Lab06-Linear Programming

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

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

∗ Name: Renyang Guan   Student ID: 519021911058   Email: guanrenyang@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.

$$
\begin{aligned}
X[1..60] = &\ CMQHZZRIQOQJOCFPRWOUXXCEMYSWUJ \\
&\ TAQBKAJIETSJPWUPMZLNLOMOZNLTLQ
\end{aligned}
$$

$$
\begin{aligned}
Y[1..50] = &\ SUYLVMUSDROFBXUDCOHAATBKN \\
&\ AAENXEVWNLMYUQRPEOCJOCIMZ
\end{aligned}
$$

Solution.

(a) The algorithm is implemented in file Code-SequenceAlignment.cpp.
Time complexity analysis:
The time complexity is $O(mn)$.

Proof. We could prove the conclusion by induction. Let $T(m, n)$ denotes the worst case time complexity. Suppose that $T(m, n) \leq kmn$, where k is any positive integer.
Basis step: When $m \leq 2$ and $n \leq 2$, the conclusion is obvious
Induction hypothesis: Suppose that for m' and n', where $m'n' < mn$ the hypothesis is true.
Proof of the induction step: We could generate such formula:

$$
\begin{aligned}
T(m, n) &\leq cmn + T(q, n/2) + T(m - q, n/2) \\
&\leq cmn + kqn/2 + k(m - q)n/2 \\
&= (c + k/2)mn
\end{aligned}
\tag{1}
$$

If $k = 2c$ the conclusion is true. □

(b) The edit distance of the two strings is **385**.

□

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_{uv} is a 0-1 indicator of the directed edge connecting vertex u and v, such that:

$$x_{uv} = \begin{cases} 1 & the\ salesman\ does\ travel\ from\ u\ to\ v\ directly \\ 0 & the\ salesman\ does\ not\ travel\ from\ u\ to\ v\ directly \end{cases}$$

Then w_{uv} denotes the weight of the directed edge from vertex u to v. Thus we can formulate the following ILP:

$$\begin{aligned} Goal: & \sum_{u,v \in V} w_{uv} x_{uv} \\ s.t. & \sum_{v \in N(u)} x_{uv} = 1 \ \ \forall\ u \in V \\ & \sum_{v \in N(u)} x_{vu} = 1 \ \ \forall\ u \in V \\ & \sum_{u,v \in U} x_{uv} < |U| \ \forall\ U \subset V \end{aligned}$$

where $N(u)$ denotes the set of neighbors of 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) Suppose that a_1 and a_2 denote the investment of *Project 1* in 2021 and 2022 respectively, b denotes the investment of *Project 2*, and c denotes the investment of *Project 3*.

We can formulate the linear programming as follows:

$$\begin{aligned} & \mathit{Goal}\ {:}max\ (120\%a_2 + 150\%b + 140\%c) \\ & s.t.\ a_1 + b = 0.3 \\ & \quad a_2 + c = 120\%a_1 \\ & \quad b \leq 0.15 \\ & \quad c \leq 0.1 \\ & \quad a_1, a_2, b, c \geq 0 \end{aligned}$$

Necessary explanation:
At the beginning of 2021, only *Project 1* and *Project 2* could be invested and all its money must be invested, such that equation $a_1 + b = 0.3$ holds. The money that can be invested at the end of 2022 is the profit of 2021, where the former is denoted as $a_2 + c$ and the later is $120\%a_1$. Thus the equation $a_2 + c = 120\%a_1$ holds. The last two constraints are the requirement of *Project 2* and *Project 3* respectively.

(b) ***Standard form:***

$$\begin{aligned} & \mathit{Goal}\ {:}max\ (120\%a_2 + 150\%b + 140\%c) \\ & s.t.\ a_1 + b \leq 0.3 \\ & \quad -a_1 - b \leq -0.3 \\ & \quad a_2 + c - 120\%a_1 \leq 0 \\ & \quad -a_2 - c + 120\%a_1 \leq 0 \\ & \quad b \leq 0.15 \\ & \quad c \leq 0.1 \\ & \quad a_1, a_2, b, c \geq 0 \end{aligned}$$

Slack form:

$$\begin{aligned} & \mathit{Goal}\ {:}max\ (120\%a_2 + 150\%b + 140\%c) \\ & s.t.\ a_1 + b = 0.3 \\ & \quad a_2 + c = 120\%a_1 \\ & \quad b + x_1 = 0.15 \\ & \quad c + x_2 = 0.1 \\ & \quad a_1, a_2, b, c \geq 0 \end{aligned}$$

slack form可以从原规划方程中得出，不一定能够要从standard form转化过来

x_1 and x_2 are slack variables.

(c) ***Dual form:***

$$\begin{aligned} & \mathit{Goal}\ {:}min\ (0.3y_1 - 0.3y_2 + 0.15y_5 + 0.1y_6) \\ & s.t.\ y_1 - y_2 - 120\%y_3 + 120\%y_4 \geq 0 \\ & \quad y_3 - y_4 \geq 120\% \\ & \quad y_1 - y_2 + y_5 \geq 150\% \\ & \quad y_3 - y_4 + y_6 \geq 140\% \\ & \quad y_1, y_2, y_3, y_4, y_5, y_6 \geq 0 \end{aligned}$$

(d) **Step.1** Converting the LP into slack form which could be solved by simplex method. To do this, we must combine the two equations firstly:

$$\begin{cases} a_1 + b = 0.3 \\ a_2 + c = 1.2a_1 \end{cases} \Rightarrow a_2 + 1.2b + c = 0.36 \tag{2}$$

and then bring the result int our goal, such that we have the slack form LP:

$$\begin{aligned} Goal\ &:max\ (0.06b + 0.2c + 0.432) \\ s.t.\ &0.15 - b = x_1 \\ &0.1 - c = x_2 \\ &b, c, x_1, x_2 \geq 0 \end{aligned} \tag{3}$$

Step.2 Obtaining basic solution. In the question, the basic solution is

$$\bar{X} = (\bar{b}, \bar{c}, \bar{x_1}, \bar{x_2}) = (0, 0, 0.15, 0.1)$$

Step.3 Selecting nonbasic variable b.Since the *tighter* constraint for b is $0.15 - b = x_1$, we change it into $0.15 - x_1 = b$ **Step.4** Pivoting
Currently the LP is transformed as:

$$\begin{aligned} Goal\ &:max\ (-0.06x_1 + 0.2c + 0.441) \\ s.t.\ &0.15 - x_1 = b \\ &0.1 - c = x_2 \\ &b, c, x_1, x_2 \geq 0 \end{aligned}$$

Step.5 Repeat Step2 to Step4 The final form of primitive LP is

$$\begin{aligned} Goal\ &:max\ (-0.06x_1 - 0.2x_2 + 0.461) \\ s.t.\ &0.15 - x_1 = b \\ &0.1 - x_2 = c \\ &b, c, x_1, x_2 \geq 0 \end{aligned}$$

Final solution:

$$\bar{x} = (0.15, 0.1, 0, 0)$$

and the goal 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:

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

	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

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.

Model: The linear programming model is described in file *FactoryPlanning.mod*.
Results: The results are shown in , respectively.

i. The result is shown in Fig. 1. The number in the table illustrate the number of machines which are down for maintenance in each particular month.

ii. A. is shown in Fig. 2.

    B. is shown in Fig. 3.

    C. is shown in Fig. 4.

iii. The total selling profit is **109330**.

iv. The total holding cost is **475**.

v. The total net profit is **108855**.

Months (大小 6)	Process (大小 5)				
	"Grind"	"VDrill"	"HDrill"	"Bore"	"Plane"
1	0	0	1	0	0
2	1	0	1	0	0
3	1	0	1	0	0
4	2	1	0	1	1
5	0	1	0	0	0
6	0	0	0	0	0

Figure 1: The month for maintenance

Months (大小 6)	Prod (大小 7)						
	"Prod1"	"Prod2"	"Prod3"	"Prod4"	"Prod5"	"Prod6"	"Prod7"
1	500	1000	300	300	800	200	100
2	600	500	200	0	400	300	150
3	400	700	100	100	600	400	200
4	0	0	0	0	0	0	0
5	0	100	500	100	1000	300	0
6	550	550	150	350	1150	550	110

Figure 2: The amount to make in each month

Months (大小 6)	Prod (大小 7)						
	"Prod1"	"Prod2"	"Prod3"	"Prod4"	"Prod5"	"Prod6"	"Prod7"
1	500	1000	300	300	800	200	100
2	600	500	200	0	400	300	150
3	300	600	0	0	500	400	100
4	100	100	100	100	100	0	100
5	0	100	500	100	1000	300	0
6	500	500	100	300	1100	500	60

Figure 3: The amount to sell in each month

0..NbMonths (大小 7)	Prod (大小 7)						
	"Prod1"	"Prod2"	"Prod3"	"Prod4"	"Prod5"	"Prod6"	"Prod7"
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0
3	100	100	100	100	100	0	100
4	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0
6	50	50	50	50	50	50	50

Figure 4: The amount to hold at the end of each month

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.