

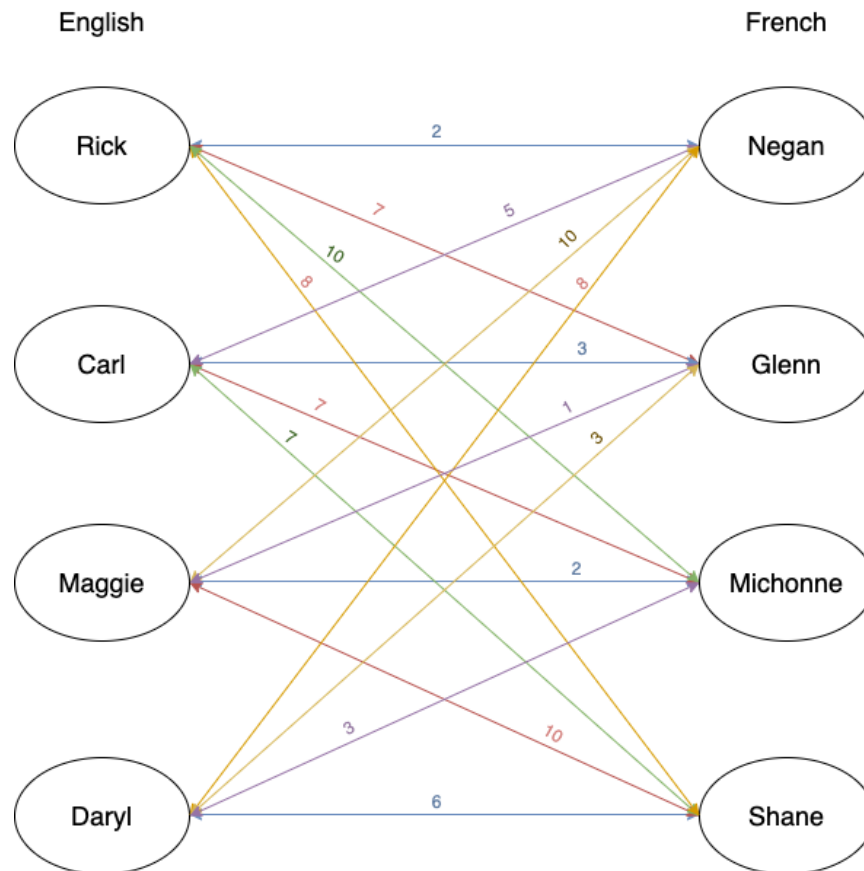
Homework 3 Advanced Analytics and Metaheuristics

Group 1: David Garza, Garrison Kleman, Nicholas Jacob, Hannah Jensen

February 23, 2024

1. Team Building

- (a) English and French. We give the English a positive weight and French a negative weight. We have the costs on the arcs. Here we slightly changed the model of the MCNFP by introducing a negative to the objective function. This cause the minimizer to return a maximum.



Here is our model file:

```
# AMPL model for the Minimum Cost Network Flow Problem
#
# By default, this model assumes that b[i] = 0, c[i,j] = 0,
# l[i,j] = 0 and u[i,j] = Infinity.
#
# Parameters not specified in the data file will get their default values.
reset;

options solver cplex;

set NODES;                                # nodes in the network
set ARCS within {NODES, NODES};           # arcs in the network

param b {NODES} default 0;                # supply/demand for node i
param c {ARCS} default 0;                 # cost of one of flow on arc(i,j)
param l {ARCS} default 0;                 # lower bound on flow on arc(i,j)
```

```

param u {ARCS} default Infinity; # upper bound on flow on arc(i,j)

var x {ARCS};                    # flow on arc (i,j)

minimize cost: sum{(i,j) in ARCS} (-1)* c[i,j] * x[i,j]; #objective: minimize arc

# Flow Out(i) - Flow In(i) = b(i)

subject to flow_balance {i in NODES}:
sum{j in NODES: (i,j) in ARCS} x[i,j] - sum{j in NODES: (j,i) in ARCS} x[j,i] = b[i];

subject to capacity {(i,j) in ARCS}: l[i,j] <= x[i,j] <= u[i,j];

data group1_HW3_p1anjan.dat;

solve;

display x;

Here is our data file:

data;

set NODES := 1, 2, 3, 4, 5, 6, 7, 8;

/*
1: Rick english
2: Negan french
3: Carl english
4: Glenn french
5: Maggie english
6: Michonne french
7: Daryl english
8: Shane french
*/

set ARCS :=
      (1,*)      2      4      6      8
      (3,*)      2      4      6      8
      (5,*)      2      4      6      8
      (7,*)      2      4      6      8
      ;

param c:  1      2      3      4      5      6      7      8 :=
      1      .      2      .      7      .      10      .      9

```

2
3	.	5	.	3	.	7	.	7
4
5	.	10	.	1	.	2	.	10
6
7	.	8	.	3	.	3	.	6
8	;

```
param b:=
    1      1
    2     -1
    3      1
    4     -1
    5      1
    6     -1
    7      1
    8     -1;
```

Here is our output:

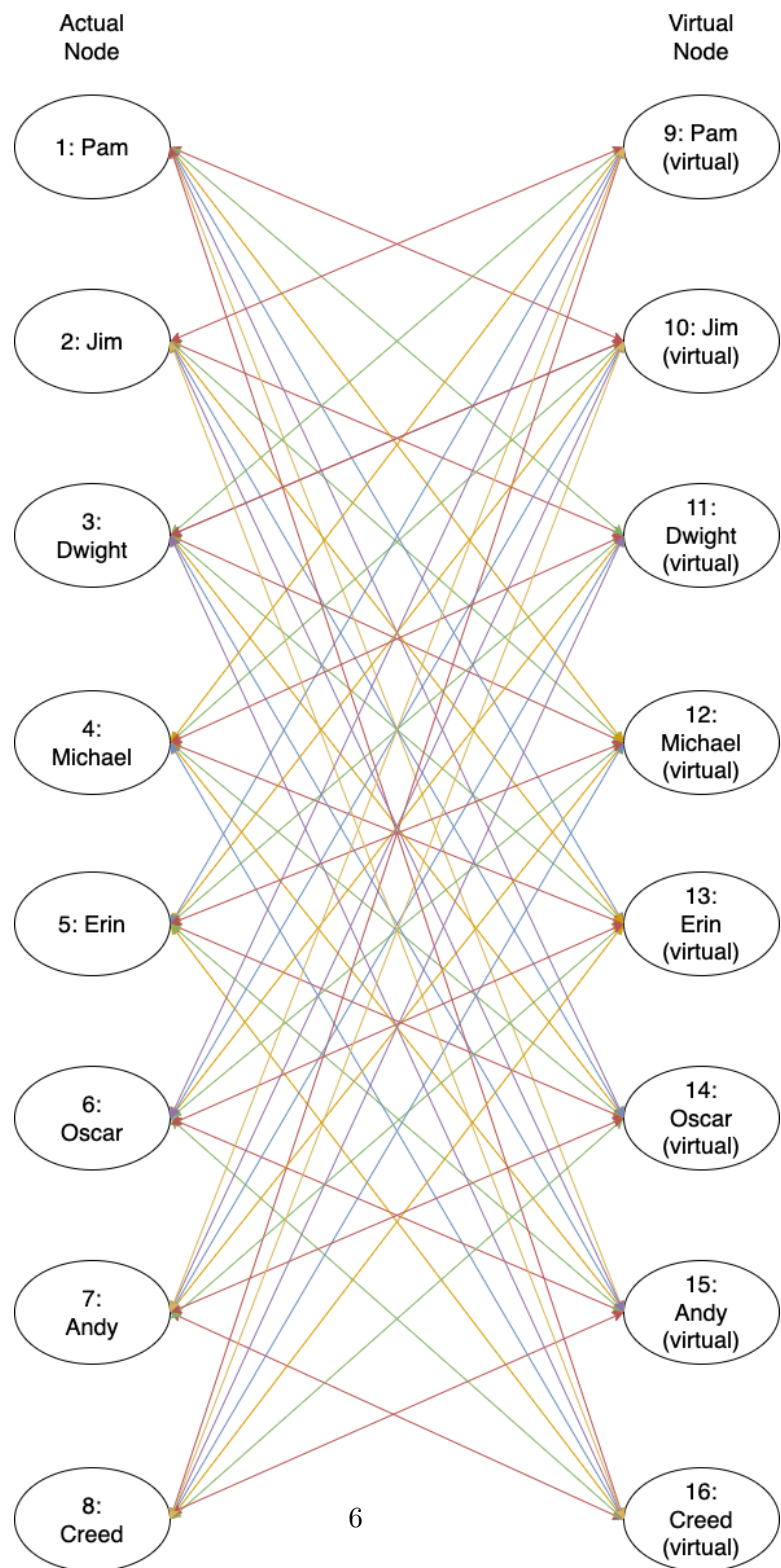
```
ampl: model group1_HW3_planj.mod
CPLEX 20.1.0.0: optimal solution; objective -32
2 dual simplex iterations (0 in phase I)
x :=
1 2  0
1 4  1
1 6  0
1 8  0
3 2  0
3 4  0
3 6  1
3 8  0
5 2  0
5 4  0
5 6  0
5 8  1
7 2  1
7 4  0
7 6  0
7 8  0
;
```

For our example we see person 1 and 4 paired, 3 and 6 paired, etc. We see this as fully generalizable. This MCNFP is always balanced we have equal number of English and French so we always have even pairs with no left over in this model when solved.

(b) Office Team Building

We attempt to build a model with the same idea as above. We assign scores of 2 as not compatible, 7 with compatible and 10 with highly compatible. We chose these values to represent the

spread between not being compatible and being compatible as being ‘large’. We attempted to create a virtual copy of each person and attach them to everyone else (not attached to yourself) and put those costs on the flow. This is represented in our document below. We end up running this with 8 people and getting a working solution but when we expanded to 16 we got some weirdness.



Here is our model file:

```
# AMPL model for the Minimum Cost Network Flow Problem
#
# By default, this model assumes that b[i] = 0, c[i,j] = 0,
# l[i,j] = 0 and u[i,j] = Infinity.
#
# Parameters not specified in the data file will get their default values.
reset;

options solver cplex;

set NODES;                # nodes in the network
set ARCS within {NODES, NODES}; # arcs in the network

param b {NODES} default 0;    # supply/demand for node i
param c {ARCS} default 0;    # cost of one of flow on arc(i,j)
param l {ARCS} default 0;    # lower bound on flow on arc(i,j)
param u {ARCS} default Infinity; # upper bound on flow on arc(i,j)

var x {ARCS};                # flow on arc (i,j)

minimize cost: sum{(i,j) in ARCS} (-1)* c[i,j] * x[i,j]; #objective: minimize arc flow cost

# Flow Out(i) - Flow In(i) = b(i)

subject to flow_balance {i in NODES}:
sum{j in NODES: (i,j) in ARCS} x[i,j] - sum{j in NODES: (j,i) in ARCS} x[j,i] = b[i];

subject to capacity {(i,j) in ARCS}: l[i,j] <= x[i,j] <= u[i,j];

data group1_HW3_plbnj2.dat;

solve;

display x;

printf "Total Compatibility Score: %f\n", cost;
printf "Pairs:\n";
for {(i, j) in ARCS: x[i,j] > 0} {
    printf "%s - %s\n", i, j;
}
```

Here is our data file:

```
data;

set NODES := 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
            v1 v2 v3 v4 v5 v6 v7 v8 v9 v10 v11 v12 v13 v14 v15 v16;

/*
1: Pam
2: Jim
3: Dwight
4: Michael
5: Erin
6: Oscar
7: Andy
8: Creed
9: Kevin
10: Angela
11: Phyllis
12: Holly
13: Kelly
14: Stanley
15: Ryan
16: Meredith
*/

/*
2 = Not Compatible
7 = Compatible
10 = Highly Compatible
*/

set ARCS :=
```

```

(1,*)          v2          v3          v4          v5          v6          v7          v8          v9          v10          v11          v12
(2,*)          v1          v3          v4          v5          v6          v7          v8          v9          v10          v11          v12
(3,*)          v1          v2          v4          v5          v6          v7          v8          v9          v10          v11          v12
(4,*)          v1          v2          v3          v5          v6          v7          v8          v9          v10          v11          v12
(5,*)          v1          v2          v3          v4          v5          v6          v7          v8          v9          v10          v11          v12
(6,*)          v1          v2          v3          v4          v5          v6          v7          v8          v9          v10          v11          v12
(7,*)          v1          v2          v3          v4          v5          v6          v7          v8          v9          v10          v11          v12
(8,*)          v1          v2          v3          v4          v5          v6          v7          v8          v9          v10          v11          v12
(9,*)          v1          v2          v3          v4          v5          v6          v7          v8          v9          v10          v11          v12
(10,*)         v1          v2          v3          v4          v5          v6          v7          v8          v9          v10          v11          v12
(11,*)         v1          v2          v3          v4          v5          v6          v7          v8          v9          v10          v11          v12
(12,*)         v1          v2          v3          v4          v5          v6          v7          v8          v9          v10          v11          v12
(13,*)         v1          v2          v3          v4          v5          v6          v7          v8          v9          v10          v11          v12
(14,*)         v1          v2          v3          v4          v5          v6          v7          v8          v9          v10          v11          v12
(15,*)         v1          v2          v3          v4          v5          v6          v7          v8          v9          v10          v11          v12
(16,*)         v1          v2          v3          v4          v5          v6          v7          v8          v9          v10          v11          v12
;

```

```

param c:          v1          v2          v3          v4          v5          v6          v7          v8          v9          v10          v11          v12          v13          v14          v15          v16 :=
1          .          10          7          7          7          7          7          2          7          2          7          7          7          2          2          2
2          .          10          .          7          7          7          7          2          7          2          7          7          2          7          7          2
3          7          7          .          7          2          2          2          2          2          2          10          2          2          2          7          2
4          7          7          7          .          7          2          2          10          2          2          2          2          10          2          2          7          2
5          7          7          2          7          .          7          10          2          7          7          7          7          7          2          7          2
6          7          7          2          2          7          .          2          2          2          7          7          7          2          2          2          2
7          7          7          2          10          10          2          .          2          7          2          2          7          2          2          7          2
8          2          2          2          2          2          2          2          .          2          2          2          2          2          2          2          10
9          7          7          2          2          7          2          7          2          .          2          7          2          2          7          2          7
10         2          2          10          2          7          7          2          2          2          .          2          2          2          2          2          2
11         7          7          2          2          7          7          2          2          7          2          .          7          7          10          2          7
12         7          7          2          10          7          7          7          2          2          2          7          .          7          2          2          2
13         7          2          2          2          7          2          2          2          2          2          7          7          .          2          10          2
14         2          7          2          2          2          2          2          2          7          2          10          2          2          .          2          2
15         2          7          7          7          7          2          7          2          2          2          2          2          10          2          .          2
16         2          2          2          2          2          2          2          10          7          2          7          2          2          2          2          .;

```

```

param b:=
1          1
2          1
3          1
4          1
5          1
6          1
7          1
8          1
9          1
10         1
11         1
12         1
13         1
14         1
15         1
16         1
v1         -1
v2         -1
v3         -1
v4         -1
v5         -1
v6         -1
v7         -1
v8         -1
v9         -1
v10        -1
v11        -1
v12        -1
v13        -1
v14        -1
v15        -1
v16        -1;

```

Here is our output:


```

ampl: model group1_HW3_plbnj.mod
CPLEX 20.1.0.0: optimal solution; objective -148
126 dual simplex iterations (40 in phase I)
x [*,*]
:   v1 v10 v11 v12 v13 v14 v15 v16 v2 v3 v4 v5 v6 v7 v8 v9   :=
1   .  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1
2   1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
3   0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0
4   0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0
5   0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0
6   0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0
7   0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0
8   0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0
9   0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0
10  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0
11  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0
12  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0
13  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0
14  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0
15  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0
16  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0
;

Total Compatibility Score: -148.000000
Pairs:
1 - v9
2 - v1
3 - v10
4 - v7
5 - v6
6 - v12
7 - v5
8 - v16
9 - v2
10 - v3
11 - v14
12 - v4
13 - v15
14 - v11
15 - v13
16 - v8
ampl:

```

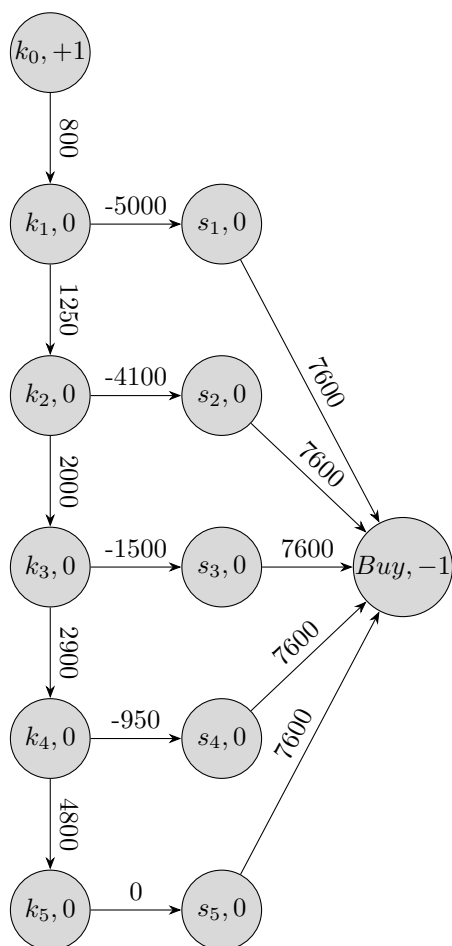
We see weirdness in our output here. There are thruple couples and quintuplet room assignments. We see 1 paired with 9 but 9 paired with 2 and then completing the loop with 2 paired with 1. There is another group of 5 assigned to rooms that will not work. We find this solution to not be generalizable even though in theory it might have worked. The optimizer did too good of job without being able to distinguish that we needed the pairings to coincide. If 1 was matched with v9 we needed 9 to be matched with v1. This was not necessarily the case.

2. Outdoor Grilling

Below is our model diagram.

Maintain

Sell



Here is our model file:

```

# AMPL model for the Minimum Cost Network Flow Problem
#
# By default, this model assumes that b[i] = 0, c[i,j] = 0,
# l[i,j] = 0 and u[i,j] = Infinity.
#
# Parameters not specified in the data file will get their default values.
reset;

```

```

options solver cplex;

set NODES;                                # nodes in the network
set ARCS within {NODES, NODES};           # arcs in the network

param b {NODES} default 0;                 # supply/demand for node i
param c {ARCS} default 0;                 # cost of one of flow on arc(i,j)
param l {ARCS} default 0;                 # lower bound on flow on arc(i,j)
param u {ARCS} default Infinity;          # upper bound on flow on arc(i,j)

var x {ARCS};                             # flow on arc (i,j)

minimize cost: sum{(i,j) in ARCS} c[i,j] * x[i,j]; #objective: minimize arc flow cost

# Flow Out(i) - Flow In(i) = b(i)

subject to flow_balance {i in NODES}:
sum{j in NODES: (i,j) in ARCS} x[i,j] - sum{j in NODES: (j,i) in ARCS} x[j,i] = b[i];

subject to capacity {(i,j) in ARCS}: l[i,j] <= x[i,j] <= u[i,j];

data group1_HW3_p2.dat;

solve;

display x;

Here is our data file:

set NODES := m0, m1, m2, m3, m4, m5,
              s1, s2, s3, s4, s5,
              b1;

set ARCS := (m0, m1), (m1, m2), (m2, m3), (m3, m4), (m4, m5),
              (m1, s1), (m2, s2), (m3, s3), (m4, s4), (m5, s5),
              (s1, b1), (s2, b1), (s3, b1), (s4, b1), (s5, b1);

param: b:=
      m0 1
      b1 -1;

param:      c l u:=
      [m0, m1]      800      .      .

```

[m1, m2]	1250	.	.
[m2, m3]	2000	.	.
[m3, m4]	2900	.	.
[m4, m5]	4800	.	.
[m1, s1]	-5000	.	.
[m2, s2]	-4100	.	.
[m3, s3]	-1500	.	.
[m4, s4]	-950	.	.
[m5, s5]	0	.	.
[s1, b1]	7600	.	.
[s2, b1]	7600	.	.
[s3, b1]	7600	.	.
[s4, b1]	7600	.	.
[s5, b1]	7600	.	.

Here is our output:

```

AMPL
ampl: model group1_HW3_p2.mod
CPLEX 20.1.0.0: optimal solution; objective 3400
0 dual simplex iterations (0 in phase I)
x :=
m0 m1 1
m1 m2 0
m1 s1 1
m2 m3 0
m2 s2 0
m3 m4 0
m3 s3 0
m4 m5 0
m4 s4 0
m5 s5 0
s1 b1 1
s2 b1 0
s3 b1 0
s4 b1 0
s5 b1 0
;

ampl:

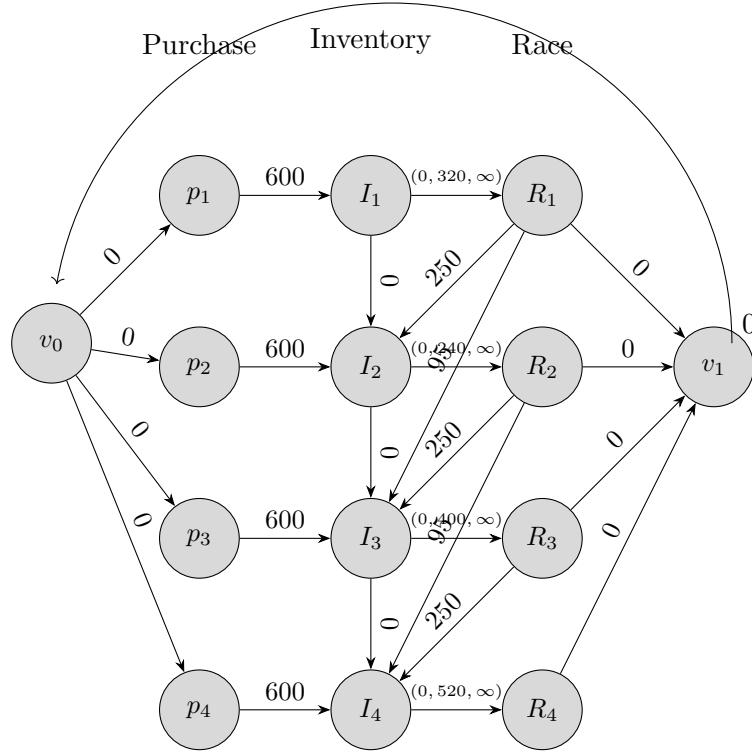
```

Buy a new one every year seems to be our cost effective method. It would cost us \$3400 each year to be a grill master.

3. Race Car Tires

Here is my flow model: All nodes have zero b costs are displayed on arcs. If minimums are needed, the ordered triple represents $(cost, lowerLimit, upperLimit)$.

The v nodes are virtual to balance the flow.



Here is our model file:

```
# AMPL model for the Minimum Cost Network Flow Problem
#
# By default, this model assumes that b[i] = 0, c[i,j] = 0,
# l[i,j] = 0 and u[i,j] = Infinity.
#
# Parameters not specified in the data file will get their default values.
reset;

options solver cplex;

set NODES;                                # nodes in the network
set ARCS within {NODES, NODES};           # arcs in the network

param b {NODES} default 0;                 # supply/demand for node i
param c {ARCS} default 0;                  # cost of one of flow on arc(i,j)
param l {ARCS} default 0;                  # lower bound on flow on arc(i,j)
param u {ARCS} default Infinity;           # upper bound on flow on arc(i,j)
```

```

var x {ARCS};                                # flow on arc (i,j)

minimize cost: sum{(i,j) in ARCS} c[i,j] * x[i,j]; #objective: minimize arc flow cost

# Flow Out(i) - Flow In(i) = b(i)

subject to flow_balance {i in NODES}:
sum{j in NODES: (i,j) in ARCS} x[i,j] - sum{j in NODES: (j,i) in ARCS} x[j,i] = b[i];

subject to capacity {(i,j) in ARCS}: l[i,j] <= x[i,j] <= u[i,j];

data group1_HW3_p3.dat;

solve;

display x;

```

Here is our data file:

```

#MCNFP Problem - data file for problem instance
#Charles Nicholson, ISE 5113, 2015

#use with MCNFP.txt model
#note: default arc costs and lower bounds are 0
#      default arc upper bounds are infinity
#      default node requirements are 0

set NODES :=          v0, p1, p2,p3,p4, i1,i2,i3,i4,r1,r2,r3,r4,v1 ;

set ARCS := (v0,p1),(v0,p2),(v0,p3),(v0,p4), #start the flow
            (p1,i1),(p2,i2),(p3,i3),(p4,i4), #purchase new tires each race
            (i1,r1),(i2,r2),(i3,r3),(i4,r4), #move inventory to race
            (r1,v1),(r2,v1),(r3,v1),(r4,v1), #move spent tires not fixed to
            (i1,i2),(i2,i3),(i3,i4), #move unused inventory
            (r1,i2),(r1,i3), #race 1 quick and slow fix
            (r2,i3),(r2,i4), #race 2 quick and slow fix
            (r3,i4), #race 3 quick fix
            (v1,v0) #move from virtual to virtual to complete flow
            ;

param:      c  l  u :=
            [p1,i1] 600 . . #purchase new tires each race

```

```

[p2,i2] 600 . .
[p3,i3] 600 . .
[p4,i4] 600 . .
[i1,r1] . 320 . #minimum tires needed each race
[i2,r2] . 240 .
[i3,r3] . 400 .
[i4,r4] . 520 .
[r1,i2] 250 . . #quick fix
[r2,i3] 250 . .
[r3,i4] 250 . .
[r1,i3] 95 . . #slowfix
[r2,i4] 95 . .
;

```

Here is our output:

```

AMPL
ampl: model group1_HW3_p3.mod
CPLEX 20.1.0.0: optimal solution; objective 490000
6 dual simplex iterations (0 in phase I)
x [*,*]
:   i1   i2   i3   i4   p1   p2   p3   p4   r1   r2   r3   r4   v0   :=
i1   .   .   .   .   .   .   .   .   .   .   .   .   .
i2   .   .   .   .   .   .   .   .   .   .   .   .   .
i3   .   .   .   .   .   .   .   .   .   .   .   .   .
i4   .   .   .   .   .   .   .   .   .   .   .   .   .
p1   320   .   .   .   .   .   .   .   .   .   .   .
p2   .   200   .   .   .   .   .   .   .   .   .   .
p3   .   .   .   .   .   .   .   .   .   .   .   .
p4   .   .   .   .   .   .   .   .   .   .   .   .
r1   .   40   280   .   .   .   .   .   .   .   .   .
r2   .   .   120   120   .   .   .   .   .   .   .   .
r3   .   .   .   400   .   .   .   .   .   .   .   .
v0   .   .   .   .   320   200   0   0   .   .   .   .   .
v1   .   .   .   .   .   .   .   .   .   .   .   .   520

:   v1   :=
r1   0
r2   0
r3   0
r4   520
;

ampl: |

```

We look to be purchasing new tires for both the needs of the first two races, 320 and 200 respectively. This is the maximum number of tires needed. We use the normal service on 280 tires from the first race and quick on the other 40. In second race we use the normal service on 120 but quick fix 120. For the third race we quick fix all 400 tires used. We end up with exactly the number of tires needed in the fourth race. Total cost is \$490 000.

4. Dunder Mifflin

By far our most complicated model. Unhired workers will flow to the worker pool, wp and flow back out to the Workers with a weight of zero to maintain the flow. g and s represent the available generalist and specialists. If they are hired, they flow to the hired pool of each type. They are then transported to each plant at the respective cost. We then convert each worker into their number of products created with the weight. Finally we combine the number of possible products they can create into each location. Inventory is created with the regular cost but with limits on the maximums. Overtime is represented in red and uses a different virtual node for each location. Inventory is then shipped from each of the factories to the different businesses we serve, meeting the demand as represented inside the node. We attempted to make the flow circular but do to the difference in weights based on employment categories, could never get it to create a balanced flow.

This model provides labor for overtime but does not account for a premium the wage for the employee working outside of regular hours. We see this as an assumption that there is not a bump in pay for working second shift but there is a 50% increase in our overhead costs of production.

Here is our model file:

```
# AMPL model for the Minimum Cost Network Flow Problem
#
# By default, this model assumes that b[i] = 0, c[i,j] = 0,
# l[i,j] = 0 and u[i,j] = Infinity.
#
# Parameters not specified in the data file will get their default values.
reset;

options solver cplex;

set NODES;                # nodes in the network
set ARCS within {NODES, NODES}; # arcs in the network

param b {NODES} default 0;    # supply/demand for node i
param c {ARCS} default 0;    # cost of one of flow on arc(i,j)
param l {ARCS} default 0;    # lower bound on flow on arc(i,j)
param u {ARCS} default Infinity; # upper bound on flow on arc(i,j)
param mu {ARCS} default 1;    # multiplier on arc(i,j) -- if one unit leaves i, mu[i,j] units arrive

var x {ARCS};                # flow on arc (i,j)

minimize cost: sum{(i,j) in ARCS} c[i,j] * x[i,j]; #objective: minimize arc flow cost

# Flow Out(i) - Flow In(i) = b(i)

subject to flow_balance {i in NODES}:
sum{j in NODES: (i,j) in ARCS} x[i,j] - sum{j in NODES: (j,i) in ARCS} mu[j,i] * x[j,i] = b[i];

subject to capacity {(i,j) in ARCS}: l[i,j] <= x[i,j] <= u[i,j];

data group1_HW3_p4.dat;

solve;

display x;
```

Here is our data file:

```
#MCNFP Problem - data file for problem instance
#Charles Nicholson, ISE 5113, 2015

#use with MCNFP.txt model
#note: default arc costs and lower bounds are 0
#      default arc upper bounds are infinity
#      default node requirements are 0

set NODES :=          #v0, v1, #virtual nodes at begining and end to get the flow going
                    g, s, #general and specialist
                    vg, vs, #virtual to get the cost of general and specialist
                    pgsc, pgv, pgst, pssc, psu, psst, #shipping cost of each employee
                    pgvsc, pgvu, pgvst, psvsc, psvu, psvst, #convert each employee to items
                    sc, u, st, #workers (as items) now at the plants
                    scot, uot, stot # overtime possible
                    isc, iu, ist, #inventory at each plant
                    tg, tfb, #transport goods to location
                    wp; #unhired worker pool

set ARCS := (s,vs),(g,vg), #hire the workers
            (s,wp), (g,wp), #unhired workers
            (wp,s), (wp,g), #flow the unhired workers back to keep the balance
            (vg,pgsc),(vg,pgv),(vg,pgst),(vs,pssc),(vs,psu),(vs,psst), #move different workers to factories
            (pgsc,pgvsc),(pgv,pgvu),(pgst,pgvst),(pssc,psvsc),(psu,psvu),(psst,psvst), #convert the workers into items
            (pgvsc,sc),(psvsc,sc),(pgvu,u),(psvu,u),(pgvst,st),(psvst,st), #more production capacity to each factory
            (sc,isc),(u,iu),(st,ist), #create the products
            (sc,scot), (u,uot), (st,stot), #overtime hours making products
            (scot,isc), (uot,iu), (stot,ist), #overtime products created go to inventory for free
            (isc,tg), (isc,tfb), (iu, tg), (iu,tfb), (ist,tg), (ist,tfb), #move the product from inventory to customer
            ;

param: b:=
        g 200
```

```

s 100
tg -1000
tfb -600;

param:      c  l u mu:=
[s,vs]      2000      .      100      . #recruit workers
[g,vg]      1700      .      200      .
[vg,pgsc]   300      .      .      . #move workers to factories
[vg,pgu]    250      .      .      .
[vg,pgst]   275      .      .      .
[vs,psc]    300      .      .      .
[vs,psu]    250      .      .      .
[vs,psst]   275      .      .      .
[pgsc,pgvsc] .      .      .      . 10 #convert workers to items
[pgu,pgvu]  .      .      .      . 10
[pgst,pgvst] .      .      .      . 10
[psc,psvsc] .      .      .      . 12
[psu,psvu]  .      .      .      . 12
[psst,psvst] .      .      .      . 12
[sc,isc]    90      .      .      . 505 #create the products
[u,iu]      105      .      .      . 465
[st, ist]   115      .      570      .
[sc, scot]  135      .      100      . #overtime possible
[u,uot]     157.5    .      100      .
[st,stot]   172.5    .      100      .
[isc, tg]    8      .      .      . #move product to customer
[isc, tfb]   15      .      .      . #####check me
[iu, tg]     14      .      .      .
[iu, tfb]    18      .      .      .
[ist, tg]    24      .      .      .
[ist,tfb]    20      .      .      .
[wp,s]       .      .      .      . 0
[wp,g]       .      .      .      . 0
;

```

Here is my output:

```

Console
AMPL
ampl: model group1_Hw3_p4.mod
CPLEX 20.1.0.0: optimal solution; objective 497016.6667
10 dual simplex iterations (0 in phase I)
x [*,*] (tr)
:      g      isc      ist      iu      pgsc      pgst      pgv      pgvsc      pgvst      pgvu      pssc      psst :=
pgvsc      .      .      .      .      0      .      .      .      .      .      .      .
pgvst      .      .      .      .      .      0      .      .      .      .      .      .
pgvu      .      .      .      .      .      .      40      .      .      .      .      .
psvsc      .      .      .      .      .      .      .      .      .      .      47.0833      .
psvst      .      .      .      .      .      .      .      .      .      .      .      47.5
sc      .      .      .      .      .      .      .      0      .      .      .      .
st      .      .      .      .      .      .      .      .      0      .      .      .
tfb      .      .      0      570      30      .      .      .      .      .      .      .
tg      .      565      .      0      435      .      .      .      .      .      .      .
u      .      .      .      .      .      .      .      .      .      .      400      .
vg      40      .      .      .      .      .      .      .      .      .      .      .
wp      160      .      .      .      .      .      .      .      .      .      .      .

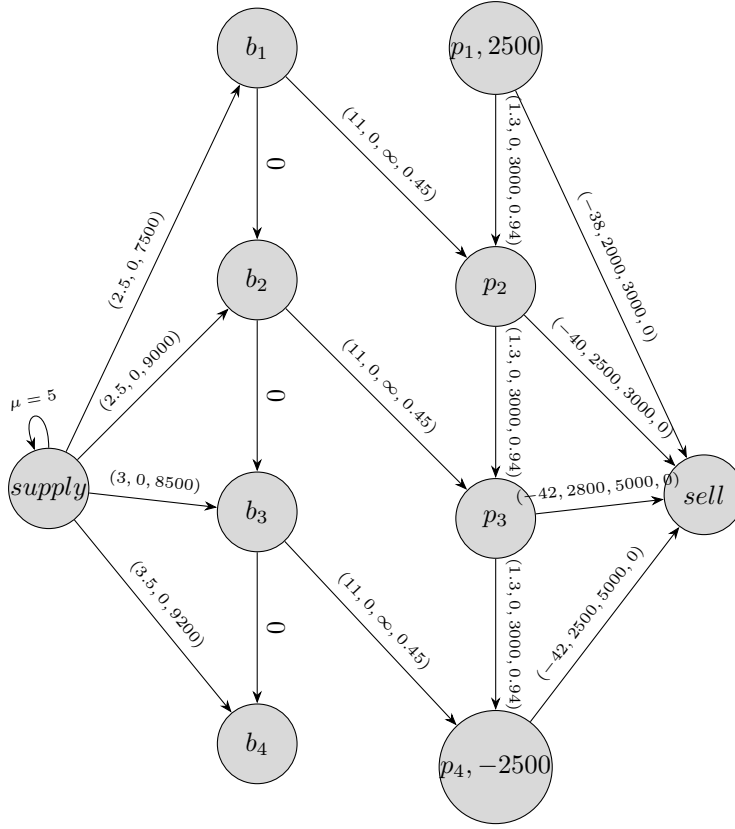
:      psu      psvsc      psvst      psvu      s      sc      scot      st      stot      u      uot      vg      :=
isc      .      .      .      .      .      505      60      .      .      .      .      .
ist      .      .      .      .      .      .      .      570      0      .      .      .
iu      .      .      .      .      .      .      .      .      .      465      0      .
pgsc      .      .      .      .      .      .      .      .      .      .      .      0
pgst      .      .      .      .      .      .      .      .      .      .      .      0
pgv      .      .      .      .      .      .      .      .      .      .      .      40
psvu      5.41667      .      .      .      .      .      .      .      .      .      .      .
sc      .      565      .      .      .      .      .      .      .      .      .      .
scot      .      .      .      .      .      60      .      .      .      .      .      .
st      .      .      570      .      .      .      .      .      .      .      .      .
stot      .      .      .      .      .      .      .      0      .      .      .      .
u      .      .      .      65      .      .      .      .      .      .      .      .
uot      .      .      .      .      .      .      .      .      .      0      .      .
vs      .      .      .      .      100      .      .      .      .      .      .      .
wp      .      .      .      .      0      .      .      .      .      .      .      .

:      vs      wp      :=
g      .      0
pssc      47.0833      .
psst      47.5      .
psu      5.41667      .
s      .      160
;

```

Solution is non-integer in employees which is unfortunate but can be the case with the generalized network flow problems. All 100 specialists are hired, 40 generalists. Scranton does use 60 products of OT, none of the other plants do but they max out production at each.

5. Mud b Gone



Here is our model file:

```
# AMPL model for the Minimum Cost Network Flow Problem
#
# By default, this model assumes that b[i] = 0, c[i,j] = 0,
# l[i,j] = 0 and u[i,j] = Infinity.
#
# Parameters not specified in the data file will get their default values.
reset;

options solver cplex;
option cplex_options 'sensitivity';

set NODES;                # nodes in the network
set ARCS within {NODES, NODES}; # arcs in the network

param b {NODES} default 0;    # supply/demand for node i
param c {ARCS} default 0;    # cost of one of flow on arc(i,j)
param l {ARCS} default 0;    # lower bound on flow on arc(i,j)
param u {ARCS} default Infinity; # upper bound on flow on arc(i,j)
param mu {ARCS} default 1;    # multiplier on arc(i,j) -- if one unit leaves i, mu[i,j] units arrive

var x {ARCS};                # flow on arc (i,j)

minimize cost: sum{(i,j) in ARCS} c[i,j] * x[i,j]; #objective: minimize arc flow cost

# Flow Out(i) - Flow In(i) = b(i)

subject to flow_balance {i in NODES}:
sum{j in NODES: (i,j) in ARCS} x[i,j] - sum{j in NODES: (j,i) in ARCS} mu[j,i] * x[j,i] = b[i];
```

```

subject to upcapacity {(i,j) in ARCS}: x[i,j] <= u[i,j];

subject to lowcapacity {(i,j) in ARCS}: l[i,j] <= x[i,j];

data group1_HW3_p5.dat;

solve;

display x;

display upcapacity,upcapacity.up, upcapacity.down;

display x.current, x.up, x.down;

```

Here is our data file:

```

#MCNFP Problem - data file for problem instance
#Charles Nicholson, ISE 5113, 2015

#use with MCNFP.txt model
#note: default arc costs and lower bounds are 0
#      default arc upper bounds are infinity
#      default node requirements are 0

set NODES := supply, #suppliers
            b1,b2,b3,b4, #base for production
            p1, p2, p3, p4, #product
            sold #sold product
;

set ARCS := (supply,*) b1 b2 b3 b4 #base purchased from supplier
            (*,sold) p1 p2 p3 p4 #product sold
            (b1,p2),(b2,p3),(b3,p4), #base converted to product
            (b1,b2),(b2,b3),(b3,b4),
            (p1,p2),(p2,p3),(p3,p4)
            (supply,supply)
;

param: b:=
        p1 2500
        p4 -2500;

param: c l u mu:=
        [supply,b1] 2.5 . 7500 . #buy new base
        [supply,b2] 2.5 . 9000 .
        [supply, b3] 3 . 8500 .
        [supply,b4] 3.5 . 9200 .

        [p1,sold] -38 2000 3000 0 #sell product
        [p2,sold] -40 2500 3000 0
        [p3,sold] -42 2800 5000 0
        [p4,sold] -42 2500 5000 0

        [p1,p2] 1.3 . 3000 .94 #store product till next month
        [p2,p3] 1.3 . 3000 .94
        [p3,p4] 1.3 . 3000 .94

        [b1,p2] 11 . . .45 #convert base into product. Assumption not too worry about max storage
        [b2,p3] 11 . . .45
        [b3,p4] 11 . . .45

        [supply,supply] . . . 5
;

```

- (a) We see a solution for our flow. We sell 2000, 2500, 4220 and 2500 in each of the respective periods. We remark on the infinite supply house obtained by creating a loop with a μ factor set to 5 but could be any value greater than 1.

```

Console
AMPL
ampl: model group1_HW3_p5.mod
CPLEX 20.1.0.0: sensitivity
CPLEX 20.1.0.0: optimal solution; objective -115840
4 dual simplex iterations (1 in phase I)

suffix up OUT;
suffix down OUT;
suffix current OUT;
x :=
b1      b2      2988.89
b1      p2      4511.11
b2      b3      2611.11
b2      p3      9377.78
b3      b4      0
b3      p4      11111.1
p1      p2      500
p1      sold    2000
p2      p3      0
p2      sold    2500
p3      p4      0
p3      sold    4220
p4      sold    2500
supply b1      7500
supply b2      9000
supply b3      8500
supply b4      0
supply supply  6250
;

```

- (b) We next look at sensitivity of the capacity. We see a value of -\$5.40 in the report, supply to b2. We interpret this as having additional gallons of base will increase (recall minimizing) our income by \$5.40. Both the up and the down are reported as 0. We are unsure of how to interpret this as we clearly would use more of the base if it was available but are unsure why these values are all zero. We even broke apart the upper and lower limit thinking this was the issue but did not change the up and down on the shadow price. We do clearly see this as the bottle neck in our model. We also note that when we moved the cap of 9000 to 9001 we earned another \$5.40. We wonder if since the number is negative, we are getting 0 as an upper limit.

```

:          upcapacity upcapacity.up upcapacity.down :=
b1    b2      0          0          0
b1    p2      0          0          0
b2    b3      0          0          0
b2    p3      0          0          0
b3    b4      0          0          0
b3    p4      0          0          0
p1    p2      0          0          0
p1    sold    0          0          0
p2    p3      0          0          0
p2    sold    0          0          0
p3    p4      0          0          0
p3    sold    0          0          0
p4    sold    0          0          0
supply b1    -5.4        0          0
supply b2    -5.4        0          0
supply b3    -4.9        0          0
supply b4      0         0          0
supply supply 0         0          0
;

```

- (c) We see that the contribution to the cost is -\$40 on the flow from p2 to Sold. This value can be modified up to -\$42 and not change the model at all. This means we could raise the price and earn more for our company keeping everything else the same.

```

:          x.current      x.up      x.down      :=
b1    b2      0          0.0861702  -1e+20
b1    p2      11         1e+20      10.9138
b2    b3      0          1.82872    -3.19744e-15
b2    p3      11         11         9.17128
b3    b4      0          0          0
b3    p4      11         12.8287    11
p1    p2      1.3        1.48      -1e+20
p1    sold    -38        1e+20      -38.18
p2    p3      1.3        1e+20      -2.52
p2    sold    -40        1e+20      -42
p3    p4      1.3        1e+20      -2.52
p3    sold    -42        -42        -1e+20
p4    sold    -42        1e+20      -42
supply b1      2.5        7.9      -1e+20
supply b2      2.5        7.9      -1e+20
supply b3      3          7.9      -1e+20
supply b4      0          0          0
supply supply 0          19.6     -1e+20
;

```