Lab06-Linear Programming

CS214-Algorithm and Complexity, Xiaofeng Gao, Spring 2021.

∗ Name: Haoyi You Student ID: 519030910193 Email: yuri-you@sjtu.edu.cn

1. *Hirschberg Algorithm.* Recall the **String Similarity** problem in class, in which we calculate the edit distance between two strings in a sequence alignment manner.

   (a) Implement the algorithm combining **dynamic programming** and **divide-and-conquer** strategy in C/C++. Analyze the time complexity of your algorithm. (The template *Code-SequenceAlignment.cpp* is attached on the course webpage).

   Solution. i. The code please refer to appendix 1.

   ii. Time Complexity:O(mn)
   Let $f(x, y)$ be the time complexity of the problem with x words in X and y words in Y.
   In the algorithm, we divide the problem into two parts with x and $\frac{y}{2}$.For each parts we separately use recursion and dynamic programming. So we can get the recursive formula:

   $$f(x, y) = 2 * f(x, \frac{y}{2}) + 2 * O(x * \frac{y}{2}) \tag{1}$$

   Assume $f(x, y) \leq cxy$.
   Firstly,when $y == 2$ this is correct.
   Hypothesis:when $y < k$ is correct.
   When $y == k$,

   $$f(x, k) \leq 2 * cx\frac{k}{2} + 2 * cx\frac{k}{2} = c * x * k \tag{2}$$

   So from the mathematical induction we get $f(x, y) = O(xy)$

   □

   (b) Given $\alpha(x, y) = |ascii(x) - acsii(y)|$, where $ascii(c)$ is the ASCII code of character c, and $\delta = 13$. Find the edit distance between the following two strings.

   $$X[1..60] = CMQHZZRIQOQJOCFPRWOUXXCEMYSWUJ$$
   $$TAQBKAJIETSJPWUPMZLNLOMOZNLTLQ$$

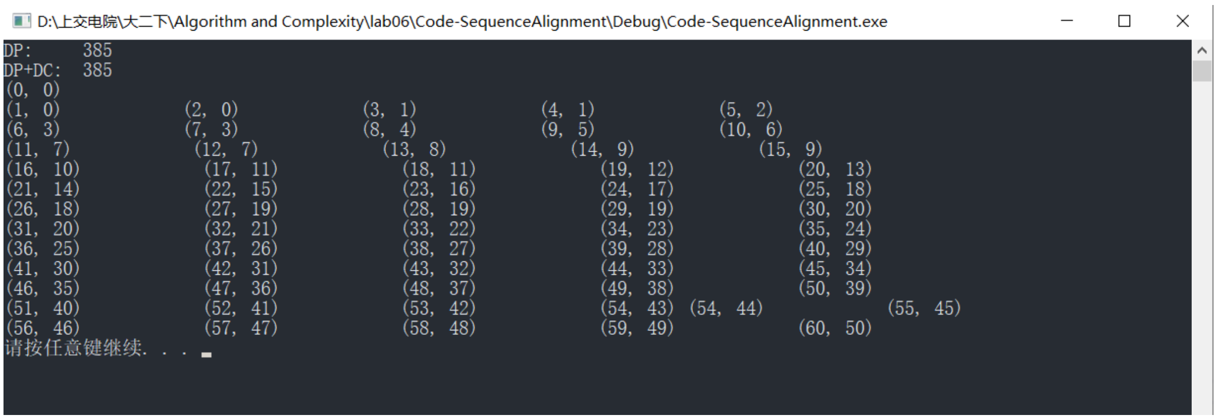
   $$Y[1..50] = SUYLVMUSDROFBXUDCOHAATBKN$$
   $$AAENXEVWNLMYUQRPEOCJOCIMZ$$

   Solution. The answer is 385.The result please refer to figure 1. As for the previous output is too long to print in one figure, I adjust some of the '*endl*' to '\t'. □

2. *Travelling Salesman Problem.* Given a list of cities and the distances between each pair of cities ($G = (V, E, W)$), we want to find the shortest possible route that visits each city exactly once and returns to the origin city. Similar to **Maximum Independent Set** and '

   Dominating Set, please turn the traveling salesman problem into an ILP form.

   Remark: W is the set of weights corresponds to the edges that connecting adjacent cities.



```
D:\上交电院\大二下\Algorithm and Complexity\lab06\Code-SequenceAlignment\Debug\Code-SequenceAlignment.exe
DP:     385
DP+DC:  385
(0, 0)
(1, 0)            (2, 0)            (3, 1)            (4, 1)            (5, 2)
(6, 3)            (7, 3)            (8, 4)            (9, 5)            (10, 6)
(11, 7)            (12, 7)            (13, 8)            (14, 9)            (15, 9)
(16, 10)            (17, 11)            (18, 11)            (19, 12)            (20, 13)
(21, 14)            (22, 15)            (23, 16)            (24, 17)            (25, 18)
(26, 18)            (27, 19)            (28, 19)            (29, 19)            (30, 20)
(31, 20)            (32, 21)            (33, 22)            (34, 23)            (35, 24)
(36, 25)            (37, 26)            (38, 27)            (39, 28)            (40, 29)
(41, 30)            (42, 31)            (43, 32)            (44, 33)            (45, 34)
(46, 35)            (47, 36)            (48, 37)            (49, 38)            (50, 39)
(51, 40)            (52, 41)            (53, 42)            (54, 43) (54, 44)            (55, 45)
(56, 46)            (57, 47)            (58, 48)            (59, 49)            (60, 50)
请按任意键继续. . .
```

Figure 1: Code SequenceAlignment output

Solution.

Suppose that x_e is a 0-1 indicator of edge e, such that

$$x_e = \begin{cases} 1, & e \textit{ is selected in the travelling route} \\ 0, & e \textit{ is not selected in the travelling route} \end{cases} \tag{3}$$

$\sum_{e\in E} x_e * w_e$ is the total length of the route, thus we can formulate the following **ILP**:

$$\begin{aligned} max \quad & \sum_{e\in E} x_e * w_e \\ \text{s.t.} \quad & x_e \in \{0,1\}, \qquad \forall e \in E \\ & d(v) = 2, \qquad \forall v \in V \\ & \sum_{v\in V'} d(v) < 2 * |V'| \qquad \forall V' \subset V \\ with \quad & d(v) = \sum_{e\in N(v)} x_e, \ N(v) \textit{ is the total edges of } v \end{aligned} \tag{4}$$

□

3. *Investment Strategy.* A company intends to invest 0.3 million yuan in 2021, with a proper combination of the following 3 projects:

- **Project 1:** Invest at the beginning of a year, and can receive a 20% profit of the investment in this project at the end of this year. Both the capital and profit can be invested at the beginning of next year;
- **Project 2:** Invest at the beginning of 2021, and can receive a 50% profit of the investment in this project at the end of 2022. The investment in this project cannot exceed 0.15 million dollars;
- **Project 3:** Invest at the beginning of 2022, and can receive a 40% profit of the investment in this project at the end of 2022. The investment in this project cannot exceed 0.1 million dollars.

Assume that the company will invest *all* its money at the beginning of a year. Please design a scheme of investment in 2021 and 2022 which maximizes the overall sum of capital and profit at the end of 2022.

(a) Formulate a linear programming with necessary explanations.

(b) Transform your LP into its standard form and slack form.

(c) Transform your LP into its dual form.

(d) Use the simplex method to solve your LP.

Solution.

(a) Assume in 2021,the investment in Project 1 is x_1 and in Project 2 is x_2. In 2021, the investment in Project 1 is x_3 and in Project 3 is x_4.(unit: million dollars)

i. Total sum of money $S = 1.5 * x_2 + 1.2 * x_3 + 1.4 * x_4$, the target is to maximize S

ii. You cannot invest more money than you have, so there are 2 constrains in formula 5.

$$\begin{aligned} x_1 + x_2 &\leq 0.3 \\ x_3 + x_4 &\leq 0.2 * x_1 + 0.3 - x_2 \end{aligned} \tag{5}$$

iii. You cannot invest negative money. So there are

$$x_i \geq 0, i = 1, 2, 3, 4 \tag{6}$$

(b) i. Stardard form:

$$\begin{aligned} \mathit{maximize} \qquad & 1.5x_2 + 1.2x_3 + 1.4x_4 \\ \textbf{s.t.} \qquad & x_1 + x_2 \leq 0.3 \\ & -0.2x_1 + x_2 + x_3 + x_4 \leq 0.3 \\ & x_2 \leq 0.15 \\ & x_4 \leq 0.1 \\ & x_1, x_2, x_3, x_4 \geq 0 \end{aligned} \tag{7}$$

ii. Slack form:

$$\begin{aligned} \mathit{maximize} \qquad & 1.5x_2 + 1.2x_3 + 1.4x_4 \\ \textbf{s.t.} \qquad & x_1 + x_2 + s_1 = 0.3 \\ & -0.2x_1 + x_2 + x_3 + x_4 + s_2 = 0.3 \\ & x_2 + s_3 = 0.15 \\ & x_4 + s_4 = 0.1 \\ & x_1, x_2, x_3, x_4, s_1, s_2, s_3, s_4 \geq 0 \end{aligned} \tag{8}$$

(c) Dual form:

$$\begin{aligned} \mathit{minimize} \qquad & 0.3y_1 + 0.3y_2 + 0.15y_3 + 0.1y_4 \\ \textbf{s.t.} \qquad & y_1 + y_2 + y_3 \geq 1.5 \\ & y_1 - 0.2y_2 \geq 0 \\ & y_2 \geq 1.2 \\ & y_2 + y_4 \geq 1.4 \\ & y_1, y_2, y_3, y_4 \geq 0 \end{aligned} \tag{9}$$

(d) i. Based on the slack form,let $(x_1, x_2, x_3, x_4, s_1, s_2, s_3, s_4) = (0, 0, 0, 0, 0.3, 0.3, 0.15, 0.1)$

$$\begin{aligned} \textit{maximize} \quad & 1.5x_2 + 1.2x_3 + 1.4x_4 \\ \textbf{s.t.} \quad & 0.3 - x_1 - x_2 = s_1 \\ & 0.3 + 0.2x_1 - x_2 - x_3 - x_4 = s_2 \\ & 0.15 - x_2 = s_3 \\ & 0.1 - x_4 = s_4 \\ & x_1, x_2, x_3, x_4, s_1, s_2, s_3, s_4 \geq 0 \end{aligned} \tag{10}$$

ii. Choose x_4,the limitation is 0.1, so we modify $S = 1.5s_3 + 1.2x_3 + 1.4(0.1 - s_4)$

iii. $(x_1, x_2, x_3, x_4, s_1, s_2, s_3, s_4) = (0, 0, 0, 0.1, 0.3, 0.2, 0.15, 0)$ and change the equation

$$\begin{aligned} \textit{maximize} \quad & 1.5x_2 + 1.2x_3 + 1.4(0.1 - s_4) \\ \textbf{s.t.} \quad & 0.3 - x_1 - x_2 = s_1 \\ & 0.2 + 0.2x_1 - x_2 - x_3 + s_4 = s_2 \\ & 0.15 - x_2 = s_3 \\ & 0.1 - s_4 = x_4 \\ & x_1, x_2, x_3, x_4, s_1, s_2, s_3, s_4 \geq 0 \end{aligned} \tag{11}$$

iv. Choose x_2, the limitation is 0.15, so we modify $S = 1.5(0.15 - s_3) + 1.2x_3 + 1.4(0.1 - s_4)$

v. $(x_1, x_2, x_3, x_4, s_1, s_2, s_3, s_4) = (0, 0.15, 0, 0.1, 0.15, 0.05, 0, 0)$ and change the equation

$$\begin{aligned} \textit{maximize} \quad & 1.5(0.15 - s_3) + 1.2x_3 + 1.4(0.1 - s_4) \\ \textbf{s.t.} \quad & 0.15 + s_3 - x_1 = s_1 \\ & 0.05 + 0.2x_1 - x_3 + s_4 + s_3 = s_2 \\ & 0.15 - s_3 = x_2 \\ & 0.1 - s_4 = x_4 \\ & x_1, x_2, x_3, x_4, s_1, s_2, s_3, s_4 \geq 0 \end{aligned} \tag{12}$$

vi. Choose x_3, the limitation is 0.05, so we modify $S = 1.5(0.15 - s_3) + 1.2(0.05 - s_2 + 0.2x_1 + s_4 + s_3) + 1.4(0.1 - s_4) = 0.425 + 0.24x_1 - 1.2s_2 - 0.3s_3 - 0.2s_4$

vii. $(x_1, x_2, x_3, x_4, s_1, s_2, s_3, s_4) = (0, 0.15, 0.05, 0.1, 0.15, 0, 0, 0)$ and change the equation

$$\begin{aligned} \textit{maximize} \quad & 0.425 + 0.24x_1 - 1.2s_2 - 0.3s_3 - 0.2s_4 \\ \textbf{s.t.} \quad & 0.15 + s_3 - x_1 = s_1 \\ & 0.05 + 0.2x_1 - s_3 + s_4 + s_3 = x_3 \\ & 0.15 - s_3 = x_2 \\ & 0.1 - s_4 = x_4 \\ & x_1, x_2, x_3, x_4, s_1, s_2, s_3, s_4 \geq 0 \end{aligned} \tag{13}$$

viii. Choose x_1, the limitation is 0.15, so we modify $S = 0.461 - 0.24s_1 - 0.06s_3 - 1.2s_2 - 0.2s_4$

ix. $(x_1, x_2, x_3, x_4, s_1, s_2, s_3, s_4) = (0.15, 0.15, 0.08, 0.1, 0, 0, 0, 0)$

x. Here there are no positive valuable in target. So the total sum is 0.461.

$\square$

	PROD 1	**PROD 2**	**PROD 3**	**PROD 4**	**PROD 5**	**PROD 6**	**PROD 7**
Contribution to profit	10	6	8	4	11	9	3
Grinding	0.5	0.7	-	-	0.3	0.2	0.5
Vertical drilling	0.1	0.2	-	0.3	-	0.6	-
Horizontal drilling	0.2	-	0.8	-	-	-	0.6
Boring	0.05	0.03	-	0.07	0.1	-	0.08
Planing	-	-	0.01	-	0.05	-	0.05

4. *Factory Production.* An engineering factory makes seven products (PROD 1 to PROD 7) on the following machines: four grinders, two vertical drills, three horizontal drills, one borer and one planer. Each product yields a certain contribution to profit (in £/unit). These quantities (in £/unit) together with the unit production times (hours) required on each process are given below. A dash indicates that a product does not require a process.

   There are marketing limitations on each product in each month, given in the following table:

	PROD 1	**PROD 2**	**PROD 3**	**PROD 4**	**PROD 5**	**PROD 6**	**PROD 7**
January	500	1000	300	300	800	200	100
February	600	500	200	0	400	300	150
March	300	600	0	0	500	400	100
April	200	300	400	500	200	0	100
May	0	100	500	100	1000	300	0
June	500	500	100	300	1100	500	60

   It is possible to store up to 100 of each product at a time at a cost of £0.5 per unit per month (charged at the end of each month according to the amount held at that time). There are no stocks at present, but it is desired to have a stock of exactly 50 of each type of product at the end of June. The factory works six days a week with two shifts of 8h each day. It may be assumed that each month consists of only 24 working days. Each machine must be down for maintenance in one month of the six. No sequencing problems need to be considered.

   When and what should the factory make in order to maximize the total net profit?

   (a) Use *CPLEX Optimization Studio* to solve this problem. Describe your model in *Optimization Programming Language* (OPL). Remember to use a separate data file (.dat) rather than embedding the data into the model file (.mod).

   (b) Solve your model and give the following results.

      i. For each machine:

         A. the month for maintenance.

      ii. For each product:

         A. The amount to make in each month.

         B. The amount to sell in each month.

         C. The amount to hold at the end of each month.

      iii. The total selling profit.

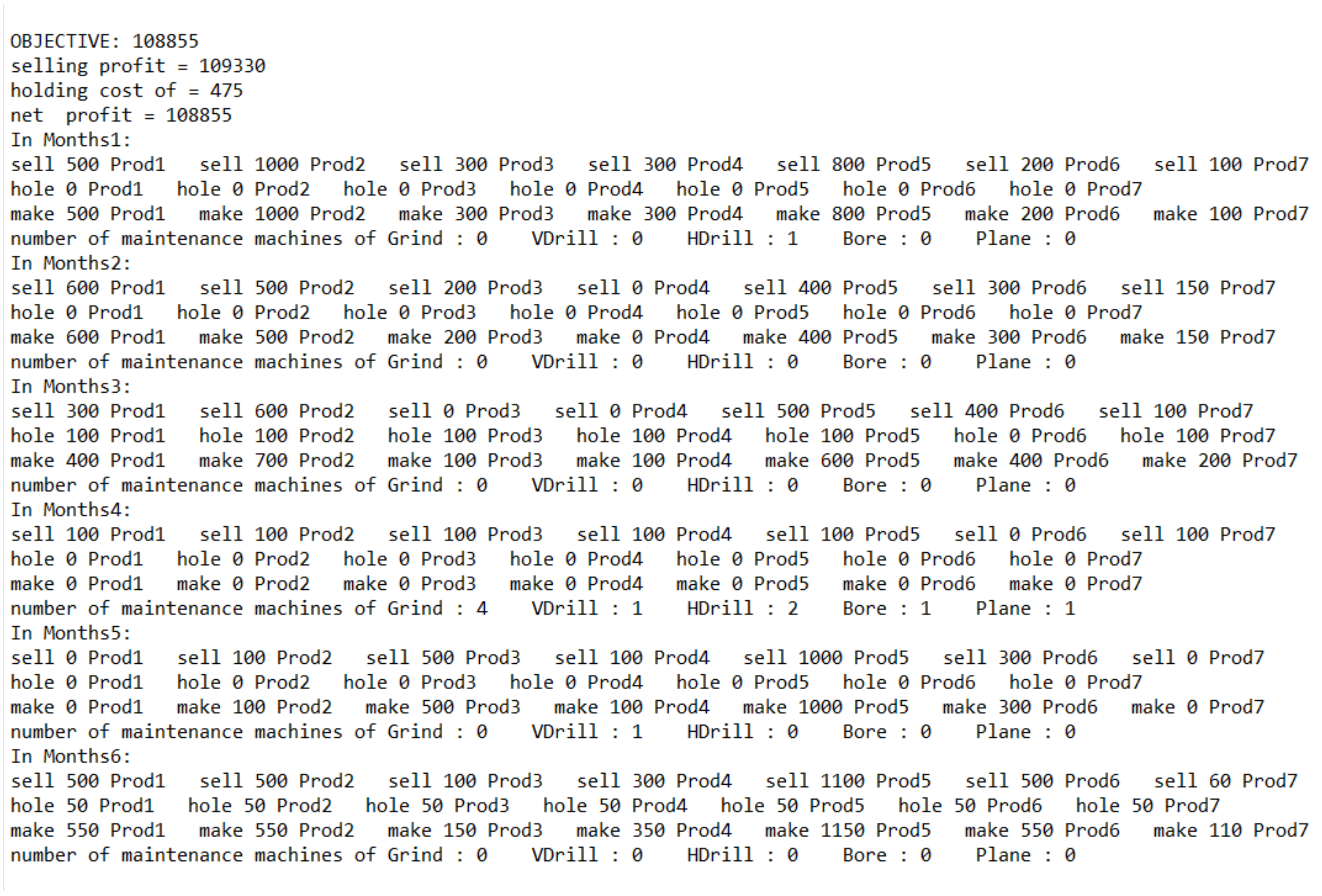
      iv. The total holding cost.

      v. The total net profit (selling profit minus holding cost).

Solution. (a) Please refer to appendix 3.

(b) Please refer to figure **??**

□



```
OBJECTIVE: 108855
selling profit = 109330
holding cost of = 475
net  profit = 108855
In Months1:
sell 500 Prod1   sell 1000 Prod2   sell 300 Prod3   sell 300 Prod4   sell 800 Prod5   sell 200 Prod6   sell 100 Prod7
hole 0 Prod1   hole 0 Prod2   hole 0 Prod3   hole 0 Prod4   hole 0 Prod5   hole 0 Prod6   hole 0 Prod7
make 500 Prod1   make 1000 Prod2   make 300 Prod3   make 300 Prod4   make 800 Prod5   make 200 Prod6   make 100 Prod7
number of maintenance machines of Grind : 0   VDrill : 0   HDrill : 1   Bore : 0   Plane : 0
In Months2:
sell 600 Prod1   sell 500 Prod2   sell 200 Prod3   sell 0 Prod4   sell 400 Prod5   sell 300 Prod6   sell 150 Prod7
hole 0 Prod1   hole 0 Prod2   hole 0 Prod3   hole 0 Prod4   hole 0 Prod5   hole 0 Prod6   hole 0 Prod7
make 600 Prod1   make 500 Prod2   make 200 Prod3   make 0 Prod4   make 400 Prod5   make 300 Prod6   make 150 Prod7
number of maintenance machines of Grind : 0   VDrill : 0   HDrill : 0   Bore : 0   Plane : 0
In Months3:
sell 300 Prod1   sell 600 Prod2   sell 0 Prod3   sell 0 Prod4   sell 500 Prod5   sell 400 Prod6   sell 100 Prod7
hole 100 Prod1   hole 100 Prod2   hole 100 Prod3   hole 100 Prod4   hole 100 Prod5   hole 0 Prod6   hole 100 Prod7
make 400 Prod1   make 700 Prod2   make 100 Prod3   make 100 Prod4   make 600 Prod5   make 400 Prod6   make 200 Prod7
number of maintenance machines of Grind : 0   VDrill : 0   HDrill : 0   Bore : 0   Plane : 0
In Months4:
sell 100 Prod1   sell 100 Prod2   sell 100 Prod3   sell 100 Prod4   sell 100 Prod5   sell 0 Prod6   sell 100 Prod7
hole 0 Prod1   hole 0 Prod2   hole 0 Prod3   hole 0 Prod4   hole 0 Prod5   hole 0 Prod6   hole 0 Prod7
make 0 Prod1   make 0 Prod2   make 0 Prod3   make 0 Prod4   make 0 Prod5   make 0 Prod6   make 0 Prod7
number of maintenance machines of Grind : 4   VDrill : 1   HDrill : 2   Bore : 1   Plane : 1
In Months5:
sell 0 Prod1   sell 100 Prod2   sell 500 Prod3   sell 100 Prod4   sell 1000 Prod5   sell 300 Prod6   sell 0 Prod7
hole 0 Prod1   hole 0 Prod2   hole 0 Prod3   hole 0 Prod4   hole 0 Prod5   hole 0 Prod6   hole 0 Prod7
make 0 Prod1   make 100 Prod2   make 500 Prod3   make 100 Prod4   make 1000 Prod5   make 300 Prod6   make 0 Prod7
number of maintenance machines of Grind : 0   VDrill : 1   HDrill : 0   Bore : 0   Plane : 0
In Months6:
sell 500 Prod1   sell 500 Prod2   sell 100 Prod3   sell 300 Prod4   sell 1100 Prod5   sell 500 Prod6   sell 60 Prod7
hole 50 Prod1   hole 50 Prod2   hole 50 Prod3   hole 50 Prod4   hole 50 Prod5   hole 50 Prod6   hole 50 Prod7
make 550 Prod1   make 550 Prod2   make 150 Prod3   make 350 Prod4   make 1150 Prod5   make 550 Prod6   make 110 Prod7
number of maintenance machines of Grind : 0   VDrill : 0   HDrill : 0   Bore : 0   Plane : 0
```

Figure 2: Factory Production linear solution

Appendix 1

A. FactoryPlanning.dat

```
NbMonths = 6;

Prod = {Prod1, Prod2, Prod3, Prod4, Prod5, Prod6, Prod7};
Process = {Grind, VDrill, HDrill, Bore, Plane};

// profitProd[j] is profit per unit for product j
ProfitProd = [10 6 8 4 11 9 3];

// processProd[i][j] gives hours of process i required by product j
ProcessProd = [[0.5 0.7 0.0 0.0 0.3 0.2 0.5 ]
[0.1 0.2 0.0 0.3 0.0 0.6 0.0 ]
[0.2 0.0 0.8 0.0 0.0 0.0 0.6 ]
[0.05 0.03 0.0 0.07 0.1 0.0 0.08]
[0.0 0.0 0.01 0.0 0.05 0.0 0.05]];

// marketProd[i][j] gives marketing limitation on product j for month i
MarketProd = [[500 1000 300 300 800 200 100]
[600 500 200 0 400 300 150]
[300 600 0 0 500 400 100]
[200 300 400 500 200 0 100]
[0 100 500 100 1000 300 0 ]
[500 500 100 300 1100 500 60 ]];

CostHold = 0.5;
StartHold = 0;
```

```
    EndHold = 50;
    MaxHold = 100;

    // process capacity
    HoursMonth = 384; // 2 eight hour shifts per day, 24 working days per month;

    // number of each type of machine
    NumProcess = [4 2 3 1 1];

    // how many machines must be down over 6 month period
    NumDown = [4 2 3 1 1];
```

Appendix 3

```
{string} Prod = ...;
{string} Process = ...;

int NbMonths = ...;
range Months=1..NbMonths;
float ProfitProd[Prod]=...;
float ProcessProd[Process][Prod]= ...;
int MarketProd[Months][Prod] = ...;

float CostHold = ...;
int StartHold = ...;
int EndHold = ...;
int MaxHold = ...;

int HoursMonth = ...;

int NumProcess[Process] = ...;
int NumDown[Process] = ...;
dvar int+ product[Months][Prod];
dvar int+ produce[Months][Prod];
dvar int+ sell[Months][Prod];
dvar int+ remain[Months][Prod];
dvar int+ down[Months][Process];
dvar float+ Profit;
dvar float+ Cost;
maximize Profit-Cost;
subject to{
        /*relationship constrain*/
        forall(i in Prod){
                forall(j in Months){
                        remain[j][i]==product[j][i]-sell[j][i];
                 }
        }
        forall(i in Months){
                forall(j in Prod){
                        sell[i][j]<=MarketProd[i][j];
                }
        }
        forall(i in Months){
                forall(j in Prod){
                        if(i==1){
                                product[i][j]==produce[i][j]+StartHold;
                        }
                        else{
                                product[i][j]==produce[i][j]+remain[i-1][j];
                        }
                }
        }

        /*machine down constrain*/
        forall(j in Process){
                sum(i in Months)down[i][j] == NumDown[j];
        }

        /*produce machine constarin */
        forall(i in Months){
                forall(j in Process){
                        sum(k in Prod)(produce[i][k]*ProcessProd[j][k])<=HoursMonth
                        *(NumProcess[j]-down[i][j]);
                }
        }
        /*store constrain*/
        forall(i in Months){
                forall(j in Prod){
                        if(i==NbMonths){
                                remain[i][j]==EndHold;
                        }
                        else{
                                remain[i][j]<=MaxHold;
                        }
                }
        }
        Profit == sum(i in Months) sum(j in Prod)sell[i][j]*ProfitProd[j];
        Cost == sum(i in 1..NbMonths)sum(j in Prod)remain[i][j]*CostHold;
}
execute {
        if (cplex.getCplexStatus() == 1) {
        writeln('selling profit = ', Profit)
        writeln('holding cost of = ', Cost)
        writeln('net  profit = ', Profit-Cost)
                for(i in Months){
                        write('In Months',i)
                        write(':\n')
                        for(j in Prod)  {
                                write("sell  ",sell[i][j])
                                write(" ",j)
                                write("    ")
                        }
                        write("\n")
                        for(j in Prod)  {
                                write("hole  ",remain[i][j])
                                write(" ",j)
                                write("    ")
                }
                write("\n")
                for(j in Prod)  {
                    write("make ",produce[i][j])
                    write(" ",j)
                    write("   ")
                }
                write("\n")
                write("number of maintenance machines of ")
                for(j in Process)       {
                    write(j)
                    write(" : ",down[i][j])
                    write("     ")
                }
                write("\n")
            }
        }
}
```