490294887	
	invest	[Euro]	building_1		10
			building_2	10	
	isBuilt	[-]	building_1		1
			building_2	1	
	operation	[kW$_{el}$*h/a]	building_1		18006,66667
			building_2	73365	
		[kW$_{el}$*h]	building_1		49,33333333
			building_2	201	