Exam 1

Adv. Analytics and Metaheuristics

Daniel Carpenter

March 2022

Contents

1 - Problem 1

1.1 Mathematical Formulation

1.1.1 Sets

Set Name	Description
P	The three types of products, High-Gloss, Semi-Gloss, and Flat

1.1.2 Parameters

Parameter Name	Description
$rawA_p$	The amount of raw ingredient A needed to produce product $p \in P$
$rawB_p$	The amount of raw ingredient B needed to produce product $p \in P$
$demand_p$	The minimum demand to be met for product $p \in P$
$profit_p$	The associated profit for product $p \in P$

1.1.3 Decision Variables

Variable Name	Description
$amtToProduce_p$	The amount of product $p \in P$ to produce and is ion the set of integers

1.1.4 Objective Function

$$maximize\ theProfit : \sum_p amtToProduce_p \times profit_p$$

1.1.5 Constraints

C1: Meet the minimum demand for each product

$$meetMinDemand : amtToProduce_p \geq demand_p, \ \forall \ p \in P$$

C2: Cannot exceed the supply of Raw Material A

$$rawSupplyA : \sum_p amtToProduce_p \times rawA_p \leq 4,000$$

C3: Cannot exceed the supply of Raw Material B

$$rawSupplyB : \sum_p amtToProduce_p \times rawB_p \leq 6,000$$

C4: Ratio of 3:2 for High and Semi Gloss, respectively

- Since $\frac{3}{2} = 1.5$, the amount of high gloss produced must always be $1.5\times$ semi gloss

$$highToSemiRatio : 1.5 \times amtToProduce_{Semi \in P} = amtToProduce_{High \in P}$$

C5: Non-Negativity Constraints and is Integer

$$amtToProduce \geq 0, \in \mathbb{Z}$$

1.2 Code and Output

1.2.1 Code

problem1.mod U

C: > Users > daniel.carpenter > OneDrive - the Chickasaw Nation > Documents > GitHub > OU-DSA > Metaheuristics > 04 - Exams > Exam 1

```
# Daniel Carpenter
# Exam 1
# Problem 1

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

# SETS ============================================================
    set P circular; # The three types of products, High-Gloss, Semi-Gloss, and Flat

# PARAMETERS ======================================================
    param  rawA   {P} >= 0; # Raw ingredient A needed to produce product p ∈ P;
    param  rawB   {P} >= 0; # Raw ingredient B needed to produce product p ∈ P;
    param  demand {P} >= 0; # Min demand to be met for product p ∈ P;
    param  profit {P} >= 0; # Associated profit for product p ∈ P;

# DECISION VARIABLES ==============================================
    var amtToProduce {P} >=0 integer; # amount of product p ∈ P to produce

# OBJECTIVE FUNCTION ==============================================
    maximize theProfit: sum{p in P} amtToProduce[p] * profit[p];

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

    # C1: Meet the minimum demand for each product
    s.t. meetMinDemand{p in P}: amtToProduce[p] >= demand[p];

    # C2/3: Cannot exceed the supply of Raw Material A or B
    s.t. rawSupplyA: sum{p in P} amtToProduce[p] * rawA[p] <= 4000;
    s.t. rawSupplyB: sum{p in P} amtToProduce[p] * rawB[p] <= 6000;

    # C4: Ratio of 3:2 for High and Semi Gloss, respectively
    s.t. highToSemiRatio {p in P}: 1.5 * amtToProduce[last(P)] == amtToProduce[first(P)];

# CONTROLS ========================================================
    data problem1.dat;
    solve;

    print;
    print "For each product, product the following amounts:";
    display amtToProduce;
```

problem1.dat U

C: > Users > daniel.carpenter > OneDrive - the Chickasaw Nation > Documents > GitHub > OU-DSA > Metaheuristics > 04

```
# Daniel Carpenter
# Exam 1
# Problem 1

data;

# The three types of products, High-Gloss, Semi-Gloss, and Flat
set P := High Flat Semi;

#  rawA     The amount of raw ingredient A needed to produce product p ∈ P;
#  rawB     The amount of raw ingredient B needed to produce product p ∈ P;
#  demand   The minimum demand to be met for product p ∈ P;
#  profit   The associated profit for product p ∈ P;
param:
            rawA    rawB    demand  profit :=
    High    2       4       200     30
    Semi    3       2       200     20
    Flat    5       7       150     50
    ;
```

1.2.2 Output

- Not High to Semi is a 3:2 ratio and all demand and supply constraints are satisfied.

```
ampl: model 'C:\Users\daniel.carpenter\OneDrive - the Chickas
CPLEX 20.1.0.0: optimal integer solution; objective 42700
4 MIP simplex iterations
0 branch-and-bound nodes

For each product, product the following amounts:
amtToProduce [*] :=
High   810
Flat   152
Semi   540
;
```

2 - Problem 2

2.1 Additions to Model

Please assume that the mathematical formulation in the course videos are present as well (see code), just named objects are named differently but hopefully are convienent to interpret

2.1.1 Parameters

Name	Description
M	"Big M," which is a large scaler used to help model disjunctive constraints

2.1.2 New Decision Vars

Name	Description
z	Determines which constraint to activate.
$totalProduction$	Simple variable that is Zappers + Spacerays

2.1.3 New Constraints

Description	Constraint
Set Total Production to Zappers + Spacerays	$setTotalProd : totalProduction = spaceRays + zappers$
If total production is ≥ 400	$isGreaterThan400 : totalProduction \geq 400 + M \times z$
then zappers must be 70% of total	$zappersAre70Perc : zappers \geq 0.70 \times totalProduction + M \times z$
If total Production is ≤ 400	$isLessThan400 : totalProduction \leq 400 + M \times (1 - z)$
then no zappers at all	$noZappers : spaceRays \geq totalProduction - M \times (1 - z)$

2.2 Code and Output

2.2.1 Code

problem2.mod M ×

C: > Users > daniel.carpenter > OneDrive - the Chickasaw Nation > Documents > GitHub > OU-DSA > Metaheuristics > 04 - Exan

```
# Daniel Carpenter
# Exam 1
# Problem 2

reset;

# Set up options and the solver
option solver cplex;

# PARAMETERS -----------------------------------------------------
param M := 100000000; # Big M scaler

# DECISION VARIABLES ------------------------------------------
var spaceRays >= 0; # Number of Space Ray Products
var zappers >= 0;   # Number of Zapper Products
var totalProduction >= 0; # Zappers + Spacerays
var z binary; # Determines whether or not constraints are active or not.

# OBJECTIVE ---------------------------------------------------
maximize profit: (8*spaceRays) + (5*zappers);

# CONSTRAINTS -------------------------------------------------
s.t. plastic:    (2*spaceRays) + (1*zappers) <= 1000;
s.t. labor:      (3*spaceRays) + (4*zappers) <= 2400;
s.t. production:   (spaceRays) +   (zappers) <= 700;
s.t. management:   (spaceRays) -   (zappers) <= 350;


# New Constraints

    ## Set totalProduction = Zappers + Spacerays
    s.t. setTotalProd: totalProduction == spaceRays + zappers;

    ## If total Production is >= 400,
    s.t. isGreaterThan400: totalProduction  >= 400                  + M*z;
    s.t. zappersAre70Perc: zappers          >= 0.70*totalProduction  + M*z;

    ## If total Production is <= 400, then no zappers
    s.t. isLessThan400:    totalProduction  <= 400                  + M*(1 - z);
    s.t. noZappers:        spaceRays        >= totalProduction       - M*(1 - z);

# SOLVE -------------------------------------------------------

## Solve the model
solve;

print;
print 'Produce this amount of Space Rays and Zappers';
display spaceRays,zappers, totalProduction;

print 'If Zappers used, then must be 70% of total prod:';
display zappers / totalProduction;
```

2.2.2 Output

- Optimal solution included more than 400, so zappers are 70% of production

```
ampl: model 'C:\Users\daniel.carpenter\OneDrive - the Chi
CPLEX 20.1.0.0: sensitivity
CPLEX 20.1.0.0: optimal solution; objective 3827.027027
3 dual simplex iterations (1 in phase I)

suffix up OUT;
suffix down OUT;
suffix current OUT;

Produce this amount of Space Rays and Zappers
spaceRays = 194.595
zappers = 454.054
totalProduction = 648.649

If Zappers used, then must be 70% of total prod:
zappers/totalProduction = 0.7
```

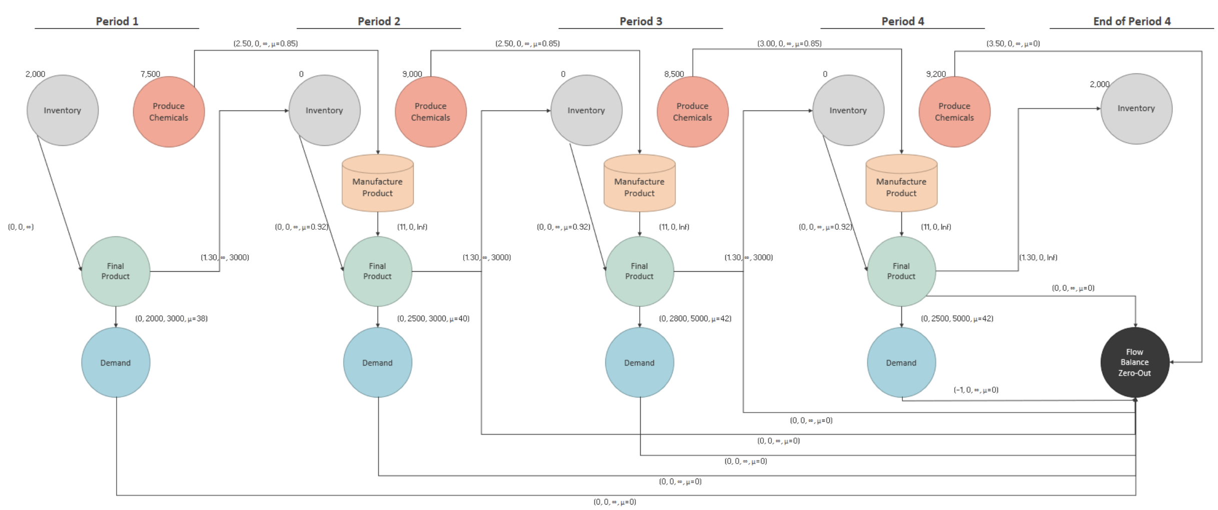
3 - Problem 3

3.1 Model Overview

3.1.1 Assumptions and Calculations for Network Flow Diagram

- Starting/Ending inventory is 2,000
- In period 1, can only use inventory to meet demand since it takes a period to produce the chemicals. Hence, we can not manufacture in period 1.
- μ used to either show:
  - The 85:100 ratio when converting raw materials in production (seen as 0.85)
  - Converting the final product into dollars when selling, or
  - Showing the 8% degradation when storing the final product in inventory that did not sell (seen as 0.92, which is 1 minus 8%)
- To balance the network, a virtual node is used (the black node) and its dependent arcs use multipliers of 0.
  - Note many nodes send through here because there must be minimum demand and maximum, therefore we cannot send all unused product through the demand and then to the balance node
  - So, some products must be zeroed out in the final product stage.
  - Period 4 assumes that some products are produced, but are zeroed out. This assumption is made since it takes a period in advance to manufacture the product. Node shown nonetheless.
- End of period 4 shown like another period for simplicity

3.1.2 Network Flow Diagram



3.2 Mathematical Formulation

3.2.1 Sets, Parameters, Decision Vars

Description	Nodes in the actual model
$NODES$	Set of all nodes in above network flow diagram:
Inventory, plus ending i in period 4	$i1, i2, i3, i4, i4end$
Produce the chemicals	$p1, p2, p3, p4$
Manufacture the Product	$m2, m3, m4$
The Final Product is made	$f1, f2, f3, f4$
Meet the demand	$d1, d2, d3, d4$
Flow balance zero out node	$z4$

Defined on a directed network: $G = (N, A)$

where N is a set of n *nodes*: $\{1, 2, \dots, n\}$

and A is a set of m arcs as a subset of $N \times N$

Each node i has an associated value $b(i)$

Arc (i, j) has certain characteristics:

- 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)
- **multiplier $\mu_{ij} \geq 0$ such that if 1 unit of flow leaves node i, then μ_{ij} units arrive at node j**

3.2.2 Objective, and Constraints

Multiplier

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

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

3.3 Code and Output

3.3.1 Model: `Problem3.mod`

- Used `gmcnfp.txt` from course website and renamed to `Problem3.mod`.
- Added `Problem3.dat;` `solve;` and `display x;`

3.3.2 Data: `Problem3.dat`

Problem3.dat ×

C: > Users > daniel.carpenter > OneDrive - the Chickasaw Nation > Documents > GitHub > OU-DSA > Metaheuristics > 04 - Exams > Exam 1 > AMPL Models > Problem3.dat

```
# Daniel Carpenter
# Exam 1
# Problem 2

#note: default arc costs and lower bounds are 0
#      default arc upper bounds are infinity
#      default node requirements are 0
#      default multiplier is 1

                                 # For each period (1..4)
set NODES :=    i1 i2 i3 i4 i4end  # Inventory, plus ending i in period 4
                p1 p2 p3 p4        # Produce chemicals
                   m2 m3 m4        # Manufacture Product
                f1 f2 f3 f4        # The Final Product
                d1 d2 d3 d4        # The demand
                z4                 # Flow balance zero out node
                ;


set ARCS :=     (i1, f1) (i2, f2) (i3, f3) (i4, f4)     # Inventory to Final Product
                (p1, m2) (p2, m3) (p3, m4) (p4, z4)     # Send product to manufacturing plant
                         (m2, f2) (m3, f3) (m4, f4)     # Manufacturer to the final product
                (f1, d1) (f2, d2) (f3, d3) (f4, d4)     # Final Product to Demand
                (f1, i2) (f2, i3) (f3, i4) (f4, i4end)  # Final Product to next period Inventory
                         (f2, z4) (f3, z4) (f4, z4)     # Send left over product to virtual to zero out/balance
                (d1, z4) (d2, z4) (d3, z4) (d4, z4)
                ;

param b:= i1 2000 # Inital inventory
          p1 7500 # Max production in each period
          p2 9000
          p3 8500
          p4 9200
          i4end -2000
          ;


#specify costs, upper bound, and multipliers for each arc
param:                  c       l       u       mu :=
        [i1, f1]        .       .       .       .       # cost per gallon, st, deterioration rate of 8%
        [i2, f2]        .       .       .       0.92
        [i3, f3]        .       .       .       0.92
        [i4, f4]        .       .       .       0.92
        [p1, m2]        2.50    .       .       0.85    # cost per gallon, 85% of raw materials make gallon of product
        [p2, m3]        2.50    .       .       0.85
        [p3, m4]        3.00    .       .       0.85
        [p4, z4]        3.50    .       .       0
        [m2, f2]        11.00   .       .       .       # cost per gallon
        [m3, f3]        11.00   .       .       .
        [m4, f4]        11.00   .       .       .
        [f1, d1]        .       2000    3000    38      # cost of selling, min/max demand, selling price per gal
        [f2, d2]        .       2500    3000    40
        [f3, d3]        .       2800    5000    42
        [f4, d4]        .       2500    5000    42
        [f1, i2]        1.30    .       3000    .      # Cost of storing, max storage of product
        [f2, i3]        1.30    .       3000    .
        [f3, i4]        1.30    .       3000    .
        [f4, i4end]     1.30    .       .       .      # Must have 2k of product at end in inventory
        [d1, z4]        .       .       .       0      # virtual to balance the network (no incentive)
        [d2, z4]        .       .       .       0
        [d3, z4]        .       .       .       0
        [d4, z4]        .       .       .       0
        [f2, z4]        .       .       .       0
        [f3, z4]        .       .       .       0
        [f4, z4]        .       .       .       0
        ;
```

3.3.3 Output

- Below shows how much of product is sent from node i to node j
- i is displayed on each row, and j is on each column
- For example, f3 sent 2,800 gallons to d3, interpreted as follows:
  - In period 3, 2,800 gallons of product were demanded.
- Note that values sent to virtual node means that nothing those values were not produced.

```
ampl: include 'C:\Users\daniel.carpenter\OneDrive - the Chickasaw Nation\Documents\
CPLEX 20.1.0.0: sensitivity
CPLEX 20.1.0.0: optimal solution; objective 335300
0 dual simplex iterations (0 in phase I)

suffix up OUT;
suffix down OUT;
suffix current OUT;
x [*,*]
:     d1     d2     d3     d4     f1     f2     f3     f4    i2   i3   i4   i4end :=
f1   2000    .      .      .      .      .      .      .     0    .    .     .
f2    .     2500    .      .      .      .      .      .     .    0    .     .
f3    .      .     2800    .      .      .      .      .     .    .    0     .
f4    .      .      .     2500    .      .      .      .     .    .    .    2000
i1    .      .      .      .     2000    .      .      .     .    .    .     .
i2    .      .      .      .      .      0      .      .     .    .    .     .
i3    .      .      .      .      .      .      0      .     .    .    .     .
i4    .      .      .      .      .      .      .      0     .    .    .     .
m2    .      .      .      .      .     6375    .      .     .    .    .     .
m3    .      .      .      .      .      .     7650    .     .    .    .     .
m4    .      .      .      .      .      .      .     7225   .    .    .     .

:     m2     m3     m4      z4      :=
d1    .      .      .     76000
d2    .      .      .     1e+05
d3    .      .      .    117600
d4    .      .      .    105000
f2    .      .      .      3875
f3    .      .      .      4850
f4    .      .      .      2725
p1   7500    .      .       .
p2    .     9000    .       .
p3    .      .     8500     .
p4    .      .      .      9200
;

ampl:
```