In [1]:
```
from IPython.display import display
import pandas as pd
pd.options.display.max_rows = None
```

In [2]:
```
import utils
from solver import Solver, SolverConfig
```

In [3]:
```
SolverConfig()
```

Out[3]:
```
SolverConfig(greenw=0.2, costw=0.2, productivityw=0.2, delayw=0.2, success_ratew=0.2)
```

Check marginal values

In [4]:
```
problem = utils.read_problem()
print(problem.marginal_productivities.mean())
print(problem.marginal_delays.mean())
print(problem.marginal_success_rates.mean())
print(problem.marginal_costs.mean() )
print(problem.marginal_green_capacities.mean())
```

```
0.83011824
10.957512
0.95351195
1.0526167
1.3453703
```

Test Integer Version

In [9]:
```
# Heuristic
heuristic_solver = Solver(
    problem,
    SolverConfig(
        greenw=0.0,
        costw=0.0,
        productivityw=1.0,
        delayw=0.0,
        success_ratew=0.0
    )
```

```
)
heuristic_solution = heuristic_solver.run_capacity_constrained()

# Multistart
multistart_solver = Solver(
    problem,
    SolverConfig(
        greenw=0.0,
        costw=0.0,
        productivityw=1.0,
        delayw=0.0,
        success_ratew=0.0
    )
)
multistart_solution = multistart_solver.multistart(maxiter=3000)

display(utils.solution_to_dataframe(multistart_solution))
```

postcode	Fleet1	Fleet2	Fleet3	Fleet4	Fleet5	Fleet6	Fleet7	Fleet8	Fleet9
1	0	0	0	1	0	0	0	0	0
2	0	1	0	0	0	0	0	0	0
3	0	0	0	1	0	0	0	0	0
4	1	0	0	0	0	0	0	0	0
5	0	1	0	0	0	0	0	0	0
6	0	0	0	1	0	0	0	0	0
7	1	0	0	0	0	0	0	0	0
8	1	0	0	0	0	0	0	0	0
9	0	0	1	0	0	0	0	0	0
10	0	1	0	0	0	0	0	0	0
11	0	1	0	0	0	0	0	0	0
12	0	1	0	0	0	0	0	0	0
13	1	0	0	0	0	0	0	0	0
14	0	0	0	0	1	0	0	0	0
15	0	0	0	0	1	0	0	0	0
16	0	0	0	0	1	0	0	0	0
17	1	0	0	0	0	0	0	0	0

18	0	1	0	0	0	0	0	0	0
19	0	0	0	0	1	0	0	0	0
20	0	0	1	0	0	0	0	0	0
21	0	0	0	0	0	0	1	0	0
22	1	0	0	0	0	0	0	0	0
23	0	1	0	0	0	0	0	0	0
24	0	1	0	0	0	0	0	0	0
25	0	1	0	0	0	0	0	0	0
26	1	0	0	0	0	0	0	0	0
27	0	0	0	1	0	0	0	0	0
28	0	1	0	0	0	0	0	0	0
29	0	1	0	0	0	0	0	0	0
30	0	0	1	0	0	0	0	0	0
31	0	0	0	1	0	0	0	0	0
32	0	0	1	0	0	0	0	0	0
33	0	0	0	0	0	0	1	0	0
34	0	0	0	0	0	0	1	0	0
35	0	0	0	0	1	0	0	0	0
36	0	1	0	0	0	0	0	0	0
37	0	0	0	1	0	0	0	0	0
38	1	0	0	0	0	0	0	0	0
39	0	0	0	0	1	0	0	0	0
40	0	1	0	0	0	0	0	0	0
41	1	0	0	0	0	0	0	0	0
42	1	0	0	0	0	0	0	0	0
43	1	0	0	0	0	0	0	0	0
44	1	0	0	0	0	0	0	0	0
45	0	1	0	0	0	0	0	0	0
46	1	0	0	0	0	0	0	0	0
47	0	0	0	0	1	0	0	0	0

48	0	1	0	0	0	0	0	0	0
49	0	0	0	0	0	0	1	0	0
50	1	0	0	0	0	0	0	0	0
51	0	0	0	0	0	0	1	0	0
52	0	1	0	0	0	0	0	0	0
53	0	0	1	0	0	0	0	0	0
54	1	0	0	0	0	0	0	0	0
55	0	0	0	0	0	0	1	0	0
56	0	1	0	0	0	0	0	0	0
57	0	1	0	0	0	0	0	0	0
58	1	0	0	0	0	0	0	0	0
59	0	0	0	0	0	0	1	0	0
60	0	1	0	0	0	0	0	0	0
61	0	0	0	0	0	0	1	0	0
62	1	0	0	0	0	0	0	0	0
63	0	1	0	0	0	0	0	0	0
64	0	1	0	0	0	0	0	0	0
65	0	0	0	1	0	0	0	0	0
66	0	0	1	0	0	0	0	0	0
67	1	0	0	0	0	0	0	0	0
68	0	1	0	0	0	0	0	0	0
69	0	0	0	0	0	0	1	0	0
70	0	1	0	0	0	0	0	0	0
71	0	1	0	0	0	0	0	0	0
72	0	1	0	0	0	0	0	0	0

In [10]:

```
# Heuristic
heuristic_solver = Solver(
    problem,
    SolverConfig(
        greenw=0.0,
        costw=1.0,
```

```
        productivityw=0.0,
        delayw=0.0,
        success_ratew=0.0
    )
)
heuristic_solution = heuristic_solver.run_capacity_constrained()

# Multistart
multistart_solver = Solver(
    problem,
    SolverConfig(
        greenw=0.0,
        costw=1.0,
        productivityw=0.0,
        delayw=0.0,
        success_ratew=0.0
    )
)
multistart_solution = multistart_solver.multistart(maxiter=3000)

display(utils.solution_to_dataframe(multistart_solution))
```

postcode	Fleet1	Fleet2	Fleet3	Fleet4	Fleet5	Fleet6	Fleet7	Fleet8	Fleet9
1	0	1	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0	0
3	1	0	0	0	0	0	0	0	0
4	1	0	0	0	0	0	0	0	0
5	1	0	0	0	0	0	0	0	0
6	1	0	0	0	0	0	0	0	0
7	1	0	0	0	0	0	0	0	0
8	1	0	0	0	0	0	0	0	0
9	1	0	0	0	0	0	0	0	0
10	1	0	0	0	0	0	0	0	0
11	1	0	0	0	0	0	0	0	0
12	1	0	0	0	0	0	0	0	0
13	1	0	0	0	0	0	0	0	0
14	1	0	0	0	0	0	0	0	0

15	1	0	0	0	0	0	0	0	0
16	1	0	0	0	0	0	0	0	0
17	0	1	0	0	0	0	0	0	0
18	0	1	0	0	0	0	0	0	0
19	1	0	0	0	0	0	0	0	0
20	1	0	0	0	0	0	0	0	0
21	1	0	0	0	0	0	0	0	0
22	1	0	0	0	0	0	0	0	0
23	1	0	0	0	0	0	0	0	0
24	1	0	0	0	0	0	0	0	0
25	0	1	0	0	0	0	0	0	0
26	1	0	0	0	0	0	0	0	0
27	0	1	0	0	0	0	0	0	0
28	0	1	0	0	0	0	0	0	0
29	0	1	0	0	0	0	0	0	0
30	0	1	0	0	0	0	0	0	0
31	0	1	0	0	0	0	0	0	0
32	0	1	0	0	0	0	0	0	0
33	0	1	0	0	0	0	0	0	0
34	0	1	0	0	0	0	0	0	0
35	0	1	0	0	0	0	0	0	0
36	0	1	0	0	0	0	0	0	0
37	0	1	0	0	0	0	0	0	0
38	0	1	0	0	0	0	0	0	0
39	0	1	0	0	0	0	0	0	0
40	0	1	0	0	0	0	0	0	0
41	0	1	0	0	0	0	0	0	0
42	0	1	0	0	0	0	0	0	0
43	0	1	0	0	0	0	0	0	0
44	0	1	0	0	0	0	0	0	0

45	1	0	0	0	0	0	0	0	0
46	0	1	0	0	0	0	0	0	0
47	0	1	0	0	0	0	0	0	0
48	0	1	0	0	0	0	0	0	0
49	0	1	0	0	0	0	0	0	0
50	0	1	0	0	0	0	0	0	0
51	0	1	0	0	0	0	0	0	0
52	0	1	0	0	0	0	0	0	0
53	0	1	0	0	0	0	0	0	0
54	0	1	0	0	0	0	0	0	0
55	0	1	0	0	0	0	0	0	0
56	0	1	0	0	0	0	0	0	0
57	0	1	0	0	0	0	0	0	0
58	0	1	0	0	0	0	0	0	0
59	0	1	0	0	0	0	0	0	0
60	0	1	0	0	0	0	0	0	0
61	0	1	0	0	0	0	0	0	0
62	0	1	0	0	0	0	0	0	0
63	0	0	0	0	1	0	0	0	0
64	0	0	1	0	0	0	0	0	0
65	0	1	0	0	0	0	0	0	0
66	0	1	0	0	0	0	0	0	0
67	0	0	0	0	1	0	0	0	0
68	0	0	0	0	0	0	0	0	0
69	0	1	0	0	0	0	0	0	0
70	0	1	0	0	0	0	0	0	0
71	0	0	1	0	0	0	0	0	0
72	0	0	0	0	0	0	0	0	0

In [11]:

```
# Heuristic
heuristic_solver = Solver(
```

```
    problem,
    SolverConfig(
        greenw=0.24375,
        costw=0.24375,
        productivityw=0.24375,
        delayw=0.025,
        success_ratew=0.24375
    )
)
heuristic_solution = heuristic_solver.run_capacity_constrained()

# Multistart
multistart_solver = Solver(
    problem,
    SolverConfig(
        greenw=0.24375,
        costw=0.24375,
        productivityw=0.24375,
        delayw=0.025,
        success_ratew=0.24375
    )
)
multistart_solution = multistart_solver.multistart(maxiter=3000)

display(utils.solution_to_dataframe(multistart_solution))
```

postcode	Fleet1	Fleet2	Fleet3	Fleet4	Fleet5	Fleet6	Fleet7	Fleet8	Fleet9
1	0	0	0	1	0	0	0	0	0
2	0	1	0	0	0	0	0	0	0
3	1	0	0	0	0	0	0	0	0
4	1	0	0	0	0	0	0	0	0
5	0	1	0	0	0	0	0	0	0
6	0	0	0	1	0	0	0	0	0
7	1	0	0	0	0	0	0	0	0
8	1	0	0	0	0	0	0	0	0
9	0	0	1	0	0	0	0	0	0
10	1	0	0	0	0	0	0	0	0
11	0	1	0	0	0	0	0	0	0
12	0	1	0	0	0	0	0	0	0

13	1	0	0	0	0	0	0	0	0
14	0	0	0	0	1	0	0	0	0
15	0	0	0	0	1	0	0	0	0
16	0	0	0	0	1	0	0	0	0
17	1	0	0	0	0	0	0	0	0
18	0	1	0	0	0	0	0	0	0
19	0	0	0	0	1	0	0	0	0
20	0	0	1	0	0	0	0	0	0
21	0	0	0	0	0	0	1	0	0
22	1	0	0	0	0	0	0	0	0
23	0	1	0	0	0	0	0	0	0
24	0	1	0	0	0	0	0	0	0
25	0	1	0	0	0	0	0	0	0
26	0	0	0	0	0	1	0	0	0
27	0	0	0	1	0	0	0	0	0
28	0	1	0	0	0	0	0	0	0
29	0	1	0	0	0	0	0	0	0
30	0	0	1	0	0	0	0	0	0
31	0	0	0	1	0	0	0	0	0
32	0	0	0	0	0	1	0	0	0
33	0	0	0	0	0	0	1	0	0
34	1	0	0	0	0	0	0	0	0
35	0	0	0	0	1	0	0	0	0
36	0	0	0	0	0	1	0	0	0
37	0	0	0	0	0	1	0	0	0
38	0	0	0	0	0	1	0	0	0
39	0	1	0	0	0	0	0	0	0
40	0	1	0	0	0	0	0	0	0
41	0	0	0	0	0	1	0	0	0
42	1	0	0	0	0	0	0	0	0

43	1	0	0	0	0	0	0	0	0
44	1	0	0	0	0	0	0	0	0
45	0	1	0	0	0	0	0	0	0
46	1	0	0	0	0	0	0	0	0
47	0	1	0	0	0	0	0	0	0
48	0	1	0	0	0	0	0	0	0
49	0	0	0	0	0	1	0	0	0
50	1	0	0	0	0	0	0	0	0
51	0	0	0	0	0	1	0	0	0
52	0	0	0	0	0	1	0	0	0
53	0	0	0	0	0	1	0	0	0
54	1	0	0	0	0	0	0	0	0
55	0	0	0	0	0	1	0	0	0
56	0	1	0	0	0	0	0	0	0
57	0	1	0	0	0	0	0	0	0
58	1	0	0	0	0	0	0	0	0
59	0	0	0	0	0	1	0	0	0
60	0	1	0	0	0	0	0	0	0
61	0	0	0	0	0	1	0	0	0
62	1	0	0	0	0	0	0	0	0
63	0	1	0	0	0	0	0	0	0
64	0	0	0	0	0	1	0	0	0
65	1	0	0	0	0	0	0	0	0
66	1	0	0	0	0	0	0	0	0
67	1	0	0	0	0	0	0	0	0
68	0	1	0	0	0	0	0	0	0
69	0	0	0	0	0	0	1	0	0
70	0	1	0	0	0	0	0	0	0
71	0	0	0	0	0	1	0	0	0
72	0	1	0	0	0	0	0	0	0

Test Smooth Version

In [4]:

```
# Heuristic
heuristic_solver = Solver(
    problem,
    SolverConfig(
        greenw=0.0,
        costw=1.0,
        productivityw=0.0,
        delayw=0.0,
        success_ratew=0.0
    )
)
heuristic_solution = heuristic_solver.run_volume_constrained()

display(utils.solution_to_dataframe(heuristic_solution))
```

postcode	Fleet1	Fleet2	Fleet3	Fleet4	Fleet5	Fleet6	Fleet7	Fleet8	Fleet9
1	0.00	1.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	1.00	0.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	1.00	0.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	1.00	0.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5	1.00	0.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6	1.00	0.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0
7	1.00	0.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0
8	1.00	0.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0
9	1.00	0.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0
10	1.00	0.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0
11	0.07	0.93	0.0	0.0	0.0	0.0	0.0	0.0	0.0
12	0.00	1.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0
13	0.00	1.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0
14	0.00	1.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0
15	0.00	1.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0
16	0.00	1.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0

17	0.00	1.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0
18	0.00	1.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0
19	0.00	1.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0
20	0.00	1.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0
21	0.00	1.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0
22	0.00	1.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0
23	0.00	1.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0
24	0.00	1.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0
25	0.00	1.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0
26	0.00	1.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0
27	0.00	1.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0
28	0.00	1.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0
29	0.00	1.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0
30	0.00	1.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0
31	0.00	1.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0
32	0.00	1.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0
33	0.00	1.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0
34	0.00	1.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0
35	0.00	1.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0
36	0.00	1.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0
37	0.00	1.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0
38	0.00	1.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0
39	0.00	1.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0
40	0.00	1.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0
41	0.00	1.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0
42	0.00	1.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0
43	0.00	1.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0
44	0.00	1.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0
45	0.00	1.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0
46	0.00	1.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0

47	0.00	1.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0
48	0.00	0.40	0.0	0.0	0.6	0.0	0.0	0.0	0.0
49	0.00	0.00	0.0	0.0	1.0	0.0	0.0	0.0	0.0
50	0.00	0.00	1.0	0.0	0.0	0.0	0.0	0.0	0.0
51	0.00	0.00	0.0	0.0	1.0	0.0	0.0	0.0	0.0
52	0.00	0.00	1.0	0.0	0.0	0.0	0.0	0.0	0.0
53	0.00	0.00	1.0	0.0	0.0	0.0	0.0	0.0	0.0
54	0.00	0.00	1.0	0.0	0.0	0.0	0.0	0.0	0.0
55	0.00	0.00	1.0	0.0	0.0	0.0	0.0	0.0	0.0
56	0.00	0.00	0.0	0.0	1.0	0.0	0.0	0.0	0.0
57	0.00	0.00	1.0	0.0	0.0	0.0	0.0	0.0	0.0
58	0.00	0.00	1.0	0.0	0.0	0.0	0.0	0.0	0.0
59	0.00	0.00	1.0	0.0	0.0	0.0	0.0	0.0	0.0
60	0.00	0.00	1.0	0.0	0.0	0.0	0.0	0.0	0.0
61	0.00	0.00	0.0	0.0	1.0	0.0	0.0	0.0	0.0
62	0.00	0.00	0.0	0.0	1.0	0.0	0.0	0.0	0.0
63	0.00	0.00	0.0	0.0	1.0	0.0	0.0	0.0	0.0
64	0.00	0.00	1.0	0.0	0.0	0.0	0.0	0.0	0.0
65	0.00	0.00	0.0	0.0	1.0	0.0	0.0	0.0	0.0
66	0.00	0.00	1.0	0.0	0.0	0.0	0.0	0.0	0.0
67	0.00	0.00	0.0	0.0	1.0	0.0	0.0	0.0	0.0
68	0.00	0.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0
69	0.00	0.00	1.0	0.0	0.0	0.0	0.0	0.0	0.0
70	0.00	0.00	1.0	0.0	0.0	0.0	0.0	0.0	0.0
71	0.00	0.00	1.0	0.0	0.0	0.0	0.0	0.0	0.0
72	0.00	0.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0

In [5]:

```
# Heuristic
heuristic_solver = Solver(
    problem,
    SolverConfig(
        greenw=1.0,
```

```
        costw=0.0,
        productivityw=0.0,
        delayw=0.0,
        success_ratew=0.0
    )
)
heuristic_solution = heuristic_solver.run_volume_constrained()

display(utils.solution_to_dataframe(heuristic_solution))
```

postcode	Fleet1	Fleet2	Fleet3	Fleet4	Fleet5	Fleet6	Fleet7	Fleet8	Fleet9
1	0.00	1.00	0.00	0.0	0.0	0.00	0.0	0.0	0.0
2	0.00	0.00	0.00	0.0	0.0	1.00	0.0	0.0	0.0
3	1.00	0.00	0.00	0.0	0.0	0.00	0.0	0.0	0.0
4	1.00	0.00	0.00	0.0	0.0	0.00	0.0	0.0	0.0
5	1.00	0.00	0.00	0.0	0.0	0.00	0.0	0.0	0.0
6	0.00	0.00	0.00	0.0	0.0	1.00	0.0	0.0	0.0
7	1.00	0.00	0.00	0.0	0.0	0.00	0.0	0.0	0.0
8	1.00	0.00	0.00	0.0	0.0	0.00	0.0	0.0	0.0
9	1.00	0.00	0.00	0.0	0.0	0.00	0.0	0.0	0.0
10	1.00	0.00	0.00	0.0	0.0	0.00	0.0	0.0	0.0
11	1.00	0.00	0.00	0.0	0.0	0.00	0.0	0.0	0.0
12	1.00	0.00	0.00	0.0	0.0	0.00	0.0	0.0	0.0
13	0.38	0.62	0.00	0.0	0.0	0.00	0.0	0.0	0.0
14	0.00	1.00	0.00	0.0	0.0	0.00	0.0	0.0	0.0
15	0.00	1.00	0.00	0.0	0.0	0.00	0.0	0.0	0.0
16	0.00	1.00	0.00	0.0	0.0	0.00	0.0	0.0	0.0
17	0.00	1.00	0.00	0.0	0.0	0.00	0.0	0.0	0.0
18	0.00	1.00	0.00	0.0	0.0	0.00	0.0	0.0	0.0
19	0.00	1.00	0.00	0.0	0.0	0.00	0.0	0.0	0.0
20	0.00	1.00	0.00	0.0	0.0	0.00	0.0	0.0	0.0
21	0.00	1.00	0.00	0.0	0.0	0.00	0.0	0.0	0.0
22	0.00	1.00	0.00	0.0	0.0	0.00	0.0	0.0	0.0

23	0.00	1.00	0.00	0.0	0.0	0.00	0.0	0.0	0.0
24	0.00	1.00	0.00	0.0	0.0	0.00	0.0	0.0	0.0
25	0.00	1.00	0.00	0.0	0.0	0.00	0.0	0.0	0.0
26	0.00	0.00	0.00	0.0	0.0	1.00	0.0	0.0	0.0
27	0.00	1.00	0.00	0.0	0.0	0.00	0.0	0.0	0.0
28	0.00	1.00	0.00	0.0	0.0	0.00	0.0	0.0	0.0
29	0.00	1.00	0.00	0.0	0.0	0.00	0.0	0.0	0.0
30	0.00	1.00	0.00	0.0	0.0	0.00	0.0	0.0	0.0
31	0.00	1.00	0.00	0.0	0.0	0.00	0.0	0.0	0.0
32	0.00	0.00	0.00	0.0	0.0	1.00	0.0	0.0	0.0
33	0.00	1.00	0.00	0.0	0.0	0.00	0.0	0.0	0.0
34	0.00	0.00	0.00	0.0	0.0	1.00	0.0	0.0	0.0
35	0.00	1.00	0.00	0.0	0.0	0.00	0.0	0.0	0.0
36	0.00	0.00	0.00	0.0	0.0	1.00	0.0	0.0	0.0
37	0.00	0.00	0.00	0.0	0.0	1.00	0.0	0.0	0.0
38	0.00	0.00	0.00	0.0	0.0	1.00	0.0	0.0	0.0
39	0.00	1.00	0.00	0.0	0.0	0.00	0.0	0.0	0.0
40	0.00	1.00	0.00	0.0	0.0	0.00	0.0	0.0	0.0
41	0.00	0.00	0.00	0.0	0.0	1.00	0.0	0.0	0.0
42	0.00	1.00	0.00	0.0	0.0	0.00	0.0	0.0	0.0
43	0.00	1.00	0.00	0.0	0.0	0.00	0.0	0.0	0.0
44	0.00	1.00	0.00	0.0	0.0	0.00	0.0	0.0	0.0
45	0.00	1.00	0.00	0.0	0.0	0.00	0.0	0.0	0.0
46	0.00	1.00	0.00	0.0	0.0	0.00	0.0	0.0	0.0
47	0.00	1.00	0.00	0.0	0.0	0.00	0.0	0.0	0.0
48	0.00	1.00	0.00	0.0	0.0	0.00	0.0	0.0	0.0
49	0.00	0.87	0.00	0.0	0.0	0.13	0.0	0.0	0.0
50	0.00	1.00	0.00	0.0	0.0	0.00	0.0	0.0	0.0
51	0.00	1.00	0.00	0.0	0.0	0.00	0.0	0.0	0.0
52	0.00	1.00	0.00	0.0	0.0	0.00	0.0	0.0	0.0

53	0.00	1.00	0.00	0.0	0.0	0.00	0.0	0.0	0.0
54	0.00	1.00	0.00	0.0	0.0	0.00	0.0	0.0	0.0
55	0.00	1.00	0.00	0.0	0.0	0.00	0.0	0.0	0.0
56	0.00	1.00	0.00	0.0	0.0	0.00	0.0	0.0	0.0
57	0.00	1.00	0.00	0.0	0.0	0.00	0.0	0.0	0.0
58	0.00	1.00	0.00	0.0	0.0	0.00	0.0	0.0	0.0
59	0.00	1.00	0.00	0.0	0.0	0.00	0.0	0.0	0.0
60	0.00	1.00	0.00	0.0	0.0	0.00	0.0	0.0	0.0
61	0.00	1.00	0.00	0.0	0.0	0.00	0.0	0.0	0.0
62	0.00	1.00	0.00	0.0	0.0	0.00	0.0	0.0	0.0
63	0.00	1.00	0.00	0.0	0.0	0.00	0.0	0.0	0.0
64	0.00	1.00	0.00	0.0	0.0	0.00	0.0	0.0	0.0
65	0.00	1.00	0.00	0.0	0.0	0.00	0.0	0.0	0.0
66	0.00	0.88	0.12	0.0	0.0	0.00	0.0	0.0	0.0
67	0.00	0.00	0.00	1.0	0.0	0.00	0.0	0.0	0.0
68	0.00	0.00	0.00	0.0	0.0	0.00	0.0	0.0	0.0
69	0.00	0.00	1.00	0.0	0.0	0.00	0.0	0.0	0.0
70	0.00	0.00	1.00	0.0	0.0	0.00	0.0	0.0	0.0
71	0.00	0.00	1.00	0.0	0.0	0.00	0.0	0.0	0.0
72	0.00	0.00	0.00	0.0	0.0	0.00	0.0	0.0	0.0

In [6]:

```
# Heuristic
heuristic_solver = Solver(
    problem,
    SolverConfig(
        greenw=0.24375,
        costw=0.24375,
        productivityw=0.24375,
        delayw=0.025,
        success_ratew=0.24375
    )
)
heuristic_solution = heuristic_solver.run_volume_constrained()

display(utils.solution_to_dataframe(heuristic_solution))
```

postcode	Fleet1	Fleet2	Fleet3	Fleet4	Fleet5	Fleet6	Fleet7	Fleet8	Fleet9
1	0.0	0.0	0.0	1.0	0.0	0.00	0.00	0.0	0.0
2	0.0	1.0	0.0	0.0	0.0	0.00	0.00	0.0	0.0
3	1.0	0.0	0.0	0.0	0.0	0.00	0.00	0.0	0.0
4	1.0	0.0	0.0	0.0	0.0	0.00	0.00	0.0	0.0
5	0.0	1.0	0.0	0.0	0.0	0.00	0.00	0.0	0.0
6	0.0	0.0	0.0	1.0	0.0	0.00	0.00	0.0	0.0
7	1.0	0.0	0.0	0.0	0.0	0.00	0.00	0.0	0.0
8	1.0	0.0	0.0	0.0	0.0	0.00	0.00	0.0	0.0
9	0.0	0.0	1.0	0.0	0.0	0.00	0.00	0.0	0.0
10	1.0	0.0	0.0	0.0	0.0	0.00	0.00	0.0	0.0
11	0.0	1.0	0.0	0.0	0.0	0.00	0.00	0.0	0.0
12	0.0	1.0	0.0	0.0	0.0	0.00	0.00	0.0	0.0
13	1.0	0.0	0.0	0.0	0.0	0.00	0.00	0.0	0.0
14	0.0	0.0	0.0	0.0	1.0	0.00	0.00	0.0	0.0
15	0.0	0.0	0.0	0.0	1.0	0.00	0.00	0.0	0.0
16	0.0	0.0	0.0	0.0	1.0	0.00	0.00	0.0	0.0
17	1.0	0.0	0.0	0.0	0.0	0.00	0.00	0.0	0.0
18	0.0	1.0	0.0	0.0	0.0	0.00	0.00	0.0	0.0
19	0.0	0.0	0.0	0.0	1.0	0.00	0.00	0.0	0.0
20	0.0	0.0	1.0	0.0	0.0	0.00	0.00	0.0	0.0
21	0.0	0.0	0.0	0.0	0.0	0.00	1.00	0.0	0.0
22	1.0	0.0	0.0	0.0	0.0	0.00	0.00	0.0	0.0
23	0.0	1.0	0.0	0.0	0.0	0.00	0.00	0.0	0.0
24	0.0	1.0	0.0	0.0	0.0	0.00	0.00	0.0	0.0
25	0.0	1.0	0.0	0.0	0.0	0.00	0.00	0.0	0.0
26	0.0	0.0	0.0	0.0	0.0	1.00	0.00	0.0	0.0
27	0.0	0.0	0.0	1.0	0.0	0.00	0.00	0.0	0.0
28	0.0	1.0	0.0	0.0	0.0	0.00	0.00	0.0	0.0

29	0.0	1.0	0.0	0.0	0.0	0.00	0.00	0.0	0.0
30	0.0	0.0	1.0	0.0	0.0	0.00	0.00	0.0	0.0
31	0.0	0.0	0.0	1.0	0.0	0.00	0.00	0.0	0.0
32	0.0	0.0	0.0	0.0	0.0	1.00	0.00	0.0	0.0
33	0.0	0.0	0.0	0.0	0.0	0.00	1.00	0.0	0.0
34	0.0	0.0	0.0	0.0	0.0	1.00	0.00	0.0	0.0
35	0.0	0.0	0.0	0.0	1.0	0.00	0.00	0.0	0.0
36	0.0	1.0	0.0	0.0	0.0	0.00	0.00	0.0	0.0
37	0.0	0.0	0.0	0.0	0.0	1.00	0.00	0.0	0.0
38	0.0	0.0	0.0	0.0	0.0	1.00	0.00	0.0	0.0
39	0.0	1.0	0.0	0.0	0.0	0.00	0.00	0.0	0.0
40	0.0	1.0	0.0	0.0	0.0	0.00	0.00	0.0	0.0
41	0.0	0.0	0.0	0.0	0.0	1.00	0.00	0.0	0.0
42	1.0	0.0	0.0	0.0	0.0	0.00	0.00	0.0	0.0
43	1.0	0.0	0.0	0.0	0.0	0.00	0.00	0.0	0.0
44	1.0	0.0	0.0	0.0	0.0	0.00	0.00	0.0	0.0
45	0.0	1.0	0.0	0.0	0.0	0.00	0.00	0.0	0.0
46	1.0	0.0	0.0	0.0	0.0	0.00	0.00	0.0	0.0
47	0.0	1.0	0.0	0.0	0.0	0.00	0.00	0.0	0.0
48	0.0	1.0	0.0	0.0	0.0	0.00	0.00	0.0	0.0
49	0.0	0.0	0.0	0.0	0.0	0.58	0.42	0.0	0.0
50	1.0	0.0	0.0	0.0	0.0	0.00	0.00	0.0	0.0
51	0.0	0.0	0.0	0.0	0.0	0.00	1.00	0.0	0.0
52	0.0	0.0	0.0	0.0	0.0	1.00	0.00	0.0	0.0
53	0.0	0.0	1.0	0.0	0.0	0.00	0.00	0.0	0.0
54	1.0	0.0	0.0	0.0	0.0	0.00	0.00	0.0	0.0
55	0.0	0.0	0.0	0.0	0.0	0.00	1.00	0.0	0.0
56	0.0	1.0	0.0	0.0	0.0	0.00	0.00	0.0	0.0
57	0.0	1.0	0.0	0.0	0.0	0.00	0.00	0.0	0.0
58	1.0	0.0	0.0	0.0	0.0	0.00	0.00	0.0	0.0

59	0.0	0.0	0.0	0.0	0.0	0.00	1.00	0.0	0.0
60	0.0	1.0	0.0	0.0	0.0	0.00	0.00	0.0	0.0
61	0.0	0.0	0.0	0.0	0.0	1.00	0.00	0.0	0.0
62	1.0	0.0	0.0	0.0	0.0	0.00	0.00	0.0	0.0
63	0.0	1.0	0.0	0.0	0.0	0.00	0.00	0.0	0.0
64	0.0	0.0	0.0	0.0	0.0	1.00	0.00	0.0	0.0
65	1.0	0.0	0.0	0.0	0.0	0.00	0.00	0.0	0.0
66	1.0	0.0	0.0	0.0	0.0	0.00	0.00	0.0	0.0
67	1.0	0.0	0.0	0.0	0.0	0.00	0.00	0.0	0.0
68	0.0	1.0	0.0	0.0	0.0	0.00	0.00	0.0	0.0
69	0.0	0.0	0.0	0.0	0.0	0.00	1.00	0.0	0.0
70	0.0	1.0	0.0	0.0	0.0	0.00	0.00	0.0	0.0
71	0.0	1.0	0.0	0.0	0.0	0.00	0.00	0.0	0.0
72	0.0	1.0	0.0	0.0	0.0	0.00	0.00	0.0	0.0

Test Smooth Version with MinVolume Constraints

In [5]:

```
# Heuristic
heuristic_solver = Solver(
    problem,
    SolverConfig(
        greenw=1.0,
        costw=0.0,
        productivityw=0.0,
        delayw=0.0,
        success_ratew=0.0
    )
)
heuristic_solution = heuristic_solver.run_volume_constrained()

display(utils.solution_to_dataframe(heuristic_solution))

for fleet in problem.fleets:
    print(f"Min Volume Capacity : {fleet.minvolume} - Assigned Volume : {sum(i[1] for i in fleet.customers)}")
```

	Fleet1	Fleet2	Fleet3	Fleet4	Fleet5	Fleet6	Fleet7	Fleet8	Fleet9

postcode									
1	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.0	0.0
2	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.0	0.0
3	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.0	0.0
4	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.0	0.0
5	0.29	0.71	0.00	0.00	0.00	0.00	0.00	0.0	0.0
6	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.0	0.0
7	0.00	0.77	0.00	0.23	0.00	0.00	0.00	0.0	0.0
8	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.0	0.0
9	0.00	0.00	0.91	0.09	0.00	0.00	0.00	0.0	0.0
10	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.0	0.0
11	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.0	0.0
12	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.0	0.0
13	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.0	0.0
14	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.0	0.0
15	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.0	0.0
16	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.0	0.0
17	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.0	0.0
18	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.0	0.0
19	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.0	0.0
20	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.0	0.0
21	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.0	0.0
22	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.0	0.0
23	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.0	0.0
24	0.00	0.00	0.00	0.00	0.09	0.00	0.91	0.0	0.0
25	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.0	0.0
26	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.0	0.0
27	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.0	0.0
28	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.0	0.0
29	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.0	0.0

30	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.0	0.0
31	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.0	0.0
32	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.0	0.0
33	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.0	0.0
34	0.00	0.00	0.00	0.00	0.00	0.92	0.08	0.0	0.0
35	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.0	0.0
36	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.0	0.0
37	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.0	0.0
38	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.0	0.0
39	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.0	0.0
40	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.0	0.0
41	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.0	0.0
42	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.0	0.0
43	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.0	0.0
44	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.0	0.0
45	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.0	0.0
46	0.67	0.00	0.00	0.33	0.00	0.00	0.00	0.0	0.0
47	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.0	0.0
48	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.0	0.0
49	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.0	0.0
50	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.0	0.0
51	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.0	0.0
52	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.0	0.0
53	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.0	0.0
54	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.0	0.0
55	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.0	0.0
56	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.0	0.0
57	0.42	0.00	0.00	0.00	0.00	0.58	0.00	0.0	0.0
58	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.0	0.0
59	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.0	0.0

60	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.0	0.0
61	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.0	0.0
62	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.0	0.0
63	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.0	0.0
64	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.0	0.0
65	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.0	0.0
66	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.0	0.0
67	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.0	0.0
68	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.0	0.0
69	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.0	0.0
70	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.0	0.0
71	0.90	0.10	0.00	0.00	0.00	0.00	0.00	0.0	0.0
72	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.0	0.0

```
Min Volume Capacity : 8000.0 - Assigned Volume : 40000.0
Min Volume Capacity : 16000.0 - Assigned Volume : 16574.0
Min Volume Capacity : 11200.0 - Assigned Volume : 11200.0
Min Volume Capacity : 11200.0 - Assigned Volume : 11200.0
Min Volume Capacity : 16000.0 - Assigned Volume : 16000.0
Min Volume Capacity : 11200.0 - Assigned Volume : 24000.0
Min Volume Capacity : 16000.0 - Assigned Volume : 16000.0
Min Volume Capacity : 0.0 - Assigned Volume : 0
Min Volume Capacity : 0.0 - Assigned Volume : 0
```

In [6]:

```
# Heuristic
heuristic_solver = Solver(
    problem,
    SolverConfig(
        greenw=.0,
        costw=0.0,
        productivityw=1.0,
        delayw=0.0,
        success_ratew=0.0
    )
)
heuristic_solution = heuristic_solver.run_volume_constrained()

display(utils.solution_to_dataframe(heuristic_solution))
for fleet in problem.fleets:
    print(f"Min Volume Capacity : {fleet.minvolume} - Assigned Volume : {sum(i[1] for i in fleet.customers)}")
```

postcode	Fleet1	Fleet2	Fleet3	Fleet4	Fleet5	Fleet6	Fleet7	Fleet8	Fleet9
1	0.00	0.0	0.00	1.00	0.00	0.00	0.00	0.0	0.0
2	0.00	1.0	0.00	0.00	0.00	0.00	0.00	0.0	0.0
3	0.00	0.0	0.00	1.00	0.00	0.00	0.00	0.0	0.0
4	0.00	0.0	0.00	0.00	1.00	0.00	0.00	0.0	0.0
5	0.00	1.0	0.00	0.00	0.00	0.00	0.00	0.0	0.0
6	0.00	0.0	0.00	1.00	0.00	0.00	0.00	0.0	0.0
7	0.00	0.0	0.00	1.00	0.00	0.00	0.00	0.0	0.0
8	1.00	0.0	0.00	0.00	0.00	0.00	0.00	0.0	0.0
9	0.00	0.0	1.00	0.00	0.00	0.00	0.00	0.0	0.0
10	0.00	1.0	0.00	0.00	0.00	0.00	0.00	0.0	0.0
11	0.00	0.0	0.00	0.00	0.00	0.00	1.00	0.0	0.0
12	0.00	1.0	0.00	0.00	0.00	0.00	0.00	0.0	0.0
13	0.99	0.0	0.00	0.00	0.00	0.00	0.01	0.0	0.0
14	0.00	0.0	0.00	0.00	1.00	0.00	0.00	0.0	0.0
15	0.00	0.0	0.00	0.00	1.00	0.00	0.00	0.0	0.0
16	0.00	0.0	0.00	0.00	1.00	0.00	0.00	0.0	0.0
17	1.00	0.0	0.00	0.00	0.00	0.00	0.00	0.0	0.0
18	0.00	0.0	0.00	0.00	1.00	0.00	0.00	0.0	0.0
19	0.00	0.0	0.00	0.00	1.00	0.00	0.00	0.0	0.0
20	0.00	0.0	1.00	0.00	0.00	0.00	0.00	0.0	0.0
21	0.00	0.0	0.00	0.00	0.00	0.00	1.00	0.0	0.0
22	0.00	0.0	0.00	0.00	0.00	0.00	1.00	0.0	0.0
23	0.00	0.0	0.00	0.00	1.00	0.00	0.00	0.0	0.0
24	0.00	1.0	0.00	0.00	0.00	0.00	0.00	0.0	0.0
25	0.00	1.0	0.00	0.00	0.00	0.00	0.00	0.0	0.0
26	1.00	0.0	0.00	0.00	0.00	0.00	0.00	0.0	0.0
27	0.00	0.0	0.00	1.00	0.00	0.00	0.00	0.0	0.0
28	0.00	1.0	0.00	0.00	0.00	0.00	0.00	0.0	0.0

29	0.00	1.0	0.00	0.00	0.00	0.00	0.00	0.0	0.0
30	0.00	0.0	1.00	0.00	0.00	0.00	0.00	0.0	0.0
31	0.00	0.0	0.00	1.00	0.00	0.00	0.00	0.0	0.0
32	0.00	0.0	1.00	0.00	0.00	0.00	0.00	0.0	0.0
33	0.00	0.0	0.00	0.00	0.00	0.00	1.00	0.0	0.0
34	0.00	0.0	0.34	0.00	0.00	0.66	0.00	0.0	0.0
35	0.00	0.0	0.00	0.00	1.00	0.00	0.00	0.0	0.0
36	0.00	0.0	0.00	0.71	0.29	0.00	0.00	0.0	0.0
37	0.00	0.0	0.00	1.00	0.00	0.00	0.00	0.0	0.0
38	0.00	0.0	0.00	0.00	1.00	0.00	0.00	0.0	0.0
39	0.00	0.0	0.00	0.00	1.00	0.00	0.00	0.0	0.0
40	0.00	1.0	0.00	0.00	0.00	0.00	0.00	0.0	0.0
41	0.00	0.0	0.00	0.00	0.10	0.41	0.49	0.0	0.0
42	0.00	0.0	0.00	0.00	0.00	0.00	1.00	0.0	0.0
43	1.00	0.0	0.00	0.00	0.00	0.00	0.00	0.0	0.0
44	1.00	0.0	0.00	0.00	0.00	0.00	0.00	0.0	0.0
45	0.00	1.0	0.00	0.00	0.00	0.00	0.00	0.0	0.0
46	1.00	0.0	0.00	0.00	0.00	0.00	0.00	0.0	0.0
47	0.00	0.0	0.00	0.00	1.00	0.00	0.00	0.0	0.0
48	0.00	1.0	0.00	0.00	0.00	0.00	0.00	0.0	0.0
49	0.00	0.0	0.00	0.00	0.00	0.00	1.00	0.0	0.0
50	1.00	0.0	0.00	0.00	0.00	0.00	0.00	0.0	0.0
51	0.00	0.0	0.00	0.00	0.00	0.00	1.00	0.0	0.0
52	0.00	1.0	0.00	0.00	0.00	0.00	0.00	0.0	0.0
53	0.00	0.0	1.00	0.00	0.00	0.00	0.00	0.0	0.0
54	0.00	0.0	0.00	0.00	1.00	0.00	0.00	0.0	0.0
55	0.00	0.0	0.00	0.00	0.00	0.00	1.00	0.0	0.0
56	0.00	1.0	0.00	0.00	0.00	0.00	0.00	0.0	0.0
57	0.00	1.0	0.00	0.00	0.00	0.00	0.00	0.0	0.0
58	1.00	0.0	0.00	0.00	0.00	0.00	0.00	0.0	0.0

59	0.00	0.0	0.00	0.00	0.00	0.00	1.00	0.0	0.0
60	0.00	1.0	0.00	0.00	0.00	0.00	0.00	0.0	0.0
61	0.00	0.0	0.00	0.00	1.00	0.00	0.00	0.0	0.0
62	1.00	0.0	0.00	0.00	0.00	0.00	0.00	0.0	0.0
63	0.00	0.0	0.00	0.00	0.00	0.00	1.00	0.0	0.0
64	0.00	0.0	0.00	0.00	0.00	1.00	0.00	0.0	0.0
65	0.00	0.0	0.00	1.00	0.00	0.00	0.00	0.0	0.0
66	0.00	0.0	1.00	0.00	0.00	0.00	0.00	0.0	0.0
67	0.00	0.0	0.00	0.00	0.00	0.00	1.00	0.0	0.0
68	0.00	1.0	0.00	0.00	0.00	0.00	0.00	0.0	0.0
69	0.00	0.0	0.00	0.00	0.00	0.00	1.00	0.0	0.0
70	0.00	1.0	0.00	0.00	0.00	0.00	0.00	0.0	0.0
71	0.00	1.0	0.00	0.00	0.00	0.00	0.00	0.0	0.0
72	0.00	1.0	0.00	0.00	0.00	0.00	0.00	0.0	0.0

```
Min Volume Capacity : 8000.0 - Assigned Volume : 18253.0
Min Volume Capacity : 16000.0 - Assigned Volume : 44746.0
Min Volume Capacity : 11200.0 - Assigned Volume : 13347.0
Min Volume Capacity : 11200.0 - Assigned Volume : 15428.0
Min Volume Capacity : 16000.0 - Assigned Volume : 16000.0
Min Volume Capacity : 11200.0 - Assigned Volume : 11200.0
Min Volume Capacity : 16000.0 - Assigned Volume : 16000.0
Min Volume Capacity : 0.0 - Assigned Volume : 0
Min Volume Capacity : 0.0 - Assigned Volume : 0
```

In [7]:

```
# Heuristic
heuristic_solver = Solver(
    problem,
    SolverConfig(
        greenw=0.24375,
        costw=0.24375,
        productivityw=0.24375,
        delayw=0.025,
        success_ratew=0.24375
    )
)
heuristic_solution = heuristic_solver.run_volume_constrained()

display(utils.solution_to_dataframe(heuristic_solution))
```

```
for fleet in problem.fleets:
    print(f"Min Volume Capacity : {fleet.minvolume} - Assigned Volume : {sum(i[1] for i in fleet.customers)}")
```

postcode	Fleet1	Fleet2	Fleet3	Fleet4	Fleet5	Fleet6	Fleet7	Fleet8	Fleet9
1	0.0	0.00	0.00	1.00	0.00	0.00	0.00	0.0	0.0
2	0.0	1.00	0.00	0.00	0.00	0.00	0.00	0.0	0.0
3	1.0	0.00	0.00	0.00	0.00	0.00	0.00	0.0	0.0
4	0.0	0.00	0.00	0.00	1.00	0.00	0.00	0.0	0.0
5	0.0	1.00	0.00	0.00	0.00	0.00	0.00	0.0	0.0
6	0.0	0.00	0.00	1.00	0.00	0.00	0.00	0.0	0.0
7	0.0	0.00	0.00	1.00	0.00	0.00	0.00	0.0	0.0
8	1.0	0.00	0.00	0.00	0.00	0.00	0.00	0.0	0.0
9	0.0	0.00	1.00	0.00	0.00	0.00	0.00	0.0	0.0
10	1.0	0.00	0.00	0.00	0.00	0.00	0.00	0.0	0.0
11	0.0	0.00	0.00	0.00	0.00	0.00	1.00	0.0	0.0
12	0.0	1.00	0.00	0.00	0.00	0.00	0.00	0.0	0.0
13	1.0	0.00	0.00	0.00	0.00	0.00	0.00	0.0	0.0
14	0.0	0.00	0.00	0.00	1.00	0.00	0.00	0.0	0.0
15	0.0	0.00	0.00	0.00	1.00	0.00	0.00	0.0	0.0
16	0.0	0.00	0.00	0.00	1.00	0.00	0.00	0.0	0.0
17	1.0	0.00	0.00	0.00	0.00	0.00	0.00	0.0	0.0
18	0.0	0.21	0.00	0.00	0.79	0.00	0.00	0.0	0.0
19	0.0	0.00	0.00	0.00	1.00	0.00	0.00	0.0	0.0
20	0.0	0.00	1.00	0.00	0.00	0.00	0.00	0.0	0.0
21	0.0	0.00	0.00	0.00	0.00	0.00	1.00	0.0	0.0
22	0.0	0.00	0.00	0.00	1.00	0.00	0.00	0.0	0.0
23	0.0	1.00	0.00	0.00	0.00	0.00	0.00	0.0	0.0
24	0.0	1.00	0.00	0.00	0.00	0.00	0.00	0.0	0.0
25	0.0	1.00	0.00	0.00	0.00	0.00	0.00	0.0	0.0
26	0.0	0.00	0.00	0.00	0.00	1.00	0.00	0.0	0.0

27	0.0	0.00	0.00	1.00	0.00	0.00	0.00	0.0	0.0
28	0.0	1.00	0.00	0.00	0.00	0.00	0.00	0.0	0.0
29	0.0	0.58	0.00	0.00	0.00	0.00	0.42	0.0	0.0
30	0.0	0.00	1.00	0.00	0.00	0.00	0.00	0.0	0.0
31	0.0	0.00	0.00	1.00	0.00	0.00	0.00	0.0	0.0
32	0.0	0.00	1.00	0.00	0.00	0.00	0.00	0.0	0.0
33	0.0	0.00	0.00	0.00	0.00	0.00	1.00	0.0	0.0
34	0.0	0.00	0.34	0.00	0.00	0.66	0.00	0.0	0.0
35	0.0	0.00	0.00	0.00	1.00	0.00	0.00	0.0	0.0
36	0.0	0.00	0.00	0.71	0.29	0.00	0.00	0.0	0.0
37	0.0	0.00	0.00	1.00	0.00	0.00	0.00	0.0	0.0
38	0.0	0.00	0.00	0.00	1.00	0.00	0.00	0.0	0.0
39	0.0	0.00	0.00	0.00	1.00	0.00	0.00	0.0	0.0
40	0.0	1.00	0.00	0.00	0.00	0.00	0.00	0.0	0.0
41	0.0	0.00	0.00	0.00	0.00	1.00	0.00	0.0	0.0
42	0.0	0.00	0.00	0.00	0.00	0.00	1.00	0.0	0.0
43	1.0	0.00	0.00	0.00	0.00	0.00	0.00	0.0	0.0
44	1.0	0.00	0.00	0.00	0.00	0.00	0.00	0.0	0.0
45	0.0	1.00	0.00	0.00	0.00	0.00	0.00	0.0	0.0
46	0.0	0.00	0.00	0.00	1.00	0.00	0.00	0.0	0.0
47	0.0	0.00	0.00	0.00	1.00	0.00	0.00	0.0	0.0
48	0.0	1.00	0.00	0.00	0.00	0.00	0.00	0.0	0.0
49	0.0	0.00	0.00	0.00	0.00	0.00	1.00	0.0	0.0
50	0.0	0.00	0.00	0.00	0.00	0.00	1.00	0.0	0.0
51	0.0	0.00	0.00	0.00	0.00	0.00	1.00	0.0	0.0
52	0.0	0.00	0.00	0.00	0.00	1.00	0.00	0.0	0.0
53	0.0	0.00	1.00	0.00	0.00	0.00	0.00	0.0	0.0
54	0.0	0.00	0.00	0.00	1.00	0.00	0.00	0.0	0.0
55	0.0	0.00	0.00	0.00	0.18	0.00	0.82	0.0	0.0
56	0.0	1.00	0.00	0.00	0.00	0.00	0.00	0.0	0.0

57	0.0	1.00	0.00	0.00	0.00	0.00	0.00	0.0	0.0
58	1.0	0.00	0.00	0.00	0.00	0.00	0.00	0.0	0.0
59	0.0	0.00	0.00	0.00	0.00	0.00	1.00	0.0	0.0
60	0.0	1.00	0.00	0.00	0.00	0.00	0.00	0.0	0.0
61	0.0	0.00	0.00	0.00	1.00	0.00	0.00	0.0	0.0
62	1.0	0.00	0.00	0.00	0.00	0.00	0.00	0.0	0.0
63	0.0	0.00	0.00	0.00	0.00	0.00	1.00	0.0	0.0
64	0.0	0.00	0.00	0.00	0.00	0.63	0.37	0.0	0.0
65	0.0	0.00	0.00	1.00	0.00	0.00	0.00	0.0	0.0
66	0.0	0.00	1.00	0.00	0.00	0.00	0.00	0.0	0.0
67	0.0	0.00	0.00	0.00	1.00	0.00	0.00	0.0	0.0
68	0.0	1.00	0.00	0.00	0.00	0.00	0.00	0.0	0.0
69	0.0	0.00	0.00	0.00	0.00	0.00	1.00	0.0	0.0
70	0.0	1.00	0.00	0.00	0.00	0.00	0.00	0.0	0.0
71	0.0	0.00	0.00	0.00	0.00	1.00	0.00	0.0	0.0
72	0.0	1.00	0.00	0.00	0.00	0.00	0.00	0.0	0.0

```
Min Volume Capacity : 8000.0 - Assigned Volume : 23732.0
Min Volume Capacity : 16000.0 - Assigned Volume : 34591.0
Min Volume Capacity : 11200.0 - Assigned Volume : 13347.0
Min Volume Capacity : 11200.0 - Assigned Volume : 13992.0
Min Volume Capacity : 16000.0 - Assigned Volume : 16000.0
Min Volume Capacity : 11200.0 - Assigned Volume : 17312.0
Min Volume Capacity : 16000.0 - Assigned Volume : 16000.0
Min Volume Capacity : 0.0 - Assigned Volume : 0
Min Volume Capacity : 0.0 - Assigned Volume : 0
```

In [ ]: