min_cost_flow

October 3, 2016

In [16]:
```
#Sergiu Buciumas
#Model the problem as a
#minimum cost network flow problem. First find a BFS using auxiliary netwo
#using the network simplex method.


from gurobipy import *

# Model
model = Model("costflow")
#model.setParam(GRB.param.Method, 0)

# Create decision variables
x12 = model.addVar(name='x12')
x13 = model.addVar(name='x13')
x15 = model.addVar(name='x15')
x23 = model.addVar(name='x23')
x35 = model.addVar(name='x35')
x24 = model.addVar(name='x24')
x34 = model.addVar(name='x34')
x56 = model.addVar(name='x56')
x46 = model.addVar(name='x46')

# Update model to integrate new variables
model.update()

#model objective
model.setObjective(3 * x12 + 3 * x13 + 3 * x15 + 2 * x23 + 2 * x35 + 4 * x
# The objective is to maximize (this is redundant now, but it will overwri

# Add constraints to the model
model.addConstr(3*x12 + x13 + x15 ,GRB.EQUAL, 1, "c1")
model.addConstr(x23 + x24 - x12 ,GRB.EQUAL, 3, "c2")
model.addConstr(x34 + x35 - x13 - x23 ,GRB.EQUAL, 0,  "c3")
model.addConstr(x46 - x34 - x24 ,GRB.EQUAL, 0,  "c4")
model.addConstr(x56 - x15 - x35 ,GRB.EQUAL, 0,  "c5")
model.addConstr(-x56 - x46 ,GRB.EQUAL, -4,  "c6")
```

```
        # Solve
        model.optimize()
        data = []
        variable = []
        # Let's print the solution
        for v in model.getVars():
            print v.varName, v.x
            data.append(v.varName)
            variable.append(v.x)

        dictionary = dict(zip(data, variable))
        print dictionary
```

```
Optimize a model with 6 rows, 9 columns and 18 nonzeros
Coefficient statistics:
  Matrix range    [1e+00, 3e+00]
  Objective range [1e+00, 4e+00]
  Bounds range    [0e+00, 0e+00]
  RHS range       [1e+00, 4e+00]
Presolve removed 2 rows and 3 columns
Presolve time: 0.00s
Presolved: 4 rows, 6 columns, 12 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    1.3000000e+01   7.333333e+00   0.000000e+00      0s
       2    1.9000000e+01   0.000000e+00   0.000000e+00      0s

Solved in 2 iterations and 0.01 seconds
Optimal objective  1.900000000e+01
x12 0.0
x13 0.0
x15 1.0
x23 3.0
x35 3.0
x24 0.0
x34 0.0
x56 4.0
x46 0.0
{'x46': 0.0, 'x56': 4.0, 'x34': 0.0, 'x35': 3.0, 'x24': 0.0, 'x12': 0.0, 'x13': 0.0
```

```
In [21]: #test the dictionary value access
         print(dictionary["x56"])

         #calculate minimum cost flow based on LP minimization.
         cost_flow = 3 * dictionary['x12'] + 3 * dictionary['x13'] + 3 * dictionary
         cost_flow
```

```
4.0
```

Out[21]: 19.0

In [22]:
```
# Let's print the dual variables as well
for c in model.getConstrs():
    print c.constrName, c.pi
```

```
c1 2.0
c2 3.0
c3 1.0
c4 1.0
c5 -1.0
c6 -2.0
```

In [ ]: