ETS061 Take Home Assignment 2

Minh Vuong Pham (soc13mph) och Adrian Hansson (dic13aha)

Task 1

*Note: Actual real-world X-values will be X*1000 and Z-values will be Z*1000*10000*
But we used the numbers without zeroes for easier readability.

a) <u>Total profit (Z):</u>
$Z = 4X_1 + 3X_2 + 2X_3 + 2X_4 + 1X_5$

<u>Constraints:</u>
1) $2X_1 \leq 36$
2) $2X_2 + 2X_3 + 2X_4 + X_5 \leq 216$
3) $0.2X_1 + X_2 + 0.5X_4 \leq 18$
4) $X_1 \leq 16$
5) $X_2 \leq 2$
6) $X_1 + X_2 + X_3 \leq 34$
7) $X_4 + X_5 \leq 28$

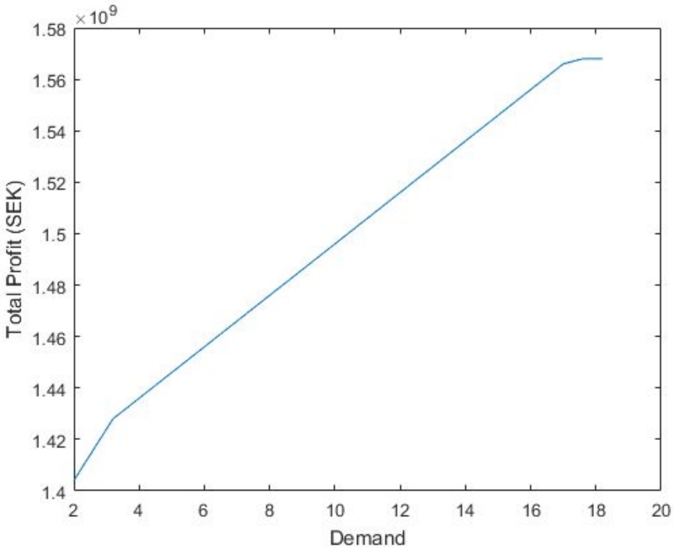
b) $X_1 = 16$
$X_2 = 14.8$
$X_3 = 2$
$X_4 = 0$
$X_5 = 28$
$Z = 140.4$

c)

Constraint	Shadow Price (10 000 SEK)
1	0
2	0
3	3
4	3.4
5	2
6	0

7	1

d)



The shadow price changes as many times as the derivative of the graph above. For an example, at the first change of the graph's derivative, we exceed the maximum allowable increase for Demand III (as seen in the Excel solver)

e) We implemented code in Matlab where we incrementally increased and decreased the X_1-value by 0.01. Our result was:
$\alpha = 0.59 \quad \beta = 4.0$

Task 2

a) $X_1 = 3 \quad X_2 = 2 \quad z = 13$

b) The linear programing gave the solution $X_1 = 1.6 \quad X_2 = 2.6 \quad z = 14.6$ which can be rounded in four different ways:

Rounding 1 (when X_1 and X_2 are rounded down)
$X_1 = 1 \quad X_2 = 2 \quad z = 11$
Fulfills all the constraints

Rounding 2 (when X_1 is rounded up and X_2 is rounded down)
$X_1 = 2 \quad X_2 = 2 \quad z = 12$
Fulfills all the constraints

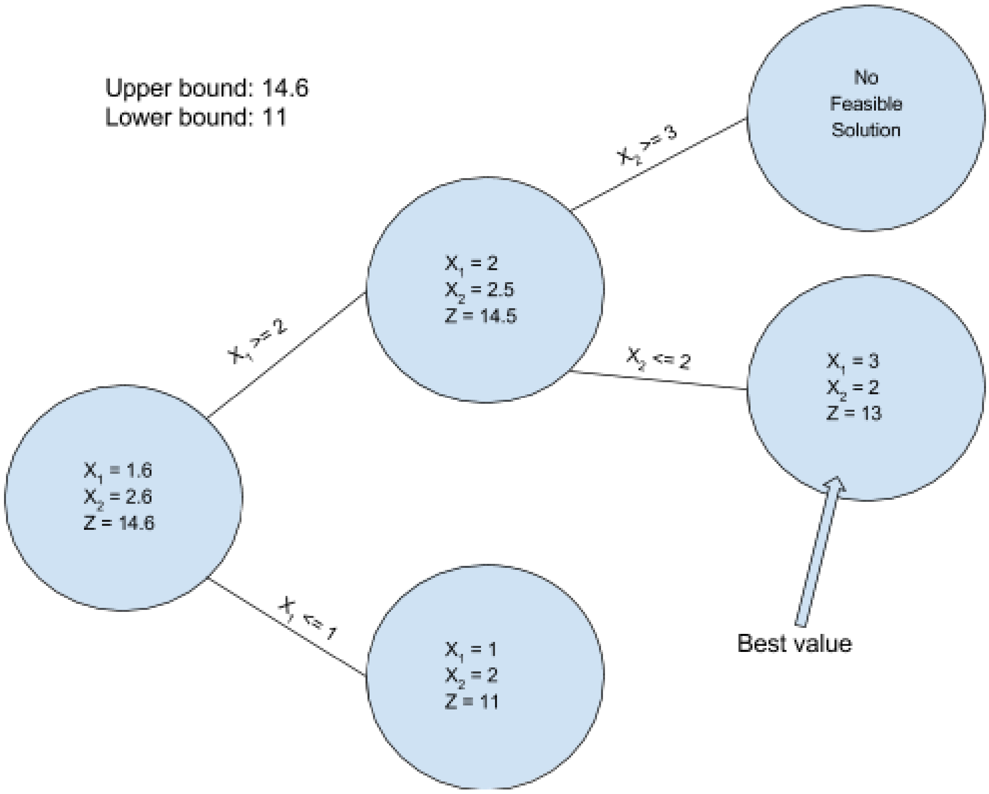
Rounding 3 (when X_1 is rounded down and X_2 is rounded up)
$X_1 = 1 \quad X_2 = 3 \quad z = 16$
Does not achieve constraint 2. $(-x_1 + x_2 \leq 1)$

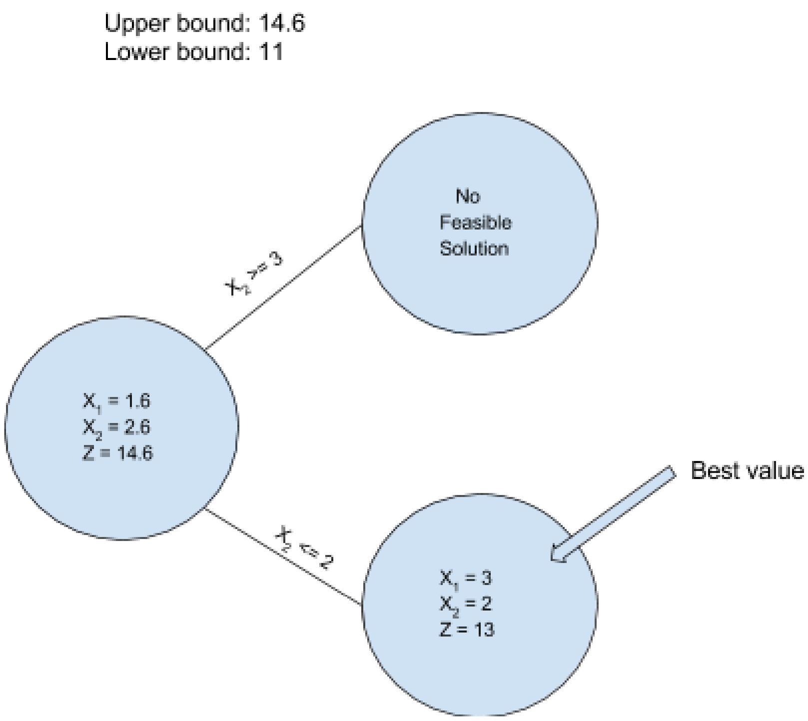
Rounding 4 (when X_1 and X_2 are rounded up)

$X_1 = 2$ $X_2 = 3$ $z = 17$
Does not achieve constraint 3. $(x_1 + 4x_2 \leq 12)$

Rounding 2 is feasible and gives the maximum value for the objective function. This differs from the solution in a) where it gives a lower value (12 < 13) and different numbers on X_1.

c) Here are two different ways to start the branch-and-bound, both of which give the same results.





MATLAB code

Task 1b

```
clc
clear all

c = [-4 -3 -2 -2 -1];

A = [2 0 0 0 0;
    0 2 2 2 1;
    0.2 1 0 0.5 0;
    1 0 0 0 0;
    0 0 1 0 0;
    1 1 1 0 0;
    0 0 0 1 1
    ];

b = [36;
    216;
    18;
    16;
    2;
    34;
    28];
```

```
lb = zeros(5, 1);

options = optimoptions('linprog', 'Algorithm', 'dual-simplex', 'Display', 'off');
[x,fval,exitflag,output,lambda_1b] = linprog(c', A, b, [], [], lb, [], [], options);
```

Task 1c

See attached Excel-file and images below

Task 1d

```
clc
clear all

c = [-4 -3 -2 -2 -1];

A = [2 0 0 0 0;
    0 2 2 2 1;
    0.2 1 0 0.5 0;
    1 0 0 0 0;
    0 0 1 0 0;
    1 1 1 0 0;
    0 0 0 1 1
    ];

 b = [36;
     216;
     18;
     16;
     2;
     34;
     28];

 b_incr = [0;
          0;
          0;
          0;
          0.6;
          0;
          0];

 lb = zeros(5, 1);

options = optimoptions('linprog', 'Algorithm', 'dual-simplex', 'Display', 'off');
[x,fval,exitflag,output,lambda_1b] = linprog(c', A, b, [], [], lb, [], [], options);

oldProfit = 0;
profit = -c*x
```

```
profitArray = [profit];
demandArray = [b(5)];
shadowPriceArray = [20000];

 while true
    oldProfit = profit;

    b = b + b_incr;

    [x,fval,exitflag,output,lambda_1d] = linprog(c', A, b, [], [], lb, [], [], options);
    profit = -c*x
    profitArray=[profitArray profit];
    demandArray=[demandArray b(5)];
    shadowPriceArray=[shadowPriceArray lambda_1d.ineqlin(5)];
    if((profit-oldProfit) < 0.1), break, end
 end
plot(demandArray, profitArray*1000*10000)
ylabel('Total Profit (SEK)')
xlabel('Demand')
```

Task 1e

```
clc
clear all

c = [-4 -3 -2 -2 -1];

A = [2 0 0 0 0;
     0 2 2 2 1;
     0.2 1 0 0.5 0;
     1 0 0 0 0;
     0 0 1 0 0;
     1 1 1 0 0;
     0 0 0 1 1
     ];

b = [36;
     216;
     18;
     16;
     2;
     34;
     28];

lb = zeros(5, 1);
```

```
options = optimoptions('linprog', 'Algorithm', 'dual-simplex', 'Display', 'off');
[x,fval,exitflag,output,lambda_1ea] = linprog(c', A, b, [], [], lb, [], [], options); %to reset our x values
startC = c; %values to reset to for the beta loop
startX = x; %values to reset to for the beta loop

%Now let's start with the actual task 1e
oldX = x;
price_incr = -0.01;
alfa = 4;
beta = 4;

%let's find the lower interval bound
while true
   c = c - [price_incr 0 0 0 0];
   alfa = alfa + price_incr;
   [x,fval,exitflag,output,lambda_1ea] = linprog(c', A, b, [], [], lb, [], [], options);
   if(isequal(oldX, x) == false), break, end
   oldX = x;
end
alfa %lower bound

%let's find the upper interval bound
x = startX;
c = startC;
while true
   c = c + [price_incr 0 0 0 0];
   beta = beta - price_incr;
   [x,fval,exitflag,output,lambda_1eb] = linprog(c', A, b, [], [], lb, [], [], options);
   if(isequal(oldX, x) == false), break, end
   %if(beta > 10), break, end %can be used to break otherwise infinite loop
   oldX = x;
end
beta %upper bound
```

Task 2a

```
clc
clear all

c = [-1;
    -5];
A = [2 -1;
    -1 1;
    1 4];
 b = [4;
     1;
```

```
     12];
 lb = [0;
      0];
 %options = optimoptions('intlinprog', 'Display', 'iter');
 options = optimoptions('intlinprog', 'Display', 'off');
 intcon = [1;
          2];
[x, fval, exitflag, output] = intlinprog(c', intcon, A, b, [], [], lb, [], options)

z=(-c)'*x;
```

Task 2b

```
clc
clear all

c = [-1;
    -5];
A = [2 -1;
    -1 1;
     1 4];
 b = [4;
     1;
     12];
 lb = [0;
      0];
 %options = optimoptions('intlinprog', 'Display', 'iter');
 options = optimoptions('intlinprog', 'Display', 'off');
 intcon = [1;
          2];
[x, fval, exitflag, output] = intlinprog(c', intcon, A, b, [], [], lb, [], options)

z=(-c)'*x;

%Just to test our manually calculated solutions to Task2b using Matlab code
%as well
options_l = optimoptions('linprog', 'Algorithm', 'dual-simplex', 'Display', 'off');
[x_l,fval_l,exitflag_l,output_l,lambda_l] = linprog(c', A, b, [], [], lb, [], [], options_l);
x_l_rounded = round(x_l);
z_l=(-c).*x_l;

x_l_collection = [];
z_l_collection = [];
for i = 1:4
    if(i == 1)
        x_l_rounded = [floor(x_l(1)), floor(x_l(2))];
    elseif (i==2)
```

```
        x_I_rounded = [ceil(x_I(1)), ceil(x_I(2))];
    elseif (i==3)
        x_I_rounded = [ceil(x_I(1)), floor(x_I(2))];
    else
        x_I_rounded = [floor(x_I(1)), ceil(x_I(2))];
    end
    xMulA = x_I_rounded .* A
    xMulASummed = sum(xMulA, 2)

    satisfiesEquation = true;
    for j = 1:3
        if(xMulASummed(j) > b(j))
            satisfiesEquation = false;
        end
    end
    if(satisfiesEquation == true)
        x_I_collection=[x_I_collection x_I_rounded];
        z_I_collection=[z_I_collection ((-c(1))*x_I_rounded(1) + (-c(2))*x_I_rounded(2))];
    end
end
%See x_I_collection and z_I_collection for seeing which answer we got
```

H18

	B	C	D	E	F	G	H	I	J
4		x1	x2	x3	x4	x5			
5	Price	4	3	2	2	1			
6	Results	16	14,8	2	0	28		Total prof	140,4
7									
8	**Constraints**						**Used**	**Capacity**	
9	Macro len	2	0	0	0	0	32	36	
10	Prime len	0	2	2	2	1	61,6	216	
11	Wide lens	0,2	1	0	0,5	0	18	18	
12									
13	PI						16	16	
14	PIII						2	2	
15	PI+PII+PIII						32,8	34	
16	PIV+PV						28	28	

Sensitivity Report 2 | Sheet1

Ready

G21 | 1,2

Variable Cells

Cell	Name	Final Value	Reduced Cost	Objective Coefficient	Allowable Increase	Allowable Decrease
C6	Results x1	16	0	4	1E+30	3,4
D6	Results x2	14,8	0	3	17	1
E6	Results x3	2	0	2	1E+30	2
F6	Results x4	0	-0,5	2	0,5	1E+30
G6	Results x5	28	0	1	1E+30	0,5

Constraints

Cell	Name	Final Value	Shadow Price	Constraint R.H. Side	Allowable Increase	Allowable Decrease
H10	Prime lens Used	61,6	0	216	1E+30	154,4
H11	Wide lens Used	18	3	18	1,2	14,8
H13	PI Used	16	3,4	16	1,5	16
H14	PIII Used	2	2	2	1,2	2
H15	PI+PII+PIII Used	32,8	0	34	1E+30	1,2
H16	PIV+PV Used	28	1	28	154,4	28
H9	Macro lens Used	32	0	36	1E+30	4

Sensitivity Report 2 | Sheet1