In [2]:

```python
import numpy as np
```

In [3]:

```python
NUM_FEATS = 4
NUM_STATES = 4
NUM_ACTIONS = 2
GAMMA = 0.9
```

In [4]:

```python
# Expected rewards for each state action pair
R = np.array([-40, -80, -160, -380, -100, -100, -100, -100])
```

In [5]:

```python
c = np.ones(NUM_STATES)
PHI = np.eye(NUM_FEATS)
P_0 = np.array(
    [
        [0.5, 0.5, 0, 0],
        [0, 0.5, 0.5, 0],
        [0, 0, 0.5, 0.5],
        [0.5, 0, 0, 0.5]
    ]
)
P_1 = np.concatenate((np.ones((4,1)), np.zeros((4, 3))), axis=1)
A = np.concatenate((np.eye(4), np.eye(4))) - GAMMA * np.concatenate((P_0, P_1))
c, PHI, A
```

Out[5]:

```
(array([1., 1., 1., 1.]),
 array([[1., 0., 0., 0.],
        [0., 1., 0., 0.],
        [0., 0., 1., 0.],
        [0., 0., 0., 1.]]),
 array([[ 0.55, -0.45,  0.  ,  0.  ],
        [ 0.  ,  0.55, -0.45,  0.  ],
        [ 0.  ,  0.  ,  0.55, -0.45],
        [-0.45,  0.  ,  0.  ,  0.55],
        [ 0.1 ,  0.  ,  0.  ,  0.  ],
        [-0.9 ,  1.  ,  0.  ,  0.  ],
        [-0.9 ,  0.  ,  1.  ,  0.  ],
        [-0.9 ,  0.  ,  0.  ,  1.  ]]))
```

In [6]:

```python
PHI_HAT = np.eye(8)
R_TILDE = np.concatenate((np.diag(R[0:4]), np.diag(R[4:])), axis=0)
PHI_HAT, R_TILDE
```

Out[6]:

```
(array([[1., 0., 0., 0., 0., 0., 0., 0.],
        [0., 1., 0., 0., 0., 0., 0., 0.],
        [0., 0., 1., 0., 0., 0., 0., 0.],
        [0., 0., 0., 1., 0., 0., 0., 0.],
        [0., 0., 0., 0., 1., 0., 0., 0.],
        [0., 0., 0., 0., 0., 1., 0., 0.],
        [0., 0., 0., 0., 0., 0., 1., 0.],
        [0., 0., 0., 0., 0., 0., 0., 1.]]),
 array([[ -40,    0,    0,    0],
        [   0,  -80,    0,    0],
        [   0,    0, -160,    0],
        [   0,    0,    0, -380],
        [-100,    0,    0,    0],
        [   0, -100,    0,    0],
        [   0,    0, -100,    0],
        [   0,    0,    0, -100]]))
```

In [7]:

```
import gurobipy as gp
from gurobipy import GRB

m = gp.Model('lp')

w_z = m.addMVar(8, lb=float('-inf'))
c = c.T @ PHI
m.setMObjective(None, c=c, constant=0.0, xc=w_z[0:4], sense=GRB.MINIMIZE)

m.addConstr(A @ PHI @ w_z[0:4] >= PHI_HAT @ R_TILDE @ w_z[4:])
m.addConstr(sum(w_z[4:]) == 1)
m.addConstrs(w_z[i] >= 0  for i in range(4, 8))

m.optimize()
```

```
Academic license - for non-commercial use only - expires 2021-05-28
Using license file C:\Users\Brendan\gurobi.lic
Gurobi Optimizer version 9.1.1 build v9.1.1rc0 (win64)
Thread count: 4 physical cores, 8 logical processors, using up to 8 threads
Optimize a model with 13 rows, 8 columns and 31 nonzeros
Model fingerprint: 0xa4996764
Coefficient statistics:
  Matrix range     [1e-01, 4e+02]
  Objective range  [1e+00, 1e+00]
  Bounds range     [0e+00, 0e+00]
  RHS range        [1e+00, 1e+00]
Presolve removed 8 rows and 0 columns
Presolve time: 0.04s
Presolved: 5 rows, 8 columns, 30 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0   -3.2388430e+03   1.774277e+01   0.000000e+00      0s
       4   -9.4815014e+02   0.000000e+00   0.000000e+00      0s

Solved in 4 iterations and 0.07 seconds
Optimal objective -9.481501404e+02
```

In [8]:

```
w_opt = np.array([v.x for v in m.getVars()])[0:4]
```

In [9]:

```
PHI @ w_opt
```

Out[9]:

```
array([-236.00208435, -267.46902893, -229.00943   , -215.66959708])
```