

# Homework 3 - Integer Programming

Adv. Analytics and Metaheuristics

Daniel Carpenter and Iker Zarandona

March 2022

## Contents

<b>1 - Problem 1</b>	<b>2</b>
1.1 Mathematical Formulation . . . . .	2
1.2 Code and Output . . . . .	4
<b>2 - Problem 2</b>	<b>6</b>
2.1 Mathematical Formulation (Part a) . . . . .	6
2.2 Code and Output (Part a) . . . . .	8
2.3 Problem 2 b . . . . .	10
2.4 Code and Output (Part b) . . . . .	10

# 1 - Problem 1

## 1.1 Mathematical Formulation

### 1.1.1 Sets

Set Name	Description
<i>GENERATORS</i>	Set of generators $i$ that can be used (A,B,C)
<i>PERIODS</i>	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
$watts_{i,p}$	<i>Integer variable:</i> Number of watts to produce per generator ( $i \in GENERATORS$ ) per period ( $p \in PERIODS$ )
$x_{i,p}$	<i>Binary variable:</i> 1 if a generator ( $i \in GENERATORS$ ) is in period $p$ ( $p \in PERIODS$ ), 0 if not turned on at all
$y_i$	<i>Binary variable:</i> 1 if a generator ( $i \in GENERATORS$ ) is used, 0 if not turned on at all

#### 1.1.4 Objective Function

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

#### 1.1.5 Constraints

**C1:** For each period, meet the demanded megawatts

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

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

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

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

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

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

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

**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

```

1 # Homework 3 - Integer Programming
2 # Adv. Analytics and Metaheuristics
3 # Daniel Carpenter and Iker Zarazona
4 # March 2022
5 # Problem 1
6
7 reset; # Reset globals
8 options solver cplex; # Using cplex for simplex alg
9
10 # SETS =====
11 set GENERATORS; # Set of generators to use
12 set PERIODS; # Periods in the day
13
14 # PARAMETERS =====
15 param S (GENERATORS) >= 0; # Fixed cost to start
16 param F (GENERATORS) >= 0; # Fixed cost to operate
17 param C (GENERATORS) >= 0; # Variable cost per megawatt
18 param U (GENERATORS) >= 0; # Upper bound on megawatts in a day
19 param M (GENERATORS) >= 0; # Map decision variables
20 param demand (PERIODS) >= 0; # Megawatts required per period
21
22 # DECISION VARIABLES =====
23 var watts {GENERATORS, PERIODS} >= 0 integer; # Megawatts to use
24 var x {GENERATORS, PERIODS} binary; # Map to watts for fixed daily costs
25 var y {GENERATORS} binary; # Map to watts for fixed daily costs
26
27 # OBJECTIVE FUNCTION =====
28 minimize cost:
29 (sum(i in GENERATORS) (sum(p in PERIODS) watts[i,p])*C[i])
30 + (sum(i in GENERATORS) F[i]*sum(p in PERIODS)x[i,p])
31 + (sum(i in GENERATORS) S[i]*y[i]);
32
33 # CONSTRAINTS =====
34
35 # C1: For each period, meet the demanded megawatts
36 subject to requiredWatts (p in PERIODS):
37 (sum(i in GENERATORS) watts[i,p]) = demand[p];
38
39 # C2: For each generator, don't surpass the allowable megawatts
40 subject to upperBound (i in GENERATORS):
41 (sum(p in PERIODS) watts[i,p]) <= U[i];
42
43 # C3: For each generator, map decision variables together to account for the
44 fixed costs in a given day S1
45 subject to mapVars (i in GENERATORS):
46 (sum(p in PERIODS) watts[i,p]) <= M[i] * y[i];
47
48 # C4: For each generator and period, map decision variables y and watts together
49 to account for the fixed costs in a per period p
50 subject to mapVars2 (i in GENERATORS, p in PERIODS):
51 watts[i,p] <= M[i] * x[i,p];
52
53 # CONTROLS =====
54 data group23_HW3_p1.dat;
55 solve;
56
57 print;
58 print "Which generators are used?";
59 display y;
60
61 print "Which periods were the generators used?";
62 display x;
63
64 print "Optimal Amount of Megawatts for each generator and period:";
65 display watts;
66

```

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

#### 2.1.2 Parameters

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

#### 2.1.3 Decision Variables

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

#### 2.1.4 Objective Function

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

### 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  $p$ ,

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

**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

```
F group23_HW3_p2.mod M X  Untitled-1
C:\Users\danielcarpenter> OneDrive - the Chickasaw Nation > Documents > GitHub > OU-DSA > Metaheuristics > 03 - Homework > HW 03 > AMPL Models
1 # Homework 3 - Integer programming
2 # Adv. Analytics and Metaheuristics
3 # Daniel Carpenter and Iker Zarazonda
4 # March 2022
5 # Problem X
6
7 reset; # Reset globals
8 options solver cplex; # Using cplex for simplex alg
9
10 # SETS =====
11 set PRODUCTS; # The 5 products
12 set SILOS; # The 8 silos for storage
13
14 # PARAMETERS =====
15 param cost {PRODUCTS, SILOS}; # Cost of storing product in silo
16 param supply {PRODUCTS}; # Supply of products
17 param capacity {SILOS}; # Capacity of the silos
18 param M; # Map decision variables together
19
20 # DECISION VARIABLES =====
21 var tonsOfProduct {PRODUCTS, SILOS} >= 0; # Amount of each product p to store in silo s
22 var isStored {PRODUCTS, SILOS} binary; # If a product is stored in a silo or not
23
24 # OBJECTIVE FUNCTION =====
25
26 minimize costOfStorage:
27 sum(p in PRODUCTS, s in SILOS) tonsOfProduct[p,s] * cost[p,s];
28
29 # CONSTRAINTS =====
30
31 # C1: For each silo s, the tons of the supplied product p must be less than or equal to
32 # the capacity limit of silo s
33 subject to meetCapacity {s in SILOS}:
34 (sum(p in PRODUCTS) tonsOfProduct[p,s]) <= capacity[s];
35
36 # C2: For each product p, must use all of the total product that is available
37 subject to useAllProduct {p in PRODUCTS}:
38 (sum(s in SILOS) tonsOfProduct[p,s]) == supply[p];
39
40 # C3: Only one product can be in a silo
41 subject to oneProductInSilo {s in SILOS}:
42 sum(p in PRODUCTS) isStored[p,s] == 1;
43
44 # C4: Map decision variables together
45 subject to mapVars {p in PRODUCTS, s in SILOS}:
46 tonsOfProduct[p,s] <= M * isStored[p,s];
47
48 # CONTROLS =====
49 data group23_HW3_p2.dat;
50 solve;
51
52 print;
53 print "Which silo(s) stores what product?";
54 display isStored;
55
56 print "Optimal tons of product allocated to each silo:";
57 display tonsOfProduct;
```

```
F group23_HW3_p2.dat M X
C:\Users\danielcarpenter> OneDrive - the Chickasaw Nation > Documents > GitHub > OU-DSA > Metaheuristics > 03 - Homework > HW 03 > AMPL Models
1 # Homework 3 - Integer programming
2 # Adv. Analytics and Metaheuristics
3 # Daniel Carpenter and Iker Zarazonda
4 # March 2022
5 # Problem X
6
7 # SETS =====
8
9 set PRODUCTS := A B C D E; # 5 types of products
10 set SILOS := 1 2 3 4 5 6 7 8; # 8 different silos to store product p
11
12 # PARAMETERS =====
13
14 # Cost of storing one ton of product p in silo s
15 param cost:
16 1 2 3 4 5 6 7 8 :=
17 A 1 2 2 1 4 4 5 3
18 B 2 3 3 3 1 4 5 2
19 C 3 4 1 2 1 4 5 1
20 D 1 1 2 2 3 4 5 2
21 E 1 1 1 1 1 1 5 5
22 ;
23
24 # Supply of each product that is available
25 param supply :=
26 A 75
27 B 50
28 C 25
29 D 80
30 E 20
31 ;
32
33 # Capacity of each silo
34 param capacity :=
35 1 25
36 2 25
37 3 30
38 4 60
39 5 80
40 6 85
41 7 100
42 8 50
43 ;
44
45 # Variable to map decision variable tonsOfProduct p,s to
46 # isStored p,s. Value is slightly more than the capacity of each silo.
47 param M := 200;
```



### 2.2.2 Output (Part a)

```
ampl: model 'C:\Users\daniel.carpenter\OneDrive - the Chickasaw Nation\DCPLEX 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 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*

$$\text{minimize capacityActualVariance : } \text{capacity}_s - \sum_{p \in \text{PRODUCTS}} \text{tonsOfProduct}_{p,s}, \forall s \in \text{SILOS}$$

## 2.4 Code and Output (Part b)

### 2.4.1 Code

```

group23_HW3_p2.2.mod
1 # Homework 3 - Integer Programming
2 # Adv. Analytics and Metaheuristics
3 # Daniel Carpenter and Iker Zarandona
4 # March 2022
5 # Problem X
6
7 reset; # Reset globals
8 options solver cplex; # Using cplex for simplex alg
9
10 # SETS =====
11 set PRODUCTS; # The 5 products
12 set SILOS; # The 8 silos for storage
13
14 # PARAMETERS =====
15 param cost {PRODUCTS, SILOS}; # Cost of storing product in silo
16 param supply {PRODUCTS}; # Supply of products
17 param capacity {SILOS}; # Capacity of the silos
18 param M; # Map decision variables together
19
20 # DECISION VARIABLES =====
21 var tonsOfProduct {PRODUCTS, SILOS} >= 0; # Amount of each product p to store in silo s
22 var isStored {PRODUCTS, SILOS} binary; # If a product is stored in a silo or not
23
24 # OBJECTIVE FUNCTION =====
25
26 # Minimize the cost of the storage
27 minimize costOfStorage:
28   sum(p in PRODUCTS, s in SILOS) tonsOfProduct[p,s] * cost[p,s];
29
30 # For each silo, minimize the variance between the total capacity and the tons of product
31 minimize capacityActualVariance(s in SILOS):
32   capacity[s] - sum(p in PRODUCTS) tonsOfProduct[p,s];
33
34 # CONSTRAINTS =====
35
36 # C1: For each silo s, the tons of the supplied product p must be less than or equal to
37 # the capacity limit of silo s
38 subject to meetCapacity {s in SILOS}:
39   (sum(p in PRODUCTS) tonsOfProduct[p,s]) <= capacity[s];
40
41 # C2: For each product p, must use all of the total product that is available
42 subject to useAllProduct {p in PRODUCTS}:
43   (sum(s in SILOS) tonsOfProduct[p,s]) == supply[p];
44
45 # C3: Only one product can be in a silo
46 subject to oneProductInSilo {s in SILOS}:
47   sum(p in PRODUCTS) isStored[p,s] == 1;
48
49 # C4: Map decision variables together
50 subject to mapVars {p in PRODUCTS, s in SILOS}:
51   tonsOfProduct[p,s] <= M * isStored[p,s];
52
53 # CONTROLS =====
54 data group23_HW3_p2.2.dat;
55 solve;
56
57 print;
58 print "Which silo(s) stores what product?";
59 display isStored;
60
61 print "Optimal tons of product allocated to each silo:";
62 display tonsOfProduct;
63
64
group23_HW3_p2.2.dat
1 # Homework 3 - Integer Programming
2 # Adv. Analytics and Metaheuristics
3 # Daniel Carpenter and Iker Zarandona
4 # March 2022
5 # Problem X
6
7 # SETS =====
8
9 set PRODUCTS := A B C D E; # 5 types of products
10 set SILOS := 1 2 3 4 5 6 7 8; # 8 different silos to store product p
11
12 # PARAMETERS =====
13
14 # Cost of storing one ton of product p in silo s
15 param cost:
16   1 2 3 4 5 6 7 8 :=
17   A 1 2 2 1 4 4 5 3
18   B 2 3 3 3 1 4 5 2
19   C 3 4 1 2 1 4 5 1
20   D 1 1 2 2 3 4 5 2
21   E 1 1 1 1 1 1 5 5
22 ;
23
24 # Supply of each product that is available
25 param supply :=
26   A 75
27   B 50
28   C 25
29   D 80
30   E 20
31 ;
32
33 # Capacity of each silo
34 param capacity :=
35   1 25
36   2 25
37   3 30
38   4 60
39   5 80
40   6 85
41   7 100
42   8 50
43 ;
44
45 # Big M Scaler for mapping decision variables in binary prog.
46 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.