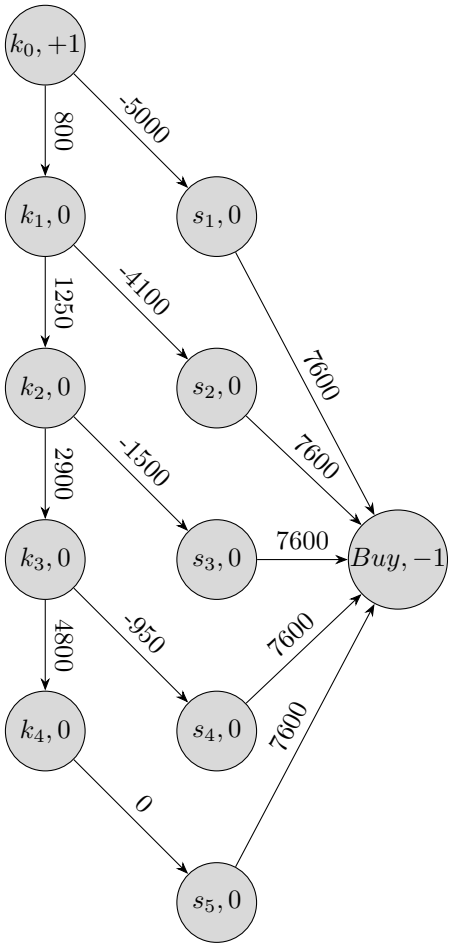
# Homework 3 Advanced Analytics and Metaheuristics

Group 1: Nicholas Jacob, Garrison Kleman, Hannah Jensen

February 20, 2024

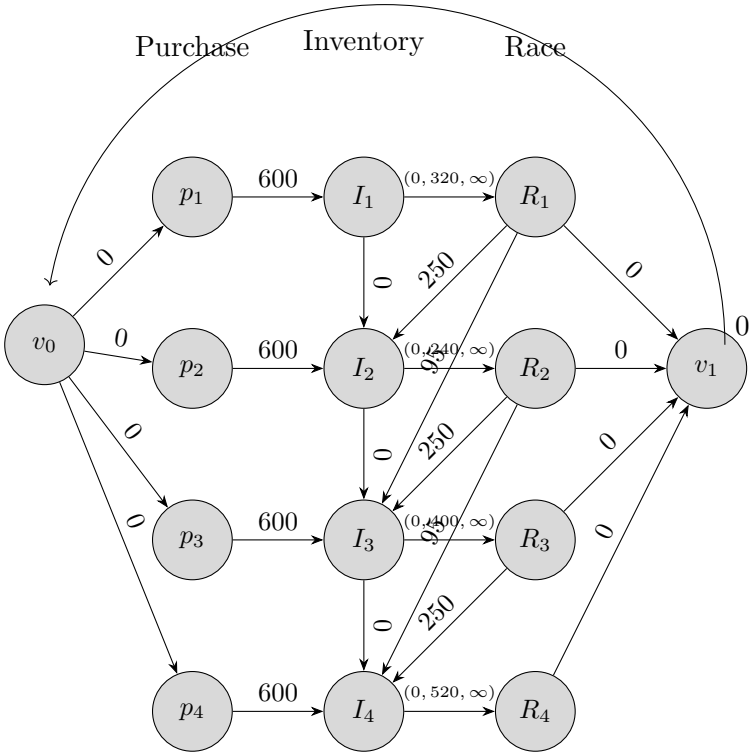
1. Team Building
   (a)
   (b)
2. Outdoor Grilling

Maintain Sell




3. Race Car Tires

   Here is my flow model:




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)
```

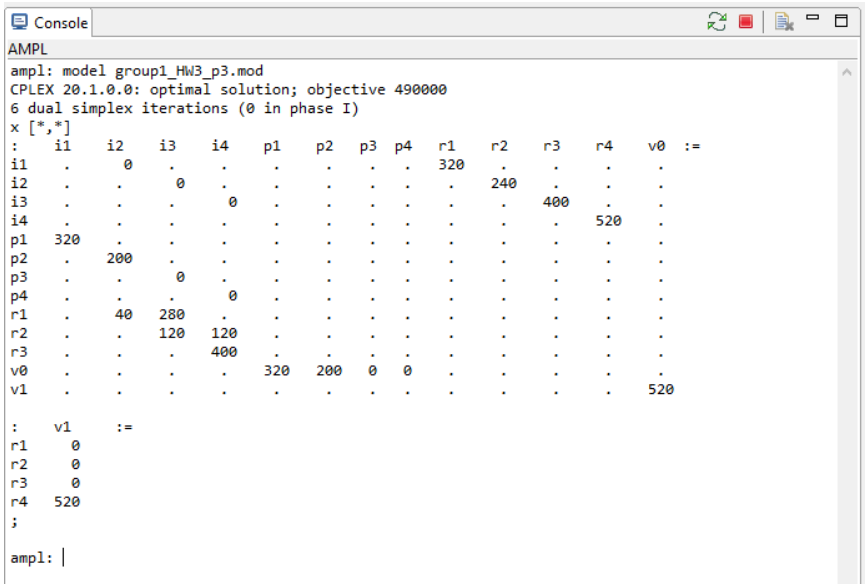
```
minimize cost: sum{(i,j) in ARCS} c[i,j] * x[i,j];  #objective: minimize arc flow cost

# 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  . .
```

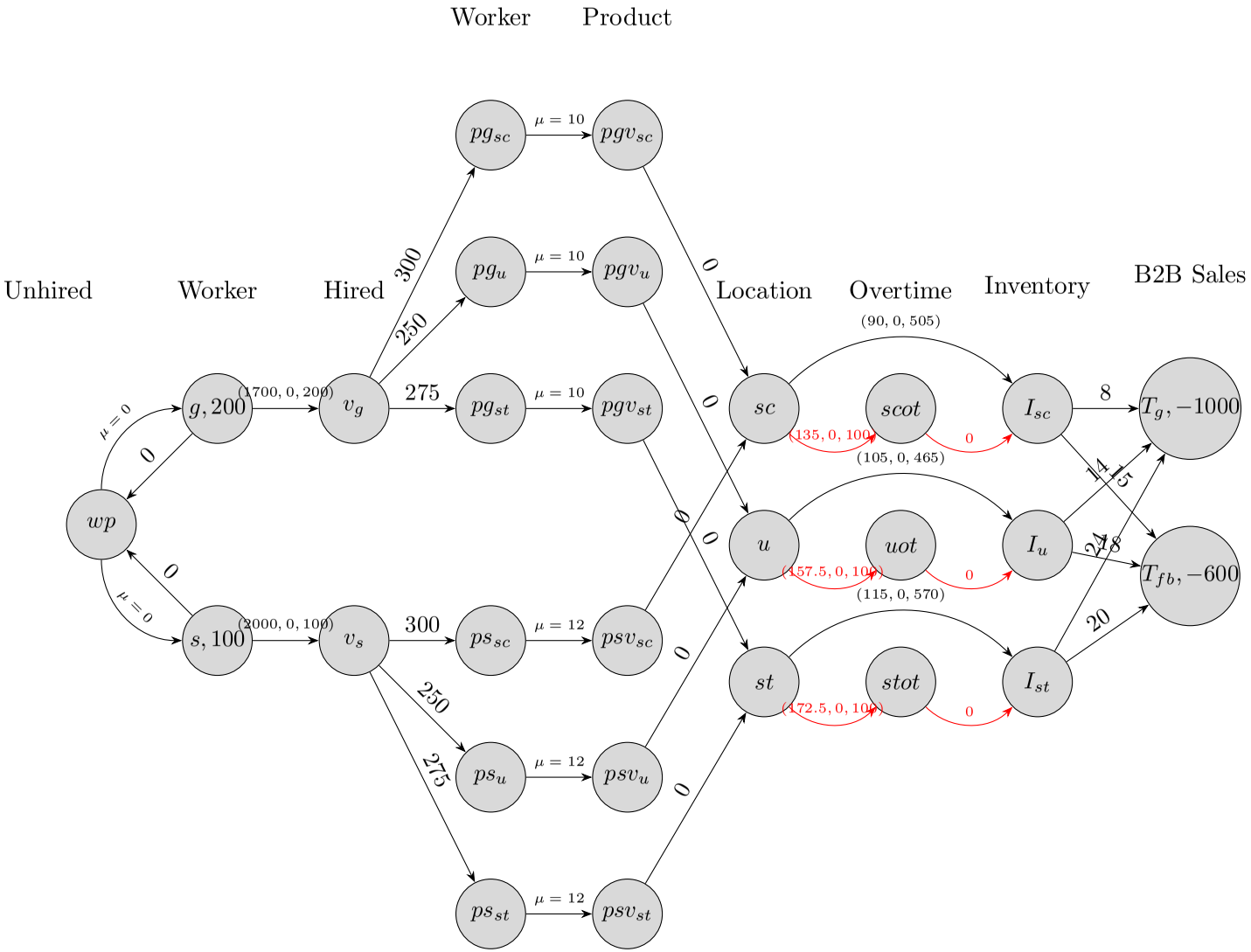
```
[p3,i3] 600   . .
[p4,i4] 600   . .
[i1,r1] .         320 . #minimum tires needed each race
[i2,r2] .         240 .
[i3,r3] .         400 .
[i4,r4] .         520 .
[r1,i2] 250 .          . #quick fix
[r2,i3] 250 .         .
[r3,i4] 250 .         .
[r1,i3] 95  .         .#slowfix
[r2,i4] 95      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

Here is our output:




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


Worker
Product
Unhired
Worker
Hired
Location
Overtime
Inventory
B2B Sales
$wp$
$g, 200$
$s, 100$
$v_g$
$v_s$
$pg_{sc}$
$pg_u$
$pg_{st}$
$ps_{sc}$
$ps_u$
$ps_{st}$
$pgv_{sc}$
$pgv_u$
$pgv_{st}$
$psv_{sc}$
$psv_u$
$psv_{st}$
$sc$
$u$
$st$
$scot$
$uot$
$stot$
$I_{sc}$
$I_u$
$I_{st}$
$T_g, -1000$
$T_{fb}, -600$
$\mu = 0$
$\mu = 10$
$\mu = 12$
(1700, 0, 200)
(2000, 0, 100)
300
250
275
0
(90, 0, 505)
(135, 0, 100)
(105, 0, 465)
(157.5, 0, 100)
(115, 0, 570)
(172.5, 0, 100)
8
14
17
24
18
20

Here is our model file:

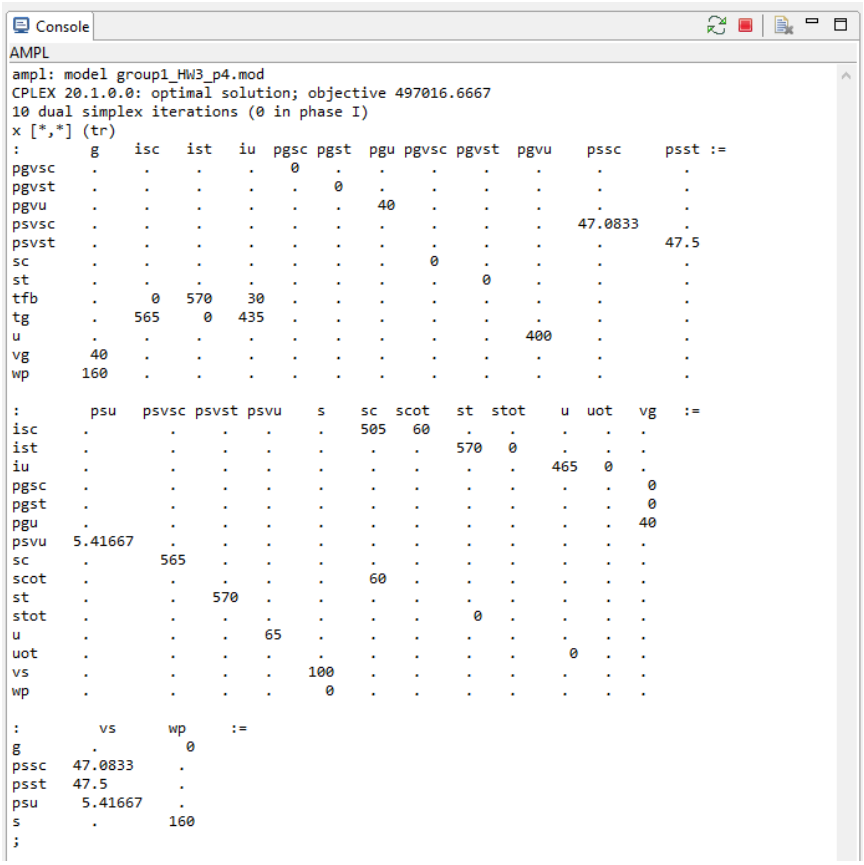
```
# AMPL model for the Minimum Cost Network Flow Problem
#
# By default, this model assumes that b[i] = 0, c[i,j] = 0,
# l[i,j] = 0 and u[i,j] = Infinity.
#
# Parameters not specified in the data file will get their default values.
reset;

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:

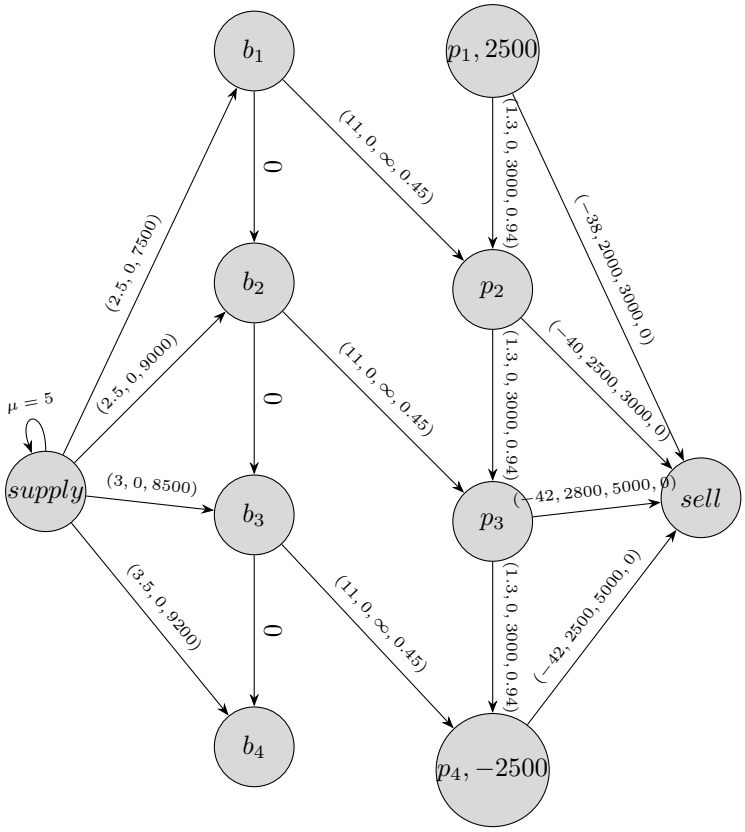
```
#MCNFP Problem - data file for problem instance
#Charles Nicholson, ISE 5113, 2015

#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, pgu, 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,pgu),(vg,pgst),(vs,pssc),(vs,psu),(vs,psst), #move different workers to factories
                        (pgsc,pgvsc),(pgu,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,pssc]                    300          .         .           .
                [vs,psu]                     250          .         .           .
                [vs,psst]                    275          .         .           .
                [pgsc,pgvsc]         .       .          .         .           10 #convert workers to items
                [pgu,pgvu]                   .          .         .           .          10
                [pgst,pgvst]         .       .          .         .           10
                [pssc,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  pgu 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
pgu  .  .  .  .  .  .  .  .  .  .  .  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
;

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;

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_p5.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 :=  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
;
```