Methods and Models for Solution of Optimization Problems in Logistics

Assignment 1: Integer Optimization Problems

Mathematical models are taken from the lectures (if not specified in the report). In some problems mathematical models from lectures are changed in an obvious way to fit the problem.

Problem 1. Fixed Cost Problem

Code:

.mod

```
set SITES;
set POL;

param fixed {SITES} >= 0;
param cost {SITES} >= 0;

param rate {SITES, POL} >= 0;
param required {POL} >= 0;


var Build {SITES} binary;

var Water {SITES} >= 0;

minimize Total_Cost:
	sum {i in SITES} (fixed[i] * Build[i] + cost[i] * Water[i]);

subject to Builded {i in SITES}:
	Water[i] <= 1000000000 * Build[i];

subject to Polutants {j in POL}:
	sum {i in SITES} Water[i] * rate[i,j] >= required[j];
```

.dat

```
data;

set SITES :=	1	2	3;
set POL := 1	2;

param fixed := 1 10000	2 60000	3 40000;
param cost := 1 20	2 30	3 40;

param required := 1 80000	2 50000;

param rate(tr):
	1	2	3 :=
1	0.4	0.25	0.2
2	0.3	0.2	0.25;
```

.run

```
model my/1.1/fc.mod;
data my/1.1/fc.dat;

option solver cplex;
option cplex_options 'sensitivity';
option omit_zero_rows 1;
solve;

display Total_Cost > my/1.1/fc.sol;
display Build > my/1.1/fc.sol;
display Water > my/1.1/fc.sol;

exit;
```

.sol

```
Total_Cost = 4010000

Build [*] :=
1  1
;

Water [*] :=
1  2e+05
;
```

Solution:

The cheapest solution is to build only one pollution control station in the 1st site. The cost is 4010000.

Problem 2. Vendor Selection Problem

2.1 Mathematical Model

AMPL names:

Formulation

(2.1) $min \sum_{j \in J} f_j y_j + \sum_{i \in P} \sum_{j \in J} c_{ij} X_{ij}$ — Total_Cost

st

(2.2) $\sum_{j \in J} X_{ij} = d_i, \forall i \in P$ — Demand{p in PRODUCTS}

(2.3) $0 \leq X_{ij} \leq d_i y_j, \forall i \in P, \forall j \in J$ — VendorUsing{p in PRODUCTS, j in VENDORS}

Notation

Sets:

P - set of products	PRODUCTS
J - set of vendors	VENDORS

Parametrs:

d_i = demand for product $i, i \in P$	demand{p in PRODUCTS}
c_{ij} = cost of purchase one unit of product i from vendor $j, i \in P, j \in J$	cost{p in PRODUCTS, j in VENDORS}
f_i = fixed cost of establishing business with vendor j, $j \in J$	fixed{j in VENDORS}

Variables:

X_{ij} = amount of product i to buy from vendor j, $i \in P, j \in J$	Buy{p in PRODUCTS, j in VENDORS}
y_j = to select(1) or not(0) vendor $j, j \in J$	UseVendor{j in VENDORS}

Description

The objective function (2.1) expresses the total cost of purchasing all products and establishing business with vendors. Constraints (2.2) represent a family of constraints, one for each product: demand has to be satisfied. The limits for purchase are defined in bounds (2.3).

Problem size

The resulting model has following dimensions:

- 16 variables (12 integer + 4 binary)
- 3 constraints
- 12 bounds,

AMPL Code

File exam5.2.mod:

```
set PRODUCTS;
set VENDORS;
param demand {PRODUCTS} >= 0;
param fixed {VENDORS} >= 0;
param cost {PRODUCTS, VENDORS} >= 0;
var UseVendor {VENDORS} binary;
var Buy {PRODUCTS, VENDORS} >= 0;
minimize Total_Cost:
```

```
sum {j in VENDORS} UseVendor[j]*fixed[j] + sum {p in PRODUCTS} sum {j in VENDORS} cost[p,j]*Buy[p, j];
subject to Demand{p in PRODUCTS}:
sum {j in VENDORS} Buy[p,j] = demand[p];
subject to VendorUsing {p in PRODUCTS, j in VENDORS}:
Buy[p,j] <= demand[p]*UseVendor[j];
```

File exam5.2.dat:

```
data;
set PRODUCTS := 1 2 3;
set VENDORS := 1 2 3 4;
param demand := 1 80 2 70 3 40;
param fixed := 1 400 2 500 3 300 4 150;
param cost(tr): 1 2 3:=
1 20 40 50
2 48 15 26
3 26 35 18
4 24 50 35;
```

File exam5.2.run:

```
model exam5.2.mod;
```



```
data exam5.2.dat;
option solver cplex;
option cplex_options 'sensitivity';
option omit_zero_rows 1;
solve;
display Total_Distance > exam5.2.sol;
display UseVendor > exam5.2.sol;
display Buy > exam5.2.sol;
exit;
```

File exam5.2.sol:

```
Total_Distance = 4570
UseVendor [*] :=
1 1
2 1
3 1
;
Buy :=
1 1 80
2 2 70
3 3 40
;
```

Solution

The minimal total cost is 4570. It can be achived whith establishing business with vendors 1, 2 and 3 and buying
80 units of product 1 from vendor 1
70 units of product 2 from vendor 2
40 units of product 3 from vendor 3.

Problem 3. Knapsack Problem

Code:

.mod

```
param N >= 0;
param M >= 0;

param value {1..N} >= 0;
#param budget {1..N} >= 0;
#param staff {1..N} >= 0;
param weight {1..M, 1..N};

#param budget_limit;
#param staff_limit;
param limit {1..M};

param not_with {1..N, 1..N} binary default 0;
param with {1..N, 1..N} binary default 0;

var Use {1..N} binary;


maximize Total_Profit:
	sum {i in 1..N} Use[i] * value[i];

subject to Constraints {i in 1..M}:
	sum{j in 1..N} Use[j] * weight[i,j] <= limit[i];

subject to Not_With {i in 1..N, j in 1..N: not_with[i,j] == 1}:
	Use[j] * Use[i] <= 0;

subject to With {i in 1..N, j in 1..N: with[i,j] == 1}:
	Use[i] <= Use[j];


.dat
data;

param N := 15;
param M := 2;

param value :=
1	600
2	400
3	100
4	150
5	80
6	120
7	200
8	220
```

```
9       90
10      380
11      290
12      130
13      80
14      270
15      280;

param weight(tr):
        1       2:=
1       35      5
2       34      3
3       26      4
4       12      2
5       10      2
6       18      2
7       32      4
8       11      1
9       10      1
10      22      5
11      27      3
12      18      2
13      16      2
14      29      4
15      22      3;

param limit := 1 225  2 28;

param not_with :=
1 10 1
5 6 1
6 5 1
10 1 1
11 15 1
15 11 1;

param with :=
3 15 1
4 15 1
8 7 1
13 2 1
14 2 1;
```

.run

```
model my/1.3/kp.mod;
data my/1.3/kp.dat;

option solver cplex;
option cplex_options 'sensitivity';
option omit_zero_rows 1;
solve;
```

```
display Total_Profit > my/1.3/kp.sol;
display Use > my/1.3/kp.sol;

exit;
```

.sol

```
Total_Profit = 2460

Use [*] :=
 1  1
 2  1
 4  1
 6  1
 7  1
 8  1
 9  1
12  1
14  1
15  1
;
```

Solution:

The highest achievable NVP is 2460. Projects 1, 2, 4, 6, 7, 8, 9, 12, 14, 15 should be selected.

Problem 4. Bin Packing Problem

Code:

.mod

```
param M >= 0;
param N >= 0;

param weight {1..M} >= 0;

param limit;


var UseBin {1..N} binary;
var Put {1..M, 1..N} binary;

minimize Bins:
	sum {j in 1..N} UseBin[j];

subject to Items {i in 1..M}:
	sum{j in 1..N} Put[i,j] = 1;

subject to BinLimit {j in 1..N}:
	sum{i in 1..M} weight[i] * Put[i,j] <= limit * UseBin[j];
```

.dat

```
data;

param M := 17;
param N := 8;

param weight :=
1	252
2	252
3	252
4	252
5	228
6	228
7	228
8	180
9	180
10	180
11	140
12	140
13	140
14	120
15	120
16	120
17	120;

param limit := 600;
```

.run

```
model my/1.4/bp.mod;
data my/1.4/bp.dat;

option solver cplex;
option cplex_options 'sensitivity';
option omit_zero_rows 1;
solve;

display Bins > my/1.4/bp.sol;
display UseBin > my/1.4/bp.sol;
display Put > my/1.4/bp.sol;

exit;
```

.sol

```
Bins = 6

UseBin [*] :=
1  1
2  1
3  1
4  1
5  1
6  1
;

Put :=
1  1   1
2  1   1
3  2   1
4  2   1
5  3   1
6  3   1
7  4   1
8  4   1
9  4   1
10 5   1
11 3   1
12 5   1
13 5   1
14 5   1
15 6   1
16 6   1
17 6   1
;
```

Solution:

The minimal number of bins is 6.

Bin	1	2	3	4	5	6
Items	1, 2	3,4	5,6,11	7,8,9	12,13,14	15, 16, 17

Problem 5. Lot Sizing Problem

Code:

.mod

```
param T >= 0;

param demand {1..T} >= 0;
param capacity {1..T} >= 0;

param setup;
param holding;

var Use {1..T} binary;
var Prod {1..T} >= 0;
var Inv {0..T} >= 0;

minimize TotalCost:
	sum {t in 1..T} (Use[t] * setup + holding * Inv[t]);

subject to Inventory {t in 1..T}:
	Inv[t] = Inv[t-1] + Prod[t] - demand[t];

subject to Production {t in 1..T}:
	Prod[t] <= capacity[t] * Use[t];

subject to InitInv:
	Inv[0] = 0;
```

.dat

```
data;

param T := 6;

param: demand capacity :=
1	335	600
2	200	600
3	140	600
4	440	400
5	300	200
6	200	200;


param setup := 200;
param holding := 0.3;
```

.run

```
model my/1.5/ls.mod;
data my/1.5/ls.dat;

option solver cplex;
```

```
option cplex_options 'sensitivity';
option omit_zero_rows 1;
solve;

display TotalCost > my/1.5/ls.sol;
display Prod > my/1.5/ls.sol;
display Inv > my/1.5/ls.sol;
display Use > my/1.5/ls.sol;

exit;
```

.sol

```
TotalCost = 1052

Prod [*] :=
1  535
3  480
4  400
6  200
;

Inv [*] :=
1  200
3  340
4  300
;

Use [*] :=
1  1
3  1
4  1
6  1
;
```

Solution:

The cheapest solution's cost is 1052. The solution is the following.

Day	0	1	2	3	4	5	6
Setup production		Yes	No	Yes	Yes	No	Yes
Produce		535	0	480	400	0	200
Inventory	0	200		340	300	0	0

Problem 6. Job Sequencing Problem

Code:

.mod

```
param N >= 0;
param M;

param procTime {1..N} >= 0;
param goal {1..N} >= 0;
param weight {1..N} >= 0;

var Before {1..N, 1..N} binary;
var Start {1..N} >= 0;
var Delay {1..N} >= 0;

minimize WeightedDelay:
	sum {i in 1..N} weight[i] * Delay[i];

subject to Del {i in 1..N}:
	Delay[i] >= Start[i] + procTime[i] - goal[i];

subject to OrderBefore {i in 1..N, j in 1..N: i<>j}:
	Start[i] + procTime[i] <= Start[j] + M * (1 - Before[i,j]);

subject to OrderAfter {i in 1..N, j in 1..N: i<> j}:
	Start[j] + procTime[j] <= Start[i] + M * Before[i,j];
```

.dat

```
data;

param N := 4;
param M := 1000;

param: procTime goal weight :=
1	6	8	1
2	4	4	1
3	5	12	2
4	8	16	2;
```

.run

```
model my/1.6/js.mod;
data my/1.6/js.dat;

option solver cplex;
option cplex_options 'sensitivity';
option omit_zero_rows 1;
solve;

display Start > my/1.6/js.sol;
```

```
display Delay > my/1.6/js.sol;
display Before > my/1.6/js.sol;

exit;
```

.sol

```
Start [*] :=
1  17
3   4
4   9
;

Delay [*] :=
1  15
4   1
;

Before :=
2 1   1
2 3   1
2 4   1
3 1   1
3 4   1
4 1   1
;
```

Solution:

The minimum weighted delay is 17.
Job sequence is 2 -> 3 -> 4 -> 1.