Dragon Transport

Given your great Optimization Wizard training, the Ministry of Magic has asked that you transport 20 dragons from Romania to Hogwarts for the Triwizard tournament. The dragons will be transported on a route through five cities, with a choice of three different modes of transport between each of the pairs of cities on the route. The route to be followed is exactly:

1. Romania
2. Transylvania
3. Egypt
4. Godric's Hollow
5. Hogwarts

On each leg of the route, the dragons are to all be transported by Hogwarts Express (Train), Portkey, or Thestral. In any of the intermediate cities, it is possible to change the mode of transport, but you must use a single mode of transport for all the dragons between two consecutive cities. The following table lists the cost of transport in galleons per dragon between the pairs of cities:

Pairs of cities	1-2	2-3	3-4	4-5
Train	30	25	40	60
Portkey	25	40	45	50
Thestral	40	20	50	45

The next table shows the cost of changing the mode of transport in galleons/dragon. (This cost is independent of the location at which the change is made):

From/To	Train	Portkey	Thestral
Train	0	5	12
Portkey	8	0	10
Thestral	15	10	0

How should the transport be organized to minimize the cost? What is the minimum cost for transporting the 20 dragons?

In [2]:
```
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=100,absolute_optimality_gap=0,relat

cont = Container(options=options)
```

In [3]:

```
M = Set(cont,'M',records=['HogwartsExpress', 'Portkey', 'Thestral'])
M1 = cont.addAlias('M1', M)
M2 = cont.addAlias('M2', M)
M3 = cont.addAlias('M3', M)

L = Set(cont,'L',records=['1-2', '2-3', '3-4', '4-5'])
L1 = cont.addAlias('L1', L)

TravelCost = Parameter(cont, 'TravelCost', domain=[M, L], records = np.array



ChangeCost = Parameter(cont, 'ChangeCost', domain=[M, M], records = np.array



x = cont.addVariable('x','binary',domain=[M,L])

assign_path = cont.addEquation('assign_path', domain=L)
assign_path[L] = Sum(M, x[M,L]) == 1

z = cont.addVariable('z','binary',domain=[M1,M2, L])

trix_3 = m=cont.addEquation('trix_3', domain=[M1,M2, L])
trix_3[M1, M2, L].where[Ord(L) < Card(L)]= z[M1,M2, L] >= x[M1,L] + x[M2,L.l

trix_4 = cont.addEquation('trix_4', domain=[M1,M2, L])
trix_4[M1, M2, L].where[Ord(L) < Card(L)]= x[M1,L] >= z[M1,M2,L]

dragon = cont.addModel('dragon',
    equations=cont.getEquations(),
    problem=Problem.MIP,
    sense=Sense.MIN,
    objective=Sum([M1,M2,L], ChangeCost[M1, M2]*z[M1,M2,L]) + Sum([M, L], Tr
)


dragon.solve()
```

Out[3]:

	Solver Status	Model Status	Objective	Num of Equations	Num of Variables	Model Type	Solver	Solve Tim
0	Normal	OptimalGlobal	150	59	46	MIP	CPLEX	0.05

In [4]:
```
# Solution could be reported using the following:
print(f'Cost = {dragon.objective_value}')
display(x.pivot())
elements = x.records['level'] > 0
display(x.records.loc[elements,['M', 'L']].set_index('L'))
```

```
Cost = 150.0
```

	1-2	2-3	3-4	4-5
HogwartsExpress	0.0	0.0	0.0	0.0
Portkey	1.0	0.0	0.0	0.0
Thestral	0.0	1.0	1.0	1.0

	M
L	
1-2	Portkey
2-3	Thestral
3-4	Thestral
4-5	Thestral