Homework 3 - Integer Programming

Adv. Analytics and Metaheuristics

Daniel Carpenter and Iker Zarandona

March 2022

Contents

1 - Problem 1

1.1 Mathematical Formulation

1.1.1 Sets

Set Name	Description
$GENERATORS$	Set of generators i that can be used (A,B,C)
$PERIODS$	2 possible periods p (1, 2) in the production day

1.1.2 Parameters

Parameter Name	Description
S_i	Fixed cost to start a generator ($i \in GENERATORS$) in the entire day
F_i	Fixed cost to operate a generator ($i \in GENERATORS$) in any period
C_i	Variable cost per megawatt to operator a generator ($i \in GENERATORS$) in any period
U_i	Max. megawatts generated for generator ($i \in GENERATORS$) in any period
$demand_p$	Total demanded megawatts for period ($p \in PERIODS$)
M	Large constant to map watts used by each generator ($i \in GENERATORS$)

1.1.3 Decision Variables

Variable Name	Description
$watts_{i,p}$	*Integer variable*: Number of watts to produce per generator ($i \in GENERATORS$) per period ($p \in PERIODS$)
$x_{i,p}$	*Binary variable*: 1 if a generator ($i \in GENERATORS$) is in period p ($p \in PERIODS$), 0 if not turned on at all
y_i	*Binary variable*: 1 if a generator ($i \in GENERATORS$) is used, 0 if not turned on at all

1.1.4 Objective Function

$$minimize\,cost : \sum_{i \in GENERATORS} \Big((\textstyle\sum_{p \in PERIODS} (watts_{i,p}) \times C_i) + (F_i \times \sum_{p \in PERIODS} x_{i,p}) + (S_i \times y_i) \Big)$$

1.1.5 Constraints

C1: For each period, meet the demanded megawatts

$$requiredWatts : \sum_{i \in GENERATORS} (watts_{i,p}) = demand_p, \forall\ p \in PERIODS$$

C2: For each generator, don't surpass the allowable megawatts

$$upperBound : \sum_{p \in PERIODS} (watts_{i,p}) \leq U_i, \forall\ i \in GENERATORS$$

C3: For each generator, map decision variables together to account for the fixed costs in a given day S_i

$$mapVars : \sum_{p \in PERIODS} (watts_{i,p}) \leq M_i \times y_i, \forall\ i \in GENERATORS$$

C4: For each generator and period, map decision variables y and $watts$ together to account for the fixed costs in a per period p

$$mapVars2 : watts_{i,p} \leq M_i \times x_{i,p}, \forall\ i \in GENERATORS,\ p \in PERIODS$$

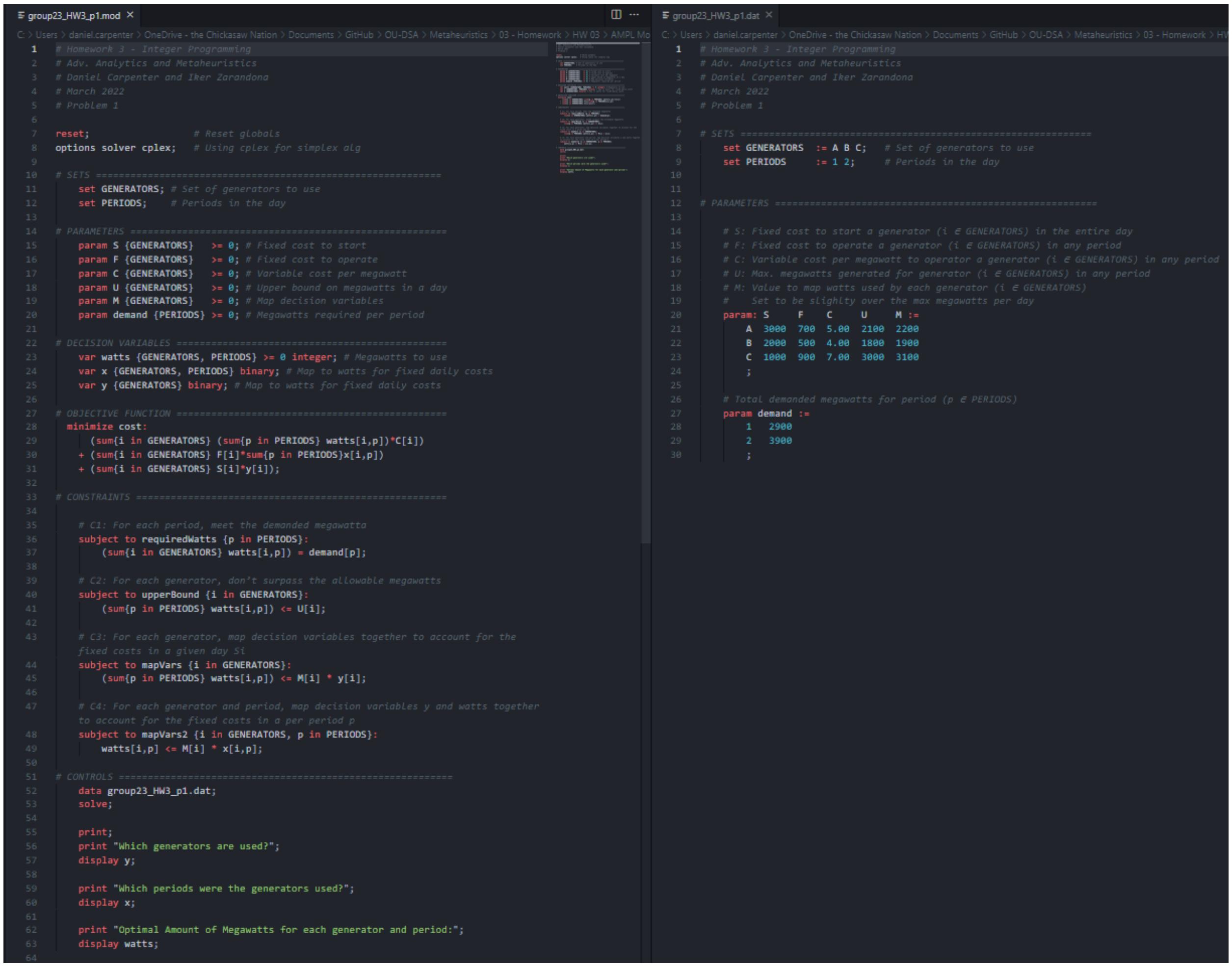
C5 Non-negativity or Binary restraints of decision variables

$$watts_{i,p} \geq 0$$

$$x_{i,p},\ y_i \in (0, 1)$$

1.2 Code and Output

1.2.1 Code



group23_HW3_p1.mod

```
# Homework 3 - Integer Programming
# Adv. Analytics and Metaheuristics
# Daniel Carpenter and Iker Zarandona
# March 2022
# Problem 1

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

# SETS ==========================================================
    set GENERATORS; # Set of generators to use
    set PERIODS;    # Periods in the day

# PARAMETERS ====================================================
    param S {GENERATORS}   >= 0; # Fixed cost to start
    param F {GENERATORS}   >= 0; # Fixed cost to operate
    param C {GENERATORS}   >= 0; # Variable cost per megawatt
    param U {GENERATORS}   >= 0; # Upper bound on megawatts in a day
    param M {GENERATORS}   >= 0; # Map decision variables
    param demand {PERIODS} >= 0; # Megawatts required per period

# DECISION VARIABLES ============================================
    var watts {GENERATORS, PERIODS} >= 0 integer; # Megawatts to use
    var x {GENERATORS, PERIODS} binary; # Map to watts for fixed daily costs
    var y {GENERATORS} binary; # Map to watts for fixed daily costs

# OBJECTIVE FUNCTION ============================================
  minimize cost:
      (sum{i in GENERATORS} (sum{p in PERIODS} watts[i,p])*C[i])
    + (sum{i in GENERATORS} F[i]*sum{p in PERIODS}x[i,p])
    + (sum{i in GENERATORS} S[i]*y[i]);

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

    # C1: For each period, meet the demanded megawatta
    subject to requiredWatts {p in PERIODS}:
        (sum{i in GENERATORS} watts[i,p]) = demand[p];

    # C2: For each generator, don't surpass the allowable megawatts
    subject to upperBound {i in GENERATORS}:
        (sum{p in PERIODS} watts[i,p]) <= U[i];

    # C3: For each generator, map decision variables together to account for the
    fixed costs in a given day Si
    subject to mapVars {i in GENERATORS}:
        (sum{p in PERIODS} watts[i,p]) <= M[i] * y[i];

    # C4: For each generator and period, map decision variables y and watts together
    to account for the fixed costs in a per period p
    subject to mapVars2 {i in GENERATORS, p in PERIODS}:
        watts[i,p] <= M[i] * x[i,p];

# CONTROLS ======================================================
    data group23_HW3_p1.dat;
    solve;

    print;
    print "Which generators are used?";
    display y;

    print "Which periods were the generators used?";
    display x;

    print "Optimal Amount of Megawatts for each generator and period:";
    display watts;
```

group23_HW3_p1.dat

```
# Homework 3 - Integer Programming
# Adv. Analytics and Metaheuristics
# Daniel Carpenter and Iker Zarandona
# March 2022
# Problem 1

# SETS ==========================================================
    set GENERATORS  := A B C;   # Set of generators to use
    set PERIODS     := 1 2;     # Periods in the day


# PARAMETERS ====================================================

    # S: Fixed cost to start a generator (i ∈ GENERATORS) in the entire day
    # F: Fixed cost to operate a generator (i ∈ GENERATORS) in any period
    # C: Variable cost per megawatt to operator a generator (i ∈ GENERATORS) in any period
    # U: Max. megawatts generated for generator (i ∈ GENERATORS) in any period
    # M: Value to map watts used by each generator (i ∈ GENERATORS)
    #    Set to be slighlty over the max megawatts per day
    param: S     F    C     U     M :=
        A  3000  700  5.00  2100  2200
        B  2000  500  4.00  1800  1900
        C  1000  900  7.00  3000  3100
        ;

    # Total demanded megawatts for period (p ∈ PERIODS)
    param demand :=
        1   2900
        2   3900
        ;
```

1.2.2 Output

```
ampl: model 'C:\Users\daniel.carpenter\OneDrive - the Chickasaw
CPLEX 20.1.0.0: optimal integer solution; objective 46100
7 MIP simplex iterations
0 branch-and-bound nodes

Which generators are used?
y [*] :=
A  1
B  1
C  1
;

Which periods were the generators used?
x :=
A 1   0
A 2   1
B 1   0
B 2   1
C 1   1
C 2   0
;

Optimal Amount of Megawatts for each generator and period:
watts :=
A 1      0
A 2   2100
B 1      0
B 2   1800
C 1   2900
C 2      0
;
```

1.2.2.1 Analysis of the Output

- The minimized cost is $\$46,100$
- Generator A, B, and C run
- Generator C runs in period 1. Generator A and B run in period 2
- Generator A produces $2,100$ megawatts in total
- Generator B produces $1,800$ megawatts in total
- Generator C produces $2,900$ megawatts in total

2 - Problem 2

2.1 Mathematical Formulation (Part a)

2.1.1 Sets

Set Name	Description
$PRODUCTS$	5 types of landscaping and construction products (e.g., cement, sand, etc.) labeled product (p) $A, B, C, D,$ and E
$SILOS$	8 different silos s that each product must be stored in $(1, 2, \ldots, 8)$

2.1.2 Parameters

Parameter Name	Description
$cost_{s,p}$	Cost of storing *one ton* of product $p \in PRODUCTS$ in silo $s \in SILOS$
$supply_p$	Total supply *in tons* available of product $p \in PRODUCTS$
$capacity_s$	Total capacity *in tons* of silo $s \in SILOS$. Can store products.
M	Variable to map *decision variable* $tonsOfProduct_{p,s}$ to $isStored_{p,s}$. Uses big M method.

2.1.3 Decision Variables

Variable Name	Description
$tonsOfProduct_{p,s}$	*Tons* of product $p \in PRODUCTS$ to store in silo $s \in SILOS$. Non-negative.
$isStored_{p,s}$	*Binary variable* indicating if product $p \in PRODUCTS$ is stored in silo $s \in SILOS$.

2.1.4 Objective Function

$$minimize\ costOfStorage : \sum_{p \in PRODUCTS} \sum_{s \in SILOS} tonsOfProduct_{p,s} \times cost_{p,s}$$

2.1.5 Constraints

C1: For each silo s, the *tons* of the supplied product p must be less than or equal to the capacity limit of silo s

$$meetCapacity : \sum_{p \in PRODUCTS} tonsOfProduct_{p,s} \leq capacity_s, \ \forall \ s \in SILOS$$

C2: For each product p, must use all of the total product that is available

$$useAllProduct : \sum_{s \in SILOS} tonsOfProduct_{p,s} = supply_p, \ \forall \ p \in PRODUCTS$$

C3: For each silo s and product p,

$$oneProductInSilo : \sum_{pinPRODUCTS} isStored_{p,s} = 1, \ \forall \ s \in SILOS$$

C4: Map the decision variables together using the Big M method

$$mapVars : tonsOfProduct_{p,s} \leq M \times isStored_{p,s}, \ \forall \ p \in PRODUCTS, \ \forall \ s \in SILOS$$

C5 Non-negativity or Binary restraints of decision variables

$$tonsOfProduct_{p,s} \geq 0$$

$$isStored_{p,s} \in (0, 1)$$

2.2 Code and Output (Part a)

2.2.1 Code

group23_HW3_p2.mod

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

```
# Homework 3 - Integer Programming
# Adv. Analytics and Metaheuristics
# Daniel Carpenter and Iker Zarandona
# March 2022
# Problem X

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

# SETS ==============================================================
    set PRODUCTS;     # The 5 products
    set SILOS;        # The 8 silos for storage

# PARAMETERS ========================================================
    param cost     {PRODUCTS, SILOS}; # Cost of storing product in silo
    param supply   {PRODUCTS};        # Supply of products
    param capacity {SILOS};           # Capacity of the silos
    param M;                          # Map decision variables together

# DECISION VARIABLES ================================================
    var tonsOfProduct {PRODUCTS, SILOS} >= 0;  # Amount of each product p to store in silo s
    var isStored      {PRODUCTS, SILOS} binary; # If a product is stored in a silo or not

# OBJECTIVE FUNCTION ================================================

    minimize costOfStorage:
        sum{p in PRODUCTS, s in SILOS} tonsOfProduct[p,s] * cost[p,s];

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

    # C1: For each silo s, the tons of the supplied product p must be less than or equal to
    the capacity limit of silo s
    subject to meetCapacity {s in SILOS}:
        (sum{p in PRODUCTS} tonsOfProduct[p,s]) <= capacity[s];

    # C2: For each product p, must use all of the total product that is available
    subject to useAllProduct {p in PRODUCTS}:
        (sum{s in SILOS} tonsOfProduct[p,s]) == supply[p];

    # C3: Only one product can be in a silo
    subject to oneProductInSilo {s in SILOS}:
        sum{p in PRODUCTS} isStored[p,s] == 1;

    # C4: Map decision variables together
    subject to mapVars {p in PRODUCTS, s in SILOS}:
        tonsOfProduct[p,s] <= M * isStored[p,s];

# CONTROLS ==========================================================
    data group23_HW3_p2.dat;
    solve;

    print;
    print "Which silo(s) stores what product?";
    display isStored;

    print "Optimal tons of product allocated to each silo:";
    display tonsOfProduct;
```

group23_HW3_p2.dat

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

```
# Homework 3 - Integer Programming
# Adv. Analytics and Metaheuristics
# Daniel Carpenter and Iker Zarandona
# March 2022
# Problem X

# SETS ==============================================================

    set PRODUCTS := A B C D E;      # 5 types of products
    set SILOS    := 1 2 3 4 5 6 7 8; # 8 different silos to store product p

# PARAMETERS ========================================================

    # Cost of storing one ton of product p ∈ PRODUCTS in silo s ∈ SILOS
    param cost:
            1   2   3   4   5   6   7   8 :=
        A   1   2   2   1   4   4   5   3
        B   2   3   3   3   1   4   5   2
        C   3   4   1   2   1   4   5   1
        D   1   1   2   2   3   4   5   2
        E   1   1   1   1   1   1   5   5
        ;

    # Supply of each product that is available
    param supply :=
        A   75
        B   50
        C   25
        D   80
        E   20
        ;

    # Capacity of each silo
    param capacity :=
        1   25
        2   25
        3   30
        4   60
        5   80
        6   85
        7   100
        8   50
        ;

    # Variable to map decision variable tonsOfProduct p,s to
    # isStored p,s . Value is slightly more than the capacity of each silo.
    param M := 200;
```

2.2.2 Output (Part a)

```
ampl: model 'C:\Users\daniel.carpenter\OneDrive - the Chickasaw Nation\D
CPLEX 20.1.0.0: optimal integer solution; objective 320
48 MIP simplex iterations
0 branch-and-bound nodes

Which silo(s) stores what product?
isStored [*,*] (tr)
:   A   B   C   D   E    :=
1   1   0   0   0   0
2   0   0   0   1   0
3   0   0   1   0   0
4   1   0   0   0   0
5   0   1   0   0   0
6   0   0   0   0   1
7   0   0   0   1   0
8   0   0   0   1   0
;

Optimal tons of product allocated to each silo:
tonsOfProduct [*,*] (tr)
:    A    B    C    D    E     :=
1   25    0    0    0    0
2    0    0    0   25    0
3    0    0   25    0    0
4   50    0    0    0    0
5    0   50    0    0    0
6    0    0    0    0   20
7    0    0    0    5    0
8    0    0    0   50    0
;
```

2.2.2.1 Analysis of the Output

- Minimized loading cost for 250 tons of 5 products over the 8 silos is 320 (problem does not state cost units).
- Product A stores 25 $tons$ in $silo$ 1 and 50 $tons$ in $silo$ 4
- Product B stores 50 $tons$ in $silo$ 5
- Product C stores 25 $tons$ in $silo$ 3
- Product D stores 25 $tons$ in $silo$ 2, 5$tons$ in $silo$ 7, and and 50 $tons$ in $silo$ 8
- Product E stores 20 $tons$ in $silo$ 6

2.3 Problem 2 `b`

- Create a new objective that also minimizes the distance between capacity and stored tons of product
- *For each silo, minimize the variance between the total capacity and the tons of product*

$$minimize\ capacityActualVariance : capacity_s - \sum_{p \in PRODUCTS} tonsOfProduct_{p,s},\ \forall s \in SILOS$$

2.4 Code and Output (Part `b`)

2.4.1 Code

group23_HW3_p2.2.mod

```
# Homework 3 - Integer Programming
# Adv. Analytics and Metaheuristics
# Daniel Carpenter and Iker Zarandona
# March 2022
# Problem X

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

# SETS =============================================================
    set PRODUCTS;     # The 5 products
    set SILOS;        # The 8 silos for storage

# PARAMETERS =======================================================
    param cost     {PRODUCTS, SILOS}; # Cost of storing product in silo
    param supply   {PRODUCTS};        # Supply of products
    param capacity {SILOS};           # Capacity of the silos
    param M;                          # Map decision variables together

# DECISION VARIABLES ===============================================
    var tonsOfProduct {PRODUCTS, SILOS} >= 0;  # Amount of each product p to store in silo s
    var isStored      {PRODUCTS, SILOS} binary; # If a product is stored in a silo or not

# OBJECTIVE FUNCTION ===============================================

    # Minimize the cost of the storage
    minimize costOfStorage:
        sum{p in PRODUCTS, s in SILOS} tonsOfProduct[p,s] * cost[p,s];

    # For each silo, minimuze the variance between the total capacity and the tons of product
    minimize capacityActualVariance{s in SILOS}:
        capacity[s] - sum{p in PRODUCTS} tonsOfProduct[p,s];


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

    # C1: For each silo s, the tons of the supplied product p must be less than or equal to
    the capacity limit of silo s
    subject to meetCapacity {s in SILOS}:
        (sum{p in PRODUCTS} tonsOfProduct[p,s]) <= capacity[s];

    # C2: For each product p, must use all of the total product that is available
    subject to useAllProduct {p in PRODUCTS}:
        (sum{s in SILOS} tonsOfProduct[p,s]) == supply[p];

    # C3: Only one product can be in a silo
    subject to oneProductInSilo {s in SILOS}:
        sum{p in PRODUCTS} isStored[p,s] == 1;

    # C4: Map decision variables together
    subject to mapVars {p in PRODUCTS, s in SILOS}:
        tonsOfProduct[p,s] <= M * isStored[p,s];

# CONTROLS =========================================================
    data group23_HW3_p2.2.dat;
    solve;

    print;
    print "Which silo(s) stores what product?";
    display isStored;

    print "Optimal tons of product allocated to each silo:";
    display tonsOfProduct;
```

group23_HW3_p2.2.dat

```
# Homework 3 - Integer Programming
# Adv. Analytics and Metaheuristics
# Daniel Carpenter and Iker Zarandona
# March 2022
# Problem X

# SETS =============================================================

    set PRODUCTS := A B C D E;      # 5 types of products
    set SILOS    := 1 2 3 4 5 6 7 8; # 8 different silos to store product p

# PARAMETERS =======================================================

    # Cost of storing one ton of product p ∈ PRODUCTS in silo s ∈ SILOS
    param cost:
            1   2   3   4   5   6   7   8 :=
        A   1   2   2   1   4   4   5   3
        B   2   3   3   3   1   4   5   2
        C   3   4   1   2   1   4   5   1
        D   1   1   2   2   3   4   5   2
        E   1   1   1   1   1   1   5   5
        ;

    # Supply of each product that is available
    param supply :=
        A   75
        B   50
        C   25
        D   80
        E   20
        ;

    # Capacity of each silo
    param capacity :=
        1   25
        2   25
        3   30
        4   60
        5   80
        6   85
        7   100
        8   50
        ;

    # Big M Scaler for mapping decision variables in binary prog.
    param M := 1000;
```

2.4.2 Output (Part `b`)

```
ampl: model 'C:\Users\daniel.carpenter\OneDrive - the Chickasaw
CPLEX 20.1.0.0: optimal integer solution; objective 320
48 MIP simplex iterations
0 branch-and-bound nodes
Objective = costOfStorage

Which silo(s) stores what product?
isStored [*,*] (tr)
:   A   B   C   D   E    :=
1   1   0   0   0   0
2   0   0   0   1   0
3   0   0   1   0   0
4   1   0   0   0   0
5   0   1   0   0   0
6   0   0   0   0   1
7   0   0   0   1   0
8   0   0   0   1   0
;

Optimal tons of product allocated to each silo:
tonsOfProduct [*,*] (tr)
:   A    B    C    D    E     :=
1   25    0    0    0    0
2    0    0    0   25    0
3    0    0   25    0    0
4   50    0    0    0    0
5    0   50    0    0    0
6    0    0    0    0   20
7    0    0    0    5    0
8    0    0    0   50    0
;
```

2.4.2.1 Analysis of the Output

- The optimal cost actually stays the same, but the amount of iterations to get to that solution is much more.
- The values of the decision variables are the same.

3 - Problem 3

3.1 Mathematical Formulation

3.1.1 Sets

Set Name	Description

3.1.2 Parameters

Parameter Name	Description

3.1.3 Decision Variables

Variable Name	Description

3.1.4 Objective Function

3.1.5 Constraints

C1:

3.2 Code and Output

3.2.1 Code

group23_HW3_p3.mod

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

```
# Homework 3 - Integer Programming
# Adv. Analytics and Metaheuristics
# Daniel Carpenter and Iker Zarandona
# March 2022
# Problem 3

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

# GLOBAL PARAMETERS ================================================
param theDemand := 55000;     # The demanded amount of products
param M         := 10000000; # Large scaler that is not inf

# WII - Basic Marginal Cost Model ===============================

    # PARAMETERS ---------------------------------------------------
    param mcWII    := 4.95;   # Marginal cost compnent of WII
    param availWII := 18000;  # Amount of WII that is available

    # DECISION VARIABLES -------------------------------------------
    var WII >= 0;   #amt of product WOW to produce

    # CONSTRAINTS --------------------------------------------------
    s.t. upperBoundWII: WII <= availWII;

# END OF WII - Basic Marginal Cost Model =========================


# WRS - Marginal Cost + Fixed Cost Model =========================

    # PARAMETERS ---------------------------------------------------
    param mcWRS    := 2.30;  # Marginal cost compnent of WRS
    param fixWRS   := 20000; # Fixed Cost component of WRS
    param availWRS := 14000; # Amount of WRS that is available

    # DECISION VARIABLES -------------------------------------------
    var WRS >= 0;     # amt of product WRS to produce
    var yWRS1 binary; # Binary used for fixed cost if used

    # CONSTRAINTS --------------------------------------------------
    s.t. map_yWRS1: WRS <= availWRS * yWRS1; # Upper bound and map

# END OF WRS - Basic Marginal Cost Model =========================


# WE - Basic Marginal Cost Model ===============================

    # PARAMETERS ---------------------------------------------------
    param mcWE1   := 3.95;  # If buy from WRS, m. cost for WE
    param mcWE2   := 4.10;  # Else m. cost for WE
    param availWE := 7000;  # Amount of WE that is available

    # DECISION VARIABLES -------------------------------------------
        # WE decision vars
        var WE1 >= 0;   # Decision variable associated with $3.95 marginal cost
        var WE2 >= 0;   # Decision variable associated with $4.10 marginal cost
        var WE  >= 0;   # Decision variable for final output

        # Binary Vars to see what product is selected
        var yWRS  binary;  # If WRS is selected
        var yWII  binary;  # If WII is selected
        var yWE1  binary;  # If WE  is selected
        var yWE2  binary;  # If WE  is selected
        var z     binary;  # Octivates only one constraint
```

group23_HW3_p3.mod

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

```
    # CONSTRAINTS --------------------------------------------------------
        # Map binary variables to show selection of products
        s.t. mapWE1: WE1  <= M * yWE1; # Map the W vars to the y binary
        s.t. mapWE2: WE2  <= M * yWE2; # ""
        s.t. mapWRS: WRS  <= M * yWRS; # ""
        s.t. mapWII: WII  <= M * yWII; # ""

        # Logical Constraints
            # If buy from WRS, then can do WE1. (Use of Mz to choose one constraint)
            s.t. ifWRS_ThenWE1:    yWRS <= yWE1 + M*z;

            # If WE2, cannot do WII. (Use of Mz to choose one constraint)
            s.t. ifWRS_thenNotWII: yWE2 +  yWII <= 1 + M*(1-z);

            # If WE1, then cannot do WE2, Must choose one
            s.t. only1WE:   yWE1 + yWE2 <= 1;

            # Finally, set WE to the sum of WE1 and WE2 for the final output
            s.t. setWE: WE == WE1 + WE2;

        s.t. upperBoundWE: WE <= availWE; # Meet the upper bound limit

# END OF WE - Basic Marginal Cost Model ========================


# WU - Marginal Cost + Fixed Cost Model ==============================

    # PARAMETERS ---------------------------------------------------------
    param mcWU       := 4.25;  # Marginal cost compnent of WU
    param availWU    := 22000; # Amount of WU that is available
    param minBuyAmt := 15000; # Must buy at least 15k

    # DECISION VARIABLES -------------------------------------------------
    var WU >= 0;    # amt of product WU to produce
    var yWU binary; # Binary used for fixed cost if used

    # CONSTRAINTS --------------------------------------------------------
    s.t. buyAtLeastMin: WU <= availWU   * yWU;    # Buy at least min amount
    s.t. map_yWU:       WU >= minBuyAmt * yWU;  # Under the upper bound

# END OF WU - Basic Marginal Cost Model ========================


# WOW - Piecewise Linear Cost Model ===============================

    # PARAMETERS --------------------------------------------------------

    #assume the linear costs for decision variable WOW are as follows
    #cost = 9.50 for     0 <=WOW < 3000
    #cost = 4.90 for  3000 <=WOW < 9000
    #cost = 2.75 for  9000 <=WOW < INFINITY

    param mcWOW1 := 9.50; param mcWOW1Upper := 3000;  # 3000 upper bound
    param mcWOW2 := 4.90; param mcWOW2Upper := 6000;  # 3000 + 6000 = 9000 upper bound
    param mcWOW3 := 2.75; param mcWOW3Upper := 25000; # Cannot exceed 25000 due to supply
    # DECISION VARIABLES -------------------------------------------------
    var WOW >= 0;   #amt of product WOW to produce

    var d1WOW >=0;  # piecewise component 1 of var WOW
    var d2WOW >=0;  # piecewise component 2 of var WOW
    var d3WOW >=0;  # piecewise component 3 of var WOW

    var y1WOW binary; #to model piecewise cost for var WOW
    var y2WOW binary; #to model piecewise cost for var WOW

```

group23_HW3_p3.mod

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

```

    # CONSTRAINTS -----------------------------------------------------

        #connect WOW with d1WOW, d2WOW, and d3WOW;
        s.t. X_WOW: WOW = d1WOW + d2WOW + d3WOW;

        #ensure that the piece wise costs are used correctly,
        #i.e., you have to use all of d1WOW before you use d2WOW,...
        # First Piece (Between 0 and Upper)
        s.t. piece1a: mcWOW1Upper*y1WOW <= d1WOW;
        s.t. piece1b: d1WOW <= mcWOW1Upper;

        # Second Piece (Between last piece and Upper)
        s.t. piece2a: mcWOW2Upper*y2WOW <= d2WOW;
        s.t. piece2b: d2WOW <= mcWOW2Upper*y1WOW;

        # Third Piece (Between last piece and Upper)
        s.t. piece3: d3WOW <= mcWOW3Upper*y2WOW;

        # Cannot go over upper
        s.t. upperBoundWOW: WOW <= mcWOW3Upper;

# END OF WOW - Piecewise Linear Cost Model ==========================

# Last Constraint: Must meet the demand
s.t. meetTheDemand: WII + WRS + WE + WU + WOW >= theDemand;


# ==================================================================
# OBJECTIVE FUNCTION
# ==================================================================

minimize cost:    mcWII*WII                 # WII: Variable cost only
                + fixWRS*yWRS1 + mcWRS*WRS  # WRS: Fixed plus variable
                + mcWE1*WE1    + mcWE2*WE2  # WE: Continguint mc based on scenario
                + mcWU*WU                   # WU: Restricted range to over 15k
                + mcWOW1*d1WOW + mcWOW2*d2WOW + mcWOW3*d3WOW # WOW: Piecewise
                ;


# CONTROLS =========================================================

solve;

print;
printf  "Demand\t| WII\t| WRS\t| WE\t|  WU\t|  WOW\t| Total Cost";
printf "\n%s\t %s\t %s\t %s\t %s\t %s\t %f", theDemand, WII, WRS, WE, WU, WOW, cost;
print;

```

3.2.2 Output

Summary table of Output

Demand	WII	WRS	WE	WU	WOW	Total Cost
5000	0	0	5000	0	0	19750.000000
10000	3000	0	7000	0	0	42500.000000
25000	4000	14000	7000	0	0	99650.000000
35000	0	14000	6000	15000	0	139650.000000
45000	0	14000	6000	0	25000	177800.000000
50000	4000	14000	7000	0	25000	201550.000000
55000	0	14000	1000	15000	25000	221800.000000

Snapshots of Compilation

```
ampl: model 'C:\Users\daniel.carpenter\OneDrive - the Chickasaw
CPLEX 20.1.0.0: optimal integer solution; objective 19750
3 MIP simplex iterations
0 branch-and-bound nodes

Demand    | WII     | WRS     | WE      |  WU     |  WOW    | Total Cost
5000        0         0         5000      0         0         19750.000000
```

```
Demand    | WII     | WRS     | WE      |  WU     |  WOW    | Total Cost
10000       3000      0         7000      0         0         42500.000000
```

```
Demand    | WII     | WRS     | WE      |  WU     |  WOW    | Total Cost
25000       4000      14000     7000      0         0         99650.000000
```

```
Demand    | WII     | WRS     | WE      |  WU     |  WOW    | Total Cost
35000       0         14000     6000      15000     0         139650.000000
```

```
Demand    | WII     | WRS     | WE      |  WU     |  WOW    | Total Cost
45000       0         14000     6000      0         25000     177800.000000
```

```
Demand    | WII     | WRS     | WE      |  WU     |  WOW    | Total Cost
50000       4000      14000     7000      0         25000     201550.000000
```

```
Demand    | WII     | WRS     | WE      |  WU     |  WOW    | Total Cost
55000       0         14000     1000      15000     25000     221800.000000
```