Lab06-Linear Programming

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

∗ If there is any problem, please contact TA Haolin Zhou.
∗ Name: WendiChen Student ID: 519021910071 Email: chenwendi-andy@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) Please refer to *Code-SequenceAlignment.cpp*.
Let $T(m, n)$ be the max running time of algorithm on strings of length m and n. Then we will prove that $T(m, n) = O(mn)$. Actually, we have the recurrence, where finding index q costs $O(mn)$

$$T(m, n) \leq T(q, \frac{n}{2}) + T(m - q, \frac{n}{2}) + O(mn)$$

According to the algorithm, we can choose a constant c so that

$$\begin{aligned} T(m, 2) &\leq cm \\ T(2, n) &\leq cn \\ T(m, n) &\leq cmn + T(q, \frac{n}{2}) + T(m - q, \frac{n}{2}) \end{aligned}$$

Then we prove the statement by induction on n.
For basis, when $n = 2$, we have $T(m, 2) \leq cm < 2cmn$.
Then for every $n = 2^k, k > 1$. We assume $T(m, \frac{n}{2}) \leq cmn$ and get

$$\begin{aligned} T(m, n) &\leq T(q, \frac{n}{2}) + T(m - q, \frac{n}{2}) + cmn \\ &\leq 2cq\frac{n}{2} + 2c(m - q)\frac{n}{2} + cmn \\ &= cqn + cmn - cqn + cmn \\ &= 2cmn \end{aligned}$$

Thus, the time complexity is $O(mn)$.

(b) The running result of the program is shown below (Figure 1).

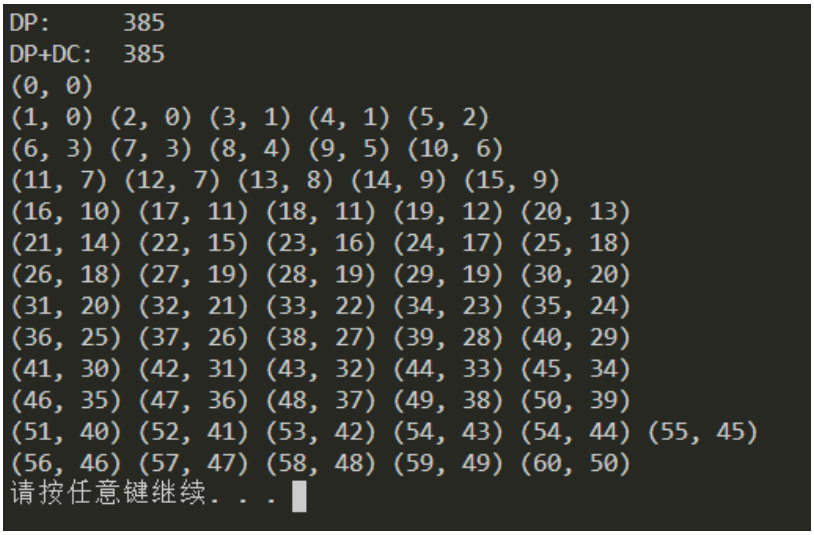

Figure 1: Result of the Hirschberg Algorithm

□

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

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

   Then $\sum_{e \in E} x_e W_e$ denotes the length of the route, and thus we can formulate the following ILP:

$$\begin{aligned} min \quad & \sum_{e \in E} x_e W_e \\ s.t. \quad & \sum_{e \in N(u)} x_e = 2 \quad \forall u \in V \\ & \sum_{u \in V'} \sum_{e \in N(u)} x_e < 2|V'| \quad \forall V' \subsetneq V \\ & x_e \in \{0, 1\} \quad \forall e \in E \end{aligned}$$

   where $N(u)$ denotes the set of edges that are connected to vertex u. □

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) During the whole process, we need to make decisions at two points in time–at the beginning of 2021 and at the beginning of 2022. At the beginning of 2021, we can divide the funds into two parts. One part for project 1 and the other part for project 2. Thus, we have $x_1 + x_2 = 0.3$ and $x_2 \leq 0.15$. Thus, at the beginning of 2022, all money we have make use of is $1.2x_1$. Similarly, we divide the money into two parts and have $x_3 + x_4 = 1.2x_1$ and $x_4 \leq 0.1$. So the overall sum of capital and profit at the end of 2022 is $1.5x_2 + 1.2x_3 + 1.4x_4$. Of course, all x_i should be positive. Thus, we have derived a LP:

$$\begin{aligned} max \quad & f(x_2, x_3, x_4) = 1.5x_2 + 1.2x_3 + 1.4x_4 \\ s.t. \quad & x_1 + x_2 = 0.3 \\ & x_2 \leq 0.15 \\ & x_3 + x_4 - 1.2x_1 = 0 \\ & x_4 \leq 0.1 \\ & x_1, x_2, x_3, x_4 \geq 0 \end{aligned}$$

(b) We transform the LP into its standard form first.

$$\begin{aligned} max \quad & f(x_2, x_3, x_4) = 1.5x_2 + 1.2x_3 + 1.4x_4 \\ s.t. \quad & x_1 + x_2 \leq 0.3 \\ & -x_1 - x_2 \leq -0.3 \\ & x_2 \leq 0.15 \\ & x_3 + x_4 - 1.2x_1 \leq 0 \\ & -x_3 - x_4 + 1.2x_1 \leq 0 \\ & x_4 \leq 0.1 \\ & x_1, x_2, x_3, x_4 \geq 0 \end{aligned}$$

Then we transform it into a slack form.

$$\begin{aligned} max \quad & f(x_2, x_3, x_4) = 1.5x_2 + 1.2x_3 + 1.4x_4 \\ s.t. \quad & x_1 + x_2 = 0.3 \\ & x_2 + x_5 = 0.15 \\ & x_3 + x_4 - 1.2x_1 = 0 \\ & x_4 + x_6 = 0.1 \\ & x_1, x_2, x_3, x_4, x_5, x_6 \geq 0 \end{aligned}$$

(c) We define six multipliers y_1, y_2, y_3, y_4, y_5 and y_6. And after that we get

$$\begin{aligned} & (y_1 - y_2 - 1.2y_4 + 1.2y_5)x_1 + (y_1 - y_2 + y_3)x_2 + (y_4 - y_5)x_3 + (y_4 - y_5 + y_6)x_4 \\ & \leq 0.3y_1 - 0.3y_2 + 0.15y_3 + 0.1y_6 \end{aligned}$$

Then we derive the dual form.

$$\begin{aligned} min \quad & f(y_1, y_2, y_3, y_4) = 0.3y_1 - 0.3y_2 + 0.15y_3 + 0.1y_6 \\ s.t. \quad & y_1 - y_2 - 1.2y_4 + 1.2y_5 \geq 0 \\ & y_1 - y_2 + y_3 \geq 1.5 \\ & y_4 - y_5 \geq 1.2 \\ & y_4 - y_5 + y_6 \geq 1.4 \\ & y_1, y_2, y_3, y_4, y_5, y_6 \geq 0 \end{aligned}$$

(d) We set x_1, x_5, x_6, x_7 as basic variables. Thus the slack form can be written as

$$\begin{aligned} max \quad & f(x_2, x_3, x_4) = 1.5x_2 + 1.2x_3 + 1.4x_4 \\ s.t. \quad & 0.3 - x_2 = x_1 \\ & 0.15 - x_2 = x_5 \\ & 0.36 - 1.2x_2 - x_4 - x_3 = x_7 \\ & 0.1 - x_4 = x_6 \\ & x_1, x_2, x_3, x_4, x_5, x_6, x_7 \geq 0 \end{aligned}$$

The basic solution is $\overline{x} = (0.3, 0, 0, 0, 0.15, 0.1, 0.36)$. Next, we choose the nonbasic variable x_2 and $0.15 - x_2 = x_5$ is the tightest constraint for x_2, so we transform it into $0.15 - x_5 = x_2$. The nonbasic variables become x_3, x_4, x_5.

$$\begin{aligned} max \quad & f(x_5, x_3, x_4) = 1.5(0.15 - x_5) + 1.2x_3 + 1.4x_4 \\ s.t. \quad & 0.15 + x_5 = x_1 \\ & 0.15 - x_5 = x_2 \\ & 0.18 + 1.2x_5 - x_4 - x_3 = x_7 \\ & 0.1 - x_4 = x_6 \\ & x_1, x_2, x_3, x_4, x_5, x_6, x_7 \geq 0 \end{aligned}$$

And we get $\overline{x} = (0.15, 0.15, 0, 0, 0, 0.1, 0.18)$. Similarly, we choose the nonbasic variable x_4 and $0.1 - x_4 = x_6$ is the tightest constraint. And the nonbasic variable becomes

x_3, x_5, x_6.

$$\begin{aligned} max \quad & f(x_5, x_3, x_6) = 1.5(0.15 - x_5) + 1.2x_3 + 1.4(0.1 - x_6) \\ s.t. \quad & 0.15 + x_5 = x_1 \\ & 0.15 - x_5 = x_2 \\ 0.08 + 1.2x_5 + x_6 - x_3 &= x_7 \\ & 0.1 - x_6 = x_4 \\ x_1, x_2, x_3, x_4, x_5, x_6, x_7 &\geq 0 \end{aligned}$$

And we get $\overline{x} = (0.15, 0.15, 0, 0.1, 0, 0, 0.08)$. At last, we have $0.08 + 1.2x_5 + x_6 - x_7 = x_3$. So, we have to max $f(x_5, x_6, x_7) = 1.5(0.15 - x_5) + 1.2(0.08 + 1.2x_5 + x_6 - x_7) + 1.4(0.1 - x_6) = 0.461 - 0.06x_5 - 0.2x_6 - 1.2x_7$. Now all coefficients are negative. The optimal solution is $\overline{x} = (0.15, 0.15, 0.08, 0.1, 0, 0, 0)$. The overall sum of capital and profit is 0.461 million yuan.

□

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) Please refer to *model.mod* and *FactoryPlanning.dat*.

(b) We assume that every unit of product for making, selling or storing is integer.

i. A.

	Grinders	**Vertical Drills**	**Horizontal Drills**	**Borer**	**Planner**
January	0	0	1	0	0
February	0	1	0	0	0
March	0	0	0	0	0
April	4	1	2	1	1
May	0	0	0	0	0
June	0	0	0	0	0

ii. A.

	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	400	700	100	100	600	400	200
April	0	0	0	0	0	0	0
May	0	100	500	100	1000	300	0
June	550	550	150	350	1150	550	110

B.

	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	100	100	100	100	100	0	100
May	0	100	500	100	1000	300	0
June	500	500	100	300	1100	500	60

C.

	PROD 1	**PROD 2**	**PROD 3**	**PROD 4**	**PROD 5**	**PROD 6**	**PROD 7**
January	0	0	0	0	0	0	0
February	0	0	0	0	0	0	0
March	100	100	100	100	100	0	100
April	0	0	0	0	0	0	0
May	0	0	0	0	0	0	0
June	50	50	50	50	50	50	50

iii. The total selling profit is 109330£.

iv. The total holding cost is 475£.

v. The total net profit is 108855£.

□

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.