Internship logs from Apr.29 to May.05

Zhichuan MA

Table of contents

Last week I have done the reading for the official website, learned the constraints used in the REHO optimization model. This week's focus will be laid on the scripts. The expected output of this week will be the **introduction of LCA method they utilised and critisize** and **the beginning of using current REHO to do a case study**.

I also decide to read the codes again (though I have done the reading last week) more in detail.

1 Apr.29

Today's key point is to read through the Package structure combined with the codes.

1.1 /data

1.1.1 elcom

This is a dataset composed of **the city id, commune, id_operator and operator**. But I haven't seen its use in the model since there is no numerical value related to calculation.

In my case study, **maybe** I need to build a database like this by myself, or not.

1.1.2 emissions

According to the name "electricity_matrix_2019_reduced.csv", this is some environmental impacts of the electricity, but the numbers in the columns remain unknown.

More importantly, I didn't find corresponding value in ecoinvent.

1.1.3 infrastructure

These are datasets for different units used in building or districts. It involves in 13 kinds of environmental impacts. The storage units are lack of a lot of datas relevant to the environmental impacts.

1.1.4 QBuilding

They are online databases and we are not be able to access.

1.1.5 SIA

- sia2024_data.xlsx: Properties for different rooms.
- sia2024_rooms_sia380_1.csv: Room compositions for different types of building.

1.1.6 skydome

skypatch means the sky area we observe.

Irradiation gives one year's irradiation data for 2005.

1.1.7 weather

1 year's data for 6 different cities in Switzerland.

1.2 model

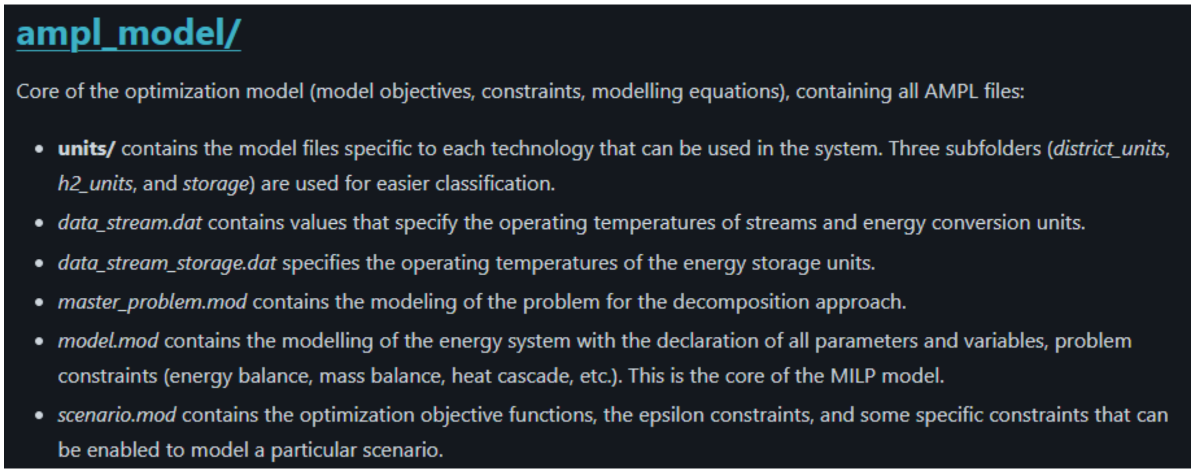
The main purpose to read the scripts are:

- 1. Know the inputs of REHO for my further implement.
- 2. Discover the objective function.
- 3. Find the method they used for LCA.

1.2.1 ampl_model

Based on the DW decomposition the model structure is like master problem + subproblems

EMOO means evolutionary multi objective optimization.



ampl_model/

Core of the optimization model (model objectives, constraints, modelling equations), containing all AMPL files:

- **units/** contains the model files specific to each technology that can be used in the system. Three subfolders (*district_units*, *h2_units*, and *storage*) are used for easier classification.
- *data_stream.dat* contains values that specify the operating temperatures of streams and energy conversion units.
- *data_stream_storage.dat* specifies the operating temperatures of the energy storage units.
- *master_problem.mod* contains the modeling of the problem for the decomposition approach.
- *model.mod* contains the modelling of the energy system with the declaration of all parameters and variables, problem constraints (energy balance, mass balance, heat cascade, etc.). This is the core of the MILP model.
- *scenario.mod* contains the optimization objective functions, the epsilon constraints, and some specific constraints that can be enabled to model a particular scenario.

Figure 1: ample_model

1.2.2 postprocessing

1.2.3 preprocessing

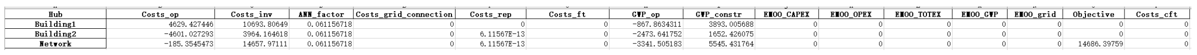
1.3 scripts

Here I will implement the file "3a_Read_csv" in different scenarios.

1.3.1 Building-scale

1.3.1.1 Single optimization

```
from reho.model.reho import *

if __name__ == '__main__':

    # Set building parameters
    # you can as well define your district from a csv file instead of reading the database
    reader = QBuildingsReader()
    qbuildings_data = reader.read_csv(buildings_filename='../template/data/buildings.csv', nb

    # Select weather data
    cluster = {'Location': 'Geneva', 'Attributes': ['I', 'T', 'W'], 'Periods': 10, 'PeriodDu

    # Set scenario
    scenario = dict()
    scenario['Objective'] = 'TOTEX'
    scenario['name'] = 'totex'
    scenario['exclude_units'] = ['NG_Cogeneration', 'OIL_Boiler', 'ThermalSolar']
    scenario['enforce_units'] = []


    # Initialize available units and grids
    grids = infrastructure.initialize_grids()
    units = infrastructure.initialize_units(scenario, grids)

    # Set method options
    method = {'building-scale': True}

    # Run optimization
    reho = reho(qbuildings_data=qbuildings_data, units=units, grids=grids, cluster=cluster, s
    reho.single_optimization()

    # Save results
    reho.save_results(format=['xlsx', 'pickle'], filename='3a')

    # Plot results
    plotting.plot_performance(reho.results, plot='costs').show()
    plotting.plot_sankey(reho.results["totex"][0]).show()
```



Hub	Costs_op	Costs_inv	ANN_factor	Costs_grid_connection	Costs_rep	Costs_ft	GWP_op	GWP_constr	EMOO_CAPEX	EMOO_OPEX	EMOO_TOTEX	EMOO_GWP	EMOO_grid	Objective	Costs_cft
Building1	4629.427446	10693.80649	0.061156718	0	0	0	-867.8634311	3893.005688	0	0	0	0	0	0	0
Building2	-4601.027293	3964.164618	0.061156718	0	6.11567E-13	0	-2473.641752	1652.426075	0	0	0	0	0	0	0
Network	-185.3545473	14657.97111	0.061156718	0	6.11567E-13	0	-3341.505183	5545.431764	0	0	0	0	0	14686.39759	0

Figure 2: Building-scale single optimization

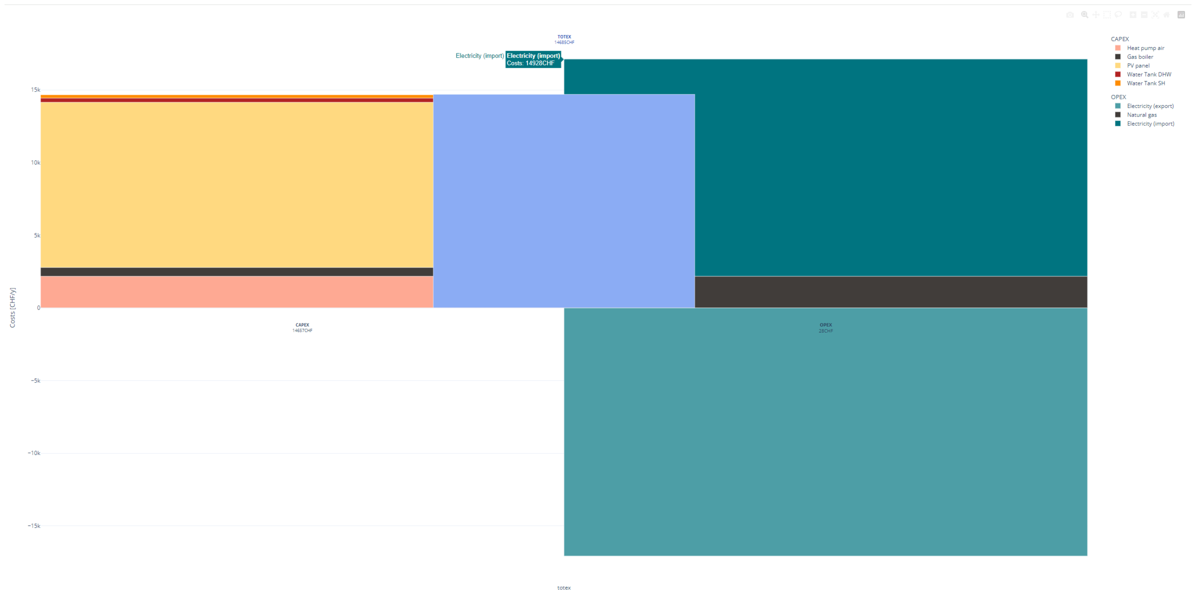


Figure 3: Building-scale single optimization Cost

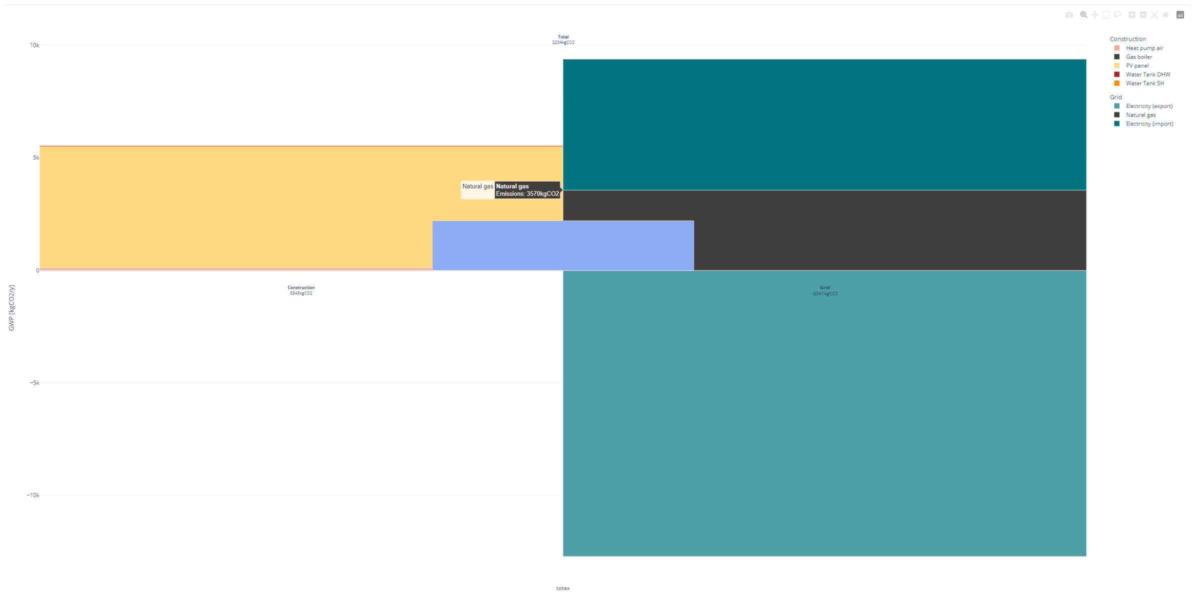


Figure 4: Building-scale single optimization GWP

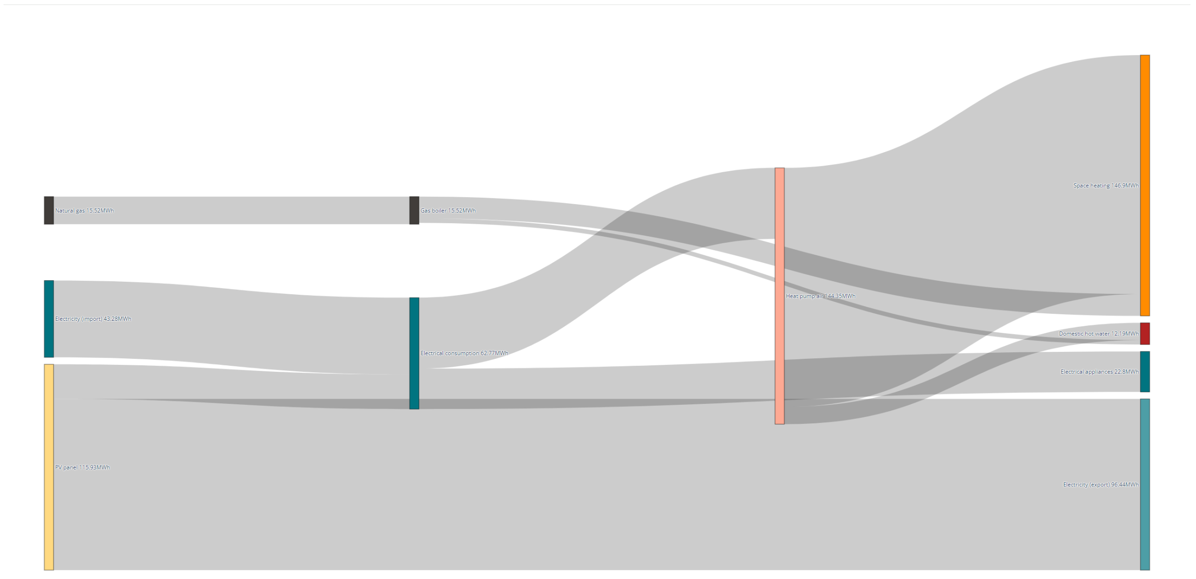


Figure 5: Building-scale single optimization sankey

1.3.1.2 Pareto curve

```
from reho.model.reho import *
from reho.plotting import plotting

if __name__ == '__main__':

    # Set building parameters
    # you can as well define your district from a csv file instead of reading the database
    reader = QBuildingsReader()
    qbuildings_data = reader.read_csv(buildings_filename='../template/data/buildings.csv', nb

    # Select weather data
    cluster = {'Location': 'Geneva', 'Attributes': ['I', 'T', 'W'], 'Periods': 10, 'PeriodDu

    # Set scenario
    scenario = dict()
    scenario['Objective'] = ['CAPEX', 'OPEX']
    scenario['nPareto'] = 2
    scenario['name'] = 'pareto'
    scenario['exclude_units'] = ['NG_Cogeneration', 'OIL_Boiler', 'ThermalSolar']
    scenario['enforce_units'] = []


    # Initialize available units and grids
    grids = infrastructure.initialize_grids()
    units = infrastructure.initialize_units(scenario, grids)

    # Set method options
    method = {'building-scale': True}

    # Run optimization
    reho = reho(qbuildings_data=qbuildings_data, units=units, grids=grids, cluster=cluster, s
    reho.generate_pareto_curve()

    # Save results
    reho.save_results(format=['xlsx', 'pickle'], filename='3a')

    # Performance plot : costs and gwp
    plotting.plot_performance(reho.results, plot='costs', indexed_on='Pareto_ID', label='EN_l
    plotting.plot_performance(reho.results, plot='gwp', indexed_on='Pareto_ID', label='EN_lon

    # Sankey diagram
    for key in reho.results['pareto'].keys():
        plotting.plot_sankey(reho.results['pareto'][key], label='EN_long', color='ColorPaste
```

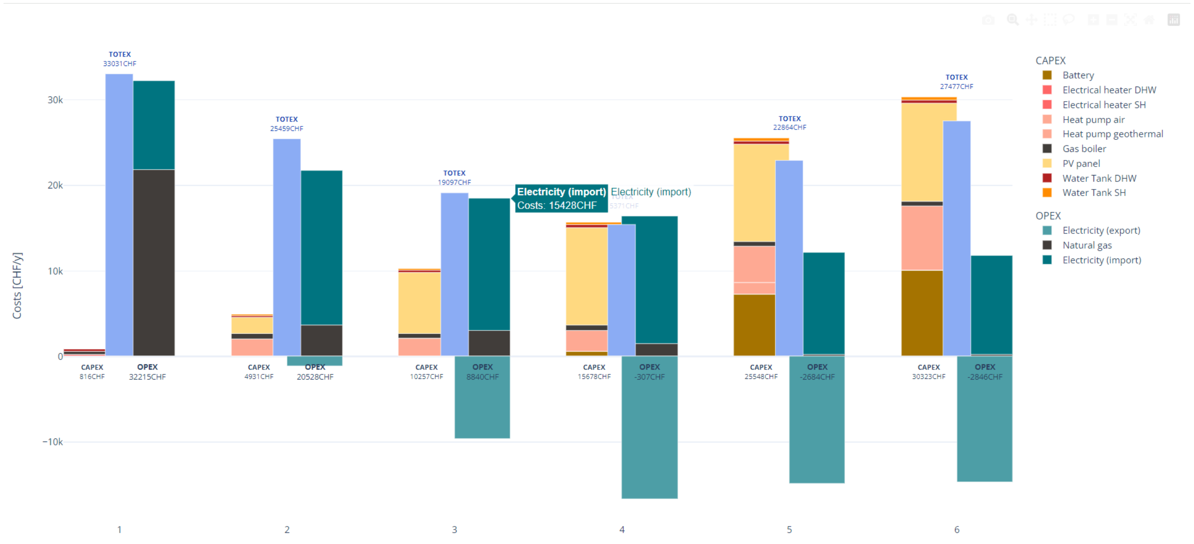


Figure 6: Pareto cost change for building scale

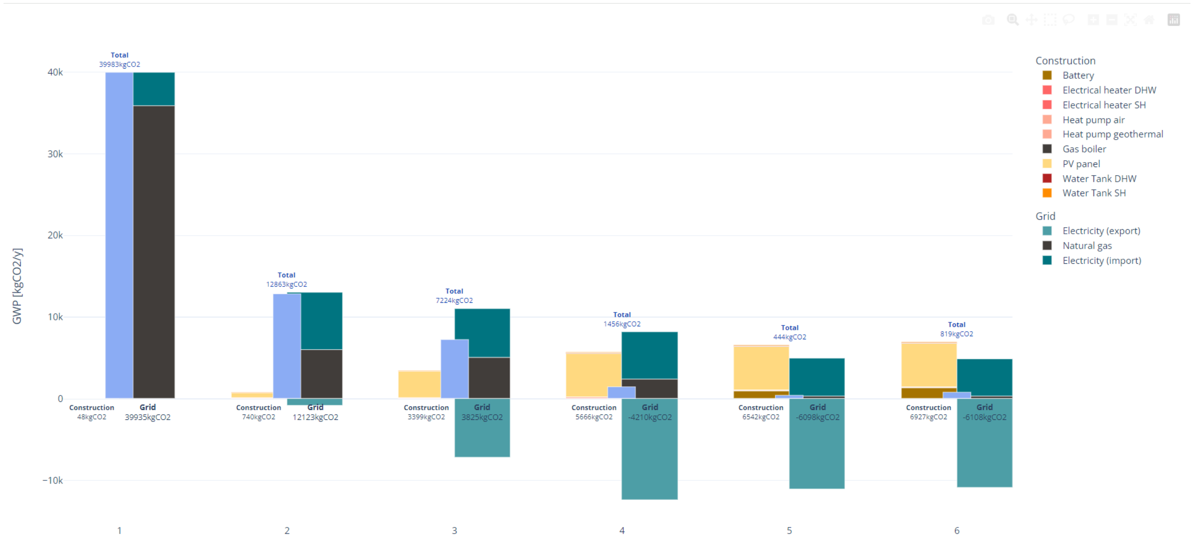


Figure 7: Pareto GWP change for building scale

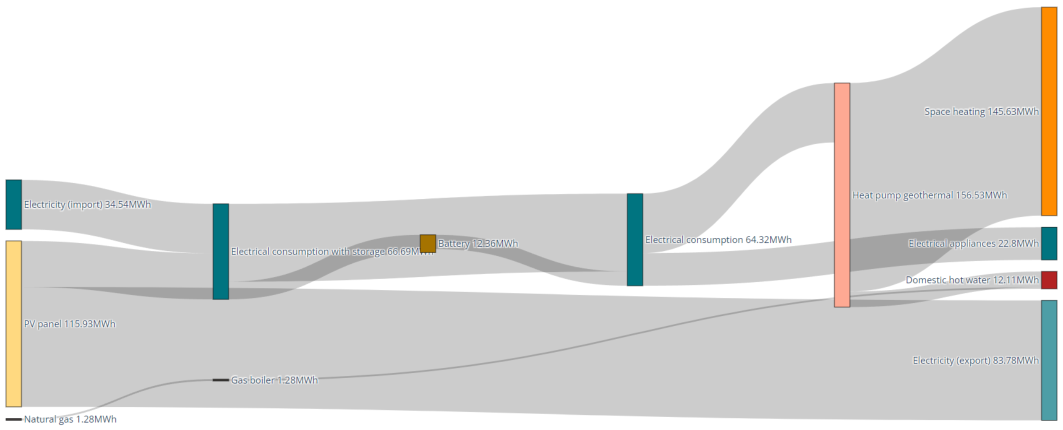


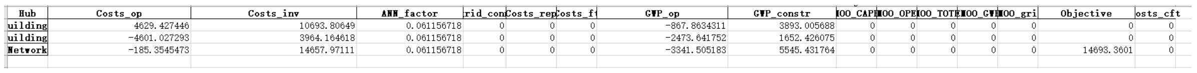
Figure 8: Pareto sankey for building scale

1.3.2 District-scale

1.3.2.1 Single optimization

```
from reho.model.reho import *

if __name__ == '__main__':

    # Set building parameters
    # you can as well define your district from a csv file instead of reading the database
    reader = QBuildingsReader()
    qbuildings_data = reader.read_csv(buildings_filename='../template/data/buildings.csv', nb

    # Select weather data
    cluster = {'Location': 'Geneva', 'Attributes': ['I', 'T', 'W'], 'Periods': 10, 'PeriodDu

    # Set scenario
    scenario = dict()
    scenario['Objective'] = 'TOTEX'
    scenario['name'] = 'totex'
    scenario['exclude_units'] = ['NG_Cogeneration', 'OIL_Boiler', 'ThermalSolar']
    scenario['enforce_units'] = []
```

```
# Initialize available units and grids
grids = infrastructure.initialize_grids()
units = infrastructure.initialize_units(scenario, grids)

# Set method options
method = {'district-scale': True}

# Run optimization
reho = reho(qbuildings_data=qbuildings_data, units=units, grids=grids, cluster=cluster,
reho.single_optimization()

# Save results
reho.save_results(format=['xlsx', 'pickle'], filename='3a')
```



Hub	Costs_op	Costs_inv	ANN_factor	rid_con	Costs_rep	Costs_f	GWP_op	GWP_constr	IOO_CAPE	IOO_OPE	IOO_TOTE	IOO_GW	IOO_gri	Objective	osts_cft
uilding	4629.427446	10693.80649	0.061156718	0	0	0	-867.8634311	3893.005688	0	0	0	0	0	0	0
uilding	-4601.027293	3964.164618	0.061156718	0	0	0	-2473.641752	1652.426075	0	0	0	0	0	0	0
Network	-185.3545473	14657.97111	0.061156718	0	0	0	-3341.505183	5545.431764	0	0	0	0	0	14693.3601	0

Figure 9: District-scale single optimization

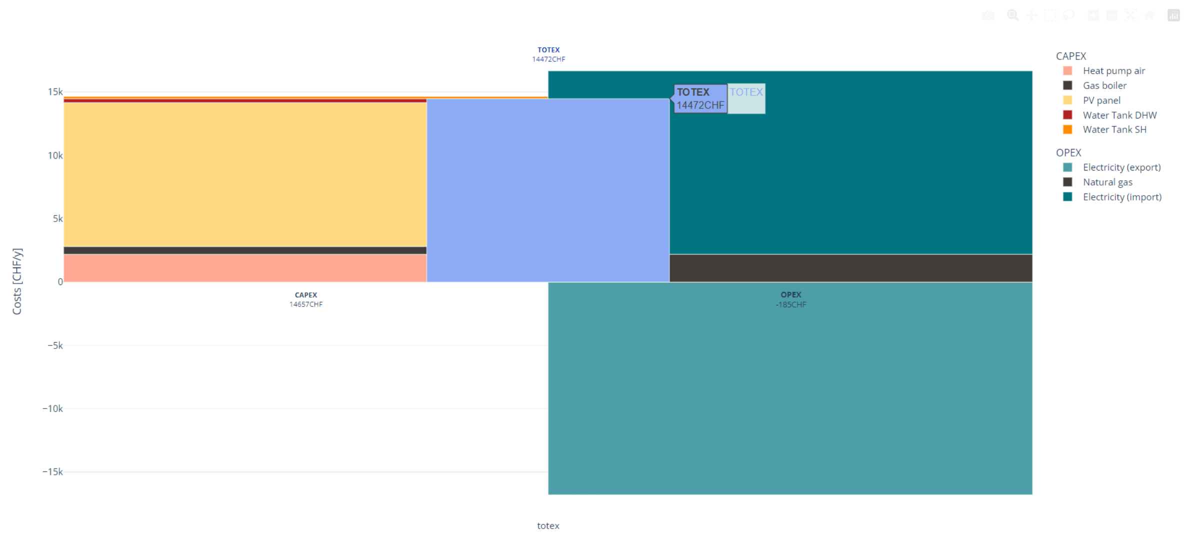


Figure 10: Building-scale single optimization Cost

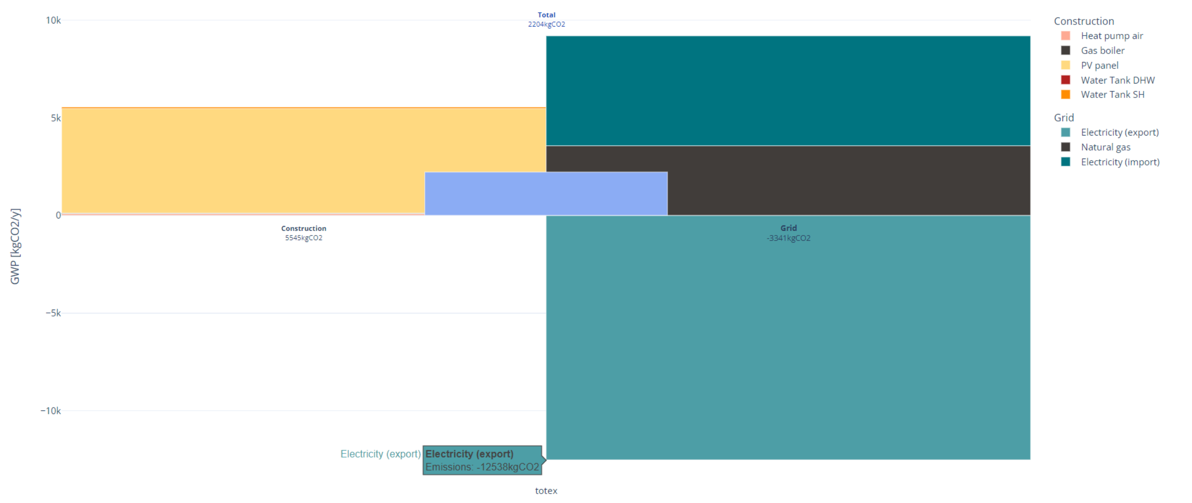


Figure 11: Building-scale single optimization GWP

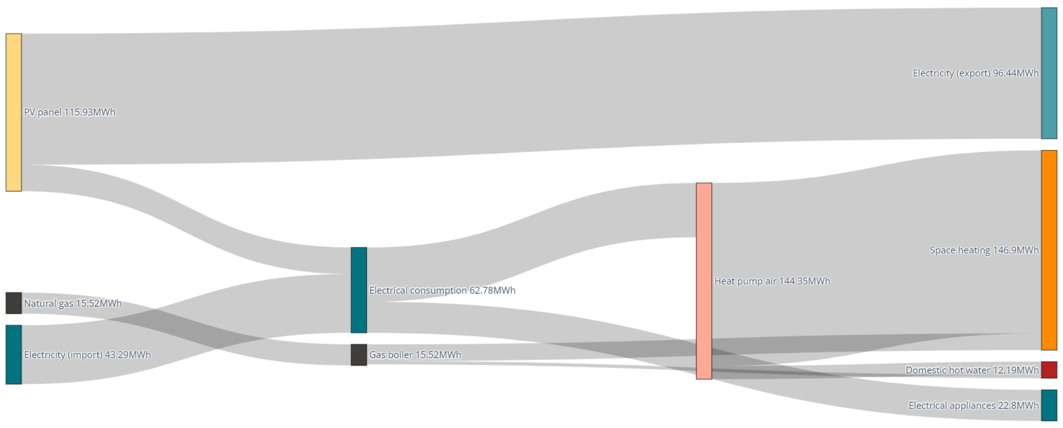

PV panel 115.93MWh
Natural gas 15.52MWh
Electricity (import) 43.29MWh
Electrical consumption 62.78MWh
Gas boiler 15.52MWh
Heat pump air 144.35MWh
Electricity (export) 96.44MWh
Space heating 146.9MWh
Domestic hot water 12.19MWh
Electrical appliances 22.8MWh

Figure 12: Building-scale single optimization sankey

Here I didn't add the limit for the iterations of DW decompositions since the number of buildings are only 2.

1.3.2.2 Pareto curve

```
from reho.model.reho import *
from reho.plotting import plotting

if __name__ == '__main__':

    # Set building parameters
    # you can as well define your district from a csv file instead of reading the database
    reader = QBuildingsReader()
    qbuildings_data = reader.read_csv(buildings_filename='../template/data/buildings.csv', nb

    # Select weather data
    cluster = {'Location': 'Geneva', 'Attributes': ['I', 'T', 'W'], 'Periods': 10, 'PeriodDu

    # Set scenario
    scenario = dict()
    scenario['Objective'] = ['CAPEX', 'OPEX']
    scenario['nPareto'] = 2
    scenario['name'] = 'pareto'
    scenario['exclude_units'] = ['NG_Cogeneration', 'OIL_Boiler', 'ThermalSolar']
    scenario['enforce_units'] = []


    # Initialize available units and grids
    grids = infrastructure.initialize_grids()
    units = infrastructure.initialize_units(scenario, grids)

    # Set method options
    method = {'district-scale': True}
    DW_params = {'max_iter': 2}


    # Run optimization
    reho = reho(qbuildings_data=qbuildings_data, units=units, grids=grids, cluster=cluster, s
    reho.generate_pareto_curve()

    # Save results
    reho.save_results(format=['xlsx', 'pickle'], filename='3a')

    # Performance plot : costs and gwp
    plotting.plot_performance(reho.results, plot='costs', indexed_on='Pareto_ID', label='EN_l
    plotting.plot_performance(reho.results, plot='gwp', indexed_on='Pareto_ID', label='EN_lon
    # Sankey diagram
    for key in reho.results['pareto'].keys():
        plotting.plot_sankey(reho.results['pareto'][key], label='EN_long', color='ColorPastel
```

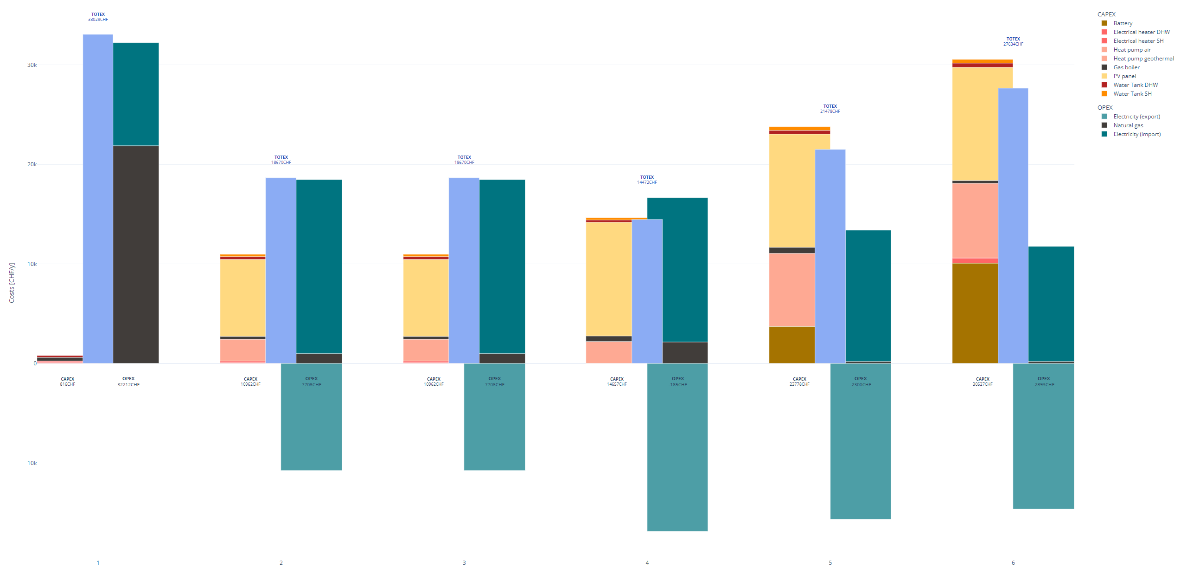


Figure 13: Pareto cost change for district scale

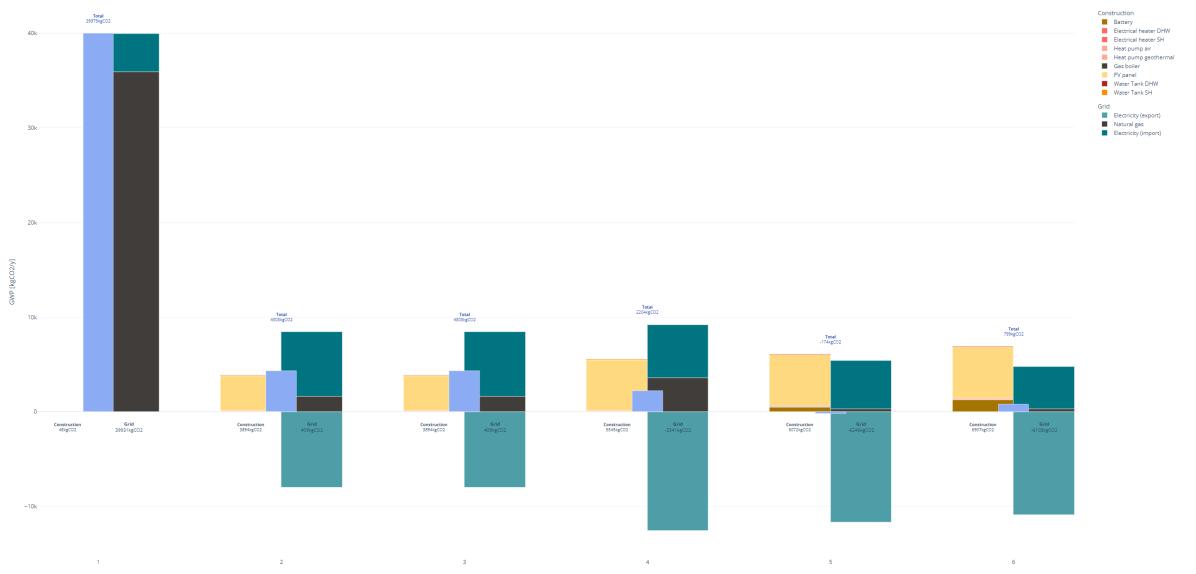


Figure 14: Pareto GWP change for district scale

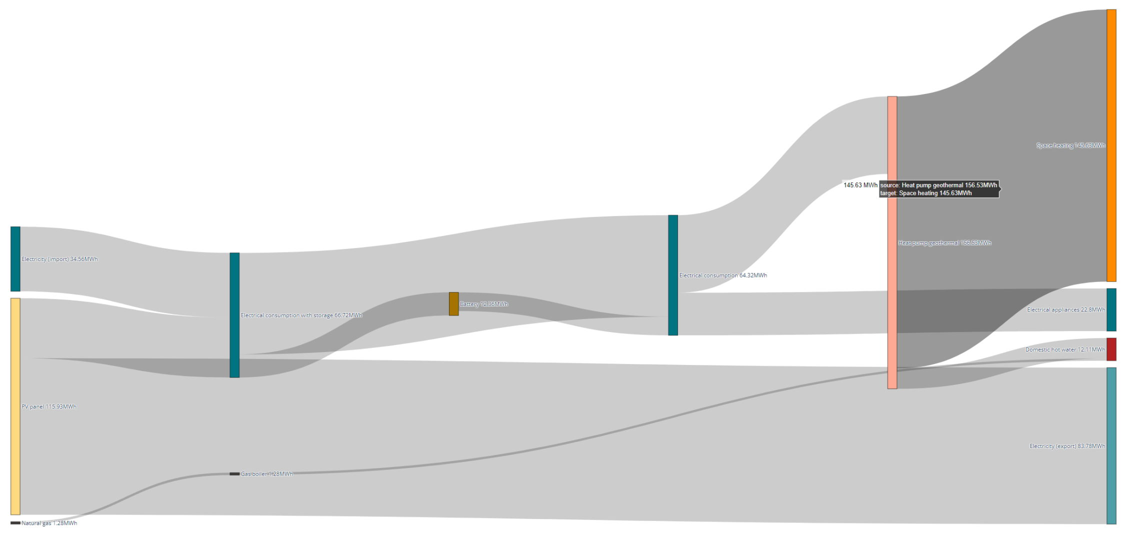

Electricity (import) 34.56MWh
PV panel 115.93MWh
Natural gas 1.28MWh
Electrical consumption with storage 66.72MWh
Gas boiler 1.28MWh
Battery 12.36MWh
Electrical consumption 64.32MWh
145.63 MWh
source: Heat pump geothermal 156.53MWh
target: Space heating 145.63MWh
Heat pump geothermal 156.53MWh
Space heating 145.63MWh
Electrical appliances 22.8MWh
Domestic hot water 12.11MWh
Electricity (export) 83.78MWh

Figure 15: Pareto sankey for district scale

1.3.3 Various features

1.3.3.1 stochastic profiles for EUD

```
# add stochasticity in the demand profiles given by the SIA standards, tunable with:
    # - standard deviation on the peak demand
    # - standard deviation on the time-shift
    method = {'building-scale': True, 'include_stochasticity': True, 'sd_stochasticity': [0.1
```

1.3.3.2 fix unit size

```
# Run new optimization with the capacity of PV and electrical heater fixed with the size of t
    reho.df_fix_Units = reho.results['totex'][0]["df_Unit"]  # load data on the capacity of u
    reho.fix_units_list = ['PV', 'ElectricalHeater_DHW', 'ElectricalHeater_SH']  # select the
    reho.scenario['Objective'] = 'CAPEX'
    reho.scenario['name'] = 'fixed'
    reho.method['fix_units'] = True  # select the method fixing the unit sizes
    reho.single_optimization()
```

1.3.3.3 change heat pump temperature

```
    # Set specific parameters
    parameters = {"T_source": np.repeat({"Geothermal": 17.0}, 2)}

    # Run optimization
    reho = reho(qbuildings_data=qbuildings_data, units=units, grids=grids, parameters=paramet
```

1.3.3.4 Add other energy layers

```
    # Initialize available units and grids
    # You can add more resources layers besides electricity and natural gas, and adapt their
    # or keep the default values from data/parameters/grids.csv
    grids = infrastructure.initialize_grids({'Electricity': {"Cost_supply_cst": 0.30, "Cost_d
                                             'NaturalGas': {"Cost_supply_cst": 0.15},
                                             'Wood': {},
                                             'Oil': {},
                                             })
    path_to_custom_units = '../../reho/data/infrastructure/building_units.csv'
    units = infrastructure.initialize_units(scenario, grids, building_data=path_to_custom_uni
```

1.3.3.5 Include electrical vehicles

```
# Set scenario
scenario = dict()
scenario['Objective'] = 'TOTEX'
scenario['name'] = 'totex'
scenario['exclude_units'] = ['Battery', 'NG_Cogeneration']
scenario['enforce_units'] = ['EV_district']

# Initialize available units and grids
grids = infrastructure.initialize_grids()
units = infrastructure.initialize_units(scenario, grids, district_data=True)

# Set method options
method = {'building-scale': True}

# Set specific parameters
parameters = {'n_vehicles': 6}
```

For other scenarios, directly go the website: https://reho.readthedocs.io/en/main/sections/6_Examples.html.

1.4 Conclusion

1. About how to use REHO In order to use REHO, what I need to do is to

- input relevant building characteristics, weather conditions;
- know about the scenario;
- adjust corresponding parameters and conditions.

2. The difference between Pareto and single optimization is the objectives. (2 or 1)
3. Difference between building levels and district levels:

- building-level: Optimizes by considering than each building is an independent system
- district-level: Optimizes by allowing exchanges between buildings and the use of district units

1.5 Questions

1. Can I choose other objectives except for “TOTEX”, “CAPEX” and “OPEX”?

- YES, like “GWP”

2. Can I have more than 3 objectives?

- YES, here is an example.

```
# Set scenario
scenario = dict()
scenario['Objective'] = ['GWP', 'OPEX', 'CAPEX']
scenario['name'] = 'pareto'
scenario['nPareto'] = 3
scenario['exclude_units'] = ['NG_Cogeneration', 'OIL_Boiler', 'ThermalSolar']
scenario['enforce_units'] = []
```

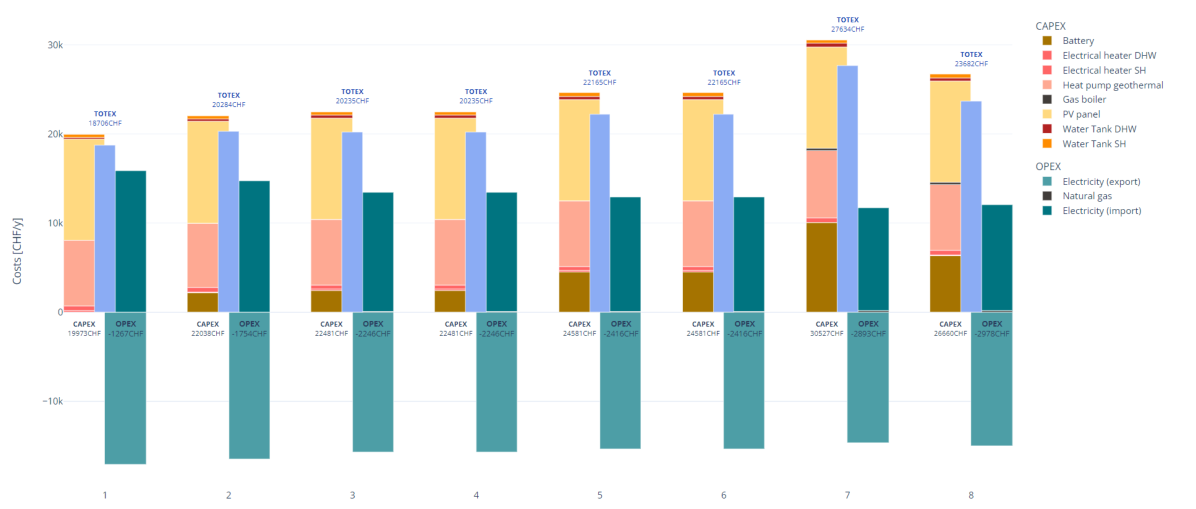


Figure 16: example, [‘GWP’, ‘OPEX’, ‘CAPEX’] cost

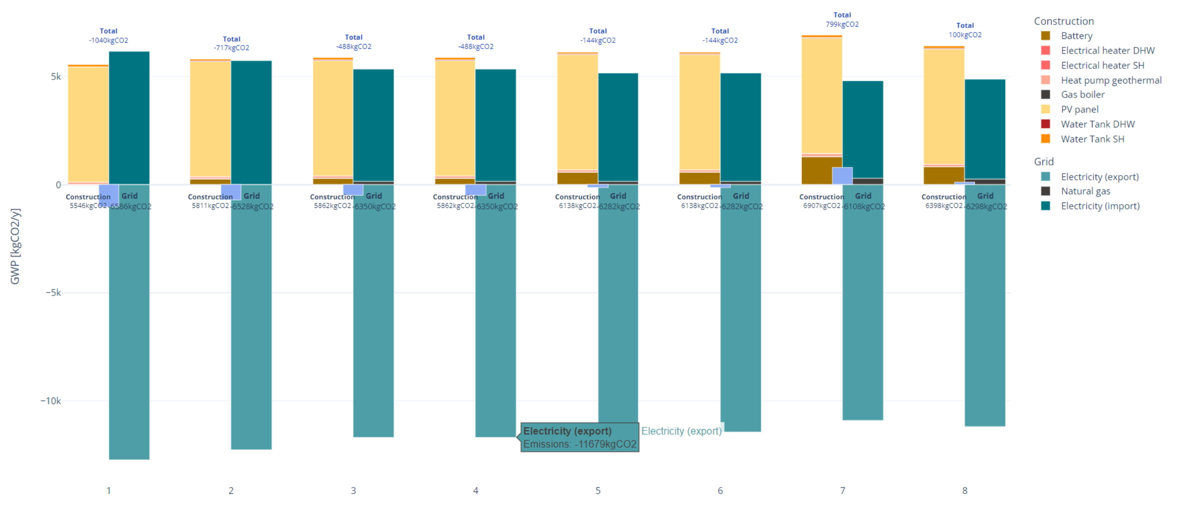


Figure 17: example, [‘GWP’, ‘OPEX’, ‘CAPEX’] GWP

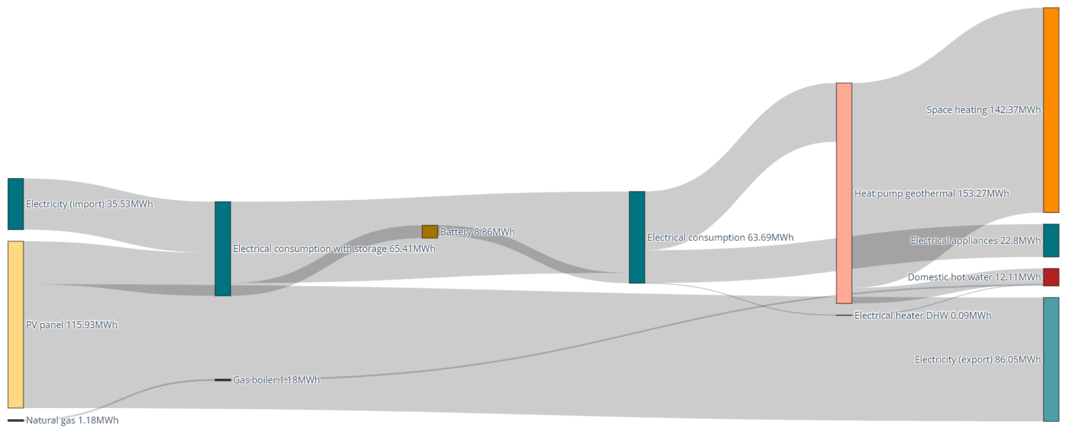

Electricity (import) 35.53MWh
PV panel 115.93MWh
Natural gas 1.18MWh
Electrical consumption with storage 65.41MWh
Gas boiler 1.18MWh
Battery 8.86MWh
Electrical consumption 63.69MWh
Heat pump geothermal 153.27MWh
Electrical heater DHW 0.09MWh
Space heating 142.37MWh
Electrical appliances 22.8MWh
Domestic hot water 12.11MWh
Electricity (export) 86.05MWh

Figure 18: example, [‘GWP’, ‘OPEX’, ‘CAPEX’] Sankey

3. What’s the relationship between the paramters we input for objectives? (eg. What’s the difference between scenario[‘Objective’] = [‘GWP’, ‘OPEX’, ‘CAPEX’] & scenario[‘Objective’] = [‘OPEX’, ‘CAPEX’, ‘GWP’]?)

1.5.1 [‘OPEX’, ‘CAPEX’, ‘GWP’]

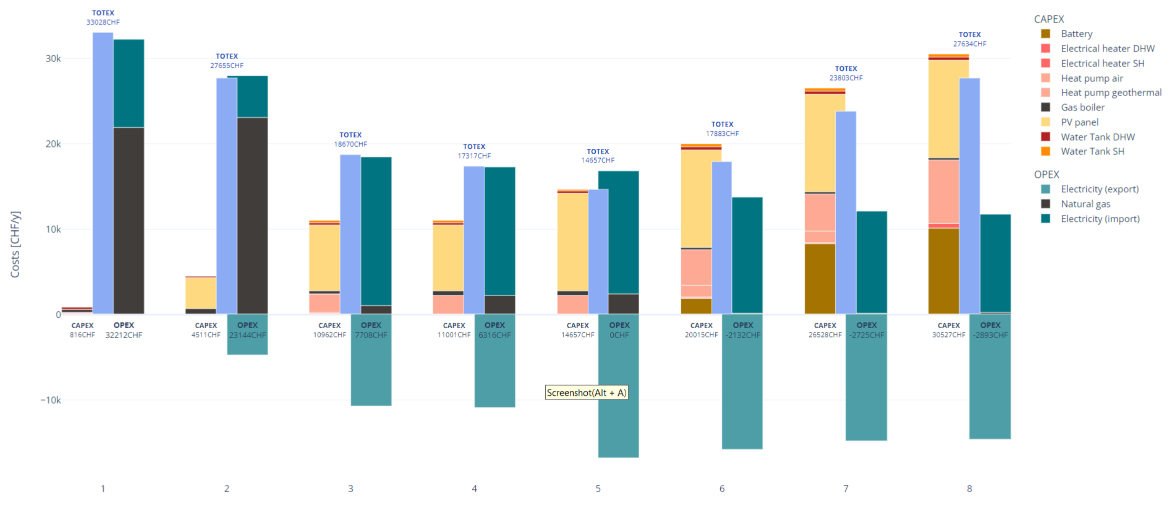


Figure 19: example, [‘OPEX’, ‘CAPEX’, ‘GWP’] cost

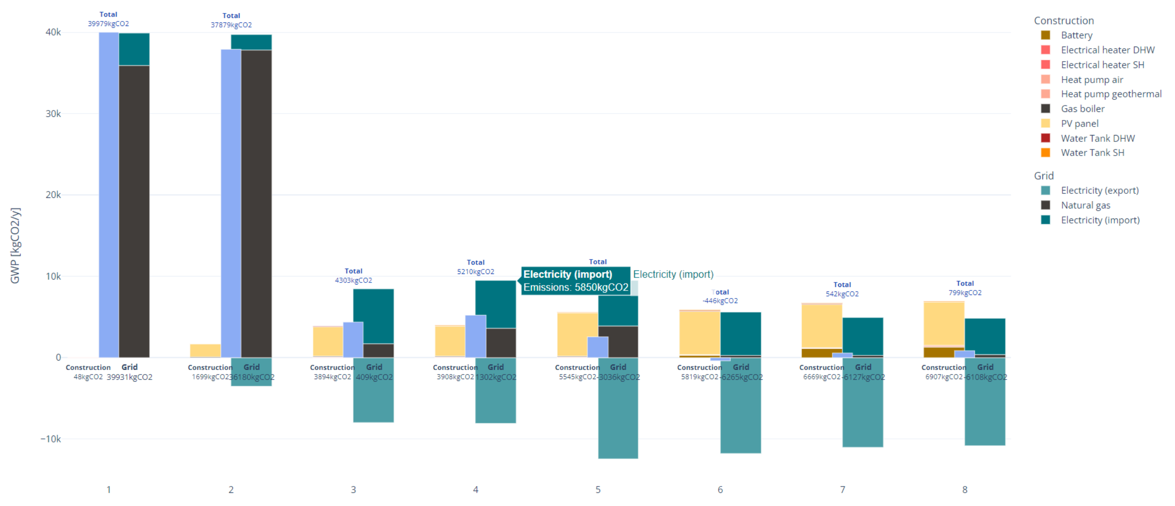


Figure 20: example, [‘OPEX’, ‘CAPEX’, ‘GWP’] GWP

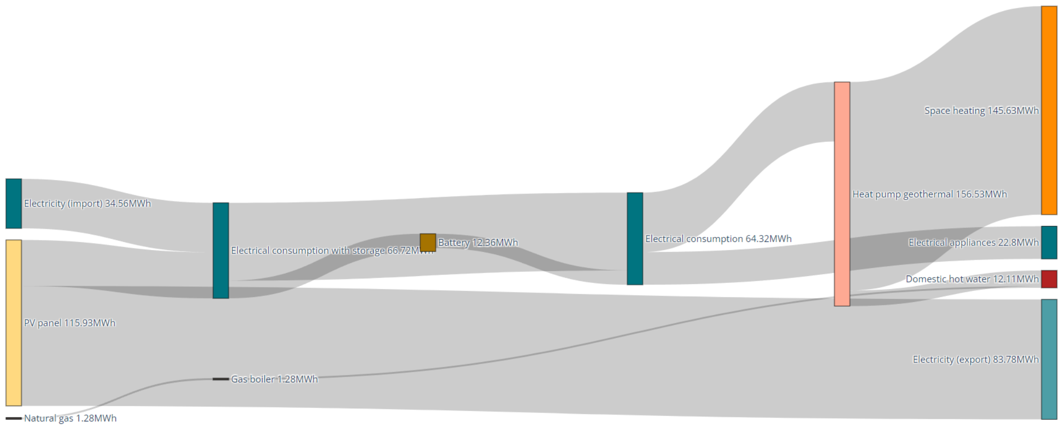


Figure 21: example, [‘OPEX’, ‘CAPEX’, ‘GWP’] Sankey

1.6 What should I do tomorrow?

1. try to create a building csv file which could be treated as an input
2. think of different scenarios and implement REHO.