Owl delivery service

Ollivander, the famous wand maker of Diagon Alley, has a logistics problem. He produces wands of different types for which he must arrange delivery. For each type, he knows the quantity that need to be delivered. To make deliveries, Ollivander may use a set of owls, and each owl has a known capacity (maximum number of wands the owl can carry). Wands are highly volatile, and an owl will explode mid-flight if it carries two wands of the same type. The code that describes the instance is given below.

In [5]:
```
import sys
import pandas as pd
import numpy as np

from gamspy import (
    Container,Set,Alias,Parameter,Variable,Equation,Model,Problem,Sense,Opti
    Domain,Number,Sum,Product,Smax,Smin,Ord,Card,SpecialValues,
    ModelStatus,SolveStatus,
)

options = Options(equation_listing_limit=0)
m = Container(options=options)
```

Data

In [6]:
```
nodes = Set(m,'nodes',records=['s','t']+[f"w{ind+1}" for ind in range(7)]+[f
i = Alias(m,'i',nodes)
j = Alias(m,'j',nodes)

wands = Set(m,'wands',domain=i,records=[f"w{ind+1}" for ind in range(7)])
owls = Set(m,'owls',domain=i,records=[f"o{ind+1}" for ind in range(5)])

s = Set(m,'s',domain=i,is_singleton=True,description='sources',records=['s']
t = Set(m,'t',domain=i,is_singleton=True,description='sinks',records=['t'])

OwlCapacity = Parameter(m,'OwlCapacity',domain=owls,records=np.array([6, 4,
WandsToDeliver = Parameter(m,'WandsToDeliver',domain=wands,records=3*np.ones
```

Model

In [7]:
```
# WRITE YOUR DATA AND MODEL HERE
arcs = Set(m, "arcs", domain=[nodes, i])
arcs[s, wands] = True
arcs[wands, owls] = True
arcs[owls, t] = True
```

```
bound = m.addParameter('u',domain=[nodes,i])
bound[s,wands] = WandsToDeliver[wands]
bound[wands,owls] = 1
bound[owls,t] = OwlCapacity[owls]

num = Variable(m, "num", type="positive", domain=[nodes, i])

balance_s = Equation(m, 'balance_s', domain=[nodes])
balance_s[nodes].where[~s[nodes] & ~t[nodes]] = Sum(arcs[nodes, j], num[node

num.up[arcs] = bound[arcs]

maxflow = Model(m,
    name="maxflow",
    equations=m.getEquations(),
    problem=Problem.LP,
    sense=Sense.MAX,
    objective=Sum(arcs[s, i], num[s, i])
)


maxflow.solve(solver='cplex',solver_options={'lpmethod': 3, 'netfind': 1, 'p
```

Out[7]:

	Solver Status	Model Status	Objective	Num of Equations	Num of Variables	Model Type	Solver	Solve Tim
0	Normal	OptimalGlobal	21	13	48	LP	CPLEX	0.00

Post Processing

In [8]:
```
print(f"Objective Function Value: {round(maxflow.objective_value, 4)}\n")
print(f"Number of equations: {maxflow.num_equations:.0f}\n")
print(f"Number of variables: {maxflow.num_variables:.0f}\n")

display(WandsToDeliver.toDense())
if maxflow.objective_value >= np.sum(WandsToDeliver.toDense()):
    print("Delivered all")
else:
    print("Some wands not delivered")
```

```
Objective Function Value: 21.0

Number of equations: 13

Number of variables: 48

array([3., 3., 3., 3., 3., 3., 3.])
Delivered all
```