Lab06-Linear Programming

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

∗ If there is any problem, please contact TA Haolin Zhou.

∗ Name:Xin Xu Student ID:519021910726 Email: xuxin20010203@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).

   (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. (a) Let $T(m, n)$ be the max running time of algorithm on strings of length m and n. We know from the algorithm that $T(m, n) \leqslant cmn + T(q, n/2) + T(m - q, n/2)$. We assume that $T(m, n) = O(mn)$ for any integers $m, n \leqslant 2$ with the conditions $T(m, 2) \leqslant cm; T(2, n) \leqslant cn$.
$T(m, n) \leqslant cmn + T(q, n/2) + T(m - q, n/2)$
$\leqslant 2cqn/2 + 2c(m - q)n/2 + cmn$
$= cqn + cmn - cqn + cmn$
$= 2cmn$
So, the time complexity is $O(mn)$.

(b) The answer is 385. More details are in .cpp file.

□

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.

   Solution. We construct two matrices A and W, and

$$a_{ij} = \begin{cases} 1 \text{ if the route includes a path from } v_i \text{ to } v_j \\ 0, \text{ else} \end{cases}$$

$$w_{ij} = \begin{cases} w_{ij} \text{ if the route includes a path from } v_i \text{ to } v_j \text{ with the weight of } w_{ij} \\ 0, \text{ else} \end{cases}$$

   The ILP form is:
   minimize the sum of elements on the diagonal of $A \times W$.
   The sum of each row and the sum of each column are both 1. □

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) Let x_1 be the investment to project 2, x_2 be the investment to project 1 in the first year, x_3 be the investment to project 3 in the second year, and x_4 be the investment to project 1 in the second year.
   The linear programming is:
   $\max 1.5x_1 + 1.4x_3 + 1.2x_4$
   $x_1 + x_2 = 0.3, x_3 + x_4 = 1.2x_2, x_1 \leqslant 0.15, x_3 \leqslant 0.1, x_1 \geqslant 0, x_2 \geqslant 0, x_3 \geqslant 0, x_4 \geqslant 0.$

   (b) standard form:
   $\max 1.5x_1 + 1.4x_3 + 1.2x_4$
   $1.2x_1 + x_3 + x_4 \leqslant 0.36$
   $-1.2x_1 - x_3 - x_4 \leqslant -0.36$
   $x_1 \leqslant 0.15$
   $x_3 \leqslant 0.1$
   $x_1 \geqslant 0, x_3 \geqslant 0, x_4 \geqslant 0$
   slack form:
   $\max 1.5x_1 + 1.4x_3 + 1.2x_4$
   $1.2x_1 + x_3 + x_4 = 0.36$
   $x_1 + x_5 = 0.15$
   $x_3 + x_6 = 0.1$
   $x_1 \geqslant 0, x_3 \geqslant 0, x_4 \geqslant 0, x_5 \geqslant 0, x_6 \geqslant 0$

   (c) dual form:
   $\min 0.36y_1 + 0.15y_2 + 0.1y_3$
   $1.2y_1 + y_2 \geqslant 1.5$
   $y_1 + y_3 \geqslant 1.4$
   $y_1 \geqslant 1.2$
   $y_1, y_2, y_3 \geqslant 0$

(d) simplex method:
slack form:
max $1.5x_1 + 1.4x_3 + 1.2x_4$
$1.2x_1 + x_3 + x_4 = 0.36$
$x_1 + x_5 = 0.15$
$x_3 + x_6 = 0.1$
$x_1 \geqslant 0, x_3 \geqslant 0, x_4 \geqslant 0, x_5 \geqslant 0, x_6 \geqslant 0$
The basic solution is $x = (0, 0, 0.36, 0.15, 0.1)$
Update x_1 to most: $x = (0.15, 0, 0.18, 0, 0.1)$
Update the function: max $0.225 - 1.5x_5 + 1.4x_3 + 1.2x_4$
Update x_3 to most: $x = (0.15, 0.1, 0.08, 0, 0)$
Update the function: max $0.365 - 1.5x_5 - 1.4x_6 + 1.2x_4$
The answer of this solution is $0.365 + 1.2 * 0.08 = 0.461$. Since one of the solutions to dual form is: $y = (1.2, 0.06, 0.2)$, min$= 0.36 * 1.2 + 0.15 * 0.06 + 0.1 * 0.2 = 0.461$, the solution is $x = (0.15, 0.1, 0.08, 0, 0)$, and the maximum of capital and profit is 0.461.

□

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.

	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

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).

Remark: You can choose to use the attached .dat file or write it yourself.

Solution. (a) .dat file and .mod file are in the folder.

(b) The answer of first two question is:

```
Sell = [[500
             600 300 100 0 500]
             [1000 500 600 100 100 500]
             [300 200 0 100 500 100]
             [300 0 0 100 100 300]
             [800 400 500 100 1000 1100]
             [200 300 400 0 300 500]
             [100 150 100 100 0 60]];
Hold = [[0 0 0 100 0 0 50]
             [0 0 0 100 0 0 50]
             [0 0 0 100 0 0 50]
             [0 0 0 100 0 0 50]
             [0 0 0 100 0 0 50]
             [0 0 0 0 0 0 50]
             [0 0 0 100 0 0 50]];
Make = [[500 600 400 0 0 550]
             [1000 500 700 0 100 550]
             [300 200 100 0 500 150]
             [300 0 100 0 100 350]
             [800 400 600 0 1000 1150]
             [200 300 400 0 300 550]
             [100 150 200 0 0 110]];
Down = [[0 1 1 2 0 0]
             [0 0 0 1 1 0]
             [1 1 1 0 0 0]
             [0 0 0 1 0 0]
             [0 0 0 1 0 0]];
```

Figure 1: machine and product

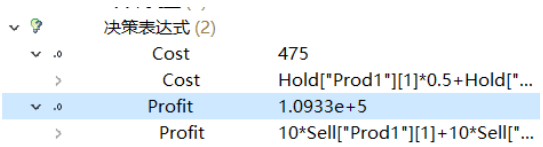

Figure 2: cost and profit

□

Appendix

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];
```

Remark: You need to include your .cpp, .mod, .dat, .pdf and .tex files in your uploaded .zip file.