ps2_final

November 8, 2020

0.1 Daniel Alonso, UID: 100444499

0.2 Problem 1-a

Let i be the source of a movement and j be the destination

Let x be movements from one node to another, where $x_{i,j}$ represents the movement from node i to node j

Integer optimization formulation:

minimize $44x_{1,2} + 18x_{1,3} + 85x_{1,4} + 17x_{2,5} + 13x_{2,6} + 55x_{3,5} + 53x_{3,6} + 7x_{3,7} + 26x_{4,6} + 66x_{4,7} + 6x_{5,8} + 7x_{5,9} + 26x_{6,8} + 66x_{6,9} + 26x_{6,10} + 6x_{7,9} + 60x_{7,10} + 58x_{8,11} + 70x_{9,11} + 27x_{10,11}$

subject to:

$x_{1,2} + x_{1,3} + x_{1,4} = 1$

$x_{1,2} - x_{2,5} - x_{2,6} = 0$

$x_{1,3} - x_{3,5} - x_{3,6} - x_{3,7} = 0$

$x_{1,4} - x_{4,6} - x_{4,7} = 0$

$x_{2,5} + x_{3,5} - x_{5,8} - x_{5,9} = 0$

$x_{2,6} + x_{3,6} + x_{4,6} - x_{6,8} - x_{6,9} - x_{6,10} = 0$

$x_{3,7} + x_{4,7} - x_{7,9} - x_{7,10} = 0$

$x_{5,8} + x_{6,8} - x_{8,11} = 0$

$x_{5,9} + x_{6,9} + x_{7,9} - x_{9,11} = 0$

$x_{6,10} + x_{7,10} - x_{10,11} = 0$

$-x_{8,11} - x_{9,11} - x_{10,11} = -1$

The integer optimization formulation (in my opinion) is more appropriate. It not only will provide the optimal integer solution but it also describes the problem better and more accurately.

Using ordinary linear optimization forces us to estimate values and would make the process more convoluted even though the integer optimization problem could, perhaps be more computationally challenging depending on the size of the graph.

0.3 Problem 1-b

[160]:
```
# Gurobi-python implementation
from gurobipy import *
import pandas as pd
import seaborn as sns
from collections import defaultdict
import numpy as np
import matplotlib.pyplot as plt
```

[161]:
```
nodes, supply = multidict({
         1: 1,
         2: 0,
         3: 0,
         4: 0,
         5: 0,
         6: 0,
         7: 0,
         8: 0,
         9: 0,
         10: 0,
         11: -1})

arcs, distance = multidict({
    (1, 2): 44,
    (1, 3): 18,
    (1, 4): 85,
    (2, 5): 17,
    (2, 6): 13,
    (3, 5): 55,
    (3, 6): 53,
    (3, 7): 7,
    (4, 6): 26,
    (4, 7): 66,
    (5, 8): 6,
    (5, 9): 7,
    (6, 8): 26,
    (6, 9): 66,
    (6, 10): 26,
    (7, 9): 6,
    (7, 10): 60,
    (8, 11): 58,
    (9, 11): 70,
    (10, 11): 27})

num_nodes = 11
```

```
# Optimization model
m = Model('shortest_path')
x = m.addVars(arcs, obj=distance, name="dist")
m.addConstrs((x.sum(i, '*') - x.sum('*', i) == supply[i] for i in nodes),␣
 ↪"supply")
m.ModelSense = GRB.MINIMIZE
m.optimize()
```

```
Gurobi Optimizer version 9.0.3 build v9.0.3rc0 (linux64)
Optimize a model with 11 rows, 20 columns and 40 nonzeros
Model fingerprint: 0xd11ad6da
Coefficient statistics:
  Matrix range     [1e+00, 1e+00]
  Objective range  [6e+00, 8e+01]
  Bounds range     [0e+00, 0e+00]
  RHS range        [1e+00, 1e+00]
Presolve removed 6 rows and 6 columns
Presolve time: 0.00s
Presolved: 5 rows, 14 columns, 28 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    2.5000000e+01   2.000000e+00   0.000000e+00      0s
       2    1.0100000e+02   0.000000e+00   0.000000e+00      0s

Solved in 2 iterations and 0.00 seconds
Optimal objective  1.010000000e+02
```

[162]:
```
# Printing solution
# Variable information including sensitivity information
varnames = [f"x[{tuple(x)}]" for x in tuple(arcs)]
for var, v in zip(varnames,m.getVars()):
    print(f"{var} = {v.X}")
```

```
x[(1, 2)] = 0.0
x[(1, 3)] = 1.0
x[(1, 4)] = 0.0
x[(2, 5)] = 0.0
x[(2, 6)] = 0.0
x[(3, 5)] = 0.0
x[(3, 6)] = 0.0
x[(3, 7)] = 1.0
x[(4, 6)] = 0.0
x[(4, 7)] = 0.0
x[(5, 8)] = 0.0
x[(5, 9)] = 0.0
x[(6, 8)] = 0.0
x[(6, 9)] = 0.0
```

```
x[(6, 10)] = 0.0
x[(7, 9)] = 1.0
x[(7, 10)] = 0.0
x[(8, 11)] = 0.0
x[(9, 11)] = 1.0
x[(10, 11)] = 0.0
```

We can clearly see the shortest route from node 1 to node 11 is the following:

$x_{1,3} \rightarrow x_{3,7} \rightarrow x_{7,9} \rightarrow x_{9,11}$

0.4 Problem 1-c

[163]:
```python
# Optimal shadow prices
for n,c in enumerate(m.getConstrs()):
    print(f"{c.ConstrName} : shadow price = {int(c.Pi)}")
```

```
supply[1] : shadow price = 48
supply[2] : shadow price = 4
supply[3] : shadow price = 30
supply[4] : shadow price = -37
supply[5] : shadow price = 0
supply[6] : shadow price = 0
supply[7] : shadow price = 23
supply[8] : shadow price = 5
supply[9] : shadow price = 17
supply[10] : shadow price = -26
supply[11] : shadow price = -53
```

[164]:
```python
# Printing solution
# Variable information including sensitivity information
varnames = [f"x[{tuple(x)}]" for x in tuple(arcs)]
for var, v in zip(varnames,m.getVars()):
    print(f"{var} = {int(v.X)}, reduced cost = {abs(int(v.RC)):.2f}")
```

```
x[(1, 2)] = 0, reduced cost = 0.00
x[(1, 3)] = 1, reduced cost = 0.00
x[(1, 4)] = 0, reduced cost = 0.00
x[(2, 5)] = 0, reduced cost = 13.00
x[(2, 6)] = 0, reduced cost = 9.00
x[(3, 5)] = 0, reduced cost = 25.00
x[(3, 6)] = 0, reduced cost = 23.00
x[(3, 7)] = 1, reduced cost = 0.00
x[(4, 6)] = 0, reduced cost = 63.00
x[(4, 7)] = 0, reduced cost = 126.00
x[(5, 8)] = 0, reduced cost = 11.00
x[(5, 9)] = 0, reduced cost = 24.00
x[(6, 8)] = 0, reduced cost = 31.00
x[(6, 9)] = 0, reduced cost = 83.00
```

```
x[(6, 10)] = 0, reduced cost = 0.00
x[(7, 9)] = 1, reduced cost = 0.00
x[(7, 10)] = 0, reduced cost = 11.00
x[(8, 11)] = 0, reduced cost = 0.00
x[(9, 11)] = 1, reduced cost = 0.00
x[(10, 11)] = 0, reduced cost = 0.00
```

While very interesting, there might not be a ton to do with the shadow prices or reduced costs as they might impose changes on a seemingly impossible to change graph. Assuming, obviously, that stuff we move always comes from node 1, then in order to effectively use the information we get from the shadow prices or reduced costs we would have to modify the graph to see how the route between node 1 and node 11 changes.

Distances between locations rarely ever change, therefore the information isn't particularly practical.

We could maybe impose a different scenario where our starting node is different to node 1 and we could, perhaps, obtain a different path.

Maybe we can experiment a little bit with the reduced costs:

[165]:
```
# constructing a loop to modify the distances in an unrealistic manner to see␣
↪how routes would change using our reduced costs
sols = {k:{s:0 for s in tuple(arcs)} for k in tuple(arcs)}
orig_distance = distance.copy()
for d, red_cost in zip(distance,[v.RC for v in m.getVars()]):
    distance = orig_distance.copy()
    if red_cost < distance[d]:
        distance[d] -= red_cost

        # Optimization model
        m = Model('shortest_path')
        x = m.addVars(arcs, obj=distance, name="dist")
        m.addConstrs((x.sum(i, '*') - x.sum('*', i) == supply[i] for i in␣
↪nodes), "supply")
        m.ModelSense = GRB.MINIMIZE
        m.optimize()

        varnames = [x for x in tuple(arcs)]
        for var, v in zip(varnames,m.getVars()):
            sols[d][var] = v.X
```

```
Gurobi Optimizer version 9.0.3 build v9.0.3rc0 (linux64)
Optimize a model with 11 rows, 20 columns and 40 nonzeros
Model fingerprint: 0xd11ad6da
Coefficient statistics:
  Matrix range     [1e+00, 1e+00]
  Objective range  [6e+00, 8e+01]
  Bounds range     [0e+00, 0e+00]
  RHS range        [1e+00, 1e+00]
```

```
Presolve removed 6 rows and 6 columns
Presolve time: 0.00s
Presolved: 5 rows, 14 columns, 28 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    2.5000000e+01   2.000000e+00   0.000000e+00      0s
       2    1.0100000e+02   0.000000e+00   0.000000e+00      0s

Solved in 2 iterations and 0.00 seconds
Optimal objective  1.010000000e+02
Gurobi Optimizer version 9.0.3 build v9.0.3rc0 (linux64)
Optimize a model with 11 rows, 20 columns and 40 nonzeros
Model fingerprint: 0xd11ad6da
Coefficient statistics:
  Matrix range     [1e+00, 1e+00]
  Objective range  [6e+00, 8e+01]
  Bounds range     [0e+00, 0e+00]
  RHS range        [1e+00, 1e+00]
Presolve removed 6 rows and 6 columns
Presolve time: 0.00s
Presolved: 5 rows, 14 columns, 28 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    2.5000000e+01   2.000000e+00   0.000000e+00      0s
       2    1.0100000e+02   0.000000e+00   0.000000e+00      0s

Solved in 2 iterations and 0.00 seconds
Optimal objective  1.010000000e+02
Gurobi Optimizer version 9.0.3 build v9.0.3rc0 (linux64)
Optimize a model with 11 rows, 20 columns and 40 nonzeros
Model fingerprint: 0xd11ad6da
Coefficient statistics:
  Matrix range     [1e+00, 1e+00]
  Objective range  [6e+00, 8e+01]
  Bounds range     [0e+00, 0e+00]
  RHS range        [1e+00, 1e+00]
Presolve removed 6 rows and 6 columns
Presolve time: 0.00s
Presolved: 5 rows, 14 columns, 28 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    2.5000000e+01   2.000000e+00   0.000000e+00      0s
       2    1.0100000e+02   0.000000e+00   0.000000e+00      0s

Solved in 2 iterations and 0.00 seconds
Optimal objective  1.010000000e+02
Gurobi Optimizer version 9.0.3 build v9.0.3rc0 (linux64)
Optimize a model with 11 rows, 20 columns and 40 nonzeros
```

```
Model fingerprint: 0xa13d88f5
Coefficient statistics:
  Matrix range     [1e+00, 1e+00]
  Objective range  [4e+00, 8e+01]
  Bounds range     [0e+00, 0e+00]
  RHS range        [1e+00, 1e+00]
Presolve removed 6 rows and 6 columns
Presolve time: 0.00s
Presolved: 5 rows, 14 columns, 28 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    2.5000000e+01   2.000000e+00   0.000000e+00      0s
       3    1.0100000e+02   0.000000e+00   0.000000e+00      0s

Solved in 3 iterations and 0.00 seconds
Optimal objective  1.010000000e+02
Gurobi Optimizer version 9.0.3 build v9.0.3rc0 (linux64)
Optimize a model with 11 rows, 20 columns and 40 nonzeros
Model fingerprint: 0x7d4b8cce
Coefficient statistics:
  Matrix range     [1e+00, 1e+00]
  Objective range  [4e+00, 8e+01]
  Bounds range     [0e+00, 0e+00]
  RHS range        [1e+00, 1e+00]
Presolve removed 6 rows and 6 columns
Presolve time: 0.00s
Presolved: 5 rows, 14 columns, 28 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    2.5000000e+01   2.000000e+00   0.000000e+00      0s
       2    1.0100000e+02   0.000000e+00   0.000000e+00      0s

Solved in 2 iterations and 0.00 seconds
Optimal objective  1.010000000e+02
Gurobi Optimizer version 9.0.3 build v9.0.3rc0 (linux64)
Optimize a model with 11 rows, 20 columns and 40 nonzeros
Model fingerprint: 0xda71a686
Coefficient statistics:
  Matrix range     [1e+00, 1e+00]
  Objective range  [6e+00, 8e+01]
  Bounds range     [0e+00, 0e+00]
  RHS range        [1e+00, 1e+00]
Presolve removed 6 rows and 6 columns
Presolve time: 0.00s
Presolved: 5 rows, 14 columns, 28 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    2.5000000e+01   2.000000e+00   0.000000e+00      0s
```

```
       3    1.0100000e+02   0.000000e+00   0.000000e+00      0s

Solved in 3 iterations and 0.00 seconds
Optimal objective  1.010000000e+02
Gurobi Optimizer version 9.0.3 build v9.0.3rc0 (linux64)
Optimize a model with 11 rows, 20 columns and 40 nonzeros
Model fingerprint: 0xcaaa6154
Coefficient statistics:
  Matrix range     [1e+00, 1e+00]
  Objective range  [6e+00, 8e+01]
  Bounds range     [0e+00, 0e+00]
  RHS range        [1e+00, 1e+00]
Presolve removed 6 rows and 6 columns
Presolve time: 0.00s
Presolved: 5 rows, 14 columns, 28 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    2.5000000e+01   2.000000e+00   0.000000e+00      0s
       2    1.0100000e+02   0.000000e+00   0.000000e+00      0s

Solved in 2 iterations and 0.00 seconds
Optimal objective  1.010000000e+02
Gurobi Optimizer version 9.0.3 build v9.0.3rc0 (linux64)
Optimize a model with 11 rows, 20 columns and 40 nonzeros
Model fingerprint: 0xd11ad6da
Coefficient statistics:
  Matrix range     [1e+00, 1e+00]
  Objective range  [6e+00, 8e+01]
  Bounds range     [0e+00, 0e+00]
  RHS range        [1e+00, 1e+00]
Presolve removed 6 rows and 6 columns
Presolve time: 0.00s
Presolved: 5 rows, 14 columns, 28 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    2.5000000e+01   2.000000e+00   0.000000e+00      0s
       2    1.0100000e+02   0.000000e+00   0.000000e+00      0s

Solved in 2 iterations and 0.00 seconds
Optimal objective  1.010000000e+02
Gurobi Optimizer version 9.0.3 build v9.0.3rc0 (linux64)
Optimize a model with 11 rows, 20 columns and 40 nonzeros
Model fingerprint: 0xd11ad6da
Coefficient statistics:
  Matrix range     [1e+00, 1e+00]
  Objective range  [6e+00, 8e+01]
  Bounds range     [0e+00, 0e+00]
  RHS range        [1e+00, 1e+00]
```

```
Presolve removed 6 rows and 6 columns
Presolve time: 0.00s
Presolved: 5 rows, 14 columns, 28 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    2.5000000e+01   2.000000e+00   0.000000e+00      0s
       2    1.0100000e+02   0.000000e+00   0.000000e+00      0s

Solved in 2 iterations and 0.00 seconds
Optimal objective  1.010000000e+02
Gurobi Optimizer version 9.0.3 build v9.0.3rc0 (linux64)
Optimize a model with 11 rows, 20 columns and 40 nonzeros
Model fingerprint: 0xd11ad6da
Coefficient statistics:
  Matrix range     [1e+00, 1e+00]
  Objective range  [6e+00, 8e+01]
  Bounds range     [0e+00, 0e+00]
  RHS range        [1e+00, 1e+00]
Presolve removed 6 rows and 6 columns
Presolve time: 0.00s
Presolved: 5 rows, 14 columns, 28 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    2.5000000e+01   2.000000e+00   0.000000e+00      0s
       2    1.0100000e+02   0.000000e+00   0.000000e+00      0s

Solved in 2 iterations and 0.00 seconds
Optimal objective  1.010000000e+02
Gurobi Optimizer version 9.0.3 build v9.0.3rc0 (linux64)
Optimize a model with 11 rows, 20 columns and 40 nonzeros
Model fingerprint: 0x0e0c647c
Coefficient statistics:
  Matrix range     [1e+00, 1e+00]
  Objective range  [6e+00, 8e+01]
  Bounds range     [0e+00, 0e+00]
  RHS range        [1e+00, 1e+00]
Presolve removed 6 rows and 6 columns
Presolve time: 0.00s
Presolved: 5 rows, 14 columns, 28 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    2.5000000e+01   2.000000e+00   0.000000e+00      0s
       2    1.0100000e+02   0.000000e+00   0.000000e+00      0s

Solved in 2 iterations and 0.00 seconds
Optimal objective  1.010000000e+02
Gurobi Optimizer version 9.0.3 build v9.0.3rc0 (linux64)
Optimize a model with 11 rows, 20 columns and 40 nonzeros
```

```
Model fingerprint: 0xd11ad6da
Coefficient statistics:
  Matrix range     [1e+00, 1e+00]
  Objective range  [6e+00, 8e+01]
  Bounds range     [0e+00, 0e+00]
  RHS range        [1e+00, 1e+00]
Presolve removed 6 rows and 6 columns
Presolve time: 0.00s
Presolved: 5 rows, 14 columns, 28 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    2.5000000e+01   2.000000e+00   0.000000e+00      0s
       2    1.0100000e+02   0.000000e+00   0.000000e+00      0s

Solved in 2 iterations and 0.00 seconds
Optimal objective  1.010000000e+02
Gurobi Optimizer version 9.0.3 build v9.0.3rc0 (linux64)
Optimize a model with 11 rows, 20 columns and 40 nonzeros
Model fingerprint: 0xd11ad6da
Coefficient statistics:
  Matrix range     [1e+00, 1e+00]
  Objective range  [6e+00, 8e+01]
  Bounds range     [0e+00, 0e+00]
  RHS range        [1e+00, 1e+00]
Presolve removed 6 rows and 6 columns
Presolve time: 0.00s
Presolved: 5 rows, 14 columns, 28 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    2.5000000e+01   2.000000e+00   0.000000e+00      0s
       2    1.0100000e+02   0.000000e+00   0.000000e+00      0s

Solved in 2 iterations and 0.00 seconds
Optimal objective  1.010000000e+02
Gurobi Optimizer version 9.0.3 build v9.0.3rc0 (linux64)
Optimize a model with 11 rows, 20 columns and 40 nonzeros
Model fingerprint: 0xd11ad6da
Coefficient statistics:
  Matrix range     [1e+00, 1e+00]
  Objective range  [6e+00, 8e+01]
  Bounds range     [0e+00, 0e+00]
  RHS range        [1e+00, 1e+00]
Presolve removed 6 rows and 6 columns
Presolve time: 0.00s
Presolved: 5 rows, 14 columns, 28 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    2.5000000e+01   2.000000e+00   0.000000e+00      0s
```

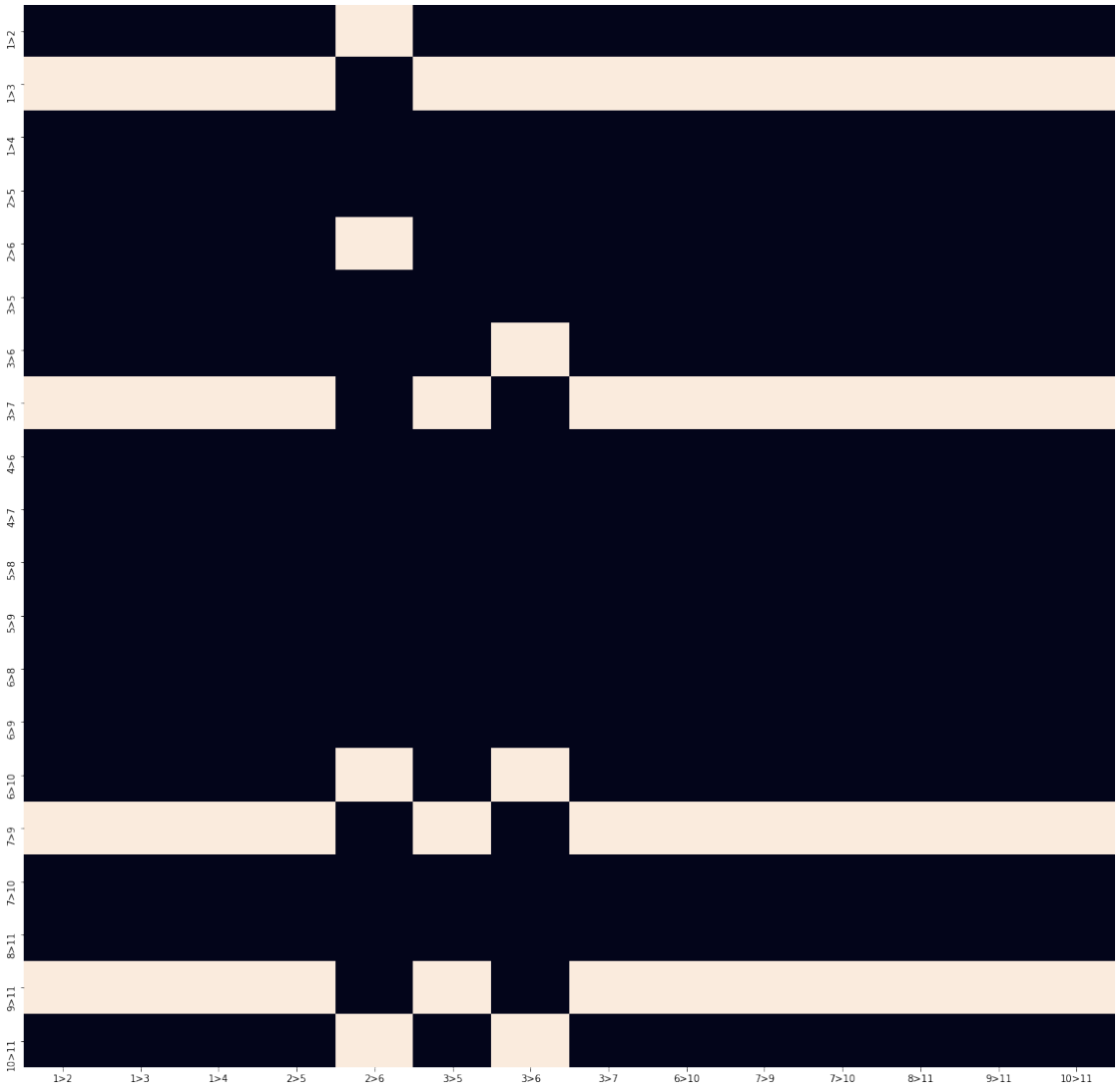
```
       2     1.0100000e+02    0.000000e+00    0.000000e+00       0s

Solved in 2 iterations and 0.00 seconds
Optimal objective  1.010000000e+02
```

[166]:
```
# route changes after applying reduced costs that don't make distances negative␣
↪(as that's not possible)
sols2 = {}
remove = []
for key in sols.keys():
    sols2[f"{key[0]}>{key[1]}"] = {}
for key in sols.keys():
    for key2,val in sols[key].items():
        sols2[f"{key[0]}>{key[1]}"][f"{key2[0]}>{key2[1]}"] = val
    if pd.DataFrame(sols2)[f"{key[0]}>{key[1]}"].sum() == 0:
        remove.append(f"{key[0]}>{key[1]}")
df = pd.DataFrame(sols2)
df = df[[x for x in df.columns if x not in remove]]

fig = plt.figure(figsize=(20,20))
sns.heatmap(df, cbar=False)
```

[166]:
```
<AxesSubplot:>
```



We clearly see, after applying each reduced cost individually that the reduced costs that seem to affect the route are uniquely the reduced cost applied to $x_{2,6}$ and the reduced cost applied to $x_{3,6}$.

Of course, different combinations of reduced costs would change the path in different ways, however, we can see that by themselves, these would be the changes to the paths

Our heatmap shows in BLACK the nodes that would NOT be taken. while the nodes in white would be taken.

our shortest path after applying the reduced cost for $x_{2,6}$ would be (basically reducing the distance between node 2 and node 6):

$x_{1,2} \rightarrow x_{2,6} \rightarrow x_{6,10} \rightarrow x_{10,11}$

our shortest path after applying the reduced cost for $x_{3,6}$ would be (basically reducing the distance between node 2 and node 6):

$x_{1,3} \rightarrow x_{3,6} \rightarrow x_{6,10} \rightarrow x_{10,11}$

Which are, admittedly, similar paths.

0.5 Problem 2-a

$x_1, x_2, x_3, x_4 \in \mathbb{N}$

$y_1, y_2, y_3, y_4 \in \{0, 1\}$

maximize $49x_1 - 993y_1 + 54x_2 - 715y_2 + 47x_3 - 834y_3 + 51x_4 - 940y_4$

subject to:

$x_1 \leq 2500y_1$, $x_2 \leq 2500y_2$, $x_3 \leq 2500y_3$, $x_4 \leq 2500y_4$

$3x_1 + 3x_2 + 5x_3 + 6x_4 \leq 598$

$6x_1 + 3x_2 + 4x_3 + 3x_4 \leq 287$

$6x_1 + 6x_2 + 2x_3 + 7x_4 \leq 392$

0.6 Problem 2-b

[180]:
```
# cols and rows
m = 3
n = 4

# dims
op = range(m)
prod = range(n)
arr = range(n)

# primal problem setup
r_coeff = [49, 54, 47, 51]
s_coeff = [993, 715, 834, 940]
A_coeff = [[3, 3, 5, 6],
           [6, 3, 4, 3],
           [6, 6, 2, 7]]
b_coeff = [598, 287, 392]

# dicts setup
b, A = {}, {}
for i in op:
    A[i] = {val:A_coeff[i][val] for val in prod}
    b[i] = b_coeff[i]

r = {val:r_coeff[val] for val in prod}
s = {val:s_coeff[val] for val in arr}


# model definition
```

```
m = Model('prod_ops')

# variables and constraints
x = m.addVars(prod, name="x", vtype=GRB.INTEGER)
y = m.addVars(arr, name="y", vtype=GRB.BINARY)
m.addConstrs((quicksum(A[i][j] * x[j] for j in prod) <= b[i] for i in op), name␣
 ↪= "pi")
m.addConstrs((2500*y[j] >= x[j] for j in prod), name = "const")

# objective
obj = quicksum(r[j] * x[j] for j in prod) - quicksum(s[j] * y[j] for j in arr)
m.setObjective(obj, GRB.MAXIMIZE)

# other params
m.setParam(GRB.Param.Presolve, 0),
m.setParam(GRB.Param.Heuristics, 0),
m.setParam(GRB.Param.Cuts, 0)

# optimize
m.optimize()
```

```
Changed value of parameter Presolve to 0
   Prev: -1  Min: -1  Max: 2  Default: -1
Changed value of parameter Heuristics to 0.0
   Prev: 0.05  Min: 0.0  Max: 1.0  Default: 0.05
Changed value of parameter Cuts to 0
   Prev: -1  Min: -1  Max: 3  Default: -1
Gurobi Optimizer version 9.0.3 build v9.0.3rc0 (linux64)
Optimize a model with 7 rows, 8 columns and 20 nonzeros
Model fingerprint: 0x5a2b5452
Variable types: 0 continuous, 8 integer (4 binary)
Coefficient statistics:
  Matrix range     [1e+00, 2e+03]
  Objective range  [5e+01, 1e+03]
  Bounds range     [1e+00, 1e+00]
  RHS range        [3e+02, 6e+02]
Variable types: 0 continuous, 8 integer (4 binary)

Root relaxation: objective 4.381754e+03, 5 iterations, 0.00 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0 4381.75391    0    4          - 4381.75391      -     -    0s
     0     0 4381.75391    0    4          - 4381.75391      -     -    0s
     0     2 4381.75391    0    4          - 4381.75391      -     -    0s
*   15    16               4    1310.0000000 2847.33333   117%   1.7    0s
```

```
*   18    16               4    2795.0000000 2845.57143  1.81%   1.6    0s
*   27     9               5    2824.0000000 2843.60280  0.69%   1.4    0s
*   34     0               6    2831.0000000 2831.00000  0.00%   1.3    0s

Explored 37 nodes (48 simplex iterations) in 0.01 seconds
Thread count was 16 (of 16 available processors)

Solution count 4: 2831 2824 2795 1310

Optimal solution found (tolerance 1.00e-04)
Best objective 2.831000000000e+03, best bound 2.831000000000e+03, gap 0.0000%
```

[179]:
```
# Printing solution
# Variable information including sensitivity information
varnames = [f"x[{x+1}]" for x in range(len(m.getVars()))]
for var, v in zip(varnames,m.getVars()):
    print(f"{var} = {int(v.X)}")
```

```
x[1] = 0
x[2] = 55
x[3] = 30
x[4] = 0
x[5] = 0
x[6] = 1
x[7] = 1
x[8] = 0
```

[182]:
```
# Linear relaxation
x = m.addVars(range(n), name="x")

# Constraints
m.addConstrs((quicksum(A[i][j] * x[j] for j in prod) <= b[i] for i in op), name␣
 ↪= "pi")
m.addConstrs((2500*y[j] >= x[j] for j in prod), name = "const")

# Objective
obj = quicksum(r[j] * x[j] for j in prod) - quicksum(s[j] * y[j] for j in arr)
m.setObjective(obj, GRB.MAXIMIZE)
m.optimize()
```

```
Gurobi Optimizer version 9.0.3 build v9.0.3rc0 (linux64)
Optimize a model with 21 rows, 16 columns and 60 nonzeros
Model fingerprint: 0x255b3ad2
Variable types: 8 continuous, 8 integer (4 binary)
Coefficient statistics:
  Matrix range     [1e+00, 2e+03]
  Objective range  [5e+01, 1e+03]
  Bounds range     [1e+00, 1e+00]
```

```
  RHS range        [3e+02, 6e+02]

MIP start from previous solve produced solution with objective 2858.67 (0.00s)
Loaded MIP start from previous solve with objective 2858.67

Variable types: 8 continuous, 8 integer (4 binary)

Root relaxation: objective 4.381754e+03, 5 iterations, 0.00 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0 4381.75391    0    2 2858.66667 4381.75391  53.3%     -    0s
     0     0 4381.75391    0    2 2858.66667 4381.75391  53.3%     -    0s
     0     2 4381.75391    0    2 2858.66667 4381.75391  53.3%     -    0s

Explored 5 nodes (10 simplex iterations) in 0.01 seconds
Thread count was 16 (of 16 available processors)

Solution count 1: 2858.67

Optimal solution found (tolerance 1.00e-04)
Best objective 2.858666666667e+03, best bound 2.858666666667e+03, gap 0.0000%
```

the solutions are the following:

For our integer model:

- The objective value is 2831

For our linear relaxation:

- The objective value is $2848.\bar{6}$ We could consider the the linear relaxation an upper bound for the integer optimization solution similar to how we saw in the plots where the LO solution was superimposed on the integer feasable area, the integer solutio area is fully contained inside the LO solution area. Therefore being effectively fully bounded by it from all sides. This solution is perhaps not the most useful, so we keep the integer solution, but it's interesting to see the differences.

0.7 Problem 3-a

[167]:
```python
# objects
n = 24

# define k
def k(v):
    return np.floor(np.log(2*v))
```

```
# weight
w = [2**(k(n) + n + 1) + 2**(k(n) + j) + 1 for j in range(1,n+1)]

# value/profit
r = [2**(k(n) + n + 1) + 2**(k(n) + j) + 1 for j in range(1,n+1)]

# kanpsack's capacity
b = sum([np.floor(w[j]/2) for j in range(n)])

# defining the model
m = Model('knapsack')

# Binary variables
x = m.addVars(range(n), name="x", vtype=GRB.BINARY)

# Capacity constraints
m.addConstr(quicksum(w[j]*x[j] for j in range(n)) <= b)

# Objective
obj = quicksum(r[j]*x[j] for j in range(n))

m.setObjective(obj, GRB.MAXIMIZE)

m.optimize()
```

```
Gurobi Optimizer version 9.0.3 build v9.0.3rc0 (linux64)
Optimize a model with 1 rows, 24 columns and 24 nonzeros
Model fingerprint: 0x16d202d5
Variable types: 0 continuous, 24 integer (24 binary)
Coefficient statistics:
  Matrix range     [3e+08, 4e+08]
  Objective range  [3e+08, 4e+08]
  Bounds range     [1e+00, 1e+00]
  RHS range        [3e+09, 3e+09]
Warning: Model contains large matrix coefficients
Warning: Model contains large rhs
         Consider reformulating model or setting NumericFocus parameter
         to avoid numerical issues.
Found heuristic solution: objective 3.221291e+09
Presolve time: 0.00s
Presolved: 1 rows, 24 columns, 24 nonzeros
Variable types: 0 continuous, 24 integer (24 binary)

Root relaxation: objective 3.355443e+09, 5 iterations, 0.00 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time
```

```
     0     0 3.3554e+09    0    1 3.2213e+09 3.3554e+09  4.16%     -    0s
H    0     0                    3.353526e+09 3.3554e+09  0.06%     -    0s
     0     0 3.3554e+09    0    1 3.3535e+09 3.3554e+09  0.06%     -    0s
H    0     0                    3.354935e+09 3.3554e+09  0.02%     -    0s
     0     0 3.3554e+09    0    1 3.3549e+09 3.3554e+09  0.02%     -    0s
     0     0 3.3554e+09    0    1 3.3549e+09 3.3554e+09  0.02%     -    0s
     0     0 3.3554e+09    0    1 3.3549e+09 3.3554e+09  0.02%     -    0s
     0     0 3.3554e+09    0    1 3.3549e+09 3.3554e+09  0.02%     -    0s
     0     2 3.3554e+09    0    1 3.3549e+09 3.3554e+09  0.02%     -    0s
H  455   155                    3.355288e+09 3.3554e+09  0.00%   1.1    0s

Cutting planes:
  Cover: 7

Explored 511 nodes (546 simplex iterations) in 0.02 seconds
Thread count was 16 (of 16 available processors)

Solution count 4: 3.35529e+09 3.35494e+09 3.35353e+09 3.22129e+09

Optimal solution found (tolerance 1.00e-04)
Best objective 3.355287564000e+09, best bound 3.355443192000e+09, gap 0.0046%
```

[121]:
```
print(m.NodeCount)
print(m.Runtime)
print(m.ObjVal)
```

```
511.0
0.024396181106567383
3355287564.0
```

0.7.1 - Our optimal solution (using a tolerance of 1×10^{-4}) is $\sim 3.355287564 \times 10^{9}$

0.7.2 - The solution time is 0.024396181106567383 seconds

0.7.3 - The number of nodes explored is of 511 nodes

[168]:
```
# Linear relaxation
x = m.addVars(range(n), name="x")

# Capacity constraints
m.addConstr(quicksum(w[j]*x[j] for j in range(n)) <= b)

# Objective
obj = quicksum(r[j]*x[j] for j in range(n))

m.setObjective(obj, GRB.MAXIMIZE)
```

```
m.optimize()
```

```
Gurobi Optimizer version 9.0.3 build v9.0.3rc0 (linux64)
Optimize a model with 2 rows, 48 columns and 48 nonzeros
Model fingerprint: 0x83b9db41
Variable types: 24 continuous, 24 integer (24 binary)
Coefficient statistics:
  Matrix range     [3e+08, 4e+08]
  Objective range  [3e+08, 4e+08]
  Bounds range     [1e+00, 1e+00]
  RHS range        [3e+09, 3e+09]
Warning: Model contains large matrix coefficients
Warning: Model contains large rhs
         Consider reformulating model or setting NumericFocus parameter
         to avoid numerical issues.

MIP start from previous solve did not produce a new incumbent solution

Found heuristic solution: objective 3.355443e+09
Presolve removed 1 rows and 47 columns
Presolve time: 0.00s

Explored 0 nodes (0 simplex iterations) in 0.00 seconds
Thread count was 1 (of 16 available processors)

Solution count 1: 3.35544e+09
No other solutions better than 0

Model is unbounded
Best objective 3.355443192000e+09, best bound -, gap -
```

0.8 Problem 3-b

[80]:
```
from random import choice

weights = [(i,x) for i,x in enumerate(w)]
indexes = {i:[] for i in range(100)}

for i in range(100):
    k = b
    weights_set = set(weights)
    while k > 0:
        chosen_x = choice(list(weights_set))
        weights_set -= {chosen_x}
        k -= chosen_x[1]
        indexes[i].append(chosen_x[0])
```

[81]:
```
models = {i:{'model':0, 'rt':0, 'nd':0} for i in range(50)}

# k
def k(n): return np.floor(np.log(2*n))

for idx, index in indexes.items():
    # objects
    n = 24
    # weight
    w = [2**(k(n) + n + 1) + 2**(k(n) + j) + 1 for j in range(1,n+1)]
    # value/profit
    r = [2**(k(n) + n + 1) + 2**(k(n) + j) + 1 for j in range(1,n+1)]
    # kanpsack's capacity
    b = sum([np.floor(w[j]/2) for j in range(n)])
    # defining the model
    m = Model(f'knapsack_{idx}')
    # Binary variables
    x = m.addVars(range(n), name="x", vtype=GRB.BINARY)
    # Capacity constraints
    m.addConstr(quicksum(w[j]*x[j] for j in range(n)) <= b)
    m.addConstr(quicksum(x[j] for j in index) <= len(index)-1)
    # Objective
    obj = quicksum(r[j]*x[j] for j in range(n))
    m.setObjective(obj, GRB.MAXIMIZE)
    print(index)
    m.optimize()
    # appending to models
    models[idx]['model'] = m
    models[idx]['rt'] = m.Runtime
    models[idx]['nd'] = m.NodeCount
```

```
09 3.3554e+09  0.01%     -    0s
     0     0 3.3554e+09    0    1 3.3549e+09 3.3554e+09  0.01%     -    0s
     0     2 3.3554e+09    0    1 3.3549e+09 3.3554e+09  0.01%     -    0s
H  142    87                     3.355193e+09 3.3554e+09  0.01%   1.1    0s

Cutting planes:
  Cover: 6

Explored 159 nodes (185 simplex iterations) in 0.02 seconds
Thread count was 16 (of 16 available processors)

Solution count 4: 3.35519e+09 3.35494e+09 3.35345e+09 3.22129e+09

Optimal solution found (tolerance 1.00e-04)
Best objective 3.355193388000e+09, best bound 3.355443192000e+09, gap 0.0074%
[1, 10, 6, 0, 7, 23, 12, 14, 22, 3, 21, 13]
```

```
Gurobi Optimizer version 9.0.3 build v9.0.3rc0 (linux64)
Optimize a model with 2 rows, 24 columns and 36 nonzeros
Model fingerprint: 0x85b5ed38
Variable types: 0 continuous, 24 integer (24 binary)
Coefficient statistics:
  Matrix range     [1e+00, 4e+08]
  Objective range  [3e+08, 4e+08]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+01, 3e+09]
Warning: Model contains large matrix coefficients
Warning: Model contains large rhs
         Consider reformulating model or setting NumericFocus parameter
         to avoid numerical issues.
Found heuristic solution: objective 3.221291e+09
Presolve time: 0.00s
Presolved: 2 rows, 24 columns, 36 nonzeros
Variable types: 0 continuous, 24 integer (24 binary)

Root relaxation: objective 3.355443e+09, 5 iterations, 0.00 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0 3.3554e+09    0    1 3.2213e+09 3.3554e+09  4.16%     -    0s
H    0     0                     3.354935e+09 3.3554e+09  0.02%     -    0s
     0     0 3.3554e+09    0    1 3.3549e+09 3.3554e+09  0.02%     -    0s
H    0     0                     3.355198e+09 3.3554e+09  0.01%     -    0s

Cutting planes:
  Cover: 1

Explored 1 nodes (6 simplex iterations) in 0.01 seconds
Thread count was 16 (of 16 available processors)

Solution count 3: 3.3552e+09 3.35494e+09 3.22129e+09

Optimal solution found (tolerance 1.00e-04)
Best objective 3.355197500000e+09, best bound 3.355443192000e+09, gap 0.0073%
[20, 8, 6, 12, 7, 10, 23, 19, 14, 17, 16, 4]
Gurobi Optimizer version 9.0.3 build v9.0.3rc0 (linux64)
Optimize a model with 2 rows, 24 columns and 36 nonzeros
Model fingerprint: 0xe1c24edd
Variable types: 0 continuous, 24 integer (24 binary)
Coefficient statistics:
  Matrix range     [1e+00, 4e+08]
  Objective range  [3e+08, 4e+08]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+01, 3e+09]
```

```
Warning: Model contains large matrix coefficients
Warning: Model contains large rhs
         Consider reformulating model or setting NumericFocus parameter
         to avoid numerical issues.
Found heuristic solution: objective 3.221291e+09
Presolve time: 0.00s
Presolved: 2 rows, 24 columns, 36 nonzeros
Variable types: 0 continuous, 24 integer (24 binary)

Root relaxation: objective 3.355443e+09, 5 iterations, 0.00 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0 3.3554e+09    0    1 3.2213e+09 3.3554e+09  4.16%     -    0s
H    0     0                     3.354542e+09 3.3554e+09  0.03%     -    0s
     0     0 3.3554e+09    0    1 3.3545e+09 3.3554e+09  0.03%     -    0s
     0     0 3.3554e+09    0    1 3.3545e+09 3.3554e+09  0.03%     -    0s
     0     0 3.3554e+09    0    1 3.3545e+09 3.3554e+09  0.03%     -    0s
     0     0 3.3554e+09    0    5 3.3545e+09 3.3554e+09  0.03%     -    0s
     0     0 3.3554e+09    0    5 3.3545e+09 3.3554e+09  0.03%     -    0s
H    0     0                     3.355247e+09 3.3554e+09  0.01%     -    0s

Cutting planes:
  Cover: 4

Explored 1 nodes (9 simplex iterations) in 0.01 seconds
Thread count was 16 (of 16 available processors)

Solution count 3: 3.35525e+09 3.35454e+09 3.22129e+09

Optimal solution found (tolerance 1.00e-04)
Best objective 3.355246652000e+09, best bound 3.355443192000e+09, gap 0.0059%
[19, 15, 22, 17, 7, 13, 14, 18, 11, 0, 12, 10, 4]
Gurobi Optimizer version 9.0.3 build v9.0.3rc0 (linux64)
Optimize a model with 2 rows, 24 columns and 37 nonzeros
Model fingerprint: 0x2030afb9
Variable types: 0 continuous, 24 integer (24 binary)
Coefficient statistics:
  Matrix range     [1e+00, 4e+08]
  Objective range  [3e+08, 4e+08]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+01, 3e+09]
Warning: Model contains large matrix coefficients
Warning: Model contains large rhs
         Consider reformulating model or setting NumericFocus parameter
         to avoid numerical issues.
Found heuristic solution: objective 3.221291e+09
```

```
Presolve time: 0.00s
Presolved: 2 rows, 24 columns, 37 nonzeros
Variable types: 0 continuous, 24 integer (24 binary)

Root relaxation: objective 3.355443e+09, 5 iterations, 0.00 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0 3.3554e+09    0    1 3.2213e+09 3.3554e+09  4.16%     -    0s
H    0     0                     3.353526e+09 3.3554e+09  0.06%     -    0s
H    0     0                     3.354739e+09 3.3554e+09  0.02%     -    0s
     0     0 3.3554e+09    0    1 3.3547e+09 3.3554e+09  0.02%     -    0s
     0     0 3.3554e+09    0    1 3.3547e+09 3.3554e+09  0.02%     -    0s
     0     0 3.3554e+09    0    1 3.3547e+09 3.3554e+09  0.02%     -    0s
     0     0 3.3554e+09    0    1 3.3547e+09 3.3554e+09  0.02%     -    0s
     0     0 3.3554e+09    0    1 3.3547e+09 3.3554e+09  0.02%     -    0s
     0     2 3.3554e+09    0    1 3.3547e+09 3.3554e+09  0.02%     -    0s
H  561     0                     3.355410e+09 3.3554e+09  0.00%   1.1    0s

Cutting planes:
  Cover: 7

Explored 698 nodes (1045 simplex iterations) in 0.02 seconds
Thread count was 16 (of 16 available processors)

Solution count 4: 3.35541e+09 3.35474e+09 3.35353e+09 3.22129e+09

Optimal solution found (tolerance 1.00e-04)
Best objective 3.355410444000e+09, best bound 3.355410444000e+09, gap 0.0000%
[22, 3, 18, 17, 10, 8, 0, 9, 19, 11, 16, 7, 1]
Gurobi Optimizer version 9.0.3 build v9.0.3rc0 (linux64)
Optimize a model with 2 rows, 24 columns and 37 nonzeros
Model fingerprint: 0xaa10b427
Variable types: 0 continuous, 24 integer (24 binary)
Coefficient statistics:
  Matrix range     [1e+00, 4e+08]
  Objective range  [3e+08, 4e+08]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+01, 3e+09]
Warning: Model contains large matrix coefficients
Warning: Model contains large rhs
         Consider reformulating model or setting NumericFocus parameter
         to avoid numerical issues.
Found heuristic solution: objective 3.221291e+09
Presolve time: 0.00s
Presolved: 2 rows, 24 columns, 37 nonzeros
Variable types: 0 continuous, 24 integer (24 binary)
```

```
Root relaxation: objective 3.355443e+09, 5 iterations, 0.00 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0 3.3554e+09    0    1 3.2213e+09 3.3554e+09  4.16%     -    0s
H    0     0                     3.353526e+09 3.3554e+09  0.06%     -    0s
H    0     0                     3.354739e+09 3.3554e+09  0.02%     -    0s
     0     0 3.3554e+09    0    1 3.3547e+09 3.3554e+09  0.02%     -    0s
     0     0 3.3554e+09    0    1 3.3547e+09 3.3554e+09  0.02%     -    0s
     0     0 3.3554e+09    0    1 3.3547e+09 3.3554e+09  0.02%     -    0s
     0     0 3.3554e+09    0    1 3.3547e+09 3.3554e+09  0.02%     -    0s
     0     0 3.3554e+09    0    1 3.3547e+09 3.3554e+09  0.02%     -    0s
     0     2 3.3554e+09    0    1 3.3547e+09 3.3554e+09  0.02%     -    0s
*  529     0               4    3.355410e+09 3.3554e+09  0.00%   1.1    0s

Cutting planes:
  Cover: 7

Explored 661 nodes (1016 simplex iterations) in 0.02 seconds
Thread count was 16 (of 16 available processors)

Solution count 4: 3.35541e+09 3.35474e+09 3.35353e+09 3.22129e+09

Optimal solution found (tolerance 1.00e-04)
Best objective 3.355410444000e+09, best bound 3.355410444000e+09, gap 0.0000%
[12, 0, 15, 8, 22, 21, 5, 9, 14, 2, 23, 16]
Gurobi Optimizer version 9.0.3 build v9.0.3rc0 (linux64)
Optimize a model with 2 rows, 24 columns and 36 nonzeros
Model fingerprint: 0x37010f4b
Variable types: 0 continuous, 24 integer (24 binary)
Coefficient statistics:
  Matrix range     [1e+00, 4e+08]
  Objective range  [3e+08, 4e+08]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+01, 3e+09]
Warning: Model contains large matrix coefficients
Warning: Model contains large rhs
         Consider reformulating model or setting NumericFocus parameter
         to avoid numerical issues.
Found heuristic solution: objective 3.221291e+09
Presolve time: 0.00s
Presolved: 2 rows, 24 columns, 36 nonzeros
Variable types: 0 continuous, 24 integer (24 binary)

Root relaxation: objective 3.355443e+09, 5 iterations, 0.00 seconds
```

```
    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0 3.3554e+09    0    1 3.2213e+09 3.3554e+09  4.16%     -    0s
H    0     0                     3.354935e+09 3.3554e+09  0.02%     -    0s
     0     0 3.3554e+09    0    1 3.3549e+09 3.3554e+09  0.02%     -    0s
     0     0 3.3554e+09    0    1 3.3549e+09 3.3554e+09  0.02%     -    0s
     0     0 3.3554e+09    0    1 3.3549e+09 3.3554e+09  0.02%     -    0s
     0     0 3.3554e+09    0    1 3.3549e+09 3.3554e+09  0.02%     -    0s
     0     0 3.3554e+09    0    1 3.3549e+09 3.3554e+09  0.02%     -    0s
     0     2 3.3554e+09    0    1 3.3549e+09 3.3554e+09  0.02%     -    0s
H   19    24                     3.355198e+09 3.3554e+09  0.01%   1.9    0s

Cutting planes:
  Cover: 6

Explored 23 nodes (52 simplex iterations) in 0.01 seconds
Thread count was 16 (of 16 available processors)

Solution count 3: 3.3552e+09 3.35494e+09 3.22129e+09

Optimal solution found (tolerance 1.00e-04)
Best objective 3.355197500000e+09, best bound 3.355443192000e+09, gap 0.0073%
[16, 11, 2, 17, 18, 4, 15, 10, 14, 9, 22, 12, 23]
Gurobi Optimizer version 9.0.3 build v9.0.3rc0 (linux64)
Optimize a model with 2 rows, 24 columns and 37 nonzeros
Model fingerprint: 0xd45e7559
Variable types: 0 continuous, 24 integer (24 binary)
Coefficient statistics:
  Matrix range     [1e+00, 4e+08]
  Objective range  [3e+08, 4e+08]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+01, 3e+09]
Warning: Model contains large matrix coefficients
Warning: Model contains large rhs
         Consider reformulating model or setting NumericFocus parameter
         to avoid numerical issues.
Found heuristic solution: objective 3.221291e+09
Presolve time: 0.00s
Presolved: 2 rows, 24 columns, 37 nonzeros
Variable types: 0 continuous, 24 integer (24 binary)

Root relaxation: objective 3.355443e+09, 5 iterations, 0.00 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0 3.3554e+09    0    1 3.2213e+09 3.3554e+09  4.16%     -    0s
```

```
H    0     0                    3.353690e+09 3.3554e+09  0.05%     -    0s
H    0     0                    3.354596e+09 3.3554e+09  0.03%     -    0s
     0     0 3.3554e+09    0    1 3.3546e+09 3.3554e+09  0.03%     -    0s
     0     0 3.3554e+09    0    1 3.3546e+09 3.3554e+09  0.03%     -    0s
     0     0 3.3554e+09    0    1 3.3546e+09 3.3554e+09  0.03%     -    0s
     0     0 3.3554e+09    0    1 3.3546e+09 3.3554e+09  0.03%     -    0s
     0     0 3.3554e+09    0    1 3.3546e+09 3.3554e+09  0.03%     -    0s
     0     2 3.3554e+09    0    1 3.3546e+09 3.3554e+09  0.03%     -    0s
H  142    93                    3.355036e+09 3.3554e+09  0.01%   1.1    0s
H  243   147                    3.355247e+09 3.3554e+09  0.01%   1.1    0s

Cutting planes:
  Cover: 7

Explored 551 nodes (580 simplex iterations) in 0.02 seconds
Thread count was 16 (of 16 available processors)

Solution count 5: 3.35525e+09 3.35504e+09 3.3546e+09 … 3.22129e+09

Optimal solution found (tolerance 1.00e-04)
Best objective 3.355247148000e+09, best bound 3.355443192000e+09, gap 0.0058%
[18, 6, 22, 9, 2, 1, 14, 15, 7, 21, 13, 17, 11]
Gurobi Optimizer version 9.0.3 build v9.0.3rc0 (linux64)
Optimize a model with 2 rows, 24 columns and 37 nonzeros
Model fingerprint: 0x1c52b681
Variable types: 0 continuous, 24 integer (24 binary)
Coefficient statistics:
  Matrix range     [1e+00, 4e+08]
  Objective range  [3e+08, 4e+08]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+01, 3e+09]
Warning: Model contains large matrix coefficients
Warning: Model contains large rhs
         Consider reformulating model or setting NumericFocus parameter
         to avoid numerical issues.
Found heuristic solution: objective 3.221291e+09
Presolve time: 0.00s
Presolved: 2 rows, 24 columns, 37 nonzeros
Variable types: 0 continuous, 24 integer (24 binary)

Root relaxation: objective 3.355443e+09, 5 iterations, 0.00 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0 3.3554e+09    0    1 3.2213e+09 3.3554e+09  4.16%     -    0s
H    0     0                    3.354542e+09 3.3554e+09  0.03%     -    0s
     0     0 3.3554e+09    0    1 3.3545e+09 3.3554e+09  0.03%     -    0s
```

```
     0     0 3.3554e+09    0    1 3.3545e+09 3.3554e+09  0.03%     -    0s
H    0     0                     3.355184e+09 3.3554e+09  0.01%     -    0s

Cutting planes:
  Cover: 3

Explored 1 nodes (7 simplex iterations) in 0.01 seconds
Thread count was 16 (of 16 available processors)

Solution count 3: 3.35518e+09 3.35454e+09 3.22129e+09

Optimal solution found (tolerance 1.00e-04)
Best objective 3.355183756000e+09, best bound 3.355443192000e+09, gap 0.0077%
[1, 5, 3, 4, 6, 14, 9, 7, 21, 19, 13, 22, 12]
Gurobi Optimizer version 9.0.3 build v9.0.3rc0 (linux64)
Optimize a model with 2 rows, 24 columns and 37 nonzeros
Model fingerprint: 0x75f02910
Variable types: 0 continuous, 24 integer (24 binary)
Coefficient statistics:
  Matrix range     [1e+00, 4e+08]
  Objective range  [3e+08, 4e+08]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+01, 3e+09]
Warning: Model contains large matrix coefficients
Warning: Model contains large rhs
         Consider reformulating model or setting NumericFocus parameter
         to avoid numerical issues.
Found heuristic solution: objective 3.221291e+09
Presolve time: 0.00s
Presolved: 2 rows, 24 columns, 37 nonzeros
Variable types: 0 continuous, 24 integer (24 binary)

Root relaxation: objective 3.355443e+09, 5 iterations, 0.00 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0 3.3554e+09    0    1 3.2213e+09 3.3554e+09  4.16%     -    0s
H    0     0                     3.354542e+09 3.3554e+09  0.03%     -    0s
     0     0 3.3554e+09    0    1 3.3545e+09 3.3554e+09  0.03%     -    0s
     0     0 3.3554e+09    0    1 3.3545e+09 3.3554e+09  0.03%     -    0s
     0     0 3.3554e+09    0    1 3.3545e+09 3.3554e+09  0.03%     -    0s
     0     0 3.3554e+09    0    1 3.3545e+09 3.3554e+09  0.03%     -    0s
     0     0 3.3554e+09    0    1 3.3545e+09 3.3554e+09  0.03%     -    0s
     0     2 3.3554e+09    0    1 3.3545e+09 3.3554e+09  0.03%     -    0s
H   52    64                     3.355198e+09 3.3554e+09  0.01%   1.3    0s

Cutting planes:
```

```
  Cover: 6

Explored 63 nodes (86 simplex iterations) in 0.01 seconds
Thread count was 16 (of 16 available processors)

Solution count 3: 3.3552e+09 3.35454e+09 3.22129e+09

Optimal solution found (tolerance 1.00e-04)
Best objective 3.355197548000e+09, best bound 3.355443192000e+09, gap 0.0073%
[1, 16, 7, 15, 22, 5, 0, 18, 10, 19, 2, 12, 23]
Gurobi Optimizer version 9.0.3 build v9.0.3rc0 (linux64)
Optimize a model with 2 rows, 24 columns and 37 nonzeros
Model fingerprint: 0xccb84856
Variable types: 0 continuous, 24 integer (24 binary)
Coefficient statistics:
  Matrix range     [1e+00, 4e+08]
  Objective range  [3e+08, 4e+08]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+01, 3e+09]
Warning: Model contains large matrix coefficients
Warning: Model contains large rhs
         Consider reformulating model or setting NumericFocus parameter
         to avoid numerical issues.
Found heuristic solution: objective 3.221291e+09
Presolve time: 0.00s
Presolved: 2 rows, 24 columns, 37 nonzeros
Variable types: 0 continuous, 24 integer (24 binary)

Root relaxation: objective 3.355443e+09, 5 iterations, 0.00 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0 3.3554e+09    0    1 3.2213e+09 3.3554e+09  4.16%     -    0s
H    0     0                     3.353526e+09 3.3554e+09  0.06%     -    0s
H    0     0                     3.354739e+09 3.3554e+09  0.02%     -    0s
     0     0 3.3554e+09    0    1 3.3547e+09 3.3554e+09  0.02%     -    0s
     0     0 3.3554e+09    0    1 3.3547e+09 3.3554e+09  0.02%     -    0s
     0     0 3.3554e+09    0    1 3.3547e+09 3.3554e+09  0.02%     -    0s
     0     0 3.3554e+09    0    1 3.3547e+09 3.3554e+09  0.02%     -    0s
     0     0 3.3554e+09    0    1 3.3547e+09 3.3554e+09  0.02%     -    0s
     0     2 3.3554e+09    0    1 3.3547e+09 3.3554e+09  0.02%     -    0s
H   60    64                     3.355181e+09 3.3554e+09  0.01%   1.4    0s

Cutting planes:
  Cover: 6

Explored 63 nodes (93 simplex iterations) in 0.01 seconds
```

```
Thread count was 16 (of 16 available processors)

Solution count 4: 3.35518e+09 3.35474e+09 3.35353e+09 3.22129e+09

Optimal solution found (tolerance 1.00e-04)
Best objective 3.355181244000e+09, best bound 3.355443192000e+09, gap 0.0078%
[11, 10, 0, 9, 6, 2, 12, 14, 7, 18, 4, 19, 15]
Gurobi Optimizer version 9.0.3 build v9.0.3rc0 (linux64)
Optimize a model with 2 rows, 24 columns and 37 nonzeros
Model fingerprint: 0xbd62c3f4
Variable types: 0 continuous, 24 integer (24 binary)
Coefficient statistics:
  Matrix range     [1e+00, 4e+08]
  Objective range  [3e+08, 4e+08]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+01, 3e+09]
Warning: Model contains large matrix coefficients
Warning: Model contains large rhs
         Consider reformulating model or setting NumericFocus parameter
         to avoid numerical issues.
Found heuristic solution: objective 3.221291e+09
Presolve time: 0.00s
Presolved: 2 rows, 24 columns, 37 nonzeros
Variable types: 0 continuous, 24 integer (24 binary)

Root relaxation: objective 3.355443e+09, 5 iterations, 0.00 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0 3.3554e+09    0    1 3.2213e+09 3.3554e+09  4.16%     -    0s
H    0     0                     3.353526e+09 3.3554e+09  0.06%     -    0s
     0     0 3.3554e+09    0    1 3.3535e+09 3.3554e+09  0.06%     -    0s
H    0     0                     3.354935e+09 3.3554e+09  0.02%     -    0s
     0     0 3.3554e+09    0    1 3.3549e+09 3.3554e+09  0.02%     -    0s
     0     0 3.3554e+09    0    1 3.3549e+09 3.3554e+09  0.02%     -    0s
     0     0 3.3554e+09    0    1 3.3549e+09 3.3554e+09  0.02%     -    0s
     0     0 3.3554e+09    0    1 3.3549e+09 3.3554e+09  0.02%     -    0s
H    0     0                     3.355312e+09 3.3554e+09  0.00%     -    0s

Cutting planes:
  Cover: 4

Explored 1 nodes (9 simplex iterations) in 0.01 seconds
Thread count was 16 (of 16 available processors)

Solution count 4: 3.35531e+09 3.35494e+09 3.35353e+09 3.22129e+09
```

```
Optimal solution found (tolerance 1.00e-04)
Best objective 3.355312188000e+09, best bound 3.355443192000e+09, gap 0.0039%
```

```
␣
↪---------------------------------------------------------------------------

       KeyError                                  Traceback (most recent call␣
↪last)

       <ipython-input-81-1cfa5b15daac> in <module>()
        26     m.optimize()
        27     # appending to models
   ---> 28     models[idx]['model'] = m
        29     models[idx]['rt'] = m.Runtime
        30     models[idx]['nd'] = m.NodeCount


       KeyError: 50
```

[126]:
```
# original model runtime
original_runtime = 0.024396181106567383

# keeping ONLY constraints which reduce the runtime
kept_indexes = []
for idx, model in models.items():
    if model['rt'] < original_runtime:
        kept_indexes.append(idx)

# filtering indexes
new_indexes = {key:val for key,val in indexes.items() if key in kept_indexes}
```

[127]:
```
len(new_indexes)
```

[127]: 47

We have correctly identified about 100 constraints out of which 47 should theoretically reduce our general runtime

We repeat the model adding such constraints:

[128]:
```
def run_model_with_new_constraints(indexes_list):
    # objects
    n = 24

    # define k
    def k(v):
```

```
        return np.floor(np.log(2*v))

    # weight
    w = [2**(k(n) + n + 1) + 2**(k(n) + j) + 1 for j in range(1,n+1)]

    # value/profit
    r = [2**(k(n) + n + 1) + 2**(k(n) + j) + 1 for j in range(1,n+1)]

    # kanpsack's capacity
    b = sum([np.floor(w[j]/2) for j in range(n)])

    # defining the model
    m = Model('knapsack')

    # Binary variables
    x = m.addVars(range(n), name="x", vtype=GRB.BINARY)

    # adding new constraints
    for index_list in indexes_list:
        m.addConstr(quicksum(x[j] for j in index_list) <= len(index_list)-1)

    # Capacity constraints
    m.addConstr(quicksum(w[j]*x[j] for j in range(n)) <= b)

    # Objective
    obj = quicksum(r[j]*x[j] for j in range(n))
    m.setObjective(obj, GRB.MAXIMIZE)

    #optimize
    m.optimize()
    return m

m = run_model_with_new_constraints([sorted(x) for x in new_indexes.values()])
```

```
Gurobi Optimizer version 9.0.3 build v9.0.3rc0 (linux64)
Optimize a model with 48 rows, 24 columns and 612 nonzeros
Model fingerprint: 0xa22b8a47
Variable types: 0 continuous, 24 integer (24 binary)
Coefficient statistics:
  Matrix range     [1e+00, 4e+08]
  Objective range  [3e+08, 4e+08]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+01, 3e+09]
Warning: Model contains large matrix coefficients
Warning: Model contains large rhs
         Consider reformulating model or setting NumericFocus parameter
         to avoid numerical issues.
```

```
Found heuristic solution: objective 3.221291e+09
Presolve time: 0.00s
Presolved: 48 rows, 24 columns, 612 nonzeros
Variable types: 0 continuous, 24 integer (24 binary)

Root relaxation: objective 3.355443e+09, 10 iterations, 0.00 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0 3.3554e+09    0    1 3.2213e+09 3.3554e+09  4.16%     -    0s
H    0     0                     3.354398e+09 3.3554e+09  0.03%     -    0s
     0     0 3.3554e+09    0    1 3.3544e+09 3.3554e+09  0.03%     -    0s
     0     0 3.3554e+09    0    3 3.3544e+09 3.3554e+09  0.03%     -    0s
*    0     0               0    3.355410e+09 3.3554e+09  0.00%     -    0s

Cutting planes:
  Cover: 3

Explored 1 nodes (21 simplex iterations) in 0.01 seconds
Thread count was 16 (of 16 available processors)

Solution count 3: 3.35541e+09 3.3544e+09 3.22129e+09

Optimal solution found (tolerance 1.00e-04)
Best objective 3.355410444000e+09, best bound 3.355410444000e+09, gap 0.0000%
```

[129]:
```
original_runtime - m.Runtime
```

[129]:
```
0.01208806037902832
```

we can see the runtime has been reduced in 0.01208806037902832 seconds, however, this can be improved, perhaps we should use significantly less constraints, maybe we can choose a few hand picked constraints which should reduce our runtime more than just forcefully adding 41 new constraints to the model, which, sure, improves our runtime but seems to be less than ideal

[146]:
```
# keeping ONLY constraints which reduce the runtime significantly
kept_indexes = [idx for idx,model in models.items() if model['rt'] <␣
 ↪original_runtime/2.5]

# filtering indexes
new_indexes = {key:val for key,val in indexes.items() if key in kept_indexes}
```

[156]:
```
new_indexes # we pick the inequality containing the following xjs
```

[156]:
```
{23: [1, 6, 7, 8, 9, 11, 12, 16, 18, 19, 22, 23],
 34: [2, 7, 9, 11, 13, 15, 16, 17, 18, 19, 21, 22, 23],
 35: [0, 1, 3, 4, 6, 9, 12, 14, 16, 17, 21, 23],
```

```
 36: [1, 2, 4, 5, 7, 8, 9, 12, 16, 21, 22, 23],
 38: [1, 4, 6, 7, 8, 10, 11, 12, 15, 20, 21, 23],
 39: [0, 1, 6, 7, 9, 10, 12, 13, 17, 21, 22, 23],
 41: [0, 1, 3, 6, 7, 10, 12, 13, 14, 21, 22, 23]}
```

which corresponds to the following constraints: $w_1 + w_6 + w_7 + w_8 + w_9 + w_{11} + w_{12} + w_{16} + w_{18} + w_{19} + w_{22} + w_{23} \leq 11$

$w_2 + w_7 + w_9 + w_{11} + w_{13} + w_{15} + w_{16} + w_{17} + w_{18} + w_{19} + w_{21} + w_{22} + w_{23} \leq 12$

$w_0 + w_1 + w_3 + w_4 + w_6 + w_9 + w_{12} + w_{14} + w_{16} + w_{17} + w_{21} + w_{23} \leq 11$

$w_1 + w_2 + w_4 + w_5 + w_7 + w_8 + w_9 + w_{12} + w_{16} + w_{21} + w_{22} + w_{23} \leq 11$

$w_1 + w_4 + w_6 + w_7 + w_8 + w_{10} + w_{11} + w_{12} + w_{15} + w_{20} + w_{21} + w_{23} \leq 11$

$w_0 + w_1 + w_6 + w_7 + w_9 + w_{10} + w_{12} + w_{13} + w_{17} + w_{21} + w_{22} + w_{23} \leq 11$

$w_0 + w_1 + w_3 + w_6 + w_7 + w_{10} + w_{12} + w_{13} + w_{14} + w_{21} + w_{22} + w_{23} \leq 11$

[148]:
```
m = run_model_with_new_constraints([sorted(x) for x in new_indexes.values()])
```

```
Gurobi Optimizer version 9.0.3 build v9.0.3rc0 (linux64)
Optimize a model with 8 rows, 24 columns and 109 nonzeros
Model fingerprint: 0xd1e53754
Variable types: 0 continuous, 24 integer (24 binary)
Coefficient statistics:
  Matrix range     [1e+00, 4e+08]
  Objective range  [3e+08, 4e+08]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+01, 3e+09]
Warning: Model contains large matrix coefficients
Warning: Model contains large rhs
         Consider reformulating model or setting NumericFocus parameter
         to avoid numerical issues.
Found heuristic solution: objective 3.221291e+09
Presolve time: 0.00s
Presolved: 8 rows, 24 columns, 109 nonzeros
Variable types: 0 continuous, 24 integer (24 binary)

Root relaxation: objective 3.355443e+09, 11 iterations, 0.00 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0 3.3554e+09    0    1 3.2213e+09 3.3554e+09  4.16%     -    0s
H    0     0                     3.304523e+09 3.3554e+09  1.54%     -    0s
H    0     0                     3.354854e+09 3.3554e+09  0.02%     -    0s
*    0     0               0     3.355410e+09 3.3554e+09  0.00%     -    0s
```

```
Cutting planes:
  Cover: 1

Explored 1 nodes (15 simplex iterations) in 0.01 seconds
Thread count was 16 (of 16 available processors)

Solution count 4: 3.35541e+09 3.35485e+09 3.30452e+09 3.22129e+09

Optimal solution found (tolerance 1.00e-04)
Best objective 3.355410444000e+09, best bound 3.355410444000e+09, gap 0.0000%
```

[149]:
```
original_runtime - m.Runtime
```

[149]: 0.013563156127929688

[153]:
```
(original_runtime - m.Runtime)/(original_runtime-0.01208806037902832)
```

[153]: 1.1019680768634743

We can see an improvement of about 0.013563156127929688 seconds vs the model we first ran. Which compared to using all 47 found constraints that improve our time is about a 10% improvement and vs our original model we see an improvement of about 125.2%, so about twice as fast.

[119]:
```
print(m.NodeCount)
print(m.Runtime)
print(m.ObjVal)
```

```
1.0
0.00957798957824707
3355410444.0
```

0.8.1 - Our optimal solution (using a tolerance of 1×10^{-4}) is $\sim 3.355410444 \times 10^{9}$

0.8.2 - The solution time is 0.00957798957824707 seconds

0.8.3 - The number of nodes explored is of 1 node

[ ]:
```

```