Homework 2 - Advanced LP & Network Flow Models

Adv. Analytics and Metaheuristics

Daniel Carpenter and Christopher Ferguson

February 2022

Contents

1 - Problem 1 **2**

1.1 Problems `a` and `b` 2

1.2 Model Assumptions and Overview 2

1.3 Mathematical Formulation 3

1.4 Code and Output 5

1.5 Problems 1 `c (ii-vii)` 6

1.6 Problems 1 `d` 7

2 - Problem 2 **9**

2.1 Model Overview 9

2.2 Mathematical Formulation 9

2.3 Code and Output 10

3 - Problem 3 **11**

3.1 Model Overview 11

3.2 Mathematical Formulation 12

3.3 Code and Output 13

4 - Problem 4 **15**

4.1 Model Overview 15

4.2 Mathematical Formulation 17

4.3 Code and Output 18

1 - Problem 1

1.1 Problems `a` and `b`

- Below shows how we get to the answer of Problems `a` and `b`

Problem 1 (a): Calculating the Upper Bound for the Raisin Product

Total Raisins Calculated	1,240,000	
Less Raisin Dedicated Portion to Grade A	(930,000)	75% <- note this is the % towards A
Grade B Portion Implied*	310,000	25% <- "" B
Total Grade B Available	5,270,000	
% Grade B Used Towards Raisins, i.e. tier 8	5.88%	

Interpretation and Assumptions:

*There must be 5.88% (Total of 310,000) of grade B's total supply that are on tier 8, which raisins are able to use since it satisfies the minimum points.
For raisins, this implies that 25% of the raisin production will stem from Grade B (and 75% towards A)
For juice, this implies that 75% of the juice production will stem from Grade B (and 25% towards A)
Note 100% Grade B used for Jelly.

Problem 1(b): Calculate the Fruit Cost seen in Table 3

Grade	Cost per Pound	
Grade A	$ 0.45	<- noted in table 3 bullet 4
Grade B	$ 0.25	

	Raisins	Juice	Jelly	
Pounds per Product	6.5	14.0	18.0	<- Noted in Table 2 footer

% Breakdown of Grades				
Grade A	75.0%	25.0%	0.0%	<- this is from the aboves assumptions text
Grade B	25.0%	75.0%	100.0%	<- 1 minus above
Weighted Avg. Cost per Pound	**$ 0.40**	**$ 0.30**	**$ 0.25**	
Fruit Cost (Pounds * Wt. Avg. Cost)	$ 2.60	$ 4.20	$ 4.50	***<- this is how they determine the fruit cost in Table 3***

1.2 Model Assumptions and Overview

- This model calculates the maximum `net profit` for a set of products and inputs grades (See below sets).

1.3 Mathematical Formulation

1.3.1 Sets

Set Name	Description
$FRUIT$	Fruit grade levels of grapes, which are `Grade A` and `Grade B` grapes
$PRODUCTS$	Types of products to be sold, which are `Raisins`, `Juice`, and `Jelly`

1.3.2 Parameters

Parameter Name	Description
$amountOfFruit_f$	*Pounds* of fruit grade ($f \in FRUIT$) available to use
$avgGradeOfFruit_f$	Avg. point quality of fruit grade ($f \in FRUIT$)
$productLimit_p$	Upper bound of number of products ($p \in PRODUCTS$) to produce
$poundsPerProduct_p$	The amount of pounds associated with a single unit of product ($p \in PRODUCTS$)
$contrToProfit_p$	The contribution to profit of product ($p \in PRODUCTS$)
$netProfit_p$	The net profit (*net of OH allocation*) of product ($p \in PRODUCTS$)
$productGradeLimit_p$	Requirement of mean point quality of a product ($p \in PRODUCTS$)

1.3.3 Decision Variables

Variable Name	Description
$produce_{f,p}$	Number (*as integer*) of products ($p \in PRODUCTS$) to be produced by fruit grade ($f \in FRUIT$)

1.3.4 Objective Function

$$Maximize\ Net\_Profit : \sum_{f \in FRUIT, p \in PRODUCTS} produce_{f,p} \times netProfit_p$$

1.3.5 Constraints

C1: For each fruit grade ($f \in FRUIT$), the number of products produced ($p \in PRODUCTS$) must be equal to numbers of fruit provided

$$maxWeight: \sum_{p \in PRODUCTS} (produce_{f,p} \times numbersPerProduct_p) = amountOfFruit_f, \; \forall \, f \in FRUIT$$

C2: For each product produced ($p \in PRODUCTS$), limit the number of products p produced to $\leq$ to the demanded (product numbers)

$$demand: \sum_{f \in FRUIT} produce_{f,p} \leq productLimit_p, \; \forall \, p \in PRODUCTS$$

C3: For each product ($p \in PRODUCTS$), the grade of fruit ($f \in FRUIT$) is greater than the minimum required grade of the product ($p \in PRODUCTS$).
$minAvgGrade:$

$$\sum_{f \in FRUIT} (produce_{f,p} \times avgGradeOfFruit_f) \geq productGradeLimit_p \times \sum_{f \in FRUIT} (produce_{f,p}),$$
$$\forall \, p \in PRODUCTS$$

C4: Non-negativity constraints

$$produce_{\, f,p} \geq 0, \; \forall \, f \in FRUIT, \; \forall \, p \in PRODUCTS$$

1.4 Code and Output

1.4.1 Code

group12_HW2_p1.mod

C: > Users > daniel.carpenter > OneDrive - the Chickasaw Nation > Documents > GitHub > OU-DSA > Metaheuristics > 03 - Homework > HW 02 > AMPL Models

```
reset;                    # Reset globals
options solver cplex;     # Using cplex for simplex alg

# SETS ==============================================================
    set FRUIT circular; # Circular to index the kth element for constraint 2
    set PRODUCTS circular; # ""

# PARAMETERS ========================================================
    param amountOfFruit {FRUIT} >= 0;   # Pounds of fruit available by grade
    param avgGradeOfFruit {FRUIT};      # Avg fruit grade
    param productLimit {PRODUCTS} >= 0; # Demand for products
    param poundsPerProduct {PRODUCTS};  # Pounds associated with a product
    param contrToProfit {PRODUCTS};     # Product contribution to profit
    param netProfit {PRODUCTS};         # Net profit per product
    param productGradeLimit {PRODUCTS}; # The min grade for each product


# DECISION VARIABLES ================================================
    ## Includes upper and lower bounds
    var produce {f in FRUIT, p in PRODUCTS} >= 0 integer;
    # var grade {p in PRODUCTS} >= 0, <= 10;

# OBJECTIVE FUNCTION ================================================
    maximize Total_Profit:
        sum {f in FRUIT, p in PRODUCTS} produce[f, p] * netProfit[p];

# CONSTRAINTS =======================================================

    ## C1: Number of pounds of the products produced (p ∈ PRODUCTS) must be <=  pounds of fruit provided (f ∈ FRUIT)
    subject to maxWeight {f in FRUIT}:
        (sum {p in PRODUCTS} produce[f,p] * poundsPerProduct[p]) == amountOfFruit[f];

    ## C2:  Limit the number of produced to <= the demand
    subject to demand {p in PRODUCTS}:
        (sum {f in FRUIT} produce[f,p]) <= productLimit[p];

    ## C3: For all products (p ∈ PRODUCTS), the weighted-average grade of fruit is greater than the minimum required grade.
    subject to minAvgGrade {p in PRODUCTS}:
        (sum {f in FRUIT} produce[f,p] * avgGradeOfFruit[f])
        >=  productGradeLimit[p] * (sum {f in FRUIT} produce[f,p]);

# CONTROLS ==========================================================
    data group12_HW2_p1.dat;
    solve;

    print;
    print "At maximum profit, the number of products to produce (by fruit grade):";
    display produce;
```

group12_HW2_p1.dat

C: > Users > daniel.carpenter > OneDrive - the Chickasaw Nation > Documents > GitHub > OU-DSA > Metaheuristics > 03 - Homew

```
# February 2022
# Problem 1

# SETS ==============================================================
    # Fruit grade levels of grapes, which are `Grade A` and `Grade B` grapes
    set FRUIT := gradeA gradeB;
    # Types of products to be sold, which are `Raisins`, `Juice`, and `Jelly`
    set PRODUCTS := raisins juice jelly;

# PARAMETERS ========================================================
    # Tons of fruit grade (f ∈ FRUIT) available to use
    param amountOfFruit :=
    gradeA           930000
    gradeB          5270000;

    # Avg. point quality of fruit grade (f ∈ FRUIT)
    param avgGradeOfFruit :=
    gradeA          9
    gradeB          5;

    # Upper bound of production *tons* of product (p ∈ PRODUCTS)
    param productLimit :=
    raisins         INFINITY
    juice           190000
    jelly           210000;

    # The amount of pounds associated with a single unit of product (p ∈ PRODUCTS)
    param poundsPerProduct :=
    raisins         6.5
    juice           14
    jelly           18;

    # The contribution to profit of a product (p ∈ PRODUCTS)
    param contrToProfit :=
    raisins         1.40
    juice           2.46
    jelly           2.35;

    # The net profit (net of OH allocation) of a product (p ∈ PRODUCTS) per *ton*
    param netProfit :=
    raisins          0.35
    juice           -0.14
    jelly            0.43;

    # Requirement of mean point quality of a product (p ∈ PRODUCTS)
    param productGradeLimit :=
    raisins         8
    juice           6
    jelly           0
```

1.4.2 Output

```
CPLEX 20.1.0.0: optimal integer solution within mipgap or absmipgap; objective 107593.36
14 MIP simplex iterations
0 branch-and-bound nodes
absmipgap = 2.60826, relmipgap = 2.42418e-05

At maximum profit, the number of products to produce (by fruit grade):
produce :=
gradeA raisins     74996
gradeA juice       31609
gradeA jelly           0
gradeB raisins     24988
gradeB juice       94827
gradeB jelly      210000
;
```

Optimal profit is `$107,600`. `74,996 Raisins` produced using `Grade A`, `24,988` of `Grade B`, and so on for each product. See validation of the weighted-average point constraint below.

	Grade A	Grade B
Avg Grade of Fruit	9	5

Product	Grade A	Grade B	Pounds per Product	Equation using the produce var: LHS		RHS	inequality Holds?	Product Grade Limit	Calculated Weighted Avg Points by Product: Weight A	Weight B	Weighted Avg Limit	Calculated Net Profit: Net Profit	Profit
Raisin	74,996	24,988	7	799,904	>=	799,872	TRUE	8	75%	25%	8.0003	0.35	34,994.40
Juice	31,609	94,827	14	758,616	>=	758,616	TRUE	6	25%	75%	6.0000	(0.14)	(17,701.04)
Jelly	-	210,000	18	1,050,000	>=	-	TRUE	0	0%	100%	5.0000	0.43	90,300.00
Total per Grade	**930,000**	**5,270,000**											**$ 107,593.36**

Total Lbs	6,200,000

*Equation: SUM_f (produce $_{f,p}$ * avgGradeOfFruit $_f$) >= productGradeLimit $_p$ * SUM_f (produce $_{f,p}$), for all p in PRODUCTS*

1.5 Problems 1 `c (ii-vii)`

i.

ii.

iii.

iv.

v.

vi.

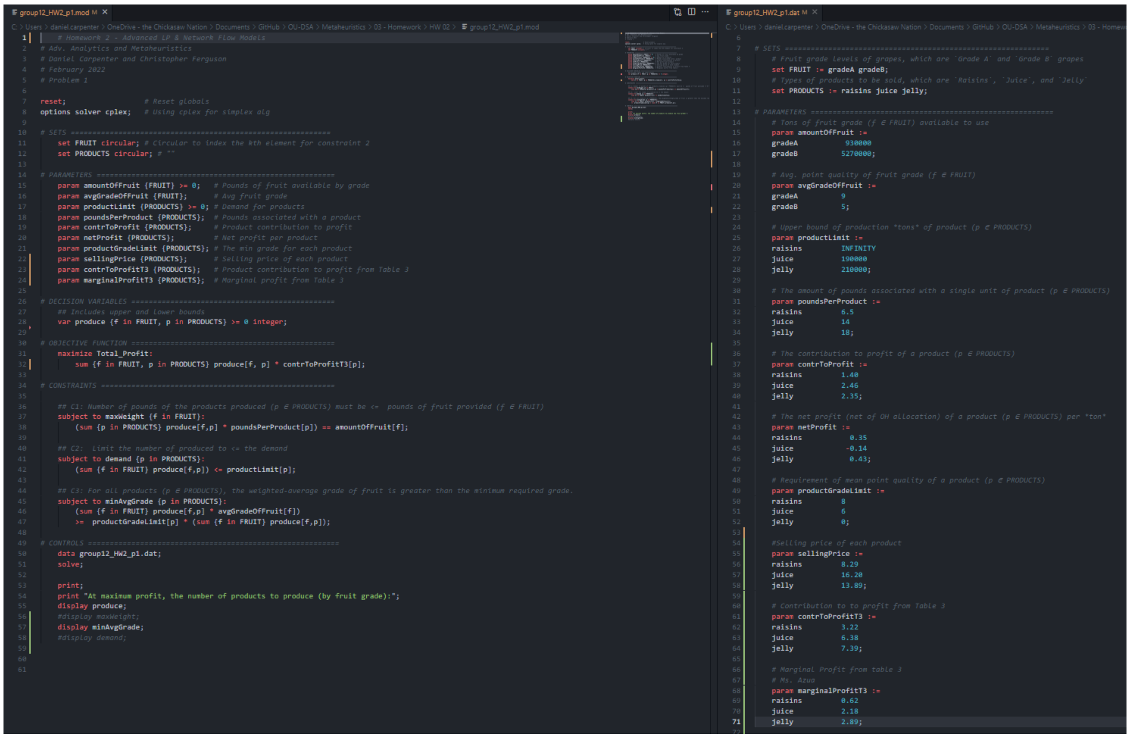
vii.

1.6 Problems 1 d

i.

ii.

iii.



```
group12_HW2_p1.mod
C: > Users > daniel.carpenter > OneDrive - the Chickasaw Nation > Documents > GitHub > OU-DSA > Metaheuristics > 03 - Homework > HW 02 > group12_HW2_p1.mod
    # Homework 2 - Advanced LP & Network Flow Models
# Adv. Analytics and Metaheuristics
# Daniel Carpenter and Christopher Ferguson
# February 2022
# Problem 1

reset;                  # Reset globals
options solver cplex;   # Using cplex for simplex alg

# SETS ============================================================
    set FRUIT circular; # Circular to index the kth element for constraint 2
    set PRODUCTS circular; # ""

# PARAMETERS =======================================================
    param amountOfFruit {FRUIT} >= 0;   # Pounds of fruit available by grade
    param avgGradeOfFruit {FRUIT};      # Avg fruit grade
    param productLimit {PRODUCTS} >= 0; # Demand for products
    param poundsPerProduct {PRODUCTS};  # Pounds associated with a product
    param contrToProfit {PRODUCTS};     # Product contribution to profit
    param netProfit {PRODUCTS};         # Net profit per product
    param productGradeLimit {PRODUCTS}; # The min grade for each product
    param sellingPrice {PRODUCTS};      # Selling price of each product
    param contrToProfitT3 {PRODUCTS};   # Product contribution to profit from Table 3
    param marginalProfitT3 {PRODUCTS};  # Marginal profit from Table 3

# DECISION VARIABLES ===============================================
    ## Includes upper and lower bounds
    var produce {f in FRUIT, p in PRODUCTS} >= 0 integer;

# OBJECTIVE FUNCTION ===============================================
    maximize Total_Profit:
        sum {f in FRUIT, p in PRODUCTS} produce[f, p] * contrToProfitT3[p];

# CONSTRAINTS ======================================================

    ## C1: Number of pounds of the products produced (p ∈ PRODUCTS) must be <=  pounds of fruit provided (f ∈ FRUIT)
    subject to maxWeight {f in FRUIT}:
        (sum {p in PRODUCTS} produce[f,p] * poundsPerProduct[p]) == amountOfFruit[f];

    ## C2:  Limit the number of produced to <= the demand
    subject to demand {p in PRODUCTS}:
        (sum {f in FRUIT} produce[f,p]) <= productLimit[p];

    ## C3: For all products (p ∈ PRODUCTS), the weighted-average grade of fruit is greater than the minimum required grade.
    subject to minAvgGrade {p in PRODUCTS}:
        (sum {f in FRUIT} produce[f,p] * avgGradeOfFruit[f])
        >=  productGradeLimit[p] * (sum {f in FRUIT} produce[f,p]);

# CONTROLS =========================================================
    data group12_HW2_p1.dat;
    solve;

    print;
    print "At maximum profit, the number of products to produce (by fruit grade):";
    display produce;
    #display maxWeight;
    display minAvgGrade;
    #display demand;
```

```
group12_HW2_p1.dat
C: > Users > daniel.carpenter > OneDrive - the Chickasaw Nation > Documents > GitHub > OU-DSA > Metaheuristics > 03 - Homew

# SETS ============================================================
    # Fruit grade levels of grapes, which are `Grade A` and `Grade B` grapes
    set FRUIT := gradeA gradeB;
    # Types of products to be sold, which are `Raisins`, `Juice`, and `Jelly`
    set PRODUCTS := raisins juice jelly;

# PARAMETERS =======================================================
    # Tons of fruit grade (f ∈ FRUIT) available to use
    param amountOfFruit :=
    gradeA          930000
    gradeB         5270000;

    # Avg. point quality of fruit grade (f ∈ FRUIT)
    param avgGradeOfFruit :=
    gradeA         9
    gradeB         5;

    # Upper bound of production *tons* of product (p ∈ PRODUCTS)
    param productLimit :=
    raisins        INFINITY
    juice          190000
    jelly          210000;

    # The amount of pounds associated with a single unit of product (p ∈ PRODUCTS)
    param poundsPerProduct :=
    raisins        6.5
    juice          14
    jelly          18;

    # The contribution to profit of a product (p ∈ PRODUCTS)
    param contrToProfit :=
    raisins        1.40
    juice          2.46
    jelly          2.35;

    # The net profit (net of OH allocation) of a product (p ∈ PRODUCTS) per *ton*
    param netProfit :=
    raisins          0.35
    juice           -0.14
    jelly            0.43;

    # Requirement of mean point quality of a product (p ∈ PRODUCTS)
    param productGradeLimit :=
    raisins        8
    juice          6
    jelly          0;

    #Selling price of each product
    param sellingPrice :=
    raisins        8.29
    juice          16.20
    jelly          13.89;

    # Contribution to to profit from Table 3
    param contrToProfitT3 :=
    raisins        3.22
    juice          6.38
    jelly          7.39;

    # Marginal Profit from table 3
    # Ms. Azua
    param marginalProfitT3 :=
    raisins        0.62
    juice          2.18
    jelly          2.89;
```

1.6.1 Output

```
ampl: model 'C:\Users\daniel.carpenter\OneDrive - the Chickasaw Nation\Documents\GitHub\OU-
CPLEX 20.1.0.0: optimal integer solution within mipgap or absmipgap; objective 2695531.61
6 MIP simplex iterations
0 branch-and-bound nodes
absmipgap = 4.89857, relmipgap = 1.81729e-06

At maximum profit, the number of products to produce (by fruit grade):
produce :=
gradeA raisins    40780
gradeA juice      47495
gradeA jelly          0
gradeB raisins    13592
gradeB juice     142481
gradeB jelly     177051
;

minAvgGrade [*] :=
raisins  0
  juice  0
  jelly  0
;

ampl:
```

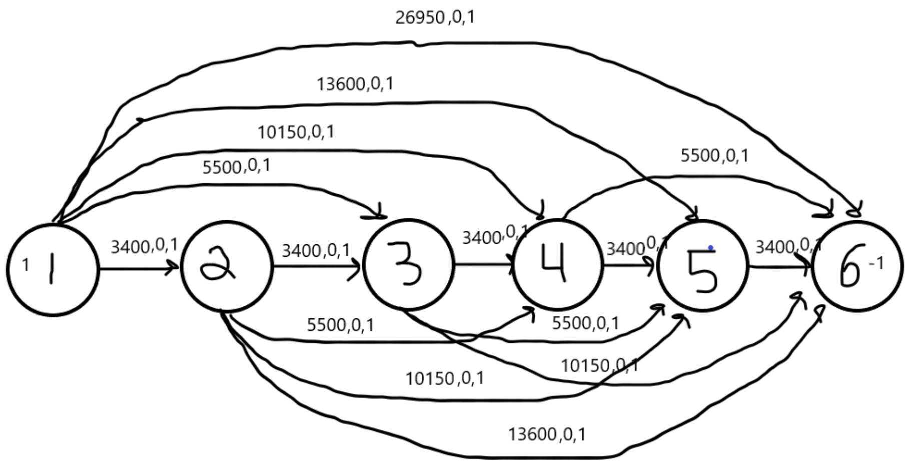
- The maximum to pay per pound would be equal to the dual variable of the active constraint, which does not show any values.

1.6.2 Code

- The model who should be used is ________

2 - Problem 2

2.1 Model Overview



2.2 Mathematical Formulation

$NODES$: Set of all nodes in above network flow diagram resembling a time period.

The set A is a set of *arcs,* e.g. (i, j) for $i \in N, j \in N$
each of which may carry *flow of a commodity*

Decision variable: x_{ij} determines the units of flow on arc (i, j)

Arc (i, j)

- cost c_{ij} per unit of flow on arc (i, j)
- upper bound on flow of u_{ij} (capacity)
- lower bound on flow of ℓ_{ij} (usually 0)

2.2.1 Objective, and Constraints

$$
\begin{aligned}
\text{minimize} \quad & \sum_{(i,j)\in A} c_{ij}x_{ij} \\
\text{subject to} \quad & \sum_{j:(i,j)\in A} x_{ij} - \sum_{j:(j,i)\in A} x_{ji} = b_i \quad \forall i \in N \\
& l_{ij} \leq x_{ij} \leq u_{ij} \qquad\qquad \forall (i,j) \in A
\end{aligned}
$$

2.3 Code and Output

2.3.1 Model `group12_HW2_p2.mod`

- Used `mcnfp.txt` from course website and renamed to `group12_HW2_p2.mod`.
- Added `data group12_HW2_p2.dat; solve;` and `display x;`

2.3.2 Data `group12_HW2_p2.dat`

```
#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 :=    1 2 3 4 5 6; #nodes for problem #2

set ARCS :=     (1,2) (1,3) (1,4) (1,5) (1,6) (2,3) (2,4) (2,5) (2,6) (3,4) (3,5) (3,6) (4,5) (4,6) (5,6); #arcs for problem #2

param b:=   1  1     #for shortest path problem, start node supply = 1
            6 -1;    #and the destination node supply = -1

#note: to make things a little more compact, you can use a "template"
# for setting up costs.  See Chapter 9 "Specificying Data" in the AMPL textbook

#for example, the first line below: [1, *] 2 5   3 10
#is short hand for: [1, 2] 5   [1,3]  10

#this comes in handy for some larger, more complex data files.

param c:=   [1, *] 2 3400    3 5550      4 10150     5 13600     6 26950         #arc cost equals the distance
                [2, *] 3 3400       4 5550      5 10150     6 13600
                [3, *] 4 3400       5 5550      6 10150
                [4, *] 5 3400       6 5550
                [5, *] 6 3400;
```

2.3.3 Output

- The lowest total cost over the `5 years` will be `$14,500` by traveling from nodes `(1,2)`, `(2,4)`, `(4,6)`.

```
AMPL
ampl: model HW2Q2.mod;
CPLEX 20.1.0.0: optimal solution; objec
2 dual simplex iterations (0 in phase I
cost = 14500

x :=
1 2   1
1 3   0
1 4   0
1 5   0
1 6   0
2 3   0
2 4   1
2 5   0
2 6   0
3 4   0
3 5   0
3 6   0
4 5   0
4 6   1
5 6   0
;
```

3 - Problem 3

3.1 Model Overview

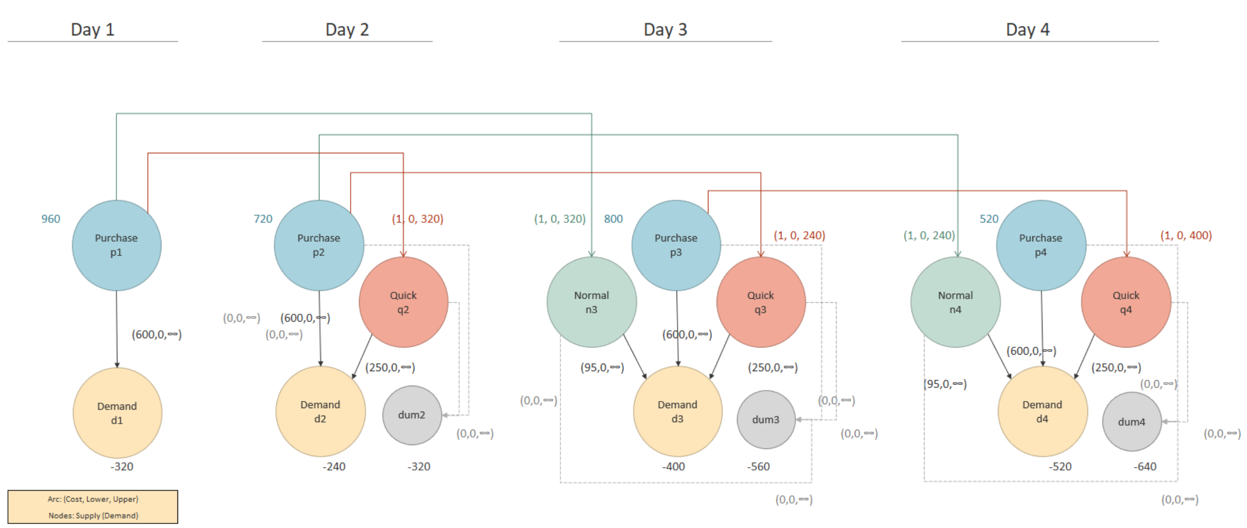
3.1.1 Assumptions and Calculations for Network Flow Diagram

- Below shows how we decided to balance the network with supply and dummy nodes
- In order to obtain a balanced network (i.e. supply equals demand), we must allow for all possible routes to have enough supply.
- Excess supply allowed on certain days is captured by dummy nodes so that the business does not actually produce the tires.

Determine Supply and Dummy Allocation

Day	Demand (d)	Incoming Arcs (in)	Potential Supply *in**d*	Dummy Allocation $d_{day-1} + d_{day-2}$
1	-320	3	960	0
2	-240	3	720	-320
3	-400	2	800	-560
4	-520	1	520	-640

3.1.2 Network Flow Diagram



3.2 Mathematical Formulation

3.2.1 Sets, Parameters, Decision Vars

Set Name	Description
$NODES$	Set of all nodes in above network flow diagram:
`p1 p2 p3 p4`	Number of tires to *purchase* on day ∈(1-4)
`n1 n2 n3 n4`	Number of tires to reshape using the *normal* service on day ∈(1-4)
`q1 q2 q3 q4`	Number of tires to reshape using the *quick* service on day ∈(1-4)
`d1 d2 d3 d4`	Demand of tires on each day ∈(1-4)
`dum2 dum3 dum4`	Dummy nodes to balance excess supply for days 2 3 and 4 ∈(2-4). Do not need one for day 1 since purchasing

The set A is a set of *arcs,* e.g. (i, j) for $i \in N, j \in N$
each of which may carry *flow of a commodity*

Decision variable: x_{ij} determines the units of flow on arc (i, j)

Arc (i, j)

- cost c_{ij} per unit of flow on arc (i, j)
- upper bound on flow of u_{ij} (capacity)
- lower bound on flow of ℓ_{ij} (usually 0)

3.2.2 Objective, and Constraints

$$\begin{aligned} \text{minimize} \quad & \sum_{(i,j)\in A} c_{ij}x_{ij} \\ \text{subject to} \quad & \sum_{j:(i,j)\in A} x_{ij} - \sum_{j:(j,i)\in A} x_{ji} = b_i \quad \forall i \in N \\ & l_{ij} \le x_{ij} \le u_{ij} \qquad\qquad \forall (i,j) \in A \end{aligned}$$

- Upper and lower bounds use to direct the flow of tires from *purchasing* to *quick* or *normal* service

3.3 Code and Output

3.3.1 Model: `group12_HW2_p3.mod`

- Used `mcnfp.txt` from course website and renamed to `group12_HW2_p3.mod`.
- Added `data group12_HW2_p3.dat; solve;` and `display x;`

3.3.2 Data: `group12_HW2_p3.dat`

```
# Problem 4

data;

# Set of Nodes for each day (1,2,3,4)
set NODES :=    p1 p2 p3 p4     # Purchases
                n1 n2 n3 n4     # Normal Service
                q1 q2 q3 q4     # Quick Service
                d1 d2 d3 d4     # Demand
                dum2 dum3 dum4  # Dummy for days 2 3 and 4
                ;

# Set of ARCS from Node i to Node j
set ARCS :=
            (p1, *) d1 q2 n3
            (p2, *) d2 q3 n4 dum2
            (p3, *) d3 q4    dum3
            (p4, dum4)
            (q2, *) d2       dum2
            (q3, *) d3       dum3
            (q4, *) d4       dum4
            (n3, *) d3       dum3
            (n4, *) d4       dum4
            ;

# Number of Products demanded
param b :=
    # "Supply" nodes. Not actually supply but needs to have origin
    p1 960
    p2 720
    p3 800
    p4 520

    # Transshippment
    n1 0 n2 0 n3 0 n4 0
    q1 0 q2 0 q3 0 q4 0

    # Dummy Nodes to take excess supply
    dum2 -320 # Dif between d2 and p2
    dum3 -560 # ""3
    dum4 -640

    # Demand nodes
    d1   -320
    d2   -240
    d3   -400
    d4   -520
    ;
```

Data Continued:

```
# The cost (c), lower (l). and upper (u), from node i to j in IN NODES
param:      c       l   u :=
    [p1, d1] 600    .   .
    [p2, d2] 600    .   .
    [p3, d3] 600    .   .
    [q2, d2] 250    .   .
    [q3, d3] 250    .   .
    [q4, d4] 250    .   .
    [n3, d3] 95     .   .
    [n4, d4] 95     .   .
    [p1, q2] 1      .   320
    [p1, n3] 1      .   320
    [p2, q3] 1      .   240
    [p2, n4] 1      .   240
    [p3, q4] 1      .   400
    [p2, dum2] 0    .   .
    [p3, dum3] 0    .   .
    [p4, dum4] 0    .   .
    [q2, dum2] 0    .   .
    [q3, dum3] 0    .   .
    [q4, dum4] 0    .   .
    [n3, dum3] 0    .   .
    [n4, dum4] 0    .   .
    ;
```

3.3.3 Output

- Total minimized cost: `396,720`
- Interpretation of the tires purchased on each day:

  1. 320 tires purchased
  2. 240 tires reshaped with Quick Service from previous day
  3. 80 Reshaped with quick service from previous day. 320 tires used from reshaping via Normal service from day 1.
  4. 280 Reshaped with quick service from previous day. 240 tires used from reshaping via Normal service from day 2.

```
ampl: model 'C:\Users\daniel.carpenter\OneDrive - the Chickasaw Na
CPLEX 20.1.0.0: optimal solution; objective 396720
1 dual simplex iterations (0 in phase I)
x [*,*] (tr)
:       n3     n4     p1     p2     p3     p4     q2     q3     q4      :=
d1       .      .    320      .      .      .      .      .      .
d2       .      .      .      0      .      .    240      .      .
d3     320      .      .      .      0      .      .     80      .
d4       .    240      .      .      .      .      .      .    280
dum2     .      .      .    240      .      .     80      .      .
dum3     0      .      .      .    400      .      .    160      .
dum4     .      0      .      .      .    520      .      .    120
n3       .      .    320      .      .      .      .      .      .
n4       .      .      .    240      .      .      .      .      .
q2       .      .    320      .      .      .      .      .      .
q3       .      .      .    240      .      .      .      .      .
q4       .      .      .      .    400      .      .      .      .
;
```

4 - Problem 4

4.1 Model Overview

4.1.1 Assumptions and Calculations for Network Flow Diagram

- Goal of below tables are to put all data on a *per unit of product* basis
- Need to be on per unit basis so that we can effectively minimize the cost
- *Color of tables* correspond to the network nodes on the next page

Labor, Manufacturing, and Transportation Cost Calculations for Arcs

Labor (Cost per Unit Output and Total Supply Available)					
Type	Cost per Person	Unit Output per Person	Cost / Unit	Total Labor Avail	TTL Product Supply
Specialist	$ 2,000	12	$ 166.67	100	1,200
Generalist	$ 1,700	10	$ 170.00	200	2,000

Cost of Transportation			
	Scranton, PA	Utica, NY	Stamford, CT
Per Person	300	250	275
Per Unit of Product (trans. Cost / unit output by type)			
Specialist	$ 25.00	$ 20.83	$ 22.92
Generalist	$ 30.00	$ 25.00	$ 27.50

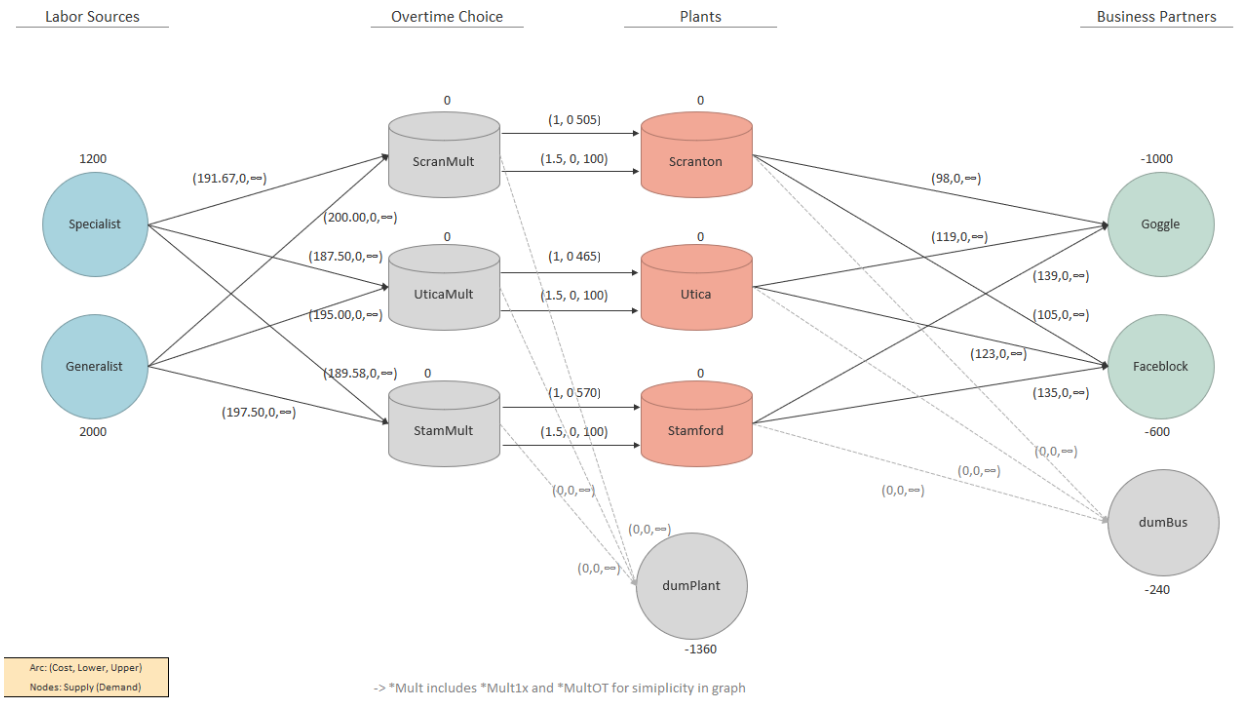
Cost of Transportation + Labor per Unit of Output						
	Scranton, PA	Utica, NY	Stamford, CT	Scranton, PA	Utica, NY	Stamford, CT
Specialist	$ 191.67	$ 187.50	$ 189.58	(191.67,0,∞)	(187.50,0,∞)	(189.58,0,∞)
Generalist	$ 200.00	$ 195.00	$ 197.50	(200.00,0,∞)	(195.00,0,∞)	(197.50,0,∞)

Plant Production Limits			
	Base	OT	OT Mult
Scranton	505	100	1.5
Utica	465	100	1.5
Stamford	570	100	1.5

Manufacturing and Transportation Costs			
	Manufacture	Goggle	Faceblock
Scranton	$ 90	$ 8	$ 15
Utica	$ 105	$ 14	$ 18
Stamford	$ 115	$ 24	$ 20

Man. + Trans Cost	
Goggle	Faceblock
$ 98	$ 105
$ 119	$ 123
$ 139	$ 135

4.1.2 Network Flow Diagram



4.2 Mathematical Formulation

4.2.1 Sets, Parameters, Decision Vars

Set Name	Description
$NODES$	Set of all nodes in above network flow diagram:
`Specialist`, `Generalist`	The two types of Supply of Labor
`ScranMult1x`, `UticaMult1x`, `StamMult1x`	Passed through if *did not* use overtime
`ScranMultOT`, `UticaMultOT`, `StamMultOT`	Passed through if *did* use overtime
`Scranton`, `Utica`, `Stamford`	Transshipment nodes which are the plants
`dumPlant`, `dumBus`	Dummy nodes that account for excess supply from unbalanced supply from labor nodes

The set A is a set of *arcs,* e.g. (i, j) for $i \in N, j \in N$
each of which may carry *flow of a commodity*

Decision variable: x_{ij} determines the units of flow on arc (i, j)

Arc (i, j)

- cost c_{ij} per unit of flow on arc (i, j)
- upper bound on flow of u_{ij} (capacity)
- lower bound on flow of ℓ_{ij} (usually 0)

4.2.2 Objective, and Constraints

$$\begin{aligned}
\text{minimize} \quad & \sum_{(i,j)\in A} c_{ij}x_{ij} \\
\text{subject to} \quad & \sum_{j:(i,j)\in A} x_{ij} - \sum_{j:(j,i)\in A} x_{ji} = b_i \quad \forall i \in N \\
& l_{ij} \le x_{ij} \le u_{ij} \qquad\qquad \forall (i,j) \in A
\end{aligned}$$

- Upper and lower bounds use to direct the flow of the product

4.3 Code and Output

4.3.1 Model: `group12_HW2_p4.mod`

- Used `mcnfp.txt` from course website and renamed to `group12_HW2_p4.mod`.
- Added `data group12_HW2_p4.dat; solve;` and `display x;`

4.3.2 Data: `group12_HW2_p4.dat`

group12_HW2_p4.mod M   group12_HW2_p4.dat M

C: > Users > daniel.carpenter > OneDrive - the Chickasaw Nation > Documents > GitHub > OU-DSA > Metaheuristics > 03 - Homework > HW 02 > AMPL Models >

```
# Homework 2 - Advanced LP & Network Flow Models
# Adv. Analytics and Metaheuristics
# Daniel Carpenter and Christopher Ferguson
# February 2022
# Problem 4

data;

# Set of Nodes containing all sources of Labor, Plants, and Business Partners
set NODES :=    Specialist Generalist                # Labor
                ScranMult1x UticaMult1x StamMult1x   # Multiplier dumBus 1x no OT
                ScranMultOT UticaMultOT StamMultOT   # Multiplier dumBus 1.5x for OT
                dumPlant                             # Dummy to limit supply before plants
                Scranton Utica Stamford              # Plants
                Goggle Faceblock dumBus              # Business Partners + dumBus for unbalanced supply/demand
                ;

# Set of ARCS from Labor Sources, to Plants, to Business Partners
set ARCS :=
            # No OT used
            (Specialist,  *) ScranMult1x UticaMult1x StamMult1x
            (Generalist,  *) ScranMult1x UticaMult1x StamMult1x
            (ScranMult1x, *) Scranton dumPlant
            (UticaMult1x, *) Utica    dumPlant
            (StamMult1x,  *) Stamford dumPlant

            # If Use OT
            (Specialist, *) ScranMultOT UticaMultOT StamMultOT
            (Generalist, *) ScranMultOT UticaMultOT StamMultOT
            (ScranMultOT,*) Scranton dumPlant
            (UticaMultOT,*) Utica    dumPlant
            (StamMultOT, *) Stamford dumPlant

            # Plants to demanders and dumBus for unbalanced network
            (Scranton,  *) Goggle Faceblock dumBus
            (Utica,     *) Goggle Faceblock dumBus
            (Stamford,  *) Goggle Faceblock dumBus
            ;

# Number of Products demanded by Business Partner
param b :=
    # Supply
    Specialist  1200    # 12 units * 100 labor hours
    Generalist  2000    # 10 units * 200 labor hours

    # Transshippment (Plants or OT Multiplier)
    ScranMult1x 0  # Not using OT
    UticaMult1x 0
    StamMult1x  0
    ScranMultOT 0   # Using OT
    UticaMultOT 0
    StamMultOT  0
    Scranton    0   # Plant Arrival
    Utica       0
    Stamford    0
```

Data Continued:

group12_HW2_p4.mod M   group12_HW2_p4.dat M ●

C: > Users > daniel.carpenter > OneDrive - the Chickasaw Nation > Documents > GitHub > OU-DSA > Metaheuristics > 03 - Homework > HW 02 > AMPL M

```
    # Demand
    dumPlant    -1360   # (505+465+570 + 3*100) - (1200+2000)
    Goggle      -1000
    Faceblock   -600
    dumBus      -240    # (-1000-600) - (-1360)
    ;

# The cost (c), lower (l). and upper (u), from node i to j in IN NODES
param:                         c       l   u :=
    # Cost of Transportation + Labor per Unit of Output (see screenshot)
    # PATH **NOT** USING OT ------------------------------
    [Specialist,  ScranMult1x] 191.67  .   .  # Supply -> Mult
    [Specialist,  UticaMult1x] 187.5   .   .
    [Specialist,  StamMult1x]  189.58  .   .
    [Generalist,  ScranMult1x] 200     .   .
    [Generalist,  UticaMult1x] 195     .   .
    [Generalist,  StamMult1x]  197.5   .   .
    [ScranMult1x, Scranton]    1       .   505 # Mult -> Plant
    [UticaMult1x, Utica]       1       .   465
    [StamMult1x,  Stamford]    1       .   570
    [ScranMult1x, dumPlant]    0       .   .  # Mult -> Dummy
    [UticaMult1x, dumPlant]    0       .   .
    [StamMult1x,  dumPlant]    0       .   .

    # PATH USING OT ---------------------------------------
    [Specialist,  ScranMultOT] 191.67  .   .  # Supply -> Mult
    [Specialist,  UticaMultOT] 187.5   .   .
    [Specialist,  StamMultOT]  189.58  .   .
    [Generalist,  ScranMultOT] 200     .   .
    [Generalist,  UticaMultOT] 195     .   .
    [Generalist,  StamMultOT]  197.5   .   .
    [UticaMultOT, Utica]       1.5     .   100 # Mult -> Plant
    [ScranMultOT, Scranton]    1.5     .   100
    [StamMultOT,  Stamford]    1.5     .   100
    [ScranMultOT, dumPlant]    0       .   .  # Mult -> Dummy
    [UticaMultOT, dumPlant]    0       .   .
    [StamMultOT,  dumPlant]    0       .   .

    # Plant to Business Partners
    [Scranton,   Goggle]       98      .   .  # Plant -> Demanders
    [Scranton,   Faceblock]    105     .   .
    [Scranton,   dumBus]       0       .   .
    [Utica,      Goggle]       119     .   .
    [Utica,      Faceblock]    123     .   .
    [Utica,      dumBus]       0       .   .
    [Stamford,   Goggle]       139     .   .
    [Stamford,   Faceblock]    135     .   .
    [Stamford,   dumBus]       0       .   .
    ;
```

4.3.3 Output

- Total minimized cost: `$806,192.95`
- Scranton, Utica, and Stamford produce `0`, `430`, and `170` units of product for **Faceblock**, respectively.
- Scranton, Utica, and Stamford produce `605`, `0`, and `395` units of product for **Goggle**, respectively.
- All possible products produced (using a portion of the available regular and overtime hours). `200` products produced by Specialists using OT, and `100` from generalists using OT.

```
ampl: model 'C:\Users\daniel.carpenter\OneDrive - the Chickasaw Nation\Documents\GitHu
CPLEX 20.1.0.0: optimal solution; objective 806192.95
13 dual simplex iterations (0 in phase I)
x [*,*] (tr)
# $1 = Generalist
# $2 = ScranMult1x
# $3 = ScranMultOT
# $4 = Scranton
# $5 = Specialist
# $6 = StamMult1x
# $7 = StamMultOT
# $8 = Stamford
# $10 = UticaMult1x
# $11 = UticaMultOT
:              $1     $2     $3     $4     $5     $6     $7     $8  Utica    $10    $11 :=
Faceblock       .      .      .      0      .      .      .    430    170      .      .
Goggle          .      .      .    605      .      .      .      0    395      .      .
ScranMult1x     0      .      .      .    505      .      .      .      .      .      .
ScranMultOT     0      .      .      .    100      .      .      .      .      .      .
Scranton        .    505    100      .      .      .      .      .      .      .      .
StamMult1x     75      .      .      .    495      .      .      .      .      .      .
StamMultOT      0      .      .      .    100      .      .      .      .      .      .
Stamford        .      .      .      .      .    570    100      .      .      .      .
Utica           .      .      .      .      .      .      .      .      .    465    100
UticaMult1x  1825      .      .      .      0      .      .      .      .      .      .
UticaMultOT   100      .      .      .      0      .      .      .      .      .      .
dumBus          .      .      .      0      .      .      .    240      0      .      .
dumPlant        .      0      0      .      .      0      0      .      .   1360      0
;
```