SP-EVCSAP optimization with processed data

Step 0: Set-up `Python` Env. (with Anaconda)

Find the `py_env_setup` folder in the repository, then:

- For advanced Python programmer:

  requirements.txt is provided to set up the EVCSAP project environment. Copy it in your system user folder (e.g., C:\Users\z004ffpm). The link below provides a way of using this for environment set up

  > https://stackoverflow.com/questions/48787250/set-up-virtualenv-using-a-requirements-txt-generated-by-conda (https://stackoverflow.com/questions/48787250/set-up-virtualenv-using-a-requirements-txt-generated-by-conda)

- For others: (anaconda is required for set up)
  1. Read the **Spec List** or **Environment.yml** section of this blog https://www.anaconda.com/blog/moving-conda-environments (https://www.anaconda.com/blog/moving-conda-environments).
  2. Copy either `EVCSAP_env_list.yml` , or `OptPyomoSP.txt` (depending on which file you use) in your system user folder where the anaconda can find the environment file. (e.g., C:\Users\z004ffpm)

Furthermore:

> - If `geopy` is missing, execute `pip install geopy` in (anaconda) prompt
> - If `MPI-SPPy` is missing, use `conda install openmpi` . Then, `conda install mpi4py` and finally `pip install mpi-sppy` in (anaconda) prompt

In [1]:
```
# Add file path to system, so that csap_packages_sp can be found
import os, sys
currentdir = os.path.dirname(os.path.realpath(''))
parentdir = os.path.dirname(currentdir)
sys.path.append(currentdir)

# Import Local EVCSAP packages
from csap_packages_sp import sp_data_process as Dap
from csap_packages_sp import sp_model_setup_by_fm_data as SupSP
from csap_packages_sp import sp_stat_compu as Stac
from csap_packages_sp import sp_sce_generation as Sceg
from csap_packages_sp.sp_mpd_frame_model_setup import _build_mpdp_csap_frame

# Import Pyomo utils
import idaes
from mpisppy.opt.ef import ExtensiveForm
from pyomo.core.expr.current import evaluate_expression
import pyomo.environ as pyo
from pyomo.opt import SolverFactory

# Import Opensource packages
import numpy as np
import pandas as pd
import geopandas as gpd
from pyproj import CRS
from shapely.geometry import Point, MultiPoint
import matplotlib.pyplot as plt
from tqdm import tqdm
from cProfile import label
import plotly.express as px
import time
import pickle
import seaborn
from datetime import datetime
```

```
[    0.00] Initializing mpi-sppy
```

MPDP by data retrieved from frame_model

Load Data of Frame Model and MPDP Grid Connection Scenario

In [5]:
```
currentdir
```

Out[5]:
```
'C:\\Users\\z004ffpm\\Work_Documents\\CSallocModel\\MasterThesis_Stochastic_EVCSAP_Gen_LI'
```

In [6]:
```
mpdp_con_path = r'\Data\mpdp_grid_con_scenario.pickle'
mpdp_con_sce = pd.read_pickle(currentdir+mpdp_con_path)
data_by_frame_model_path = r'\Data\data_of_mpdp_frame.pickle'
frame_model_data = pd.read_pickle(currentdir+data_by_frame_model_path)

```

In [9]:
```
mpdp_by_frame_data = SupSP._build_mpsp_csap_from_mpdp_frame(
    mpdp_frame_data = frame_model_data, # test_results_dict[f'phi_IJ_{i}/60']['solved_model_data'],
    cs_ss_connect_sce = mpdp_con_sce.stack().to_dict()
)
```

```
Done! set up MPSP_CSAP took 1.85 seconds.
```

The Solution to `mpdp_by_frame_data` will be exactly the same as that of `mpdp_frame_model`.

In [10]:
```
# Define solver
Solver = SolverFactory(
#     'glpk'
    'cplex'
    , tee = True,
)
```

In [11]:
```
solver_results_fast_mpdp = Solver.solve(mpdp_by_frame_data)
csap_is_solved = True
print("Solver Message:", solver_results_fast_mpdp)
```

```
Solver Message:
Problem:
- Name: tmp806oluch
  Lower bound: 1599429.4024634599
  Upper bound: 1599429.4024634599
  Number of objectives: 1
  Number of constraints: 3486
  Number of variables: 6842
  Number of nonzeros: 14704
  Sense: maximize
Solver:
- Status: ok
  User time: 0.13
  Termination condition: optimal
  Termination message: MIP - Integer optimal solution\x3a Objective = 1.5994294025e+06
  Statistics:
    Branch and bound:
      Number of bounded subproblems: 0
      Number of created subproblems: 0
  Error rc: 0
  Time: 0.34859704971313477
Solution:
- number of solutions: 0
  number of solutions displayed: 0
```

In [12]:
```
print_decision = True
if print_decision:
    Stac._print_decision(mpdp_by_frame_data)
#     _print_decision(mpdp_frame_model_4_sp)
```

```
Decision to build new CSs,
(loc_id, Nr_CP):
[(36, 2.0), (48, 10.0), (58, 8.0), (61, 8.0), (96, 8.0), (110, 7.0), (115, 12.0), (178, 25.0)]

Decision to update old CSs,
(loc_id, Nr_CP, Nr_total_CPs):
[]
Decision to expand SSs,
(loc_id, size_expansion):
[(6, 200.0), (8, 300.0), (74, 100.0)]

Decision to use expensive backstop tech at SSs,
((loc_id, period), amount_backstop usage):
[]
```

Set-up and solve MPSP (An example with 10 Scenarios)

In [15]:
```
con_sces_path = r'\Data\\10_con_sces_dict.pickle'
all_connection_sces_dict = pd.read_pickle(currentdir+con_sces_path)
data_by_frame_model_path = r'\Data\data_of_mpdp_frame.pickle'
frame_model_data = pd.read_pickle(currentdir+data_by_frame_model_path)
```

In [16]:
```
# 0. Define solver options:
num_threads = 2
solver_options = {"solver": "cplex",
    "threads": num_threads, # Define the max. number of CPU threads used. MPI-SPPy suggested small number, such as 2.
    "warmstart": False
}
```

Set up Extensive Form (EF)

through `csap_scenario_creator`, `data_mpdp_frame` (Generated in **Step 3.2**) `all_connection_sces_dict` with 10 scenarios (Created in **Step 2**).

In [17]:
```
scenario_names = list(all_connection_sces_dict.keys()) # `all_connection_sces_dict` Created in Step 2
tic = time.perf_counter()
print(f"""Building Stochastic Model with {len(all_connection_sces_dict)} Scenarios.
    You will see {len(all_connection_sces_dict)} times the same model set-up message. \n""")
MPSP_ef = ExtensiveForm (options = solver_options ,
    all_scenario_names = scenario_names,
    scenario_creator = Sceg.csap_scenario_creator,
    scenario_creator_kwargs = {
        "mpdp_frame_data": frame_model_data, # Generated in Step 3.2
        "all_sces_dict": all_connection_sces_dict, # Created in Step 2
        }
    )
toc = time.perf_counter()
print(f"Succeed! SP Model set-up took {round(toc - tic,2)} seconds in total.\n")
# # 2. Pass EF to the solver to solve :
# solver_results = MPSP_ef . solve_extensive_form ()
```

```
Building Stochastic Model with 10 Scenarios.
    You will see 10 times the same model set-up message.

[  171.67] Initializing SPBase
Done! set up EVCSAP_MPSP took 1.52 seconds.
Done! set up EVCSAP_MPSP took 1.14 seconds.
Done! set up EVCSAP_MPSP took 1.26 seconds.
Done! set up EVCSAP_MPSP took 1.27 seconds.
Done! set up EVCSAP_MPSP took 1.2 seconds.
Done! set up EVCSAP_MPSP took 1.33 seconds.
Done! set up EVCSAP_MPSP took 1.49 seconds.
Done! set up EVCSAP_MPSP took 1.03 seconds.
Done! set up EVCSAP_MPSP took 1.06 seconds.
Done! set up EVCSAP_MPSP took 1.44 seconds.
Succeed! SP Model set-up took 13.4 seconds in total.
```

Step 5.2 Solve EF

In [18]:
```
print('Solving SP-EVCSAP')
tic = time.perf_counter()
solver_results = MPSP_ef.solve_extensive_form()
toc = time.perf_counter()
print(f"  Done! It took {round(toc - tic, 2)} seconds to solve the extensive form of SP. \n")
# print(solver_results)

```

```
Solving SP-EVCSAP
  Done! It took 13.15 seconds to solve the extensive form of SP.
```

Retrieve data of solved MPSP and show results

In [19]:
```
test_results = Stac._get_data_from_solved_MPSP(
    ef_solver_results = solver_results,
    solved_MPSP_extensive_form = MPSP_ef,
    num_scenarios = 10,
    ids_scenarios = scenario_names,
    )
```

In [20]:
```
test_results.keys()
```

Out[20]:
```
dict_keys(['num_scenarios', 'ids_scenarios', 'objective_value', 'prob_description', 'solver_info', 'gap', 'mpsp_csap_decisions', 'subsce_1_data'])
```

In [21]:
```
pd.Series(test_results)
```

Out[21]:
```
num_scenarios                                                  10
ids_scenarios                             [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
objective_value                                       1280203.519633
prob_description       {'Lower bound': 1280203.5196326366, 'Upper bou...
solver_info            {'User time': 0.66, 'Termination message': 'MI...
gap                                                              0.0
mpsp_csap_decisions    {'x': {'48': 1.0, '58': 1.0, '61': 1.0, '96': ...
subsce_1_data          {'obj_total_profit': 1316703.5196326352, 'cs_p...
dtype: object
```

In [22]:
```
test_results['prob_description']
```

Out[22]:
```
{'Lower bound': 1280203.5196326366,
 'Upper bound': 1280203.5196326366,
 'Number of constraints': 96015,
 'Number of variables': 68411}
```

In [23]:
```
test_results['solver_info']
```

Out[23]:
```
{'User time': 0.66,
 'Termination message': 'MIP - Integer optimal solution\\x3a Objective = 1.2802035196e+06',
 'Statistics': {'Branch and bound': {'Number of bounded subproblems': 13, 'Number of created subproblems': 13}, 'Black box': {}},
 'Time': 1.3849174976348877}
```

In [24]:

```python
for deci_var, dv_values in test_results['mpsp_csap_decisions'].items():
    if deci_var != 'z':
        print(f"{deci_var}: {dv_values}")
    else:
        print(f"z:\n {pd.Series(dv_values)}")
```

```
x: {'48': 1.0, '58': 1.0, '61': 1.0, '96': 1.0, '110': 1.0, '114': 1.0, '115': 1.0, '178': 0.9999999999999996}
y: {'48': 10.0, '58': 8.0, '61': 7.000000000000014, '96': 8.0, '110': 8.0, '114': 16.0, '115': 12.0, '178': 10.999999999999995}
h: {'0': 5.0, '2': 0.9999999999999997, '6': 0.9999999999999997, '8': 5.0, '10': 3.000000000000001, '13': 1.0, '92': 1.0}
z:
 48,15,day_normal      0.113531
48,15,night           1.000000
48,46,day_normal      0.485299
48,46,day_peak        0.041068
48,46,night           0.096736
                        ...
195,111,day_peak      0.097585
195,144,night         0.014131
195,160,night         0.180419
195,176,day_peak      0.078758
195,182,day_normal    0.136105
Length: 272, dtype: float64
```

END